REALITY®

by Microdata

Bisync

Operator's Guide

LET'S COMMUNICATE

Microdata is interested in your experiences with our products in use. Let us know about . . .

New applications
New techniques
Modifications
Changes
Corrections
Anything appropriate

Users of Microdata hardware, software and computer-related documents are regularly developing appropriate changes or new applications for our products. We like to know about them. Often, we can help you develop or improve a new idea. Or, you can help us with a change. It is our policy to regularly communicate appropriate new ideas and applications to the users of our products. Let us be helpful. Let's communicate.

Attn: Marketing Technical Support
MICRODATA CORPORATION
17481 Red Hill Avenue, Irvine, CA 92714
Post Office Box 19501, Irvine, CA 92713
Phone: 714/540-6730
TWX: 910-595-1764

REALITY®
(3.0 SERIES)

Bisync
Operator's Guide

771043

Price: $10.00

Microdata Corporation
17481 Red Hill Avenue, Irvine, California 92714
Post Office Box 19501, Irvine, California 92713
Telephone: 714/540-6730 · TWX: 910-595-1764

CONTENTS

1.1 INTRODUCTION

> The major function of Binary Synchronous Communication (Bisync) is to effect the orderly transfer of large amounts of data from one location to another using communications facilities.

Data is transferred as binary-coded characters comprising text information and is blocked for transmission. In addition, data-link control characters are required with each transmission to delimit various portions of the text information and to control the communications line.

Reality Bisync is an emulation of the IBM 2780 Data Transmission Terminal. The Process can be used to transmit Reality file items at communications line speed to another Reality or any other machine following the IBM 2780 Bisync Protocol.

Reality Bisync incorporates many of the optional features of an IBM 2780 as standard features. Some of the special features are listed in Figure A.

The first section of this manual is a general discussion of data communications. The remaining sections describe the operation of Reality Bisync.

- Normal EBCDIC Transmission
- Transparent EBCDIC Transmission
- Transparent ASCII Transmission
- Multiple Record Blocks
- Extended Retry Feature
- Auto Answer

Figure A. Standard Features of Reality Bisync

1.2 TRANSMISSION METHODS

Two classifications of data communications used by the computer industry are asynchronous and synchronous. Binary Synchronous Communication incorporates synchronous data communication.

Asynchronous Transmission

In asynchronous transmission, synchronizing bits (one 'start' and one or two 'stop' bits) are sent with each byte that is transmitted. The start bit signals that a character is coming and enables the receiving station to become synchronized with the transmitting station at the beginning of each character. Since sync bits accompany each character transmitted, asynchronous transmission can occur at irregular intervals. Thus, asynchronous transmission is well suited to low volume transmission.

The advantage of asynchronous transmission is the simplicity of the modems, making the cost of transmission minimal. However, since each character must have associated start and stop bits, overhead is high, making the system inefficient for high volume transmission.

Synchronous Transmission

With this method of transmission, characters are blocked and sent at a definite frequency. This eliminates start and stop bits as signals of when characters are being sent. Synchronizing characters are required only at the beginning of each block of data. The sync characters establish the required time base for placing the receiver and transmitter in step. Once the receiver and transmitter are in step, the entire block may be transmitted. Normally, since large blocks of data are being transmitted, various types of block checking routines are used to validate the data.

Usually associated with synchronous transmission is the use of special line control characters to signal beginning and ending of text, and identify any special data control characters that are being used.

Binary Synchronous Transmission

Binary Synchronous Communications (Bisync) is a specific communications protocol used in IBM communications systems. Bisync has become an industry standard protocol for data transmission.

Bisync is built around sending blocks of data and the use of a specified set of control characters to identify beginning and ending of data, control the line, respond to data, and perform error checking. If data is received correctly, the receiving station responds with a positive acknowledgment using an 'ACK' character. A 'NAK' character is used as a negative acknowledgment.

This page intentionally blank

1.3 TRANSMISSION CODES

The major function of Bisync is to effect the orderly transfer of data from one location to another using communications facilities. Bisync procedures can accommodate three transmission codes. These codes may be expanded when using the transparent mode.

