SERIES 60 (LEVEL 64)

COMMUNICATIONS PROCESSING FACILITY ADDENDUM A

SUBJECT

Corrections and Additions to the Manual

SPECIAL INSTRUCTIONS

Insert the attached pages into the manual (Rev. 1, dated September 1978) according to the collating instructions on the back of this cover. Changebars indicate new and changed information; asterisks denote deletions.

Note:

Insert this cover behind the manual cover as evidence that the manual is updated with this addendum.

SOFTWARE SUPPORTED

Level 64 GCOS Release 0400

ORDER NUMBER

AQ53-01A

June 1979

24067
1.5679
Printed in U.S.A.

Honeywell

COLLATING INSTRUCTIONS

To update this manual, remove old pages and insert new pages as follows:

Remove	Insert
v through viii	v through viii
1-01, 1-02	1-01, 1-02
1-05, 1-06	1-05, 1-06
5-23 through 5-26	5-23 through 5-26
5-33, 5-34	5-33, 5-34
A-01 through A-04	A-01 through A-04
B-23, B-24	B-23, B-24
B-31, B-32	B-31, B-32
D-03, D-04	D-03, D-04
G-05 through G-12	G-05 through G-12

File No.: 1N13

6/79
AQ53-01A

LEVEL 64 DOCUMENT LIST

Order Number	Title
AQ02	*Series 100 Program Mode Operator Guide*
AQ03	*Series 100 Conversion Guide*
AQ04	*Series 200/2000 Conversion Guide*
AQ05	*System 360/370 Conversion Guide*
AQ09	*System Management Guide*
AQ10	*Job Control Language (JCL) Reference Manual*
AQ11	*Job Control Language (JCL) User Guide*
AQ13	*System Operation Operator Guide*
AQ14	*System Operation Console Messages*
AQ18	*Operator Reference Manual*
AQ20	*Data Management Utilities Manual*
AQ21	*Series 200/2000 Program Mode User Guide*
AQ22	*Series 200/2000 Program Mode Operator Guide*
AQ26	*Series 100 File Translator*
AQ27	*Series 200/2000 File Translator*
AQ28	*Library Maintenance Manual*
AQ40	*System 3 Conversion Guide*
AQ49	*Network Control Terminal Operation Manual*
AQ50	*Terminal Operations Manual*
AQ52	*Program Checkout Facility Manual*
AQ53	*Communications Processing Facility Manual*
AQ55	*TDS/64 Standard Processor Site Manual*
AQ56	*TDS/64 User Guide*
AQ57	*Standard Processor Programmer Reference Manual*
AQ59	*Unit Record Devices User Guide*
AQ63	*COBOL User Guide*
AQ60	*Interactive Operation Facility*
AQ64	*COBOL Language Reference Manual*
AQ65	*FORTRAN Language Reference Manual*
AQ66	*FORTRAN User Guide*
AQ67	*FORTRAN Mathematical Library*
AQ68	*RPG Language Reference Manual*
AQ69	*RPG User Guide*
AQ72	*Series 200/2000 COBOL to Level 64 COBOL Translator*
AQ73	*IBM COBOL Translator*
AQ82	*BFAS User Guide*
AQ83	*HFAS User Guide*
AQ84	*UFAS User Guide*
AQ85	*Sort/Merge Manual*
AQ86	*Catalog Management Manual*
AQ87	*Library Maintenance User Guide*
AQ88	*I-D-S/II User Guide, Volume 1*
AQ89	*I-D-S/II User Guide, Volume 2*
AQ90	*COBOL Reference Card*
AQ92	*Operator's Reference Card*
AQ93	*RPG Reference Card*
AQ94	*FORTRAN Reference Card*

TABLE OF CONTENTS

SECTION I

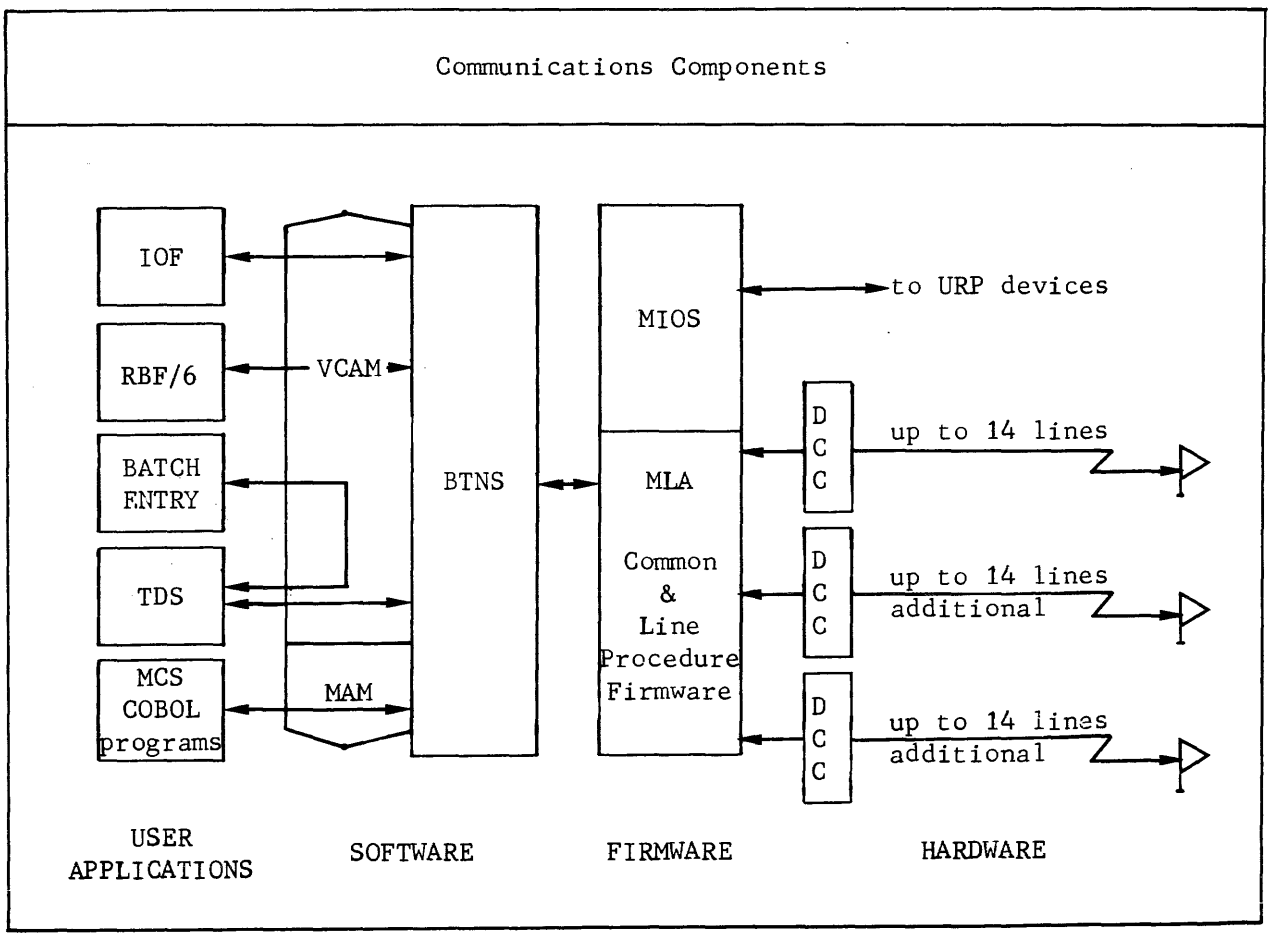
INTRODUCTION TO LEVEL 64 COMMUNICATIONS

The Level 64 GCOS Communications Processing Facility enables the user

- to run MCS COBOL applications that require access thru queues to a communications network
- to run communications subsystems collectively termed VCAM (virtual communications access method) and which comprise IOF (interactive operation facility) and TDS (transaction driven system).

The communications processing facility is described in terms of its components,

- Hardware
- Firmware
- Software
- User applications



HARDWARE

In addition to standard site equipment, the hardware needed to support a communications network comprises,

- Data Communications Controller(s): The DCC is attached to the Unit Record Processor (URP) and provides the interface for the terminals.
  The DCC controls line transmissions in assembling and disassembling characters.
  It can support as many as 14 line simultaneities, each of which is a data path defined as follows,
  - a two-way-alternate (TWA) line uses one simultaneity
  - a two-way-simultaneous line uses two simultaneities.

  The versions of the DCC are,
  - DCC4100 or DCC4901 connectable to the IURP
  - DCC4200 or DCC4902 and DCC4903 can be connected to the additional URP.

- Communications Lines : Communications lines can be,
  - Local, which is a direct connection by cable
  - Switched, which uses a dial-up mechanism to select an available telephone link
  - Leased, which is a private user line with supporting modems.

  The line is used as follows,
  - Point-to-point, connecting one terminal to the central processor
  - Multipoint, connecting several terminals on the same line to the central processor.

  Transmission can be,
  - Synchronous, ranging up to 19200 bauds
  - Asynchronous, ranging from 50 thru 2400 bauds and can be any of 4 pre-selected speeds per DCC.

- Terminals : The terminal is the communications device for data entry and reception.
  It is associated with
  - Terminal type
  - Line procedure
  - Terminal subtype.

  For further details on the network, see Section V.

Terminal Types & Line Procedures			
Terminal	Line Procedure A=Asynchronous S=Synchronous	Terminal	Line Procedure A=Asynchronous S=Synchronous
		TWU/PRU1001 TWU/PRU1003 TWU/PRU1005	TTY
		TWU/PRU1901	VIP(S)
HL62 HL64 HL66	BSC	TTY33 TTY35 TTY37 TTY38	TTY
		VIP765	VIP(A)
		VIP775 VIP785	VIP(S)
		VIP7100 VIP7200	TTY
		VIP7700 VIP7760	VIP(S)
TN300 TN1200	TTY		

BSC = Line procedure 4

TTY = Line procedure 1
VIP = Line procedure 3

Line procedure 1 = TTY

Line procedure 3 = VIP
Line procedure 4 = BSC

Terminal Subtypes		
subtype	abbreviation	meaning
CAS	CASsette	cassette handler
CPU	Central Processor Unit	levels 62/64/66
CRT	Cathode Ray Tube	screen
KB	KeyBoard	keyboard
KCT	Keyboard Cathode-ray Tube	keyboard and screen
KPR	Keyboard with PRinter	keyboard and printer
PRT	PRinTer	printer

FIRMWARE

Communications firmware is the multiline attachment (MLA) which consists of,

- A set of common attachments to distinguish line connections
- An attachment for each of the 3 line procedures to be handled by the unit record processor (URP or IURP).

The MLA functions with MIOS (micro-operating system) for terminal management.

SOFTWARE

Communications software comprises 5 major components,

- BTNS, basic terminal network support
- MAM, message access method
- VCAM, virtual communications access method
- CNC, communications network configurator
- QMAINT, queue maintenance

Basic Terminal Network Support (BTNS)

BTNS is the nucleus of Level 64 communications performing such functions as,

- Managing physical operations of the network, as initializing the lines and transferring data over the lines
- Allocating communications resources, i.e., allocating terminals to application programs, and vice versa, and sharing lines among application programs
- Transmitting messages from terminals to program queues and from terminal queues to terminals
- Honoring connection and disconnection requests from terminals, and data transfer requests for VCAM subsystems
- Monitoring constantly network activity such as message switching and error logging.

A detailed description of BTNS is dealt with in Section II.

Message Access Method (MAM)

MAM is a distributed queueing access method which operates with BTNS to form the Message Control System (MCS).

MAM provides such functions as,

- Compatibility thru MCS with standard COBOL communications language elements, i.e., CD area, ENABLE, DISABLE, SEND, RECEIVE and ACCEPT verbs
- Access to memory and disk queues, with 2 levels of queue available for input
- Checkpoint/restart capability for disk queues
- Allocating queues to application programs
- Message editing according to terminal type
- End-to-end protocol between the terminal operator and the application program as provided either by control messages generated by MAM or by status generated by BTNS

- Communication between process groups, i.e., communications load modules, and between processes, i.e., tasks
- Multitasking within an application program from 1 thru 6 user processes.

A detailed description of MAM is dealt with in Section III.

Virtual Communications Access Method (VCAM)

VCAM is a distributed access method which allows direct communication between communications subsystems and terminals via BTNS, or between the subsystems themselves.

The communications subsystems are,
- IOF, interactive operation facility
- RBF/6, remote batch facility
- TDS, transaction driven system

An example of such communication provided for by VCAM is the communication between TDS and a batch entry utility which allows access to transactions from the batch simulating terminals for TDS.

The subject of IOF and TDS is treated in "User Applications" later on in the section.

Communications Network Configurator (CNC)

CNC is a utility program for creating a communications environment which includes the structure of the network and the way it is to function. This environment is described through the Network Description Language (NDL). The set of NDL commands for the communications environment is defined by the user to describe pertinent aspects of the lines, stations, terminals, queues and correspondents, such as BTNS, VCAM subsystems and batch programs. This description is then processed by CNC.

CNC performs such functions as,

- Checking the syntax of commands
- Crosschecking description details against information previously established in system tables
- Generating terminal tables and line buffer pools
- Preformatting a disk queue file, where applicable, on the information provided.

A detailed description of CNC is dealt with in Section V.

Queue Maintenance (QMAINT)

QMAINT is a utility program for executing maintenance actions on memory queues or disk queues.

QMAINT performs such functions as,

- Printing the contents of a queue
- Displaying the status of a queue
- Purging a queue
- Filling a queue with defined data.

A detailed description of QMAINT is dealt with in Section VI.

USER APPLICATIONS

User applications available thru Level 64 communications are,

- IOF
- RBF/6
- TDS
- MCS COBOL applications

Interactive Operation Facility (IOF)

IOF is a system interactive utility which provides the user with the necessary tools for program preparation, namely,

- Creating, updating and deleting source files or subfiles containing data or JCL
- Launching batch jobs from a terminal
- Scanning and receiving output reports from batch jobs.

IOF is a system load module. When the user requests connection to IOF, an IOF step is launched dynamically by BTNS. An IOF step is launched for each remote user working with IOF. The step is terminated on disconnection of the user.

For further details, see Interactive Operator Facility Manual.

Remote Batch Facility/6

RBF/6 is a system utility for executing remote job entry from Level 6 satellite processors to a Level 64 host system. Printed output reports can be routed to the Level 6 processors. For further details, see the Level 64 GCOS Remote Batch Facility Manual, Order No. CQ35.

Transaction Driven System (TDS)

The user provides COBOL programs in the form of Transaction Processing Routines (TPR). Each TPR can receive a message to process data and generate a response.

TPRs are linked dynamically to the TDS executive. The advantage of dynamic linking is that TDS generation is not necessary each time a TPR is modified or added.

Unlike IOF, TDS occupies 1 level of user multiprogramming.

For debugging purposes, the TDS version can be executed as a job step with a simulated network of batch entry programs functioning as terminals.

For further details on TDS, see the appropriate manuals listed,

- TDS Programmer Reference Manual, Order No. AQ57.
- TDS/64 User Guide, Order No. AQ56.
- TDS/64 Standard Processor Site Manual, Order No. AQ55.

MCS COBOL Applications

An MCS COBOL program can be either a monoprocess load module or a multiprocess load module executed as a job step. The load module is created from COBOL compile units (cu) by the Static Linker, see Section III "Linking MCS COBOL Programs".

An MCS COBOL program is recognizable either to BTNS or to another MCS COBOL program by the input queue(s) from which it receives messages. Conversely, an MCS COBOL program recognizes only the output queue(s) to which it send messages; it does not recognize BTNS or the destination terminal or another MCS COBOL program.

RELATING THE COMMUNICATIONS ELEMENTS

The following points summarize the communications package,

- BTNS is a system load module H_BTNS residing in the system load-module library SYS.HLMLIB and run as a service job, like IOF, which does not occupy a level of user multiprogramming.
- MAM and VCAM are the software support components for user applications and communications subsystems. They are distributed access methods shared at system level among BTNS, QMAINT, MAM application programs and VCAM subsystems.
- CNC is a system load module H_CNC residing in the system load-module library SYS.HLMLIB and run as a job step each time a network needs to be generated or modified.
- QMAINT is a system load module H_QMAINT residing in the system load-module library SYS.HLMLIB and run as a job step each time maintenance actions are required on MAM queues existing on the site.
- MCS COBOL programs and VCAM subsystems, i.e., IOF and TDS, are load-modules residing in a "private" library or a system library and run as job steps. TDS occupies one level of user multiprogramming. Each IOF user occupies one level of interactive-job multiprogramming.

SECTION II

BASIC TERMINAL NETWORK SUPPORT

During the communications session, BTNS performs the following functions,

- prepares the communications environment at BTNS start-up
- dispatches events
- monitors the network thru the site controller
- processes input/output requests
- processes messages
- handles errors
- performs shutdown procedures
- performs message-switching

START-UP FUNCTIONS OF BTNS

BTNS is started by the network control command ST. Each time BTNS is run, it performs the following functions,

- restores the network tables
- initializes and sizes buffer pools
- initializes the IURP (integrated unit record processor)
- initializes the VCAM interface (virtual communications access method)
- initializes the network

Restoring the Network Tables

Network tables are stored in one segment created at network generation by the CNC utility. These tables are restored each time BTNS is started from a copy of the communications system segments existing on backing store. Any modifications made to the network during a previous session are ignored and only the original network description generated is restored.

The tables concerning MAM or VCAM, and in particular queue pools, are not restored because they can still be used by the application programs in the absence of BTNS.

Initializing and Sizing Buffer Pools

BTNS uses buffer pools to execut data transfers over communications lines and for network control functions. A ffer pool is created for the exchange of messages between terminals and queues. is a set of chained buffer units of fixed length. The number and size of buffer units are specified at network generation. Each buffer pool occupies a segment no greater than 64K bytes of main memory. At start-up, BTNS creates the segment(s) and initializes the buffer unit chains.

If VCAM subsystems are to be run, a second buffer pool, the VCAM buffer pool, is created to contain messages sent from the subsystems to the terminals.

Both buffer pools are either tailored by the user or automatically generated by H_CNC according to default values, see GENCOM command in Section V.

Initializing the IURP

When system initialization is performed, IURP firmware is loaded and all the unit record devices are initialized. At BTNS start-up, a further initialization sequence is required for the following,

- to prepare firmware tables of the communications lines
- to enable equipment, such as modems
- to activate firmware functions concerning communications lines.

For each line declared active at network generation, the following functions are performed as part of IURP initialization, viz.,

- loading the TCT (translation control table)
- reading, updating and loading the TLT (terminal line table)
- loading the polling list
- initializing data-sets (modems)
- error handling during IURP initialization.

LOADING THE TCT

The firmware uses the TCT to translate internal code EBCDIC to line code, generally ASCII, when sending, and vice versa, when receiving messages. In addition, the TCT defines special functions such as,

- an "erase" character on input
- end-of-block characters for input and output.

For each line procedure, there is a separate TCT.

READING, UPDATING AND LOADING THE TLT

The characteristics of each line are defined in its TLT which includes,

- line type, "leased" or "private"
- automatic connection of the modem
- half-duplex or full-duplex.

Certain other characteristics can be modified at network generation by the user, e.g., TOLEN time-out parameter of the LINE command, see Section V, or by the field engineer. BTNS must therfore read the tables in order to ascertain if any modifications are to be made before loading the tables.

LOADING THE POLLING LIST

Except for lines using the TTY line procedure, a polling list must be loaded for each line. BTNS loads into the IURP a list of stations to be polled on the lines declared at network generation.

Whenever BTNS requests data on a line, MLA firmware begins polling stations indicated on the list either in "round-robin" or "linear" sequence until a station replies. Polling recommences at the station defined by the type of sequence used, see LINE command in Section V.

INITIALIZING DATA-SETS (MODEMS)

At start-up, BTNS initializes the modem at the central processor (CPU) for leased and private lines. BTNS requests the starting of the search for the ring indicator for each switched line.

ERROR HANDLING DURING IURP INITIALIZATION

If an error is detected on a line during initialization of the IURP by BTNS, the system console operator is notified.

The line is declared as HELD and the system operator can retry the initialization of the line immediately or later on by issuing the RT network control command.

Initializing the VCAM Interface

BTNS identifies itself as a correspondent of VCAM and accepts any connection requests from any of the VCAM subsystems.

Connection enables exchanges between BTNS and the subsystems. These requests could have been enqueued earlier, awaiting the start-up of BTNS.

Initializing the Network

The final stage in BTNS start-up is the launching of network control procedures to perform the logical initialization of the network which consists of,

- initializing the network control interface
- connecting, if possible, all terminals identified as automatic and assigned at network generation
- activating the BTNS dispatcher.

BTNS DISPATCHER

The BTNS dispatcher is responsible for event management. It activates the appropriate BTNS functional component according to the type and source of the event that has occurred. These events originate from the network, application programs or other system components.

SITE CONTROLLER FUNCTIONS

Functions of the site controller are broadly divided as follows,

- controlling the network
- sharing network resources.

Controlling the Network

The entities constituting the communications system are,

- BTNS
- communications lines
- terminals
- queues
- VCAM subsystems.

The status of each of these entities is determined by its availability, i.e., either "open" or "close", and the parameter values used to describe it at network generation. The status can be altered thru the network control commands and to a lesser extent, by the terminal operator commands, see Appendix F.

A change in the status of any entity involves updating the communications tables and notifying all other entities affected by the change.

Activity on communications lines is constantly monitored. The two events which cause BTNS to take automatic action are,

- error detection
- attention notifications.

ERROR DETECTION

Any error detected on a line or terminal during message exchange is recorded by IURP firmware as an abnormal termination of transmission. The BTNS dispatcher is notified of this event which results in subsequent message retries according to the number specified for the appropriate LINE command at network generation. If the exchange still cannot be successfully completed, the following actions are taken automatically,

- the appropriate line, station or terminal is closed
- the network control operator or the system console operator is notified
- the disconnected terminals are then reported to the program queues defined with the BREAK option or VCAM subsystems to which they were currently connected.

A line, station or terminal which has been closed, can be reactivated by the RT network control command.

ATTENTION NOTIFICATIONS

Certain events unrelated to data transfer but concerned with a change in the status of an entity, are reported to BTNS thru attention notifications. These notifications which result in automatic actions are,

- ring indicator : A search for the ring indicator is begun for all switched lines at BTNS start-up. At the outset, all switched lines are closed. When a call is made over such a line, a ring indicator attention is issued, notifying the network control interface to set the status of the line to "open". This in turn causes the resource sharing portion of the site controller to begin a log-on dialog with the calling terminal.

- disconnect line : A disconnect line attention is issued when a specified time, preset in IURP firmware has elapsed without activity on an open switched line. This causes the network control interface to,
  - disconnect the terminal on the line
  - close the line
  - notify the network control operator.

  BTNS then recommences the search for the ring indicator.
  The disconnect line condition can only occur during the log-on sequence. Once the terminal is logged-on to an application, BTNS tries to receive data from the terminal when no output is pending. If no data is entered within the lapse of time defined by the TOLEN parameter of the LINE command, then "time-out" occurs and the condition is treated in the same way as for disconnect line.

- station on/station off : When the control unit of a station is turned on, the IURP firmware issues an attention. The network control interface notifies the resource sharing interface whether log-on procedures may be accepted. Terminals which are declared automatic or assigned must be initialized.
  When the station is turned off, the network control operator is notified, and each terminal is disconnected. Application programs and program queues are also notified of the disconnections.

Sharing Network Resources

Network resources are,

- communications lines
- terminals.

These resources are managed by the network resources sharing interface of the site controller which oversees their distribution between MCS COBOL applications and VCAM subsystems.

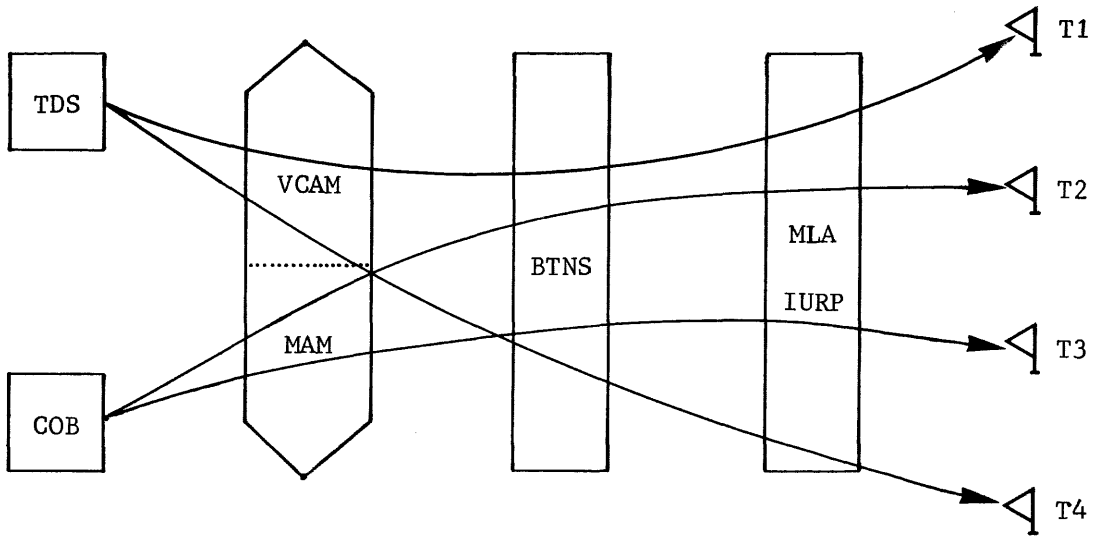
The communications system establishes the link between a terminal and the user application. A terminal is reserved for the application to which it is connected until the connection is broken, in which case, it is then eligible to be connected to another application. An application can have several terminals connected to it at any given time.

A line is not exclusively reserved for any one application. Several applications can therefore be connected simultaneously over a multipoint line. A network can theoretically be generated to allow all its terminals to communicate with all applications sometime during a communications session.

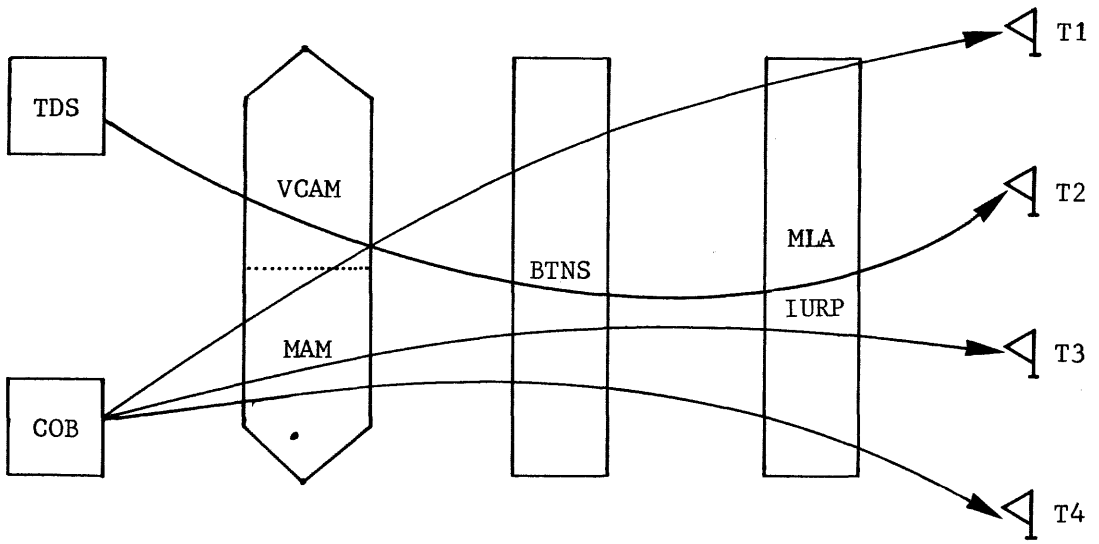
The two types of connection are,

- direct connection : For VCAM subsystems, the connection to the terminal is direct.
- logical connection : For MCS COBOL application programs, the connection to the terminal is logical to allow for the following conditions,
  - the program can be connected to an output queue whose destination terminal is not active
  - the terminal can be connected to an input queue whose associated program is not executing.

Sharing Network Resources



- terminals T1 and T4 are connected to TDS (VCAM subsystem)
- terminals T2 and T3 are connected to the MCS COBOL application named COB
- the terminals have been logged on either automatically or manually



- terminals T1 and T4 have logged off TDS and are now connected to COB
- terminal T2 has logged off COB and is now connected to TDS
- terminal T3 remains logged on to COB

LOG-ON PROCEDURES

The log-on procedure to be used depends on the type of terminal, namely,

- general purpose terminal : A general purpose terminal can communicate with any MCS COBOL application thru a program queue or a VCAM subsystem present in the system. A program queue defined at network generation is available to such a terminal as soon as BTNS is started. The VCAM subsystem is present in the system if it has been declared in the DCTAP command at generation, see Section V.

  The request for log-on depends on the line type,
  - for switched lines, a call is placed
  - for leased lines or local lines (multipoint or point-to-point), a BREAK signal is sent.

  Connection is established if:
  - the terminal and line are "open"
  - the application or VCAM subsystem is known to the system and active
  - all information has been correctly supplied by the operator for a terminal declared with the CONTROL option in its TERMNL command, namely,
    - ° user identification, which is checked against the system catalog
    - ° name of the application to which the terminal is to be connected.
  - for VCAM subsystems, the number of terminals has not exceeded the maximum.

  A message other than the BREAK signal sent by a terminal that is not connected will be ignored. After connection has been established, the BREAK signal is only considered if the QUEUE command defining the queue to which the terminal is connected has been declared with the BREAK option or if the terminal is connected to a VCAM subsystem.

- dedicated terminal : A dedicated terminal can log on only to a specific application named in the ASSIGN option of the TERMNL command at network generation. When the request to log on is made, BTNS establishes the connection if the conditions as for the general purpose terminal have been fulfilled.

  The MTT network control command can alter the status of a dedicated terminal to a general purpose terminal and vice versa.

- automatic terminal : A terminal declared with the AUTO option in the TERMNL command at network generation requires no operator intervention to establish a connection.

  For general purpose terminals, the connection is established as follows,
  - with MCS COBOL program(s) when the output queue associated with the terminal is enabled and contains data to be dispatched
  - with VCAM subsystems when they request the terminal.

  For dedicated terminals, the connection is established as follows,
  - with the MCS COBOL program when BTNS is started
  - with VCAM subsystems, only with the TDS version when it becomes active; for IOF and ROF, connection is established thru the automatic start by the site controller.

  This capability is designed for, but not restricted to,
  - terminals without a log-on capability, such as receive-only terminals and central processors, to be connected to a program queue
  - terminals which cannot be dedicated because they are used successively by different users.

- terminal cluster : The terminal cluster or multi-terminal station, such as the VIP7700, can be defined with one or several masters for other slave units attached to the same station. This is done by defining consecutive TERMNL commands with the SLAVE option following a TERMNL command for which a valid terminal type has been specified, see Section V.

  This master/slave concept is solely for the convenience of logging on which involves the following,
  - when the master logs on to an application, all its slaves are logged on simultaneously
  - if the master is a dedicated terminal, all its slaves are dedicated to the same application
  - no slave can log on or be dedicated to an application independent of its master.

LOG-OFF PROCEDURES

A terminal is logged off in a variety of ways, namely,

- the terminal is turned off
- a transmission error has occurred and a number of retries has been unsuccessfully attempted
- an abnormal condition has occurred while BTNS attempts to queue a message received from the terminal
- the terminal is disconnected by the VCAM subsystem to which it was connected
- the VCAM subsystem to which the terminal was connected, has terminated
- the following operator commands have been issued,
  - HT network control command identifying the terminal to be released
  - DIS terminal operator command issued at the terminal itself; reconnection for a dedicated terminal is made thru the mechanism specific to the type of terminal.

PROCESSING INPUT/OUTPUT REQUESTS

From the point of view of BTNS, messages can be exchanged between the following entities,

- between a terminal and the site controller during the log-on procedure or thru the site controller commmands
- between a terminal and a VCAM subsystem if the subsystem is executing
- from a terminal to a program queue if the queue is enabled
- from an output queue to a terminal if the queue is enabled.

Input Requests

When no output requests are outstanding, the following takes place,

- for non-polled lines of the TTY line procedure, BTNS requests the MLA, that is the IURP firmware, to accept any input
- for polled lines, the search for input requests is made either in the "round-robin" or "linear" sequence.
  The polling sequence is defined for the line by the order in which the stations are specified at network generation or by declaring explicitly the polling list thru the POLLIST parameter of the LINE command. This sequence can be modified by the MTP network control command.
  When polling finds an input request, the BTNS input request routine directs the message according to the application as follows,
  - if the terminal is connected to a queue of an MCS COBOL program, then one of the alternatives occurs,
    - the message is sent to the enabled queue
    - the message is discarded if the queue has been disabled.
  - if the terminal is connected to a VCAM subsystem, then one of the alternatives occurs,
    - the message is sent directly to the VCAM subsystem
    - the message is discarded if the terminal in question is to receive data, that is, an output message.

Output Requests

When an output request is outstanding, the BTNS output request routine moves the message into the BTNS buffer pool associated with the line to which the destina tion terminal is attached, and the output request is queued until it can be honored.

Notification of the BTNS output request routine means that,

- for an MCS COBOL program, the message has been sent to the output queue associated with the destination terminal
- for a VCAM subsystem, the subsystem itself identifies the message and the terminal to which it is to be sent.

An output request is honored when an end-of-operation occurs on a line , i.e.,

- when a data exchange (input or output) has been completed
- when an end to the polling cycle is requested by BTNS.

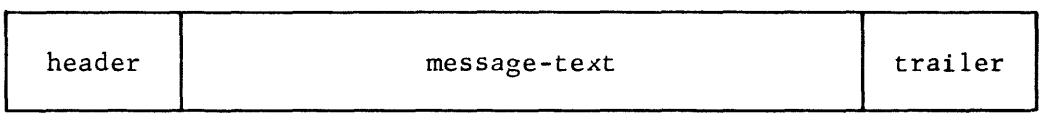
Output requests are queued as they are received and are handled on a first-in/ first-out basis according to the following scheme,

- output requests for VCAM subsystems are serviced before those for MCS COBOL programs
- when no further output requests for VCAM subsystems are pending, the BTNS output request routine will then service output requests for MCS COBOL programs; if an output request for a VCAM subsystem should occur at the time when output requests for MCS COBOL programs are being serviced, servicing of the MCS COBOL requests is halted to allow the VCAM request to be serviced.

If there are no output requests pending for the line, polling of the line continues according to the scanning sequence declared at network generation, i.e., "round-robin" or "linear".

MESSAGE PROCESSING

The general format of a message exchanged over a line can be represented as follows,



The actual format a message takes is dependent on the line procedure used, viz.,

- TTY line procedure
- VIP line procedure
- BSC line procedure

The line procedures and terminal programming considerations are treated in Appendix B. MCS data formats are treated in Section IV.

The focus of the message processing routines is the logical header of the message, namely,

- for an input message, the logical header is built by IURP firmware from the transmission header sent by the terminal
- for an output message, the message processing routines pass a logical header to the IURP firmware which converts it into a transmission header.

ERROR HANDLING

The communications link comprises a variety of equipment including,

- terminals
- modems
- lines
- adapters
- processors.

A hardware error in any of these components must be recognized by the system and handled as follows,

- recovering from the error condition
- logging the incidence of errors for statistical purposes and maintenance.

Error Recovery

Recovering from hardware errors is done by both BTNS and IURP firmware (MLA).

- unrecoverable errors : These errors are normally hardware failures detected by the MLA during line initialization or message exchange. An error thus detected is reported to the BTNS error handling routine which performs the following,
  - closes the line or terminal affected
  - notifies the network control operator.

  When either the line or terminal has been repaired, it can then resume transmission when an RT network control command identifying the entity is issued.

  If the error is detected during line initialization, the system console operator has the option to request a retry or exclude the line from being used by the communications session.

- recoverable errors : A transmission error results in abnormal termination which can be caused by,
  - a parity error over the line
  - a loss of characters due to line interference.

  When this type of error is detected by the MLA, it attempts a number of retries specified at network generation. If none of the retries is successful, the error is considered unrecoverable and handled as above.

  Error conditions which can be recovered thru operator intervention include the following examples for the overflow condition on the VIP7700 which results in the ERROR indicator being illuminated on the terminal,
  - where the operator has failed to clear the screen, the actions for recovery are,
    - ° clear the screen
    - ° issue the RDY communications command
  - where the message size is too great for the screen, the actions for recovery are one of either,
    - ° issue the MTE communications command altering the "line-length" and "block-size" to truncate the message to a defined size
    - ° issue the RDY STRONG network control command to cancel the message.

Error Logging

All hardware errors known to BTNS are classified as either retryable or nonretryable and are logged accordingly.

Error logging is described in the System Operation: Operator Guide Manual.

SHUTDOWN FUNCTIONS

The communications session is shut down when BTNS is terminated.

Normal Shutdown

BTNS is terminated normally when the TT network control command is issued which results in the following,

- each line is systematically closed after all MAM exchanges on the line are completed
- VCAM subsystems are notified in order to be able to terminate exchanges at a "clean point" and disconnect any terminals
- housekeeping duties are then performed.

Fast Shutdown

Issuing the TT STRONG network control command results in a fast shutdown, namely,

- all lines are closed
- VCAM subsystems are notified that a fast shutdown is in progress
- the network control or system console operator is notified
- housekeeping duties are then performed.

Emergency Shutdown

Such a condition is due to an internal failure of BTNS affecting its performance. The procedure for emergency shutdown is the same as that for fast shutdown.

Housekeeping Duties

Housekeeping duties performed at shutdown include,

- deleting BTNS buffer pools
- disabling attention notifications on all lines
- clearing the areas used to enqueue output requests for VCAM subsystems, if present
- terminating the communications job step.

MESSAGE SWITCHING THRU BTNS

BTNS provides two methods of message exchange between terminals, thereby bypassing both MCS COBOL programs and VCAM subsystems, which are

- direct exchange
- queued exchange

Direct Exchange

Messages can be exchanged directly between terminals by issuing the BT communications command at the sender terminal specifying the receiver terminal and the message text. The receiver terminal is informed of the identity of the sender terminal by the network control interface which precedes the message text with FROM followed by the name of the sender terminal.

The conditions under which the command can be used are,

- both sender and receiver terminals are active and known to the site controller
- the receiver terminal is other than a cassette or processor
- the message text does not exceed 128 characters.

Queued Exchange

The sender terminal uses the log-on procedure to identify the eventual destination of the message, which can be either another terminal or itself.

During log-on, one of the following messages appears,

- CC00 ID,APPL?
- CC01 ID,USER/PROJECT/BILLING,APPL?

In response to APPL, the operator can key in the name of the receiver terminal or enter a "space" as an option if the receiver terminal is itself.

The only prerequisite is that the terminal and, by implication, its related queue which bears the same name, are defined at network generation.

- echo exchange :

  This involves sending back the message to the sender terminal thereby verifying its ability to send and receive.
  When the terminal transmits a message, BTNS places it in the queue named at log-on, i.e., the name of the terminal itself. When BTNS scans for output requests to the line, it discovers the message in the queue and sends it back to the sender terminal.

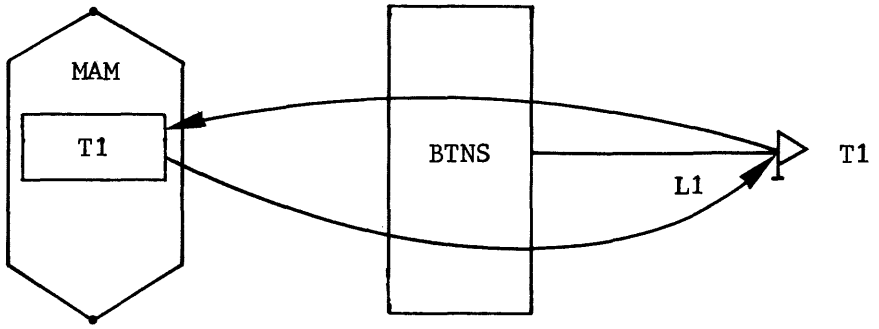
- exchange between interactive terminals :

  Such an exchange is possible with terminals having both input and output capability.
  Two terminals, say, can log on to each other's queue whereby messages input on the one terminal are output on the other terminal, and vice versa, i.e., assuming that both the terminals are connected.
  If one of the terminals is not connected at the time that the other is transmitting, the messages input are stored to the capacity of the queue until such time that the receiver terminal logs on. The stored messages are then delivered to the receiver terminal.

- sending messages to a cassette:

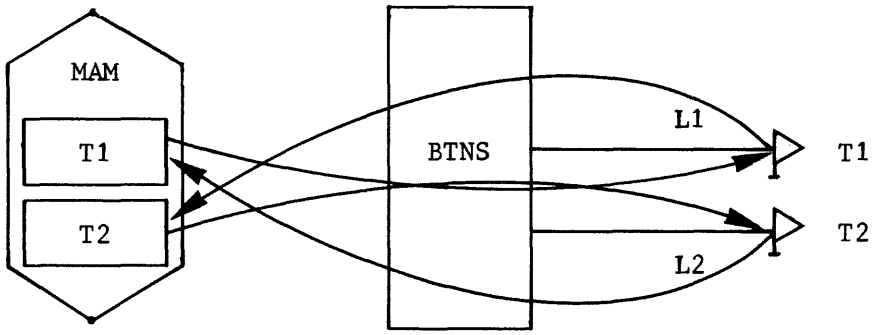
  The sender terminal is logged on to the queue related to the cassette , which will eventually be the receiver terminal. The cassette is logged on to the same queue in order to receive messages input to it by the sender terminal.
  More than one sender terminal may log on to this queue.

Echo Exchange



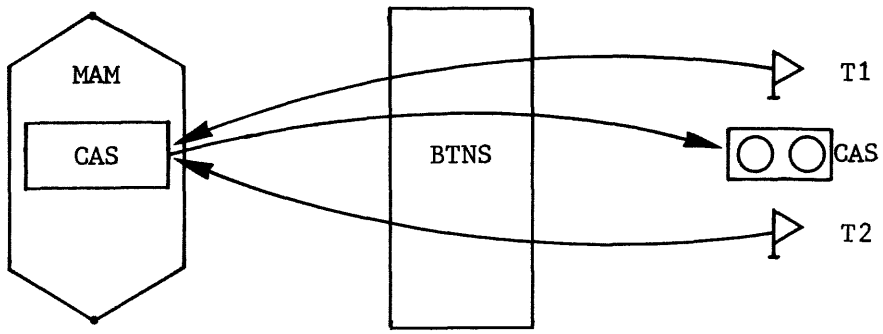
- terminal T1 transmits a message which BTNS places in queue T1
- when BTNS scans for output requests on line L1, it finds the message in queue T1 and sends it back to terminal T1

Exchange between Interactive Terminals



- terminal T1 transmits a message which BTNS places in queue T2
- when BTNS scans for output requests on line L2, it finds the message in queue T2 and sends it to terminal T2
- terminal T2 transmits a message which BTNS places in queue T1
- when BTNS scans for output requests on line L1, it finds the message in queue T1 and sends it to terminal T1

Sending Messages to a Cassette



- terminals T1 and T2 transmit messages which BTNS places in queue CAS
- the messages are delivered to the cassette associated with the queue CAS

SECTION III

MESSAGE ACCESS METHOD

MAM (Message Access Method) provides the means for the exchange of messages. It operates with BTNS (Basic Terminal Network Support) to form the Message Control System (MCS). MCS is the interface between the communications program and the terminals.

MAM is described in terms of:

- Communications elements of MCS COBOL
- Queue correspondence
- Structure of an MCS COBOL load module
- Device handling and message editing
- Dialog handling
- Data integrity
- Communication between local MCS COBOL programs
- Communication between remote applications

COMMUNICATIONS ELEMENTS OF MCS COBOL

The MCS COBOL communications elements are:

- The communications description (CD) area of the MCS COBOL program
- The communications verbs
- Message delimiters

CD Area of the MCS COBOL Program

CD areas contain information about queues, terminals and messages to be input and output. The MCS COBOL program must have at least 1 CD area for messages to be sent or received. If the application requires messages to be sent and received, then at least 2 CD areas, the input CD area and the output CD area, must be present.

Communications Verbs

The 5 communications verbs are:

- ACCEPT
- DISABLE
- ENABLE
- RECEIVE
- SEND

The communications verbs provide the user interface as follows,

- between MCS and the application,
  - ACCEPT : the MCS COBOL program ascertains the number of messages in a symbolic queue thru the ACCEPT with COUNT (ACCEPT MESSAGE COUNT) statement.
    Coding in the statement includes the name of the input CD area identifying the symbolic queue whose message number is to be ascertained.
  - RECEIVE : requests a message from a specified symbolic queue identified in the input CD area.
  - SEND : directs a message to a specified symbolic queue identified in the output CD area.
- between MCS and the terminals,
  - DISABLE : terminates the logical connection with specified sources or destinations for data transfer.
  - ENABLE : establishes the logical connection with specified sources or destinations for data transfer.

The status of the MCS COBOL interface is given by a set of key codes which are described in Appendix H.

Message Delimiters

A message is delimited by either an EMI (end-of-message indicator) or an EGI (end-of-group indicator).

The END-KEY value of the message delimiters are,
- "2" for EMI
- "3" for EGI

The following cases denote the way in which messages are delimited:

- from the terminal to the application (input),
  - non BSC terminal : the message is delimited by an END-KEY value of "3" or EGI.
  - BSC terminal : each block is terminated by the control character "ETB" (End-of-Transmission Block) and is treated by the application as an END-KEY value of "2" or EMI.
    The final block terminating the message text is delimited by the control character "ETX" (End-of-TeXt) and is treated by the application as an END-KEY value of "3" or EGI.
    NOTE : Where the message is of "zero" length, the END-KEY value may be either "2" or "3".
    After coding a RECEIVE verb, the programmer should test for the STATUS KEY code, TEXT-LENGTH and END-KEY before deciding if there is a message to be processed.
- from the application to the terminal (output),
  - non BSC terminal : the message can be delimited by an END-KEY value of either "2" (EMI) or "3" (EGI).
    In the case of TWA (Two-Way Alternate) mode, see QUEUE command in Section V, the EGI is necessary to allow dialog.

- BSC terminal : the message is transmitted in blocks, the maximum size of which is dependent on the type of the receiving terminal.
  A SEND with EMI statement results in the transmission of a data block terminated by the control character "ETB".
  A SEND with EGI statement results in the transmission of a data block terminated by the control character "ETX".

- communication between applications,
  EMI and EGI delimiters are transmitted by MCS and converted into the appropriate END-KEY value in the destination CD when a RECEIVE statement 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 at network generation by the QUEUE command, see Section V, and are identified by external-queue-names.

The physical queue can be
- a terminal queue : an output queue thru which a message is sent to the terminal.
  The name of the terminal queue is the name of the terminal as identified in the TERMNL command.
- a program queue : an input queue thru which a message is sent to the MCS COBOL program.
  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 in data-name-1 of the CD area in the MCS COBOL program. The queue can be either input (message reception) or output (message dispatch). In the case of the symbolic input queue, further partitioning into subqueues is possible by defining these in data-name-2 of the CD area. Only one level of subqueues is permitted.

Relating Physical Queues to Symbolic Queues

The $QASSIGN statement, see "Executing a Communications Step" at the end of the section, defines the symbolic queue or subqueue and establishes its correspondence with the physical queue. This correspondence is unique.

The $QASSIGN also defines the processing mode as follows,

- symbolic input queues : a symbolic input queue must be defined for each program queue from which messages are to be received by the program.
  The IN keyword of the $QASSIGN 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 COBOL program may issue a RECEIVE statement 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.

  Explicit definition of the symbolic output queue :

  The OUT keyword of the $QASSIGN serves
  - to explicitly identify the symbolic queue as an output queue
  - to allocate the terminal queue to the current step until step termination.

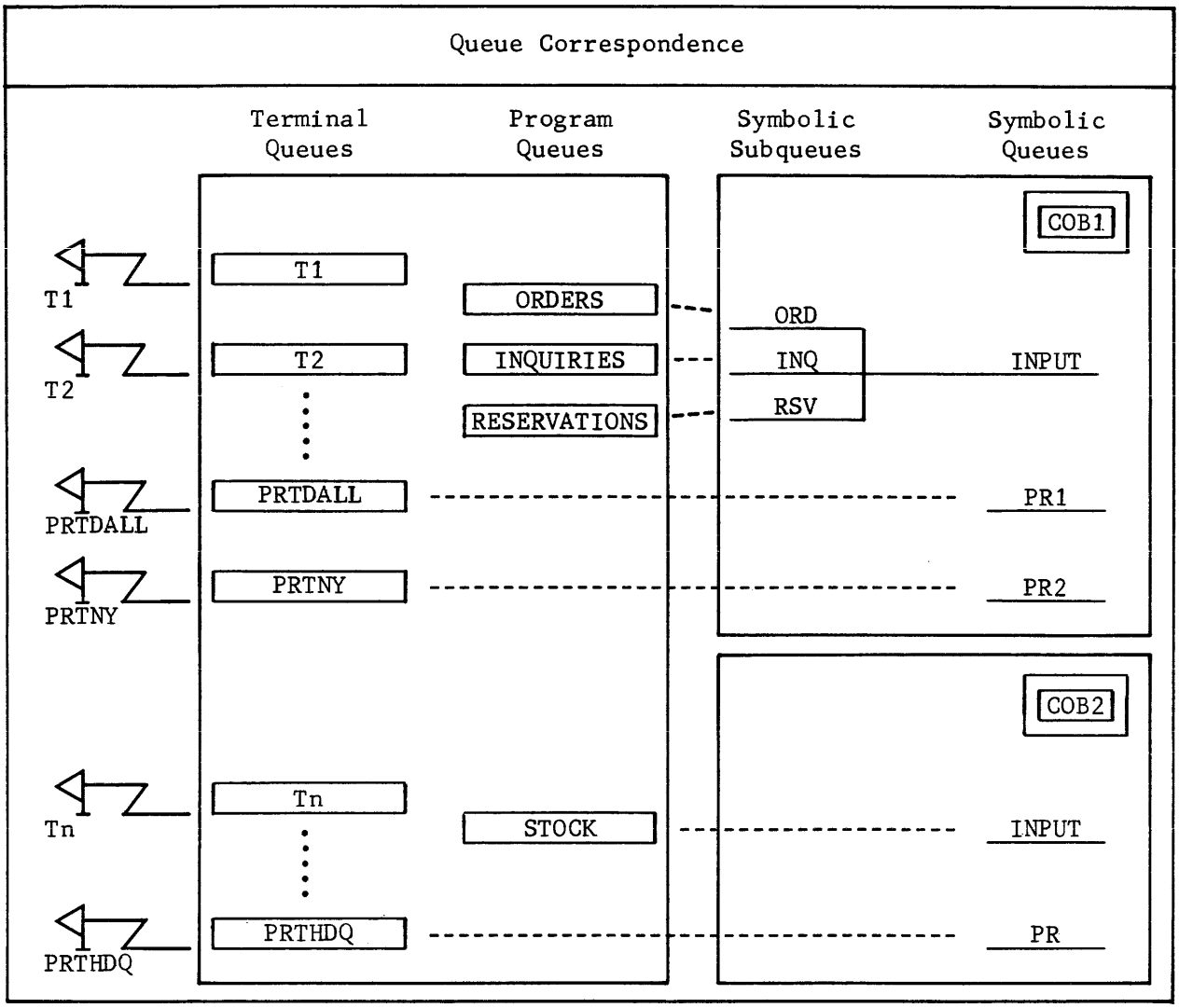
  The terminal is known to the MCS COBOL program exclusively by its explicitly defined symbolic output queue. When a message is received from this terminal, the symbolic source field of the input CD is updated by MCS with the output queue name.
  No other MCS COBOL program may issue a SEND command to this terminal queue and the terminal cannot be connected to another MCS COBOL application.

  Implicit definition of the symbolic output queue :

  The symbolic output queue is defined implicitly thru the log-on procedure of the destination terminal. When a message is received from the terminal, the symbolic source field of the input CD 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.

- symbolic I/O queues : a symbolic input/output queue is a program queue and hence does not have subqueues.
  The INOUT keyword of $QASSIGN 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 COBOL program may issue a SEND or RECEIVE command to the program queue thus allocated.



Program COB1 queue relationship

- Each of the program queues is related to a corresponding symbolic subqueue as follows, ORDERS to ORD, INQUIRIES to INQ and RESERVATIONS to RSV.
- The symbolic subqueues ORD, INQ and RSV are associated with the symbolic queue INPUT which means that when a RECEIVE statement is issued to INPUT, its subqueues are scanned until a message is found in 1 of them, e.g., INQ if say either T1 or T2 is logged on to INQUIRIES.
- The symbolic output queues PR1 and PR2 are related to their respective terminal queues PRTDALL and PRTNY associated with receive-only printers.

Program COB2 queue relationship

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

STRUCTURE OF AN MCS COBOL LOAD MODULE

An MCS COBOL load module comprises 1 thru 6 user processes which are basically equivalent to tasks and linked together by the Static Linker, see Section VI. Each process is associated with a program or "run-unit".

System handling of the load module is as follows,

- 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 MAM to perform housekeeping functions, whether the process terminates normally or abnormally.
  As a result of this feature, the program must be terminated with an EXIT PROGRAM statement and not a STOP RUN statement which returns control immediately to the system.

- multiprocess load module : the system procedure STUSERS starts each of the STUSERn processes, n ranging from 1 thru 6.
  Each STUSERn executes a call to the program thru a user-specified entry point.
  On termination of the process, STUSERn calls MAM to perform housekeeping functions before returning control to STUSERS.
  When all 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 Processes of a Multiprocess Program" later in the section
  - the checkpoint/restart facility is not available
  - only one process can execute the ACCEPT and DISPLAY verbs to gain access to the standard SYSIN and SYSOUT files.

DEVICE HANDLING AND MESSAGE EDITING

An MCS COBOL program is capable of such control functions as,

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

Control Messages

Control messages are commands affecting the MCS COBOL program-to-terminal interface and can be passed between the program and BTNS. The program views the command like any other message that it sends to a 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 : connection of a terminal
- DIS : disconnection of a terminal
- PRG : program request to BTNS to purge all existing messages in a terminal queue; the command is given top priority in the queue by BTNS
- RVI : program to issue a "reverse interrupt" to a BSC (Binary Synchronous Communications) line procedure terminal related to the specified output queue; the command is stored in the queue and processed in sequence
- SHT : request to application to shutdown

Points to note:

- The BRK, CNT and DIS commands should only be used for terminal simulation purposes. The application receiving these commands will be notified by the corresponding STATUS KEY code in the input CD.
- The SHT command can also be sent by the BT network control command of the format : BT xxxx ><CTLSHT, where xxxx is the program-queue-name, see Communications Network Control Terminal Operations Manual.

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. On completion of the RECEIVE statement, both parameters TEXT-LENGTH and END-KEY will be 0.

Status Changes

If the program queue has been defined with the BREAK option at CNC (Communications Network Configurator) generation, status changes of the terminal or of the system will be passed to the application thru the status key of the input CD as a result of a RECEIVE statement. The symbolic source identifies the related terminal.

If the BREAK option has not been specified, then the user will not be notified of any events occurring when a RECEIVE statement is issued.
Status key codes are explained in detail in Appendix H.

Data Flow Control

The control of data flow between MCS and the terminals is thru the ENABLE and DISABLE verbs functioning as follows,

- Input without terminal : ENABLE INPUT
  The related program queue specified in the input CD is "enabled", i.e., connection requests from terminals can now be accepted.
  NOTE : Terminals described in CNC with automatic log-on and assigned to one of queues or subqueues will be immediately connected.

  DISABLE INPUT
  The related program queue specified in the input CD is "disabled", i.e., no more connections to the queue can be accepted and any terminals previously connected are now disconnected.
  The application may continue to empty the queue thru RECEIVE statements and when the queue is empty, the status key is flagged as "disabled".

- Input with terminal : ENABLE INPUT TERMINAL
  The terminal whose name is specified as the symbolic source in the input CD can commence or resume input to the queue to which it was connected.

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

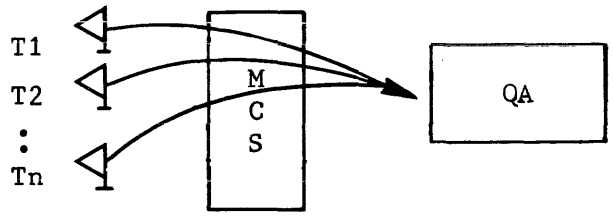
- Output : ENABLE OUTPUT and DISABLE OUTPUT apply only to terminal queues.
  If either verb is applied to program queues, no action results or a status key of "9F" is flagged.

  ENABLE OUTPUT
  The related terminal queue specified in the output CD is "enabled" as follows,
  - if the terminal is working in input mode, output to the terminal is resumed
  - if the terminal is working only in output mode, then the terminal becomes activated.

  DISABLE OUTPUT
  The related terminal queue specified in the output CD is "disabled" as follows,
  - if the terminal is working in input mode, output to the terminal is suspended
  - if the terminal is working only in output mode, then the terminal becomes deactivated
  - the application may continue to send messages to the queue.

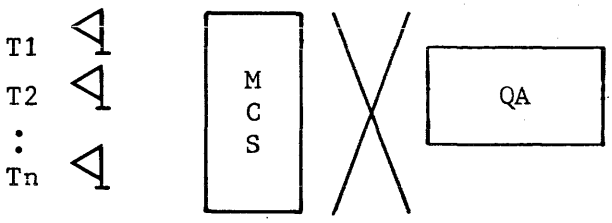
ENABLE and DISABLE

ENABLE INPUT



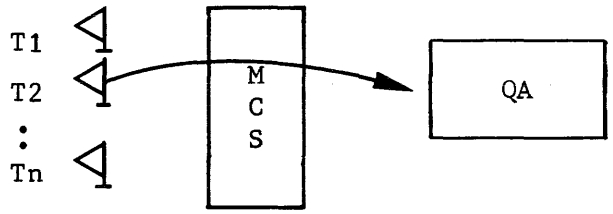
ENABLE INPUT with input CD specifying QA : connection requests to the queue and its subqueues are now accepted.

DISABLE INPUT



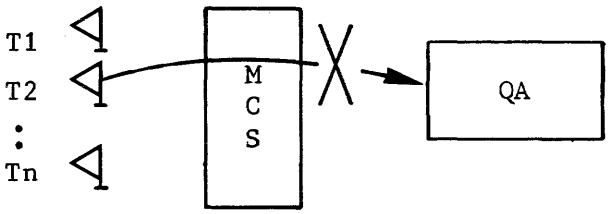
DISABLE INPUT with input CD specifying QA : all connected terminals are now disconnected; the application may continue to empty the queue.

ENABLE INPUT TERMINAL



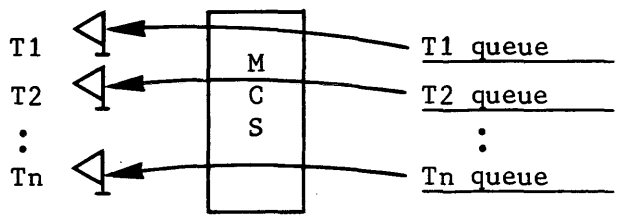
ENABLE INPUT TERMINAL with input CD specifying Tn, where Tn identifies the terminal and its related queue : the terminal identified can now transmit in input mode. If T2 is specified, it can transmit; other terminals continue to remain in the state they originally were.

DISABLE INPUT TERMINAL



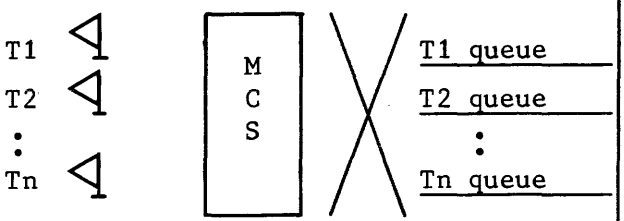
DISABLE INPUT TERMINAL with input CD specifying Tn, where Tn identifies the terminal and its related queue : the terminal identified can no longer transmit in input mode. If T2 is specified, it ceases to transmit; other terminals continue to remain in the state they originally were.

ENABLE OUTPUT



ENABLE OUTPUT with output CD specifying Tn, where Tn identifies the terminal and its related queue : data flow can now resume from the queue to its related terminal.
If T1 is specified, then only the queue T1 and its related terminal are "enabled".

DISABLE OUTPUT



DISABLE OUTPUT with output CD specifying Tn, where Tn identifies the terminal and its related queue : data flow is halted from the queue to its related terminal.
If T2 is specified, then only the queue T2 and its related terminal are "disabled".

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

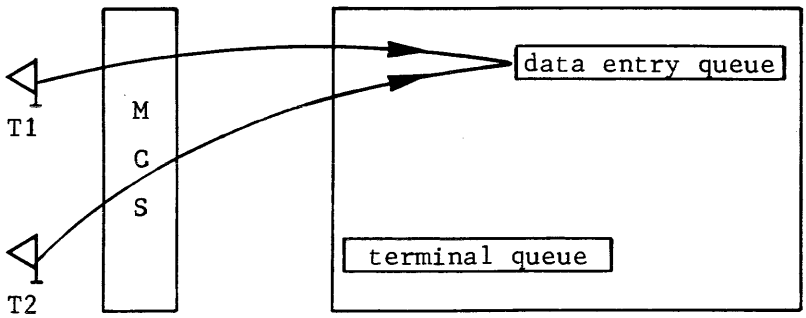
A typical non-interactive application involves the following stages,

- Data Entry : The application may be absent at the time when data is collected from terminals into program queues. Only BTNS has to be present in the system and the terminals must be connected to program queues.
  In order to allow the terminal to enter data without the application being present, the program queue must be described at CNC generation without the TWA option, i.e., defining the queue as a data entry queue.
- Batch Processing : When all data has been entered, the application can begin processing during off-peak hours, say, later at night. Processing involves updating files and preparing output data to be placed into terminal queues.
  BTNS is absent during batch processing.
- Data Distribution : BTNS empties the output data in the terminal queues to the respective terminals when these are activated.
  The application may be absent during data distribution.