All data in Bisync is transferred as binary-coded characters comprising text information. In addition, data-link control characters are required to delimit various portions of the data and control its transmission. The data and control characters make up a transmission message, with the data being the body of the message.

Bisync can accommodate three specific transmission code sets. Each set consists of data characters, function characters (e.g., tabs and form control), and line control characters. The transmission code sets are listed in Figure A.

In the normal mode of transmission, the control characters and function characters may not be transmitted as data characters within the body of the message. However, if using the "transparent" mode, these characters may be transmitted as data. Transparency is described below. The transmission codes supported in Reality Bisync are EBCDIC, transparent EBCDIC, and a type of transparent ASCII the ASCII facilitates transmission between Reality systems - data transmitted is in ASCII and line control characters are in EBCDIC.

Transparency

Transparency is actually part of a line protocol. When used with ASCII, EBCDIC or Six-Bit Trans Code, transparency permits greater versatility in the range of coded data that can be transmitted. This is because all data, including the normally restricted data-link control characters, are treated only as specific 'bit patterns.' All data link control characters can be transmitted as transparent data without taking on control meaning.

Any data-link control characters transmitted in the transparent mode are preceded by a DLE character. This identifies them as a control function. A DLE data character is identified as data by a preceding DLE (DLE DLE). A detailed discussion of data-link control characters and sequences is given in Sections 2.13 through 2.16.

EBCDIC	256 Assignment Positions
ASCII	128 Assignment Positions
Six-Bit Transcode	64 Assignment Positions
Transparency	Used with the above codes, gives added flexibility.

Figure A. Transmission Codes

1.4 TRANSMISSION LINKS

The data transmission link consists of equipment provided by a common carrier, such as AT&T. The three transmission modes are simplex, half duplex, and full duplex. The link may be over a private, leased line, or the public switched network. Modems are required at each end of the link, and are discussed in Section 3.

Various types of transmission links are available. The basic equipment required for the link is the telephone line between the stations and the modems for each station. Modems are discussed in Section 3 of this manual. Some physical considerations on the telephone line are discussed in the following paragraphs.

Simplex

In the simplex mode, data is transmitted in only one direction. Therefore, the transmitter is always the transmitter, the receiver always the receiver. Simplex is a two-wire physical configuration. Figure A illustrates a simplex link.

Half Duplex

Half duplex may have two physical configurations, two-wire or four-wire. Half duplex two-wire has one wire for send and one for receive. Half duplex four-wire has two wires for send and two for receive. In either configuration, transmission in both directions is not simultaneous. The line must be 'turned around' between send and receive sequences. Some of the modems available offer a lower line turnaround time with the four-wire configuration, or require the four-wire configuration for use with private line service. Figure B illustrates half duplex.

Reality Bisync, operating under 2780 emulation, uses the half duplex mode.

Full Duplex

Full duplex may also have a two-wire or four-wire physical configuration. In full duplex, transmission may be simultaneous in both directions. In the two-wire configuration, each wire is set up to send or receive. In the four-wire configuration, two wires are set up to send and two to receive. Figure C illustrates full duplex. Reality Bisync cannot be full duplex.

Switched Network

The switched (or dial-up) link is over the regular telephone network. With this type of data link, a Data Access Arrangement (DAA) is required from the local telephone company. The DAA is a special dial-up telephone that interfaces with the modem. When initiating communications, the user dials up the other station, the call goes over the regular switched network, the call is answered at the other station, and communications may begin.

Private Line

With a private line setup, the telephone line is dedicated for use by the communications link and does not go through the switched telephone network. A dial-up DAA set is not required for a private line, although alternate voice capability is an option on most modems. Usually, the Request-To-Send signal is continuous. Options are available on some modems to also have the data-set-ready signal continuous. When initiating communications, no dial-up is required. As soon as the transmitting station receives a clear-to-send from the receiving station, transmission may begin.

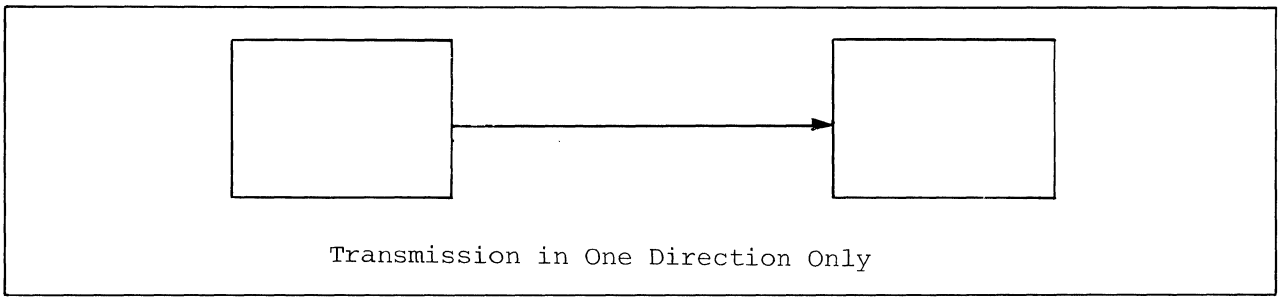


Figure A. Simplex

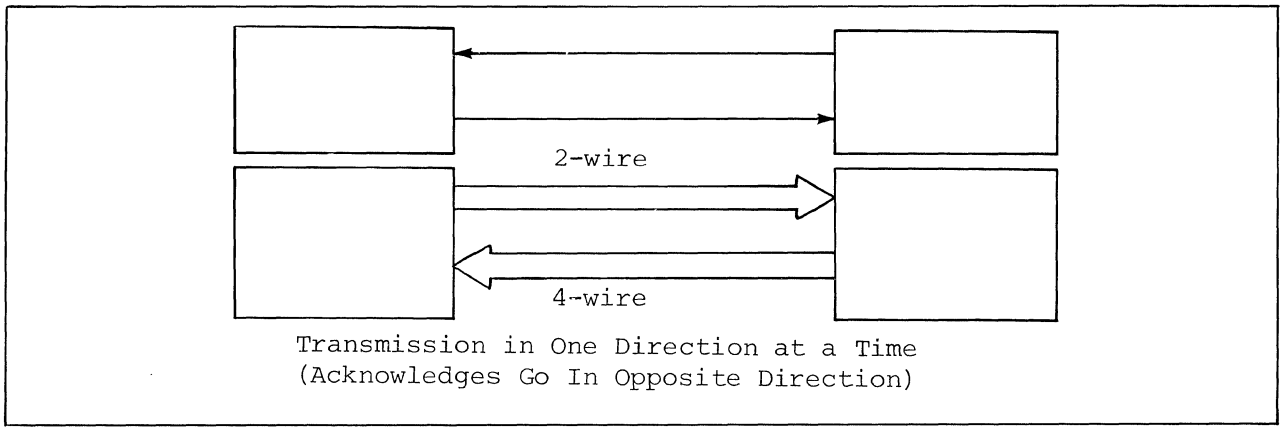


Figure B. Half Duplex

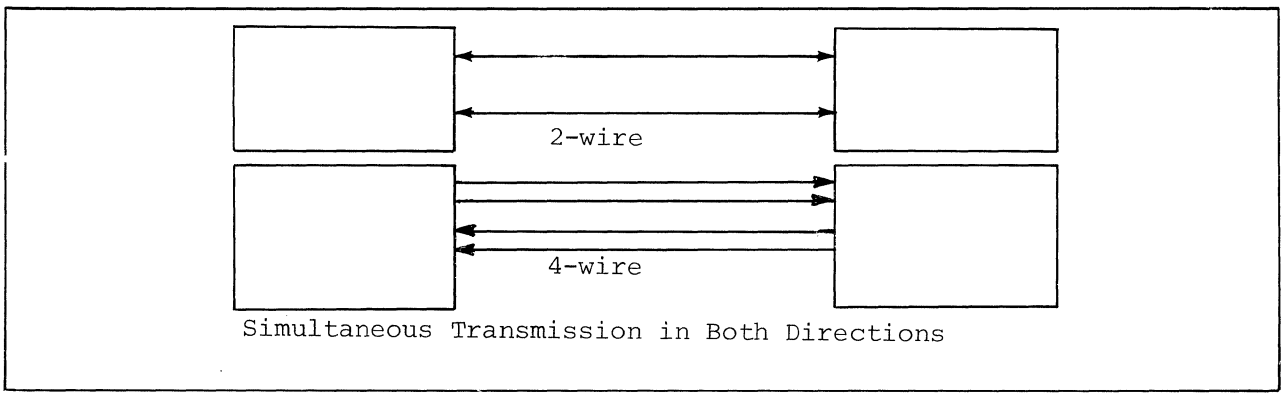


Figure C. Full Duplex

1.5 DATA LINKS

The two basic types of data links are point-to-point and multipoint.

Point-to-Point

A point-to-point data link involves two stations. When a station desires to use the communications line, the station bids for the line by using a specified sequence of line control characters.

In point-to-point operation a contention situation exists, whereby both stations may attempt to use the communications line simultaneously. If simultaneous bidding occurs, one station must persist in its bidding attempt to break the contention condition. Once a station gains control of the line, transmission may begin.

To minimize the possibility of a contention situation, one station on the line is generally considered the primary station and the other the secondary station. The distinction between a primary and a secondary station is normally made by specifying different receive timeout periods for each. For example, the secondary might be configured to transmit a bid for the line every three seconds, while the primary is configured to bid every one second. In effect, this gives the primary station precedence.

A point-to-point data link can be on a switched (dial-up) network or on a private line setup. Figure A shows a point-to-point setup.

Multipoint

A multipoint data link involves two or more stations. In this environment, one station is always the primary station and the others are secondary stations. The primary station either polls or selects the secondary stations. Polling is an "invitation to send" transmitted from the primary station to a specific secondary station. Selection is a "request to receive" notification from the primary station to one of the secondary stations, instructing it to receive data. This allows the primary station to control the data link.

Each station in the data link is assigned a unique address which is used to acquire the station's attention during either a polling or selection sequence. Station addresses consist of from one to seven characters.

A multipoint data link requires a dedicated private line for the transmission link. Figure B shows a sample layout of a multipoint setup.