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

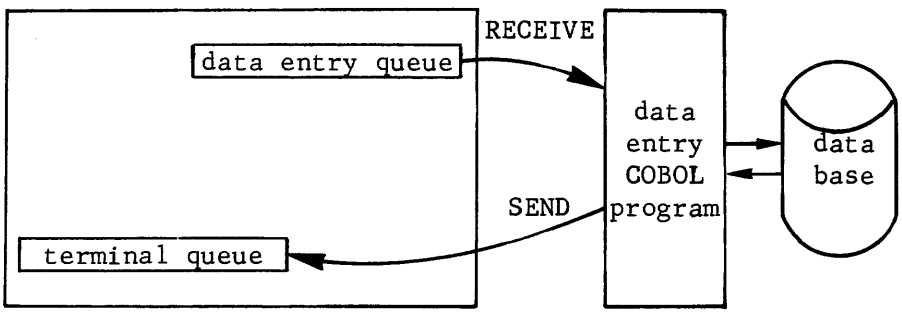
Non-interactive Application

data entry



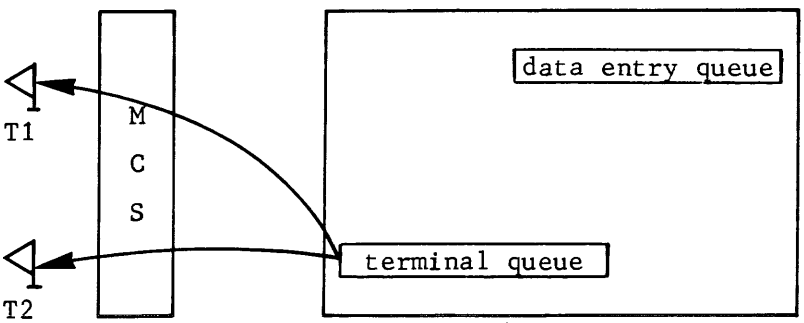
1 Terminals T1 and T2 can enter data at will to the data entry queue which is a program queue declared without the TWA option.

batch processing



2 Data collected previously from the terminals can now be processed at a time when the terminals have ceased transmission.

data distribution



3 When the terminals are activated, BTNS outputs the results at a time when the application program may be absent.

Interactive Applications

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

The TWA option determines the way in which the dialog is handled by the system.

- With TWA option : The program queue must be described with the TWA option at CNC generation. This enables an application-terminal dialog on a message basis.

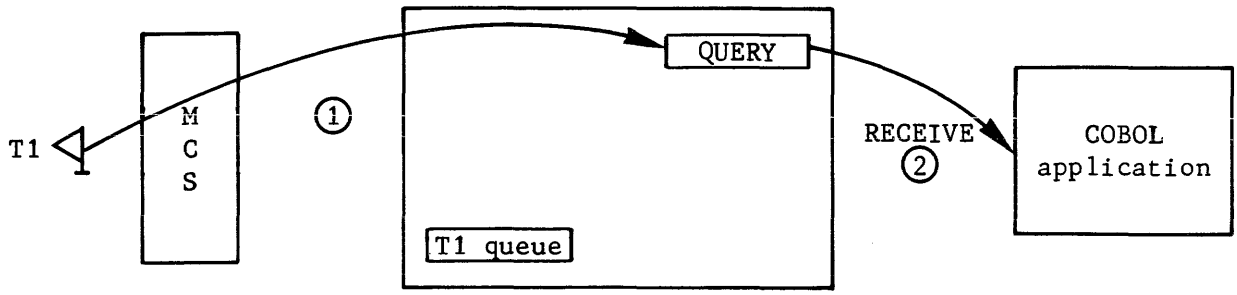
  On connection, the terminal has the turn-to-transmit.
  A message transmitted by the terminal is delimited by a COBOL END-KEY value of "3" (EGI) and the application then has the turn-to-transmit.
  The application can then transmit using the SEND statement.
  Transmission by the application is performed as follows,
  - Several messages delimited by EMI (end-of-message indicator) can be transmitted and the turn-to-transmit is still retained by the application
  - A message delimited by EGI (end-of-group indicator) transfers the turn-to-transmit to the terminal.

  The terminal's turn-to-transmit can be overridden at any time by the application.

- Without TWA option : In this case, the terminal can transmit messages at will.

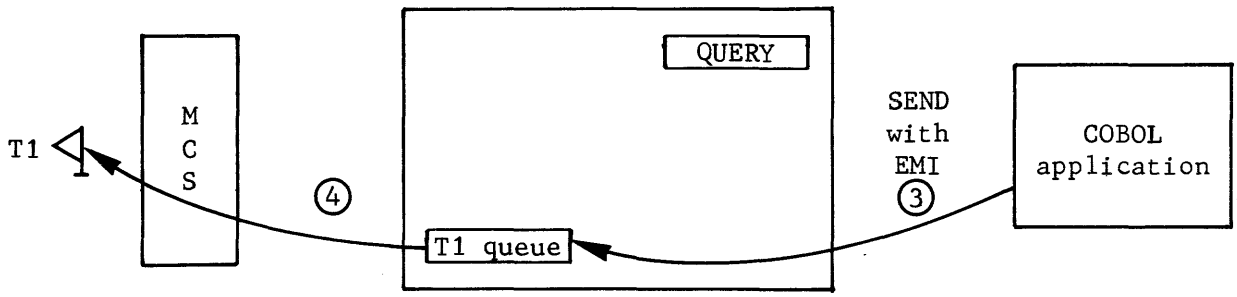
  For the visibility of messages by the application, see "Message Delimiters" earlier on in the section.

Interactive Application with TWA Option



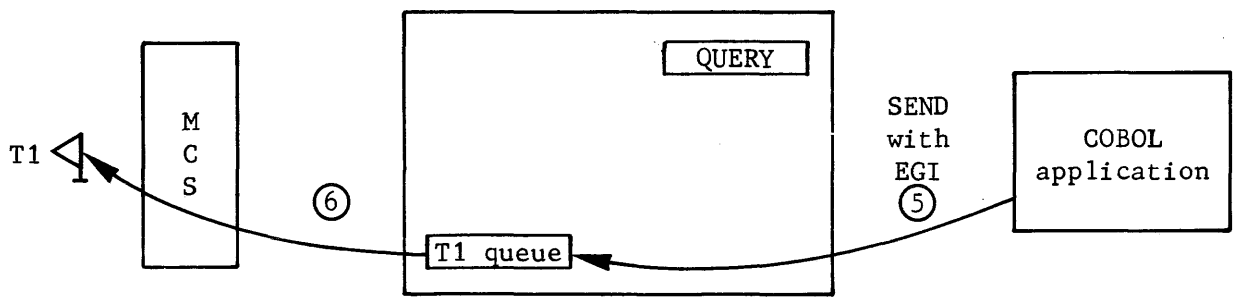
1 Terminal T1 enters a message into the program queue QUERY which has been declared with the TWA option at CNC generation.
The message is delimited by an END-KEY value of "3" to denote EGI.

2 The message is RECEIVEd by the COBOL application for processing.



3 After the application has processed the message, it can SEND several messages delimited by EMIs to the terminal queue T1.
For as long as messages from the application are delimited by EMIs, the turn-to-transmit is still retained by the application and any message entered on the terminal will be discarded.

4 As soon as a message is available for output, MCS will begin transmission to the terminal.



5 The last message in the stream to be sent by the application is delimited by EGI.
Altho EGI transfers the turn-to-transmit to the terminal, the application can override the terminal.

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

DATA INTEGRITY

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

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

Retention of Messages in Terminal Queues

When an MCS COBOL program issues a SEND statement, 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 is ENABLEd and BTNS is active, MCS attempts to transmit the message from the terminal queue to the associated terminal as follows,

- if transmission is successful, the message is deleted from the terminal queue
- if transmission is not successful, several retries according to the number specified are attempted
- 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 to the terminal
- a shutdown is effected; if the terminal queue is a core queue or disk queue without the RESTART option, then the message is lost
- H_CNC step is executed; even if the terminal queue is specified with the RESTART option, the message is lost
- ISL with REFORMAT option is performed; even if the terminal queue is specified with the RESTART option, the message is lost.

Message retention in terminal queues applies under such conditions as,

- BTNS "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 occurs,

- it is successfully retrieved by the application for processing
- a shutdown is effected; if the program queue is a core queue or disk queue without the RESTART option, then the message is lost
- H_CNC step is executed; even if the program queue is specified with the RESTART option, the message is lost
- ISL with REFORMAT option is performed; even if the program queue is specified with the RESTART option, the message is lost.

Checkpoint/Restart Facility

The facility allows the following functionalities,

- disk queues to be journalized in order to be rolled back, see RESTART option of the NDL QUEUE command in Section V
- the user program to be restarted if the REPEAT option has been specified in the $STEP statement.

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 program and user files are in the same state when restart takes place , as they were at the last checkpoint. If checkpoints have not been taken, then restart is at the beginning of the step.

The features of the facility are,

- issuing checkpoints :

  Only MCS COBOL monoprocess programs with disk queues can issue checkpoints by,
  - RERUN statements thru which checkpoints are taken at specified intervals of records processed for a particular file, see COBOL Language Reference Manual
  - CALL statements issued to a system checkpoint procedure at appropriate places in the program, see COBOL User Guide.

  The program queue is journalized if it has been
  - defined on disk with with the RESTART option in the appropriate QUEUE command at CNC generation
  - identified as an input queue for the step by the $QASSIGN statement specifying either the IN or INOUT parameters.

  Journalization is performed on the disk queue file itself and no additional file space need be allocated.
  A journal is considered "active" for a given program queue only for the duration of the MCS COBOL program step.

  Checkpoints have no effect on terminal queues except for terminal queues corresponding to a L64 CPU, to be discussed later in "Communications between Remote Applications".

- implementing checkpoints :

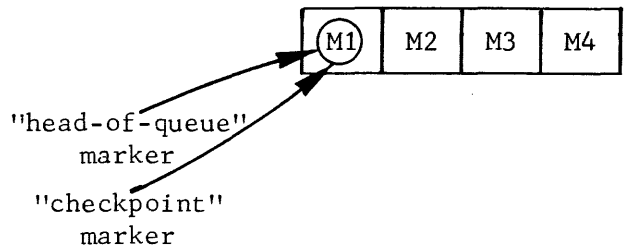
  For program queues described with the RESTART option, disk space related to messages received will be released only at checkpoint time or on termination of the MCS COBOL program. The user must take this into account when defining the size of those program queues at CNC generation and when deciding the frequency of checkpoints in his program.

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

Implementing Checkpoints

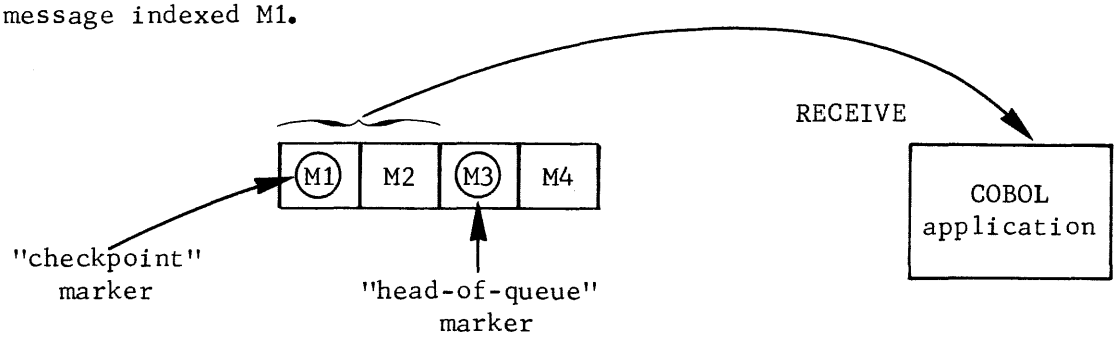
queue status after checkpoints

t0



At time t0, both "head-of-queue" marker and "checkpoint" marker point to the message indexed M1.

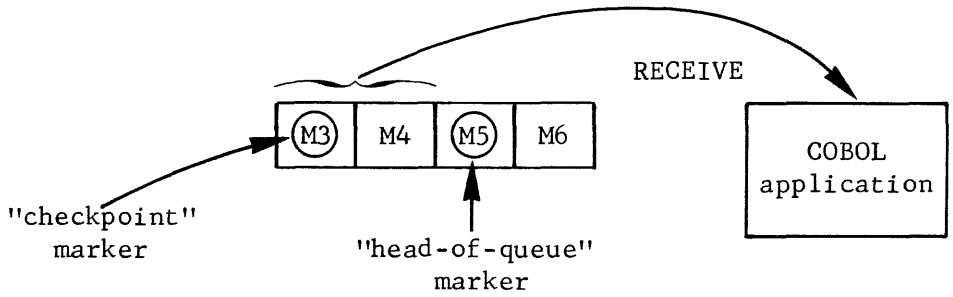
t1



At time t1, the following has occurred,

- the COBOL application has RECEIVEd the messages indexed M1 and M2
- the "head-of-queue" marker now points to the message indexed M3
- the "checkpoint" marker has not moved and still points to the message M1.

t2



At time t2, when the application has successfully processed the messages indexed M1 and M2, and taken a checkpoint, the following has occurred,

- the "checkpoint" marker has moved to point at the message indexed M3
- messages M5 and M6 now arrive in the queue
- the application has now RECEIVEd messages M3 and M4
- the "head-of-queue" marker now points to the message indexed M5.

If an abort now occurs, the status at RESTART will be,

- the "head-of-queue" marker will be "rolled back" to the position of the "checkpoint" marker, namely to the message indexed M3
- the messages to be RECEIVEd by the application will be M3 ,M4, M5 and M6.

- conditions for restart :

A job step can be restarted at the beginning or from the latest checkpoint if the $STEP statement contains the REPEAT parameter.

- 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 reinitialized.
  - 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 being transmitted.
    Restart of terminal queues with the CTLRST option, see Section V, is dealt with in "Communications between Remote Applications".
  - 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, i.e., queues not assigned by the MCS COBOL program at the time of the crash, are restarted from their current state.

  Points to consider:
  - The MCS COBOL program restarted from a 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 which case, the output message is duplicated.
  - If the program wishes a terminal to be aware of its checkpoints, it can send a message to the terminal indicating the checkpoint number. On restart from checkpoint "i", the program should send a message to the terminal indicating restart from checkpoint "i", so that the terminal operator is aware of the repetition of messages.
  - Messages being transmitted at the time of crash should be retransmitted by the terminal operator at restart; the program should check for redundant messages by requiring that all input messages include sequence numbers.
  - If the program queue needs to be purged following a restart, the program may purge the queues by issuing a series of RECEIVE statements equal to the message count.

- 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 RESTART commence following the message successfully transmitted or at the one being transmitted.
    Program queues with RESTART commence from
    - the "head-of-queue" for normal termination of the session
    - the last checkpoint under the conditions,
      - that the queues have an active journal
      - that the session terminated abnormally.

- MAM=NO Unless a new CNC session takes place, MAM is not available. Disk queue contents remain in the state they were at the time of termination of the session.
  This action is taken in cases where MAM is not required to save start-up time and space in system resident memory.
- MAM=REFORMAT MAM start-up operations are performed whereby all queues are reinitialized.
  The disk queue file is reformatted using the copy of the telecommunications system tables in backing store.

Points to consider in using the RESTART facility,
- Program queues can be filled during a session when no MCS COBOL programs are active
- The contents of such queues can then be retrieved in subsequent sessions when MCS COBOL programs 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 clause of the CD area to be updated with the appropriate return code to inform the user. For details of Status Keys, see Appendix H and the COBOL Language Reference Manual.

- message overflow : The maximum message length is related to the queue block size parameter declared at CNC generation.
  See Section V "QUEUE command" (QDBLKSZ for disk queues, QCBLKSZ for core queues).

  Maximum message length is given by the algorithm,

  $$\left[(b - 50) \div 4 + 2 \times \left[(b - 4) \div 4\right]\right] \times (b - 5)$$

  where b = queue block size, and
  $\div$ = integer division

  When message length, including control characters or marks, is exceeded, the excess portion of the message is truncated.
  Status key code setting is as follows,
  - 94, (MSGOV) for the sender
  - 95, (NOTALL) for the receiver.

  Maximum message size should be defined by the application according to,
  - terminal display size
  - transmission line 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 statement, the application should issue a CALL to the timer before attempting to execute that SEND.

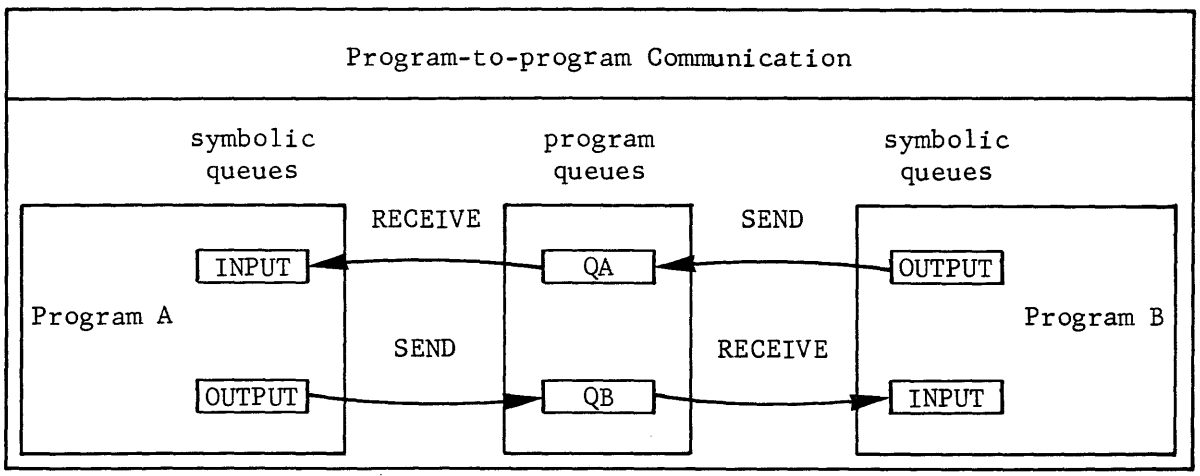
Status key code setting for the relevant condition is,
- 91, (SPACENAV) for the sender to be notified of lack of space in a disk queue
- 92, (BUFNAV) for the sender and the receiver to be notified that there is insufficient memory space to handle a disk I/O operation during message transfer
- 95, (NOTALL) for the sender to be notified that the number of core blocks is insufficient.

COMMUNICATION BETWEEN LOCAL PROGRAMS

The term "local" implies that the programs are in the same central processor. Communication is established by manipulating program queues. The program queue whose name is to appear in the symbolic source of the destination input CD is specified by the keyword REPLY in $QASSIGN OUT.
The ENABLE and DISABLE verbs are ineffective.

The 3 types of local communication are,
- Program-to-program communication
- Program communicating with itself
- Communication between processes of a multiprocess program

Program-to-program Communication

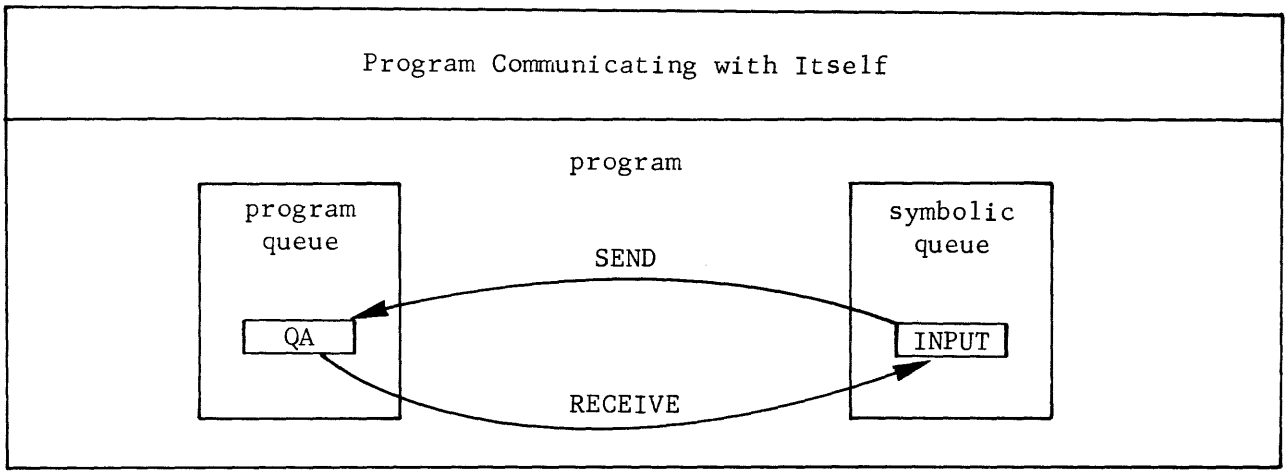


Program A receives as input, messages supplied by program B and vice versa.

The queue assignments required to allow these exchanges are as follows,

- Program A : The symbolic queue INPUT must correspond to program queue QA for input (parameter IN).
  The symbolic queue OUTPUT must correspond to program queue QB for output (parameter OUT) and must be specified with REPLY=QA.
- Program B : The symbolic queue INPUT must correspond to program queue QB for input (parameter IN).
  The symbolic queue OUTPUT must correspond to program queue QA for output (parameter OUT) and must be specified with REPLY=QB.

Program Communicating with Itself



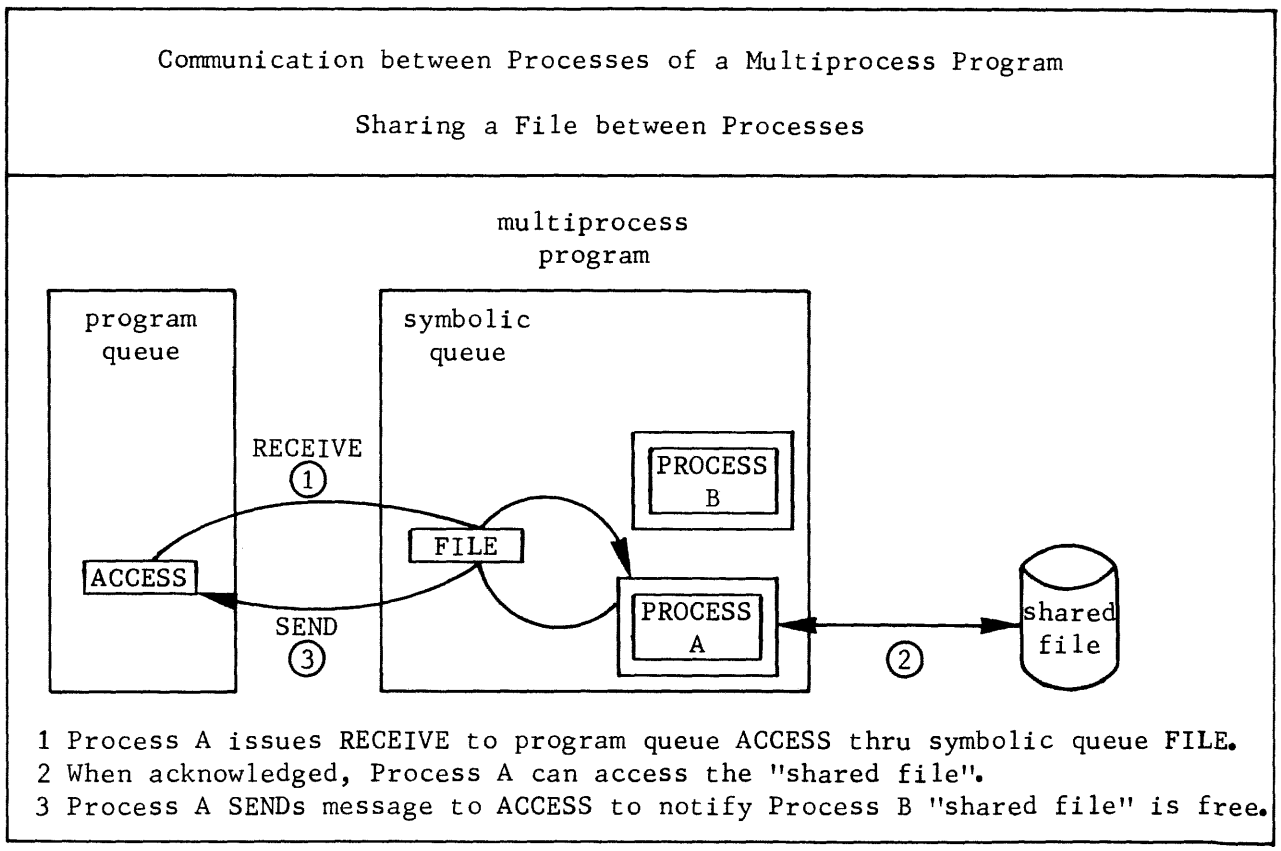
The program communicates with itself in the following stages,

- It acts as a terminal and fills the program queue QA
- It then receives messages from the same program queue QA as it would receive messages from any terminal connected to it.

The queue assignment required for this exchange is :

- The symbolic queue INPUT corresponds to program queue QA for input and output (parameter INOUT) and must be specified with REPLY= QA.

Communication between Processes of a Multiprocess Program



The queue assignment required to regulate exchanges between processes for access to a shared file is :

- The symbolic queue FILE corresponds to program queue ACCESS for input and output (parameter INOUT).

The programmer is responsible for synchronizing file accesses to a shared file for multiprocess MCS COBOL applications. The same mechanism can be used on a queue as "access" between several applications belonging to different steps provided that the queue has been defined with the SHARE option at CNC generation.

The stages in implementing file sharing are,

- At step execution, the program must prime the program queue ACCESS with a message which serves to notify the first process wishing to access the "shared file" that it is available.
- Process A issues a RECEIVE statement to the program queue ACCESS thru the symbolic queue FILE and 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 ACCESS which serves to notify Process B that the "shared file" is now available to be accessed.

COMMUNICATIONS BETWEEN REMOTE APPLICATIONS

The term "remote" implies that the programs are in different central processors which are linked thru a BSC (binary synchronous communication) line procedure. Communication is established at any given time by only 2 applications at either site by means of describing the other site as a BSC terminal with the following parameters (see TERMNL command in Section V),

- STYPE=CPU
- AUTO
- ASSIGN=program-queue

Communications between remote applications is described in terms of,

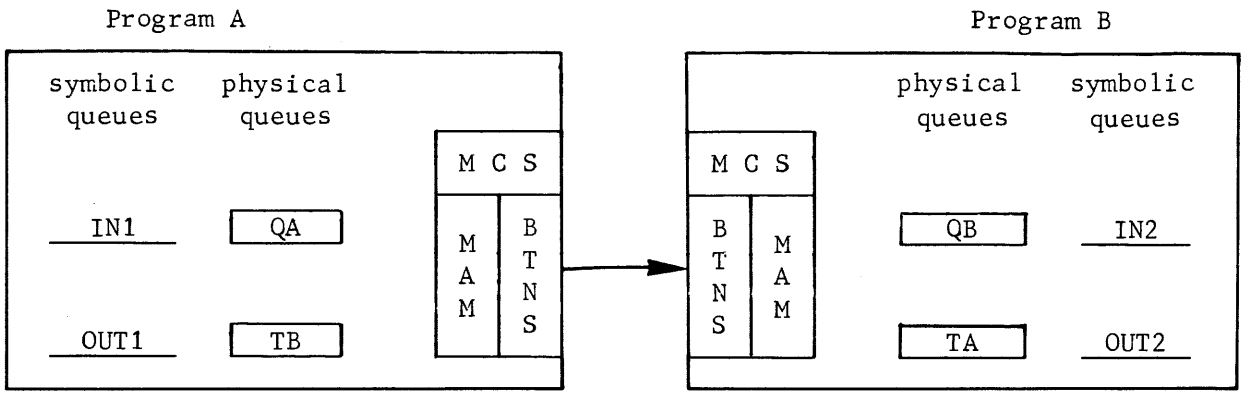
- Implementation
- Recovery on the Emission Side
- Recovery on the Reception Side

Implementation

The communications link-up is established at both sites by,

- The CNC description for the network
- The run-time JCL
- The sender and receiver applications.

Communication between Remote Applications



Program A at Site 1 is the sender.

Program B at Site 2 is the receiver.

CNC Description for BSC Link-up

Site 1

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

Site 2

```
LINE LNnn,CLOSE;
STATN STB2,2;
TERMNL TA,TYPE=HL64,STYPE=CPU,
       ASSIGN=QB,AUTO;
QUEUE QB,...,RESTART;
QUEUE TA,...,CTLRST;
```

In the CNC description of NDL commands, see Section V, the names of the program queues and terminal queues are the physical-queue-names. The lines described with CLOSE require both the site network operators to establish the link-up thru the command RT LNnn at the respective sites.

Run-time JCL

Site 1

```
QASSIGN IN1,QA,IN;
QASSIGN OUT1,TB,OUT;
```

Site 2

```
QASSIGN IN2,QB,IN;
QASSIGN OUT2,TA,OUT;
```

The $QASSIGN statement relates the symbolic queues to the physical queues.

The operation of the BSC link-up depends on the nature of the communication established and the recovery protocol :

- interactive communication : Exchanges of data will take place in the 2 directions successively from Program A to Program B and vice versa.
  The link-up must be established before the 2 applications are launched.

- unidirectional communication : As in the case of file transfer, the sender application can be launched first provided its output queue TB, in the preceding diagram, is on disk with enough space.

  In some cases the rate at which the output queue TB is being filled in by the sender Program A is greater than the rate at which the output queue is being emptied to the receiver Program B. When this happens as in file transfer from magnetic tape, the sender Program A will encounter "queue-overflow" incidents.
  To avoid such incidents, the sender program should issue a call to a system procedure to become temporarily suspended for a specified lapse of time before attempting another SEND to the output queue TB. Transfers from the output queue TB to Program B proceed unaffected.
  A "loop on the SEND statement has the effect of a temporary suspension, altho this method is not advised, since it takes up too much CPU time, see "Optimizing MCS COBOL Applications" later on in this section.

- recovery protocol : In the case of 2 remote applications where the link-up is used for a 1-way transfer, recovery protocol uses the system checkpoint and restart facility.
  A recovery mark is emitted by the sender application 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,
  - Program queues must be defined with the RESTART option
  - Terminal queues must be defined with the CTLRST option.

  Recovery protocol is defined in terms of
  - Recovery on the emission side
  - Recovery on the reception side.

Programming Normal Message Flow

call checkpoint
① test MOD
SEND ("><CTLCKP00") EMI

[before sending data, program A takes first checkpoint to send control message showing checkpoint numbered "00".]

MOD = 00

② SEND (data) {EMI | EGI}

[program A sends 1 or several files terminated by appropriate indicator.]

call checkpoint
③ test MOD
SEND ("><CTLCKPnn") EMI

[program A takes sequentially numbered checkpoints to send as control messages.]

MOD = 00

④ SEND (data) {EMI | EGI}

call checkpoint
⑤ test MOD
SEND ("><CTLCKPnn") EGI

[program A takes checkpoint; to indicate end of transmission, control message is sent with EGI.]

MOD = 00

⑥ end of normal transmission

ⓐ RECEIVE
call checkpoint

[first message received is checkpoint numbered "00"; program B takes its own first checkpoint; both checkpoints have initial value "00".]

ⓑ RECEIVE

[program B receives data.]

ⓒ RECEIVE
call checkpoint

[on receiving the checkpoint, program B takes its own checkpoint; numbers of both checkpoints match 1-to-1.]

ⓓ RECEIVE

ⓔ RECEIVE
call checkpoint

[on receiving the checkpoint with EGI, program B takes its final checkpoint.]

Recovery on the Emission Side

- calling checkpoints

  Each transmission is started with a checkpoint call. On return of the call, the output parameter MOD is tested for either normal transmission, MOD = 00, or restart after an abort, MOD ≠ 00.

  - using the H_CK_UCHKPT run-time package

1 In the DATA DIVISION declare as follows :

```
77 MOD COMP-2.
77 CKINF PICTURE X(32).
```

2 In the PROCEDURE DIVISION, code as follows :

```
CALL "H_CK_UCHKPT" USING MOD,CKINF.
```

- normal transmission, MOD = 00

  - image of control message at start of and during session

```
SEND (">><CTLCKPnn) EMI
```

, where nn is sequentially numbered from 00 thru 99.

  - image of control message to terminate session

```
SEND (">><CTLCKPnn) EGI
```

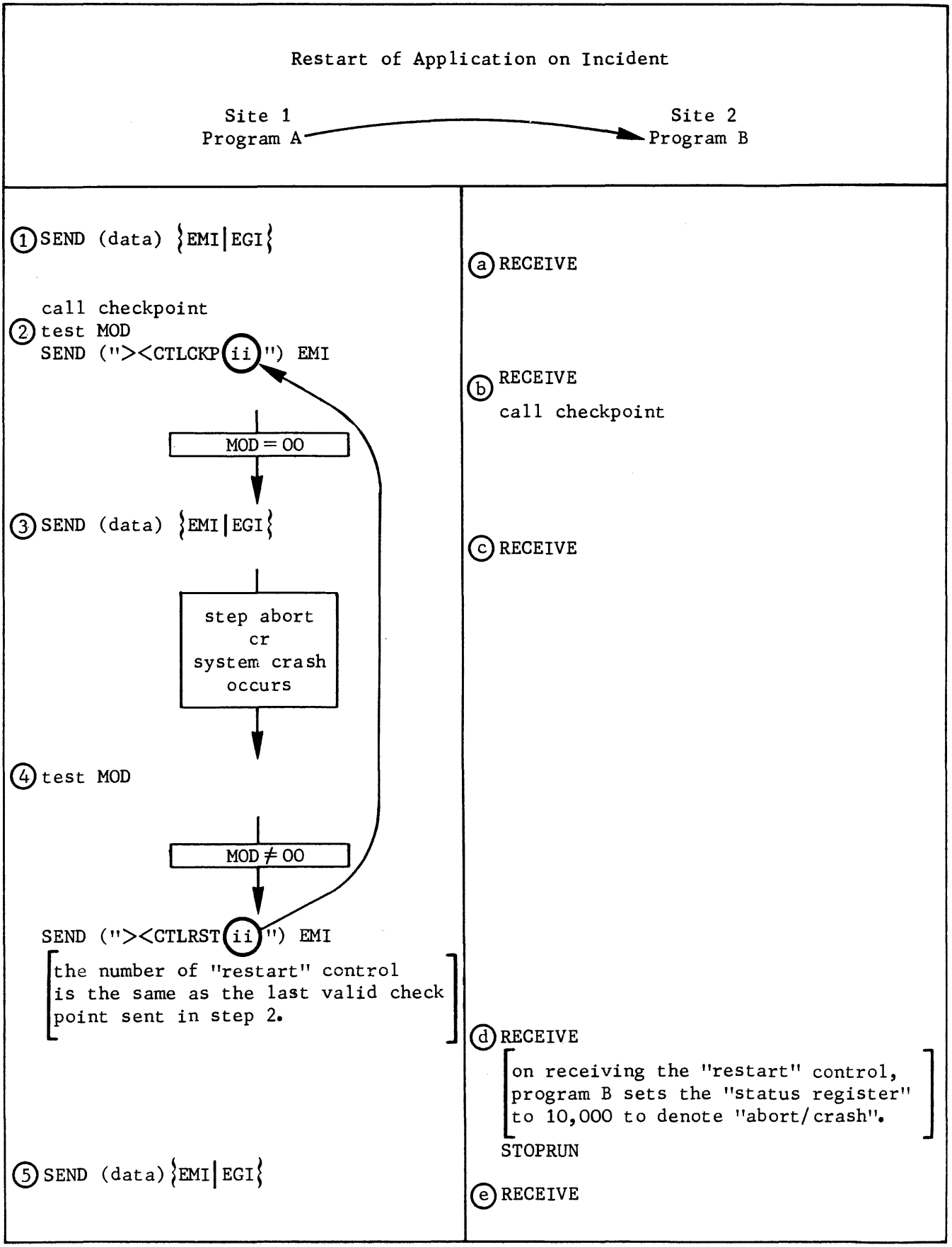
, for nn, see above

Control messages 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 MOD = 00 in order to continue normal transmission. The checkpoint is then sent as a control message to head the next stream of data flow. When the emission of a checkpoint has been completed, 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

$n = (QDBLKSZ - 4) \div 2$, where QDBLKSZ is the disk block size in bytes, see GENCOM command.

See "MAM Tuning" in Section V.


Restart of Application on Incident
Site 1
Program A
Site 2
Program B
1 SEND (data) {EMI | EGI}
call checkpoint
2 test MOD
SEND ("><CTLCKP ii ") EMI
MOD = 00
3 SEND (data) {EMI | EGI}
step abort
cr
system crash
occurs
4 test MOD
MOD ≠ 00
SEND ("><CTLRST ii ") EMI
the number of "restart" control is the same as the last valid check point sent in step 2.
5 SEND (data) {EMI | EGI}
a RECEIVE
b RECEIVE
call checkpoint
c RECEIVE
d RECEIVE
on receiving the "restart" control, program B sets the "status register" to 10,000 to denote "abort/crash".
STOPRUN
e RECEIVE

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

- image for transmission in transparent mode

```
SEND ("><CTLCKPnn") EMI
SEND ("><U06" data) EMI , ><U06 must head data text
          :
          :
SEND (data) {EMI|EGI}
SEND ("><CTLCKPnn") EGI , to terminate the session
```

- restart after an abort/crash, MOD≠00

  - image of restart control message

```
SEND ("><CTLRSTnn") EMI
```

, where nn is the number of the last valid checkpoint sent.

The "restart" control message can be emitted from 2 sources,
- the sender application in the case of a step abort
- 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-sent.

If at the time of system restart, the sender application is also to be re-started, then the receiver application will see 2 restart phases in the following sequence,
- firstly, the restart control message from the system
- then the restart control message from the sender application.

- restart 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.

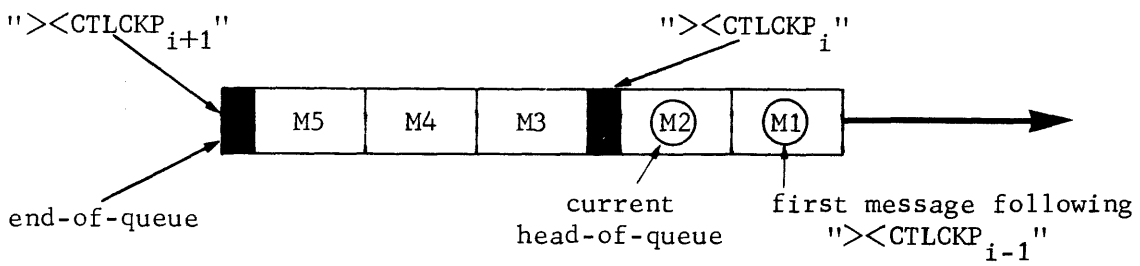
The line can be re-opened by the network control command of the format, RT LNnn, where nn specifies the line.

The effects of this command on a BSC line are,
- the line will be re-activated
- 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.

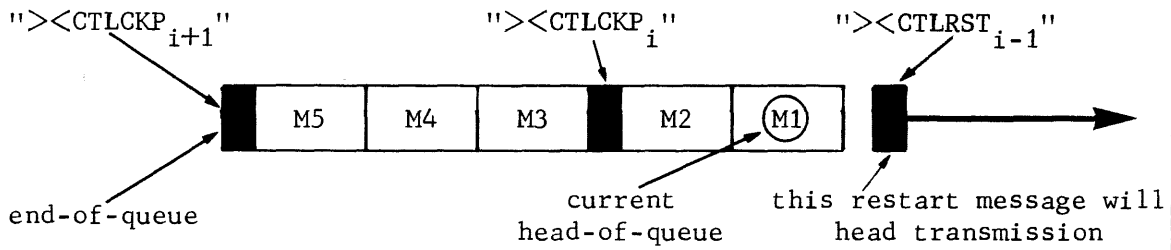
Transmission will resume on successful emission of the restart control message. Operator intervention at both sites may be required.

Sender Application Absent

Situation at Failure

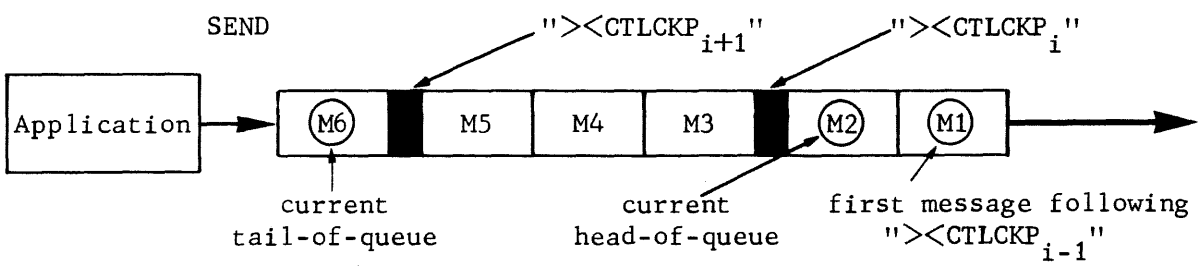


Situation at Restart

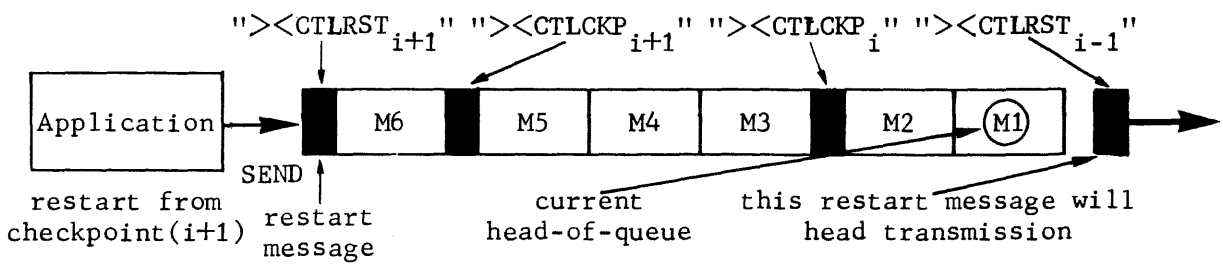


Sender Application Present

Situation at Failure



Situation at Restart



Recovery on the Reception Side

- calling checkpoints

  Checkpoints are called using the H_CK_UCHKPT run-time package, see "Recovery on the Emission Side".

  When the receiver application receives the control message "><CTLCKPnn", it calls a checkpoint. The checkpoint called has the same sequence number as the checkpoint from the sender application that triggered it. No control message is generated by the receiver application. The checkpoint serves as a recovery mark.

- reception of restart message

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

  The receiver application performs the following decisions on the sequence number "ii" of the "restart" control message, namely,

  - If ii = nn + 1, where nn is the sequence number of the last checkpoint called by the receiver application, then the "restart" control message is treated as a checkpoint and the response of the receiver application is to call its own checkpoint and continue with normal transmission.
  - If ii = nn, then the receiver application must restart from its current checkpoint.
    To do this, the application sets the STATUS register to an abnormal value, say 10,000, and then issues a STOPRUN.
    The application will terminate abnormally, and the system operator will have to restart the application.
  - using the H_CBL_USETST run-time package for setting the STATUS register

1 In the DATA DIVISION declare as follows :

```
77 STATUS COMP-1.
```

2 In the PROCEDURE DIVISION, code as follows :

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

- application restart

  The application is restarted from its previous checkpoint.

  Restart does not affect the sender application.

  The following events take place on system restart after a crash,

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

  Operator intervention at both sites may be required to re-activate the line.

EXECUTING COMMUNICATIONS APPLICATIONS

Preparing and running communications applications involves,

- Preallocating a disk queue file
- Linking MCS COBOL programs
- Executing a communications step
- Optimizing MCS COBOL applications.

Preallocating a Disk Queue File

If queues are to be held on a disk, a file must be preallocated on a disk. Disk queueing is the default option of the QUEUE command, see Section V.

Preallocating a disk file is performed

- Using the extended JCL statement $PREALLOC, see Data Management Utilities Manual
- Before H_CNC utility is executed since the disk queue file must be preformatted as a result of network generation.

If the file is to be modified, e.g., altering the file size, the following procedure is performed in the sequence given,

- The file is deallocated by either method,
  - using the extended JCL statement $DEALLOC
  - declaring MAM=NO at system initialization
- It is then preallocated using $PREALLOC
- H_CNC utility must then be run 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.

Example of Preallocating a Disk Queue File

```
$JOB ALLOCDQ,USER=UNAME,PROJECT=WAGE;
PREALLOC DQUEUE,EXPDATE=365,DEVCLASS=MS/M400,
         GLOBAL=(MEDIA=VOL1,SIZE=5),
         BFAS=(SEQ=(BLKSIZE=500,RECSIZE=500));
$ENDJOB;
```

Syntax concerning parameters in $PREALLOC :

- The external file name DQUEUE must be specified in a $ASSIGN statement at H_CNC step execution.
- The size of the file specified in the example as 5 cylinders is subject to limits of the device class, see Section V.
- 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)) should be specified as indicated.

Linking MCS COBOL Programs

An MCS COBOL program is compiled using the standard $COBOL statement, see COBOL User Guide.
Compile units (cu) of the MCS COBOL program are linked by the $LINKER statement to form communications load modules.

Linking an MCS COBOL Program

```
$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

```
$INPUT input-enclosure-name;

ENTRY=STUSERS,
STARTASG=(PRIVATE=8,SHARE=A),
LINKTYPE=BMAM,
```

to enter & start MAM for executing calls thru user-specified entry points

```
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),
```

TASK & REPLACE to be specified for each user process

```
$ENDINPUT;

$ENDJOB;
```

Syntax for JCL used to link an MCS COBOL program :

- $LIB statement
  - specifies the system library SYS.HCULIB containing the MAM run-time package
  - specifies the libraries containing the user compile units (cu) to be linked
  - describes the search path to the Static Linker
  - for details of parameters, see Library Management Manual.
- $LINKER statement
  - 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 program; the default entry point is the load-module-name.
    - 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
  - ENTRY, STARTASG and LINKTYPE must be entered exactly as indicated with the parameters given.
  - TASK statement
    - USERn is used by MAM system procedure to start the indicated process in the order from USER1 thru USER6, where appropriate.
  - REPLACE statement
    - cu-namen defines the entry point in the program which is to be linked to the process name USERn in the corresponding TASK statement.

Example of Linking a Monoprocess Load Module

```
$JOB LINK1,USER=UNAME,PROJECT=ACCT;
LIB CU,INLIB1=SYS.HCULIB,
       INLIB2=(UCULIB,DEVCLASS=MS/M400,MEDIA=VOL1);
LINKER MAM1,
       OUTLIB=(ULMLIB,DEVCLASS=MS/M400,MEDIA=VOL1),
       COMFAC;
$ENDJOB;
```

Syntax in example of linking a monoprocess load module :

- $LIB statement
  - UCULIB is the compile unit (cu) library on the volume VOL1 containing the user program.
- $LINKER statement
  - MAM1 is the load-module-name and is also the entry point to the user program.
  - ULMLIB is the user load module library on the volume VOL1 in which the load module is to be stored.

Example of Linking a Multiprocess Load Module

```
$JOB LINK2,USER=UNAME, PROJECT=WAGE;

LIB CU,INLIB1=SYS.HCULIB,
       INLIB2=(UCULIB1,DEVCLASS=MS/M400,MEDIA=VOL1),
       INLIB3=(UCULIB2,DEVCLASS=MS/M400,MEDIA=VOL1);

LINKER MAM2,
       OUTLIB=(ULMLIB,DEVCLASS=MS/M400,MEDIA=VOL1),
       COMFILE=*LINK;

$INPUT LINK;

ENTRY=STUSERS,
STARTASG=(PRIVATE=8,SHARE=A),
LINKTYPE=BMAM,

TASK=(USER1,START=STUSER1),
TASK=(USER2,START=STUSER2),

REPLACE=(DUMMY,DATCOLL,CU=STUSER2),
REPLACE=(DUMMY,INQRESP,CU=STUSER1),

$ENDINPUT;

$ENDJOB;
```

Syntax in example of linking a multiprocess load module :

- $LIB statement
  - UCULIB1 and UCULIB2 are the compile unit libraries on the volume VOL1 containing the user programs.
- $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)
  - REPLACE statement
    - DATCOLL is an entry point to the user program which is to be linked to the process USER2.
    - INQRESP is an entry point to the user program which is to be linked to the process USER1.

Executing a Communications Step

A communications step is executed like any other GCOS job step. The only difference is the $QASSIGN statement which is required in the JCL for the MCS COBOL step. Points to note in specifying run-time JCL with $QASSIGNs are :

- Any other output queues used by the application in the example given in $QASSIGN are implicitly specified.
- In the case where the application uses disk queues, no $ASSIGN statement is required for the disk queue file because the file is opened at system level and is therefore sharable by all process groups.

Definition

The $QASSIGN statement performs the following,

- Defines the symbolic queues and subqueues, where applicable, used by the program
- Establishes a correspondence between the symbolic queue or subqueue and the external-queue-name specified at CNC generation, see Section V
- Identifies the processing mode of the queue with respect to "send" and "receive" actions
- Allocates queues (input and optionally, output) to the step.

Format of Statement

```
QASSIGN symbolic-queue-name [.symbolic-subqueue-name] ,external-queue-name-1
    [, { IN    [,ACCESS = {LIN}]}] [,REPLY = external-queue-name-2] ;
       { INOUT            {RR }}
       { OUT  }
```

Parameter Descriptions

external-queue-name
- ranges from 1 thru 4 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command, see Section V.

symbolic-queue-name
- the name of the logical queue defined in data-name-1 of the CD area of the MCS COBOL program.

symbolic-subqueue-name
- applicable only in the case of input queues and is the name given to the partitioning of the logical queue defined in data-name-2 of the input CD area of the MCS COBOL program.

ACCESS
- applicable only to input queues partitioned into subqueues and specifies the way in which the subqueues are to be scanned in a "receive" action, as follows,
  - LIN : on each RECEIVE statement to the queue, the search begins with the first subqueue specified, i.e., linear, which is the default value.
  - RR : on each RECEIVE statement to the queue, the search resumes at the subqueue immediately following the subqueue that provided input data for the last RECEIVE statement issued, i.e., round-robin.

IN
- applicable only to program queues and allocates the queue to the step in input processing mode, which is the default value.

INOUT
- applicable only to program queues and must not be specified for subqueues. Allocates the queue to the step in input and output processing modes.

OUT
- must not be specified for subqueues; allocates the queue to the step in output processing mode.

REPLY
- applicable only for application-to-application communications and must be specified with either INOUT or OUT modes, see "Usage Rules", overleaf.

$QASSIGN
continued

Usage Rules

1. The order of the list of subqueues within a symbolic input queue is determined by the order of $QASSIGN statements within the step enclosure.
2. The maximum number of $QASSIGN statements permitted within the step enclosure is 24.
3. The ACCESS parameter is only relevant in the first $QASSIGN statement defining a queue structure, i.e., subqueues within the queue.
4. A $QASSIGN statement is mandatory for each input queue associated with the program.
   For output queues, the $QASSIGN statement is optional.
5. REPLY defines another program queue which is identified as the source of input messages in the destination application's input CD.
   The REPLY facility enables the destination application to answer back, using in its turn the program queue thus defined as "destination".

Example of Run-time JCL with $QASSIGNs

```
$JOB TELECOM,USER=UNAME,PROJECT=ACCT,CLASS=H;
STEP MAM3,(ULMLIB,DEVCLASS=MS/M400,MEDIA=VOL1);
```

group a	QASSIGN INQ.SUB1,PRG1,ACCESS=RR; QASSIGN INQ.SUB2,PRG2; QASSIGN INQ.SUB3,PRG3; QASSIGN INQ.SUB4,PRG4;	these 4 $QASSIGNs identify a queue structure for symbolic-input-queue INQ
group b	QASSIGN OUTQ,TRM1,OUT; QASSIGN OUTW,TRM2,OUT;	relates symbolic-output-queues to terminal-queues

```
ENDSTEP
$ENDJOB;
```

Syntax in example of run-time JCL with $QASSIGNs :

- $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 statements
  - group a : each symbolic-subqueue corresponds to a unique external-queue-name, i.e., SUBn to PROGn, specified during 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 TERM1 and TERM2.

BT PAYR END

Optimizing MCS COBOL (MAM) Applications

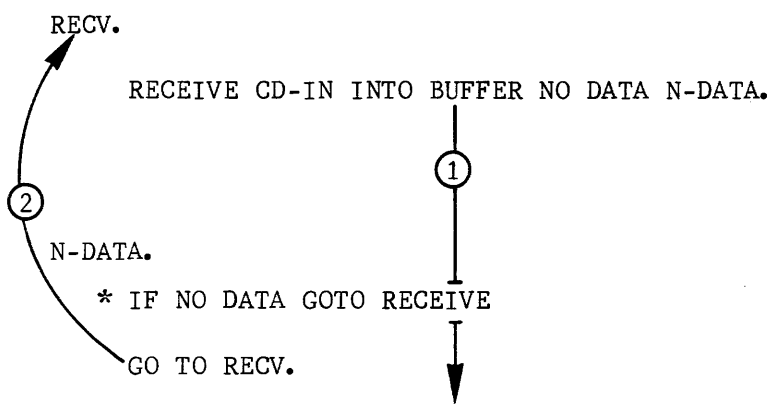
General recommendations for optimizing MCS COBOL programming include,

- using the RECEIVE verb (not qualified by the NO DATA clause)

  One of the 2 following conditions will occur,

  - If at least 1 message is queued, then it will be delivered into the user specified working area, and control is returned to the application program.
  - If the queue is empty, MAM suspends the application program until a message arrives in the queue.
    When a message arrives, it is delivered into the user specified working area, and control is then returned to the application program.

- using the RECEIVE verb with NO DATA clause



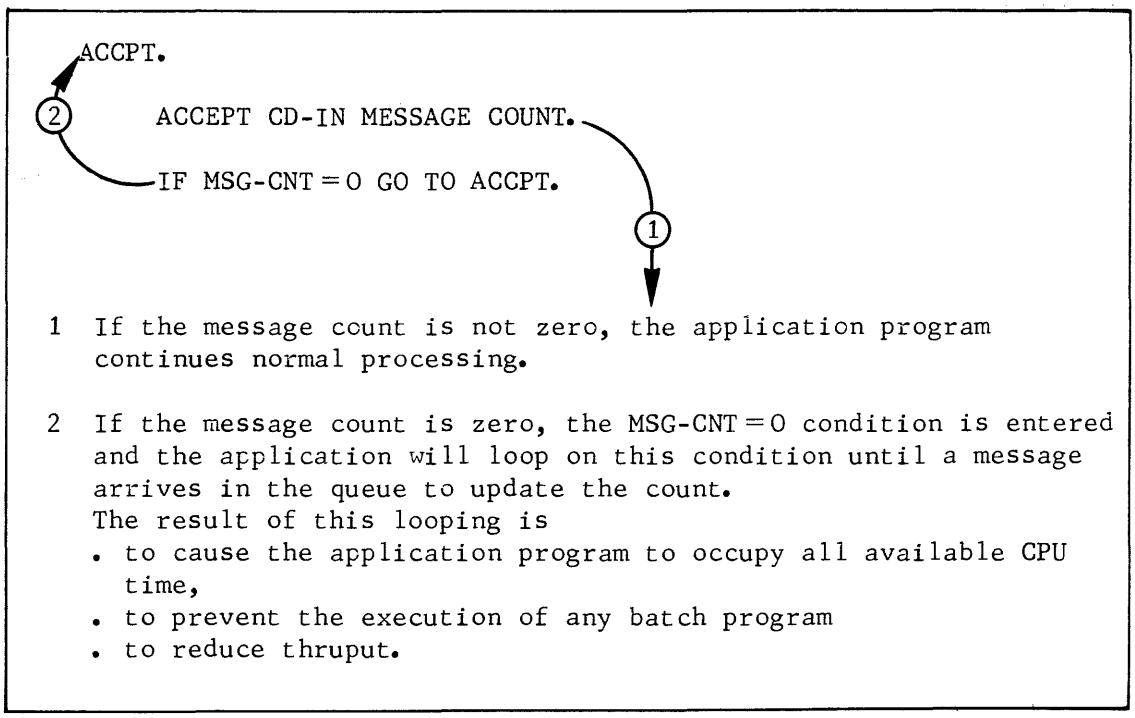
1 If a message is available in the queue, it is delivered into the user specified working area,
- control is returned to the application program, and
- the NO DATA condition is bypassed.

2 If no message is available, the NO DATA condition entered and the application will loop on this condition until a message arrives in the queue, thereby causing the application program
- to occupy all the CPU time not used by BTNS or the system, and therefore
- to prevent the execution of any batch program, and consequently
- to reduce thruput.

RECEIVE with NO DATA should be used under the following conditions,

- To scan a queue while executing another process or while waiting for the occurrence of other events
- To include in the NO DATA loop a call to the run-time package H_TM_USETTM to allow the application program to be suspended for a user specified time before being re-activated.

. using the ACCEPT verb



1 If the message count is not zero, the application program continues normal processing.

2 If the message count is zero, the MSG-CNT = 0 condition is entered and the application will loop on this condition until a message arrives in the queue to update the count.
The result of this looping is
- to cause the application program to occupy all available CPU time,
- to prevent the execution of any batch program
- to reduce thruput.

ACCEPT should be used under the following conditions,

- To ascertain the number of messages in a queue while executing another process or while waiting for the occurrence of other events
- To include in the loop, in the example given, the MSG-CNT = 0 condition, a call to the run-time package H_TM_USETTM to allow the application program to be suspended for a user specified time before being re-activated.

. using the H_TM_USETTM run-time package

1 In the DATA DIVISION declare as follows :

```
01 T COMP-2.
```

, where T is specified in milliseconds

2 In the PROCEDURE DIVISION, as in the loop, code as follows :

```
CALL "H_TM_USETTM" USING T.
```

- handling several program queues

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

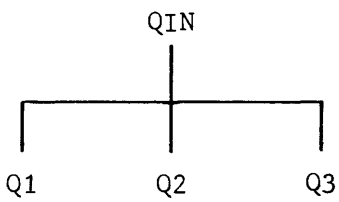
  In some cases, the application program may have to handle several program queues if
  - an order of priority is to be established between more than 1 source of messages so that one particular source can be selected at a given time
  - the application program is to communicate with a complex of terminals and with either or both,
    - ° a process of the same application
    - ° another application program.

  Scanning the different program queues:
  Consider 3 program queues, Q1, Q2 and Q3 in which the application is to receive messages. Q1 has the lowest priority, Q3 has the highest.

  Solution 1 : The application scans each queue in the order of priority indicated, i.e., first Q3, then Q2 and finally Q1, by
  - ° looping on RECEIVE NO DATA successively for each queue
  - ° including in each loop a "wait-time" using the facility H_TM_USETTM.

  The drawbacks of this solution are,
  - ° if the "wait-time" is too large, then the "turn-around" time will be too slow
  - ° if the "wait-time" is too small, then the application will loop on itself until a message became available, which would defeat the purpose of using H_TM_USETTM.

  Solution 2 : A queue structure can be defined thru the $QASSIGN statement in the run-time JCL in the following way,



  In this case, the application program only has to issue a RECEIVE (without the NO DATA clause) to the queue QIN, without having to specify the individual "sub-queues".

  The advantages of this solution are,
  - ° if no message is available in any of the "sub-queues", MAM suspends the application, and processing will only resume as soon as a message arrives in any of the "sub-queues".
  - ° the RECEIVE command will indicate in the input CD from which "sub-queue" the message has been received, and the application can then place whatever priority it wants on processing it.

SECTION IV

MCS DATA FORMATS

This section deals with the format of data sent to or received from a terminal by an MCS COBOL application program. The data is held in the working area defined as the user buffer and accessed by the SEND and RECEIVE verbs.

Data comprises,

- Information data composed of characters corresponding to graphic symbols as,
  - numeric characters
  - alphabetic characters, both upper and lower case
  - special characters, including punctuation and mathematical symbols.
- Control characters providing terminal management functions as,
  - editing or service functions, including carriage-return, tabulation, cursor positioning, etc.,
  - auxiliary device commands, such as for the cassette handler
  - timing functions, including "fill" characters to allow for slow mechanical functions on any printer device
  - message delimiters, including trailers and VIP headers.

The control characters are generated by
- control touch-keys on the device, used either individually or in combination
- a sequence of encoded characters.

MARK REPRESENTATION

Mark representation is a symbolic representation of data at user program level provided by MCS and uses the symbols >< to prefix
- VIP headers
- Control characters
- Character-encoding
- Repeat

VIP Headers in Mark Form

Format : ><U03abc, where a = status, b = function-code-1, c = function-code-2

Control Characters in Mark Form

Format : ><ccc, where ccc = 3-letter control character symbolic representation, see "Control Characters and EBCDIC Values"

Character-encoding

Format : ><Cab, where ab = 2-digit hexadecimal value corresponding to any character including a control function, see "EBCDIC Values & Control Characters"

Repeat

Format : ><Rnn, where nn = 2-digit decimal value denoting that the following character or hexadecimal value is to be repeated for as many times as specified.

TRANSMISSION MODES

Transmission from the terminal is in ASCII code, altho in the case of the BSC line procedure, such as BSC2780 terminals, transmission may be in EBCDIC code which is the Level 64 internal code.

Translation from ASCII code to EBCDIC, and vice versa, is performed in the IURP or URP by means of the TCT (translation control table).

The user buffer will contain the EBCDIC values of data input from the terminal or to be output to the terminal.

The transmission modes are

- Input Normal Mode
- Input Marked Mode
- Input Unedited Mode
- Output Normal Mode
- Output Unedited Mode

In the examples following, the conventions used in the data formats are,

- For hexadecimal values, specified in internal EBCDIC code, double quotation marks " " are used for enclosure
- For graphic symbols, single-letter alphanumeric values are used
- For control characters keyed in on the device or sent to the device, the 3-letter code is used
- Divisions within the buffer denote the value weighting of 1-byte

MARK mode

Control Characters & EBCDIC Values											
Control Character		Hex. Code	Line Procedure			Control Character		Hex. Code	Line Procedure		
			TTY	VIP	BSC				TTY	VIP	BSC
ACK	ACKnowledge	2E	I/O	I	I	ETB	End of Transmission Block	26	I/O	I	I
BO1	advance 1 page see also FF▽	OC	O	O	O	ETX	End of TeXt	03	I/O	I	I
BO3	advance 1 line see also CR▽ LF▽ NL▽	OD25	O	O	O	FF▽	Form Feed see also BO1	OC	I/O	I/O	I/O
BEL	BEL1	2F	I/O	na	I/O	FS▽	File Separator	1C	I/O	I/O	I/O
	VIP terminals, BEL generates 5F	5F	na	I/O	na	GS▽	Group Separator	1D	I/O	I/O	I/O
BLK	BLinK	5F	I/O	I/O	I/O	HT▽	Horizontal Tab	05	I/O	I/O	I/O
	VIP terminals, see also BEL					LF▽	Line Feed	25	I/O	I/O	I/O
BS▽	BackSpace	16	I/O	I/O	I/O	NAK	Negative AcKnowledge	3D	I/O	I	I
CAN	CANcel	18	I/O	I	I	NL▽	New Line see also BO3 CR▽ LF▽	OD25	I/O	I/O	I/O
CR▽	Carriage Return	OD	I/O	I/O	I/O						
DC1	Device Control 1	11	I/O	I/O	I/O	NUL	NUL1	00	I/O	I/O	I/O
DC2	Device Control 2	12	I/O	I/O	I/O	RS▽	Record Separator	1E	I/O	I	I
	forward space (VIP terminals)					SI▽	Shift In	OF	I/O	I	I
DC3	Device Control 3	13	I/O	I/O	I/O	SO▽	Shift Out	OE	I/O	I	I
DC4	Device Control 4	3C	I/O	I/O	I/O	SOH	Start Of Header	01	I/O	I	I
DEL	DELete	07	I/O	I/O	I/O	SP▽	SPace	40	I/O	I/O	I/O
DLE	Data Link Escape	10	I/O	I	I	STX	Start of TeXt	02	I/O	I	I
EM▽	End of Medium	19	I/O	I/O	I/O	SUB	SUBstitute	3F	I/O	I	I
ENQ	ENQuiry	2D	I/O	I	I	SYN	SYNchronous idle	32	I/O	I	I
EOT	End Of Transmission	37	I/O	I	I	US▽	Unit Separator	1F	I/O	I	I
ESC	ESCape	27	I/O	I/O	I/O	VT▽	Vertical Tab	OB	I/O	I	I

EBCDIC Values & Control Characters

Hex. Code	Control Character
00	NUL NULl
01	SOH Start Of Header
02	STX Start of TeXt
03	ETX End of TeXt
04	not reserved
05	HT∇ Horizontal Tab
06	not reserved
07	DEL DELete
08	not reserved
09	not reserved
0A	not reserved
0B	VT∇ Vertical Tab
0C	BO1 advance 1 page FF∇ Form Feed
0D	CR∇ Carriage Return
0D25	BO3 advance 1 line NL∇ New Line
0E	SO∇ Shift Out
0F	SI∇ Shift In
10	DLE Data Link Escape
11	DC1 Device Control 1
12	DC2 Device Control 2
13	DC3 Device Control 3
14	not reserved
15	not reserved
16	BS∇ BackSpace
17	not reserved
18	CAN CANcel
19	EM∇ End of Medium
1A	not reserved
1B	not reserved
1C	FS∇ File Separator
1D	GS∇ Group Separator
1E	RS∇ Record Separator
1F	US∇ Unit Separator
20	not reserved
21	not reserved
22	not reserved
23	not reserved
24	not reserved
25	LF∇ Line Feed
26	ETB End of Transmission Block
27	ESC ESCape
28	not reserved
29	not reserved
2A	not reserved
2B	not reserved
2C	not reserved
2D	ENQ ENQuiry
2E	ACK ACKnowledge
2F	BEL BELl
30	not reserved
31	not reserved
32	SYN SYNchronous idle
33	not reserved
34	not reserved
35	not reserved
36	not reserved
37	EOT End Of Transmission
38	not reserved
39	not reserved
3A	not reserved
3B	not reserved
3C	DC4 Device Control 4
3D	NAK Negative AcKnowledge
3E	not reserved
3F	SUB SUBstitute
40	SP∇ SPace
41	not reserved
↓	not reserved
4F	not reserved
50	not reserved
↓	not reserved
5E	not reserved
5F	BLK BLinK (for VIP only)
6h	not reserved
7h	not reserved
8h	not reserved
9h	not reserved
Ah	not reserved
Bh	not reserved
Ch	not reserved
Dh	not reserved
Eh	not reserved
Fh	not reserved

* h signifies any hexadecimal unit

EBCDIC Character Set & COBOL Collating Sequence

Explanation of abbreviations :

- EBCDIC - EBCDIC hexadecimal value
- graphic - graphic symbol
- COBOL - COBOL collating sequence

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

Translation from EBCDIC to ASCII

EBCDIC	ASCII
00	00
01	01
02	02
03	03
04	9C
05	09
06	86
07	7F
08	97
09	8D
0A	8E
0B	0B
0C	0C
0D	0D
0E	0E
0F	0F
10	10
11	11
12	12
13	13
14	9D
15	85
16	08
17	87
18	18
19	19
1A	92
1B	8F
1C	1C
1D	1D
1E	1E
1F	1F
20	80
21	81
22	82
23	83
24	84
25	0A
26	17
27	1B
28	88
29	89
2A	8A

EBCDIC	ASCII
2B	8B
2C	8C
2D	05
2E	06
2F	07
30	90
31	91
32	16
33	93
34	94
35	95
36	96
37	04
38	98
39	99
3A	9A
3B	9B
3C	14
3D	15
3E	9E
3F	1A
40	20
41	A0
42	A1
43	A2
44	A3
45	A4
46	A5
47	A6
48	A7
49	A8
4A	5B
4B	2E
4C	3C
4D	28
4E	2B
4F	21
50	26
51	A9
52	AA

EBCDIC	ASCII
53	AB
54	AC
55	AD
56	AE
57	AF
58	B0
59	B1
5A	5D
5B	24
5C	2A
5D	29
5E	3B
5F	5E
60	2D
61	2F
62	B2
63	B3
64	B4
65	B5
66	B6
67	B7
68	B8
69	B9
6A	7C
6B	2C
6C	25
6D	5F
6E	3E
6F	3F
70	BA
71	BB
72	BC
73	BD
74	BE
75	BF
76	C0
77	C1
78	C2

EBCDIC	ASCII
79	60
7A	3A
7B	23
7C	40
7D	27
7E	3D
7F	22
80	C3
81	61
82	62
83	63
84	64
85	65
86	66
87	67
88	68
89	69
8A	C4
8B	C5
8C	C6
8D	C7
8E	C8
8F	C9
90	CA
91	6A
92	6B
93	6C
94	6D
95	6E
96	6F
97	70
98	71
99	72
9A	CB
9B	CC
9C	CD

EBCDIC	ASCII
9D	CE
9E	CF
9F	D0
A0	D1
A1	7E
A2	73
A3	74
A4	75
A5	76
A6	77
A7	78
A8	79
A9	7A
AA	D2
AB	D3
AC	D4
AD	D5
AE	D6
AF	D7
B0	D8
B1	D9
B2	DA
B3	DB
B4	DC
B5	DD
B6	DE
B7	DF
B8	E0
B9	E1
BA	E2
BB	E3
BC	E4
BD	E5
BE	E6

EBCDIC	ASCII
BF	E7
C0	7B
C1	41
C2	42
C3	43
C4	44
C5	45
C6	46
C7	47
C8	48
C9	49
CA	E8
CB	E9
CC	EA
CD	EB
CE	EC
CF	ED
D0	7D
D1	4A
D2	4B
D3	4C
D4	4D
D5	4E
D6	4F
D7	50
D8	51
D9	52
DA	EE
DB	EF
DC	F0
DD	F1

EBCDIC	ASCII
DE	F2
DF	F3
E0	5C
E1	9F
E2	53
E3	54
E4	55
E5	56
E6	57
E7	58
E8	59
E9	5A
EA	F4
EB	F5
EC	F6
ED	F7
EE	F8
EF	F9
F0	30
F1	31
F2	32
F3	33
F4	34
F5	35
F6	36
F7	37
F8	38
F9	39
FA	FA

EBCDIC	ASCII
FB	FB
FC	FC
FD	FD
FE	FE
FF	FF

Translation from ASCII to EBCDIC

ASCII	EBCDIC
00	00
01	01
02	02
03	03
04	37
05	2D
06	2E
07	2F
08	16
09	05
0A	25
0B	0B
0C	0C
0D	0D
0E	0E
0F	0F
10	10
11	11
12	12
13	13
14	3C
15	3D
16	32
17	26
18	18
19	19
1A	3F
1B	27
1C	1C
1D	1D
1E	1E
1F	1F
20	40
21	4F
22	7F
23	7B
24	5B
25	6C
26	50
27	7D
28	4D
29	5D
2A	5C
2B	4E
2C	6B
2D	60
2E	4B
2F	61
30	F0
31	F1
32	F2
33	F3
34	F4
35	F5
36	F6
37	F7
38	F8
39	F9
3A	7A
3B	5E
3C	4C
3D	7E
3E	6E
3F	6F
40	7C
41	C1
42	C2
43	C3
44	C4
45	C5
46	C6
47	C7
48	C8
49	C9
4A	D1
4B	D2
4C	D3
4D	D4
4E	D5
4F	D6
50	D7
51	D8
52	D9
53	E2
54	E3
55	E4
56	E5
57	E6
58	E7
59	E8
5A	E9
5B	4A
5C	E0
5D	5A
5E	5F
5F	6D
60	79
61	81
62	82
63	83
64	84
65	85
66	86
67	87
68	88
69	89
6A	91
6B	92
6C	93
6D	94
6E	95
6F	96
70	97
71	98
72	99
73	A2
74	A3
75	A4
76	A5
77	A6
78	A7
79	A8
7A	A9
7B	C0
7C	6A
7D	D0
7E	A1
7F	07
80	20
81	21
82	22
83	23
84	24
85	15
86	06
87	17
88	28
89	29
8A	2A
8B	2B
8C	2C
8D	09
8E	0A
8F	1B
90	30
91	31
92	1A
93	33
94	34
95	35
96	36
97	08
98	38
99	39
9A	3A
9B	3B
9C	04
9D	14
9E	3E
9F	E1
A0	41
A1	42
A2	43
A3	44
A4	45
A5	46
A6	47
A7	48
A8	49
A9	51
AA	52
AB	53
AC	54
AD	55
AE	56
AF	57
B0	58
B1	59
B2	62
B3	63
B4	64
B5	65
B6	66
B7	67
B8	68
B9	69
BA	70
BB	71
BC	72
BD	73
BE	74
BF	75
C0	76
C1	77
C2	78
C3	80
C4	8A
C5	8B
C6	8C
C7	8D
C8	8E
C9	8F
CA	90
CB	9A
CC	9B
CD	9C
CE	9D
CF	9E
D0	9F
D1	A0
D2	AA
D3	AB
D4	AC
D5	AD
D6	AE
D7	AF
D8	B0
D9	B1
DA	B2
DB	B3
DC	B4
DD	B5
DE	B6
DF	B7
E0	B8
E1	B9
E2	BA
E3	BB
E4	BC
E5	BD
E6	BE
E7	BF
E8	CA
E9	CB
EA	CC
EB	CD
EC	CE
ED	CF
EE	DA
EF	DB
F0	DC
F1	DD
F2	DE
F3	DF
F4	EA
F5	EB
F6	EC
F7	ED
F8	EE
F9	EF
FA	FA
FB	FB
FC	FC
FD	FD
FE	FE
FF	FF

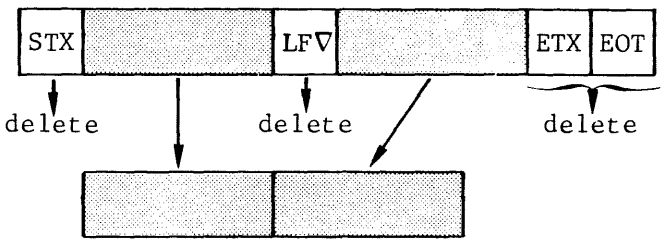
Input Normal Mode

The input normal mode is entered when

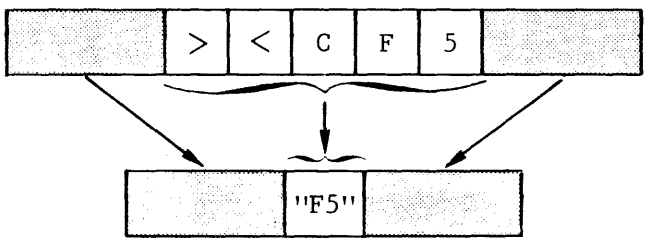
- the TERMNL command is not specified with the IM parameter, i.e., "normal" is the default input mode
- the TERMNL command is specified with the parameter IM=NL
- the network command MTE specifying the terminal and the parameter INORM is keyed in
- the terminal operator command $*$MTE specifying INORM is issued.

Treatment of data in the normal mode is as follows,

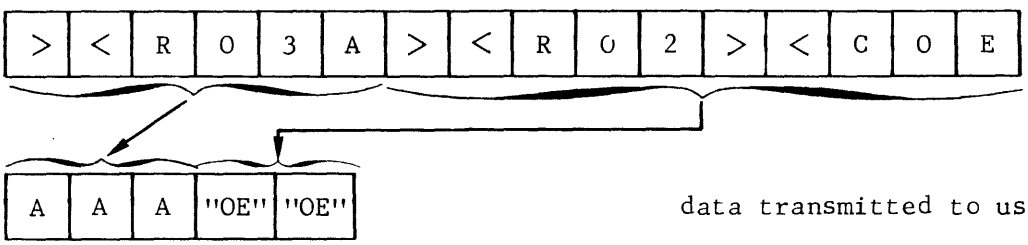
1. Headers, trailers and control characters are deleted from the user buffer.



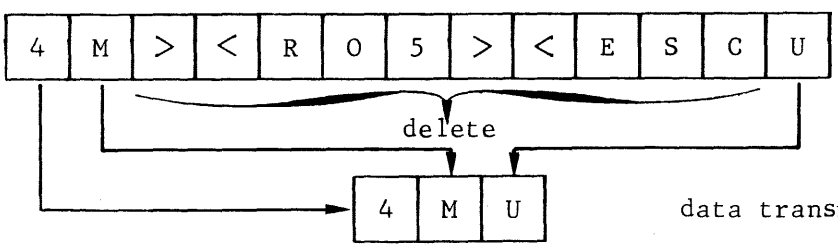
2. The character-encoding mark >< C is not transmitted; it passes the following 2 hexadecimal digits as a 1-byte hexadecimal value to the user buffer.



3. The repeat mark >< R is not transmitted; it repeats the following character or hexadecimal value as many times as specified.



4. The sequence of the repeat mark followed immediately by a control character mark is deleted from the user buffer.



5. All other characters including the marks >< are passed to the user buffer.

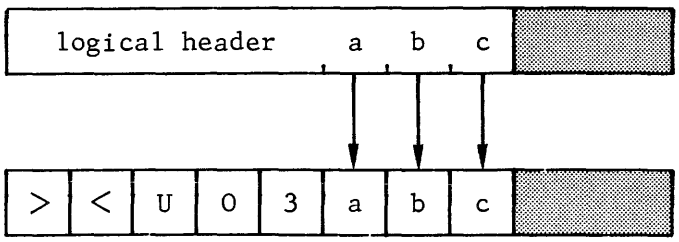
Input Marked Mode

The input marked mode is entered when

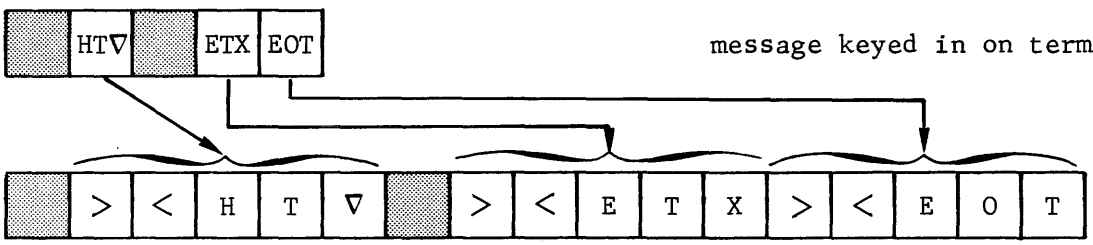
- the TERMNL command is specified with the parameter IM=MK
- the network command MTE specifying the terminal and the parameter IMARK is keyed in
- the terminal operator command $*$MTE specifying IMARK is issued.

Treatment of data in the marked mode is as follows,

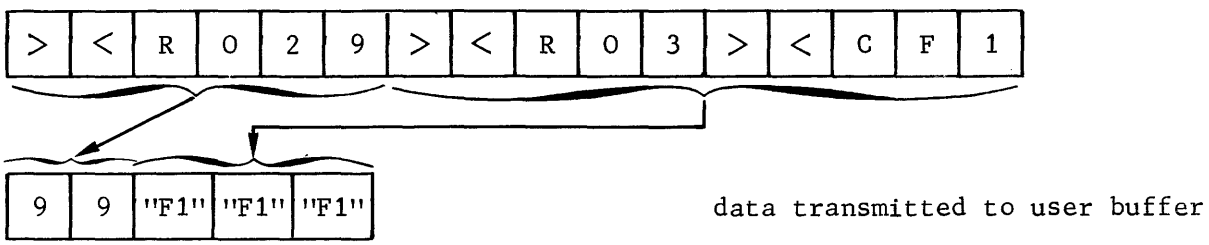
1. VIP headers are translated into mark form and passed to the user buffer.



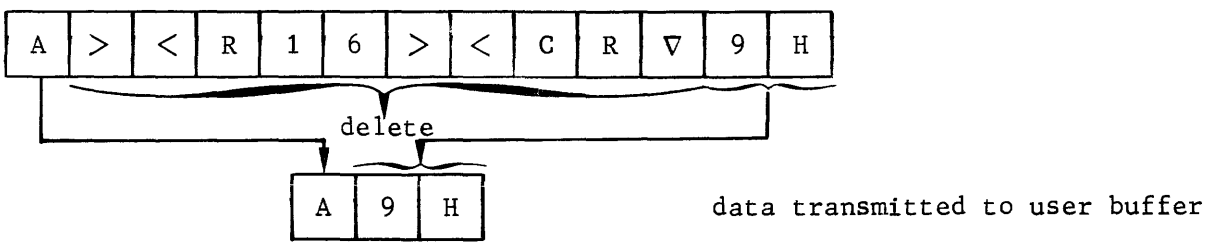
2. Control characters and trailers are translated into mark form and passed to the user buffer.



3. The character-encoding mark ><C is not transmitted; it passes the following 2 hexadecimal digits as a 1-byte hexadecimal value to the user buffer. The repeat mark ><R is not transmitted; it repeats the following character or hexadecimal value as many times as specified.



4. The sequence of the repeat mark followed immediately by a control character mark is deleted from the user buffer.



5. All other characters including the marks >< are passed to the user buffer.

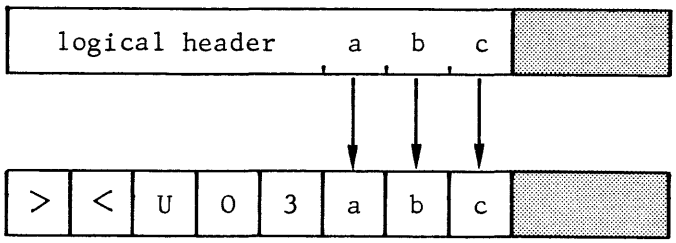
Input Unedited Mode

The input unedited mode is entered when

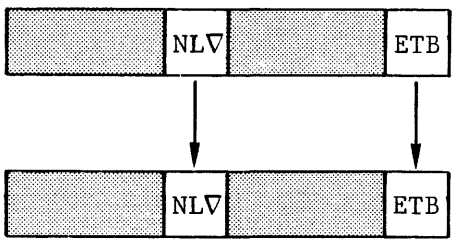
- the TERMNL command is specified with the parameter IM=UN
- the network command MTE specifying the terminal and the parameter INEDT is keyed in
- the terminal operator command $*$MTE specifying INEDT is issued.

Treatment of data in the unedited mode is as follows,

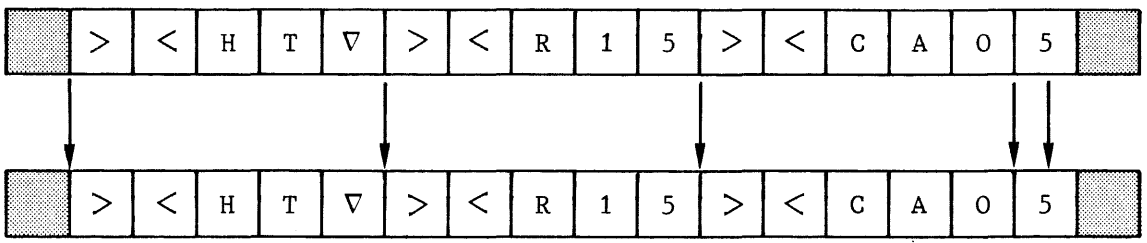
1. VIP headers are translated into mark form and passed to the user buffer.



2. Control characters and trailers are passed to the user buffer.

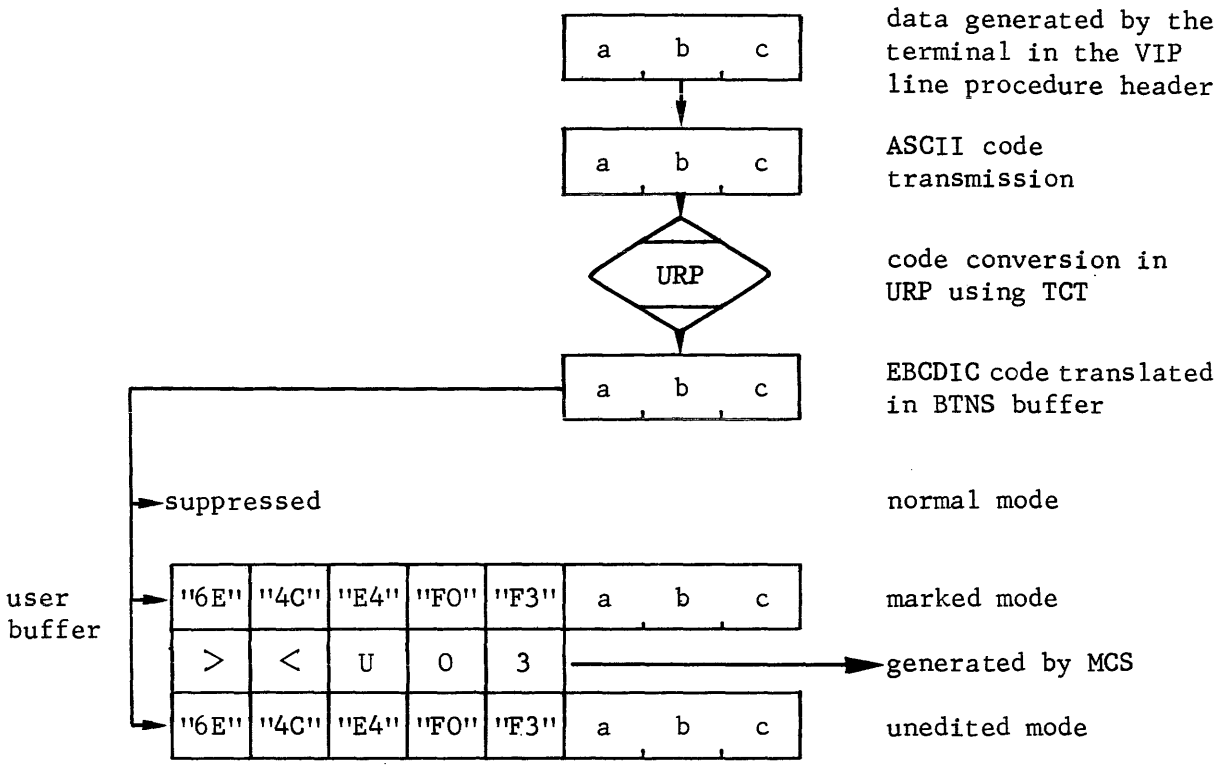


3. All other characters including >< are passed to the user buffer.

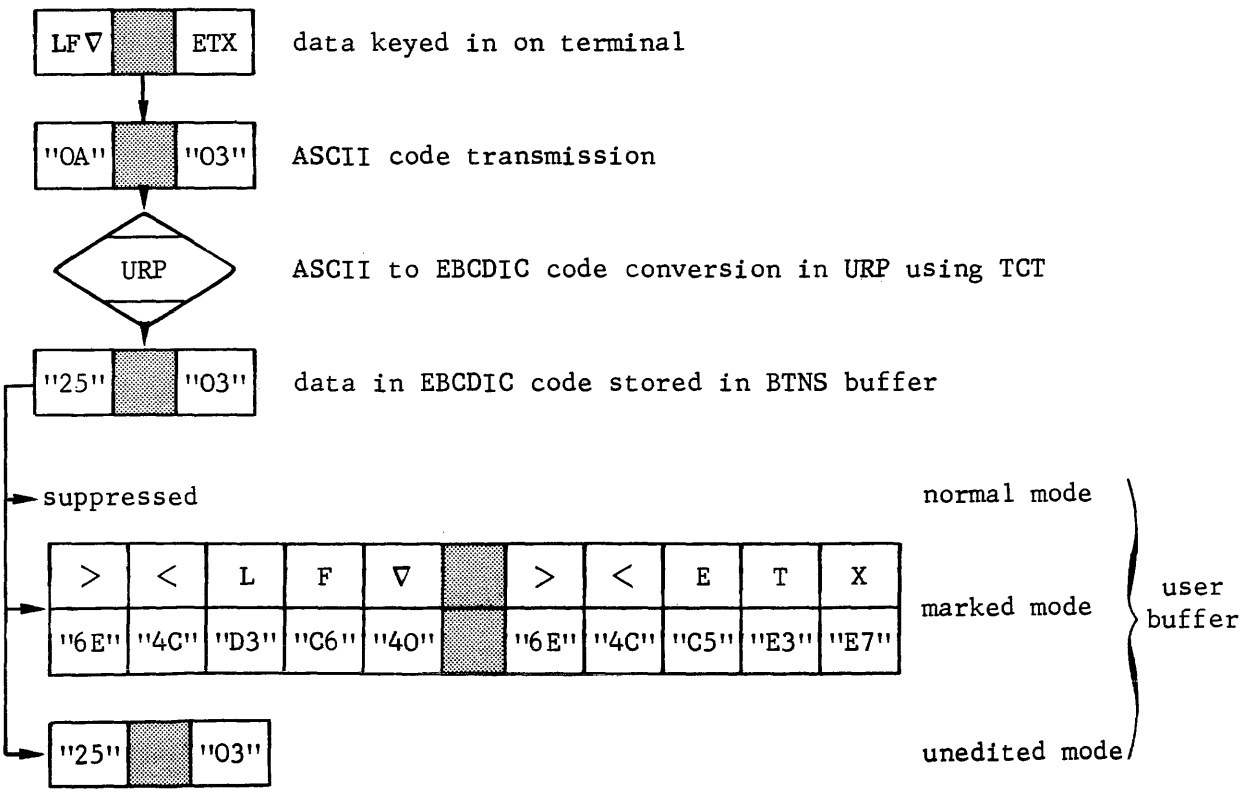


Input Message Flow

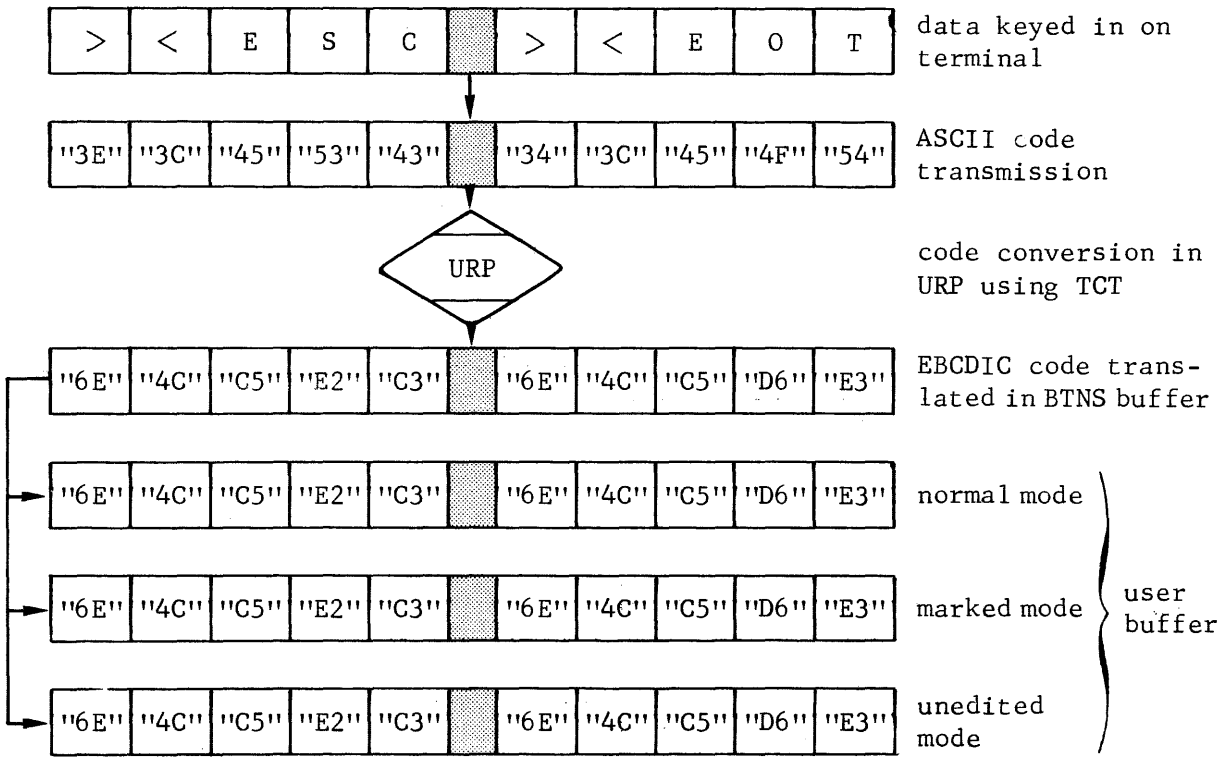
1. VIP-headers



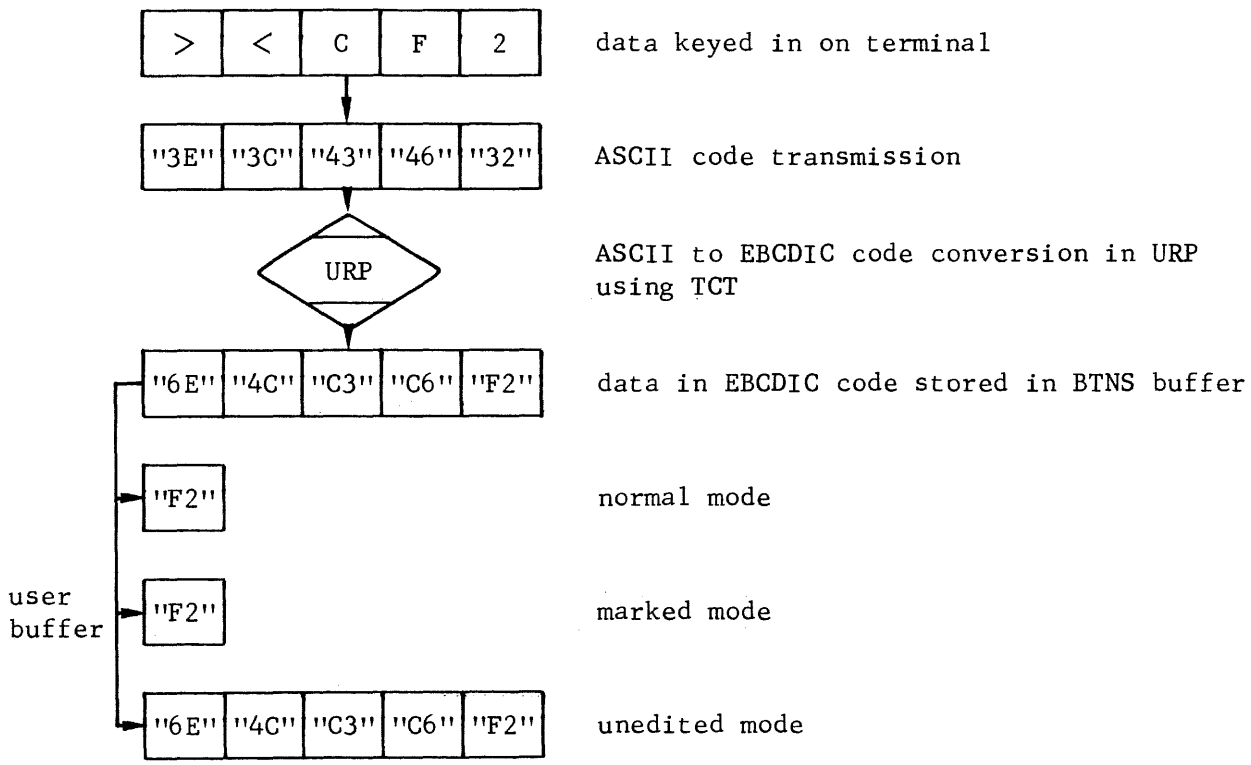
2. Control Characters and Trailers



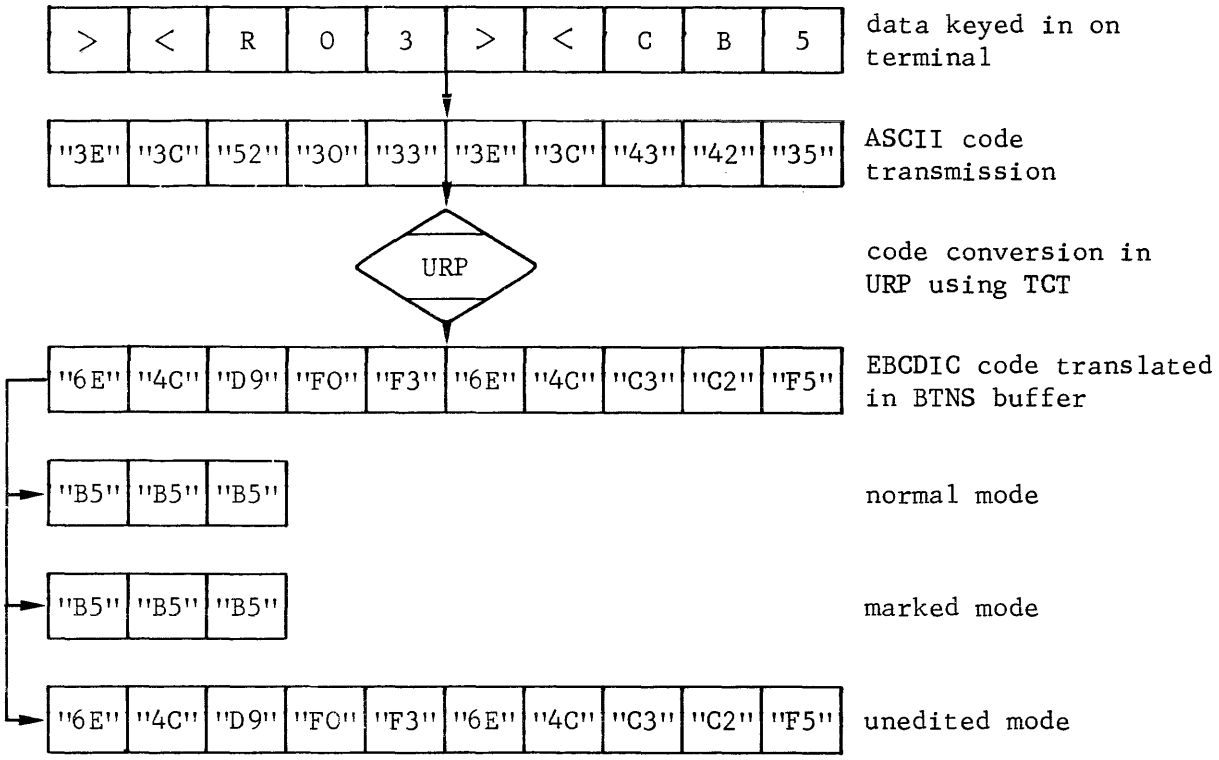
3. Control Characters and Trailers in Mark Form



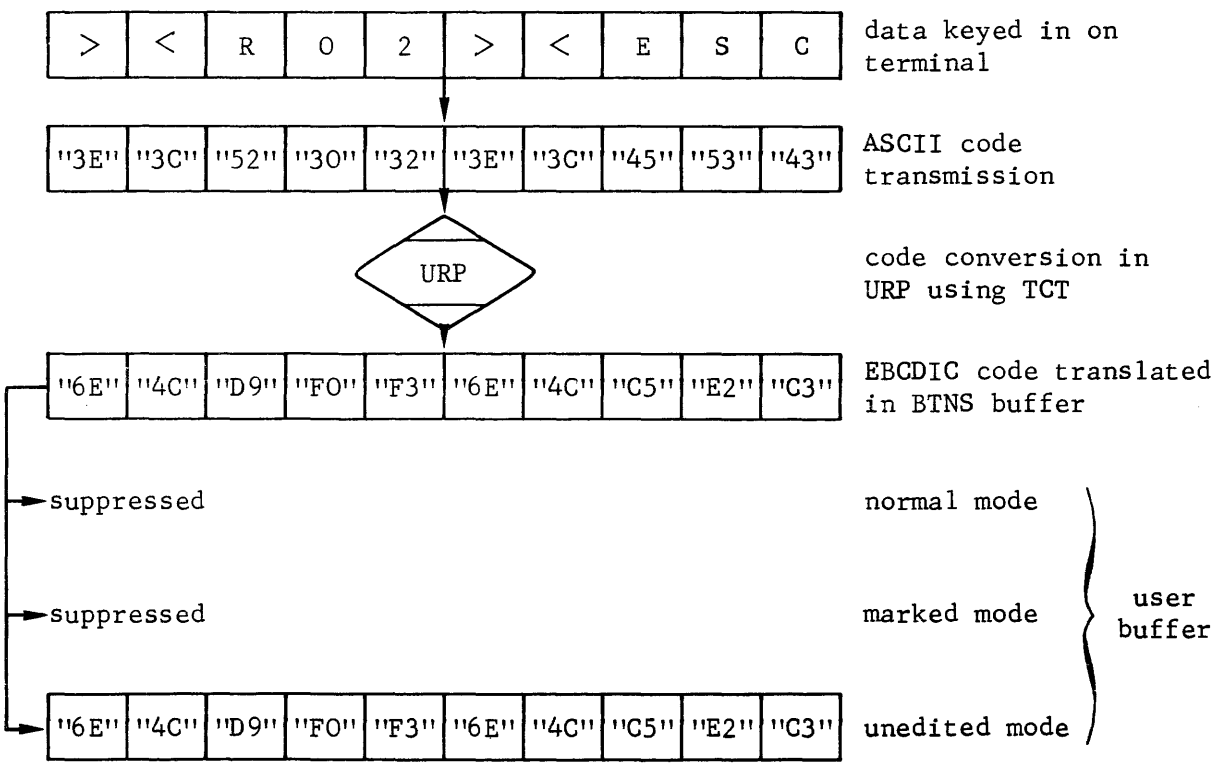
4. Character-encoding



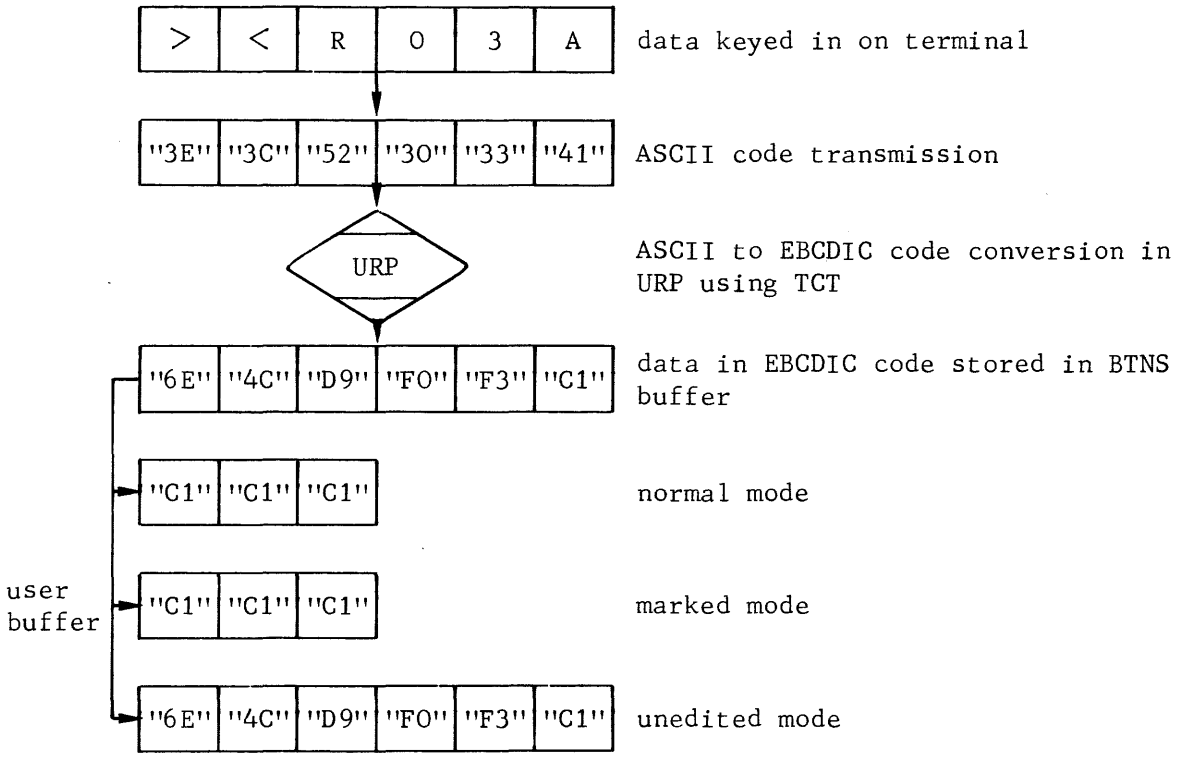
5. Repeat followed by Character-encoding



6. Repeat followed by Control Character Mark



7. Repeat followed by Graphic Symbol



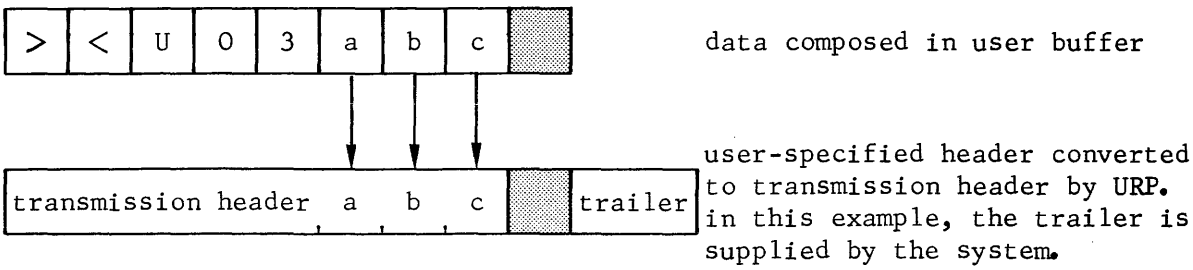
Output Normal Mode

The output normal mode is entered when

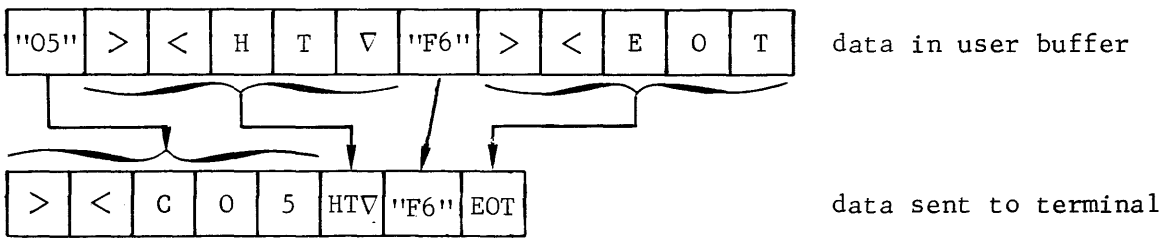
- the TERMNL command is not specified with the OM parameter, i.e., "normal" is the default output mode
- the TERMNL command is specified with the parameter OM=NL
- the network command MTE specifying the terminal and the parameter ONORM is keyed in
- the terminal operator command $*$MTE specifying ONORM is issued.

Treatment of data in the normal mode is as follows,

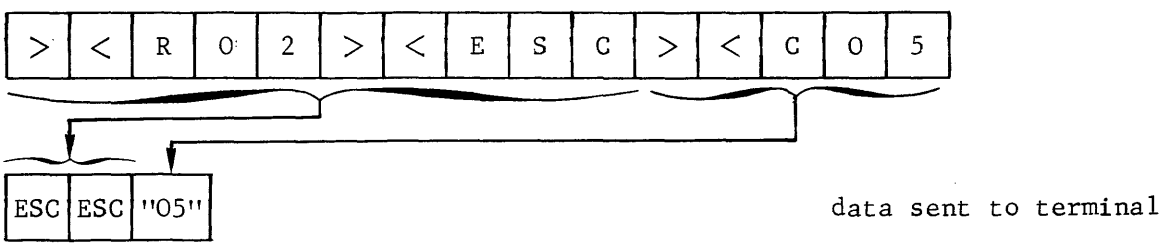
1. VIP headers can be provided in mark form to be passed to the terminal. The system provides trailers where none are specified by the user.



2. 1-byte hexadecimal characters representing control characters, as "05", are translated as 2 hexadecimal digits preceded by the character-encoding mark. 1-byte hexadecimal characters not representing control characters, as "F6", are passed directly without further process.
Control characters in mark form are translated as control characters.



3. The character-encoding mark ><C is not transmitted; it passes the following 2 hexadecimal digits as a 1-byte hexadecimal value to the terminal.
The repeat mark ><R is not transmitted; it repeats the following character, hexadecimal value or control character as many times as specified.



4. All other characters including the marks >< are passed to the terminal.

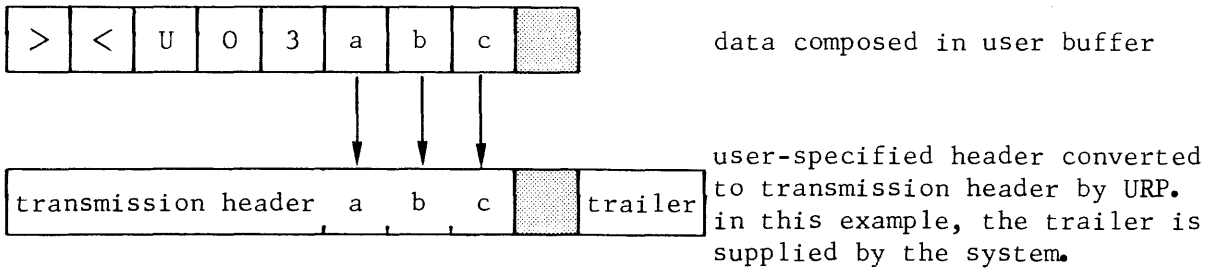
Output Unedited Mode

The output unedited mode is entered when

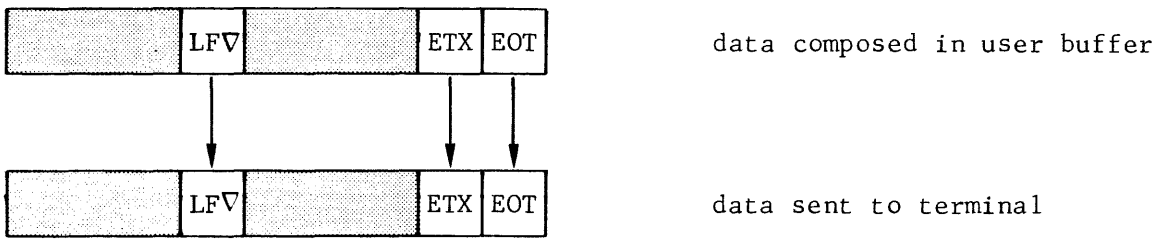
- the TERMNL command is specified with the parameter OM=UN
- the network command MTE specifying the terminal and the parameter ONEDT is keyed in
- the terminal operator command $*$MTE specifying ONEDT is issued.

Treatment of data in the unedited mode is as follows,

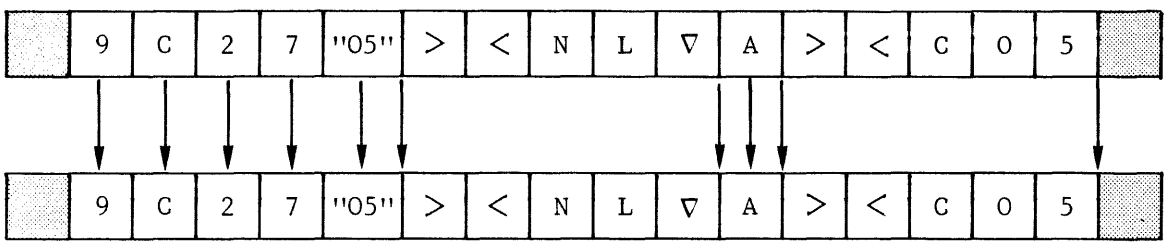
1. VIP headers can be provided in mark form to be passed to the terminal. The system provides trailers where none are specified by the user.



2. Control characters are passed to the terminal.

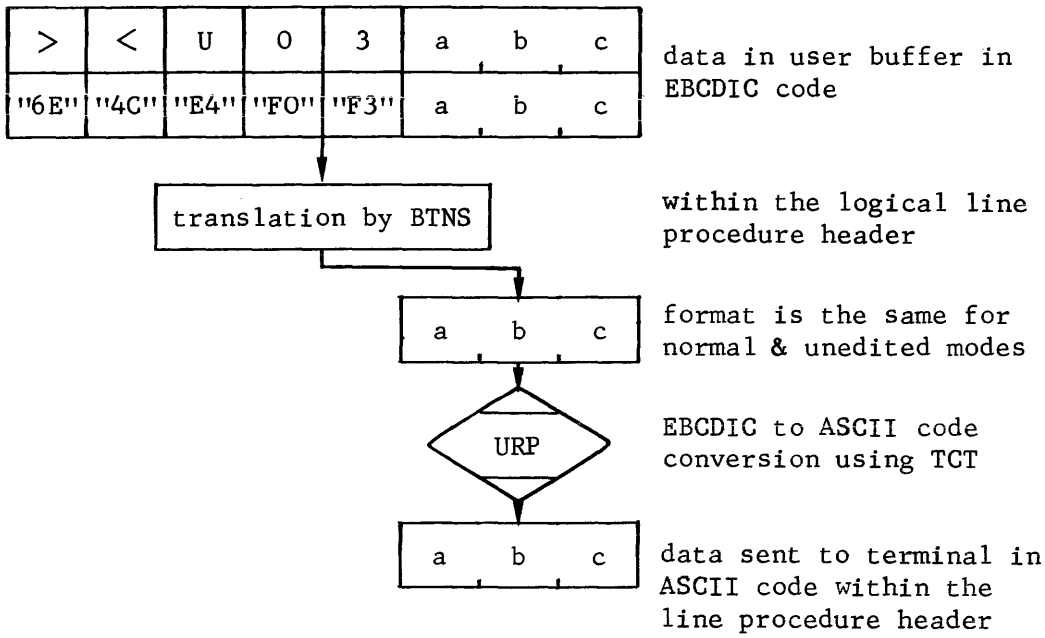


3. All other characters including the marks >< are passed to the terminal.



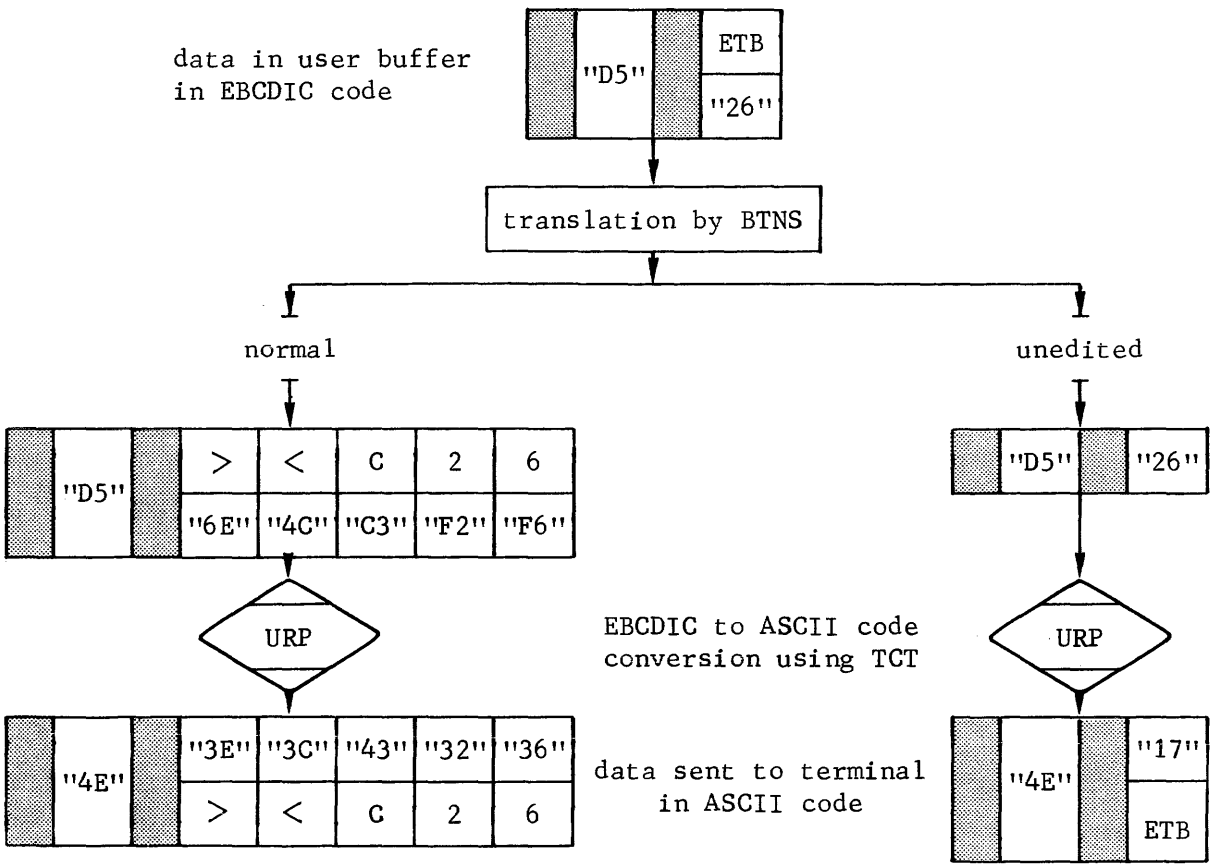
Output Message Flow

1. VIP-headers



2. Control Characters and Trailers

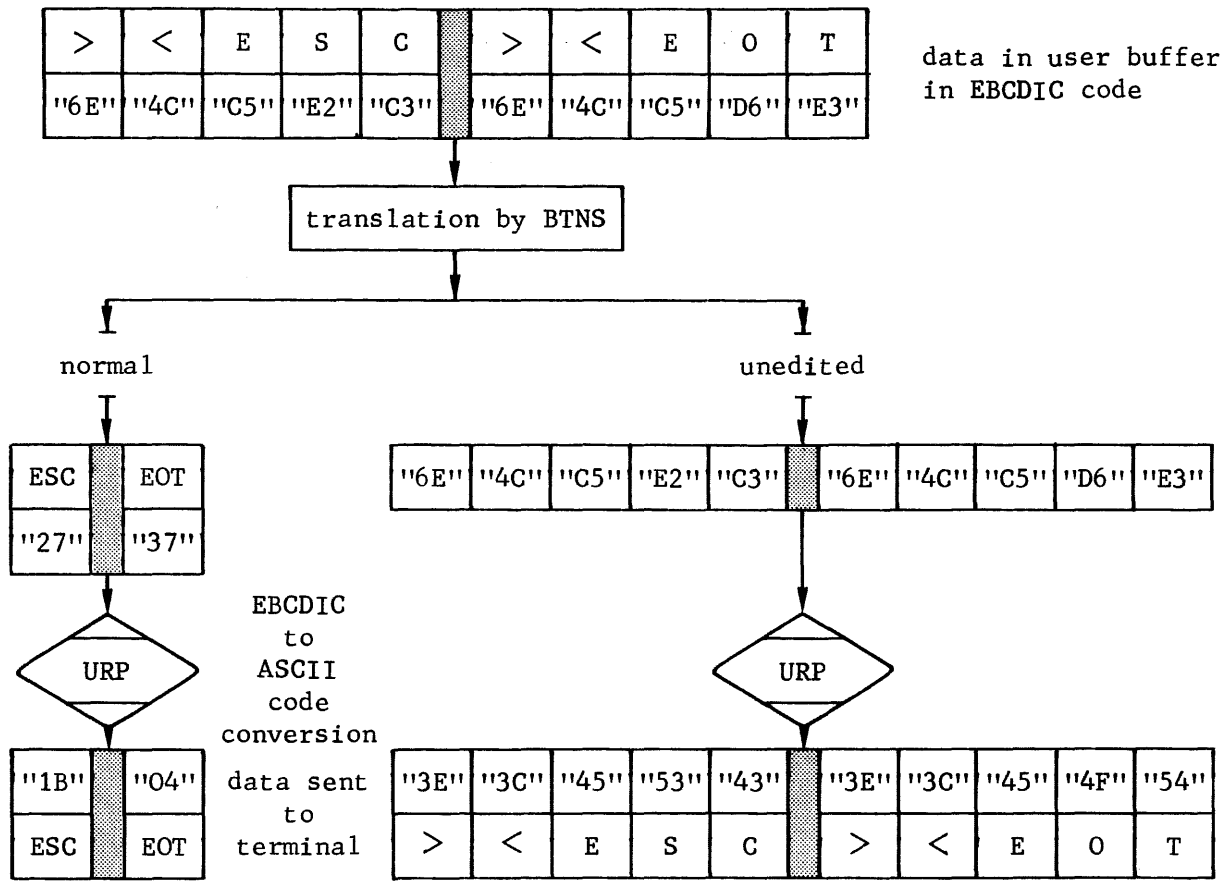
Note : "D5" is not a reserved control character value



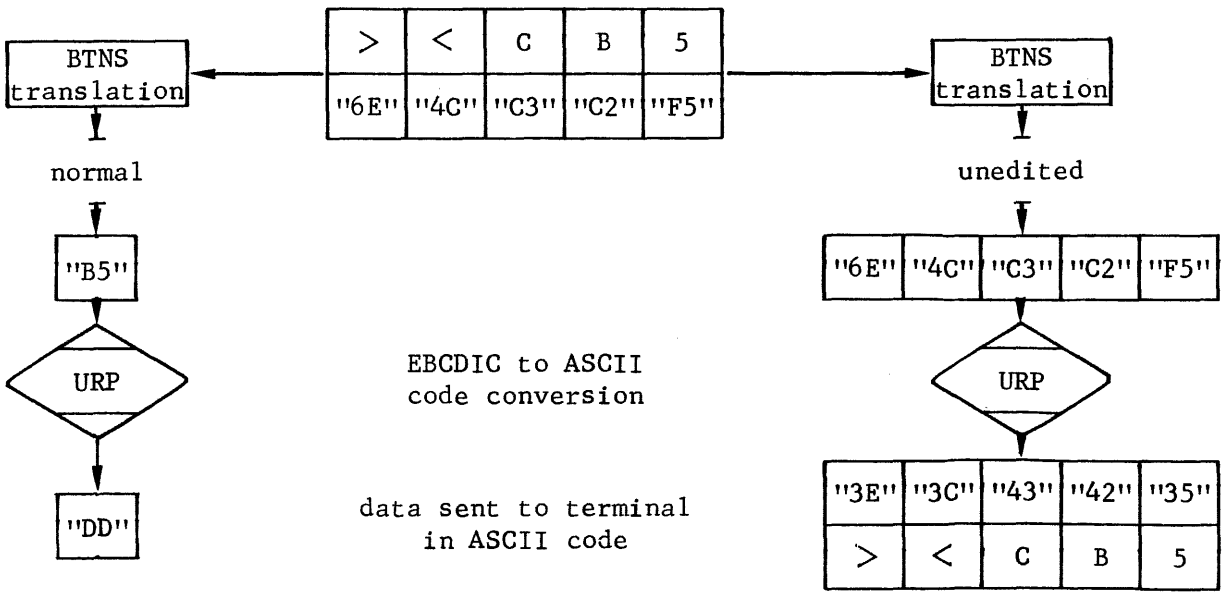
Output Message Flow

continued

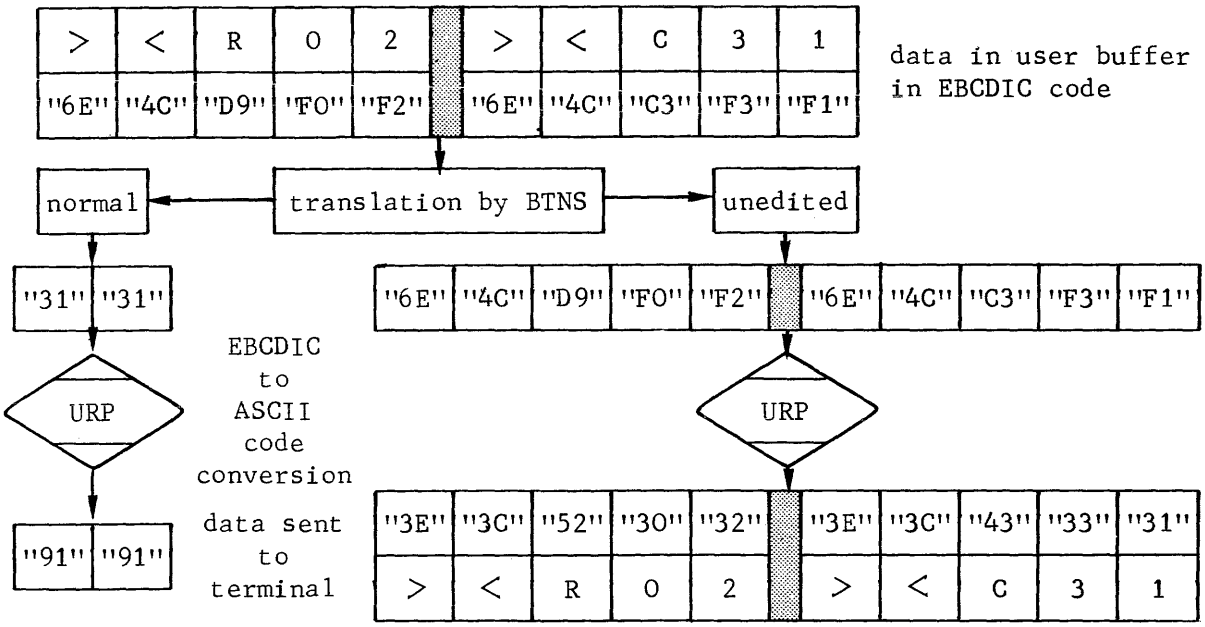
3. Control Characters and Trailers in Mark Form



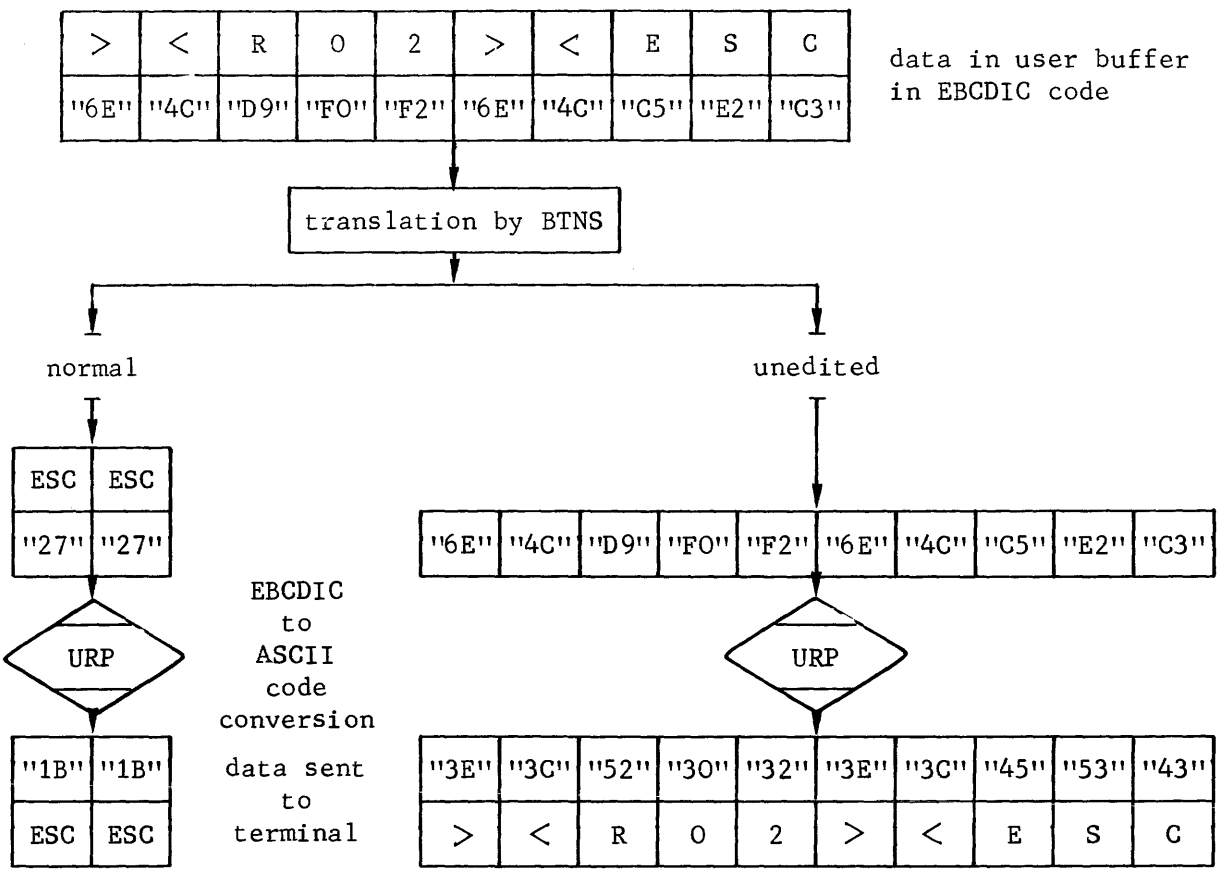
4. Character-encoding



5. Repeat followed by Character-encoding



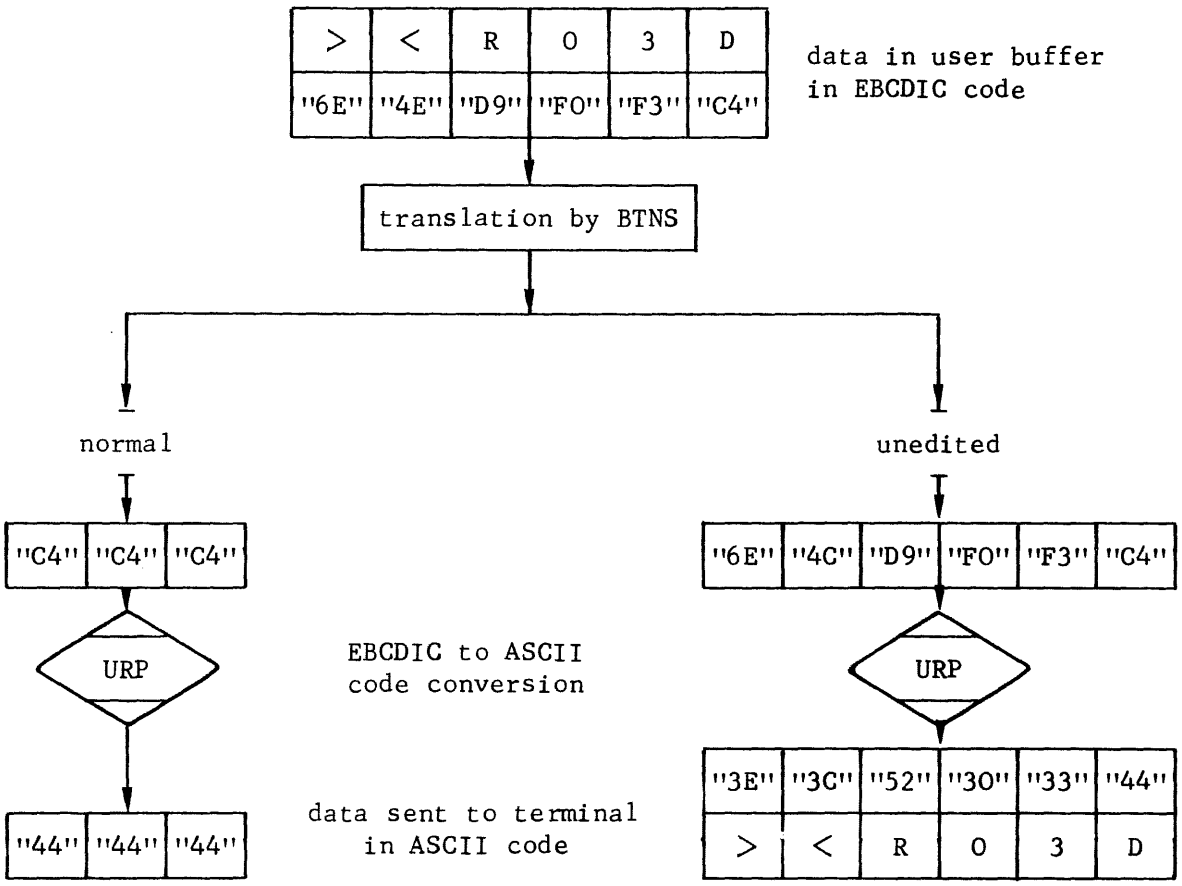
6. Repeat followed by Control Character Mark



Output Message Flow

continued

7. Repeat followed by Graphic Symbol



Summary of Data Formats					
	VIP-header	Trailer	Hexadecimal Control Character (e.g.,"27")	Character-Encoding in Mark Form (e.g.,><C27)	Control Character in Mark Form (e.g.,><ESC)
Input Normal Mode	deleted	deleted	deleted except for ∇ = "40" ~~HT = "05"~~	passes hexadecimal value "27"	untranslated passes ><ESC
Input Marked Mode	passes header in mark form ><U03abc	passes trailer in symbolic form	translates to mark form ><ESC	passes hexadecimal value "27"	untranslated passes ><ESC
Input Unedited Mode	passes header in mark form ><U03abc	passes trailer as entered	passes hexadecimal value "27"	untranslated passes ><C27	untranslated passes ><ESC
Output Normal Mode	optionally user-provided in mark form ><U03abc	system-provided if required	translates to character-encoding in mark form ><C27	passes hexadecimal value "27"	translates to hexadecimal equivalent "27"
Output Unedited Mode	optionally user-provided in mark form ><U03abc	system-provided if required	passes hexadecimal value "27"	untranslated passes ><C27	untranslated passes ><ESC

In the above table, the "escape" character corresponds to :

- The internal hexadecimal EBCDIC value "27"
- The control character in mark form : ><ESC

DATA REPRESENTATION

GCOS 64 offers the MCS COBOL programmer several modes of representing data depending on,

- the type of terminal to be used, e.g., special character set
- the complexity of the control functions to be activated
- the provisions to be made for a future transition of the program to TDS with the minimum modifications necessary to the program.

Data to be treated by the MCS COBOL program can broadly be categorized into two main groups, namely,

- information data
- control characters.

The main difference between information data and control characters is that control characters cannot be represented by graphic symbols.

In the programming examples that follow, it is assumed that the parameters listed have been correctly declared and initialized in the data division of the program,

- buffers, namely, INBUF and OUTBUF
- CD parameters, CDIN and CDOUT.

Information Data Representation

The three ways in which information data is represented are,

- graphic representation
- COBOL collating sequence
- mark representation

GRAPHIC REPRESENTATION

Problem 1 :

To send the text