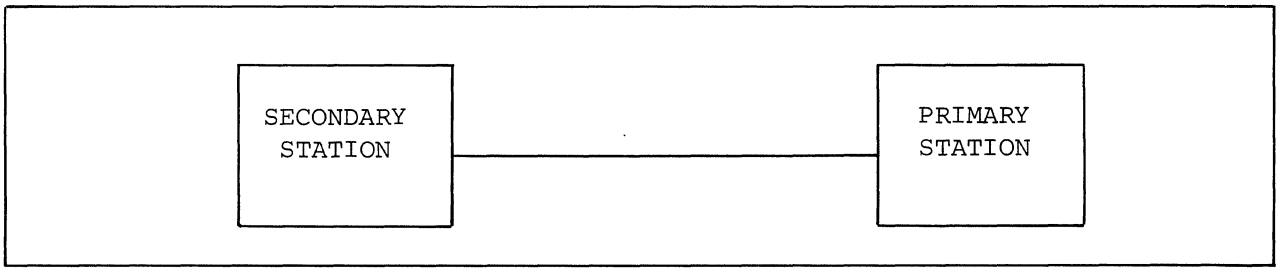


Figure A. Point-to-Point Data Link

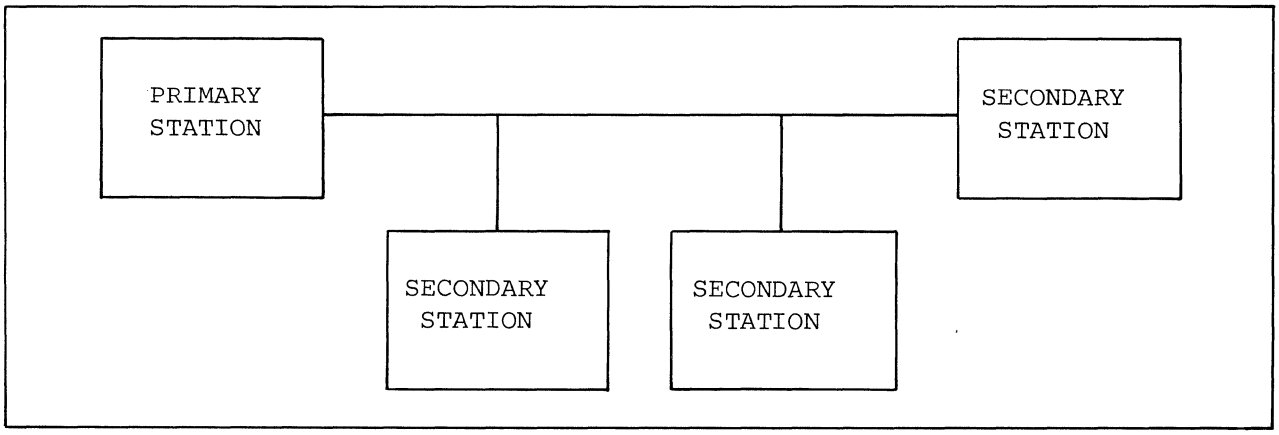


Figure B. Multipoint Data Link

1.6 COMMUNICATION PROTOCOLS

The 2770, 2780, and 3780 Communication Protocols are derived from the data I/O terminals of the same designation as marketed by IBM.

2780 Protocol

The 2780 Protocol has been in existence since 1967. It was the first IBM terminal to use bisynchronous communication. The 2780 Data Terminal is primarily a card reader/punch, and also functions as a printer station. The cards are 80 columns wide, leading to the standard 80 character record size. Bisync allows data to be transmitted to such a station at a high data rate with a low probability of errors. This is because of the blocking of card images and print lines.

The main limitations of 2780 Protocol are the fixed block length, the lack of a conversation mode, and the limited input/output device selection.

2770 Protocol

The 2770 Protocol was developed after 2780. It allows 96-column records and was designed to work with a variety of data station devices. A 2780 option is conversational mode. This means that the terminal can accept a response to an inquiry without first being selected. This allows it to run acceptably fast in an inquiry mode.

3780 Protocol

The 3780 Protocol will run faster (7200 bps versus 4800 bps for the 2780). Variable length records are standard on the IBM 3780, and it allows conversational mode. The biggest drawback is the lack of programmability.

Figure A is a comparison of some of the standard and optional features of 2780, 2770, 3780, and Reality Bisync.

FEATURE	2770		2780		3780		REALITY	
	STAND-ARD	OP-TION	STAND-ARD	OP-TION	STAND-ARD	OP-TION	STAND-ARD	OP-TION
EBCDIC	X		X		X		X	
TRANSPARENCY		X		X		X	X	
120 CHARACTER INPUT RECORD*	X			X	X		X	
140 CHARACTER INPUT RECORD*	X			X		X	X	
MULTIPLE RECORDS PER BLOCK		X		X	X		X	
VARIABLE LENGTH RECORDS	X			X	X		X	
AUTO ANSWER		X		X		X	X	
AUTO LINE TURNAROUND	X			X	X		X	
PRINTER HORIZONTAL FORMAT CONTROL*	X			X	X		X	
CONVERSATIONAL MODE		X				X		
1200 - 4800 BAUD	X		X		X		X	
1200 - 7200 BAUD					X		X	
1200 - 9600 BAUD							X	

*STANDARD FEATURE ON 2770 BASED ON CONFIGURATION ORDERED.

Figure A. Comparison of Some Features

2.1 OVERVIEW OF REALITY BISYNC

The Reality Binary Synchronous Communications Process (Bisync) is a unique portion of the system software that is designed to support data communications in the Reality multi-user virtual memory operating system.