```
* HERE IS DATA-ENTRY (78/03/29 VERSION) *
```

Solution :

The following coding may be used

1 In the DATA DIVISION declare as follows :

```
77 START PIC X(41) VALUE "* HERE IS DATA-ENTRY (78/03/29 VERSION) *".
```

2 In the PROCEDURE DIVISION code as follows :

```
MOVE START TO OUTBUF.
SEND CDOUT FROM OUTBUF WITH EMI.
```

Problem 2 :

To check if the message received is STOP which requests termination of the application program.

Solution :

The following coding may be used

1 In the DATA DIVISION declare as follows :

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

2 In the PROCEDURE DIVISION code as follows :

```
RECEIVE CDIN MESSAGE INTO INBUF.
IF INB21 = "STOP" GO TO TERM-PROG.
```

The data representation cited is valid for any transmission code.

The following problems, however, arise,

- The Level 64 internal code is EBCDIC while most of the terminals use the ISO ASCII code. The translation from EBCDIC to ASCII, and vice versa, is executed by the IURP or URP according to the TCT tables loaded by BTNS.

  A small number of special characters are rendered differently as follows,

EBCDIC graphic :	¢	|	!	¬	¦	(e.g., GCOS64)
ISO graphic :	[	!	]	^	|	(e.g., VIP terminal)

  This means that when a user wants to display on, say, a VIP terminal the text :

```
VERSION DATED : [ 78/03/28 ]
```

  he must declare in the data division of his program the following :

```
77 VERS PIC X(24) "VERSION DATED : ¢ 78/03/28 !".
```

EBCDIC representation is only used for local unit record devices, such as the system console, card reader/punch and line printer.

- The character set(s) available on the devices used to create and enter the COBOL program depends on the means of entry,
  - If the program is entered in the form of cards, the only character set possible is that of the standard EBCDIC character set, or a part of the set, allowed on the keypunch.
    Furthermore, the Level 64 card reader does not recognize lower case letters, thus restricting even more the character set available for use.
  - If the program is created under IOF (interactive operation facility) from a terminal having upper and lower case capability, i.e., shift-in/shift-out, the character range defining the set is greatly extended.
    Lower case letters in literal strings are allowed by the COBOL compiler.

A program on cards can only declare :

```
77 DAT PIC X(5) VALUE "DATE:".
```

A program created from a VIP7760 terminal can declare the format as above and, in addition, the following,

```
77 DAT PIC X(5) VALUE "date:".
```

The listing of the program on the line printer will only display upper case letters, even in the second case.

- Specific national language options have different graphic symbols.

On some terminals used in German-speaking countries,

```
Ö replaces \
```

If the programmer wants to send Ö, he has to declare \ in the data string.

For such special characters, the only way to ensure correct processing is to proceed as follows,

- refer in the appropriate terminal manual the ASCII value of the character
- find its EBCDIC equivalent, see "Translation from ASCII to EBCDIC"
- find the Level 64 graphic equivalent, where applicable, see "EBCDIC Character Set and COBOL Collating Sequence".

Problem : To send the character Å

Solution :

Proceed as follows :

1 Refer to the terminal manual for the ASCII value of Å

Let us say that the ASCII value for Å = 24

2 Refer to the table "Translation from ASCII to EBCDIC"

ASCII 24 = EBCDIC 5B

3 Refer to the table "EBCDIC Character Set & COBOL Collating Sequence"

EBCDIC 5B = graphic symbol $

4 Declare $ in the data division in order to display Å

COBOL COLLATING SEQUENCE

The facility allows the COBOL programmer to represent any character by using its value defined by its collating sequence known to the COBOL compiler.

Problem : To send the text | Date | containing upper and lower case letters

Solution :

Proceed as follows :

1 Refer to the table "EBCDIC Character Set & COBOL Collating Sequence" for the COBOL collating sequence values of the characters required

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

2 Declare in the data division of the program either form of coding :

```
77 DAT PIC X(16) VALUE "D""130,164,134"".
```

```
77 DAT PIC X(20) VALUE ""197,130,164,134"".
```

MARK REPRESENTATION

Mark representation is a general facility of MCS to represent information data and control characters in an easy-to-use symbolic form using their EBCDIC values.

The types of information data for the facility are,

- character-encoding marks :

Problem : To send the text | Date | containing upper and lower case letters

Solution :

Proceed as follows :

1 Refer to the table "EBCDIC Character Set & COBOL Collating Sequence" for the EBCDIC values of the characters required

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

2 Declare in the data division of the program either form of coding

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

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

- repeat marks :

Problem : To send a string of 80 asterisks *

Solution :

Proceed as follows :

Declare in the data division of the program either form of coding

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

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

, *=EBCDIC 5C

Control Character Representation

Control functions generated by the terminal or encoded by the programmer to send to the terminal are represented as control characters.

Control characters are generally restricted to the lower EBCDIC values; however, some terminal manufacturers use for specific control functions,

- other EBCDIC values not reserved for the terminal in question
- a combination of control characters and/or graphic symbols (escape sequences)

For detailed information on the control functions for specific terminals, see Appendix B.

VIP7700 Control Characters

The following are examples of control functions specific to the VIP7700 :

1 The "form feed" control character for clearing the screen and positioning the cursor is the standard EBCDIC value 0C.

2 The "blink" function is activated by the EBCDIC value 5F.

For other terminals, say the TN300, 5F represents the graphic symbol ^ (circumflex).

3 The cassette read command is implemented by the following sequence of 2 EBCDIC values, namely

where EBCDIC 27 = ESC or "escape" control character
and EBCDIC C5 = E, i.e., the graphic character

NO VISIBILITY OF CONTROL CHARACTERS

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

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

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

- the ADVANCING clause of the COBOL SEND verb
- the automatic editing functions provided by MCS thru the following parameters of the NDL TERMNL command, namely,
  - BLOCKING
  - LLENGTH
  - NBLOCKS

. ADVANCING clause :

The ADVANCING clause is used in the last SEND which terminates the message, i.e., SEND WITH EMI or SEND WITH EGI.

MCS then automatically generates control characters for insertion before and after the message text according to
- the type of the device receiving the message
- the control function requested.

Problem : To perform the following on the VIP7700 terminal
- to build the screen line by line
- to generate a "form feed" function on the last line of the message.

Solution :

Take the following considerations into account,
- the VIP7700 automatically performs a "new line" function at the end of each line
- to generate a "form feed" function, specify AFTER ADVANCING PAGE in the last line of the message.

Proceed as follows :

1 In the DATA DIVISION declare as follows :

```
77 OUTBUF PIC X(80).
77 IDX COMP-1.
```

2 In the PROCEDURE DIVISION use either coding :

```
      MOVE 0 TO IDX.
      MOVE TEXT TO OUTBUF.
* LOOP TO SEND 23 FIRST TEXT LINES
LOOP23.
      SEND CDOUT FROM OUTBUF.
      ADD 1 TO IDX.
      IF IDX < 23 GO TO LOOP23.
* SEND LAST LINE ADVANCING PAGE.
      SEND CDOUT FROM OUTBUF WITH EMI
      AFTER ADVANCING PAGE.
```

```
      MOVE 0 TO IDX.
      MOVE TEXT TO OUTBUF.
* LOOP TO SEND 24 TEXT LINES
LOOP24.
      SEND CDOUT FROM OUTBUF.
      ADD 1 TO IDX.
      IF IDX < 24 GO TO LOOP24.
* CLOSE MESSAGE AND SPECIFY ADVANCING.
      SEND CDOUT WITH EMI AFTER ADVANCING PAGE.
```

- automatic MCS editing :

  For the detailed description of the parameters providing automatic MCS editing, see TERMNL command in Section V.

  When this facility is activated, MCS inserts control characters in the message text to execute a "new line" function, namely,
  - at the beginning of the message
  - after each string of characters defined by LLENGTH

  NBLOCKS specifies the number of lines, each line containing the number of characters defined by LLENGTH, to be accepted for each message. If a message is greater that NBLOCKS x LLENGTH, then the characters in excess are discarded by MCS.

MARK REPRESENTATION

The transmission modes for which mark representation of control functions can be used are marked mode on input and normal mode on output.

- control character standard marks :

  For terminals recognized as standard to Level 64 by MCS, the symbolic representation of standard control characters is given in the tables "Control Characters & EBCDIC Values" and "EBCDIC Values & Control Characters".

  The control characters listed are dependent on,
  - the direction of transmission, i.e., input or output
  - the type of line procedure used, i.e., TTY, VIP or BSC.

The mark ><CAN is treated as follows,

- For a TTY terminal :
  - On being sent to or received from a TTY terminal, the mark represents the control function CANCEL, and the appropriate editing takes place in either direction.

- For a VIP terminal :
  - On being received from a VIP terminal, the mark represents the control function CANCEL, and the appropriate editing takes place on input.
  - On being sent to a VIP terminal, the mark does not represent a control function, and is, instead, treated as any other data.

. character-encoding marks :

The facility is used to send
- control characters not defined as standard marks
- a control sequence of characters which individually are neither control characters nor graphic characters.

Problem : To position the cursor of the VIP7700 terminal at column 37 of line 11

Solution :

Proceed as follows :

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

```
command : DC3  ; parameters following command are
                 . line number
                 . character position
```

2 Refer to the table overleaf, "ASCII Code for Control Characters & Graphic Symbols with EBCDIC Equivalents" for values of "line number" and "character position"

In order to determine the EBCDIC value to be given 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

In order to determine the EBCDIC value to be given character position 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 COBOL program use one of the character strings

```
><DC3*D
```

```
><DC3><C5C><CC4
```

ASCII Code for Control Characters & Graphic Symbols with EBCDIC Equivalents

Explanation of text :

"function" includes

- control characters
- graphic symbols

optional graphic symbols appear on left where 2 symbols are given.

ASCII	function	EBCDIC
00	NUL	00
01	SOH	01
02	STX	02
03	ETX	03
04	EOT	37
05	ENQ	2D
06	ACK	2E
07	BEL	2F
08	BS∇	16
09	HT∇	05
0A	LF∇	25
0B	VT∇	0B
0C	FF∇	0C
0D	CR∇	0D
0E	SO∇	0E
0F	SI∇	0F
10	DLE	10
11	DC1	11
12	DC2	12
13	DC3	13
14	DC4	3C
15	NAK	3D
16	SYN	32
17	ETB	26
18	CAN	18
19	EM∇	19
1A	SUB	3F
1B	ESC	27
1C	FS∇	1C
1D	GS∇	1D
1E	RS∇	1E
1F	US∇	1F
20	∇	40
21	! |	4F
22	"	7F
23	#	7B
24	$	5B
25	%	6C
26	&	50
27	'	7D
28	(	4D
29	)	5D
2A	*	5C
2B	+	4E
2C	,	6B
2D	-	60
2E	.	4B
2F	/	61
30	0	F0
31	1	F1
32	2	F2
33	3	F3
34	4	F4
35	5	F5
36	6	F6
37	7	F7
38	8	F8
39	9	F9
3A	:	7A
3B	;	5E
3C	<	4C
3D	=	7E
3E	>	6E
3F	?	6F
40	@	7C
41	A	C1
42	B	C2
43	C	C3
44	D	C4
45	E	C5
46	F	C6
47	G	C7
48	H	C8
49	I	C9
4A	J	D1
4B	K	D2
4C	L	D3
4D	M	D4
4E	N	D5
4F	O	D6
50	P	D7
51	Q	D8
52	R	D9
53	S	E2
54	T	E3
55	U	E4
56	V	E5
57	W	E6
58	X	E7
59	Y	E8
5A	Z	E9
5B	[¢	4A
5C	\	E0
5D	] !	5A
5E	^ ¬	5F
5F	_	6D
60	`	79
61	a	81
62	b	82
63	c	83
64	d	84
65	e	85
66	f	86
67	g	87
68	h	88
69	i	89
6A	j	91
6B	k	92
6C	l	93
6D	m	94
6E	n	95
6F	o	96
70	p	97
71	q	98
72	r	99
73	s	A2
74	t	A3
75	u	A4
76	v	A5
77	w	A6
78	x	A7
79	y	A8
7A	z	A9
7B	{	C0
7C	|	6A
7D	}	D0
7E	~	A1
7F	DEL	07

. VIP header marks :

For the VIP line procedure, control characters are embedded in the user message text by BTNS which automatically provides headers and trailers.

The three parameters in the VIP header, which are of use, are,
- STA, the status of the terminal
- FC1, function-code-1
- FC2, function-code-2.

For a detailed description of these control functions, refer to the VIP7700 terminal manual and Appendix B.

The only way to access these control characters both in input and output mode from a COBOL program is the VIP header mark of the form ><U03abc which is:
- delivered on input in front of the message text by the terminal
- provided on output in front of the message text by the user
- allowed in the unedited mode of transmission.

COBOL COLLATING SEQUENCE

The facility is available to the programmer who wishes to manipulate control characters directly. In order to do this, the terminals must be set in the unedited mode of transmission.

The EBCDIC values of the control characters must be declared thru their equivalent values of the COBOL collating sequence, as defined in the table "EBCDIC Character Set & COBOL Collating Sequence".

Problem : To send to the printer of the VIP7700 two consecutive lines of the following text BOOKINGS followed by NUMBERS:

Solution :

Proceed as follows :

1 To address the VIP7700 printer use the following values,

```
STA = EBCDIC 3F = COBOL collating sequence value 64
FC1 and FC2, to be left initially as spaces
```

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

```
CR▽ = EBCDIC 0D = COBOL collating sequence value 14
LF▽ = EBCDIC 25 = COBOL collating sequence value 38
```

3 Declare in the data division

```
77 HEAD PIC X(35) VALUE "><U03"64"▽▽BOOKINGS"14""38"NUMBERS:".
```

SECTION V

COMMUNICATIONS NETWORK CONFIGURATOR

CNC is a system utility program used to generate the communications environment to provide the network and software support for the correct execution of,

- BTNS, basic terminal network support
- MAM, message access method
- VCAM, virtual communications access method
- QMAINT, queue maintenance
- MCS COBOL application programs

CNC is described in terms of:

- Input Data
- Output Data
- Command Language
- Executing CNC
- Tuning the Network

INPUT DATA

The input data to CNC is in the form of,

- NDL commands, network description language
- Site SRST, system resource and status table
- Preallocated Disk Queue Files

NDL Commands

The NDL commands to CNC 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 NDL commands are to be used repeatedly for setting up the same communications environment, then they should be stored as a member of a library.

Site SRST

The SRST contains the hardware description and status of each communications line available to the site.
CNC checks the hardware availability of the lines declared thru NDL commands and parameters which are of importance to BTNS, such as,

- Line procedure
- Line type, i.e., leased, local or switched
- Allowed line speeds for a particular DCC (data communications controller).

Preallocated Disk Queue Files

If disk queues are to be generated, then the disk queue file must first be preallocated, see Section III "Preallocating a Disk Queue File".

The volume containing the disk queue file must be mounted, and the file must be ASSIGNed to the H_CNC step.
The disk queue file is then preformatted based on queue specifications in the network description.

OUTPUT DATA

The output data from CNC is in the form of,

- Communications System Tables
- Copy of Tables in a System File
- Preformatted Queue Files
- Output Reports

Communications System Tables

On successful completion of the H_CNC step, a set of communications system tables is created.
The contents of these tables incorporate the complete network description derived from the analysis of the specifications contained in the NDL commands, principally,

- Relating terminals to stations to lines
- Classifying component names with the appropriate component addresses
- Aligning queues with sources or destinations
- Computing buffer sizes per line.

The communications system tables are used by BTNS, MAM and VCAM to control the network for running user applications.

Copy of Tables in a System File

The set of communications system tables is copied in an area in backing store. A new version of these tables is created with each execution of the H_CNC step.

The file is used as follows,

- At system initialization : to restore the tables into reserved system segments, without having to regenerate the network using the CNC utility.
  A new generation is not required for the current network except in cases of modification.
- At BTNS startup : to restore the tables which may have been altered by operator commands during a previous communications session.
  BTNS is started up by the ST network control command which can be entered on the system console or on the network control terminal, see Appendix F.

Preformatted Queue Files

The preallocated disk queue file is preformatted on successful termination of the H_CNC step.
Control records are written at the beginning of the file for the purpose of restarting disk queues. The actual format of the control records is based on queue specifications from the network description.

Memory queues defined in the network description are preformatted in main memory.

Output Reports

Print-out reports can be of the following,

- SYSOUT Report : The information contained in the report
  - Lists the NDL commands used
  - Summarizes the makeup of the network environment generated
  - Indicates any errors detected during the execution of H_CNC utility.

  For list of NDL error messages, see Appendix C.
- JOR Messages : When system errors occur,
  - The step is invariably aborted
  - Messages are written to the JOR (job occurrence report).

  For list of JOR error messages, see Appendix C.

COMMAND LANGUAGE

NDL commands are described in terms of,

- Language Convention
- Command Repertoire

Language Convention

- Commands and their parameters are written in capitals.
- The following symbols must not be employed in user-supplied values,

, comma	/ slash	(open parenthesis
∇ blank	= equals	) close parenthesis
; semi-colon	* asterisk	" quotation mark

- Language reserved names must not be employed in user-supplied values. For summary of NDL commands and reserved syntax, see Appendix E.
- A command can span several cards, the end of the command being terminated by a semi-colon.
- Individual parameters of a command can be separated by commas, blanks or commas and blanks.
- Blank cards and "comment" cards are not processed.
- Default values are underlined; if no default is specified, a value must be supplied.
- A user-supplied value that lies beyond the range of permitted values is disallowed.

Command Repertoire

There are 7 commands, namely,

- COMM - defines a "comment"
- DCTAP - identifies the version of the VCAM subsystem
- GENCOM - describes the environment in which the network is to function
- LINE - defines the characteristics of the communications line
- QUEUE - defines all the queues to be used by the network
- STATN - defines a station attached to a polled line
- TERMNL - defines the characteristics of the terminal

The commands are listed with the following information,

- Definition
- Format of Command
- Parameter Descriptions
- Examples

Definition

The COMM command defines a comment and may appear anywhere in the description after the GENCOM command and even within the LINE-STATN-TERMNL sequence.

Format of Command

```
COMM "string";
```

Parameter Descriptions

"string"
- a character string between quotation marks that must be opened and closed on the same card.
  A maximum of 72 characters can be specified for each COMM command. If a comment is greater than 72 characters, then the excess number of characters must appear on a following COMM command.

DCTAP

Definition

The DCTAP command identifies a version of the VCAM subsystem that uses the network being described.

Format of Command

DCTAP name [,SIMU={ 1 / n }];

Parameter Descriptions

name
- ranges from 1 thru 4 alphanumeric characters and defines,
  - the name given to a particular version of TDS at TDS generation, see TDS/64 Programmer Reference Manual
  - the Interactive Operation Facility subsystem, in which case the name is IOF

SIMU
- an integer whose range depends on the version of the VCAM subsystem, namely,
  - IOF : from 1 thru 10, being the number of terminals defined by the INTERACT clause at system generation (SYSGEN).
  - TDS : from 1 thru 5, being the number of simultaneities defined by the SIMULTANEITY clause at TDS generation.

The default value is 1.

Examples

1) DCTAP TDS1;
   This command identifies a generation of TDS named TDS1 generated with 1 level of simultaneity.

2) DCTAP IOF,SIMU=5;
   This command specifies that 5 interactive terminals may be connected simultaneously to the IOF subsystem.

Definition

The GENCOM command defines the start of network generation and precedes all other commands. In addition to naming the network, this command describes the environment in which the network is to function.

Format of Command

```
GENCOM name [,BATCHNB={1|n}][,BTBFSZ={144|nnn}][,DABFSZ={144|nnnn}]
           [,KEY=password][,MAMNB={1|n}][,NBBTBF=nnn][,NBDABF=nnn]
           [,QCBLKSZ={70|nnnn}][,QCPOOL={0|nnnn}][,QDBLKSZ={70|nnnn}]
           [,SIMBRK={"$*$"|"xxx"}][,SYSLOG] ;
```

Parameter Descriptions

name
- ranges from 1 thru 4 alphanumeric characters and uniquely identifies the site name for the local network description.
  If the "name" is less than 4 characters, the system will fill in the rest of the "name" with "underscore" graphic symbols when identifying the network.
  Multiple descriptions may be cataloged as members of a source library, see Section IV.

BATCHNB
- VCAM oriented parameter which specifies the maximum number of batch entry utility processes that can be connected simultaneously to TDS.
  For a detailed description, see TDS/64 Site Manual and TDS/64 Programmer Reference Manual.
  The default value is 1.

BTBFSZ
- specifies the size in bytes of each unit forming the buffer pool used by BTNS during I/O transfers over the lines.
  The buffer pool services terminals sending to and receiving from queues and, where applicable, those terminals sending to VCAM subsystems.
  Buffer unit size ranges from 112 thru 512 in multiples of 16.
  The default value is 144.
  If the value specified is not in multiples of 16, then the size is rounded up to the nearest multiple.

DABFSZ
- VCAM oriented parameter which specifies the size in bytes of each unit forming the VCAM buffer pool used by BTNS for the sole purpose of sending messages to terminals connected to VCAM subsystems.
  The buffer pool is required by BTNS only if VCAM subsystems are declared, such as IOF and TDS in a DCTAP command.
  Buffer unit size ranges from 112 thru 1024 in multiples of 16.
  The default value is 144.
  If the value specified is not in multiples of 16, then the size is rounded up to the nearest multiple.

KEY
- MCS oriented parameter which defines the unique password used by MCS COBOL application programs to execute ENABLE and DISABLE verbs as specified thru the KEY parameter of the verbs.
  If the parameter is not specified, no checking is done by MAM when the verbs are executed.
  The password ranges from 1 thru 10 alphanumeric characters.

MAMNB
- MCS oriented parameter which specifies the maximum number of MCS COBOL application processes that can be executing concurrently.
  For the multi-process application, each constituent process is to be individually counted.
  The default value is 1.

NBBTBF
- specifies the number of units of the buffer pool used by BTNS during I/O transfers over the lines.
  The buffer pool services terminals sending to and receiving from queues and, where applicable, those terminals sending to VCAM subsystems.
  The minimum number is 8.
  The number of units is defined by the algorithm,

  $(NBBTBF \times BTBFSZ) \leqslant 65504$

  For default value, see "Tuning the Network" later in the section.

NBDABF
- VCAM oriented parameter which specifies the number of units of the VCAM buffer pool used by BTNS for the sole purpose of sending messages to terminals connected to VCAM subsystems.
  The buffer pool is required by BTNS only if VCAM subsystems are declared, such as IOF and TDS in a DCTAP command.
  The minimum number is 8.
  The number of units is defined by the algorithm,

  $(NBDABF \times DABFSZ) \leqslant 65504$

  For default value, see "Tuning the Network" later in the section.

QCBLKSZ
- MCS oriented parameter which specifies the size in bytes of each core block used to generate the core queue pool.
  Core block size ranges from 70 thru 1024.
  The default value is 70.

QCPOOL
- MCS oriented parameter which specifies the number of core blocks of the core queue pool to be shared by all queues qualified by the QCPOOL option in their respective QUEUE commands.
  The number of core blocks is defined by the algorithm,

  $(QCPOOL + NUMBLK) \times QCBLKSZ \leqslant 65535$

  For NUMBLK, see QUEUE command.
  The default value is 0.

QDBLKSZ

- MCS oriented parameter which specifies the block size in bytes of the disk queue file used for saving messages.
  The block size specified is the size of the MAM buffer unit used to store messages before being written to the file.
  The disk block size ranges from 70 thru 1024.
  The default value is 70.

SIMBRK

- ranges from 1 thru 3 alphanumeric characters enclosed within quotation marks.
  The character string serves as a break qualifier to precede the terminal operator command for identification by BTNS as a network control command.
  For terminal operations, see Communications Network Control Terminal Operations Manual and Terminal Operations GCOS Manual.
  The default value is $*$.

SYSLOG

- specifies all system messages sent by BTNS (site controller) to the network control terminal or any other terminal of the network are to be logged in order to be printed out at the end of the BTNS session, together with the job occurrence report of BTNS.

GENCOM
continued

Examples

1) GENCOM MCS1,BTBFSZ=200,KEY=PASS,MAMNB=2,NBBTBF=20,QCBLKSZ=205,QCPOOL=50;
This command names a network MCS1 which specifies:
- BTBFSZ=200,NBBTBF=20
  The BTNS buffer pool is made up of 20 units of 200 bytes per unit.
- KEY=PASS
  The ENABLE and DISABLE verbs issued by MCS COBOL application programs will have to specify the KEY parameter.
- MAMNB=2
  Concurrent execution of either
  - 2 single process (monoprocess) applications
  - 1 MCS application having 2 constituent processes or subprocesses, i.e., 1 dual process application.
- NBBTBF, see BTBFSZ
- QCBLKSZ=205
  Core block size is 205 bytes partitioned as follows,
  - 5 bytes reserved for MAM control information
  - 200 bytes for storing message text.
- QCPOOL=50
  A set of 50 core blocks is reserved in the core queue pool.

2) GENCOM DAC1,BATCHNB=3,BTBFSZ=150,DABFSZ=150,NBDABF=30,SIMBRK="%";
This command names a network DAC1 which specifies:
- BATCHNB=3
  3 batch entry utility processes can be simultaneously connected.
- BTBFSZ=150
  The size of the BTNS buffer unit is 150 bytes.
  The number of buffer units is automatically determined by CNC according to the network components.
- DABFSZ=150,NBDABF=30
  The VCAM buffer pool is made up of 30 units of 150 bytes per unit.
- NBDABF, see above
- SIMBRK="%"
  The break qualifier for network control commands is %.

Definition

The LINE command identifies the line and defines its characteristics. Each communication line to be supported by the network declared must be described by a separate LINE command.

Format of Command

```
LINE LNnn [,BKMOD][,CCINS=nnn][,CLOSE][,CLV=n][,EPX][,ERCAP]
 [,ININS=nnn][,LINPLG][,NAUCDE][,PADNB=nn][,PARITY=n]
 [,POLLIST=(nn[,nn]...)][,PRIRA={IN|OU}/nnnn/{QRRO|QLIN}][,RTCNT=nn]
 [,SPEED=x][,SPN={0|1}][,SSR][,TCTNM={ASCII|SPTTY}][,TILEN=nnn]
 [,TOLEN=nnn];
```

TLT Parameters

The TLT is a URP table attached to each line which contains all the parameters characterizing the functionality of the line. Some of these parameters not accessible to system software must be initialized during URP image generation using the LPGEN utility program. Other parameters are initialized by BTNS for each line during the start-up phase except for those lines declared CLOSEd, in which case an RT network control command must be issued for processing of the line to be activated.
The TLT loaded by BTNS is built up by CNC which reads hardware parameters from the URP and initializes system software visible parameters either with default values or with parameters explicitly specified in the NDL commands.
The default values of each parameter are summarized at the end of this command description.

The parameters may be classified into 3 categories:

- TLT Tuning Parameters which can be dynamically modified during the BTNS session by means of the MTL network control command.
- TLT System Parameters which depend on the requirements of the application program and are determined by the systems analyst/programmer or which are affected by manipulative procedures under the control of the terminal operator.
- TLT Hardware and Procedure Parameters which depend mainly on the hardware characteristics of terminals, modems and procedure specifics.

Parameter Descriptions

LNnn

- uniquely defines the line within the network and is declared in the SRST at the time of installation.
  Values range from 01 thru 99.
  CNC checks that the line declared is
  - present in the SRST
  - connected and available
  - not the system console (usually allocated LN01 or CSP in the SRST)

BKMOD
- TLT system parameter which specifies the effect of a "break" during the output of a message. The parameter is applicable only for TTY terminals equipped with the BREAK key.
  - If specified, output is immediately terminated.
  - If not specified, then output continues until the end of the current message before the "break" condition is reported to BTNS.

CCINS
- TLT hardware/procedure parameter defining the number of "fill" characters to be inserted after the control character activating a mechanical function, such as "carriage return" or "line feed", in order to avoid losing the following data characters that would be transmitted during the execution of that function.
  The number of "fill" characters ranges from 0 thru 127 and depends on the terminal type.
  If 0 is specified, no "fill" characters are inserted.

CLOSE
- specifies that BTNS must not initialize traffic over the line declared until an RT network control command is issued in order to start processing of the line.
  TLT for the line is not even loaded during BTNS start-up.
  By default a line is active.

CLV
- TLT hardware/procedure parameter which specifies the code level used on the line (including the parity bit, where applicable).
  Values range from 0 thru 3 with the following meanings,

  0 : 8 bit code
  1 : 7 bit code
  2 : 6 bit code
  3 : 5 bit code

  The 5 bit code is illegal for a synchronous line, i.e. VIP Synchronous or BSC.

EPX
- TLT system parameter which specifies that echoplex is used over a TTY line.
  A character received by the URP is turned around to the transmitting terminal.
  Echoplex can only be specified for fully duplex lines and can be used for TTY37 type terminals having a printer and keyboard disconnectable from each other.

ERCAP
- TLT system parameter which specifies the "erase" capability for TTY lines.
  When specified, the operator may delete the last character by keying in a "back slash" or "reverse slant" character (\).
  The input message length which includes "erase" characters must not be greater than (BTBFSZ - 9).
  For BTBFSZ, see GENCOM command.

ININS
- TLT hardware/procedure parameter which defines the number of SYN characters to be sent before each message to allow for synchronization.
  The parameter is only applicable for a synchronous line, i.e. VIP Synchronous or BSC.
  The number of SYN characters range from 0 thru 127 and depends on the characteristics and quality of the terminal.
  If 0 is specified, 2 SYN characters are inserted.

LINPLG
- TLT system parameter which specifies that polling for a multipoint line must be linear, i.e. polling to start at the top of the list.
  By default, round-robin polling is assumed, i.e. subsequent polling starts after the last entry successfully polled.

NAUCDE
- TLT hardware/procedure parameter only applicable for a switched line.
  Inhibits the detection of the ring indicator over the line, thus making it unavailable to the network.
  In order for the line to be made available, the parameter must be omitted in the next network generation.

PADNB
- TLT hardware/procedure parameter which defines the number of PAD characters to be sent at the end of the output message in order to avoid losing the last character(s) in case of turn-around on reception.
  The number of PAD characters ranges from to 15 and depends on modem characteristics.

PARITY
- TLT hardware/procedure parameter used to check the parity on input and depends on the type of terminal connected.
  Values range from 0 thru 3 with the following meanings,

  0 : Code without parity
  1 : Code with "pseudo-parity", i.e., the parity bit is always set to 1
  2 : Code with even parity i.e., the number of "1-bits" is even
  3 : Code with odd parity, i.e., the number of "1-bits" is odd

POLLIST
- specifies the polling list to be used over the line which must be declared multipoint in the SRST, i.e., VIP.
  The stations to be polled are identified by their STIs, station indexes, which range in value from 1 thru 32.
  The number of stations entered on the list must not be greater than the number defined for the SRST parameter POLISTLENGTH declared for the line.
  STIs may be repeated in any order provided they correspond with the STIs declared by STATN commands following this LINE command.
  The polling list may be modified by the MTP network control command provided that the number of stations in the new polling list does not exceed that originally specified.
  The default polling list for the line is established by the order in which successive STATN commands are specified for the line, i.e., 1 entry per station in the polling list.

PRIRA

- {IN | OU}/nnnn

MCS oriented parameter specifying input or output priority on a line.
The number of priority actions to be attempted before taking the other action ranges from 1 thru 4096.
The default is 1 input action for every output action.

- {QRRO | QLIN}

specifies the order in which terminals connected to MCS COBOL applications using output queues are scanned.
If QLIN is specified, the scan begins with the first output queue declared and keeps on returning to this queue until all data is output before the next queue is scanned.
The default is QRRO in which the scan begins with the first output queue declared and then resumes at the next queue, irrespective of whether all data in the previous queue has been output.

RTCNT

- TLT hardware/procedure parameter which specifies the number of retries to be attempted when an error occurs in transmission.
Values range from 1 thru 99.

SPEED

- TLT tuning parameter which defines the speed of an asynchronous line, i.e., TTY or VIP asynchronous.
Values range from A thru O representing the following speeds in bits per second.

A = 50	F = 150	K = 1050
B = 75	G = 200	L = 1200
C = 100	H = 300	M = 1800
D = 110	I = 600	N = 2000
E = 135	J = 900	O = 2400

A Data Communications Controller (DCC) is configured with 4 of the 15 speeds listed. CNC checks that the value defined for the parameter is 1 of the 4 speeds configured on the DCC which connects the line.
The speed of a line is selected from the 4 values and is specified at SRST generation time, by the LINE resource card, in order to tailor the URP buffers for that line.
In order to override the SRST value, the SPEED parameter supplies the new value which must be compatible with generated URP buffers for that line.

SPN

- TLT hardware/procedure parameter which defines the number of stop bits to be sent after each character in asynchronous transmission, i.e., TTY, or VIP asynchronous.
Values can be either 0 or 1 depending on terminal type and have the following meanings.

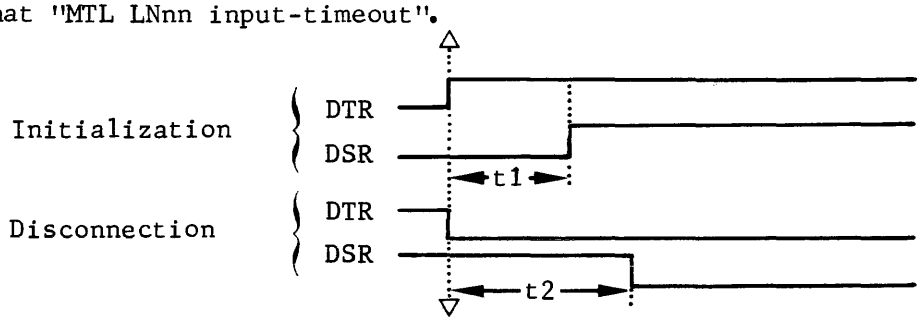
0 : 1 stop bit
1 : 2 stop bits

SSR
- TLT hardware/procedure parameter which is specified according to the type of modem and line speed, as given in the table below.

Modem Characteristics				Position of Modem	SSR
Bauds	Transmission	Selection Mode	Interface		
300	asynchronous	frequency emission	Pin 11 CANNON	terminal	specify
				CPU	omit
600	asynchronous	debit binary	Pin 23 CANNON	-any-	omit
1200	asynchronous	debit binary	Pin 23 CANNON	-any-	specify
1200	synchronous	debit binary	Pin 23 CANNON	-any-	omit
2400	synchronous	debit binary	Pin 23 CANNON	-any-	specify
>2400	-any-	-any-	-any-	-any-	omit

TCTNM
- {ASCII|SPTTY}
  TLT hardware/procedure parameter which allows selecting a translation table other than the standard one defined for the line procedure.
  ASCII is specified for some BSC transmissions, depending on the terminal type, and defines an ASCII-to-ASCII translation table.
  SPTTY defines the standard TTY translation table and may be patched to obtain specific functions, e.g., alteration of "erase" or "end-of-message" characters.

TILEN
- TLT tuning parameter which defines the "time-out" value used by the URP to control initialization or disconnection of the modem.
  The "time-out" value ranges from 20 thru 255 units, each unit being 50 milliseconds.
  In the diagram below, the value specified must be greater than either t1 or t2, both values depending on the quality of the modem.
  If the "time-out" value is too small, BTNS will display the message on initialization: "CC21 LNnn CANNOT BE OPENED REASON: ENABLE CP FAILED" ; on disconnection : "CC11 LNnn HELD BY SYSTEM REASON: DISABLE ERROR".
  The parameter can be modified by the MTL network control command of the format "MTL LNnn input-timeout".



TOLEN

- TLT tuning parameter which defines the "time-out" value used by the URP,
  - when polling a buffered terminal
  - when timing the interval between characters input from an unbuffered terminal.

  Buffered terminals:
  The "time-out" value ranges from 0 thru 65535 units, each unit being 50 milliseconds.
  0 value gives a "time-out" value of 50 x 65536 milliseconds.
  Unbuffered terminals:
  The "time-out" value ranges from 0 thru 255 units, each unit being 2 seconds.
  0 value gives a "time-out" value of 2 x 256 seconds.
  If "time-out" occurs when polling a buffered terminal, BTNS reports the condition on the network console: "CC13 station-name UNAVAILABLE".
  If "time-out" occurs for an unbuffered terminal, BTNS will deliver the text already input to the application program, and then launch a new "read".
  If "time-out" occurs for a switched line, BTNS disconnects the line.
  The parameter can be modified by the MTL network control command of the format "MTL LNnn output-timeout".

TLT Parameter Default Values					
Parameter	Line Transmission Procedure				
	TTY		VIP Asynchronous	VIP Synchronous	BSC
CCINS	3 (DLE)		0 (DLE)	6 (DLE)	2 (SYN)
CLV	0		0	0	0
ININS	na		na	4	4
PADNB	4		0	0	0
PARITY	2		2	3	0
RTCNT	0		4	4	4
SPEED	(a)		(a)	(a)	(a)
SPN	1		0	na	na
TCTNM	0		1	1	3
TILEN (b)	140		140	140	140
TOLEN (c)	64		16	16	416

Note : (a) values defined by SRST entry for the line
(b) value of unit = 50 milliseconds
(c) value of unit = 2 seconds for TTY, 50 milliseconds for others
na = not applicable

The table above only deals with parameters for which values are to be defined.
The default for options specified only by the keyword is the absence of the keyword.

Allowed Options for LINE Parameters					
Parameter	Line Transmission Procedure				
	TTY		VIP Asynchronous	VIP Synchronous	BSC
BKMOD	yes		no	no	no
CCINS	yes		yes	yes	yes
CLOSE	yes		yes	yes	yes
CLV	yes		yes	yes	yes
EPX	yes		no	no	no
ERCAP	yes		no	no	no
ININS	no		no	yes	yes
LINPLG	no		yes	yes	yes
NAUCDE (d)	yes		yes	yes	yes
PADNB	yes		yes	yes	yes
PARITY	yes		yes	yes	yes
POLLIST(e)	no		yes	yes	no
PRIRA (f)	yes		yes	yes	yes
RTCNT	yes		yes	yes	yes
SPEED	yes		yes	no	no
SPN	yes		yes	no	no
SSR	yes		yes	yes	yes
TCTNM	yes		no	no	yes
TILEN	yes		yes	yes	yes
TOLEN	yes		yes	yes	yes

Note : (d) allowed only if the line is declared switched in the SRST
(e) allowed only if the line is declared multipoint in the SRST
(f) only if the terminals connected over the line are to be used by MCS COBOL applications

Examples

1) LINE LN10;
   This command defines the line named LN10 in the SRST and which is to function with the default options defined by CNC according to the line procedure specified in the SRST.

2) LINE LNO4,LINPLG,POLLIST=(1,1,1,2,3),TOLEN=4;
   This command defines a multipoint line named LNO4 which specifies:
   - LINPLG
     Each polling cycle is to start with the station identified by its STI=1 at the "head" of the polling list defined by POLLIST.
   - POLLIST
     The variables specified for the parameter mean that
     - The length of the allowed polling list declared for the line LNO4 is at least 5, which accounts for the 5 STIs defined
     - For each polling cycle, the station identified by its STI=1 will be polled 3 times for every 1 successive time that stations 2 and 3 are polled.
   - TOLEN
     The "time-out" value used by URP firmware is 4 x 50 = 200 milliseconds. "Time-out" defines the lapse between sending the polling message and the station response.

3) LINE LN15,CLOSE,EPX,ERCAP,SPEED=D;
   This command defines a line named LN15 which specifies:
   - CLOSE
     The line will not be initialized at BTNS start-up but only thru an "RT LN15" network control command.
   - EPX
     The line declared is a TTY line for which "echoplex" is to be used. The option is valid if the keyboard of the terminal is disconnected from the printer.
   - ERCAP
     The (\) character is to be used as the "erase" character.
   - SPEED
     The line declared is a TTY line operating at a nominal speed, say, 300 bits per second (value = H).
     The speed of transmission is to be altered to 110 bits per second, i.e., value = D, which is 1 of the 4 speeds configured for the line.
     Speed can be modified at run-time by the MTL network control command.

Definition

The QUEUE command defines every physical queue, program and terminal queues, used by the network. The command gives the queue its external name and defines all its attributes.
QUEUE commands can appear anywhere in the network description after the GENCOM command except between the LINE command and its associated STATN and/or TERMNL commands.

Format of Command

QUEUE name [,BREAK][,CLOSE][,{CTLRST / RESTART}][,{NUMBLK = nnn / NUMREC = nnnnn / QCPOOL}]
[,SHARE][,TWA];

Note : If neither NUMBLK, NUMREC nor QCPOOL are specified, then the queue is a disk queue of 32767 blocks.

Parameter Descriptions

name
- ranges from 1 thru 4 alphanumeric characters and uniquely defines the queue within the network.
If the queue is a terminal queue, this external name must be identical with the name given to the terminal, see TERMNL command.

BREAK
- applicable only to program queues on which the MCS COBOL application will be notified of status concerning,
  - asynchronous events such as BREAK and RVI
  - terminal status changes such as connection and disconnection.

  For a detailed description of notifications and application program interface, see Section III.

CLOSE
- specifies that the queue is initially disabled, in which case, the first application to use the queue must first enable it, i.e., using the ENABLE verb, in order to allow data transfer between the queue and its associated terminal.

CTLRST
- applicable only to terminal disk queues enabling controlled restart from "step abort" or "system crash" using the recovery protocol defined in Section III.

NUMBLK
- number of memory blocks for the exclusive use of the declared queue.
The size of the core block in bytes is defined by the QCBLKSZ parameter in the GENCOM command.
The usable size of the core block is (QCBLKSZ - 5).
The number of core blocks is defined by the algorithm,

$(\Sigma NUMBLK + QCPOOL) \times QCBLKSZ \leqslant 65535$

For QCPOOL, see GENCOM command.

NUMREC
- number of blocks in the disk queue file for the exclusive use of the declared queue.
For block size, see QDBLKSZ parameter in the GENCOM command.
The usable size of the disk block is (QDBLKSZ - 5).
The number of blocks is defined by the algorithm,

ΣNUMREC $\leqslant$ 32767 , see "Tuning the Network" later in the section.

QCPOOL
- defines the queue as a memory queue which is to share the core queue pool reserved by the QCPOOL parameter in the GENCOM command.
All queues qualified by the QCPOOL option in their respective QUEUE commands share the same common core pool.
The option enables a saving of memory especially in the case of interactive applications where traffic is not heavy enough to fill all queues at the same time, in which case, the same memory space can be used for at least more than 1 queue.
The user must be careful since there is no protection for queues sharing the common core pool against an application exhausting the pool by its large output and thereby blocking other queues. Such an application should use a mix of pooled queues and queues defined by the NUMBLK parameter.
For selecting the type and size of queues, see "Tuning the Network" later in the section.

RESTART
- applies only to disk queues, i.e., program queues and terminal queues, enabling recovery from "step abort" or "system crash" by queue "roll back".

SHARE
- applies only to program queues thereby allowing 2 or more MCS COBOL applications to share the same queue defined either with input or input and output capability by their QASSIGN JCL statement.
The option is useful in establishing interjob communication and must not be used if the program queue is to be ASSIGNed as a subqueue, see $QASSIGN in Section III.

TWA
- applies only to program queues thereby enabling dialog between the application and the terminal according to the following procedure when either has the right of transmitting.
The application:
  - Either 1 or more SEND WITH EMI can be executed whereby the application retains the right of transmitting
  - Or a SEND WITH EGI is executed whereby the application gives up its right of transmitting and transfers it to the terminal.

The terminal:
Even when the terminal has the right of transmitting, the application is master and can override this right.
At the end of the message transmitted by the terminal, the right of transmitting passes to the application.

Examples

1) QUEUE DSK1,BREAK,RESTART,TWA;

This command declares a disk queue named DSK1 which allows:

- BREAK
  The application is notified of every asynchronous event coming from terminals associated with the queue.
- RESTART
  "Roll back" for recovery and restart is enabled even in the case of BTNS abort.
- TWA
  Dialog between the application and the terminals to which it is connected is enabled.

2) QUEUE TER1,QCPOOL;

This command declares a terminal core queue TER1 which is to share the core queue pool reserved by the GENCOM command with other queues qualified by the QCPOOL option in their respective QUEUE commands.

STATN

Definition

The STATN command defines a station attached to a polled line and must be immediately followed by TERMNL commands defining the terminals composing the station, before the next station on the line can be defined.
The STATN command is required in every case except for the TTY line procedure.

Format of Command

STATN name,station-index [,CLOSE];

Parameter Descriptions

name
- ranges from 1 thru 4 alphanumeric characters and uniquely defines the station in the network.

station-index
- an integer ranging from 1 thru 32 defining the STI by which the station is polled.
  The STI is the hardware address of the station configured in the SRST and is unique for each station on a given line.

CLOSE
- specifies that BTNS must not poll the station until an RT network control command is issued in order to start dialog with the station.

Examples

1) STATN STA1,4;
   This command defines a station named STA1 with an STI of 4 which has been previously specified in the POLLIST parameter of the associated LINE command.

2) STATN STA2,1,CLOSE;
   This command defines a station named STA2 with an STI of 1 which will not be polled until an "RT STA2" network control command has been issued.

Definition

The TERMNL command defines the terminal as a uniquely addressable unit in the network and specifies the characteristics with which it is to function. Except for the TTY line procedure, TERMNL commands defining the terminals composing the station must follow immediately the associated STATN command. For the TTY line procedure, for which the STATN command is not applicable, the TERMNL command must follow immediately the associated LINE command.

Format of Command

```
TERMNL name,TYPE={terminal-type|SLAVE},STYPE=subtype [,ADD="hh"]
[,ASSIGN=name][,AUTO][,BLOCKING][,CLOSE][,CONTROL][,IM={NL|MK|UN}]
[,LINELG=nnn][,LLENGTH=nnnn][,NBLOCKS=nnnn][,OM={NL|UN}]
[,SBKLG={80|nnn}][,SYSHEAD={"OD25"|"hhhhhhhh"}];
```

Parameter Descriptions

name
- ranges from 1 thru 4 alphanumeric characters and uniquely defines the terminal in the network.
  The reserved terminal name OPER defines the network control terminal having the following characteristics,
  - not connected over a switched line
  - of a "terminal-type" other than
    - HL62
    - HL64
    - HL66
  - of a "subtype" restricted to
    - KCT, keyboard with screen
    - KPR, keyboard with printer
  - not declared as a member of a LINE or STATN specified with the CLOSE option.

  The network control terminal may be connected to an application. The system console is the network control terminal by default, i.e., if no terminal has been so designated.

TYPE
- defines the type of standard terminal specific to the line procedure used. The terminal specified functions as a master terminal; where the terminal is to function as a slave, the parameter SLAVE is defined.
  See Section II "Log-on Procedures" for further details.

TYPE (continued)

Types of Standard Terminals				
Line Procedures				
TTY		VIP Asynchronous	VIP Synchronous	BSC
TN300 TN1200 TTU8124* TTU8126* TTY33 TTY35 TTY37 TTY38 VIP7100 VIP7200		VIP765	TTU8221* VIP775 VIP785 VIP7700 VIP7760	HL62 HL64 HL66

STYPE
- specifies the characteristics with which the terminal described is to function such as,
  - its ability to receive system messages
  - its I/O capability
  - its default address.

  The possible choices of terminal subtype defining such characteristics are,
  - CAS Cassette
  - CPU Central Processor
  - CRT Screen
  - KB Keyboard
  - KCT Keyboard with Screen
  - KPR Keyboard with Printer
  - PRT Printer

  The I/O capability of the terminal is overridden by the declaration of its subtype, i.e., if a keyboard printer terminal is declared PRT, then its keyboard will not be accessible to BTNS.

	CAS	CPU	CRT	KB	KCT	KPR	PRT
I/O Capability	I/O	I/O	output	input	I/O	I/O	output
System Messages	no	no	yes	no	yes	yes	yes

*For TWU/PRU1001 or TWU/PRU1003 use "TTU8124" keyword parameter.
For TWU/PRU1005 use "TTU8126" keyword parameter.
For TWU/PRU1901 use "TTU8221" keyword parameter.

SUBTYPE (continued)

Allowed TYPE/SUBTYPE Combinations							
TTY Line Procedure							
	CAS	CPU	CRT	KB	KCT	KPR	PRT
TN300 TN1200	n	n	n	y	n	y	y
TWU/PRU1001* TWU/PRU1003* TWU/PRU1005*	n	n	n	y	n	y	y
TTY33 TTY35 TTY37 TTY38	n	n	n	y	n	y	y
VIP7100 VIP7200	n	n	y	y	y	n	n

VIP Line Procedure							
	CAS	CPU	CRT	KB	KCT	KPR	PRT
TWU/PRU1901*	n	n	n	y/79	n	y/79	y/79
(1)VIP765 (1)VIP775 VIP785	n	n	y/79	y/79	y/79	n	n
VIP7700	y/97	n	y/79	y/79	y/79	n	y/88
VIP7760	n	n	y/79	y/79	y/79	n	y/88

*For TWU/PRU1001 or TWU/PRU1003 use "TTU8124" keyword parameter.
For TWU/PRU1005 use "TTU8126" keyword parameter.
For TWU/PRU1901 use "TTU8221" keyword parameter.

SUBTYPE (continued)

Allowed TYPE/SUBTYPE Combinations (continued)							
	BSC Line Procedure						
	CAS	CPU	CRT	KB	KCT	KPR	PRT
HL62 HL64 HL66	n	y	n	n	n	n	n

(1) For VIP765/VIP775, the default address of 88 is generated for the second screen declared on a station.

NOTE : In the above table, where the combination is allowed, i.e., "y" indicated in the appropriate column, the default address, if applicable, is given as a hexadecimal value.

ADD

- a 2-hexadecimal digit value enclosed in quotation marks to specify the hardware address of a terminal compatible with a standard terminal, see TYPE.
Values are determined at the time of installation and are obtainable from the Field Engineering Service.
The parameter is specified where the terminal address is significant and must be other than the default address generated by BTNS from the TYPE/STYPE combination.

ASSIGN

- dedicates the terminal to a particular use identified by a name ranging from 1 thru 4 alphanumeric characters which must be,
  - either the program queue declared by the QUEUE command
  - or the VCAM subsystem (IOF or TDS) declared by a DCTAP command.

The assignment can be altered by the MTT network control command.
NOTE : If the terminal is declared SLAVE, then ASSIGN must not be specified.

AUTO

not applicable if the terminal is dedicated (ASSIGN) to IOF subsystem.
ensures that BTNS automatically logs on a receive-only terminal or a master terminal, i.e., not declared SLAVE.

BLOCKING

- MCS oriented parameter which allows MAM to keep track of the terminal line size (LLENGTH) in order to generate a new-line or carriage-return/line-feed character string before the first line of each message or whenever a line is filled.
See LLENGTH.

CLOSE

- specifies that BTNS must not initialize traffic to the terminal declared until an RT network control command is issued in order for the terminal to accept a connection request from the application.
The parameter does not apply to the network control terminal designated OPER, see "name" of the TERMNL command.

CONTROL

- specifies that the log-on procedure for the terminal declared is subject to verification of the following parameters against the system catalog, viz.,
  - user identification
  - password
  - ~~account name~~, PROJ/BILLING

The parameter can only be specified for a master terminal, i.e., not declared SLAVE, and must be specified if the terminal is to work with IOF or TDS.
For a detailed description of the log-on procedure using the system catalog, see Section II and Terminal Operations GCOS Manual.

IM

- MCS oriented parameter which specifies the format of input data passed from the terminal to the MCS COBOL application, see Section IV.
  - NL denotes normal mode which is the default value and specifies that,
    - headers and control characters are not passed to the application
    - character encoding and repeat functions are performed
    - other characters, including standard marks, are passed without translation.
  - MK denotes mark mode and specifies that,
    - headers are passed as marks of the form ><U03abc
    - all hexadecimal control characters are translated into their corresponding marks
    - character encoding and repeat functions are performed
    - other characters, including standard marks, are passed without translation.
  - UN denotes unedited mode and specifies that,
    - headers are passed as marks of the form ><U03abc
    - other characters, including control characters and marks, are passed without translation.

LINELG

- a 3-decimal digit value which defines the number of characters in the physical line length of the terminal declared, see "MCS Oriented Parameters". For instance, this parameter would be used by the IOF VCAM subsystem to adapt its message length to the line length of the terminal receiving the message.

LLENGTH

- MCS oriented parameter which specifies the number of characters in the logical terminal line size to be used for automatic editing when the option BLOCKING is specified.
Values range from 5 thru 9999.
For default values, see "MCS Oriented Parameters" at the end of the TERMNL command.

NBLOCKS

- MCS oriented parameter which defines the number of logical lines to be accepted in each message sent to the terminal declared. The number of characters for each logical line is determined by the LLENGTH parameter. The length of the message transmitted is defined by the algorithm,

  (NBLOCKS x LLENGTH) = maximum length

  The part of the message surpassing this limit is truncated.
  The number of logical lines ranges from 1 thru 9999.
  For default values, see "MCS Oriented Parameters" at the end of the TERMNL command.

OM

- MCS oriented parameter which specifies the format of output data passed from the MCS COBOL application to the terminal, see Section IV.
  - NL denotes normal mode which is the default value and specifies that,
    - the user provides a header for the VIP line procedure in the mark form ><UO3abc
    - character encoding and repeat functions are performed
    - hexadecimal control characters are translated into character encoding mark form ><Cab
    - standard marks specific to the terminal are translated into hexadecimal characters
    - all other characters are passed without translation.
  - UN denotes unedited mode and specifies that,
    - the user provides a header for the VIP line procedure in the mark form ><UO3abc
    - other characters, including control characters and marks, are passed without translation.

SBKLG

- applicable only for the BSC line procedure to specify the sub-block size in characters to be generated by BTNS when transmitting a character string.
  The sub-block mode can be used in either transmission, viz.,
  - normal, in which checks are made for transmission and device control characters in order to distinguish these from graphic characters
  - transparent, in which no checks are made, i.e., all characters are treated as graphic characters.

  Values range from 50 thru 512.
  The default value is 80 characters.

SYSHEAD

- specifies the character string that must precede any system message sent by BTNS to the terminal declared.
  The character string is expressed as a hexadecimal value enclosed in quotation marks and ranges from 1 character (2 hexadecimal digits) thru 4 characters (8 hexadecimal digits).
  The default value is 2 characters (4 hexadecimal digits), viz., "OD25" which defines respectively carriage-return/line-feed.
  An "FF" value specifies that no character string must be inserted in front of system messages.
  For format of network messages, see Terminal Operations GCOS Manual.

MCS Oriented Parameters

These parameters are only effective where the terminal functions solely thru MCS, i.e., when not connected to a VCAM subsystem.

Default Values for MCS Parameters

TTY Line Procedure			
terminal	LINELG	LLENGTH	NBLOCKS
TN300 TN1200	118	118	100
TWU/PRU1001 TWU/PRU1003 TWU/PRU1005	132	132	100
TTY33 TTY35 TTY37 TTY38	72	72	100
VIP7100 VIP7200	80	80	100

VIP Line Procedure *			
terminal	LINELG	LLENGTH	NBLOCKS
TWU/PRU1901	132	960	1
VIP765 VIP775	46	1012	1
VIP785	92	2024	1
VIP7700 VIP7760	80	1920	1

* For all VIP terminals, new-line function is automatic. LLENGTH covers screen capacity.

BSC Line Procedure			
terminal	LINELG	LLENGTH	NBLOCKS
HL62 HL64 HL66	na	na	na

na = not applicable

MTT VIP1 IOF → modify terminal for IOF

$CONSOLE KURT, RMS

↖ SYS.HSLLIB

Examples

```
TERMNL TRO1,TYPE=VIP7700,STYPE=KCT,IM=MK,OM=MK;
```
This command declares a keyboard/screen VIP7700 named TRO1 which is to function in input/output mark mode when used with a program queue via MCS.

```
TERMNL OPER,TYPE=TN300,STYPE=KPR,ASSIGN=TDS,AUTO,CONTROL;
```
This command declares a keyboard/printer TN300 as the network control terminal, i.e., identified to the system by the name OPER, and specifies,
- a master terminal of TDS
- automatic log-on for immediate connection to TDS when loaded
- CONTROL
  Security controls and access rights are to be verified against the system catalog.

EXECUTING CNC

Executing CNC is dealt with in terms of,

- Command Sequencing
- When to Generate a Network
- Run-time Prerequisites
- Run-time JCL
- Simulation Facility

Command Sequencing

The sequence of NDL commands is significant except for the COMM command. The rules of command sequencing are,

- COMM - can be located anywhere within the deck except before the GENCOM command.
- DCTAP - is required when a version of the VCAM subsystem is to be defined. The command can appear anywhere after the GENCOM command so long as it does not disrupt a LINE-STATN-TERMNL command sequence.
- GENCOM - is mandatory and must be the first command specified.
- LINE - is required for each communications line in the network. Following the LINE command must be all STATN and TERMNL commands to define the terminals on the line.
- QUEUE - is required for each physical queue to be defined in the network. The command can appear anywhere after the GENCOM command so long as it does not disrupt a LINE-STATN-TERMNL command sequence.
- STATN - is applicable to all line procedures except the TTY line procedure, i.e., the command only applies to a polled line. For the VIP and BSC line procedures, more than 1 STATN command can be associated with each LINE command.
- TERMNL - is required to identify each terminal connected on the line. For the TTY line procedure, only 1 TERMNL command is associated with each LINE command.

When to Generate a Network

A network is generated using the H_CNC step under the following conditions,

- There has been no previous network generated on the system disk to be used for communications sessions.
- A previously generated network was destroyed in backing store (CTVF file) either thru a disk failure or on system restart with the CLEAN option.
- A current network needs to be updated, for instance,
  - terminals, lines or queues to be added
  - a parameter in the NDL command(s) such as buffer unit size or line speed to be altered.

Acceptable Sequence of NDL Commands

(only mandatory parameters are indicated)

```
GENCOM name...;
   [QUEUE name...;
    :
    :                ]
   DCTAP name...;
   [QUEUE name...;
    :
    :                ]
```

TTY Line Procedure

```
   LINE LNnn...;
      TERMNL name,TYPE=terminal-type,STYPE=subtype...;
   LINE LNnn...;
      TERMNL name,TYPE=terminal-type,STYPE=subtype...;
    :
    :
```

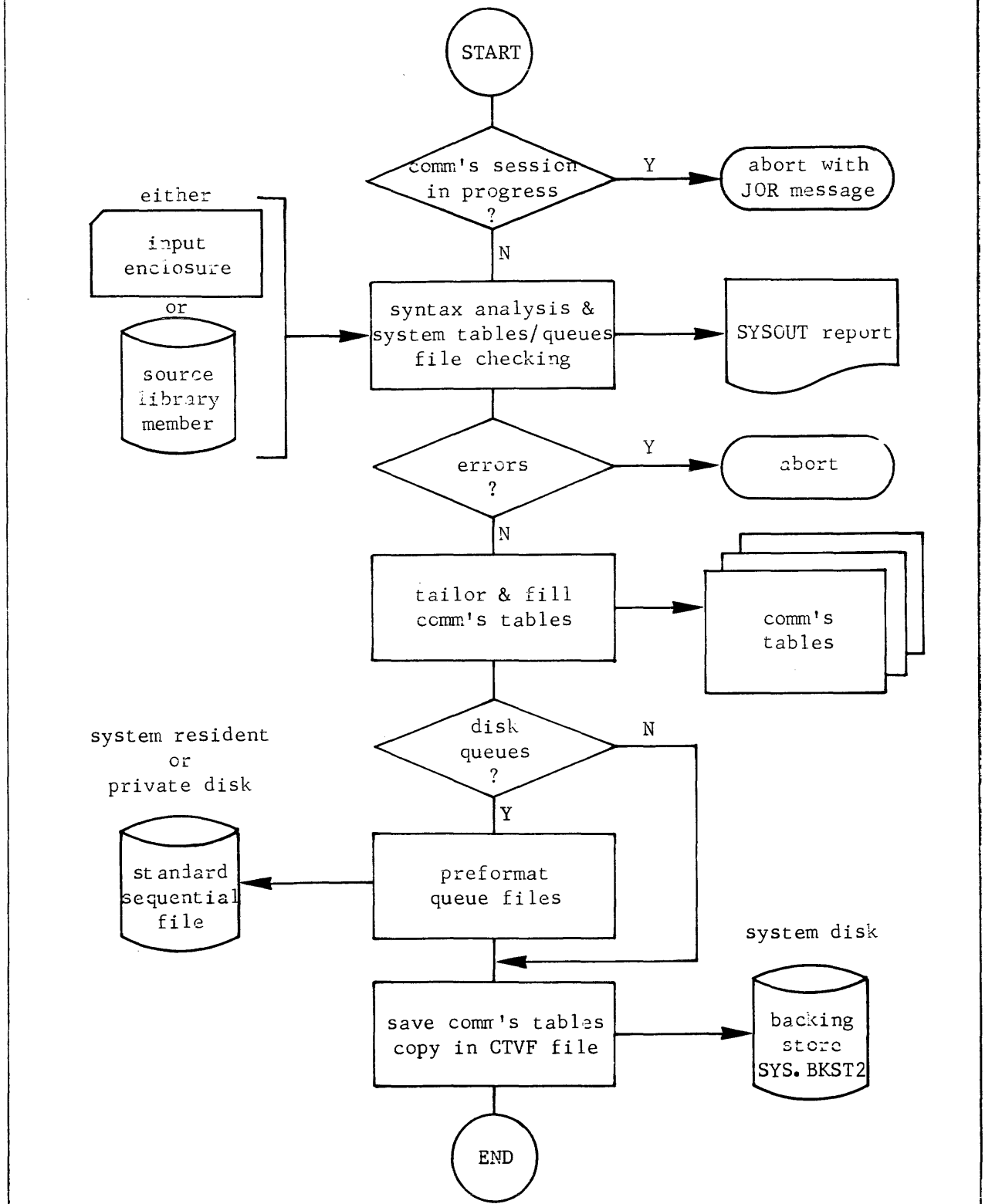
```
   [QUEUE name...;
    :
    :                ]
```

VIP, BSC Line Procedures

```
   LINE LNnn...;
      STATN name,station-index...;
         TERMNL name,TYPE=terminal-type,STYPE=subtype...;
         [TERMNL name,TYPE=SLAVE,STYPE=subtype...;
          :
          :                                       ]
         TERMNL name,TYPE=terminal-type,STYPE=subtype...;
          :
          :
      STATN name,station-index...;
         TERMNL name,TYPE=terminal-type,STYPE=subtype...;
          :
          :
     :
     :
   LINE LNnn...;
    :
    :
```

```
   [QUEUE name...;
    :
    :                ]
```

CNC Generation Main Flow



Run-time Prerequisites

All the following prerequisites must be met to execute CNC, namely,

- No communications component must be currently active, i.e.,
  - BTNS
  - MAM application program(s)
  - VCAM subsystem which can be one of:
    - IOF
    - TDS
  - QMAINT utility
  - Another CNC session

  Any attempt to execute the H_CNC utility will result in a "step abort" with the JOR message CNO4 TELECOMMUNICATIONS SESSION IN PROGRESS, see Appendix C.

- MAM and/or VCAM must not be preinitialized thru the PMM "preload main memory" OCL command, see System Operation : Operator Guide.

- If disk queues are specified in the NDL description, the disk queue file must be preallocated, see Section III "Preallocating a Disk Queue File".

  On execution of the H_CNC utility, volume premounting will be requested thru the standard device management interface, if the volume containing the queue file is not already mounted.

  On successful completion of the H_CNC utility, the disk queue file will be preformatted.

- The following conditions must be fulfilled, namely,

  - The lines declared in the NDL description must be present and connected in the SRST of the system

  - The STI (station index) values declared in the STATN commands must be consistent with the URP firmware generation values.
    If an STI value specified does not correspond to an existing station, each time that the non-existent station is polled, "time-out" will occur thereby causing
    - response time to be increased
    - the actual station for which the wrong STI value has been given, never to be polled.

  The conditions cited above must be checked with the Field Engineer.

CNC Run-time JCL

commands retrieved from an input enclosure

```
$JOB job-name,USER=user-name,PROJECT=project-name;
STEP H_CNC,SYS.HLMLIB [,OPTIONS='SIMU'];
ASSIGN H_CR,*input-enclosure-name;
[                applicable to disk queue file                ]
[ ASSIGN H_QC_FMS,external-file-name,SHARE=FREE,              ]
[        {RESIDENT                                          } ]
[        {DEVCLASS=device-class-name,MEDIA=media-name       } ]
[                                                             ]
ENDSTEP;
$INPUT input-enclosure-name [,TYPE=DATASSF];
                            {                 }
                            {  NDL commands   }
                            {                 }
$ENDINPUT;
$ENDJOB;
```

commands retrieved from a source library member

```
$JOB job-name,USER=user-name,PROJECT=project-name;
STEP H_CNC,SYS.HLMLIB [,OPTIONS='SIMU'];
ASSIGN H_CR,external-file-name,SUBFILE=member-name,
       DEVCLASS=device-class-name,MEDIA=media-name;
[                applicable to disk queue file                ]
[ ASSIGN H_QC_FMS,external-file-name,SHARE=FREE,              ]
[        {RESIDENT                                          } ]
[        {DEVCLASS=device-class-name,MEDIA=media-name       } ]
[                                                             ]
ENDSTEP;
$ENDJOB;
```

Syntax for CNC run-time JCL :

- $STEP statement
  - H_CNC is the system load module in the system load-module library SYS.HLMLIB and must be specified as given.
  - OPTIONS="SIMU" specifies that the generation is to be used for simulation purposes, see "Simulation Facility" later on in the section.
- $ASSIGN statement (1st. ASSIGN in sequence)
  - H_CR is the system-reserved internal-file-name for the file containing the NDL commands, either as an input enclosure or as a source library member, and must be specified as given.
- $ASSIGN statement (2nd. ASSIGN in sequence)
  - H_QC_FMS is the system-reserved internal-file-name for the file preallocated for disk queues.
  - SHARE=FREE denotes that the disk queue file is to be shared by BTNS and the active MCS COBOL application program(s) and must be specified as given.
  - RESIDENT is specified if the disk queue file has been preallocated on a resident disk to be accessed simultaneously by MCS COBOL application programs and system utility programs operating on the file, such as FILPRINT.
- $INPUT statement
  - TYPE=DATASSF is used to provide for the cards in the input enclosure being in the form of DATA or SSF (system standard format).

Example of Network Description as an Input Enclosure

```
$JOB GENNET,USER=UNAME,PROJECT=DEPT;
STEP H_CNC, SYS.HLMLIB;
ASSIGN H_CR,*NETIN;
ASSIGN H_QC_FMS,DQUEUE,DEVCLASS=MS/M400,MEDIA=VOL1,SHARE=FREE;
ENDSTEP;
$INPUT NETIN;
GENCOM SD20;
LINE LN12;
STATN ST09,9;
TERMNL V771,TYPE=VIP7700,STYPE=KCT,CONTROL;
LINE LN11,SPEED=H;
TERMNL ROSY,TYPE= TWU1003,STYPE=KPR,CONTROL;

LINE LN01;
STATN S761,1;
TERMNL VIP1,TYPE=VIP7700,STYPE=KCT,CONTROL;
DCTAP TDS,SIMU=4;
QUEUE PROG NUMBLK=50;
QUEUE V771,RESTART;
$ENDINPUT;
$ENDJOB;
```

Syntax for example of network description as an input enclosure :

- The job generates a network from the description contained in the input enclosure NETIN.
- Since disk queues are defined in the network description, the previously allocated disk queue file, DQUEUE, is assigned.

Example of Network Description as a Library Member

```
$JOB TCOMMS,USER=UNAME,PROJECT=WAGE;
STEP H_CNC,SYS.HLMLIB;
ASSIGN H_CR,SOURCE,DEVCLASS=MS/M400,MEDIA=VOL2,SUBFILE=NETWK;
ENDSTEP;
$ENDJOB;
```

Syntax for example of network description as a library member :

- The job generates a network from the description stored previously as a member NETWK in the source library SOURCE residing on the volume VOL2.

Simulation Facility

The simulation facility is activated thru the SIMU option in the $STEP statement.

It allows the user to run CNC in order to provide the syntax analysis of a new or updated network description without a communications session taking place after network generation.

When the simulation facility is used, the run-time prerequisites to execute CNC do not apply, i.e.,

- CNC can run concurrently with any combination of communications components and application programs interfacing with these components.
- MAM and VCAM can be pre-initialized.
- Values specified in the NDL commands do not have to be consistent with SRST values.

If disk queues are specified, then the disk queue file must be assigned. This is needed to compute the size of system tables for the queues based on the size of the preallocated file and the type of disk (volume) containing it. The file specified can be one currently used by an active communications session.

The principal effects of simulation are,

- The communications tables are not generated
- The disk queue file is not preformatted.

TUNING THE NETWORK

Tuning the network enables the best use of the communications system in terms of reducing response time and optimizing memory occupancy.

Rules for tuning fall into 3 broad categories, namely,

- BTNS/URP Tuning
- MAM Tuning
- Optimizing MCS COBOL Applications, see Section III.

BTNS/URP Tuning

- network definition

  - The segment containing the network description and permanently accessed by BTNS may be set resident in memory thru the PMM OCL command. The segment should be made as small as possible, see Appendix G.

  - Corresponding to several different communications sessions, as many cataloged network descriptions should be written. The CNC step on an average of 1 minute to generate a given network allows rapid switching from one network operation to another.

    The constraint in changing the network is to stop BTNS and any other communications program. Once the network has been generated, BTNS takes about 30 seconds to be initialized.

- BTNS buffer pool size

  Points to note :
  - A message is output from a queue to the appropriate terminal connected to a MAM application program, only if there are enough units in the BTNS buffer pool to contain the message.
  - On input, units in the BTNS buffer pool are dynamically linked to receive data from terminals connected to MAM application programs and to VCAM subsystems.
  - An overload situation occurs during a peak in traffic for too small a buffer pool; BTNS automatically retries the transmission.
  - The BTNS buffer pool occupies a segment of up to 64K bytes and is created at BTNS start-up and deleted on termination.
  - see "Considerations for buffer pool sizes" described later on.

  Specifying the BTBFSZ parameter, see GENCOM command :
  The unit size in bytes is determined by the average message length which in turn depends on,
  - Application requirements
  - Terminal characteristics.

  Specifying the NBBTBF parameter, see GENCOM command :
  The buffer size specified by the BTBFSZ and NBBTBF parameters represents the maximum space required to contain 1 message per transmission line, as BTNS may have to service ALL transmission lines simultaneously.
  The minimum number of units necessary is as follows:
  - 2 units for each non-VIP transmission line, i.e., TTY and BSC
  - 3 units for each VIP transmission line.

. VCAM buffer pool size

Points to note :
- The VCAM buffer pool is generated if VCAM subsystems are declared by at least one DCTAP command in the NDL description.
- The buffer pool is used to output messages to the appropriate terminal connected to the version of the VCAM subsystem declared.
- The VCAM buffer pool occupies a segment of up to 64K bytes and is created at BTNS start-up and deleted on termination.
- see "Considerations for buffer pool sizes" described below.

Specifying the DABFSZ parameter, see GENCOM command :
The unit size in bytes is determined by the average message length composing the response to the terminal which in turn depends on,
- Application requirements
- Terminal characteristics.

Specifying the NBDABF parameter, see GENCOM command :
The buffer size specified by the DABFSZ and NBDABF parameters represents the maximum space required to handle output data to however many terminals which are interactive at any one time.
Factors to consider in determining the buffer pool size are,
- 1 message for each active terminal in order to avoid suspending the VCAM subsystem.
- For the IOF and TDS versions of the VCAM subsystem, where 1 enquiry (input) generates 1 response (output) as a rule, the contents of the buffer pool is automatically regulated by the rate of input allowed by BTNS and the characteristics of the line(s).

. considerations for buffer pool sizes

Optimum values for buffer size parameters can be adjusted for as the network is used.
Factors influencing optimum values are,
- Transmission lines are not generally active simultaneously; sharing buffer space is to be considered.
- Characteristics of the network being as follows,
  - ° Speed of the line :
    The faster the speed of transmission, the greater the thruput of the buffer pool and consequently the smaller the buffer pool, and the quicker the "turn-around".
  - ° Total traffic of the network :
    The greater the traffic, the greater the buffer pool.
    Increase in total traffic is caused by,
    - . The larger number of terminals active in the network
    - . The type of application; "remote batch" applications impose a heavier traffic on the network than "interactive" applications.
  - ° Message length :
    The buffer unit size must range from ¼(N) to N, where N is the average message length.

If buffer size parameters are not specified, equal length pools are generated. The default size of these pools is given in Appendix G.

Using the results obtained in the BTNS JOR after several communications sessions, buffer pool sizes can be adjusted for subsequent sessions, see Appendix I.

Maximum Message Length of Terminals		
Terminal	Line Procedure A=Asynchronous S = Synchronous	Message Length (bytes)
HL62 HL64 HL66	BSC	2048
TN300 TN1200	TTY	120
TWU/PRU1001 TWU/PRU1003 TWU/PRU1005	TTY	256
TWU/PRU1901	VIP(S)	1920
TTY33 TTY35 TTY37 TTY38	TTY	120
VIP765	VIP(A)	960
VIP775	VIP(S)	960
VIP785	VIP(S)	1920
VIP7100 VIP7200	TTY	120
VIP7700 VIP7760	VIP(S)	1920

• polling algorithms

Points to note :

- Setting up the polling list:
  The polling list for stations on a multipoint line is defined by 2 parameters of the LINE command, namely,
  ° POLLIST which lists the stations by their respective STIs (station index) defined in the STATN commands following the LINE command
  ° LINPLG which defines the restart of polling after successful data entry on a given station, as follows,
    • If specified, polling resumes at the beginning of the polling list, i.e., linear
    • If not specified, polling resumes at the next station consecutive to the given station, i.e., round-robin.

- When BTNS is initialized thru the ST network control command, the polling list is loaded by BTNS into the URP.
  Polling is handled by URP firmware.

- Modifying the polling list:
  The polling list can be modified by the following network control commands, as follows,
  ° HT to inhibit the station if it is known that the station is;
    • not used to send or receive, e.g., closed down
    • out-of-order due to a malfunction
    • unavailable if "time-out", as specified by the value given to TOLEN, has occurred.
  ° RT to restore the station to the polling list,
    • if HT was previously issued
    • if the STATN command was specified with the CLOSE option at the time of BTNS start-up.
  ° MTP to modify a part or the whole of the polling list.

Specifying the POLLIST and LINPLG parameters :

- Example 1 : Equal frequency, round-robin polling

  POLLIST = (1 , 2 , 3 , 4 , 5)

  All stations 1, 2, 3, 4 and 5 are as frequently polled.

- Example 2 : Biased frequency, round-robin polling

  POLLIST = (1 , 2 , 1 , 3 , 1 , 4 , 1 , 5)

  Station 1 is polled 4 times more frequently than stations 2, 3, 4 and 5.

- Example 3 : Descending frequency, linear polling

  POLLIST = (1 , 2 , 3 , 4 , 5),LINPLG

  The stations are polled in order of descending frequency, namely, station 1 being polled most frequently, while station 5 being the least polled.

- Example 4 : Biased frequency, linear polling

  POLLIST = (1 , 1 , 1 , 2 , 3 , 4 , 5),LINPLG

  The polling bias is heavily in favor of station 1.

- considerations for station declaration
  - The effects of declaring 1 station per line are,
    - A larger number of lines are used
    - A larger network description segment is needed since approximately 200 bytes per line are needed, see Appendix G
    - The lines will not be used to their optimum capacity
    - Installation costs will be increased for either or both reasons,
      - For equipment already installed, running and maintenance costs will be greater
      - At the installation planning stage, more equipment than is necessary will be installed.
  - When using the multipoint facility, i.e., declaring more than 1 station per line, the gain in reducing the number of lines, must be considered against
    - The degradation of response time which is caused by
      - The "dead-time" of the polling mechanism
      - The time for transmitting a message, which depends on its length
    - The type of application, i.e., interactive applications are best suited for a multipoint network.

- specifying the PRIRA parameter, see LINE command :

  BTNS executes the following actions according to the parametric values specified.

  - PRIRA = IN/nnnn
    - When an output request arrives from MAM or VCAM,
      - The request is executed
      - The line is then polled for as many times as specified by "nnnn" in the parameter, each poll corresponding to 1 scanning of the station index (STI) in the polling list, before
      - The next output request is accepted.
    - If the line is inactive, i.e., no output pending, a continuous polling of the line, i.e., loop in the URP firmware, is executed.
  - PRIRA = OU/nnnn
    - When output requests are pending, i.e., enqueued messages,
      - As many messages as specified by "nnnn" in the parameter are output, before
      - The line is polled again, the poll being the scanning of 1 station index in the polling list.
    - If no messages are to be output, the line is polled continuously.
  - {QRRO|QLIN}

    see "Parameter Descriptions" of LINE command.

- choosing between disk queues and memory queues
  - When disk queues are to be used:
    - ° For the checkpoint/restart capability in case of "step abort" or "system crash"
    - ° For file transfer and data collection, where the data obtained is used as input information to application programs to be run after the communications session.
  - When memory queues are to be used:
    - ° For critical response time in reducing "turn-around".
  - When both disk queues and memory queues are to be used:
    - ° For heavy traffic loads, e.g., BSC file transmission or for a large network,
      - . Memory queues can be used to receive data
      - . Disk queues can then be used to store the data.
  - Example of using both types of queues:
    - ° Installation environment
      - . 20 VIP7700 keyboard-screen terminals
      - . 4 printers to produce hard copy
    - ° Application usage
      - . VIP terminals are used in conversational mode to enter bills on the basis of 1 enquiry (input) to 1 response (output)
      - . The printers are used to print out the bills entered on the VIP terminals and therefore many messages may be generated for output to the printers.
    - ° Choice of queues
      - . Memory queues are to be used for the VIP terminals and the application program for rapid "turn-around"
      - . Disk queues for the printers since hard-copy is slower to produce than screen operations.

- structure of the queued message

  The internal structure of a message when queued by MAM is the same in memory as it is on disk. The message is assembled in fixed length blocks specified by the user.

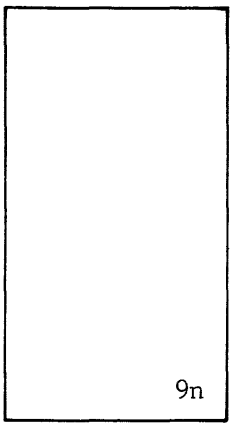
  The blocks are used as follows,

  - TCB (text control block)
    - ° Each TCB is structured as follows,
      - . 5 bytes are used to provide the link to its appropriate HCB
      - . The remaining bytes to contain the actual text of the message.
  - HCB (header control block)
    - ° An HCB occupies the same block size as a TCB and is structured thus,
      - . HCB1 : 50 bytes to contain control information and the link to HCB2, The remaining bytes to link to TCBs, each link being 4 bytes.
      - . HCB2 : 4 bytes to link HCB2 to HCB3, The remaining bytes to link to TCBs, each link being 4 bytes.
      - . HCB3 : as for HCB2, but the first 4 bytes are "lost".

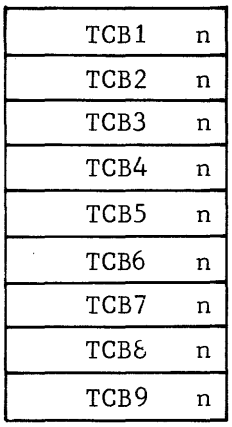
Optimizing Queue Size

CASE 1

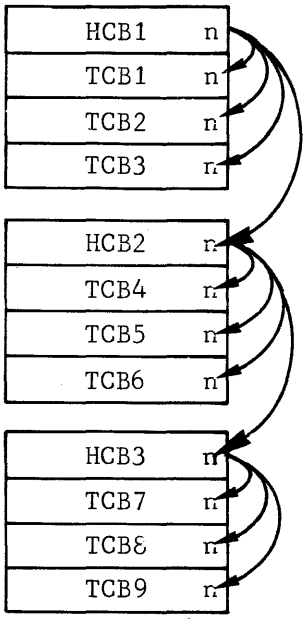
total block size to contain message = 12n bytes
1 HCB is assumed for every 3 TCBs



message of 9n bytes in working area of program



division of message into 9 data blocks each of n bytes

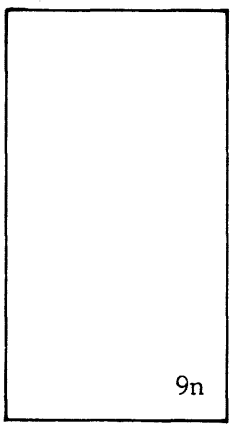


structured message with links between header and text control blocks

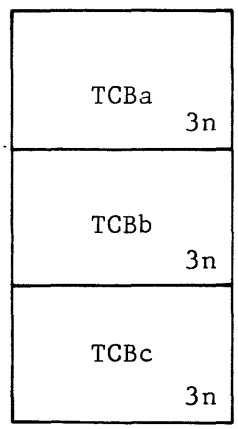
In case 1, the message is split unduly owing to the small size of the blocks declared. The number of links required slows down storage & retrieval, and hence increases response-time.

CASE 2

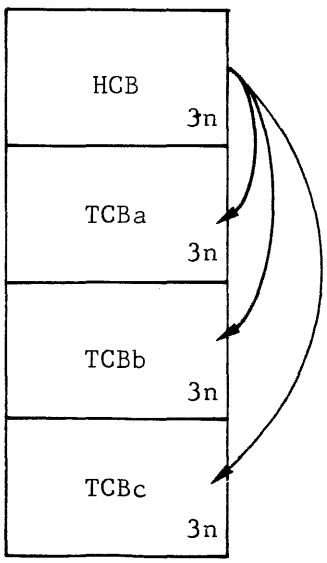
total block size to contain message = 12n bytes



message of 9n bytes in working area of program



division of message into 3 data blocks each of 3n bytes



structured message with links between header and text control blocks

In case 2, the blocks are larger in size. The number of links are fewer compared to case 1. Storage and retrieval are more rapid, thereby decreasing the response time.

The number of blocks needed to queue a message depends on,
- the length of the message
- the size of the block in bytes defined in the GENCOM command as follows,
  - ° QDBLKSZ for disk queues
  - ° QCBLKSZ for memory (core) queues.

. calculating the number of blocks

- The number of TCBs is given by the algorithm

$$N = \frac{\text{message length}}{(B - 5)} \; (+ 1 \text{, for any remainder}), \text{ where } B = \text{block size}$$

- The number of HCBs is given by the algorithms
  - ° For 1 HCB, i.e., HCB1,

    $(N \times 4) < (B - 50)$, where N = number of TCBs, and B = block size
  - ° For 2 HCBs, i.e., HCB1 and HCB2,

    $(N \times 4) < (2B - 54)$, where N = number of TCBs, and B = block size
  - ° For 3 HCBs, i.e., HCB1, HCB2 and HCB3,

    $(N \times 4) < (3B - 58)$, where N = number of TCBs, and B = block size

. specifying the QCBLKSZ and QDBLKSZ parameters, see GENCOM command :

The value in bytes must be so chosen in order
- to minimize the number of blocks needed to queue the average message, which in turn enables
- to reduce the "overhead" imposed by MAM on BTNS and the application program in terms of
  - ° message processing, i.e., unnecessary "splitting" of messages is costly in CPU time
  - ° the number of I/O operations to access HCBs and TCBs on a disk queue file.

A compromise must be found between
- performance, i.e., the number of blocks to be accessed
- space utilization, i.e., minimising the amount of "lost" space.

Example of optimizing queue size:
Consider an average message length of 980 bytes.

- If 3 TCBs are to be used, then
  - ° The block size must be no less than $(327 + 5) = 332$ bytes
  - ° The "loss" in TCB3 is $(327 \times 3) - 980 = 1$ byte
  - ° The "loss" in HCB1 is $(332 - (50 + (3 \times 4))) = 270$ bytes
  - ° The total "loss" is 271 bytes.
- If 4 TCBs are to be used, then
  - ° The block size must be no less than $(245 + 5) = 250$ bytes
  - ° The total "loss" is in HCB1, which is $(250 - (50 + (4 \times 4))) = 184$ bytes.
- If 5 TCBs are to be used, then
  - ° The block size must be no less than $(196 + 5) = 201$ bytes
  - ° The total "loss" is in HCB1, which is $(201 - (50 + (5 \times 4))) = 131$ bytes.

The more the number of TCBs used, the less the waste but the greater the access time. A reasonable estimate would be a block size between 200 and 300 bytes which in this case happens to be about ¼ of the message length.

Example of optimizing the number of blocks:
Consider a message text to fill a screen of the following characteristics,
- 24 lines
- Each line containing 80 characters.

A block size of 85 bytes has been chosen to contain the entire text per line, i.e., 80 bytes for the text and 5 bytes for the link.

° to reduce response time

Response time is reduced if BTNS can start delivery of the first lines of text before the entire text for filling the screen is queued. In order to do this,

- Each line must be sent as a separate message, i.e., SEND WITH EMI
- The message structure therefore must be
  - One TCB to contain the line
  - One HCB1 to head each TCB
  - 24 TCBs and 24 HCB1s are needed to contain the entire message text, giving 48 control blocks, each of 85 bytes, making 4080 bytes.

The added advantage of specifying the message structure in the way described is security.

° to save space

In order to save space, the entire message text to fill the screen must be queued and transmitted as follows,

- Each line is sent as a portion of the message, i.e., SEND without indicator, up to the last line
- The last line, 24th., is transmitted by SEND WITH EMI, thereby terminating the message text
- The message structure must therefore be
  - 24 TCBs to contain the entire message text
  - One HCB1 to link the first 8 TCBs, i.e., TCB1 thru TCB8
  - One HCB2 to link the next 16 TCBs, i.e., TCB9 thru TCB24
  - 1 HCB1, 1 HCB2 and 24 TCBs are needed to contain the entire message text, giving 26 control blocks, each of 85 bytes, making 2210 bytes.

A saving of 1870 bytes compared to the preceding case has been made at the expense of minimizing response time.

. defining memory queue space

Space for memory queues is restricted to a single segment of 64K bytes.

- The number of memory blocks is given by the algorithm

  $N = \Sigma NUMBLK + QCPOOL$, for NUMBLK see QUEUE command
  for QCPOOL see GENCOM command

- The restriction on the memory queue space to be declared is

  $(N \times QCBLKSZ) \leqslant 65535$, for QCBLKSZ see GENCOM command

- The total memory queue space declared includes
  ° The pool independently reserved for the exclusive use of individual queues, as declared by the NUMBLK parameter of the respective QUEUE commands
  ° The pool, as declared by the QCPOOL parameter of the GENCOM command, to be shared by <u>all</u> queues qualified by the QCPOOL option in their respective QUEUE commands.

- Violating the restriction placed on memory queue space will cause an error during network generation with the message

  ERROR CN0020 SEVERITY 4 SEGMENT TOO LARGE : buffer-pool-name

  For details, see "NDL Error Messages" in Appendix C.

- The actual size of the generated segment for memory queues is provided by the CNC report, see Appendix G.

Estimating the size of the memory buffer pool depends on

- Message length, which has been dealt with previously.

- Traffic load, which comprises
  - ° Time taken to compose the message text to be input
  - ° Time taken to key in the composed message text
  - ° BTNS transfer time
  - ° Application program processing time
  - ° Response time in outputting the reply to the terminal.

  In the case of interactive applications, up to 30% of the memory buffer pool is used on an average during peak periods.
  Tailoring the buffer pool size is based on the following information,
  - ° At the end of the session, the BTNS JOR listing, see Appendix I, gives information on the maximum percentage use of the memory queues segment.
  - ° During the session, the network control command DT QUEUE can be issued to take random percentages.

  Over a period of time, if the "maximum" has not been exceeded, the NDL parameters determining the memory buffer pool can then be reduced for a new network description, and the CNC utility rerun.

- Number of queues sharing the buffer pool:
  The QCPOOL parameter of the GENCOM command enables all or part of the memory buffer pool to be shared by several queues if all or some of the queues are defined with the QCPOOL parameter in their respective QUEUE commands.
  The disadvantage of shared buffer pools is the liability of overloading.
  As a general rule, the type of queue determines the use of the buffer pool, namely,
  - ° Terminal queues are more suitable to share the buffer pool
  - ° Program queues are more suitable for individually reserved pools in order to receive several messages in a stream.

- defining disk queues

In order to use disk queues,
- the disk queue file must be preallocated
- a buffer segment must be generated by CNC to contain
  - ° a bit map to reflect the status of each block of the file, i.e., used or unused
  - ° buffers to read/write HCBs and TCBs from/to the disk file.

The size of the buffer segment is given in the CNC report, see Appendix G.

Calculating the size of the queue file extent depends on

- The block size defined by the QDBLKSZ parameter of the GENCOM command. A block size of 500 to 1000 bytes should satisfy the requirements for most applications. Any size smaller than 500 bytes would increase the overhead for disk I/O transfers.
  The larger the disk capacity, the larger QDBLKSZ can be.

- The maximum number of blocks permitted on the file defined by the algorithm,

  $\Sigma NUMREC \leqslant 32767$, for NUMREC see QUEUE command

  If the total number of blocks exceeds the maximum, i.e., 32767, then an error message will be flagged at network generation

  ERROR CN0025 SEVERITY 4 QUEUES FILE TOO LARGE

  For details, see "NDL Error Messages" in Appendix C.

- The characteristics of the disk medium, namely,
  - ° The number of blocks per track
  - ° The number of tracks per cylinder

Disk Drive Identifier	Number of Blocks per Track	Tracks/ Cylinder	Capacity Megabytes
MSU0310	$\dfrac{(7294 - \text{QDBLKSZ})}{[101 + ((2137 \times \text{QDBLKSZ}) \div 2048)]}$	20	29
MSU0350	$\dfrac{(13165)}{(135 + \text{QDBLKSZ})}$	19	70
MSU0400			100
MSU0402			100
MSU0452			200

- preallocating disk files

  In order to optimize performance and reduce response times, contention in accessing files should be avoided. Wherever possible, files used during the communications session should be allocated on different disks.

  - The files used most frequently in a communications system are,
    - ° System backing store, SYS.BKST1, for swapping code and data segments of the system and application programs during execution. The file is accessed by the VMM (virtual memory management) component of GCOS. A "bottleneck" will occur in accessing the file if memory is overloaded by too many jobs concurrently executing.
    - ° The queue file which is accessed thru MAM by
      - BTNS and,
      - MAM (MCS COBOL) application programs.
    - ° User files which are accessed only by MAM application programs.

  - Contention in accessing files arises in the following instances,
    - ° If user files and the queue file are allocated on the same disk, then a great number of I/O operations has to be performed by the system in processing messages, thereby slowing down the thruput. Contention arises between
      - BTNS in trying to queue and dequeue messages
      - The MAM application program in trying to access both types of files.
    - ° If the queue file is allocated on the system disk, and if the load imposed on the system is such as to increase the rate of segment swapping, then contention is caused among
      - VMM swapping and loading code and data segments
      - BTNS in trying to queue and dequeue messages
      - The MAM application program in trying to access the queue file.

  - In order to optimize file allocation concerning disk I/Os and missing segments, the user has the statistics provided by the BTNS JOR listing and the MAM application program JOR. For details of the BTNS JOR, see Appendix I.

SECTION VI

QUEUE MAINTENANCE

QMAINT is a system utility program available to the user to perform maintenance on memory and disk queues defined for the network generated by a CNC session.

The utility performs the following functions,

- Prints messages from a queue
- Purges messages from a queue
- Sends messages to a queue
- Displays the status of a queue

QMAINT is described in terms of:

- Input/Output Data
- Command Language
- Executing QMAINT

INPUT/OUTPUT DATA

The input data to QMAINT is a set of commands specified by the user, which describes the actions to be performed; the output data from QMAINT is in the form of print-outs.

- input to QMAINT : The commands to QMAINT 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 to be used repeatedly, e.g., to display or purge the contents of all queues at the end of a telecommunications session, then they should be stored as a member of a library.

- output from QMAINT: Print-out reports can be of the following,

  - Messages in the JOR :
    JOR messages are printed in case of fatal user or system errors that cause QMAINT to halt abnormally. A detailed description of these messages is given in Appendix C.

  - SYSOUT Reports :
    The SYSOUT report lists the QMAINT commands which have been executed, and after each command additional information specific to it is included.
    The contents and format of the QMAINT commands are dealt with in "Command Language" later in the section.
    A summary of errors detected during the execution of QMAINT commands is printed out at the end of the SYSOUT report.

COMMAND LANGUAGE

QMAINT command language is described in terms of,

- Language convention
- Command Repertoire

Language Convention

- Commands and their parameters are written in capitals.
- The following symbols must not be employed in user-supplied values,

, comma	/ slash	(open parenthesis
∇ blank	= equals	) close parenthesis
; semi-colon	* asterisk	" quotation mark

- Language reserved names must not be employed in user-supplied values.
- A command can span several cards, the end of the command being terminated by a semicolon.
- Individual parameters of a command can be separated by commas, blanks or commas and blanks.
- Blank cards and "comment" cards are not processed.
- Default values are underlined; if no default is specified, a value must be supplied.
- The sequence of commands is not significant.

Command Repertoire

There are 6 commands, namely,

- COMM - defines a "comment"
- PRINT - prints the contents of a queue
- PURGE - purges messages in a queue
- QSTATUS - lists status and generation parameters of queues
- SEND - sends user-defined messages to a queue
- STATUS - resumes or suspends processing of QMAINT commands in case of error

The commands are listed with the following information,

- Definition
- Format of Command
- Parameter Descriptions
- Command Report
- Examples (where applicable)

COMM

Definition

The COMM command defines a comment and may appear anywhere in the sequence of commands.

Format of Command

```
COMM "string";
```

Parameter Descriptions

"string"

- a character string between quotation marks that must be opened and closed on the same card.

  A maximum of 72 characters can be specified for each COMM command. If a comment is greater than 72 characters, then the excess number of characters must appear on a following COMM card.

Command Report

Only on command listing.

PRINT

Definition

The PRINT command is used to print out, without any alteration to, the contents of one, several or all of the queues defined within the given network.
The contents of a queue is the set of messages that are completely queued, i.e., not in the transitional state of being sent or received.
The messages are printed out in the order that they would be received from the queue concerned by an application program or BTNS.

Format of Command

```
PRINT { name [,name]...  } ;
      { *                }
```

Parameter Descriptions

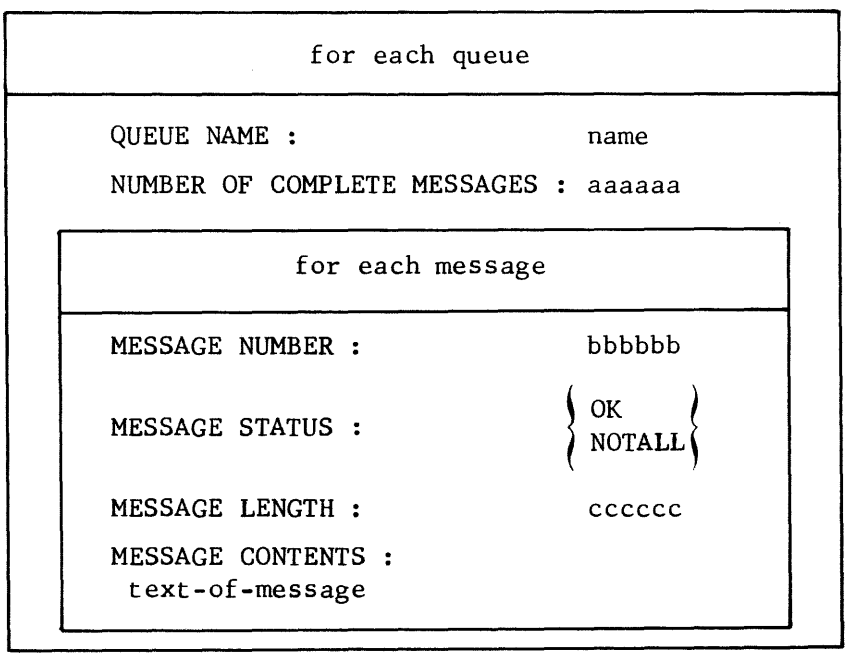
name
- ranges from 1 thru 4 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command, see Section V.

name,name...
- 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 CNC generation.

Command Report

The format of the print-out is :



name
- ranges from 1 thru 4 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command, see Section V.

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 currently present in a queue, irrespective of the status of the messages, i.e., completely queued, or in the transitional state of being sent or received.
The messages are destroyed in the order that they would be received from the queue concerned by an application program or BTNS.

Format of Command

PURGE name, { ALL | NUMBMSG = nnnnn } ;

Parameter Descriptions

name
- ranges from 1 thru 4 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command, see Section V.

ALL
- specifies that all 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 thru 99999.

Command Report

The format of the print-out is :

```
QUEUE NAME :                    name
NUMBER OF DELETED MESSAGES : aaaaaa
```

name
- external name of the queue specified in the PURGE command.

aaaaaa
- number of messages destroyed, see ALL and NUMBMSG above.

Definition

The QSTATUS command lists the current status and the generation parameters of one, several or all of the queues defined within a given network.

Format of Command

```
QSTATUS { name [,name]... }
        { *               } ;
```

Parameter Descriptions

name
- ranges from 1 thru 4 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command, see Section V.

name,name...
- 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 a given network.
  The order in which this information is listed will be the order in which the queues were declared at CNC generation.

Command Report

The format of the print-out is :

for each queue	
QUEUE NAME :	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]

name
- external name of the queue specified in the QSTATUS command.

aaaaaa
- number of messages completely queued in the current state.

bbbbbb
- number of messages partially sent to the queue concerned, i.e., messages not terminated by EMI or EGI.

cccccc
- number of messages partially received from the queue concerned.

dddddd
- number of core or disk blocks allocated to the queue concerned at CNC generation thru the respective parameters of the corresponding QUEUE command,
  - NUMBLK : number of core blocks to be used as the memory queue pool
  - NUMREC : number of blocks to be used as the disk queue file.

  The text "NUMBER OF BLOCKS ALLOCATED TO THIS QUEUE : dddddd" can only appear if the queue was not defined with the QCPOOL option in the corresponding QUEUE command at CNC generation.

eeeeee
- number of used blocks among the "dddddd" blocks reserved, see above. The text "NUMBER OF BLOCKS USED FOR THIS QUEUE : eeeeee" can only appear if the queue was not defined with the QCPOOL option in the corresponding QUEUE command at CNC generation.

ffffff
- total number of core blocks of the core queue pool to be shared by all queues qualified by the QCPOOL option in their respective QUEUE commands. The total number of core blocks is defined by the QCPOOL parameter of the GENCOM command.
  The text "MAXIMUM NUMBER OF BLOCKS IN THE POOL : ffffff" can only appear if the queue was defined with the QCPOOL option in the corresponding QUEUE command at CNC generation.

gggggg
- number of used core blocks from the "ffffff" blocks reserved, see above. The text "NUMBER OF BLOCKS USED FROM THE POOL : gggggg" can only appear if the queue was defined with the QCPOOL option in the corresponding QUEUE command at CNC generation.

BREAK
- applicable only to program queues, see QUEUE command in Section V.

CORE
- if NUMBLK parameter is specified in QUEUE command, see "dddddd" above.

DISK
- if NUMREC parameter is specified in QUEUE command, see "dddddd" above.

RESTART
- applicable only to disk program queues, see QUEUE command in Section V.

TWA
- applicable only to program queues, see QUEUE command in Section V.

Definition

The SEND command sends user-defined messages to queues.
The text of the message to be sent immediately follows the SEND command and can appear on several cards, each card spanning from column 1 thru 80.
This function is used to simulate terminals and for debugging MCS COBOL programs without a network being declared.

Format of Command

SEND name [,ENDMSG="end-of-message-string"] [,LENGTH=nnnn] ;

Parameter Descriptions

name
- ranges from 1 thru 4 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command, see Section V.

ENDMSG
- ranges from 1 thru 5 alphanumeric characters enclosed within 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, then the message text must be terminated by a //EOM card.
  Either ENDMSG or //EOM is mandatory, see Example following.

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 SEND command, a warning message is displayed by QMAINT, see Example following.

Command Report

The format of the print-out is :

```
QUEUE NAME :                    name

MESSAGE STATUS :                { OK     }
                                { NOTALL }

MESSAGE LENGTH :                aaaaaa

MESSAGE CONTENTS :
 text-of-message
```

name
- ranges from 1 thru 4 alphanumeric characters and is the external name of the queue as specified in the corresponding QUEUE command, see Section V.

aaaaaa
- length of the message in characters sent to the queue, see Example following.

text-of-message
- text of the message sent to the queue is edited in a maximum of 110 characters per line.

Examples

In the following examples, data presented on the cards is left-justified, i.e., starting at column 1 on the left.

Example 1

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, i.e.,

ADVISE TERMINAL SHUTDOWN∇ ← 56 spaces → ∇

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

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, i.e.,

ADVISE TERMINAL SHUTDOWN

Using this method of specifying the LENGTH parameter, the message can be "tailored" exactly and no space is therefore wasted in the queue.

Definition

The STATUS command continues or suspends the processing of QMAINT commands when an error has previously occurred.

Format of Command

```
STATUS { EVEN  }
       { ONLY  } ;
       { RESET }
```

Parameter Descriptions

ONLY
- following commands are executed only if no error has occurred.

EVEN
- only the following command is executed when an error has previously 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, i.e., the queue must not be allocated to any application program in IN, IO or OUT mode.
- Each terminal queue referenced by a QMAINT command must be available, i.e., the queue must not be currently allocated to any application program
  - either explicitly thru a $QASSIGN statement
  - or implicitly thru the terminal connection to the application program.

Run-time JCL

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;
```

Syntax for QMAINT run-time JCL :

- The STEP statement specifies the system load-module H_QMAINT is located in the system load-module library SYS.HLMLIB.
- The ASSIGN statement specifies the system-reserved internal-file-name H_CR as the file containing the QMAINT commands, either on cards as an input enclosure or as a source library member.

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;
COMM "CONTINUE EVEN IF WRONG RESULT FROM PURGE";
STATUS EVEN;
PURGE Q2,NUMBMSG=5;
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;
```

The job defined by QDISP executes on the queues specified by the current network. The maintenance actions to be performed are described by the QMAINT commands in the input enclosure.

SECTION VII

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, a communications environment must first be created. The H_CNC utility, see Section V, is used to generate the network. The network is successfully created by the absence of error conditions during generation.
BTNS (basic terminal network support), an MCS COBOL step or a VCAM subsystem (virtual communications access method) can then be started.
The two VCAM subsystems are,

- IOF (interactive operation facility)
- TDS (transaction driven system).

Whenever backing store is destroyed, the H_CNC utility must be executed again

- the H_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 option).

Backing store is destroyed

- either thru 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 :

- H_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.
- A VCAM subsystem cannot be run if another version of the same VCAM subsystem is currently executing.
- An MCS COBOL step with a $QASSIGN statement specifying a queue currently allocated to another MCS COBOL step cannot be run.

Failing to comply with any of the above constraints will lead to a step abort.

LEVELS OF SIMULTANEITY FOR COMMUNICATIONS

Within the limits for operability as previously defined, communications components can be started and terminated independently for as long as the maximum system multiprogramming level is not exceeded.

The following occurrences illustrate the levels of simultaneity for a communications session,

1. A data collection MCS COBOL 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 COBOL 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 launched.
   - The job requests connection to BTNS and is enqueued.
   - TDS, however, remains 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 COBOL 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 network is initialized.
   - The TDS job request is now accepted , see 3.
   - 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 (BTNS is a service job)

8. DATCOLL has completed emptying the disk queues and terminates.

   Number of Simultaneities : 3

9. The TDS batch entry, see 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 input queues and terminates.

    Number of Simultaneities : 0

12. BTNS is retained in the system to fill disk queues.
    This operation would be continued in a following session by launching an application DATCOLL, see 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 at terminal level, appropriate dispatching priorities for communications components must be chosen. Batch jobs are given low priorities as they are not subject to real-time contraints.

The user specifies a job class which determines the scheduling priority, the dispatching priority and the associated level of multiprogramming for the job. For a description of the use of job class, see System Management Guide.

The following example illustrates the effect of job class on priorities.

Priorities for MCS COBOL & VCAM		
variable	MCS COBOL step	VCAM subsystem (if TDS)
job class	H	J
scheduling priority	5	5
dispatching priority	1	1
multi-programming	1	1

Explanations for example showing priorities for MCS COBOL and VCAM :

- H_CNC and H_QMAINT utilities are not included as they are not subject to real-time constraints and can therefore be executed as normal batch jobs of class P, say.
- The dispatching priority for the MCS COBOL step and the VCAM subsystem, e.g., TDS, are the same. In order to avoid conflict when both components are available in the system, the application programmer should exercise care. Indiscriminate program queue scanning for data by the application program can degrade system performance. This could be as a result of the programmer specifying a RECEIVE statement with NO DATA clause or an ACCEPT statement instead of building a suitable queue structure and using one of the scanning techniques provided.
- The level of multiprogramming for the MCS COBOL step and the VCAM subsystem is the default value and can be modified by an OCL command, see System Operation : Operator Guide.

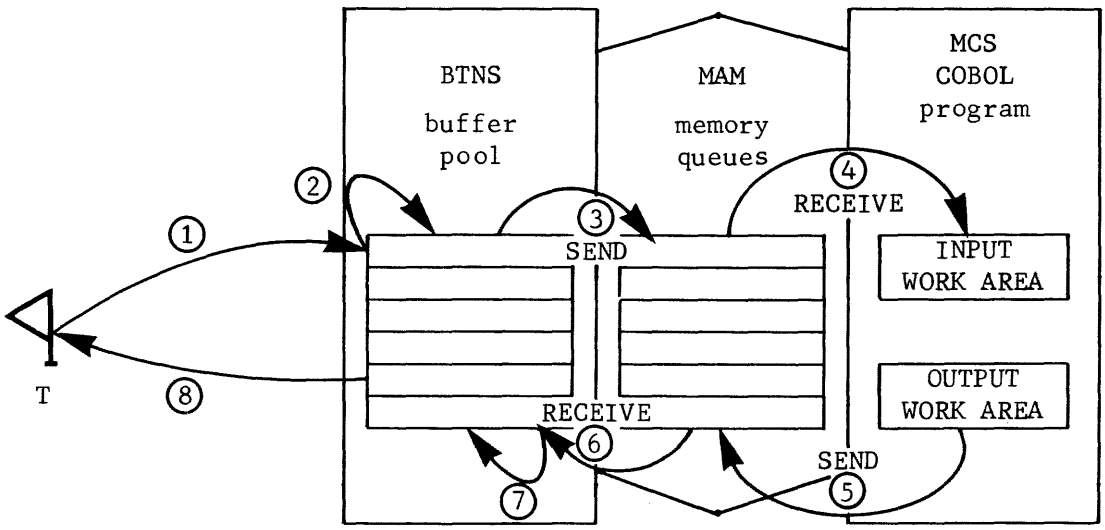
DATA FLOW DURING MESSAGE EXCHANGE

Message exchange follows a prescribed path and involves the following types,

- Exchange between an MCS COBOL program and a terminal using a memory queue
- Exchange between an MCS COBOL program and a terminal using a disk queue
- Exchange between a VCAM subsystem and a terminal.

Exchange between an MCS COBOL Program and a Terminal using a Memory Queue

Exchange between
MCS COBOL Program and a Terminal using a Memory Queue

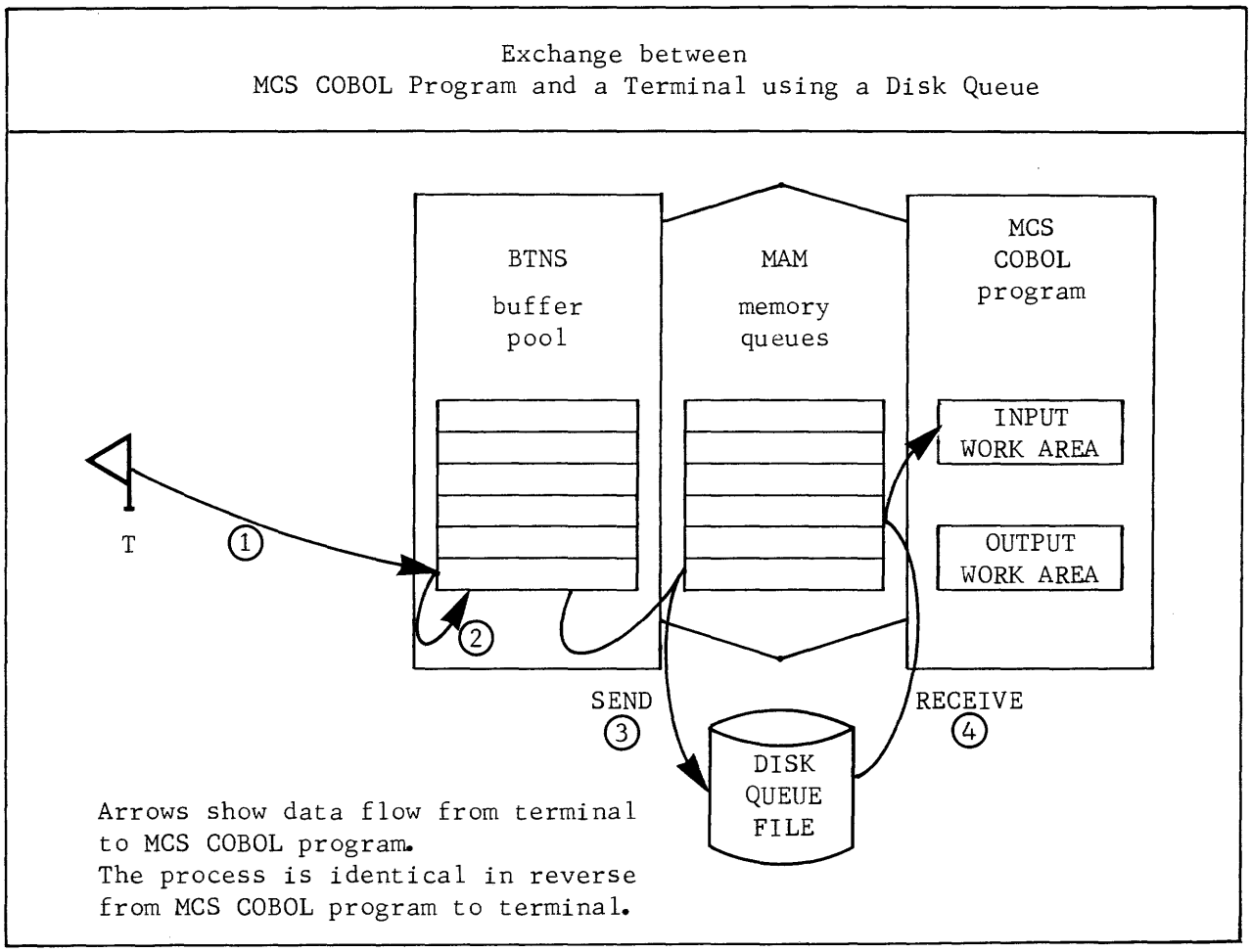


Explanations for example of exchange between MCS COBOL program and a terminal using a memory queue,

1. When the line is polled, terminal T sends a message which is stored in the BTNS buffer pool.
   Buffer units are dynamically allocated by BTNS as required by message length.
2. Control characters and marks are translated according to terminal type and data format for input.
3. BTNS stores the message in the memory input queue to which the terminal is connected by issuing a call to the MAM "send" procedure.
   A "send" procedure is performed for each buffer unit to be transferred.
4. When the program issues a RECEIVE statement to the symbolic input queue, the message or a part of it, is moved into the program's input work area.
   As many RECEIVE statements as necessary are issued to collect the complete message.
5. When the message is to be sent to the terminal, the program issues a SEND statement to the symbolic output queue, and the message is transferred from the program's output work area to the memory output queue associated with the terminal.

6. When the output request is located, BTNS issues a call to the MAM "receive" procedure to store the message in the buffer pool.
   A "receive" procedure specifying the length and address is performed for each buffer unit needed to contain the message.
7. Control characters and marks are translated according to terminal type and data format for output.
   The translation can be either "normal" or "unedited".
8. The message is then transferred from the buffer pool to the destination terminal T.

Exchange between an MCS COBOL Program and a Terminal using a Disk Queue



Explanations for example of exchange between MCS COBOL program and a terminal using a disk queue,

1. When the line is polled, terminal T sends a message which is stored in the BTNS buffer pool.
   Buffer units are dynamically allocated by BTNS as required by message length.
2. Control characters and marks are translated according to terminal type and data format for input.

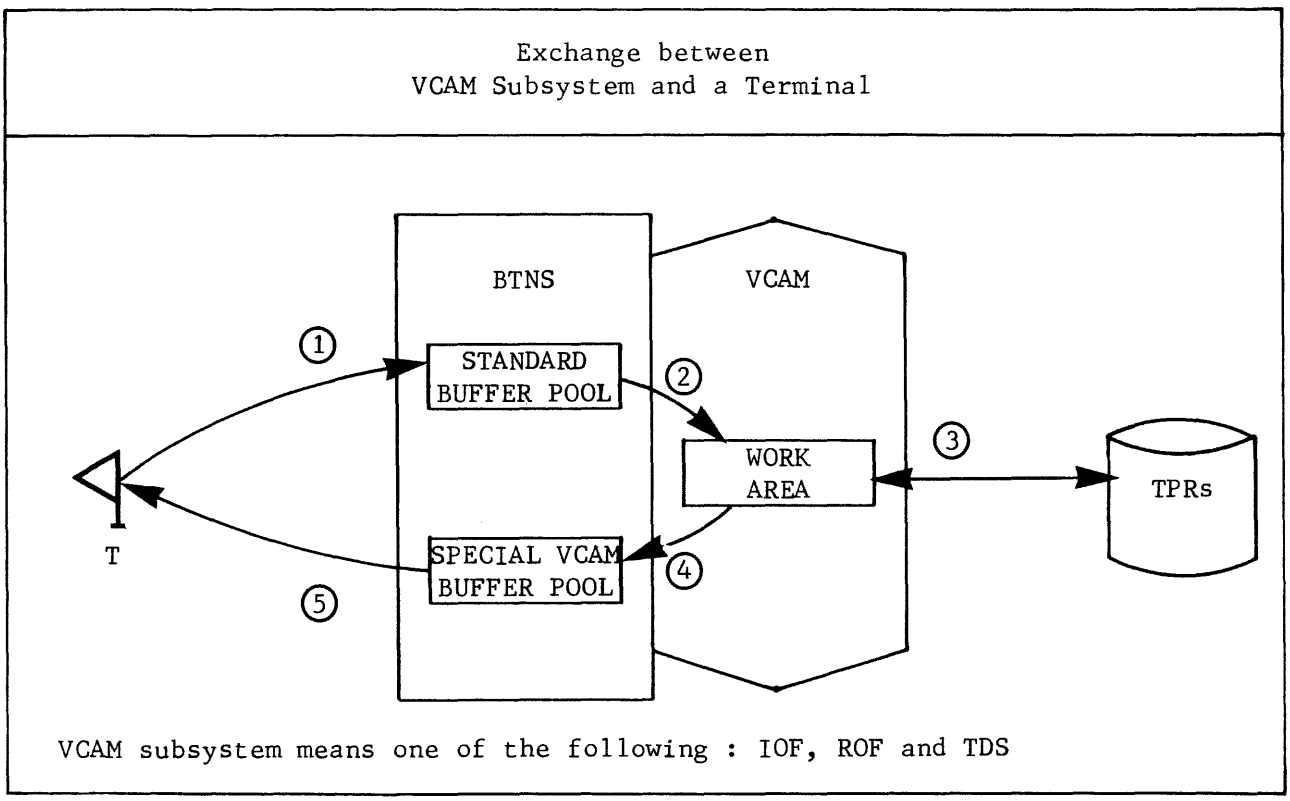
3. When BTNS issues a call to the MAM "send" procedure to store the message in the disk input queue to which the terminal is connected, the following events take place,
   - The message is first moved to the disk I/O buffer pool
   - The message is then written to the disk queue file.

   Disk access for BTNS is asynchronous which means that BTNS does not have to wait for the completion of an I/O operation.

4. When the program issues a RECEIVE statement 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 program's input work area.

   Disk access for the program is synchronous which means that the program must wait for the completion of an I/O operation.

Exchange between a VCAM Subsystem and a Terminal



Explanation for example of exchange between a VCAM subsystem and a terminal,

1. A transaction is initiated at the terminal T.
   The message is stored in the BTNS standard buffer pool, occupying as many buffer units as required.
2. The message is then moved to the VCAM work area specified at the generation of the specific version of VCAM.
3. The message activates or is processed by 1 or several transaction processing routines (TPR).

4. A reply to the terminal is moved from the VCAM work area to a special VCAM buffer pool also specified at the generation of the specific version of VCAM.

5. The message is then delivered from the buffer pool to the destination terminal.

ALLOCATING MEMORY RESOURCES

Multiprogramming in a virtual memory environment may lead to an overload situation where several programs compete for memory resources.
The effects on the communications session are,

- segments of its components not currently executing may be swapped with batch programs
- these segments, when needed to process a message on arrival, must then be reloaded, causing
  - a tremendous increase in system overhead
  - a considerable increase in response time
  - a degradation of overall thruput.

The problems to be solved are,

- avoiding memory overload of the system

  The sum of the "working sets" of programs executing concurrently must not exceed the physical memory size available.

- guaranteeing 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 (JCL) statement

  The $SIZE statement declares the "working set" of the program for the purpose of controlled scheduling to avoid memory overload.

- using the MAXMEM and MINMEM parameters of the $STEP (JCL) statement

  - MAXMEM : Used for tuning the "working set", and should be discontinued when the optimal size has been determined.
    The facility ensures that
    - the amount of memory allocated will never be less than the DWS (declared working set) 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, i.e., the step does not 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.
    The 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 (OCL) 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.

MCS COBOL (MAM) Application Programs

- determining the DWS parameter of $SIZE