Reality Bisync is an emulation of the IBM 2780 Data Communications Terminal. It may be used to transmit (or receive) data to another Reality or any other machine that supports IBM 2780 communications protocol.

2602 Controller

The 2602 Bisync Controller board is the only additional hardware required on a Reality system to interface with the data link equipment.

Process Assignment

The Bisync Process may be assigned to any line (terminal) in the system and controlled from any or all of the remaining terminals. When Bisync is assigned to a line, the terminal attached to that line is dedicated to the process and may not be used for any other purpose until Bisync has been deactivated. The terminal attached to the line becomes the Bisync console.

When Bisync is activated, the current status of the Process is displayed on the terminal. Status messages include such things as telephone line connection, line failures, blocks received and transmitted, transmission errors, etc.

Bisync Verbs

Since Bisync is a unique software process, there is a complete set of verbs supplied to control the Process's modes of operation and communicate with it. These verbs are summarized in Figure A, and are discussed in detail in the following sections of this chapter.

Transmission Data

All data in Bisync is transmitted over the data link as binary-coded characters which comprise a text message (see Section 2.2 for a detailed discussion of messages). A message may consist of one or more items, or an entire file. Reality file items or files are structured into transmission messages via the TRANSMIT verb (see Section 2.3), and may be readied for transmission before or after the Bisync Process is activated.

Bisync Queue

Messages formed by the TRANSMIT verb are put in the Bisync queue. The Bisync queue is similar to the Reality Spooler queue. It is a temporary storage area to hold the structured messages, either received messages or messages waiting to be transmitted, and is not saved by the FILE-SAVE process. The messages

have a four-digit message number in the range 0000 to 9999. When Bisync is activated, the messages in the queue are transmitted in ascending order of message numbers.

ABORT-BSC	Terminates transmission or reception of a message
DISPLAY-MESSAGE	Displays specified message in character format
DISPLAY-MSG-QUE	Displays Bisync message queue
DUMP-MESSAGE	Displays specified message in hex format
FILE-MESSAGE	Files messages into Reality file items
KILL-MESSAGE	Deletes specified message from Bisync queue
RESTART-BSC	Restarts Bisync after abnormal condition (attended mode)
SPOOL-MESSAGE	Outputs message via Reality spooler
START-BSC	Activates the Bisync process
STOP-BSC	Stops the Bisync process
TRANSMIT	Formats Reality file items into transmission messages