```
run-time JCL for estimating DWS

      STEP ...   MAXMEM;
      SIZE dws-value ...;
```

The first time that the step is executed, the dws-value specified for $SIZE is calculated from the linkage listing of the program, see "Linking MCS COBOL Programs" in Section III.
The JOR listing at the end of the job step will indicate the number of missing segments, if any.
The general rule in tailoring 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 that an increase in missing segments occurs
- If many missing segments are indicated, then a proportionately higher dws-value should be specified.

Successive executions of the job step will ultimately give an optimum dws-value.

- guaranteeing memory

```
run-time JCL for normal execution

      STEP ...   MINMEM;
      SIZE dws-value ...;
```

The dws-value specified is the optimum value estimated over successive job runs.

MAM and VCAM

Both MAM and VCAM are system functions which can be "locked" in memory by the PMM command, the format and function being as follows,

- PMM VCAM : VCAM segments are "locked" when running IOF, ROF or TDS.
- PMM MAMM : MAM segments managing memory queues are "locked".
- PMM MAMD : MAM segments managing disk queues are "locked".

BTNS

BTNS runs automatically under MINMEM when initialized by the ST network control command. It communicates to the system the memory size needed according to the configuration present.

APPENDIX A

TERMINAL CONFIGURABILITY

The following tables provide the user with information concerning the configurability of the terminal types supported by GCOS64 in terms of,

- access methods and subsystems supporting the terminal
- optional hardware features (auxiliary devices or terminal subtypes) supported and the restrictions, where applicable
- types of links supported, namely
  - point-to-point
  - multipoint
- NDL name of the terminal declared for CNC generation and its corresponding marketing identifier
  (several compatible terminals known by different identifiers can be declared under the same NDL name in the TYPE parameter of the TERMNL command)
- software and firmware releases, where applicable, supporting standard terminals connected to the L64 host computer.

The terminals are arranged in the order of their line procedures, see "Types of Standard Terminals" on page 5-24.

For supplementary information on the auxiliary devices supported with the standard terminals, see "Allowed TYPE/SUBTYPE Combinations" on pages 5-25 and 26.

Terminal Types Applicable to MAM and VCAM Subsystems					
Type	Model	MAM	VCAM Subsystems		
			TDS	IOF	
TN300/1200	TermiNet 300/1200	Y	Y	Y	
TWU/PRU1001	TWU/PRU1001	Y	Y	Y	
TWU/PRU1003	TWU/PRU1003	Y	Y	Y	
TWU/PRU1005	TWU/PRU1005	Y	Y	Y	
TTY33/35/37/38	Teletype 33/35/37/38	Y	Y	Y	
VIP7100/7200	VIP7100/7200	Y	Y	Y	
VIP765/775/785	VIP765/775/785	Y	Y	Y	
TWU/PRU1901	TWU/PRU1901	Y	Y	Y	
VIP7700	VIP7700/7700R/7705	Y	Y	Y *	
VIP7760	VIP7760	Y	Y	Y *	
HL62	62 (any model)	Y	-	-	
HL64	64 (any model)	Y	-	-	
HL66	66	Y	-	-	

Note: 1. Where no entry is made for the access method, the terminal type is not supported.

2. * denotes that IOF does not support auxiliary devices (subtypes) as printer (hard copy), cassette, disk or diskette.

Terminal Types and Link Usage						
Type	Point-to-Point			Multipoint		
	local (W8 cable)	leased (modems)	switched (1)	through modems	through MIU (2)	switched
TN300/1200	Y	Y	Y	-	-	-
TWU/PRU1001	Y	Y	Y	-	-	-
TWU/PRU1003	Y	Y	Y	-	-	-
TWU/PRU1005	Y	Y	Y	-	-	-
TTY33/35/37/38	Y	Y	Y	-	-	-
VIP7100/7200	Y	Y	Y	-	-	-
VIP765/775/785	Y(3)	Y	Y	Y	Y	-
TWU/PRU1901	Y(3)	Y	Y	Y	Y	-
VIP7700	Y(3)	Y	Y	Y	Y	-
VIP7760	Y(3)	Y	Y	Y	Y	-
HL62	-	Y	Y	-	-	-
HL64	-	Y	Y	-	-	-
HL66	-	Y	Y	-	-	-

Note: 1. Switched lines can be with or without automatic call detection.

2. Multiple Interface Unit, see "Types applicable to MAM and VCAM".

3. Require "direct timing source" option for the terminal.

Note: No entry denotes that the link is not supported.

Terminal Types and Software/Firmware Support		
Type	Model	Software/Firmware Support
TN300/1200	TermiNet 300/1200	-
TWU/PRU1001	TWU/PRU1001	-
TWU/PRU1003	TWU/PRU1003	-
TWU/PRU1005	TWU/PRU1005	-
TTY33/35/37/38	Teletype 33/35/37/38	-
VIP7100/7200	VIP7100/7200	-
VIP765/775/785	VIP765/775/785	-
TWU/PRU1901	TWU/PRU1901	-
VIP7700	VIP7700/7700R/7705	-
VIP7760	VIP7760	VIP7760 Firmware - Revision 6 (59733684 - 006)
HL62	62 (any model)	GCOS62 - Release 4 BSC2780 Package
HL64	64 (any model)	GCOS64 - Release 0400 BSC2780 Package
HL66	66	GRTS 3 Release 2H GCOS 3 Release 2H

Note: Where no entry is made in "Software/Firmware Support", the information is not applicable.

APPENDIX B

PROGRAMMING FOR TERMINALS

The information concerns the programming interface for terminals under either MCS or TDS.

> In the text for each line procedure, except the BSC2780 line procedure, non-graphic codes are described for ease of interpretation, in mark representation >< , see Section IV.
>
> The COBOL programmer must interpret this representation to match one of the following two requirements,
>
> - if for TDS, then the TDS code representation conventions
> - if for MCS, then the mode of transmission selected for programming.
>
> In the case of the BSC2780 line procedure, certain control characters reserved for the procedure must not be used by the application program. These control characters are given in the tables "Specific Transparent Control Sequences" and "BSC Controls and EBCDIC Representation" in the BSC2780 line procedure.

Terminals are classified into 3 groups, each group comprising a number of compatible terminals sharing a common line procedure, namely,

- TTY
- VIP
- BSC

For a detailed list of terminals supported for each line procedure, see page 5-24 and Appendix A.

Apart from using the same line procedure, terminals are compatible when they function as follows,

- from the point of view of hardware/firmware,
  - they share the same TCT (translation control table)
  - they are connected over the same multipoint line when using a "poll/select" facility
- from the point of view of the application program,
  - they share the same controls for formatting the character image.

Information is treated as in the following order,

- general information is given for each line procedure, which for ease of reference are listed in alphabetical order, and not according to their groups, namely,
  - BSC2780 line procedure
  - TTY line procedure
  - VIP line procedure
- detailed information is given for each terminal, where available, within the group, in terms of their control codes for
  - terminating messages
  - erasing functions
  - inserting VIP headers
  - activating auxiliary devices (terminal subtypes)
- the terminals are headed by the name of their line procedure followed underneath by its NDL name declared at CNC generation; within the line procedure, terminals are listed in alphabetical order.

This appendix is to be treated as a continuation of Section IV, giving in detail information specific to terminal programming.

BSC2780 LINE PROCEDURE

GCOS64 communications software provides the user with the facility for data exchange under the BSC2780 line procedure over

- private or leased point-to-point lines (BSC1)
- switched point-to-point lines (BSC2).

Communications is established between a Level 64 host computer and any of the following CPUs,

- Level 62 (L62)
- Level 64 (L64)
- Level 66 (L66).

The link provides a program-to-program communications facility.
On the L64 host computer side, the programs supported can be

- a user-provided MCS COBOL application.

As the facility only concerns program-to-program communication, the BSC2780 line protocol does not include automatic processing of end-to-end control characters defined for the BSC2780 IBM-type terminal, which are,

- ESC : escape, output device selection
- HT : horizontal tabulation, positioning
- EM : end-of-medium, variable data length delimiter.

Since these control characters appear within the user text, they may be handled directly by the MCS COBOL application program.

Documents for reference are:

For Level 62 communications,
AW39 : Series 60 (Level 62) BSC1/2 Communications

For Level 66 communications,
DD40 : Level 66/6000 Remote Terminal Supervisor (GRTS)
DD48 : Level 66/6000 GCOS Network Processing Supervisor (NPS)

BSC2780 LINE PROCEDURE

LINK STATES

Except where explicitly defined, the state of the link is completely controlled by the system thru 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 dialing 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 into 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 in progress.

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, namely,

- the link may be idle, i.e., absence of transmission
  - If the link is supported by a permanent point-to-point type connection, the idle condition persists until the transmission phase is initialized. When the phase is started, the L64 host computer begins to monitor the line for as long as the line stays open.
  - If the link is operated over a switched network, the idle condition is entered from the disconnected state on successful completion of the dialing procedure.
    The L64 host computer then monitors the line until a disconnect control sequence is issued for inactivity time-out.
- information exchange may take place between two stations

  Information exchange involves the transmission of control blocks and reply blocks. Initialization and termination phases of the transmission are executed in the control state.

Contention between two point-to-point stations arises when both bid for the line at the same time in order to transmit.

The station that wishes to transmit bids for the line by sending the ENQ control character 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 three times.

BTNS 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,
- if simultaneous line bids occur, the "primary" station transmits first
- the "primary" station persists bidding to a maximum of three times until it receives an appropriate reply; any ENQ control received by the "primary" station once it has taken action to request the line is ignored
- the "secondary" station must respond to any valid ENQ control that it may receive
- 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 L64 CPU may be defined either as a "primary" or "secondary" station.

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 thruout the entire transmission until the last message ended with ETX has been successfully transferred.
An end-of-transmission control signal EOT is then issued to reset the link to its control state.
The return to the control state can be achieved by the MCS COBOL application program thru either KCO or BCO.

In the message transfer state, the station functions in one of two ways, namely,

- master station

  The master station transmits control characters and block check characters containing redundant information for checking purposes.

- slave station

  The slave station 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 arises whereby transmission can no longer proceed, either station may reset the link to discontinue dialog.

LOG-ON PROCEDURE

An easy way to establish the logical connection between a remote BSC station and a program queue is to dedicate it to the queue, i.e., the corresponding TERMNL command must be declared with the ASSIGN and AUTO parameters, see Section V.

In this case, the connection is established as soon as the line is in the control state and remains until the line returns to the disconnected state. For a L64-L64 connection, the link-up may be established on both sides.

If the user wants the remote station to be connected to different application programs, i.e., different program queues, the corresponding TERMNL command must only be declared with the AUTO parameter.
In this case, the remote application program handles the log-on procedure as if it were the terminal operator, namely,

- sending a $*$BRK signal
- answering, where applicable, subsequent questions posed by the site controller if the CONTROL parameter had been specified in the TERMNL command, see Terminal Operations GCOS Manual.

ENCODING DATA

Functionally the BSC2780 line protocol allows two categories of user-provided data to be exchanged over a line, namely,

- text
- heading

Text

Altho ASCII encoding is a function of the BSC2780 line procedure provided thru the TCTNM parameter of the appropriate LINE command, exchanges between Series 60 CPUs are performed in EBCDIC code (internal code, which is the default option).

For more efficient recovery purposes, text may be split into subblocks separated by sequences of control characters irrespective of the mode of transmission.

BSC2780 LINE PROCEDURE

Text may be transmitted in either of the following two modes, depending on the requirements of the application program,

- normal mode :
  the text must not contain any control characters or sequence of control characters significant to the BSC2780 line procedure.
- transparent mode :
  the text may contain unrestricted coding of data since all control characters including the "escape" character DLE must be preceded by DLE to be recognized as a control function.

  The transparent mode is used prinicipally to transmit
  - ° binary data
  - ° floating point numbers
  - ° packed decimal data
  - ° specialized or foreign codes
  - ° machine language computer programs

Specific Transparent Control Sequences	
Control Sequence	Meaning
DLE DLE	transmits the control character 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

BSC2780 LINE PROCEDURE

BSC Controls and EBCDIC Representation			
Control Character	Hex. Code	Meaning in Control State	Meaning in Message Transfer State
ACK0 *	1070	can receive	even block received OK
ACK1 *	1061	can receive	odd block received OK
DLE	10		start of transparent mode
ENQ *	2D	can you accept transmission?	between blocks: repeat last response end block: ignore block, respond with NAK
EOT	37	drop synchronization and return to control state	drop synchronization and return to control state: text invalid
ETB *	26		end-of-block
ETX *	03		end-of-text (transmission)
IRS	1C		end-of-intermediate record
ITB	1F		end-of-intermediate record
NAK *	3D	cannot receive	block does not check: retransmit
PAD	FF	time-fill	time-fill
RVI	107C	request to stop transmission and accept messages	message received OK: I would like to transmit
SOH	01		start-of-block header
STX	02		start-of-block
SYN	32	establish character synchronization: discard character	establish character synchronization: discard character
TTD	022D	transmission begins later: respond with NAK or WACK	transmission continues later respond with NAK or WACK
WACK *	106B	request later and wait until acknowledged	current block received OK: request later and wait until acknowledged

* Line Turn-around

BSC2780 LINE PROCEDURE

Heading

Heading characters which are separated from the text in the message can be only transmitted in normal mode in one of the following ways,

- within messages containing any type of text (normal or transparent)
- alone in a message.

Heading is regarded as control information used by applications for end-to-end control related to text data block(s) following.

GENERAL FORMAT OF DATA MESSAGES

Text
SYN SYN SYN SYN STX text ETB BCC PAD SYN SYN SYN SYN STX text ETX BCC PAD SYN SYN SYN SYN STX text ITB BCC SYN SYN text ITB BCC SYN SYN text ETB BCC PAD SYN SYN SYN SYN STX text ITB BCC SYN SYN text ITB BCC SYN SYN text ETX BCC PAD
Transparent Text
SYN SYN SYN SYN DLE STX transparent text DLE ETB BCC PAD SYN SYN SYN SYN DLE STX transparent text DLE ITB BCC SYN SYN DLE STX transparent text DLE ETB BCC PAD SYN SYN SYN SYN DLE STX transparent text DLE ETX BCC PAD SYN SYN SYN SYN DLE STX transparent text DLE ITB BCC SYN SYN DLE STX transparent text DLE ETX BCC PAD
Heading
SYN SYN SYN SYN SOH heading ETB BCC PAD SYN SYN SYN SYN SOH heading STX text ETX BCC PAD SYN SYN SYN SYN SOH heading DLE STX transparent text DLE ETX BCC PAD

Explanation of General Message Format

SYN SYN SYN SYN :
to establish and maintain synchronization.
The number of SYN control characters to be inserted by the URP is defined thru the ININS parameter of the LINE command at CNC generation.
The default value of the number of SYN control characters is 4.

STX :
start-of-text

text or transparent text :
normal or transparent text respectively

ETB :
end-of-transmission block.
Direction of transmission is retained and another block terminated by either ETB or ETX must follow the current block, if the current block was correctly received in the first place.

BCC :
block-check-control character.
The accumulated parity checking control character is checked or generated by URP firmware.

PAD :
Trailing padding control character which has the EBCDIC value of FF.
The number of PAD characters is defined thru the BADNB parameter of the LINE command at CNC generation.

ETX :
end-of-text; the current transmission is terminated and the direction of transmission may now be reversed if the control message EOT is sent following the last message correctly received.

ITB :
end-of-intermediate transmission block.
See "Subblocks" later in the specification.

DLE :
data-link-escape used to provide,
- supplementary line control characters by combination with another character, Example : DLE @ is interpreted as RVI (reverse interrupt)
- transparent mode control characters, see "Specific Transparent Control Sequences" earlier on in the specification.

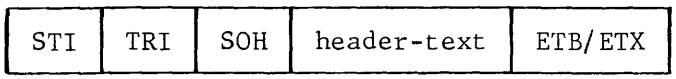
SOH :
start-of-heading defining the start of the control information between,
- SOH and STX or ETB in normal mode
- SOH and DLE STX in transparent mode.

BSC2780 LINE PROCEDURE

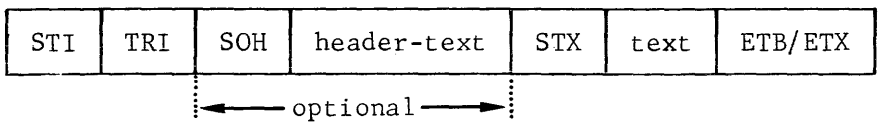
Message Structure seen by BTNS

A text message exchanged between the BTNS message processing routines and the URP firmware can take one of three formats, namely,

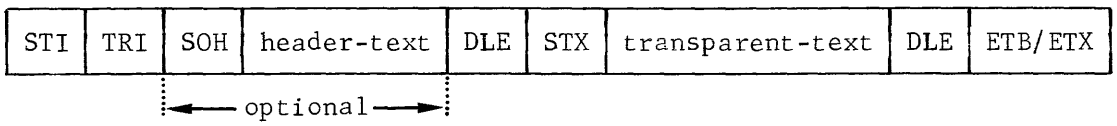
1. User-defined header message



2. Message partitioned into blocks



3. Transparent text message (non-standard codes accepted)



EXPLANATION OF TERMS

STI :
station index, set to 1 and defined by the STATN command at CNC generation, see Section V.

TRI :
terminal index, provision for multipoint handling

SOH :
start-of-heading

header-text :
contents are user-defined

STX :
start-of-text, indicating transmission in normal mode

DLE STX :
start-of-text, indicating transmission in transparent mode, see "Specific Transparent Control Sequences" earlier on in the specification.

DLE :
data-link-escape used with either ETB or ETX to denote "end-of-block" or "end-of-text" in transparent mode.

ETB/ETX :
"end-of-block" or "end-of-text" in normal mode if not preceded by DLE, see DLE.

Message Structure seen by the Application Program

The mark symbols used in the formats below are not transmitted over the line.

A message as seen by the MCS COBOL application program has the following general structure,

- emission

  [><U04 header-text ><U05][><U06][><KCO / ><BCO] text

- reception

  [><U04 header-text ><U05] text

EXPLANATION OF TERMS

><U04 :
start of a user-provided heading

><U05 :
end-of-header

><U06 :
the text following the control must be sent in transparent mode

><BCO :
communication is "broken" after sending the next end-of-file block terminated by ETX.

><KCO :
communication is "kept" (retained) after sending the next end-of-file block terminated by ETX.

text :
user-defined message

BSC2780 LINE PROCEDURE

MANAGEMENT OF DATA TRANSFER BY APPLICATION

Once the link has been established and set in the correct state to perform data transmissions, 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 ETB for normal mode or DLE ETB for transparent mode
- until it sends a block terminated by ETX for normal mode or DLE ETX for transparent mode in which case, the link state can be one of either, namely,
  - retained in the transfer state so that a new line bid is not necessary for resuming transmission in the same direction
  - returned to the control state by sending EOT after the last block
- unless RVI (reverse interrupt) has been issued by the receiving station, see "Reverse Interrupt" later on in the specification.

Unlike other line procedures for which BTNS manages the control characters, the user application, in the case of the BSC2780 line procedure, can control those control functions affecting the line usage thru the MCS interface.

Emission

The user application program can control the turn thru EMI and EGI indicators of the SEND verb, as follows,

- any message sent with EMI is dispatched by BTNS over the line with ETB or DLE ETB, thereby retaining the turn
- any message sent with EGI is dispatched by BTNS over the line with ETX or DLE ETX, signifying the end of the current transmission.

In addition, physical block messages sent over the line must take into account the buffering capability of the receiving station, namely,

- 512 characters

The user application program, however, does not have to consider the actual size of the physical block because BTNS automatically splits messages longer than the buffering capability by

- sending intermediate blocks of the message text with ETB or DLE ETB
- sending the last block with ETB or ETX (DLE ETB or DLE ETX) according to whether EMI or EGI has been specified in the application program.

Reception

Enqueueing of blocks depends on the control character terminating it, namely,

- if a block is received with ETB or DLE ETB, it is enqueued by BTNS with EMI
- if a block is received with ETX or DLE ETX, it is enqueued by BTNS with EGI.

As a result, when the application program issues a RECEIVE, the ENDKEY field of the input CD of such a block would contain the following values,

- 2 for EMI
- 3 for EGI.

See "Message Delimiters" in Section III.

Reverse Interrupt (RVI)

RVI is a line procedure control message thru which a receiving station indicates to the sending station that :

- the last block (sent with ETB or DLE ETB) has been correctly received
- transmission must be terminated as soon as possible for the following reasons,
  - 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.

The RVI is handled as follows,

- reception of RVI :

  The actions taken by BTNS when it receives an RVI while currently transmitting are,
  - the next block is transmitted with ETX or DLE ETX whatever the delimiter of the correspoding message is, i.e., EMI or EGI
  - the corresponding terminal queue from which the message was sent is set in the DISABLE OUTPUT state
  - a control message of the format ><CTLRVI is sent to the program queue to which the terminal is connected if the queue was declared with the BREAK option in the appropriate QUEUE command at CNC generation
  - the line is set in the "input" state in order to receive data from the station which requested the turn to transmit.

  In order to detect RVI, the application program needs to check
  - either the DISABLE OUTPUT state of the terminal queue
  - or the contents of the program queue, namely, the ><CTLRVI control message is delivered to the application program as a null-length message with the a STATUS KEY value of "0".

  The detection of such an event may lead the application program to perform either action,
  - purge the terminal queue by sending to it the control message ><CTLPRG with EGI, only if the terminal queue was previously re-enabled, see below
  - restart the interrupted transmission by re-enabling the terminal queue with the ENABLE OUTPUT verb.

- sending RVI :

  If the L64 application program which is receiving messages wants to stop the sending station, it has only to send into the corresponding terminal queue the control message ><CTLRVI.

  After the reception of every block ended with ETB, BTNS checks the corresponding terminal queue to determine if the RVI has been requested by the receiving application program. If so, the RVI request is sent immediately to the sending station.

Retaining the Communication

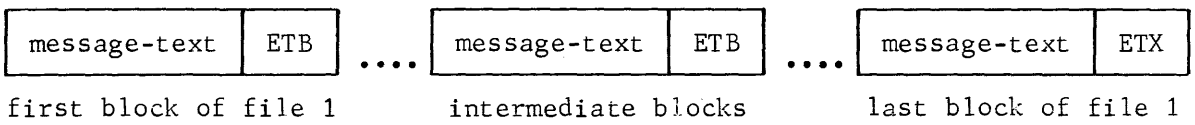
As a default option, BTNS sends an EOT control character after the last block of a transfer specified with either ETX or DLE ETX. This mechanism returns the link to the control state. When in the control state, the line needs to be bidden for in order for a new transfer to take place.

If several transfers are to be done in the same direction, the application program 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 indicating a "break"
- ><BCO is 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.

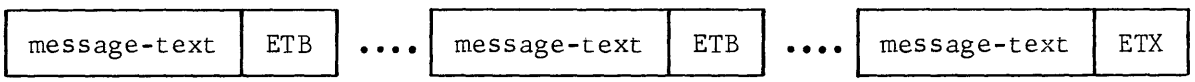
The sequence can be represented diagrammatically as follows,

1. Transfer file 1 (><KCO specified in the first block)



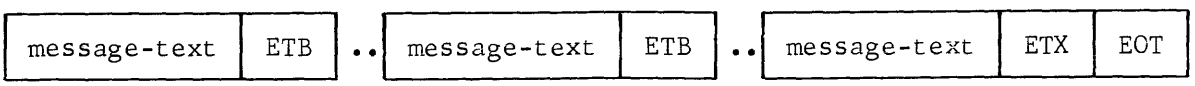
first block of file 1 intermediate blocks last block of file 1

2. Transfer file 2



first block of file 2 last block of file 2

n. Transfer file n (><BCO specified in the first block)



first block of file n last block ending transfer

CONTENTS OF USER MESSAGES

The MCS COBOL user application may send and/or receive heading data alone or with 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-data	><U05	optional message-text

Text

Text contents depends on the mode of transmission, namely,

- normal mode :

  There is no special format nor processing for the text except the processing for standard marks according to the IM or OM options specified in appropriate linked terminal, see TERMNL command in Section V and the MTE operator command in Appendix F.

- transparent mode :

  The application program receives transparent data as normal data except that no processing of standard marks is done whatever the IM option specified for the linked terminal.

  To send in transparent mode, the user must ensure that each message is preceded by the standard mark ><U06 in which case, the following actions are taken,
  - MCS does not process standard marks
  - BTNS provides the appropriate control character sequences, e.g., DLE xxx, where xxx is the control character specified in the text.

  ><U06 must be provided with any message to be sent in transparent mode because it could be useful, in some cases, to send files with mixed normal and transparent data, which is allowed by the BSC2780 line procedure.

Subblocks

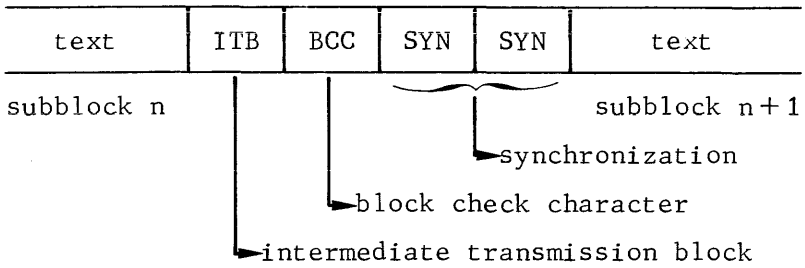
Messages are split into subblocks in order to provide a more efficient means of error checking and retransmission of the subblock in error before the reception of the entire message text.

This capability for splitting messages into subblocks is at line protocol level and as such, is not visible to the application program.
Instead, BTNS and URP firmware ensure that the intermediate control characters separating subblocks are handled as follows, for error detection and recovery,
- checked for retries
- inserted on emission
- deleted on reception.

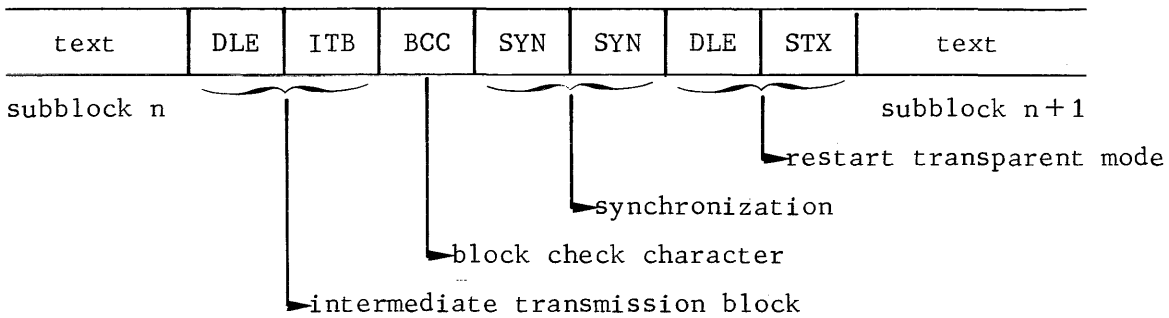
BSC2780 LINE PROCEDURE

The format of the subblock separators depends on the mode of transmission,

- normal mode



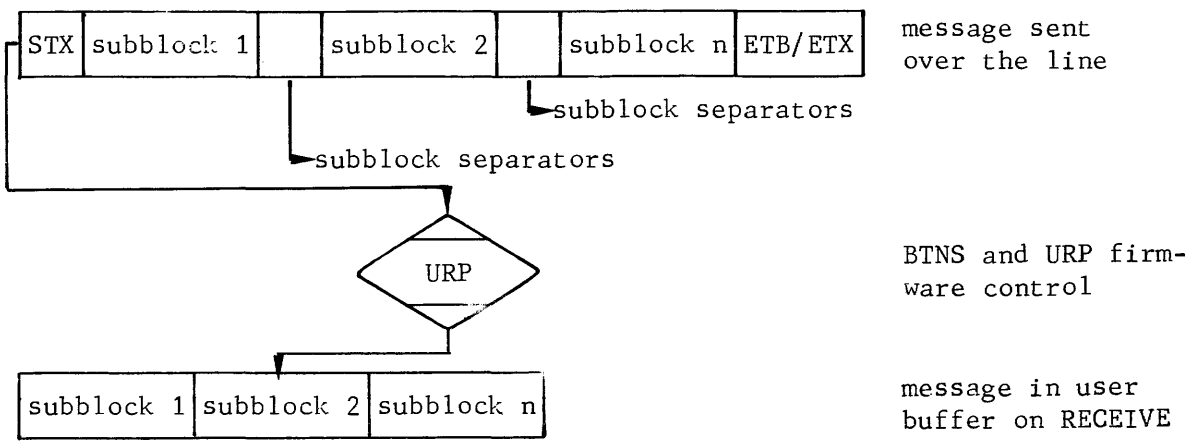
- transparent mode



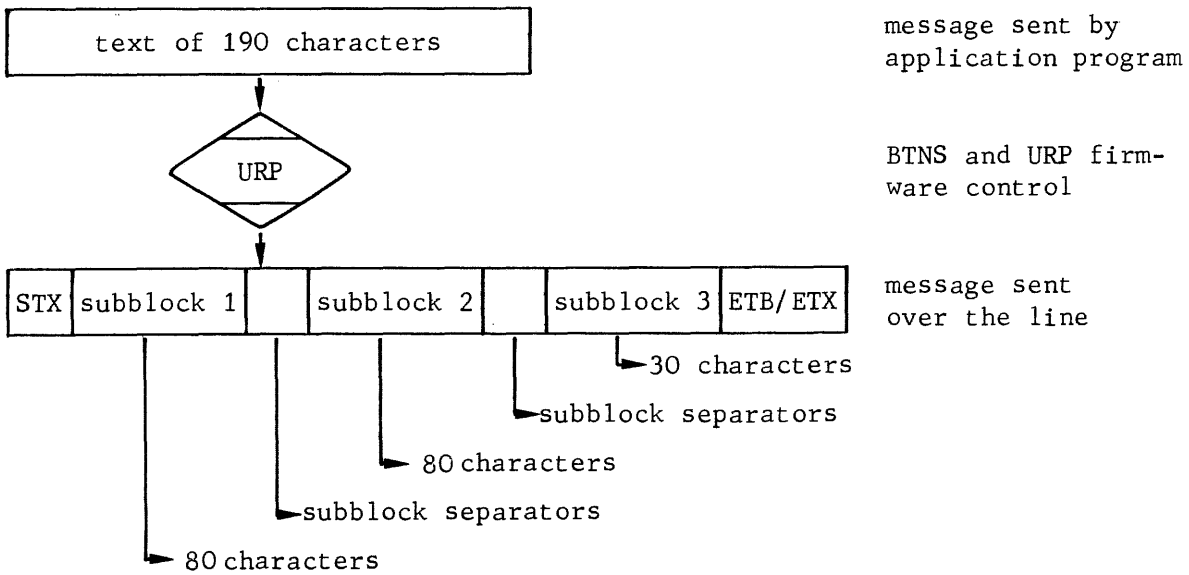
The ability to use the subblock facility over a line is configured at URP firmware generation and is available in both modes of transmission, i.e., normal and transparent.

Message text is handled according to the direction of transmission,

- on reception, subblock separators are deleted



- on emission, the following must be taken into consideration,
  - message blocks are split into subblocks according to the value given to the SBKLG parameter of the appropriate TERMNL command for the linked terminal, the default value being 80
  - the maximum number of subblocks for each message block is 16
  - a line configured without the subblock option at URP firmware generation cannot use the 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.



TTY LINE PROCEDURE

Terminals using the TTY line procedure have limited functionalities in that they have no specific control characters. Communication is performed thru a character by character mode over a point-to-point line with the L64 host computer.

Messages seen by BTNS contain,

- message text
- one trailer control character on input, in some cases.

MESSAGE HANDLING ON INPUT

On input, messages are terminated when any of the following control characters 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 URP firmware and not received by BTNS
- available only on terminals equipped with the "reverse channel" option and connected over a 1200 bps line, i.e.,

  TN1200
  TWU/PRU1005

><LF▽ line feed

><SUB substitute, automatically suppressed by 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 ><CR▽ at the end-of-line to delete the complete line and hence no message is delivered to the application program.

MESSAGE HANDLING ON OUTPUT

On output, messages are terminated on exhaustion of the message length or are divided into fixed length blocks according to the following parameters declared in the TERMNL command,

- BLOCKING
- LLENGTH.

The messages are edited by the application program before they are submitted to BTNS.

TTY LINE PROCEDURE

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 symbol, e.g., TWU/PRU1001 and TWU/PRU1003 use the graphic symbol

Further recovery procedures must be supported by the application program.

The General Electric TermiNet is a printer with the optional features,

- 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 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 type and the options available, namely,

- if TN300, 118 character positions
- if TN1200, either 80 or 120 character positions.

TERMINAL SPECIFICS

Refer to respective product manuals.

TTY
TN300,1200

CONTROL CODES

Unless specifically stated otherwise, the control code pertains 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
><DC2 starts paper tape punch
><DC3 stops paper tape reader
><DC4 stops paper tape punch
><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

><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 type,
- 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

TWU/PRU1001, 1003, 1005

The TWU/PRU1001, 1003, 1005 terminals are printers with the optional features,

- keyboard
- front feed mechanism for inserting 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 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 TWU/PRU1005, the new line movement can be suppressed.

"Paper out" condition leads to disconnection for a switched line.

CONTROL CODES

><BEL sounds an alarm

><BSV moves printer head one position to the left

><CRV 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 tabulation stops maximum)

><ESC2 clears all horizontal tabs (automatic on power-on)

><ESC3 sets a vertical tabulation at current line count (6 lines to the inch, 10 tabulation stops maximum)

><ESC4 clears all vertical tabs

><FSV ejects single sheet, automatically performed when less than 1 inch from bottom of the sheet

><GSV position single sheet to first print position, about half inch from top of the sheet

><HTV moves printer head to next horizontal tabulation or end-of-line

><LFV advances paper one line

><VTV advances paper to next vertical tabulation or top-of-page

TTY
TWU/PRU1001, 1003, 1005

TERMINAL SPECIFICS

Refer to respective product manuals.

The teletype printers have the optional features,

- keyboard
- paper tape attachment

The keyboard can send either 96 or 128 ISO codes, depending on the model.

The character set for the printer comprises 63 graphic symbols, or 94 for the 37 and 38 models.

Line length is either 72 or 86 characters.

CONTROL CODES

The applicability of the control codes depends on the model and 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

><ESC3 selects printing in red

><ESC4 selects printing in black

><ESC5 sets vertical tabulation at line count

><ESC6 clears all vertical tabulations

><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)

TTY
TTY33,35,37,38

TERMINAL SPECIFICS

Refer to respective product manuals.

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; overwise, excess characters overprint the rightmost character position, i.e., 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. An option is 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 upper case is present, then the set is reduced to 94.

> To provide for automatic new line function, the parameter BLOCKING may be specified in the TERMNL command.

CONTROL CODES

- ><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 the cursor one position to the right
- ><FF▽ resets cursor to left margin for erasing the screen
- ><LF▽ "rolls up" lines by one line in order to allow data entry in the bottom line

TTY
VIP7100

TERMINAL SPECIFICS

Refer to respective product manual.

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

><BEL sounds an alarm

><CR▽ resets the cursor at left margin on the same line

><ESC3 sets normal intensity for screen illumination

><ESC4 sets low intensity for screen illumination

><ESCA moves cursor one line up but at the same character position

><ESCB moves cursor one line down but at the same character position

><ESCC moves cursor one character position to the right on the same line

><ESCD moves cursor one character position to the left on the same line

><ESCH resets cursor in "home" position, i.e., at leftmost character position at the top of the screen

><ESCJ erases text display from current cursor position to the end-of-page

><ESCK erases text display from current cursor position to the end-of-line

><ESC\ resets terminal to initial state, as follows,

- screen erased
- cursor in "home" position
- normal intensity for screen illumination

The equivalent control sequence is ><ESC><C79

><ESCf position the cursor according to character position and line number

><ESCI sends data from start of line or page, depending on switch setting, to the current cursor position

><ESCN causes the terminal to indicate its cursor position according to the format given in ><ESCf

><LF▽ moves cursor one line down; if the cursor is on the bottom line, "roll up" will occur

To position the cursor, proceed as follows,

- refer to page 4-31 "Mark Representation" for determining the values to be given to
  - character position
  - line number
- send the control sequence of the format

  ><ESCf followed by character position, then line number

TERMINAL SPECIFICS

Refer to respective product manual.

VIP LINE PROCEDURE

MESSAGE FORMAT AS SEEN BY BTNS

A text message exchanged between BTNS message processing routines and the SURP firmware has the following format :

STI	SOH	ADR	STA	FC1	FC2	STX	message-text	ETB/ETX	EOT

STI station index byte, i.e., the station address as specified in the STATN command at CNC generation

SOH start-of-header
EBCDIC value 01

ADR terminal address as specified in the TERMNL command at CNC generation

STA status code
STA = EBCDIC 00 = NUL, for text only
STA = EBCDIC 3F = PRT, for print message text
see text following for individual terminals

FC1 function-code-1, see text following for individual terminals

FC2 function-code-2, see text following for individual terminals

STX start-of-text
EBCDIC value 02

ETB end-of-transmission-block, only used for loading BTT7300
EBCDIC value 26

ETX end-of-text
EBCDIC value 03

EOT end-of-transmission
EBCDIC value 37

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 COBOL program.
These 3 bytes are STA, FC1 and FC2. BTNS has no effect on FC1 and FC2. Both function codes are unedited either on input or output. SURP firmware uses the TCT (terminal control table) for transcoding them.

The VIP header is visible to the program when one of the following is fulfilled,

- the TERMNL command is specified with either IM = MK or IM = UN
- the network command MTE specifying the terminal and either IMARK or INEDT is keyed in
- the terminal operator command $*$MTE specifying either IMARK or INEDT is keyed in.

The control code received by the program when the above conditions are met, is of the format ><U03abc, see Section IV.

The values a, b and c are sent from the VIP terminal, where,

- a represents the status code
- b represents the first function code
- c represents the second function code.

The user can also include message headers of the same format in output messages. These headers can be included in any output mode.

For the values of VIP header characters, see text following for individual terminals as well as their respective product manuals.

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 application programs, the user should disregard empty messages.

VIP Terminals					
		Type	Model	Type	Model
		TWU/PRU1901	TWU/PRU1901	VIP7700	VIP7700R
		VIP765	VIP765		VIP7700 VIP7705
		VIP775	VIP775		
		VIP785	VIP785	VIP7760	VIP7760

The TWU/PRU1901 is equipped with a keyboard and 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 computer is that of a VIP7700 terminal with limited functionalities.

TWU/PRU1901 KEYBOARD/PRINTER

The page is organized according to parameters entered either by the operator or from the application program.

Printer and medium specifications are,

- a page is composed of from 10 to 90 lines
- a line contains from 38 to 132 characters, default value is 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).

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 computer.

Header for Keyboard/Printer

The 3 parameters in the VIP header have the following specifications,

- STA = EBCDIC 3F, when sent by the terminal, as forced by BTNS
  = EBCDIC 00, as received by the application program
- FC1 = received as sent by the application or overridden by the operator
- FC2 = "▽", when the last character sent has been printed on fanfold paper
  = "!", when the last character has been printed on single forms

Control Codes

><B01 new page function, form feed

><B03 new line function

><CR▽><LF▽ see ><B03

><DC3 set page dimensions according to number of lines and characters/line

><ESC1 set horizontal tabulation, up to 10 tab stops may be set in the 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 vertical tab stop, top-of-form is tab stop

To set page dimensions, a sequence is made up of ><DC3 followed by the number of lines in the form and the number of characters per line. To define the respective numbers required, refer to "ASCII Code for Control Characters & Graphic Symbols" on page 4-32, and proceed as follows,

- for choosing the number of lines from 10 to 90, subtract 9 from the number of lines required, e.g., if 55 lines are required, then the significant value to be defined is (55 - 9) = 46
- for choosing the number of characters per line from 38 to 132, subtract 37 from the number of characters required, e.g., if 80 characters are required, then the significant value to be defined is (80 - 37) = 43
- to define the values as graphic symbols or their EBCDIC equivalents, proceed as follows,
  - start from the ASCII code 20 which is a "space"
  - count as many codes as defined by the "significant value" from the ASCII code 20
  - the ASCII code arrived at defines the number required, e.g., for the "significant value" of 46, the corresponding graphic symbol is M
- the control sequence to send to set the page to 55 lines of 80 characters per line is ><DC3MK.

TWU/PRU1901 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 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 on the previous page applies to the "front feed" printer.

Control Codes

The control codes listed for the normal traction printer on the previous page apply to the "front feed" printer. Additional control codes are,

><DC2 applicable following paper movement, ><B01, ><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 the selection by default at "power on" and is resorted to after ejection of the single sheet by "form feed" or excessive paper skips.

The VIP765, VIP775 and VIP785 are terminals which have a display and keyboard, and are programmed in a similar manner as the VIP7700, with limitations to be found in their respective product manuals.

PRINTER ATTACHED TO VIP765/775/785

The printer is NOT defined by a TERMNL command. Print functions are managed by the application program thru the status of the VIP header.

On the terminal side, the operator can request the application to edit the transmitted message and its printed version by sending the message with the "print" key instead of the "transmit" key. This action results in the PRT status of the VIP header.

Header for Printer

The 3 parameters in the VIP header have the following specifications,

- STA = EBCDIC 3F, denoting PRT status for the printer
  = EBCDIC 00, if there are no print requests
- FC1 = last function code entered on the keyboard or echo to the program
- FC2 = last but one function code or echo to the program

TERMINAL SPECIFICS

Refer to respective product manuals.

VIP7700 DISPLAY/KEYBOARD

The screen is organized in 12 or 24 lines of 80 character positions. An option is available for 22 lines of 46 characters. Including the "space", 64 different graphic symbols, (or, 95 with the lower case option) can be displayed. The types of graphic symbols depend on national keyboard options. An entry marker is displayed as an aid in formatting the screen.

Header for Display/Keyboard

The 3 parameters in the VIP header have the following specifications,

- STA = EBCDIC 00 = COBOL Collating Sequence 1
- FC1 = last function code entered on the keyboard or echo to the program
- FC2 = last but one function code entered or echo to the program.

Text for Display

BLANKING :

><CA1 displayed as a "space" or graphic ~ starts a character string which is blanked out on display.
A "space" or the 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 the end-of-line ends the blinking string.

COBOL :

LINE and PAGE keywords have the same effect as ><BO3 and ><BO1 respectively, when used in a sentence with the SEND verb.

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 ><B01

><HT∇ same line, 1st. tabulation to the right

><LF∇ one line down, same character position

><NL∇ same as ><B03

To position the cursor, see page 4-31 "Mark Representation"

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∇ file separator, followed by a displayable character, begins a fixed field

><GS∇ group separator, begins a variable field followed by a parameter

values of variable field parameter:
0 the field may contain a displayable character to be transmitted and printed
1 printing of the displayable character is to be suppressed
2 transmission of the displayable character is to be suppressed

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 program using the following terminal control codes,

><ESCT start of message boundary

><ESCU end of message boundary

PAGE OVERFLOW :

detection
ERROR indicator on the terminal illuminates.

cause
- result of program error
- result of an operator error in failing to clear the screen

correction
- 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 CASSETTE UNIT

The VIP7700 cassette unit must be defined by a separate TERMNL command specifying STYPE=CAS.

Messages with status 00 (STA=00) in the VIP header can be exchanged between the application and the cassette unit.
Messages sent to the cassette unit must contain the following information,

- selection of the cassette unit
- type of command
- message text, where applicable.

The keyboard remains locked during a cassette operation.

Any text sent to the cassette is displayed on the screen. In order to avoid page overflow, the programmer should systematically prefix the text with ><FF, which clears the screen and which will not be recorded in the data block.

After a read command, the application has to issue a RECEIVE to the program queue to which the cassette unit is assigned in order to receive the block that was read. In blocks read from a cassette, the FC1 and FC2 parameters in the VIP header have the values at the time when the page was recorded.

SELECTION :

><ESCE selects unit 1

><ESCF selects unit 2

COMMAND :

><ESCD backspace a page with no text following

><ESCP rewind to start of cassette tape

><ESCR read a page with no text following

><ESCS write a page followed by text to maximum display size of 1920 characters; if no text follows, the contents of the display memory will be recorded instead

```
Problem  : To write "message text" to cassette unit 1
Solution : Send the following message
           ><ESCE><ESCS"message text"
```

BLANK CASSETTE BLOCK :

Because of compression of trailing spaces by the VIP7700 controller, a cassette data block filled with spaces is transfomed into an empty message which is not delivered to the program queue in normal mode.

In mark mode (IM=MK) a message containing only ><ETX><EOT is received by the program.

VIP7700 PRINTER

Printable text is defined between the start-of-te .t and the end-of-text, which can be denoted by ><EM∇. Text output to the printer does not appear on the screen. The three ways of addressing the printer are treated as follows :

Method 1 : Address the display/keyboard with STA=3F in the VIP header.

The display/keyboard is locked during printing.
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 may be needed for proper mechanical synchronization. Full line size capability can be obtained by this method.

Method 2 : . Define the printer by a separate TERMNL command specifying STYPE=PRT
. Address the display/keyboard with STA=00 in the VIP header.

The keyboard is not locked and may be used for entry into the screen and transmitted 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 3 : . Define the printer by a separate TERMNL command specifying STYPE=PRT
. Address the display/keyboard with STA=3F in the VIP header.

The keyboard is not locked and may be used for entry into the screen and transmitted while printing proceeds.
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 may be needed for proper mechanical synchronization. Full line size capability can be obtained by this method.

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.

Text fcr Display

The control sequences for tabulation, entry marker control and message boundaries are the same as for the VIP7700. Blinking and blanking may be 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 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 parameter

values of variable field parameter:

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

For integer-to-graphic conversion, see
- ASCII Code for Control Characters & Graphic Symbols, page 4-32
- EBCDIC Character Set & COBOL Collating Sequence, page 4-05

VIP7760 DISKETTE

The VIP7760 diskette is not supported on-line.

VIP7760 PRINTER

The printer must be defined by a separate TERMNL command specifying STYPE=PRT. 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
- 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 followed by a count can be used as a "time fill". To define the count, refer to "ASCII Code for Control Characters & Graphic Symbols" on page 4-32, and proceed as follows,
- start from ASCII code 20 which is a "space"
- count as many codes as "time fills" required from ASCII code 20
- the ASCII code arrived at defines the "time fill" count
- use either the graphic symbol or the EBCDIC equivalent of the code.

DISPLAY IMAGE MODE :

Method of addressing :
- address the printer with STA=00 in the VIP header
- 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
- end the message text with ><ESCM.

If the message text begins with ><ESCX, only variable fields are printed.

If the message text begins with ><ESCY, both fixed and variable fields are printed.

The printer is controlled automatically by the VIP7760 system. Text which is not preceded by either ><ESCX or ><ESCY is not printed. Variable fields defined as "inhibit printing" will not be printed.

APPENDIX C

ERROR MESSAGES

Error messages output after the execution of the H_CNC utility are,

- NDL error messages which are written to the SYSOUT file
- JOR (job occurrence report) error messages.

Error messages output after the execution of the H_QMAINT utility appear in the job occurrence report and are listed under

- QMAINT error messages.

NDL and QMAINT ERROR MESSAGES

Both types of error messages have the same format, namely,

ERROR { CN / QC } [nnnn] SEVERITY (s) error-message-text

- CN denotes that the source is H_CNC utility
- QC denotes that the source is H_QMAINT utility
- nnnn denotes the message number
- s denotes the severity of the error as follows,
  - 2 : warning, CNC generation may still be usable
  - 3 : fatal, leading to a CNC abort but allows a complete syntax analysis of the network description
  - 4 : fatal, leading to a CNC abort and may stop command analysis.
- error-message-text denotes the error condition and may be accompanied by the contents of the RC register of the following format :
  RC = xxxxxxxx→yyyyyyyy, zzzzzzzz
  - xxxxxxxx : the hexadecimal contents of the RC register
  - yyyyyyyy : the name of the procedure, SIU (system integration unit)
  - zzzzzzzz : the return code

JOR ERROR MESSAGES

The format of the message is: **XXnn** error-message-text

- **XXnn** denotes a 2-alpha character code followed by a 2-digit number
- error-message-text denotes the error condition and may be accompanied by the contents of the RC register of the format given above.

The 2-alpha character code preceding the messages indicates their source, namely,

- CC messages are from BTNS
- CN messages are from CNC
- QC messages are from MCS COBOL application.

RETURN CODES

The list of return codes are those set by MAM. For GCOS return codes, refer to JCL User Guide Manual.

NDL ERROR MESSAGES
0000 - 0004

- ERROR CN [0000] SEVERITY (4) ILLEGAL SYNTAX

  syntax : ↑ denotes the element in error.
  cause : . wrong command name, keyword or argument.
  . missing or duplicated positional parameter, i.e., name, LNnn, station-index, TYPE and STYPE.
  . misplaced GENCOM command.
  . violation of CNC reserved syntax, see Appendix E.
  action : correct the command and rerun H_CNC utility.

- ERROR CN [0001] SEVERITY (3) NO QUEUE FOR TERMINAL : terminal-name

  syntax : "terminal-name" appears in the appropriate TERMNL command.
  cause : no terminal queue has been declared for the terminal having output capability and dedicated to a program queue (ASSIGN).
  action : insert a QUEUE command bearing the "terminal-name" and other relevant parameters for the queue declared, and rerun the H_CNC utility.

- ERROR CN [0002] SEVERITY (3) SEVERAL TERMNL FOR A TTY LINE

  syntax : as denoted
  cause : only 1 TERMNL command is allowed after a TTY LINE command.
  action : remove superfluous TERMNL commands and rerun H_CNC utility.

- ERROR CN [0003] SEVERITY (3) DUPLICATED : name

  syntax : "name" can be that of the DCTAP, GENCOM, LINE, QUEUE, STATN and TERMNL commands.
  cause : "names" declared for the commands listed above must be unique for a network; the only allowed duplication is the terminal-name and its associated terminal-queue-name.
  action : substitute "name" indicated in the appropriate command and rerun H_CNC utility.

- ERROR CN [0004] SEVERITY (3) INVALID NAME LENGTH : name

  syntax : "name" can be that of the DCTAP, GENCOM, LINE, QUEUE, STATN and TERMNL commands.
  cause : "names" declared for the commands listed above must be limited to a maximum of 4 alphanumeric characters; only "password" defined by the KEY parameter of the GENCOM command may comprise up to 10 alphanumeric characters.
  action : correct "name" indicated in the appropriate command and rerun H_CNC utility.

- ERROR CN [0005] SEVERITY (3) QCPOOL NOT DEFINED BY GENCOM

  syntax : as denoted
  cause : a queue has been declared with a QCPOOL option in the appropriate QUEUE command while a QCPOOL option has not been specified in the GENCOM command.
  action : insert the QCPOOL option in the GENCOM command and rerun H_CNC utility.

- ERROR CN [0006] SEVERITY (3) STATN NOT ALLOWED ON A TTY LINE

  syntax : as denoted
  cause : as denoted
  action : remove the STATN command following all TTY LINE commands and rerun H_CNC utility.

- ERROR CN [0007] SEVERITY (3) MAMNB DECLARED BY GENCOM WHILE NO QUEUE DECLARED

  syntax : as denoted
  cause : the GENCOM command declares a number of MCS program simultaneities while no queue has been declared.
  action : insert appropriate QUEUE command(s) or remove MAMNB from GENCOM command and rerun H_CNC utility.

- ERROR CN [0008] SEVERITY (3) INVALID VALUE : value

  syntax : "value" indicated appears against the parameter of the command in error.
  cause : "value" is outside of the range allowed by the appropriate command.
  action : correct the value for the parameter in the appropriate command and rerun H_CNC utility.

- ERROR CN [0009] SEVERITY (3) MISSING STATN OR TERMNL

  syntax : as denoted
  cause : a LINE command must be followed by one of either,
  . STATN if it describes a multipoint or polled line
  . TERMNL if it describes a TTY line
  action : insert the appropriate command and rerun H_CNC utility.

- ERROR CN [0010] SEVERITY (2) ONLY TERMINAL QUEUES DECLARED

  syntax : as denoted
  cause : message switching only thru terminal queues is allowed because no program queues have been declared.
  action : none

- ERROR CN [0011] SEVERITY (3) MISSING LINE OR STATN

  syntax : as denoted
  cause : a TERMNL command must immediately follow one of either,
  . a TTY LINE command
  . a STATN command for all other line procedures.
  action : insert the appropriate command and rerun H_CNC utility.

NDL ERROR MESSAGES
0012 - 0017

- ERROR CN [0012] SEVERITY (3) DUPLICATED KEYWORD

  syntax : as denoted
  cause : a keyword has occurred more than once in a command.
  action : correct the command and rerun H_CNC utility.

- ERROR CN [0013] SEVERITY (4) UNABLE TO CREATE : name
  RC = xxxxxxxx ► yyyyyyyy, zzzzzzzz

  syntax : "name" can be that of a table, buffer or work segment. the contents of G4 specifies the reason for the abort.
  cause : a system malfunction has prevented CNC from creating the table, buffer or work segment identified.
  action 1 : if zzzzzzzz = NEXPDERR, use the OCL command CMM to cancel VCAM or MAM, and then retry the job step.

  action 2 : if the same error persists, or if zzzzzzzz is not NEXPDERR, then call the field engineering service.

- ERROR CN [0014] SEVERITY (3) MISSING GENCOM

  syntax : as denoted
  cause : the GENCOM command must be present and must head the list of NDL commands.
  action : insert the GENCOM command or correct the sequence of NDL commands, if the GENCOM command is present, and rerun H_CNC utility.

- ERROR CN [0015] SEVERITY (3) LINE NOT DEFINED OR CONNECTED TO SRST

  syntax : as denoted
  cause : the name "LNnn" of a LINE command is not present in the SRST or is not available.
  action : call field engineering service.

- ERROR CN [0016] SEVERITY (3) LINE RESERVED FOR SYSTEM CONSOLE

  syntax : as denoted
  cause : the name of a LINE command is of the type CSP in the SRST and denotes the line exclusively reserved for the system console.
  action : correct the command and rerun H_CNC utility.

- ERROR CN [0017] SEVERITY (3) NO MASTER TERMINAL DECLARED

  syntax : as denoted
  cause : a TERMNL command specified as TYPE=SLAVE follows immediately a LINE or STATN command.
  action : insert a TERMNL command for a "master" terminal before the TERMNL command for a "slave" terminal and rerun H_CNC utility.

- ERROR CN [0018] SEVERITY (3) NOT A QUEUE OR TAP NAME : name

  syntax : "name" is the argument specified for the ASSIGN parameter of a TERMNL command.
  cause : the "name" does not match any of the following,
  - program queue as declared by a QUEUE command
  - VCAM subsystem, i.e., IOF or TDS, as declared by the DCTAP command.

  action : correct the command and rerun H_CNC utility.

- ERROR CN [0019] SEVERITY (3) POLLIST GREATER THAN DECLARED IN SRST

  syntax : as denoted
  cause : the value declared for the POLLIST parameter of a LINE command defines more station index entries than those reserved in the URP for that line. (rf. POLLISTLENGTH parameter of SRST entry)
  action : reduce the number of station index entries in the appropriate LINE command and rerun H_CNC utility.

- ERROR CN [0020] SEVERITY (4) SEGMENT TOO LARGE : name

  syntax : "name" identifies one of the following,
  - the BTNS buffer pool
  - the VCAM buffer pool
  - core queues

  cause : the value(s) declared for any of the above parameters has exceeded the allowable limit.
  action : adjust the size of the parameter(s) in either the GENCOM or the QUEUE command and rerun H_CNC utility.

- ERROR CN [0021] SEVERITY (4) UNABLE TO ACCESS SYSIN
  RC = xxxxxxxx→yyyyyyyy,zzzzzzzz

  syntax : the contents of G4 specifies the reason for the abort.
  cause : system error
  action : check JCL statements and retry the job : if the same condition occurs, then call the field engineering service.

- ERROR CN [0022] SEVERITY (2) ONLY PROGRAM QUEUES DECLARED

  syntax : as denoted
  cause : MCS applications can only perform data collection because no distribution capability in the form of terminal queues has been specified.
  action : none

NDL ERROR MESSAGES
0023 - 0026

- ERROR CN [0023] SEVERITY (4) QUEUES FILE ACCESS ERROR :
  RC = xxxxxxxx→yyyyyyyy,zzzzzzzz

  syntax : the contents of RC specifies the error and the system primitive.
  cause : the error when accessing a queues files can be one of either,
  - a user error, i.e., file not preallocated
  - a system error (IOFAIL).

  action :
  - if file is not preallocated, then correct as appropriate and retry the job.
  - if IOFAIL, retry the job : if the same condition occurs, then call field engineering service.

- ERROR CN [0024] SEVERITY (4) NON STANDARD DISK

  syntax : as denoted
  cause : the disk queues file is preallocated on a device other than MS/M300, 350, 400 or 402.
  action : correct the appropriate JCL statement and retry the job.

- ERROR CN [0025] SEVERITY (4) QUEUES FILE TOO LARGE

  syntax : as denoted
  cause : the size of preallocated disk queues file has exceeded the allowable limit.
  action : adjust the file size in the appropriate QUEUE command and rerun H_CNC utility.
  (see Section V "Tuning the Network")

- ERROR CN [0026] SEVERITY (3) PARAMETERS CONFLICT : text

  syntax : "text" defines conflicts between parameters of the same or different commands.

  ° BKMOD/LINE TYPE
  action : check BKMOD parameter of the LINE command pertains to TTY terminals equipped with the BREAK key.
  correct the appropriate LINE command and rerun H_CNC utility.

  ° CLV/LINE TYPE
  action : check that the value 3, i.e., 5 bit code, has not been specified for CLV for a synchronous line procedure.
  correct the appropriate LINE command and rerun H_CNC utility.

  ° CORE QUEUE/RESTART
  action : check RESTART parameter of the QUEUE command is not specified for core queues which are defined by,
  - the QCBLKSZ and QCPOOL parameters of the GENCOM command
  - the NUMBLK parameter of the QUEUE command.

  correct the relevant command(s) and rerun H_CNC utility.

- ERROR CN [0026] SEVERITY (3) PARAMETERS CONFLICT : text

  syntax : "text" defines conflicts between parameters of the same or different commands.

  ° EPX/LINE TYPE
  action : check EPX parameter of the LINE command pertains only to TTY37 type terminals which allow echoplex.
  correct the appropriate LINE command and rerun H_CNC utility.

  ° ERCAP/LINE TYPE
  action : check ERCAP parameter of the LINE command pertains only to TTY terminals which have "erase" capability.
  correct the appropriate LINE command and rerun H_CNC utility.

  ° ININS/LINE TYPE
  action : check ININS parameter of the LINE command pertains only to synchronous line procedures which require "synchronization".
  correct the appropriate LINE command and rerun H_CNC utility.

  ° LINPLG/LINE TYPE
  action : check LINPLG parameter of the LINE command has not been specified for a TTY line procedure.
  correct the appropriate LINE command and rerun H_CNC utility.

  ° NAUCDE/LINE TYPE
  action : check NAUCDE parameter of the LINE command pertains only to a "switched" line and has not been specified for a line having 108/2 CCITT interface.
  correct the appropriate LINE command and rerun H_CNC utility.

  ° OPERATOR/CLOSE
  cause : the network control terminal identified as OPER in the appropriate TERMNL command has been declared as a member of a LINE or STATN specified with the CLOSE option.
  action : correct the appropriate TERMNL command and rerun H_CNC utility.

  ° OPERATOR/LINE TYPE
  cause : the network control terminal identified as OPER in the appropriate TERMNL command is connected over a "switched" line.
  action : correct the appropriate TERMNL command and rerun H_CNC utility.

NDL ERROR MESSAGES
0026

- ERROR CN [0026] SEVERITY (3) PARAMETERS CONFLICT : text

  syntax : "text" defines conflicts between parameters of the same or different commands.

  ° OPERATOR/SLAVE
  cause : the network control terminal identified as OPER in the appropriate TERMNL command has been declared SLAVE.
  action : correct the appropriate TERMNL command and rerun H_CNC.

  ° OPERATOR/TYPE,STYPE
  cause : the network control terminal identified as OPER in the appropriate TERMNL command has been specified with TYPE and STYPE contradictory to network control function.
  . TYPE must be other than HL62, HL64 and HL66.
  . STYPE must only be KCT or KPR.
  action : correct the appropriate TERMNL command and rerun H_CNC.

  ° POLLIST/LINE TYPE
  action : check POLLIST parameter of the LINE command has been specified for a "multipoint" line and has not been used for a TTY line procedure.
  correct the appropriate LINE command and rerun H_CNC.

  ° PROG. QUEUE/CTLRST
  cause : only disk terminals can be declared with CTLRST option.
  action : correct the appropriate QUEUE command and rerun H_CNC.

  ° RESTART/CTLRST
  cause : a disk queue cannot be declared with both CTLRST and RESTART options.
  action : correct the appropriate QUEUE command and rerun H_CNC.

  ° SLAVE/ASSIGN
  cause : a terminal declared SLAVE in the TERMNL command must immediately follow a "master" terminal.
  action : correct the appropriate TERMNL command, or insert a "master" TERMNL command before the command in question, or check the NDL command sequence and rerun H_CNC utility.

  ° SLAVE/AUTO
  action : check AUTO parameter of the TERMNL command pertains only to a "receive-only" or "master" terminal, and not to a terminal declared SLAVE.
  correct the appropriate TERMNL command and rerun H_CNC.

  ° SLAVE/CONTROL
  action : check CONTROL parameter of the TERMNL command has not been specified for a terminal declared SLAVE.
  correct the appropriate TERMNL command and rerun H_CNC.

- ERROR CN [0026] SEVERITY (3) PARAMETERS CONFLICT : text

  syntax : "text" defines conflicts between parameters of the same or different commands.

  ° SPEED/LINE TYPE
  action : check SPEED parameter of the LINE command has not been specified for a synchronous line procedure.
  correct the appropriate LINE command and rerun H_CNC.

  ° SPN/LINE TYPE
  action : check SPN parameter of the LINE command pertains only to an asynchronous line procedure.
  correct the appropriate LINE command and rerun H_CNC utility.

  ° TERM. QUEUE/BREAK
  cause : the BREAK parameter of the QUEUE command pertains only to program queues.
  action : correct the appropriate QUEUE command and rerun H_CNC.

  ° TERM. QUEUE/SHARE
  cause : the SHARE parameter of the QUEUE command pertains only to program queues.
  action : correct the appropriate QUEUE command and rerun H_CNC utility.

  ° TERM. QUEUE/TWA
  cause : the TWA parameter of the QUEUE command pertains only to program queues.
  action : correct the appropriate QUEUE command and rerun H_CNC.

  ° TYPE/ADD
  cause : the argument, i.e., the 2-hexadecimal digit defining the address of a "compatible" terminal, specified for the ADD parameter of the TERMNL command does not correspond with an existing value in the configuration.
  action : correct the appropriate TERMNL command and rerun H_CNC utility.

  ° TYPE/LINE TYPE
  cause : the "terminal-type" specified for the TYPE parameter of the TERMNL command does not correspond to a terminal supported by the line procedure.
  action : correct the appropriate TERMNL command and rerun H_CNC.

  ° TYPE/STYPE
  cause : the "subtype" specified for the STYPE parameter of the TERMNL command is not allowed for the "terminal-type" specified for the TYPE parameter in the same TERMNL command.
  action : correct the appropriate TERMNL command and rerun H_CNC utility.

NDL ERROR MESSAGES
0027 - 0031

- ERROR CN [0027] SEVERITY (3) DUPLICATED STI

  syntax : as denoted
  cause : the same "station-index" has been used in more than 1 STATN command declared for the same line.
  action : correct the appropriate STATN command and rerun H_CNC utility.

- ERROR CN [0028] SEVERITY (2) STI NOT SPECIFIED IN POLLING LIST (LINE COMMAND)

  syntax : as denoted
  cause : the "station-index" declared in a STATN command has not been included in the "polling-list" of the POLLIST parameter in the preceding LINE command.
  action : . the network control terminal operator can use the MTP command to resume polling for the station.
  . none

- ERROR CN [0029] SEVERITY (3) POLLIST CONTAINS STI OF NOT DECLARED STATIONS

  syntax : as denoted
  cause : the "polling-list" of the POLLIST parameter in the LINE command has included a "station-index" for which no subsequent STATN command has been declared.
  action : amend the "polling-list" of the appropriate LINE command or include the appropriate STATN command to follow the LINE command, and rerun H_CNC utility.

- ERROR CN [0030] SEVERITY (3) QCPOOL DECLARED BY GENCOM WHILE NO QUEUE USES IT

  syntax : as denoted
  cause : a pool of core blocks has been reserved by the QCPOOL parameter of the GENCOM command but no subsequent QUEUE commands are declared with the QCPOOL option, resulting in loss of memory space.
  action : amend the GENCOM command or QUEUE command(s) and rerun H_CNC utility.

- ERROR CN [0031] SEVERITY (3) DECLARED SPEED DOES NOT EXIST ON THE UCLA (SRST)

  syntax : as denoted
  cause : the value specified for the SPEED parameter of the LINE command defines the line speed for which the DCC (data communications controller) is not configured.
  action : correct the appropriate LINE command and rerun H_CNC utility.

- ERROR CN [0032] SEVERITY (3) QCBLKSZ DECLARED BY GENCOM WHILE NO CORE QUEUE DECLARED

  syntax : as denoted
  cause : . core block size has been declared in the QCBLKSZ parameter of the GENCOM command while no subsequent core queues have been specified.
  . the "core queue" is further defined by,
  - the QCPOOL parameter of the GENCOM command
  - the NUMBLK parameter of the QUEUE command.
  action : amend the GENCOM command or QUEUE command(s) and rerun H_CNC utility.

- ERROR CN [0033] SEVERITY (3) QDBLKSZ DECLARED BY GENCOM WHILE NO DISK QUEUE DECLARED

  syntax : as denoted
  cause : . disk queue file block size has been declared in the QDBLKSZ parameter of the GENCOM command while no subsequent disk queues have been specified.
  . the "disk queue" is further defined by the NUMREC parameter of the QUEUE command.
  action : amend the GENCOM command or QUEUE command(s) and rerun H_CNC utility.

- ERROR CN [0034] SEVERITY (2) NEITHER PROGRAM QUEUE NOR TAP DECLARED

  syntax : no means to log on an application.
  action : none, if the CNC generation is usable.

- ERROR CN [0035] SEVERITY (3) PROCEDURE NOT SUPPORTED BY URP IMAGE (NROFTCTTOTAL)

  syntax : NROFTCTTOTAL is an SRST entry to the URP.
  cause : the value specified for the TCTNM parameter of the LINE command must be greater than the value specified for NROFTCTTOTAL.
  action : call field engineering service.

- ERROR CN [0036] SEVERITY (4) UNABLE TO GET SEMAPHORE
  RC = xxxxxxxx►yyyyyyyy,zzzzzzzz

  syntax : the contents of G4 defines the system error.
  cause : system error
  action : call field engineering service.

- ERROR CN [0037] SEVERITY (3) UNABLE TO READ LINE TLT
  RC = xxxxxxxx►yyyyyyyy,zzzzzzzz

  syntax : the contents of G4 defines the system error.
  cause : system error
  action : call field engineering service.

NDL ERROR MESSAGES
0038 - 0044

- ERROR CN [0038] SEVERITY (3) OPER TERMINAL DECLARED ON A CLOSED LINE OR STATION

  syntax : as denoted
  cause : the network control terminal designated OPER in the TERMNL command cannot be declared on a LINE or STATION for which the CLOSE option has been specified.
  action : amend either LINE or STATN commands and rerun H_CNC.

- ERROR CN [0039] SEVERITY (3) AUTO TERMINAL ASSIGNED TO IOF

  syntax : as denoted
  cause : a terminal cannot be declared ASSIGN=IOF as well as AUTO because the VCAM subsystems are automatically scheduled when the terminal becomes "on-line".
  action : correct the appropriate TERMNL command and rerun H_CNC.

- ERROR CN [0040] SEVERITY (2) MEANINGLESS IN RELEASE 1.D

  syntax : as denoted
  cause : PTYPE option of LINE command has been retained in Release 0400 only for compatibility purposes with 0300.
  action : delete the PTYPE option from the appropriate LINE command; the H_CNC run is nevertheless correct.

- ERROR CN [0041] SEVERITY (2) REPLACED BY CLOSE IN RELEASE 1.D

  syntax : as denoted
  cause : INITST parameter has been replaced by CLOSE parameter for Release 0400.
  action : correct the appropriate LINE/QUEUE/STATN/TERMNL command by either method,
  . replace INITST=INACTIVE by CLOSE
  . suppress INITST=ACTIVE.
  The H_CNC run is nevertheless correct.

- ERROR CN [0042] SEVERITY (2) REPORTED TO STEP LEVEL IN RELEASE 1.D

  syntax : as denoted
  cause : SIMU option must be provided thru the $STEP H_CNC.
  action : amend the $STEP H_CNC in the sequence of JCL statements.

- ERROR CN [0043] SEVERITY (3) BUFFER UNIT SIZE INCOMPATIBLE WITH LINE SPEED

  syntax : as denoted
  cause : the buffer unit size specified thru BTBFSZ or DABFSZ must be greater than 128 for lines whose speed is greater than 9600 bauds.
  action : correct the relevant parameter in the GENCOM command to increase the buffer unit size and rerun H_CNC.

- ERROR CN [0044] SEVERITY (2) MIN NUMBER OF BUFFER UNITS FORCED BY GENERATOR

  syntax : as denoted
  cause : the number of buffer units specified by NBBTBF or NBDABF is less than the minimum required by BTNS according to the number and type of lines.
  action : increase the relevant parameter of the GENCOM command and rerun H_CNC; for the current run, H_CNC has forced the number of buffer units to the minimum required.

- ERROR QC [0103] SEVERITY (4) ILLEGAL SYNTAX

  syntax : ↑ denotes the element in error
  cause : wrong command name, keyword or argument; violation of QMAINT reserved syntax, see Appendix E.
  action : correct the error in the appropriate command and rerun H_QMAINT utility.

- ERROR QC [0110] SEVERITY (4) END OF MESSAGE STRING MISSING

  syntax : as denoted
  cause : after a SEND command, the text of the message must be terminated by one of the following methods,
  - the SEND command must contain the ENDMSG option
  - a card delimiter after the last message text card specifying //EOM starting at column 1.

  action :
  . insert the ENDMSG option in the SEND command and rerun H_QMAINT utility
  . insert the delimiter //EOM at the end of the message text and rerun H_QMAINT utility.

- ERROR QC [0113] SEVERITY (3) MESSAGE SENT TOO LONG

  syntax : as denoted
  cause : an attempt has been made to send a message text longer than the maximum acceptable to MAM, see "Control of Message and Queue Overflow" in Section III.
  action : truncate the message to be sent and rerun H_QMAINT utility.

- ERROR QC [0201] SEVERITY (4) UNABLE TO ACCESS SYSIN
  RC = xxxxxxxx →yyyyyyyy,zzzzzzzz

  syntax : the contents of G4 specifies the reason for the abort.
  cause : system error
  action :
  . check $ASSIGN statement (ASSIGN H_CR) and retry the job
  . if error persists, then call the field engineering service.

- ERROR QC [0302] SEVERITY (2) INVALID NAME LENGTH : name

  syntax : "name" specifies the external-queue-name used in the QMAINT command concerned.
  cause : "name" must obey the following rules,
  . limited to up to 4 alphanumeric characters
  . specified previously in a corresponding QUEUE command, see Section V.

  action : correct the queue name in the appropriate QMAINT command and rerun H_QMAINT utility.

QMAINT ERROR MESSAGES
0304 - 0409

- ERROR QC [0304] SEVERITY (3) DUPLICATE KEYWORD

  syntax : as denoted
  cause : a keyword has occurred more than once in a command.
  action : correct the appropriate command and rerun H_QMAINT utility.

- ERROR QC [0307] SEVERITY (2) QUEUE UNKNOWN : name

  syntax : "name" specifies the external-queue-name used in the QMAINT command concerned.
  cause : "name" is not defined for a queue in the network declared.
  action : check "name" against the list of all external-queue-names defined for the network.
  correct the "name" in the appropriate QMAINT command and rerun H_QMAINT utility.

- ERROR QC [0308] SEVERITY (2) QUEUE NOT AVAILABLE : name

  syntax : "name" specifies the external-queue-name used in the QMAINT command concerned.
  cause : "name" identifies a queue currently allocated to another application program, i.e., if a program queue, or to a terminal still logged on, i.e., if a terminal queue.
  action : wait until the queue identified becomes available and then rerun H_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 H_QMAINT was unable to create.
  the contents of G4 specifies the reason for the abort.
  cause : system error
  action : retry : if the same condition persists, then call the field engineering service.

- ERROR QC [0409] SEVERITY (3) ABNORMAL RECEIVE FROM QUEUE : name
  RC = xxxxxxxx →yyyyyyyy,zzzzzzzz

  syntax : "name" specifies the external-queue-name to which a RECEIVE was issued.
  the contents of G4 specifies the reason for the abort.
  cause : system error (MAM)
  action : retry : if the same condition persists, then 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-name to which a SEND was issued.
  the contents of G4 specifies the reason for the abort.
  cause : system error (MAM)
  action : retry : if the same condition persists, then call the field engineering service.

CC16 BTNS ERROR RC = xxxxxxxx→yyyyyyyy,zzzzzzzz

syntax : the contents of RC specifies the system error.
cause : abnormal termination of BTNS due to system error.
action 1 : if zzzzzzzz = CONFLICT, BTNS has been started for a network without LINE definition; correct NDL description, rerun H_CNC.
action 2 : for any other value of zzzzzzzz, call field engineering service.

CC17 SITE CONTROLLER FLAW : LINK SEM. POOL OVERLOADED

syntax : as denoted
cause : BTNS site controller has aborted due to an overflow on the telecommunications semaphore segment, i.e., no more free links are available.
action : notify field engineering service.

CC23 UNABLE TO ACCESS NETWORK FILE RC = xxxxxxxx→yyyyyyyy,zzzzzzzz

syntax : the contents of RC specifies the error and the system primitive.
cause : BTNS has been unable to access the network file created by H_CNC utility during the start-up phase.
action : retry : if repeatedly unsuccessful,
- perform ISL with "restart clean" option
- generate the network.

CC24 BTNS ALREADY ACTIVE

syntax : as denoted
cause : only 1 BTNS step may be active at any one time.
action : terminate the current version by the TT [STRONG] network control command and then restart a new version by the ST network control command.

CC29 TRANSMISSION CONTROL BUFFER OVERLOAD : ssssssss

syntax : ssssssss is the hexadecimal segment address of the buffer in question.
cause : transmission by BTNS has failed due to lack of buffer units,
- either in the BTNS buffer pool, i.e., segment D.44
- or in the VCAM buffer pool, i.e., segment D.43.

action : increase the number of buffer units of the segment in question using either NBBTBF or NBDABF parameters of the GENCOM command. correct the GENCOM command and rerun H_CNC utility.

CC34 BTNS IRRECOVERABLE ERROR, REASON : xxxxxxxx

syntax : **xxxxxxxx** gives the reason for failure
cause : error in BTNS
action : call field engineering service.

CN01 UNABLE TO FORMAT QUEUE FILE RC = xxxxxxxx►yyyyyyyy,zzzzzzzz

syntax : the contents of RC specifies the error and system primitive.
cause : the attempted CNC generation has been aborted due to a disk error while formatting the disk queue file.
action : retry : if the failure recurs, reallocate the file in a different area of the same disk or to another disk.

CN02 ERROR DURING {CLOSE|OPEN|PUT} SYSOUT RC = xxxxxxxx►yyyyyyyy,zzzzzzzz

syntax : the contents of RC specifies the system error.
cause : the attempted CNC generation has been aborted due to the error appropriately indicated during a SYSOUT file access.
action : retry

CN03 UNABLE TO CREATE NETWORK FILE RC = xxxxxxxx►yyyyyyyy,zzzzzzzz

syntax : as denoted
cause : the attempted CNC generation has been aborted due to an abnormal condition while copying the telecommunications system tables into backing store.
action : retry : if repeatedly unsuccessful,
- perform ISL with "restart clean" option
- generate the network

CN04 TELECOMMUNICATIONS SESSION IN PROGRESS

syntax : as denoted
cause : the attempt to execute the H_CNC utility has been aborted while BTNS, TDS, an MCS COBOL step or another CNC session is running.
action : try later.

CN05 INVALID OPTIONS STRING

syntax : as denoted
cause : the option specified in the CNC $STEP statement is other than OPTIONS = 'SIMU'.
action : check the CNC $STEP statement and retry.

DA00 VCAM NOT AVAILABLE

syntax : as denoted
cause : the attempt to start BTNS, IOF, ROF or TDS, or batch entry has been aborted because no network has been created.
action : run H_CNC utility.

JOR ERROR MESSAGES

QC00 MAM NOT AVAILABLE

syntax : as denoted
cause : the attempted MCS COBOL step has been aborted because,
- the H_CNC utility has not been run to create the network
- a CNC session is running
- the step is multiprocess and was not linked with the Linker command LINKTYPE=BMAM.

action : perform as appropriate,
- run H_CNC utility to create the network
- try later
- link the step.

QC01 MAXIMUM NUMBER OF QASSIGN STATEMENTS IS EXCEEDED

syntax : as denoted
cause : the attempted MCS COBOL step has been aborted due to more than 24 $QASSIGN statements in the step enclosure.
action : respecify JCL and retry.

QC02 external-queue-name UNKNOWN EXTERNAL QUEUE NAME

syntax : as denoted
cause : the attempted MCS COBOL step has been aborted because a $QASSIGN statement specifies an "external-queue-name" that is not defined in the network description.
action :
- respecify NDL and rerun H_CNC utility.
- retry the MCS COBOL step.

QC03 external-queue-name QUEUE NOT AVAILABLE

syntax : as denoted
cause : the attempted MCS COBOL step has been aborted because a $QASSIGN statement specifies an "external-queue-name" currently allocated to another MCS COBOL step.
action : retry later.

QC04 symbolic-queue-name DUPLICATE SYMBOLIC QUEUE NAME

syntax : as denoted
cause : the attempted MCS COBOL step has been aborted because a $QASSIGN statement specifies a "symbolic-queue-name" that has already been assigned in another $QASSIGN statement of the same step.
action : respecify JCL and retry.

QC05 external-queue-name DUPLICATE QUEUE NAME

syntax : as denoted
cause : the attempted MCS COBOL step has been aborted because a $QASSIGN statement specifies an "external-queue-name" that has already been assigned in another $QASSIGN statement of the same step.
action : respecify JCL and retry.

QC07 MAM : ABORT USER RC=xxxxxxxx→yyyyyyyy,zzzzzzzz

syntax : the contents of RC specifies the system error.
cause : the MCS COBOL step has been aborted due to an error condition at run-time.
action : see "List of Return Codes"

QC08 process-name : UNABLE TO START USER PROCESS

syntax : applicable only to a multiprocess step
cause : the MCS COBOL step has been aborted because a user process appropriately identified cannot be started.
action : check the report of the linking process for the load module.

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 : proceed as follows,
- perform ISL with "restart clean" option
- generate the network.

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 : proceed as follows,
- perform ISL with "restart clean" option
- generate the network.

QC11 QUEUES FILE MEDIA BUSY/NOT AVAILABLE

syntax : as denoted
cause : the disk queues file is not available for MCS processing or has not been mounted.
action : perform as appropriate,
- retry later
- mount the disk medium as specified.

QC12 MAXIMUM NUMBER OF MAM PROCESS GROUPS EXCEEDED

syntax : as denoted
cause : the number of MCS COBOL process groups has exceeded the number specified for the MAMNB parameter of the GENCOM command.
action : correct the GENCOM (NDL) command and rerun H_CNC utility.

QC14 INVALID KEYWORD REPLY IN QASSIGN

syntax : as denoted
cause : a $QASSIGN statement includes a "reply" keyword which does not specify a program queue.
action : respecify JCL and retry.

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 persists, then call the field engineering service.

ABTPRC

Code Definition : an invalid internal condition, e.g., invalid data or data out-of-range, was detected during processing a disk I/O request with a result that file processing is now discontinued.
Operator Action : take a dump and notify field engineering service.

CONFLICT

Code Definition : BTNS has been started for a network generated without LINE definition.
Action : Operator - none
Programmer - correct NDL pack and rerun H_CNC utility

COUNTOV

Code Definition : the threshold of I/O errors defined at initialization has been exceeded with a result that the pending I/O operation is not executed and a shutdown has occurred.
Operator Action : proceed as follows,
- try to copy the file affected
- if the file fails to be copied, then a new file must be preallocated, the system reinitialized and the network regenerated.

CPERR

Code Definition : a channel program error or hardware malfunction has occurred.
Operator Action : proceed as follows,
- retry the step
- if unsuccessful, notify field engineering service.

CPOV

Code Definition : an internal error while trying to start multiple channel programs simultaneously has occurred.
Operator Action : notify field engineering service.

EXTERR

Code Definition : a request has been made to read or write a record outside the limits of the allocated file.
Operator Action : proceed as follows,
- retry the step
- if unsuccessful, notify field engineering service.

LIST OF RETURN CODES
continued

MDNAV

Code Definition : the file was not in the "ready" state when accessed by an I/O operation.
Operator Action : proceed as follows,
- set the drive to the "ready" state or mount the disk on another drive and set it to "ready"
- retry the step with MAM=YES.

MSGOV

Code Definition : a disk I/O request specifying an invalid number of records, e.g., less than 0 or greater than 5, has been attempted.
Operator Action : take a dump and notify field engineering service.

NEXPDERR

Code Definition : either VCAM or MAM has been pre-loaded and "locked" in memory thereby preventing the execution of the H_CNC utility.
Operator Action : use the CMM command to cancel the appropriate "locked" segment and rerun H_CNC utility.

TABOV

Code Definition : an internal system error has occurred.
Operator Action : notify field engineering service.

APPENDIX D

MCS COBOL PROGRAM EXAMPLE

The program is an example of validating data exchanges with
- VIP7700 : printer and cassettes (1 or 2).

The program exchanges data with one terminal at a time as follows:

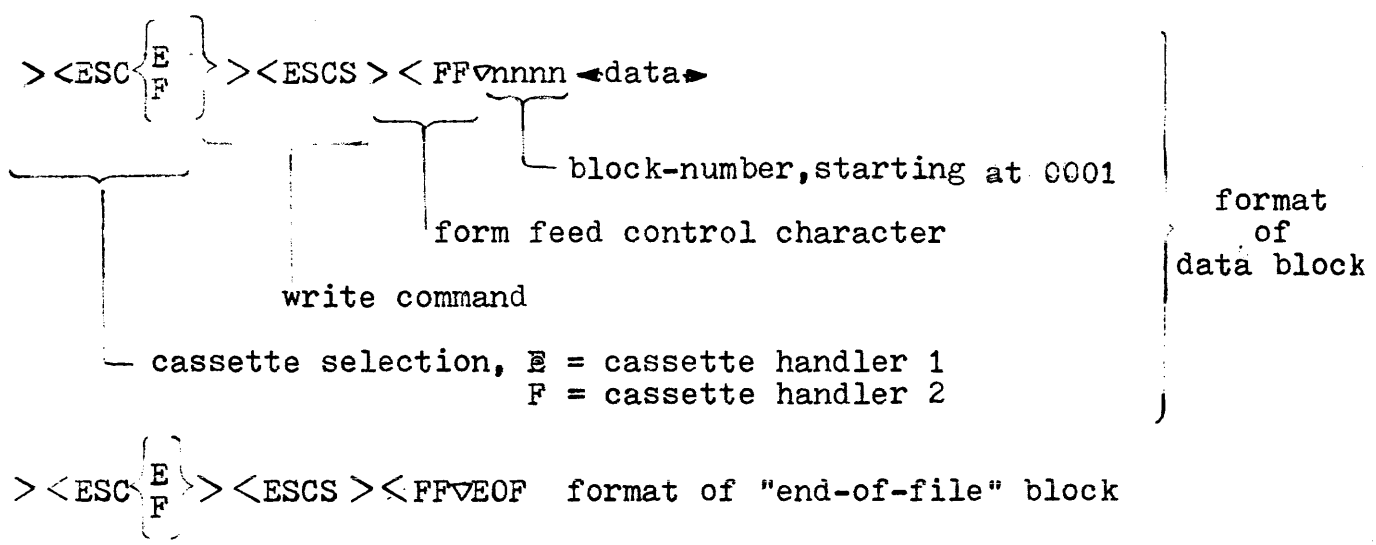
- sends a file of identical data blocks to a selected cassette through the CAS symbolic queue
- sends a file to the printer through the PRT symbolic queue
- receives the file previously transmitted from a cassette through the QAB symbolic queue

For each possible mode of operation, the program displays the option with the applicable answers for the console operator to select. A detailed description for operating the program appears as comments in the listing following.

FORMAT OF BLOCKS

The 3 types of block format used are for
- sending data to a cassette
- receiving data from a cassette
- sending data to the printer for hard copy

Block Format for Transmission to Cassette



Block Format for Transmission from Cassette

nnnn ◄ data ►
└block-number, consecutively numbered
} format of data block

EOF format of "end-of-file" block

Block Format for Transmission to Printer

nnnn ◄ data ►
└block-number, starting from 0001

HANDLING OF DATA

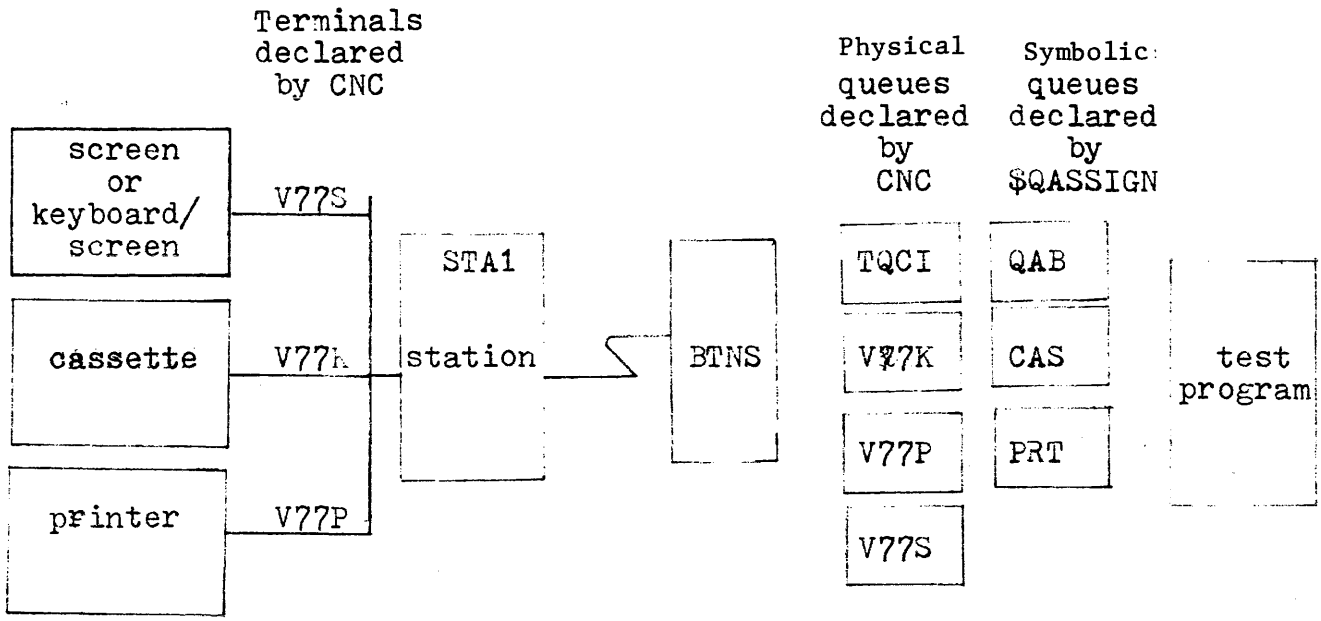
Each data block is transmitted as a separate message. After each operation, the STATUS KEY and the ERROR KEY are checked for abnormal conditions. The program displays any abnormal condition and terminates.

Data sent from or received into the working area is recorded on the SYSOUT file.

Before data is sent to or received from the cassette handler, the program issues a rewind command of the format:

> <ESC{E/F} > <ESCP
└cassette selection, E = cassette handler 1
F = cassette handler 2
└rewind command

NETWORK GENERATION



The transmission mode of the terminals is declared with the following default options, IM=NL and OM=NL, because control characters are not needed on input but must be translated on output as standard mark forms (e.g., , ><ESC).

The screen or keyboard/screen is not addressed by the program since it is declared the master terminal.

In the NDL statements that follow, the points to be noted are,

- VIP7700 is automatically logged on to the program queue TQCI, since it has been declared with the ASSIGN and AUTO parameters.
- Terminal queues are defined on disk because V77K and V77P must be able to contain a series of messages the number and size of which are specified by the L64 console operator in response to a program request.
- The program queue TQCI is declared in memory and is defined to contain 1 message at a time; each time a message is queued, the program receives it and then requests the next message to be read.

NDL STATEMENTS FOR VIP7700 (WITH CASSETTES AND PRINTER)

```
STATN STA1,1;
TERMNL V77S,TYPE=VIP7700,STYPE=KCT;
TERMNL V77K,TYPE=SLAVE,STYPE=CAS;
TERMNL V77P,TYPE=SLAVE,STYPE=PRT;
```

NDL STATEMENTS FOR QUEUES

```
QUEUE V77S;
QUEUE V77K;
QUEUE V77P;

QUEUE TQCI,NUMBLK=30;    MEMORY QUEUE
```

JCL STATEMENTS FOR COMPILING, LINKING AND EXECUTING THE PROGRAM

```
$JOB VIP,USER=JL,PROJECT=SUP4,BILLING=BILL4,CLASS=H;
$COBOL SOURCE=*TSTROVIP,
,OBJLIST,XREF,LEVEL=L64,
CULIB=(CCTL_40.TCULIB,DEVCLASS=MS/M400,MEDIA=C122);
$INPUT TSTROVIP,TYPE=COBOL;
$ENDINPUT;
LIB CU
,INLIB1=(CCTL_40.TCULIB,DEVCLASS=MS/M400,MEDIA=C122)
,INLIB2=(SYS.HCULIB);
LINKER VIPTEST,ENTRY=VIPTEST
,OUTLIB=(MCS.LMLIB,DEVCLASS=MS/M400,MEDIA=C122)
,COMFAC;
STEP VIPTEST,FILE=(MCS.LMLIB,DEVCLASS=MS/M400,MEDIA=C122),DUMP=DATA;
QASSIGN QAB,TQCI,IN;
QASSIGN CAS,V77K,OUT;
QASSIGN PRT,V77P,OUT;
ENDSTEP;
$ENDJOB;
```

MCS COBOL PROGRAM EXAMPLE

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. VIPTEST.
 AUTHOR. JL.
 DATE-WRITTEN. FEB 10 78.
*************************************************************
*    PROGRAM IS A SIMPLE TEST PROGRAM FOR:
*                  VIP7700 WITH CASSETTE AND ROP
*IT ALLOWS ON BEHALF OF THE GCOS64 SYSTEM OPERATOR
* TO PERFORM THE FOLLOWING FUNCTIONS:
*    - READ ANY FILE FROM A DATA CASSETTE PREMOUNTED
*      ON THE VIP CASSETTE DRIVE #1 OR #2
*             - DATA BLOCKS MAY BE OF ANY SIZE UP TO
*               1920 BYTES
*             - THE LAST BLOCK OF THE FILE MUST BE EOF
*               (END OF FILE)
*    - WRITE A FILE ONTO:
*      -A DATA CASSETTE PREMOUNTED ON THE CASSETTE
*         DRIVE #1 OR #2
*     - THE HARD-COPY
*             - DATA BLOCK GENERATED BY THE PROGRAM
*                MAY BE OF ANY LENGTH UP TO 1916 BYEST
*               - END OF FILE BLOCK (EOF) IS GENERATED
*                AUTOMATICALLY BY THE PROGRAM AT THE
*              (NO FOF GENERATED FOR HARD-COYPY FILES)
*                END OF WRITE SEQUENCE
*    - THE TYPE OF TERMINAL IS CONFIRMED THRU THE
*      SYSTEM CONSOLE QUESTION:
*      CONFIRM TERMINAL TYPE (VIP)
*      ANSWER : VIP FOR VIP7700
*    - THE TYPE OF OPERATION IS SELECTED BY THE
*      SYSTEM CONSOLE QUESTION:
*           TYPE OF OPERATION ?
*      ANSWER :
*            R  TO READ A FILE FROM CASSETTE
*            W  TO WRITE A FILE ONTO CASSETTE
*            P  TO WRITE A FILE ONTO HARD-COPY
*            E  TO END THE PROGRAM
*    - THE CASSETTE DRIVE IS SELECTED THRU THE
*      SYSTEM CONSOLE QUESTION :
*      CASSETTE DRIVE # ?
*      ANSWER :
*           1 FOR DEVICE ADDRESSED THRU "ESCE"
*           2 FOR DEVICE ADDRESSED THRU "ESCF"
*    BQASSIGN JCL RUN-TIME COMMANDS MUST DECLARE
*      THE FOLLOWING SYMBOLIC QUEUES :
*            - QAB FOR THE PROGRAM QUEUE
*           - CAS FOR THE CASSETTE QUEUE
*           - PRT FOR THE HARD-COPY QUEUE
*    - AT CNC GENERATION TIME THE CASSETTE AND HARD-
*      COPY MUST BE DECLARED AS WORKING IN:
*             - INPUT NORMAL MODE
*             - OUTPUT NORMAL MODE
*************************************************************
*
```

ENVIRONMENT DIVISION

```
*
**********************************************************
*                    ENVIRONMENT DIVISION
*
**********************************************************
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. LEVEL-64.
 OBJECT-COMPUTER. LEVEL-64.
```

DATA DIVISION

```
*
**********************************************************
*
*                      DATA DIVISION
**********************************************************
 DATA DIVISION.


*   DECLARE WORK VARIABLES


 WORKING-STORAGE SECTION.
 77   TRTYP   PIC XXX.
 77   DRIV    PIC X.
*     COMMAND TO REWIND CASSETTE #1
 77   RWND    PIC X(12) VALUE "><ESCE><ESCP".
 77 INIT1 PIC 9999 VALUE 1.
 77   IDX1    COMP-1.
 77   IDX2    COMP-1.
*             COMMAND TO READ CASSETTE #1
 77   SRDC1   PIC X(12)  VALUE "><ESCE><ESCR".
 01   CONS-MED.
*        COMMAND TO WRITE CASSETTE #1
      02   CMND   PIC X(12) VALUE "><ESCE><ESCS".
*        FORM FEED CHARACTER
     02   FORMF   PIC X(5) VALUE "><FF ".
 01   DUM.
        02   RDY   PIC X.
*       SIZE AND NUMBER OF BLOCKS TO BE WRITTEN
 01   CAR1.
      02   CHAR1   PIC X(8).
      02   CHAR2 REDEFINES CHAR1.
             03   BLKSZ   PIC 9999.
             03   BLKSEP    PIC X.
             03   BLKNB   PIC 999.
*         INPUT WORK AREA TO CONTAIN BLOCKS RECEIVED FROM
*         CASSETTE
 01 INBUF.
     02   INB1   PIC X(2000).
     02   INB3 REDEFINES INB1.
            03   INB31   PIC XXX.
            03   INB32   PIC X(1997).
```

```
*              OUTPUT WORK AREA TO CONTAIN:  —  -
*                 - READ COMMANDS SENT TO CASSETTE #1
*                 - DATA BLOCKS SENT TO CASSETTE #2
 01   OUTBUF.
      02   OUT1    PIC X(17).
      02   OUT2    PIC 9999.
      02   OUT21 REDEFINES OUT2.
           03   OUT22   PIC XXXX.
       02  OUT3    OCCURS 2000.
           03   OUT4    PIC X.
 01   OUT11.
      02   OUT12    PIC 9999.
      02   OUT13  OCCURS 2017.
           03   OUT131    PIC X.


*    DECLARE COMMUNICATIONS STRUCTURES


 COMMUNICATION SECTION.

*    DECLARE CD FOR INPUT

 CD CDIN FOR INPUT
    SYMBOLIC QUEUE IS SYMBQN
    MESSAGE DATE IS DATI
    MESSAGE TIME IS TIMI
    SYMBOLIC SOURCE IS SYMBSOR
    TEXT LENGTH IS TXTLGTH
    END KEY IS ENDKEY
    STATUS KEY IS STATKEY
    MESSAGE COUNT IS MESCNT.

*    DECLARE CD FOR OUTPUT

 CD CDOUT FOR OUTPUT
    DESTINATION COUNT IS DESTCNT
    TEXT LENGTH IS TXTLH
    STATUS KEY IS STATOK
    ERROR KEY IS ERRKEY
    SYMBOLIC DESTINATION IS SYMBDES.
```

PROCEDURE DIVISION

```
      /
      ***************************************************
      *
      *                PROCEDURE DIVISION
      ***************************************************
       PROCEDURE DIVISION.
       STARTX.
            DISPLAY "*** STARTING TEST ***" UPON CONSOLE.
      *    INITIALIZE:
           MOVE 1 TO DESTCNT.
      *    INITIALIZE:
      *          - SYMBOLIC QUEUE FIELD OF CD FOR INPUT
      *    WITH SYMBOLIC NAME DEFINED BY RUN-TIME JCL
      *    (I.E. $QASSIGN CARD)
           MOVE "QAB" TO SYMBQN.
```

```
*****************************************
*     TERMINAL CONFIRMATION
*****************************************

 TRMTYP.

      DISPLAY "CONFIRM TERMINAL TYPE (VIP)" UPON CONSOLE
      ACCEPT TRTYP FROM CONSOLE.
      IF TRTYP = "VIP" GO TO DISRDY.
      PERFORM ERROPT.
      GO TO TRMTYP.

*****************************************
*     OPERATION TYPE SELECTION
*****************************************

*   ASK FROM THE SYSTEM CONSOLE WHICH TYPE OF OPERATION
*   TO PERFROM:
*                - READ A FILE FROM CASSETTE
*                - WRITE A FILE ONTO CASSETTE
*                - WRITE A FILE ONTO HARD-COPY
*                - TERMINATE THE SESSION

 DISRDY.
      DISPLAY "TYPE OF OPERATION ? (R,W,P, OR E)" UPON CONSOLE
      ACCEPT RDY FROM CONSOLE.
      IF RDY = "R" GO TO DRIVE.
      IF RDY = "W" GO TO DRIVE.
      IF RDY = "P" GO TO PRTFIL.
      IF RDY = "E" GO TO FIN.
      PERFORM ERROPT.
      GO TO DISRDY.
*****************************************
*    CASSETTE DRIVE SELECTION
*****************************************

 DRIVE.
*    INITIALIZE:
*          - SYMBOLIC DESTINATION FIELD OF CD FOR OUTPUT
*            WITH SYMBOLIC NAME DEFINED BY RUN-TIME JCL
*            (I.E.$QASSIGN CARD)
      MOVE "CAS" TO SYMBDES.
      DISPLAY "CASSETTE DRIVE # ? (1 OR 2)" UPON CONSOLE.
      ACCEPT DRIV FROM CONSOLE.
      IF DRIV = "1" GO TO DRIV1.
      IF DRIV = "2"  GO TO DRIV2.
      PERFORM ERROPT.
      GO TO DRIVE.
*   DRIVE #2 SELECTED
 DRIV2.
      MOVE "><ESCF><ESCP" TO RWND.
      MOVE "><ESCF><ESCR" TO SRDC1.
      MOVE "><ESCF><ESCS" TO CMND.
      GO TO DRIV3.
*   DRIVE #1 SELECTED
```

```
 DRIV1.
      MOVE "><ESCE><ESCP" TO RWND.
      MOVE "ESCE><ESCR" TO SRDC1.
           MOVE "><ESCE><ESCS" TO CMND.
 DRIV3.
      IF RDY = "W" GO TO WRITE-CAS.
      IF RDY = "R" GO TO READ-CAS.
*****************************************
*    READ A FILE FROM CASSETTE
*****************************************
 READ-CAS.
*     REWIND CASSETTE
      PERFORM REWIND1.
*   LOOP READING THE CASSETTE UNTIL THE END OF FILE
*   BLOCK (EOF) IS FOUND THEN RETURN TO THE OPERATION
*   TYPE SELECTION
 LOOP-READ.
*   INITIALIZE THE OUTPUT WORK AREA WITH TEXT OF
*   CASSETTE READ COMMAND (ESCE ESCR)
      MOVE SPACES TO OUTBUF.
      MOVE SRDC1 TO OUTBUF.
*   INITIALIZE TEXT LENGTH FIELD OF OUTPUT CD
      MOVE 12 TO TXTLH.
*   SEND THE READ COMMAND TO THE CASSETTE THEN CHECK THAT
*   THE SEND HAS BEEN SUCCESSFULL
      PERFORM SENDING-MEDIUM.
      PERFORM RETURN-SENDING.
*   RECEIVE THE READ BLOCK FROM THE PROGRAM QUEUE
*   (QAB) THEN CHECK THAT RECEIVE HAS BEEN SUCCESSFULL
      MOVE SPACES TO INBUF.
      PERFORM RECV.
      PERFORM RETURN-RECV.
*   CHECK IF LAST BLOCK OF THE FILE (EOF)
      IF INB31 = "EOF" GO TO DISRDY.
      GO TO LOOP-READ.
*************************************************
*     WRITE A FILE ONTO CASSETTE
*************************************************

 WRITE-CAS.
*   REWIND CASSETTE
      PERFORM REWIND1.
*   WRITE THE DATA FILE
      PERFORM WRITFIL THRU END-WRITFIL.
*   WRITE END OF FILE BLOCK, CHECK THAT SEND OPERATION
*   HAS BEEN SUCCESSFUL THEN RETURN TO OPERATION TYPE
*   SELECTION
 WR-EOF.
      MOVE CONS-MED TO OUTBUF.
      MOVE "EOF " TO OUT22.
      MOVE 21 TO TXTLH.
      PERFORM SENDING-MEDIUM.
      PERFORM RETURN-SENDING.
      GO TO DISRDY.

 ****************************************
```

```
*      WRITE A FILE ONTO HARD-COPY
*****************************************

 PRTFIL.
*   CHECK OPERATION ALLOWED
*   INITIALIZE:
*      - SYMBOLIC DESTINATION FIELD OF CD FOR OUTPUT
*           WITH SYMBOLIC NAME DEFINED BY RUN-TIME JCL
*           (I.E. $QASSIGN CARD)
      MOVE "PRT" TO SYMBDES.
*   WRITE THE DATA FILE
      PERFORM WRITFIL THRU END-WRITFIL.
      GO TO DISRDY.


************************************************
*            SUBROUTINES
************************************************

************************************************
*      WRITE A DATA FILE TO CASSETTE
*              OR HARD-COPY
************************************************

 WRITFIL.
      MOVE SPACES TO OUTBUF.
      MOVE SPACES TO OUT11.
*   ASK NUMBER AND SIZE OF BLOCKS TO BE WRITTEN
 SIZNB.
      DISPLAY "SIZE/NUMBER ? (XXXX/XXX)" UPON CONSOLE.
      ACCEPT CAR1 FROM CONSOLE.
      IF BLKSEP NOT = "/"
      PERFORM ERROPT
      GO TO SIZNB.
*   CHECK TYPED BLOCK SIZE COMPATABLE WITH TERMINAL TYPE
        IF BLKSZ > 1916
      PERFORM ERROPT
      GO TO SIZNB.
 ENDSIZ.
      MOVE 1 TO IDX1.
*   FILL OUTPUT BUFFER WITH "A" CHARACTERS
 LOOP1.
      IF RDY = "P"
      MOVE "A" TO OUT13 (IDX1)
      GO TO LOOP11.
      MOVE "A" TO OUT3 (IDX1).
 LOOP11.
      ADD 1 TO IDX1.
      IF IDX1 NOT > BLKSZ GO TO LOOP1.
*   INITIALIZE BLOCK COUNTER
      MOVE 1 TO IDX2.
      IF RDY = "P" GO TO PRT1.
      ADD 17 TO BLKSZ.
*   MOVE WRITE COMMAND TO OUTPUT BUFFER
      MOVE CONS-MED TO OUT1.
*   MOVE BLOCK NUMBER TO OUTPUT BUFFER
```

```
     MOVE INIT1 TO OUT2.
     GO TO PRT2.
*  MOVE BLOCK NUMBER TO OUTPUT BUFFER (FOR HARD-COPY)
 PRT1.
     MOVE INIT1 TO OUT12.
 PRT2.
     ADD 4 TO BLKSZ.
 LOOP2.
     IF RDY = "P" MOVE OUT11 TO OUTBUF.
     MOVE BLKSZ TO TXTLH.
*  SEND THE DATA BLOCK TO THE TERMINAL QUEUE THEN CHECK
*  THAT THE SEND OPERATION HAS BEEN SUCCESSFULL
     PERFORM SENDING-MEDIUM.
     PERFORM RETURN-SENDING.
     DISPLAY "OUTPUT= " OUTBUF.
*  INCREMENT BLOCK COUNTERS
     IF RDY = "P"
     ADD 1 TO OUT12
     GO TO INCR1.
     ADD 1 TO OUT2.
 INCR1.
     ADD 1 TO IDX2.
     IF IDX2 NOT > BLKNB GO TO LOOP2.
 END-WRITFIL.
**********************************************
*   SEND A READ COMMAND OR A DATA BLOCK
*       TO CASSETTE QUEUE
**********************************************
 SENDING-MEDIUM.
     SEND CDOUT FROM OUTBUF WITH EMI.
 END-SENDING-MEDIUM.
**********************************************
*  CHECK SEND SUCCESSFUL

 RETURN-SENDING.
     IF STATOK NOT = "00" DISPLAY "OUTPUT STATUS IS " STATOK
     UPON CONSOLE GO TO FIN.
     IF ERRKEY NOT = "0" DISPLAY "OUTPUT ERROR KEY IS "
     ERRKEY UPON CONSOLE GO TO FIN.
 END-RETURN-SENDING.
**********************************************
*  RECEIVE A BLOCK FROM QAB QUEUE

 RECV.
     RECEIVE CDIN MESSAGE INTO INBUF.
     DISPLAY "INPUT = " INBUF.
 END-RECV.
**********************************************
*  CHECK RECEIVE  SUCCESSFUL.

 RETURN-RECV.
     IF STATKEY NOT = "00" DISPLAY "INPUT STATUS IS "
     STATKEY UPON CONSOLE GO TO FIN.
     IF ENDKEY NOT = "3" DISPLAY "ENDKEY IS " ENDKEY
     UPON CONSOLE
     GO TO FIN.
```

```
 END-RETURN-RECV.
*************************************************
*    REWIND CASSETTE OPERATION

 REWIND1.
      MOVE SPACES TO OUTBUF.
      MOVE RWND TO OUTBUF.
      MOVE 12 TO TXTLH.
      PERFORM SENDING-MEDIUM.
      PERFORM RETURN-SENDING.
 END-REWIND1.
*************************************************
*   DISPLAY AN ERROR OPTION MESSAGE

 ERROPT.
      DISPLAY "*** OPTION ERROR ***" UPON CONSOLE.
 END-ERROPT.
 FIN.
      DISPLAY "***END OF TEST***" UPON CONSOLE.
      STOP RUN.
```

APPENDIX E

CNC and QMAINT
COMMANDS & RESERVED SYNTAX

The names reserved for CNC and QMAINT syntax may not appear as user values when either utility is run.

The H_CNC utility uses NDL (network description language) commands but its reserved syntax is annotated as CNC.

Lists of the commands available for H_CNC and H_QMAINT utilities are included as reference to the explanatory text of reserved names that follow.

The 3 categories of reserved names are,

- Command names, introducing the respective commands for the utility
- Keywords which can be further classified as,
  - mandatory
  - optional
- Arguments, appearing as values for keywords.

COMM defines a comment.

```
COMM "string";
```

DCTAP identifies the version of the VCAM subsystem.

```
DCTAP name [,SIMU = {1 | n}];
```

GENCOM describes the environment in which the network is to function.

```
GENCOM name [,BATCHNB = {1 | n}][,BTBFSZ = {144 | nnn}][,DABFSZ = {144 | nnnn}]
 [,KEY = password][,MAMNB = {1 | n}][,NBBTBF = nnn ][ ,NBDABF = nnn ]
 [,QCBLKSZ = {70 | nnnn}][,QCPOOL = {0 | nnnn}][,QDBLKSZ = {70 | nnnn}]
 [,SIMBRK = {"$*$" | "xxx"}][,SYSLOG];
```

LINE defines the characteristics of the communications line

```
LINE LNnn [,BKMOD][,CCINS = nnn][,CLOSE][,CLV = n][,EPX][,ERCAP]
 [,ININS = nnn][,LINPLG][,NAUCDE][,PADNB = nn][,PARITY = n]
 [,POLLIST = (nn[,nn]...)][ ,PRIRA = {IN | OU}/nnnn/{QRRO | QLIN}][,RTCNT = nn]
 [,SPEED=x][,SPN={0 | 1}][,SSR][,TCTNM={ASCII | SPTTY}][,TILEN=nnn][,TOLEN=nnn] ;
```

QUEUE defines all queues, program and terminal queues, used by the network.

```
QUEUE name [,BREAK][,CLOSE][,{CTLRST | RESTART}][,{NUMBLK=nnn | NUMREC=nnnnn | QCPOOL}][,SHARE][,TWA];
```

STATN defines a station attached to a polled line.

```
STATN name,station-index [,CLOSE] ;
```

TERMNL defines the characteristics of the terminal.

```
TERMNL name,TYPE = {terminal-type | SLAVE},STYPE = subtype [,ADD = "hh"]
 [,ASSIGN = name][,AUTO][,BLOCKING][,CLOSE][,CONTROL][,IM = {NL | MK | UN}]
 [,LINELG = nnn][,LLENGTH = nnnn][,NBLOCKS = nnnn][,OM = {NL | UN}]
 [,SBLKG = {80 | nnn}][,SYSHEAD = {"OD25" | "hhhhhhhh"}];
```

COMM defines a comment.

```
COMM "string";
```

PRINT prints the contents of one or more queues in the network.

```
PRINT { name [,name]... } ;
      { *              }
```

PURGE purges all or part of the messages currently in a queue.

```
PURGE name, { ALL          } ;
            { NUMBMSG = nnnnn }
```

QSTATUS lists status and generation parameters of one or more queues.

```
QSTATUS { name [,name]... } ;
        { *              }
```

SEND sends user-defined messages to queues.

```
SEND name [,ENDMSG="end-of-message-string"] [,LENGTH = nnnn ] ;
```

STATUS continues or suspends processing of QMAINT commands on error.

```
STATUS { EVEN  } ;
       { ONLY  }
       { RESET }
```

SYNTAX SUMMARY

CNC Reserved Syntax					
ADD	DSK	LINE	OU	SPEED	TTU8124
ASCII	DTU*	LINELG	PADNB	SPN	TTU8126
ASSIGN	DTU7170	LINPLG	PARITY	SPTTY	TTU8221
AUTO	EPX	LLENGTH	POLLIST	SSR	TTY*
BATCHNB	ERCAP	LN	PRIRA	STATN	TTY33
BKMOD	GENCOM	MAMNB	PRT	STYPE	TTY35
BLOCKING	HL*	MK	QCBLKSZ	SYSHEAD	TTY37
BREAK	HL61	MTS*	QCPOOL	SYSLOG	TTY38
BTBFSZ	HL62	MTS7500	QDBLKSZ	TC*	TWA
BTT*	HL64	NAUCDE	QLIN	TC349	TYPE
BTT7300	HL66	NBBTBF	QRRO	TC380	UN
CAS	IM	NBDABF	QUEUE	TCTNM	VIP*
CCINS	IN	NBLOCKS	RESTART	TCV*	VIP765
CLOSE	ININS	NL	ROF	TCV260	VIP775
CLV	IOF	NUMBLK	RTCNT	TERMNL	VIP785
COMM	KB	NUMREC	SBKLG	TILEN	VIP7100
CONTROL	KCT	OLIV*	SHARE	TN*	VIP7200
CPU	KDS*	OLIV318	SIMBRK	TN300	VIP7700
CRT	KDS7255	OM	SIMU	TN1200	VIP7760
CTLRST	KEY	OPER	SLAVE	TOLEN	VTE*
DABFSZ	KPR			TTU*	VTE2820
DCTAP					

* generic classification of terminals should also be considered reserved.

QMAINT Reserved Syntax		
ALL	LENGTH	QSTATUS
COMM	NUMBMSG	RESET
ENDMSG	ONLY	SEND
EVEN	PRINT	STATUS
	PURGE	

CNC Reserved Syntax		
Reserved Word	Category	Explanation
ADD	optional keyword	parameter of TERMNL command specifying the hardware address of a "compatible" terminal in the form of a 2-hexadecimal digit enclosed within quotation marks.
ASCII	argument	defines the ASCII translation table for the keyword TCTNM optional to the LINE command.
ASSIGN	optional keyword	parameter of TERMNL command dedicating the terminal to a program queue or to the TDS version.
AUTO	optional keyword	parameter of TERMNL command for automatic log-on of a receive-only or master terminal.
BATCHNB	optional keyword	parameter of GENCOM command specifying the maximum number of batch entry processes that can be connected simultaneously to TDS.
BKMOD	optional keyword	parameter of LINE command applicable only for TTY terminals equipped with the BREAK key.
BLOCKING	optional keyword	parameter of TERMNL command for generating a new-line or carriage-return/line-feed before the start or after the end of line, respectively.
BREAK	optional keyword	parameter of QUEUE command applicable only to program queues on which the MCS COBOL application will receive control messages.
BTBFSZ	optional keyword	parameter of GENCOM command specifying the size in bytes of each unit forming the BTNS buffer pool used during I/O transfers.
CAS	argument	defines the terminal subtype as the "cassette handler" for the keyword STYPE mandatory to the TERMNL command.
CCINS	optional keyword	parameter of LINE command specifying the number of DLE characters to be inserted after a control character activating a mechanical function.
CLOSE	optional keyword	• parameter of LINE command specifying that BTNS must not initialize traffic over the line until an "RT LNnn" command is issued. • parameter of QUEUE command specifying that the queue is initially disabled until the first application enables it, cf. ENABLE verb.

CNC Reserved Syntax
(continued)

Reserved Word	Category	Explanation
CLOSE continued...		• parameter of STATN command specifying that BTNS must not poll the station until an "RT xxxx" command is issued, where xxxx is the name of the station. • parameter of TERMNL command specifying that BTNS must not initialize traffic to the terminal until an "RT xxxx" command is issued, where xxxx is the name of the terminal.
CLV	optional keyword	parameter of LINE command specifying the code level to be used in line transmission.
COMM	NDL command name	introduces the COMM command.
CONTROL	optional keyword	parameter of TERMNL command specifying that the log-on procedure for the terminal is subject to verification against the system catalog.
CPU	argument	defines the terminal subtype as the "central processor unit" for the keyword STYPE mandatory to the TERMNL command.
CRT	argument	defines the terminal subtype as the "screen" for the keyword STYPE mandatory to the TERMNL command.
CTLRST	optional keyword	parameter of QUEUE command applicable to terminal disk queues for "roll-back".
DABFSZ	optional keyword	parameter of GENCOM command specifying the size in bytes of each unit forming the VCAM buffer pool.
DCTAP	NDL command name	introduces the DCTAP command.
DSK	argument	defines the terminal subtype as the "disk(ette) drive" for the keyword STYPE mandatory to the TERMNL command.
EPX	optional keyword	parameter of LINE command applicable only for TTY37 type terminals (fully duplex) specifying that echoplex transmission is to be used.
ERCAP	optional keyword	parameter of LINE command applicable only for TTY terminals specifying the "erase" capability by the use of the "back slash" or "reverse slant" character (\).
GENCOM	NDL command name	introduces the GENCOM command.

CNC Reserved Syntax

(continued)

Reserved Word	Category	Explanation
HL61	argument	defines the L64 processor as a BSC terminal for the keyword TYPE mandatory to the TERMNL command.
HL62	argument	defines the L62 processor as a BSC terminal for the keyword TYPE mandatory to the TERMNL command.
HL64	argument	defines the L64 processor as a BSC terminal for the keyword TYPE mandatory to the TERMNL command.
HL66	argument	defines the L66 processor as a BSC terminal for the keyword TYPE mandatory to the TERMNL command.
IM	optional keyword	parameter of TERMNL command specifying the format of input data passed from the terminal to the MCS COBOL application.
IN	argument	defines the number of actions to have input priority on a line for the keyword PRIRA optional to the LINE command.
ININS	optional keyword	parameter of LINE command applicable only for synchronous line procedures specifying the number of SYN characters sent before each message to allow for synchronization.
IOF	keyword	name declared for "Interactive Operation Facility" in DCTAP command, defining the appropriate version of the VCAM subsystem.
KB	argument	defines the terminal subtype as the "keyboard" (input capability) for the keyword STYPE mandatory to the TERMNL command.
KCT	argument	defines the terminal subtype as the "keyboard with screen" for the keyword STYPE mandatory to the TERMNL command.
KEY	optional keyword	parameter of GENCOM command specifying the password used by MCS COBOL applications to enable checking by MAM.
KPR	argument	defines the terminal subtype as the "keyboard with printer" for the keyword STYPE mandatory to the TERMNL command.
LINE	NDL command name	introduces the LINE command.

CNC Reserved Syntax

(continued)

Reserved Word	Category	Explanation
LINELG	optional keyword	parameter of TERMNL command specifying the physical line length in characters as a 2-decimal digit.
LINPLG	optional keyword	parameter of LINE command specifying linear polling for a multipoint line; does not apply to the TTY line procedure.
LLENGTH	optional keyword	parameter of TERMNL command specifying the number of characters in the logical terminal line size.
LN	mandatory keyword	parameter of LINE command qualified by a 2-decimal digit defining the line within the network.
MAMNB	optional keyword	parameter of GENCOM command specifying the maximum number of MCS COBOL application processes that can be concurrently executed.
MK	argument	defines the format of input data passed from the terminal to the MCS COBOL application as "mark mode" for the keyword IM optional to the TERMNL command.
NAUCDE	optional keyword	parameter of LINE command applicable for all terminals if the line is declared "switched" in the SRST inhibiting the ring indicator detection thereby making the line unavailable.
NBBTBF	optional keyword	parameter of GENCOM command specifying the number of units in the buffer pool used by BTNS to service terminals sending to and receiving from queues.
NBDABF	optional keyword	parameter of GENCOM command specifying the number of units in the VCAM buffer pool used by BTNS for sending messages to terminals connected to VCAM subsystems.
NBLOCKS	optional keyword	parameter of TERMNL command specifying the number of logical lines in each message to be sent to the terminal.
NL	argument	• defines the format of input data passed from the terminal to the MCS COBOL application as "normal mode" for the keyword IM optional to the TERMNL command.

CNC Reserved Syntax

(continued)

Reserved Word	Category	Explanation
NL continued...		• defines the format of output data passed from the MCS COBOL application to the terminal as "normal mode" for the keyword OM optional to the TERMNL command.
NUMBLK	optional keyword	parameter of QUEUE command specifying the number of core blocks to be used as the memory queue pool.
NUMREC	optional keyword	parameter of QUEUE command specifying the number of blocks to be used as the disk queue file, thereby setting the ceiling of disk file extent.
OM	optional keyword	parameter of TERMNL command specifying the format of output data passed from the MCS COBOL application to the terminal.
OPER	keyword	reserved name declared for the network control terminal in the TERMNL command having restricted characteristics.
OU	argument	defines the number of actions to have output priority on a line for the keyword PRIRA optional to the LINE command.
PADNB	optional keyword	parameter of LINE command applicable only for TTY terminals specifying the number of PAD characters sent at the end of each output message in order to avoid data loss in the case of turn-around on reception.
PARITY	optional keyword	parameter of LINE command specifying the check on parity for input data.
POLLIST	optional keyword	parameter of LINE command applicable only if the line (with the exception of TTY line procedure) is declared multipoint in the SRST specifying the order in which stations are to be polled.
PRIRA	optional keyword	parameter of LINE command specifying • the number of actions having either input or output priority • the order in which terminals connected to MCS COBOL applications using output queues are scanned.

CNC Reserved Syntax

(continued)

Reserved Word	Category	Explanation
PRT	argument	defines the terminal subtype as the "printer" (output capability) for the keyword STYPE mandatory to the TERMNL command.
QCBLKSZ	optional keyword	parameter of GENCOM command specifying the size in bytes of each core block used to generate the core queue pool.
QCPOOL	optional keyword	• parameter of GENCOM command specifying the number of core blocks of the core queue pool to be shared by all queues qualified by the QCPOOL option in their respective QUEUE commands, see below. • parameter of QUEUE command defining the queue as a memory queue which is to share the core queue pool reserved by the QCPOOL parameter in the GENCOM command.
QDBLKSZ	optional keyword	parameter of GENCOM command specifying the block size in bytes of the disk queue file used for saving messages.
QLIN	argument	defines linear scanning of output queues for the keyword PRIRA optional to the LINE command.
QRRO	argument	defines round-robbin scanning of output queues for the keyword PRIRA optional to the LINE command.
QUEUE	NDL command name	introduces the QUEUE command.
RESTART	optional keyword	parameter of QUEUE command applicable to all disk queues in order to enable "roll-back" for "step abort" or "system crash" conditions.
RTCNT	optional keyword	parameter of LINE command specifying the number of retries to be attempted in case of a transmission error.
SBKLG	optional keyword	parameter of TERMNL command applicable only to the BSC line procedure for specifying the sub-block size in characters to be generated by BTNS when transmitting a character string.

CNC Reserved Syntax
(continued)

Reserved Word	Category	Explanation
SHARE	optional keyword	parameter of QUEUE command applicable only for program queues allowing more than 1 MCS COBOL application to share the same queue in order to establish interjob communication.
SIMBRK	optional keyword	parameter of GENCOM command specifying a character string as a break qualifier in the form of up to 3 alphanumeric characters enclosed within quotation marks.
SIMU	optional keyword	parameter of DCTAP command specifying - the number of simultaneities of the declared TDS generation - the maximum number of terminals that can be simultaneously connected to IOF or ROF VCAM subsystems.
SLAVE	argument	defines the terminal type as a slave to a master terminal previously declared for the keyword TYPE mandatory to the TERMNL command; not applicable to TTY terminals.
SPEED	optional keyword	parameter of LINE command applicable only for terminals in asynchronous transmission defining the line speed as a single alpha-character code.
SPN	optional keyword	parameter of LINE command applicable only for terminals in asynchronous transmission specifying the number of "stop" bits sent after each character.
SPTTY	argument	defines the standard TTY translation table for the keyword TCTNM optional to the LINE command.
SSR	optional keyword	parameter of LINE command specified according to modem type and line speed.
STATN	NDL command name	introduces the STATN command; not applicable to the TTY line procedure.
STYPE	mandatory keyword	parameter of TERMNL command specifying the terminal subtype in the case where the I/O capability of the terminal type is to be altered.
SYSHEAD	optional keyword	parameter of TERMNL command specifying the character string used to precede any system message sent by the network control terminal.
SYSLOG	optional keyword	parameter of GENCOM command specifying that every system message sent by BTNS is to be logged in order to be printed out at the end of CNC session.

CNC Reserved Syntax

(continued)

Reserved Word	Category	Explanation
TCTNM	optional keyword	parameter of LINE command applicable to the TTY and BSC line procedures allowing selection of a translation table other than the standard one defined for the line procedure.
TERMNL	NDL command name	introduces the TERMNL command.
TILEN	optional keyword	parameter of LINE command specifying the "time-out" value used by the URP to control initialization or disconnection of the modem.
TN300	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the TTY line procedure.
TN1200	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the TTY line procedure.
TOLEN	optional keyword	parameter of LINE command specifying the "time-out" value used by the URP, • when polling a buffered terminal • when timing the interval between characters from an unbuffered terminal.
TTU8124	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the TTY line procedure.
TTU8126	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the TTY line procedure.
TTU8221	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the VIP synchronous line procedure.
TTY33	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the TTY line procedure.

CNC Reserved Syntax
(continued)

Reserved Word	Category	Explanation
TTY35	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the TTY line procedure.
TTY37	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the TTY line procedure.
TTY38	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the TTY line procedure.
TWA	optional keyword	parameter of QUEUE command applicable only for program queues to enable dialog between the application and the terminal.
TYPE	mandatory keyword	parameter of TERMNL command defining the type of standard terminal specific to the line procedure.
UN	argument	• defines the format of input data passed from the terminal to the MCS COBOL application as "unedited mode" for the keyword IM optional to the TERMNL command. • defines the format of output data passed from the MCS COBOL application to the terminal as "unedited mode" for the keyword OM optional to the TERMNL command.
VIP765	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the VIP asynchronous line procedure.
VIP775	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the VIP synchronous line procedure.
VIP785	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the VIP synchronous line procedure.
VIP7100	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the TTY line procedure.
VIP7200	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the TTY line procedure.
VIP7700	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the VIP synchronous line procedure.

CNC Reserved Syntax

(continued)

Reserved Word	Category	Explanation
VIP7760	argument	defines the respective terminal for the keyword TYPE mandatory to the TERMNL command; applicable to the VIP synchronous line procedure.

QMAINT Reserved Syntax		
Reserved Word	Category	Explanation
ALL	optional keyword	parameter of PURGE command specifying that all messages in the queue are to be destroyed.
COMM	QMAINT command name	introduces the COMM command.
ENDMSG	optional keyword	parameter of SEND command specifying the "marker" as a string of up to 5 alphanumeric characters used after the message text as a delimiter.
EVEN	optional keyword	parameter of STATUS command specifying that only the following QMAINT command is to be processed when an error occurs.
LENGTH	optional keyword	parameter of SEND command specifying the length of the message text in characters to be sent to the queue.
NUMBMSG	optional keyword	parameter of PURGE command specifying the number of messages in the queue to be destroyed.
ONLY	optional keyword	parameter of STATUS command specifying that the following commands are to be executed if no error has occurred.
PRINT	QMAINT command name	introduces the PRINT command.
PURGE	QMAINT command name	introduces the PURGE command.
QSTATUS	QMAINT command name	introduces the QSTATUS command.
RESET	optional keyword	parameter of STATUS command for resetting the error count to zero.
SEND	QMAINT command name	introduces the SEND command.
STATUS	QMAINT command name	introduces the STATUS command.

APPENDIX F

COMMUNICATIONS OPERATOR COMMANDS

Communications operator commands comprise
- Network control commands which can be entered on the system console
- Terminal operator commands which are shown prefixed with the default break-qualifier $*$.

NETWORK CONTROL COMMANDS

For detailed explanation, see Communications Network Control Terminal Operations Manual.

The list of available commands for the network control operator or the GCOS system operator are,
- BT broadcast telecom
- DT display telecom
- HT hold telecom
- MTE modify terminal edit
- MTL modify telecom line
- MTP modify telecom poll
- MTT modify telecom terminal
- RDY ready
- RT release telecom
- ST start telecom
- TT terminate telecom

TERMINAL OPERATOR COMMANDS

For detailed explanation, see Terminal Operations GCOS Manual.

The break-qualifier is declared at CNC generation by the SIMBRK parameter of the GENCOM command, see Section V.
If the SIMBRK parameter is omitted, then the default break-qualifier is $*$.
In the list following, the default break-qualifier is used to distinguish the commands that are applicable to the terminal operator.

The list of commands for the terminal operator are,
- BRK break
- BT broadcast telecom
- DIS disconnect
- DT display telecom
- HM halt message
- MTE modify terminal edit
- RDY ready
- RM resume message

COMMUNICATIONS OPERATOR COMMANDS

BRK : break
sends a "break" signal for log-on initiation or to an application.

```
$*$BRK
```

BT : broadcast telecom
sends a message to destinations declared by the appropriate operands.

```
$*$BT terminal message-text

BT { program       } message-text
   { program-queue }
   { terminal      }
   { ALL           }
```

DIS : disconnect
disconnects the terminal from the application.

```
$*$DIS [ { HOLD   } ]
       [ { STRONG } ]
```

DT : display telecom
displays the status of the component declared by the appropriate operand.

```
$*$DT { program-queue [STRONG]                  }
      {               [ { QUEUE  } ]            }
      {               [ { STRONG } ]            }

DT [ { line [STRONG]                            } ]
   [ { program [STRONG]                         } ]
   [ { program-queue [STRONG]                   } ]
   [ { QUEUE                                    } ]
   [ { station                                  } ]
   [ { terminal [ { QUEUE  } ]                  } ]
   [ {          [ { STRONG } ]                  } ]
```

HM : halt message
halts delivery of the message to the terminal.

```
$*$HM
```

HT : hold telecom
inhibits the component declared by the appropriate operand.

```
HT { line          }
   { program-queue }
   { station       }
   { terminal      }
```

MTE : modify terminal edit
modifies data format sent from or received by the terminal.

```
$*$MTE [ { BLOCK  } ][ { line-length } { block-size } ][ { IMARK  } ][ { ONEDT  } ]
       [ { NBLOCK } ][ {      *     } { *          } ][ { INEDT  } ][ { ONORM } ]
                                                       [ { INORM  } ]

MTE terminal [ { BLOCK  } ][ { line-length } { block-size } ][ { IMARK  } ][ { ONEDT  } ]
             [ { NBLOCK } ][ {      *     } { *          } ][ { INEDT  } ][ { ONORM } ]
                                                             [ { INORM  } ]
```

[MTL] : modify telecom line

modifies the functionality of a line or makes available maintenance facilities for the line.

```
                  ⎧ input-timeout ⎫ ⎧ output-timeout ⎫ ⎧ read/write-ratio ⎫ ⎧       ⎫
      ⎧ line      ⎨       *       ⎬ ⎨       *       ⎬ ⎨       *       ⎬ ⎨ speed ⎬
MTL   ⎨           ⎩ TRACE         ⎭ ⎩               ⎭ ⎩               ⎭ ⎩       ⎭
      ⎩           ⎩ NTRACE        ⎭
      ⎧ TRACE     ⎫
      ⎩ NTRACE    ⎭
```

[MTP] : modify telecom poll

modifies the polling list for a line from the "start" of the list indicated in the command.

```
MTP line ⎧ 1     ⎫ station-name-1/.../station-name-n
         ⎩ start ⎭
```

[MTT] : modify telecom terminal

modifies the characteristics of a terminal.

```
      ⎧ terminal-queue-1 ⎧ terminal-queue-2 ALTER ⎫ ⎫
      ⎪                  ⎩ NALTER                 ⎭ ⎪
MTT   ⎪                                             ⎪
      ⎪ terminal         ⎧ program                ⎫ ⎪
      ⎪                  ⎨ program-queue          ⎬ ⎪
      ⎩                  ⎩ NASGN                  ⎭ ⎭
```

[RDY] : ready

resumes transmission on a VIP terminal after a page overflow condition has set the "busy" state of the terminal.

$*$RDY [STRONG]

RDY [STRONG]

[RM] : resume message

resumes delivery of the message to the terminal.

$*$RM [ALL]

[RT] : release telecom

releases the component declared by the appropriate operand.

```
      ⎧ line          ⎫
RT    ⎨ program-queue ⎬
      ⎨ station       ⎬
      ⎩ terminal      ⎭
```

[ST] : start telecom

starts a telecommunications session.

ST

[TT] : terminate telecom

causes the telecommunications session to "shut-down".

TT [STRONG]

APPENDIX G

CNC SYSOUT & QMAINT SYSOUT
REPORTS

CNC SYSOUT REPORT

The information in the CNC sysout report comprises,

- Input to H_CNC, i.e., NDL commands generating the network
- Output from H_CNC in terms of:
  - NDL error messages, see Appendix C
  - hardware and firmware parameters related to components constituting the network declared
  - sizes of tables and buffers.

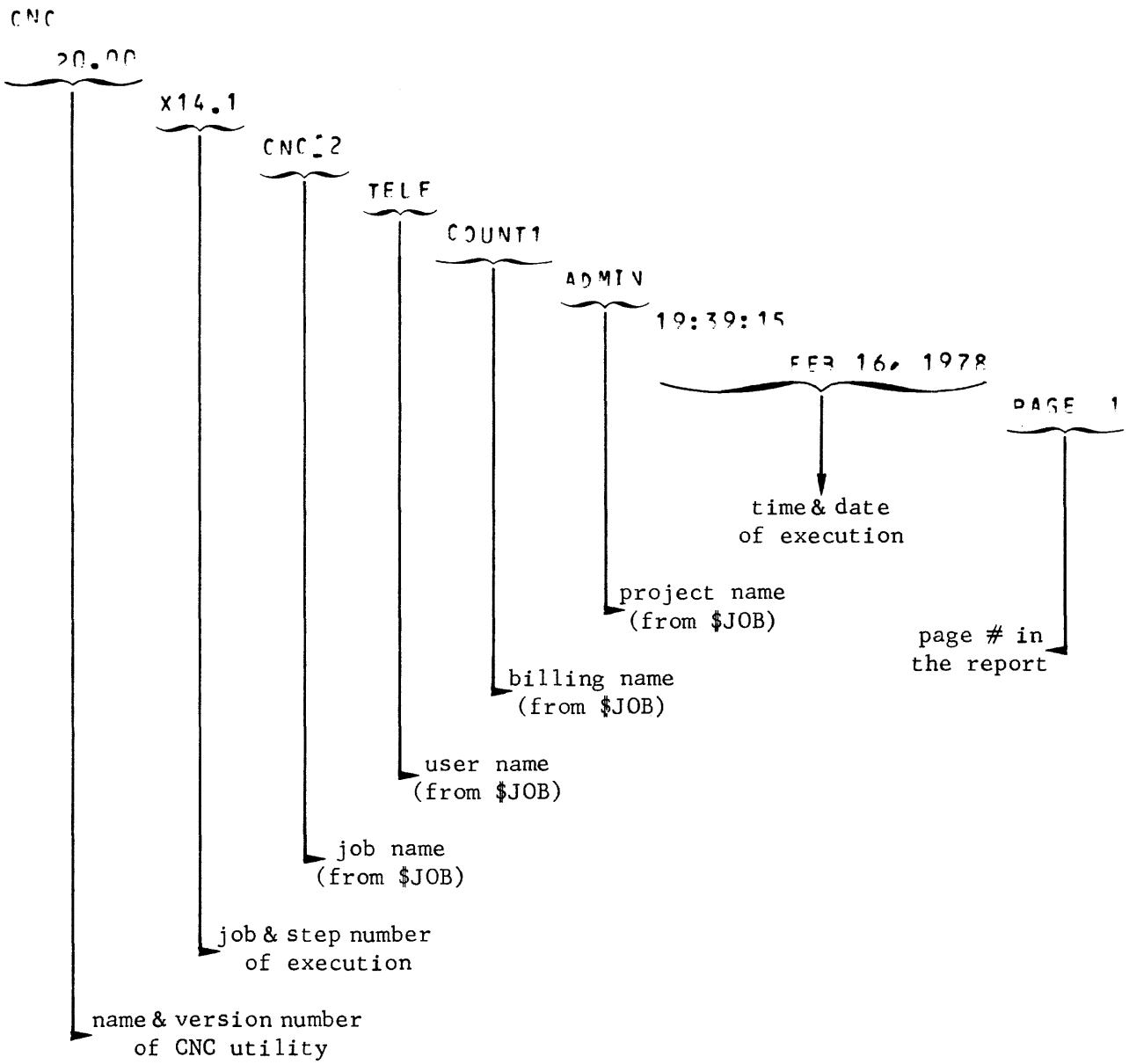
The purpose of the report is to enable the user to optimize the network and to tune its performance.

CNC Sysout Report Structure

header line CNC header banner
header line NETWORK DEFINITION
header line FIRMWARE PARAMETERS
header line SUMMARY

Header Line

The header line appears as the first line of the CNC SYSOUT report and has the standard format defined for any GCOS64 utility.



CNC Header Banner

The CNC header banner follows the standard GCOS64 utility program header format:

```
**********************************************************************
**********************************************************************
*  GCOS L64                                                          *
*                          C N C                                     *
*                                VERSION: 20.00  DATED: SEP 30,1977  *
**EXECUTION-REPORT*****************************************************
**********************************************************************
```

Network Description

The report contains the listing of the NDL commands provided by the user and any NDL error messages, if any, as defined in Appendix C.

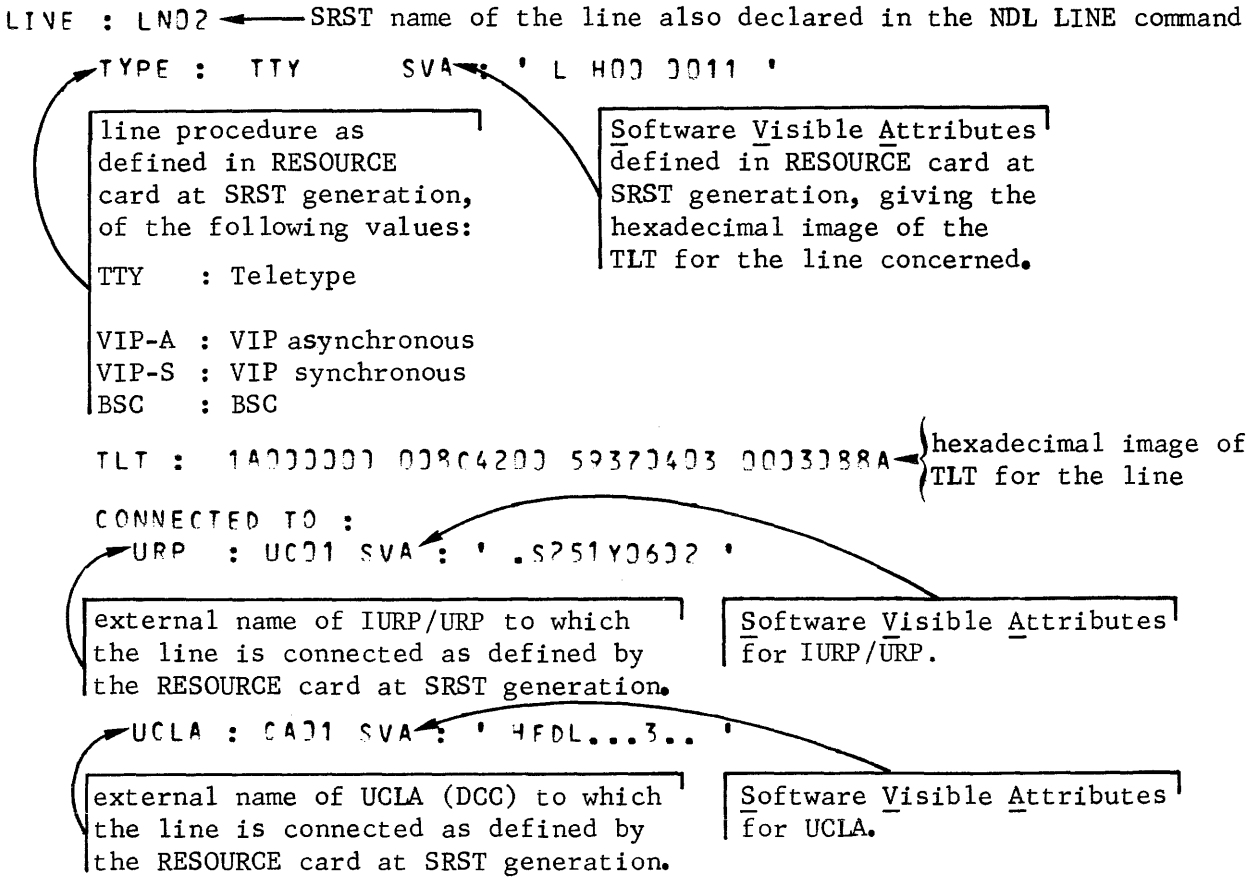
See "Examples" later in the section.

Firmware Parameters

Each line declared by the NDL LINE command is listed with the following information,

- SRST (system resource and status table) parameters defined during the firmware SRST generation thru RESOURCE cards, namely,
  - the line number
  - the UCLA (universal communications line adapter) of the DCC (data communications controller) to which the line is attached
  - the IURP or URP to which the UCLA is attached
- TLT (terminal and line table) as loaded by BTNS when initializing the line in question, containing the following,
  - the hardware parameters as defined during the IURP or URP firmware generation
  - the software parameters in the LINE command which are,
    - ° implicitly defined by their default values
    - ° explicitly specified by the user thru keywords or parameter values

The display format is as follows:



Summary

The display format is as follows:

```
ND_ ANALYSIS ERROR SUMMARY
   TOTAL NUMBER OF MESSAGES =    0   <- accumulated number of messages of severity 1, 2, 3 and 4.
   NBR.OF WARNINGS  SEV.1 =    0
   NBR.OF WARNINGS  SEV.2 =    0    <- individual count for each severity
   NBR OF ERRORS    SEV.3 =    0
   NBR OF ERRORS    SEV.4 =    0
```

SITE: NET ← site-name as defined by the NDL GENCOM command

```
NETWORK SUMMARY
   DECLARED  LINE(S)       =    7   <- total number of lines declared by the respective LINE commands including VIP lines

     LINES WITH SIMU. =    2       <- total number of VIP lines declared

   DECLARED  STATION(S)    =    7   <- total number of stations declared by the respective STATN commands

   DECLARED  TERMINAL(S)   =   15   <- total number of terminals declared by the respective TERMNL commands

     CORE QUEUE(S) =    7           <- total number of declared memory queues, i.e., QUEUE commands specifying NUMBLK or QCPOOL option

     DISK QUEUE(S) =    2           <- total number of disk queues, i.e., QUEUE commands specifying NUMREC or no option

   TOTAL DEC.QUEUE(S)      =    9   <- total number of all queues declared, i.e., memory and disk queues

   DECLARED  TAP(S)        =    5   <- total number of VCAM simultaneities as defined by the algorithm,

                                        ΣTDS + ΣSIMU

                                    where TDS  = number of TDS terminals
                                          SIMU = SIMU values specified in
                                                 DCTAP commands for IOF and
                                                 ROF
```

```
QUEUES FILE IS :  <-- queue file information will only appear if disk queues are defined in NDL deck

        EFN   = M_QUEUE  <-- external name of the queue file as defined thru the $ASSIGN card provided with the CNC JCL statements

        MEDIA = C122     <-- name of the disk medium containing the queue file

        SIZE  =    5 CYL.  <-- size of queue file in number of cylinders
```

```
TABLES AND BUFFERS SIZES
  (DECIMAL VALUES IN BYTES)
   NETWORK TABLES    =    3720  <-- size of the BTNS network table (1)

   BTNS BUFFER POOL  =    8340  <-- size of the BTNS buffer pool

   VCAM BUFFER POOL  =   12820  <-- size of the VCAM buffer pool

   MAM TABLES        =     897  <-- size of MAM tables (1)

   CORE QUEUES POOL  =    7500  <-- size of memory queue pool

   DISK QUEUES POOL  =   15745  <-- size of disk queue pool

   VCAM TABLES       =    1705  <-- size of VCAM tables (1)

   SEMAPHORES POOL   =    1088  <-- size of common semaphore segment (1)
  (NUMBER AND SIZE OF UNITS)
   BTNS B.P. =  40  UNITS OF  208  BYTES
   VCAM B.P. =  50  UNITS OF  256  BYTES   } same information as above
   CORE Q.P. =  50  UNITS OF  150  BYTES   } expressed in number and
   DISK Q.P. =  50  UNITS OF  300  BYTES   } size of buffer units (2)
```

(1) The tables (segments) are grouped as follows,
- those automatically sized by CNC depending on the network generated
- others specified by the user thru the parameters in the GENCOM command, such as BTBFSZ and NBBTBF.

(2) In the case of the disk queue pool, the following algorithm applies as the pool contains other tables related to the queue file, such as the bit allocation map, used by MAM:

size of disk queue pool > (number of units x size of unit)

Examples of Network Definition

EXAMPLE 1

********** NETWORK DEFINITION **********

```
GENCOM NET4,QCPOOL=20;
```

QUEUES DECLARATION

```
COMM " CORE QUEUES IN POOL ";
QUEUE     TT07 QCPOOL;
QUEUE     TT16 QCPOOL;
QUEUE     TT02 QCPOOL;
QUEUE     SWCH QCPOOL;

COMM " CORE QUEUES ";
QUEUE PROG,NUMBLK=20;
```

NETWORK DECLARATION

```
COMM " LOCAL TTY LINE";
LINE LN02 SPEED=H,ERCAP;
TERMNL TT02 TYPE=VIP7100  STYPE=KCT;

COMM " SWITCHED TTY LINE ";
LINE LN07;
TERMNL TT07 TYPE=TN1200   STYPE=KPR;

COMM " SWITCHED TTY LINE ";
LINE LN08 SPEED=H ERCAP;
TERMNL SWCH TYPE=TN300 STYPE=KPR;

COMM "LOCAL TTY LINE ";
LINE LN16 ERCAP;
TERMNL TT16,TYPE= TTU8124 ,STYPE=KPR;
```

The network is made up of 4 TTY lines of which
- 2 are local lines, namely, LN02 and LN16
- 2 are switched lines, namely, LN07 and LN08.

A speed of 300 bps, SPEED=H, has been specified for LN02 and LN08 since this is not the speed defined at firmware generation.

The "erase" capability (\) is used for LN02, LN08 and LN16. As a result, the messages received on these lines will not exceed 134 characters (144 - 10), see ERCAP parameter in LINE command, Section V.

The terminals connected over the lines may only communicate with a MAM application program thru the program queue PROG which is a memory queue of 20 blocks, each block having the default value of 70 bytes.
Responses to the terminals are sent thru the memory queues named TT02, TT07, TT16 and SWCH which share a pool of 20 blocks, each block having the default value of 70 bytes.
The common memory pool is declared by
- the QCPOOL parameter of the GENCOM command
- the QCPOOL parameter of each QUEUE command sharing the pool.

Since no OPER terminal has been declared, the system console is the network control terminal.

The TWU1003 type teleprinter must be declared with the "TTU8124" keyword.

EXAMPLE 2

```
********** NETWORK DEFINITION **********


GENCOM NET3;

                    DCTAP DECLARATIONS

DCTAP IOF SIMU=3;



                    NETWORK DECLARATION

COMM " LOCAL TTY LINE";
LINE LN02 SPEED=H,ERCAP;
TERMNL TT02 TYPE=VIP7100  STYPE=KCT  CONTROL;

COMM " SWITCHED TTY LINE ";
LINE LN07;
TERMNL TT07 TYPE= TN1200   STYPE=KPR  CONTROL;

COMM " SWITCHED TTY LINE ";
LINE LN08 SPEED=H ERCAP;
TERMNL SWCH TYPE=TN300 STYPE=KPR CONTROL;

COMM "LOCAL TTY LINE ";
LINE LN16 ERCAP;
TERMNL TT16,TYPE=TTU8124 ,STYPE=KPR CONTROL;
```

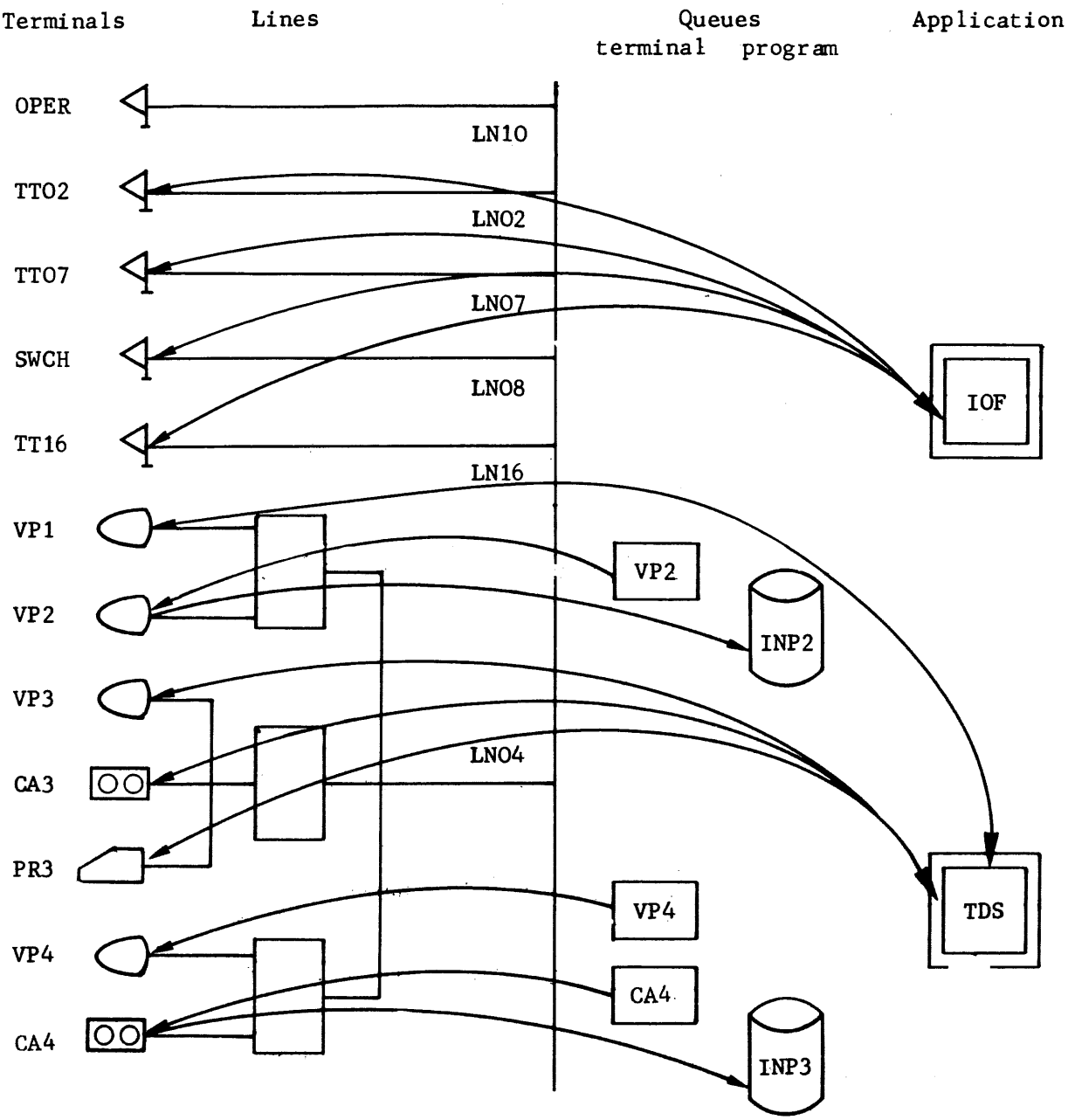
The network declared is exactly the same as that generated in the previous example but the terminals are intended to work only with the VCAM IOF subsystem.

All the terminals are declared with the CONTROL option which is mandatory for connection to IOF.

The IOF subsystem accepts 3 simultaneous connections as defined by the SIMU parameter of the DCTAP command.

EXAMPLE 3

The network following is an extension of that given in Example 2.



Note : Only a sample number of queues is shown in the diagram.

EXAMPLE 3

```
********** NETWORK DEFINITION **********


GENCOM NET BTBFSZ=200 DABFSZ=250
       QCPOOL=30,QCBLKSZ=150 QDBLKSZ=300
       MAMNB=2 BATCHNB=3
       NBBTBF=40 NPDABF=50 SYSLOG;


                QUEUES DECLARATION

COMM " DISK QUEUES ";
QUEUE    INP2 RESTART;
QUEUE    INP3;

COMM " CORE QUEUES IN POOL ";
QUEUE    TT07 QCPOOL;
QUEUE    TT16 QCPOOL;
QUEUE    TT02 QCPOOL;
QUEUE    SWCH QCPOOL;
QUEUE    VP2  QCPOOL;
QUEUE    VP4  QCPOOL;

COMM " CORE QUEUES ";
QUEUE    DK4 NUMBLK=20;


                DCTAP DECLARATIONS

DCTAP TDS;
DCTAP IOF SIMU=3;
DCTAP ROF SIMU=1;
```

```
COMM
"          NETWORK DECLARATION                    ";

COMM " NETWORK CONTROL OPERATOR LINE ";
LINE LN10;
STATN S101,1;
TERMNL OPER,TYPE=VIP765,STYPE=KCT  CONTROL;

COMM " LOCAL TTY LINE";
LINE LN02 SPEED=H,ERCAP;
TERMNL TT02 TYPE= VIP7100  STYPE=KCT  CONTROL;

COMM " SWITCHED TTY LINE ";
LINE LN07;
TERMNL TT07 TYPE= TN1200   STYPE=KPR  CONTROL;

COMM " SWITCHED TTY LINE ";
LINE LN08 SPEED=H ERCAP;
TERMNL SWCH TYPE=TN300 STYPE=KPR  CONTROL;

COMM "LOCAL TTY LINE ";
LINE LN15 ERCAP;
TERMNL TT16,TYPE= TTU8124 ,STYPE=KPR;

COMM " POLLED VIP LINE (SYNCHRONOUS) ";
LINE LN04 RTCNT=4  TOLEN=4 LINPLS;
STATN S041,1;
TERMNL VP1,TYPE=VIP775,STYPE=KCT,ASSIGN=TDS,CONTROL;
TERMNL VP2,TYPE=VIP775,STYPE=KCT,ASSIGN=INP2,CONTROL;
STATN S042,2;
TERMNL VP3,TYPE=VIP7700,STYPE=KCT,ASSIGN=TDS,AUTO;
TERMNL PR3,TYPE=SLAVE,STYPE=CAS;
TERMNL CA3,TYPE=SLAVE,STYPE=PRT;
STATN S043,3;
TERMNL VP4 TYPE=VIP7700 STYPE=KCT ASSIGN= INP3 AUTO;
TERMNL DK4 TYPE=SLAVE STYPE=CAS;
```

EXAMPLE 4 ********** NETWORK DEFINITION **********

```
GENCOM NET BTBFSZ=200 DABFSZ=250
          QCPOOL=30,QCBLKSZ=150 QDBLKSZ=300
          NBBTBF=40 NPDABF=50 SYSLOG;

        COMM
        "          QUEUES DECLARATION

COMM " DISK QUEUES ";
QUEUE    INP2 RESTART;
QUEUE    H64  CTLRST;


COMM " CORE QUEUES IN POOL ";
QUEUE    TT07 QCPOOL;
QUEUE    TT16 QCPOOL;
QUEUE    TT02 QCPOOL;
QUEUE    SWCH QCPOOL;


        COMM
        "          DCTAP DECLARATIONS

DCTAP IOF SIMU=3;
DCTAP ROF SIMU=1;


      COMM
      "          NETWORK DECLARATION              ";

COMM " LOCAL TTY LINE";
LINE LN02 SPEED=H,ERCAP;
TERMNL TT02 TYPE= VIP7100  STYPE=KCT  CONTROL;

COMM " SWITCHED TTY LINE ";
LINE LN07;
TERMNL TT07 TYPE= TN1200   STYPE=KPR  CONTROL;

COMM " SWITCHED TTY LINE ";
LINE LN08 SPEED=H ERCAP;
TERMNL SWCH TYPE=TN300 STYPE=KPR  CONTROL;

COMM "LOCAL TTY LINE ";
LINE LN16 ERCAP;
TERMNL TT16,TYPE= TTU8124 ,STYPE=KPR  CONTROL;
```

```
COMM " LEVEL 64 BSC LINE ";
LINE LN05 SSR;
STATN S051,1;
TERMN H64,TYPE=HL64,STYPE=CPU,ASSIGN=INP2,AUTO;
```

The network is an extension of that given in Example 2.
The extension consists of a file transmission line, namely,

- LNO5 used for a L64 to L64 file transmission under an MCS COBOL program control thru the disk queues named INP2 and H64.

In the case of the MCS COBOL program, the "controlled restart" facility is used, i.e., the CTLRST parameter is specified in QUEUE H64 command.

QMAINT SYSOUT REPORT

The information given in the following example of a QMAINT sysout report is,

- the JCL statements used when executing an H_QMAINT session
- the report of the execution comprising,
  - the effects of commands performed on the queues
  - the error summary of the H_QMAINT session.

H_QMAINT executes actions on the queues as follows,

- Q1, Q5 and Q6, which are disk queues
- Q2 and Q3, which are memory queues.

A detailed explanation of the report for each type of QMAINT command performed is given in Section VI.

The description of header lines and banner is the same as for those appearing in the CNC sysout report.

JCL Statements

```
$JOB TST←QMAINT
,USER=UNAME,PROJECT=ADMIN,BILLING=ADMINN;
STEP H←QMAINT
,SYS.HLMLIB;
ASSIGN H←CR,*QMAINT;
ENDSTEP;
$INPUT QMAINT,TYPE=DATASSF;
SEND Q1,ENDMSG="//FIN",LENGTH=160;
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA111
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA111
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA111
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA111
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA111
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA111
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA11
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA11
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA11
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA111
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA111
//FIN
SEND Q1,LENGTH=160;
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB2
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB2
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB2
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB2
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
//EOM
QSTATUS *;
PRINT *;
PRINT Q1,Q2,Q3;
PRINT Q4;
PRINT Q5;
PRINT Q6;
PURGE Q1,NUMBMSG=1;
PURGE Q1,NUMBMSG=1;
QSTATUS *;
PURGE Q2,ALL;
QSTATUS Q1,Q2;
PRINT *;
PURGE Q1,NUMBMSG=1;
PURGE Q3,ALL;
PURGE Q4,NUMBMSG=3;
PRINT Q3;
$ENDINPUT;
$ENDJOB;
```

Execution Report

```
SEND Q1,ENDMSG="//FIN",LENGTH=160;

        QUEUE NAME :                                    Q1
        MESSAGE STATUS :                                OK
        MESSAGE LENGTH :                                000160
        MESSAGE CONTENTS :
        AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        AAAAAAAAAAAAAAAAAAAAA111AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA111

SEND Q1,LENGTH=160;

        QUEUE NAME :                                    Q1
        MESSAGE STATUS :                                OK
        MESSAGE LENGTH :                                000160
        MESSAGE CONTENTS :
        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
        BBBBBBBBBBBBBBBBBBBBBBBB2BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB2

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 :               000010
        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 :
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAA111AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA111
MESSAGE NUMBER :                                    000002
MESSAGE STATUS :                                        OK
MESSAGE LENGTH :                                    000160
MESSAGE CONTENTS :
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBB2BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB2
```

```
        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,Q2,Q3;

        QUEUE NAME :                                    Q1
        NUMBER OF COMPLETE MESSAGES :                   000002
        MESSAGE NUMBER :                                000001
        MESSAGE STATUS :                                  OK
        MESSAGE LENGTH :                                000160
        MESSAGE CONTENTS :
        AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        AAAAAAAAAAAAAAAAAAAAA111AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA111
        MESSAGE NUMBER :                                000002
        MESSAGE STATUS :                                  OK
        MESSAGE LENGTH :                                000160
        MESSAGE CONTENTS :
        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
        BBBBBBBBBBBBBBBBBBBBBBB2BBBBBBBBBBBBBBBBBBBBBBBBBBBBB
        BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB2

        QUEUE NAME :                                    Q2
        NUMBER OF COMPLETE MESSAGES :                   000000

        QUEUE NAME :                                    Q3
        NUMBER OF COMPLETE MESSAGES :                   000000

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 *;

          QUEUE NAME :                              Q1
          NUMBER OF COMPLETE MESSAGES :             000000

          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

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
```

```
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
```

APPENDIX H

COMMUNICATIONS STATUS KEY CONDITIONS

The CD communications description statements define the parameters required to interface with MCS, namely,

- between MCS and the application,
  - ACCEPT :which makes available the current number of messages in a specified symbolic queue.
  - RECEIVE:which receives a message from a specified symbolic queue.
  - SEND :which sends a message to a specified symbolic output queue.
- between MCS and the terminals,
  - DISABLE:which closes communications between MCS and the specified queue.
  - ENABLE :which opens communications between MCS and the specified queue.

The status of this MCS COBOL 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, i.e., the "x" denotes the parameter has been specified.

The parameters listed in the following table in alphabetical order are,

- ACCEPT
- DISABLE INPUT (with TERMINAL)
- DISABLE INPUT (without TERMINAL)
- DISABLE OUTPUT
- ENABLE INPUT (with TERMINAL)
- ENABLE INPUT (without TERMINAL)
- ENABLE OUTPUT
- RECEIVE
- SEND.

Status key codes from 9A thru 9E are only applicable if the appropriate program queue has been defined with the "BREAK" option.

Communications Status Key Conditions

Status Key Code	ACCEPT (with COUNT)	DISABLE INPUT (with TERMINAL)	DISABLE INPUT (without TERMINAL)	DISABLE OUTPUT	ENABLE INPUT (with TERMINAL)	ENABLE INPUT (without TERMINAL)	ENABLE OUTPUT	RECEIVE	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 enabling/disabling 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 DISABLE statement 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.
96								x		message data returned but at least 1 previous message lost.
97									x	identifier-2 (see SEND statement) ≠ "0", "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.

* unknown means symbolic queue not defined in JCL.

Communications Status Key Conditions

(continued)

Status Key Code	ACCEPT (with COUNT)	DISABLE INPUT (with TERMINAL)	DISABLE INPUT (without TERMINAL)	DISABLE OUTPUT	ENABLE INPUT (with TERMINAL)	ENABLE INPUT (without TERMINAL)	ENABLE OUTPUT	RECEIVE	SEND	
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 the CTLRST option.

APPENDIX I

BTNS
JOB OCCURRENCE REPORT

During a communications session, BTNS gathers statistics to enable the user
- to tune the network, see Section V
- to control the traffic over each line and terminal.

The BTNS JOR listing consists of
- system information
- system messages
- statistics.

SYSTEM INFORMATION

The following print-out is self-explanatory.

```
JOBID=BTNS   USER=BTNS   PROJECT=OPERATOR   BILLING=INSTALL   RUN=X0003
-------------------------------------------------------------------

22:30:20                  JOB EXECUTION LISTING                FEB 16, 1978

-------------------------------------------------------------------
           STEP 1
           LOAD MODULE = H_BTNS (78/2 /15 15:00)
           LIBRARY = SYS.HLMLIB
           SHARED MODULE = H_SM1
           LIBRARY = SYS.HSMLIB
           SHARED MODULE = H_SM2
           LIBRARY = SYS.HSMLIB
22:30:31   STEP STARTED XPRTY=0
```

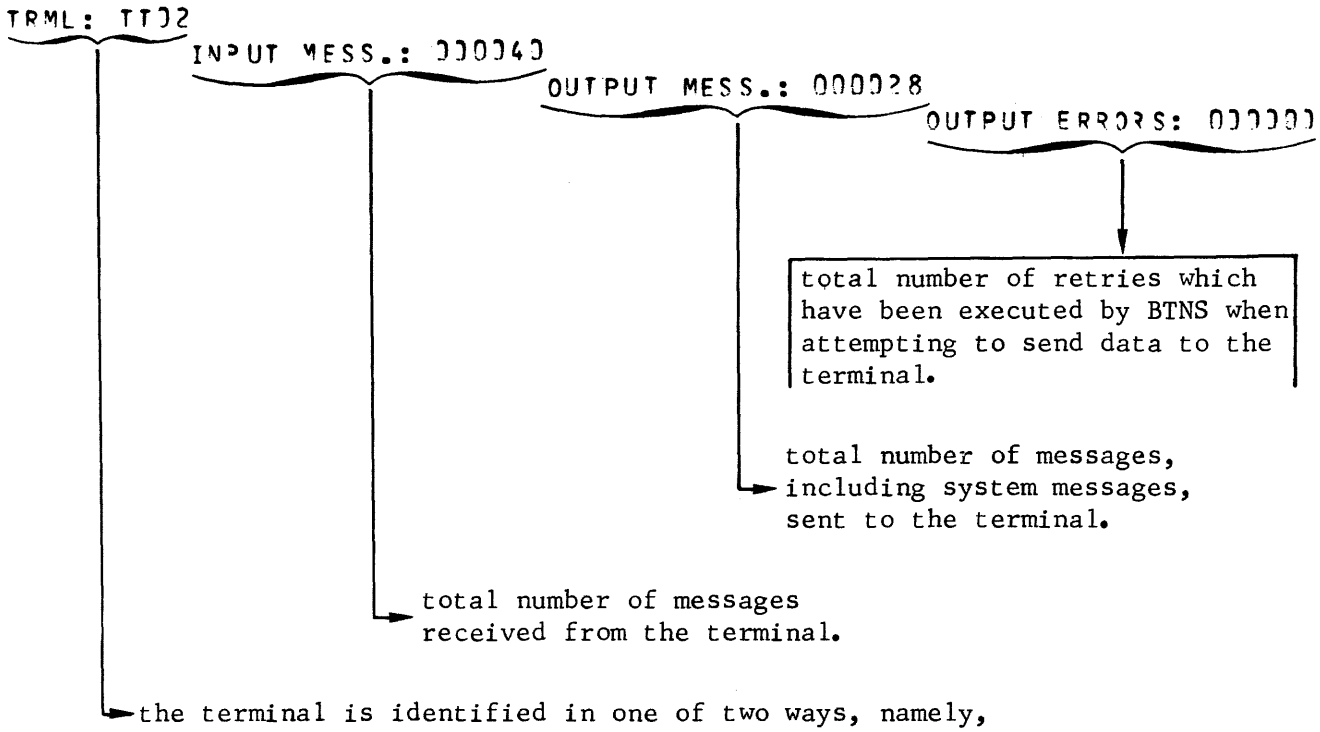
SYSTEM MESSAGES

The following 2 pages are a sample of system messages and commands exchanged during the session.
The listing will appear if the SYSLOG parameter of the NDL GENCOM command has been specified.
For details of system messages and commands, refer to
- Communications Network Control Terminal Operations Manual
- Terminal Operations GCOS Manual.

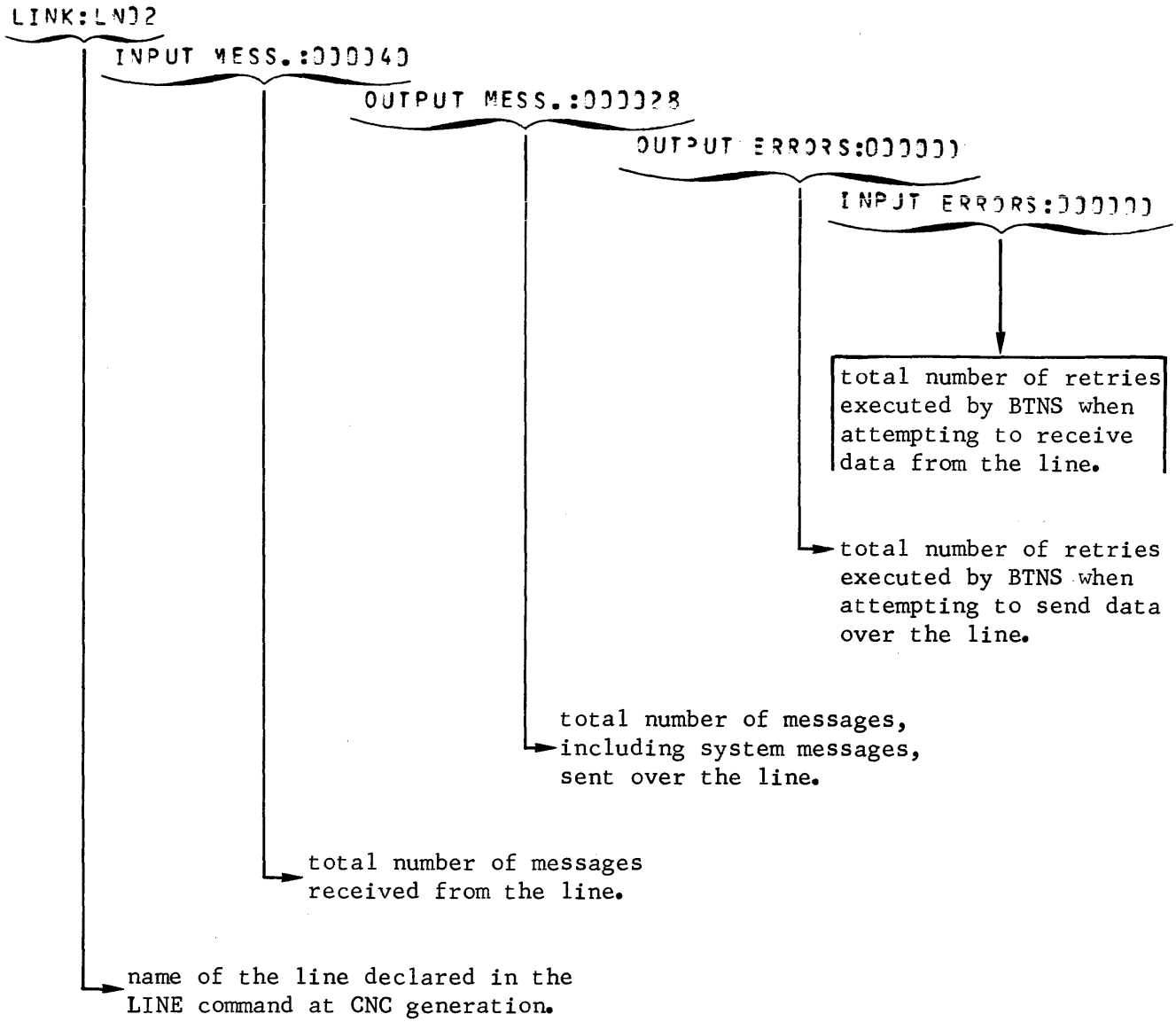
```
    CC21 LN16   CANNOT BE OPENED REASON: ENABLE CP FAILED
   22.31 NET4   STARTED
*   CC01         USER/PROJECT/BILLING,APPL?
-->22.31 TT02   TT/T02/T02,TT02
*   CC02         PASSWORD?@;
-->22.31 TT02   LL
   22.31 TT02   ACTIVATED FOR TT02 ON FEB 16, 1
   22.31 LN08   ACTIVATED
*   CC01         ID,USER/PROJECT/BILLING,APPL?
-->22.32 @@@@   RT,IOF
*   CC02         PASSWORD?@;
-->22.32 RT     HH
   22.32 RT     ACTIVATED   ON FEB 16, 1
   22.32 RT     DEACTIVATED BY USER
   22.32 LN08   DEACTIVATED
   22.32 LN08   ACTIVATED
*   CC01         ID,USER/PROJECT/BILLING,APPL?
-->22.32 @@@@   RT,RT01,IOF
*   CC02         PASSWORD?@;
-->22.32 RT     FF
    CC33 RT     LOGGED
   22.32 RT     ACTIVATED FOR IOF ON FEB 16, 1
-->22.33 RT     $*$DT STRONG
    CC08 RT     ACTIVE FOR IOF /TEMP US= RT01
    CC09         NBLOCK 100/118 IN=N OU=N
    CC32         OM=15 IM=13 OE=0
    CC10 OQ     0 MSG(S) 100% OF FREE SPACE
-->22.33 TT02   $*$DT STRONG
    CC08 TT02   ACTIVE FOR TT02 /TEMP US= TT
    CC09         NBLOCK 100/118 IN=N OU=N
    CC32         OM=8 IM=10 OE=0
    CC10 OQ     0 MSG(S) 100% OF FREE SPACE
-->22.33 OPER   HELLO!
    CC03 BT     COMPLETED
-->22.33 TT02   $*$BT MAIN HELLO
-->22.33 TT02   HELLO
    CC03 BT     COMPLETED
-->22.34 RT     $*$BT MAIN PLEASE MOUNT DISK C122
-->22.34 RT     PLEASE MOUNT DISK C122
    CC03 BT     COMPLETED
-->22.34 RT     $*$BT MAIN IS C122 MEDIA MOUNTED?
-->22.34 RT     IS C122 MEDIA MOUNTED?
    CC03 BT     COMPLETED
-->22.35 MAIN   WAIT FOR 5 MINUTES PLEASE
    CC03 BT     COMPLETED
-->22.35 TT02   $*$DT PROG QUEUE
    CC10 PROG   0 MSG(S) 100% OF FREE SPACE
-->22.36 TT02   $*$DT TT08 QUEUE
    CC03         INVALID NAME
```

```
-->22.36 TT02  $*$DT TT16 QUEUE
    CC10 TT16  0 MSG(S) 100% OF FREE SPACE
    CC10 TT07  0 MSG(S) 100% OF FREE SPACE
    CC10 TT16  0 MSG(S) 100% OF FREE SPACE
    CC10 TT02  0 MSG(S) 100% OF FREE SPACE
    CC10 SWCH  0 MSG(S) 100% OF FREE SPACE
    CC10 PROG  0 MSG(S) 100% OF FREE SPACE
    CC03 BT    COMPLETED
    CC10 TT07  0 MSG(S) 100% OF FREE SPACE
    CC10 TT16  0 MSG(S) 100% OF FREE SPACE
    CC10 TT02  0 MSG(S) 100% OF FREE SPACE
    CC10 SWCH  0 MSG(S) 100% OF FREE SPACE
    CC10 PROG  1 MSG(S) 90% OF FREE SPACE
-->22.37 MAIN  C122 PREMOUNTED
    CC03 BT    COMPLETED
-->22.38 RT    $*$DT IOF
    CC07 IOF   CNCT= 0 WAITING = 0
-->22.38 OPER  YOU ARE CONNECTED TO IOF
    CC03 BT    COMPLETED
   22.39 TT02  DEACTIVATED BY USER
*   CC01       USER/PROJECT/BILLING,APPL?
-->22.39 TT02  TT/TT?/TT2,PROG
*   CC02       PASSWORD??;
-->22.39 TT02  FF
   22.39 TT02  ACTIVATED FOR PROG ON FEB 16, 1
-->22.40 TT02  $*$DT PROG
    CC10 PROG  3 MSG(S) 70% OF FREE SPACE
-->22.40 TT02  $*$DT STRONG
    CC08 TT02  ACTIVE FOR PROG /TEMP US= TT
    CC09       NBLOCK 100/118 IN=N OU=N
    CC32       OM=24 IM=35 OE=0
    CC10 00    0 MSG(S) 100% OF FREE SPACE
-->22.41 TT02  $*$BT MAIN PLEASE START PROG(PROG QUEUE IS FULL)
-->22.41 TT02  PLEASE START PROG(PROG QUEUE IS FULL)
    CC03 BT    COMPLETED
-->22.42 OPER  *END OF SESSION IN 5 MINUTES
    CC03 BT    COMPLETED
   22.42 TT02  DEACTIVATED BY USER
   22.44 RT    DEACTIVATED BY APPL
   22.44 LN08  DEACTIVATED
    CC08 NET4
    CC08 LN02
    CC15       TT02/IDLE
    CC08 LN07  IDLE
    CC08 LN08  IDLE
    CC08 LN16  IDLE
    CC15       TT16/IDLE
    CC03 TT    COMPLETED
    CC11 LN02  HELD BY SYSTEM,REASON: SHUTDOWN
```

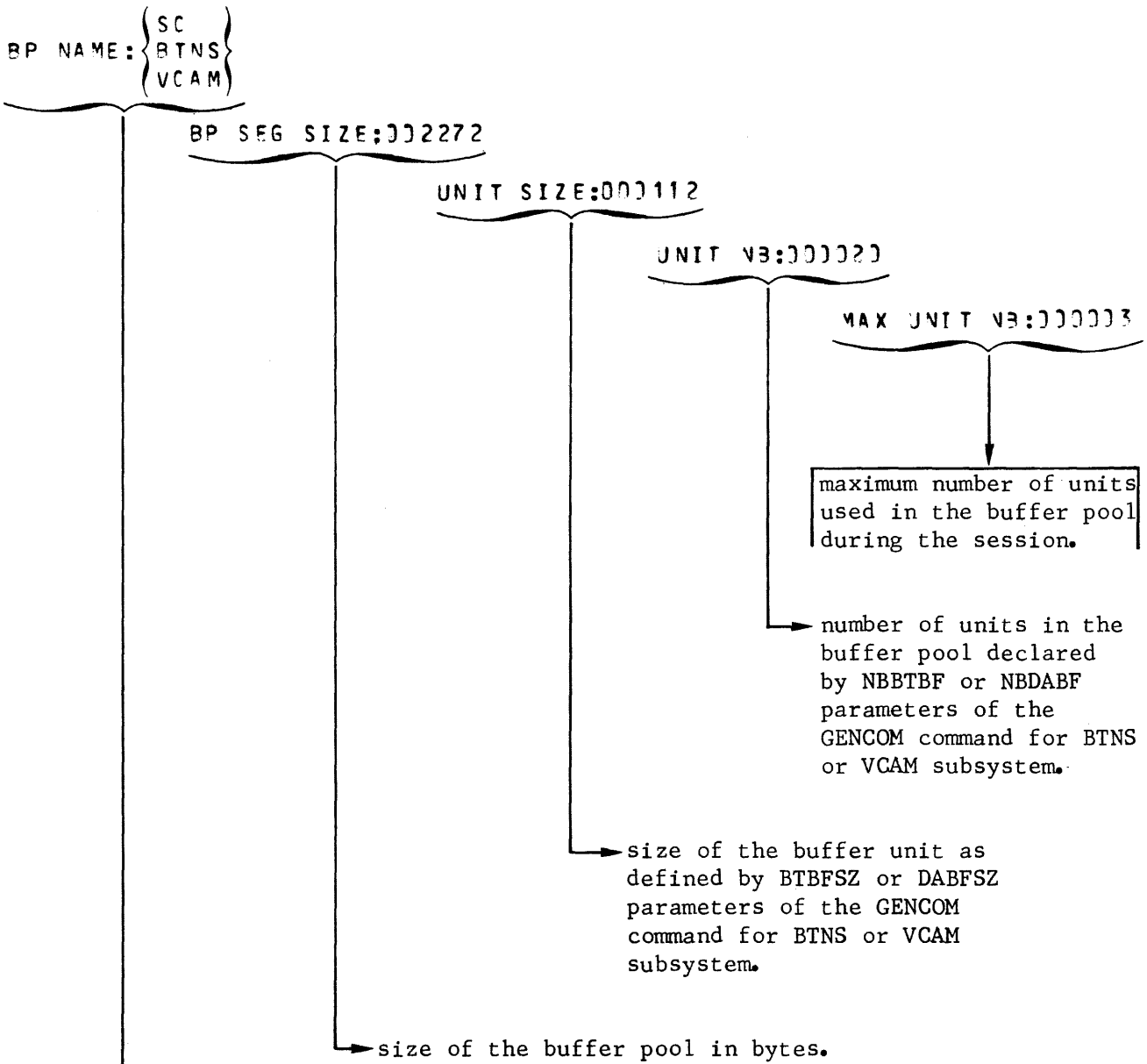
```
**********************************************************************
**********************STATISTICS**********************************
**********************************************************************
```



The above information appears for each terminal declared at network generation.

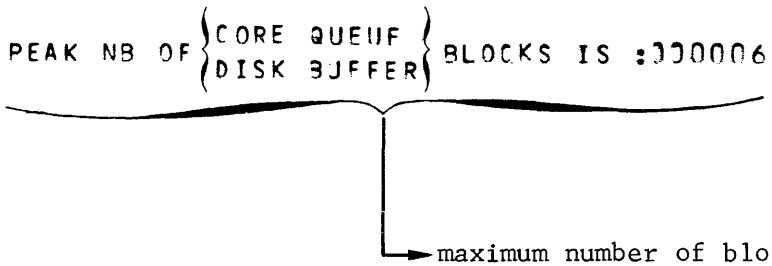


The above information appears for each line declared at network generation.



3 lines of information are displayed, one line for each buffer pool,

- SC : site controller buffer pool internal to BTNS and is automatically sized by BTNS for containing messages sent and received by the site controller.
- BTNS : information relates to BTNS buffer pool.
- VCAM : information is displayed if the communications session involves a VCAM subsystem.



maximum number of blocks used during the session in:

- the core queue segment
- the disk buffer pool used to access the disk queue file.
  The disk buffer pool is automatically sized by CNC according to the number of lines and MCS COBOL application programs to be executed simultaneously, see MAMNB parameter of the GENCOM command.

The information above applies specifically to MAM.

```
**************************************************************************
*************************END OF STATISTICS*************************
**************************************************************************
```

The following information is common to all JOR listings.

```
            22.44 NET4   TERMINATED
          TASK MAIN J=01 P=00 COMPLETED
          CPU       0.086
          ELAPSED  14.102                STACKOV              341
          LINES          0 LIMIT NOLIM MISSING SEGMENTS      33
          CARDS          0 LIMIT NOLIM BACKING STORE     195360
22:44:37  STEP COMPLETED                BUFFER SIZE         5840 CPSIZE 2272


          START     22:30:20    LINES          0
          STOP      22:44:37    CARDS          0
          CPU          0.086
          ELAPSE      14.291
22:44:37  RESULT:   JOB 00      R02
```

HONEYWELL INFORMATION SYSTEMS

Technical Publications Remarks Form

TITLE: SERIES 60 (LEVEL 64) COMMUNICATIONS PROCESSING FACILITY

ORDER NO. AQ53, REV. 1

DATED SEPTEMBER 1978

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by appropriate technical personnel and action will be taken as required. If you require a written reply, check here and furnish complete mailing address below. ☐

FROM: NAME _____ **DATE** _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

PLEASE FOLD AND TAPE –
NOTE: U. S. Postal Service will not deliver stapled forms

FIRST CLASS
PERMIT NO. 39531
WALTHAM, MA
02154

Business Reply Mail
Postage Stamp Not Necessary if Mailed in the United States

Postage Will Be Paid By:

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTENTION: PUBLICATIONS, MS 486

Honeywell

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE: SERIES 60 (LEVEL 64) COMMUNICATIONS PROCESSING FACILITY ADDENDUM A

ORDER NO. AQ53-01A

DATED JUNE 1979

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by appropriate technical personnel and action will be taken as required. If you require a written reply, check here ☐ and furnish complete mailing address below.

FROM: NAME _____ **DATE** _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

PLEASE FOLD AND TAPE—
NOTE: U. S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

Honeywell