Figure A. Bisync Verbs

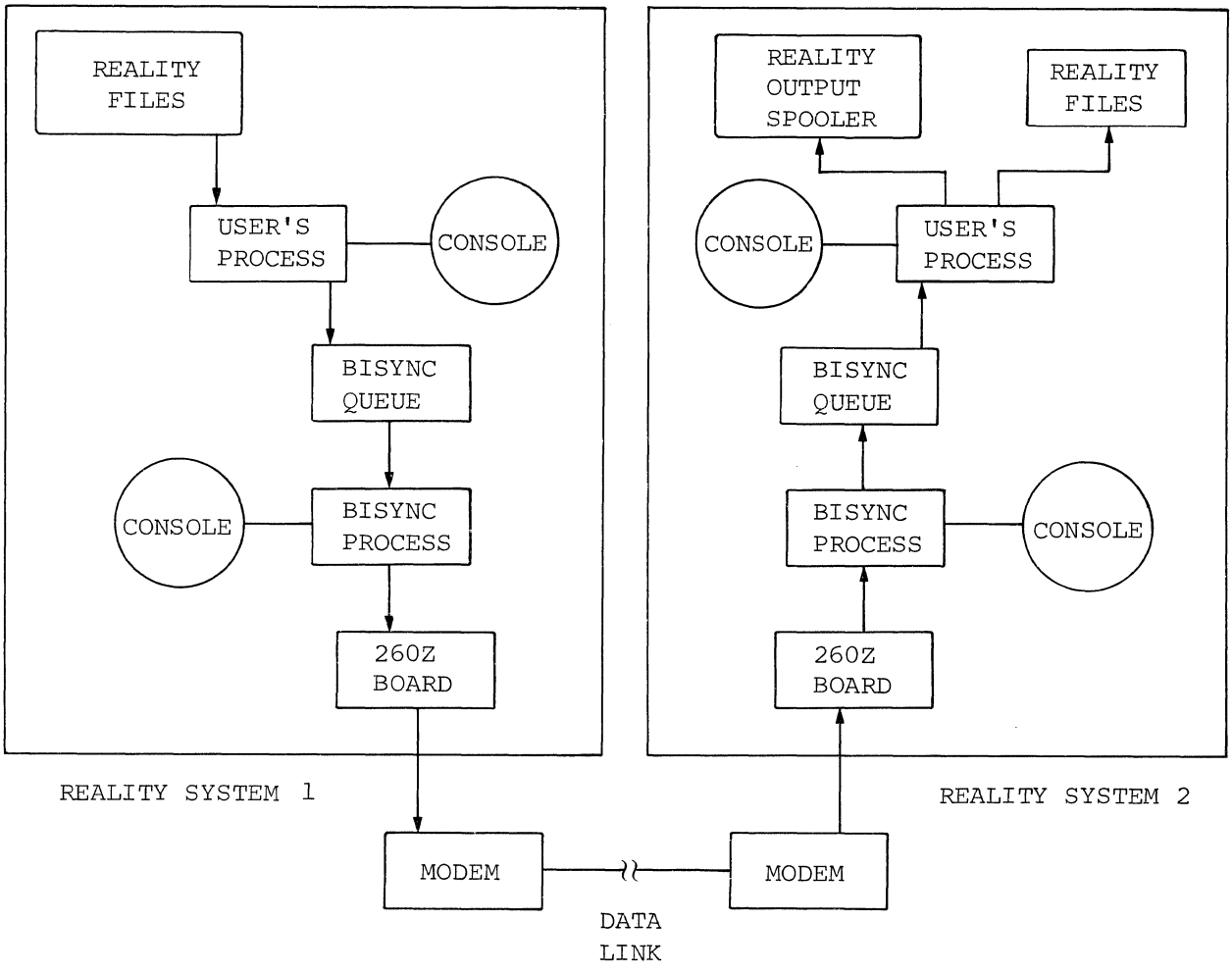


Figure B. Overall Diagram of Reality-to-Reality Bisync Communication

2.2 BISYNC MESSAGES

All data in Bisync is transmitted over the data link as binary-coded characters which comprise a text message. Data link control characters are required with each text message to delimit various portions of the message and control the transmission. The body of the message may contain programs, data, or any meaningful information that may be represented with binary-coded characters.

A structured transmission message consists of one or more blocks of text data. Text is transmitted in blocks to provide more accurate and efficient error control. Each block of text data is preceded by an STX (start of text) control character and, except for the last block, is terminated with an ETB (end of text block) control character. The last block of text data is terminated with an ETX (end of text) control character.

2780 Protocol limits the block size to 400 characters including text data, control characters, tab and vertical forms controls, block check characters, etc.

A transmission block is subdivided into records. In 2780 Protocol, these records correspond to card images, with a maximum size of 80 characters per record. Reality Bisync also limits transmission record size to 80 characters but messages received may have records up to 140 characters (line printer width plus control characters). When transmitting fixed length records (80 characters), each record in a transmission block is terminated by an ITB (end of intermediate transmission block) control character, except for the last record in the block, which is terminated by an ETB. If transmitting variable length records (0 to 80 characters), the records are terminated with an EM (end of media) control character followed by an ITB. As with fixed length records, the last record in a block is terminated by an ETB.

When transmitting text data in transparent mode, data link characters (STX, ITB, ETB, and ETX) are preceded by a data link escape character (DLE). A DLE character may be part of the actual text data, and is identified as being text with a DLE following (i.e., DLE DLE).

Figure A shows an example format for normal text blocks. Figure B shows examples of fixed and variable length records. Transparent text blocks are shown in Figure C.

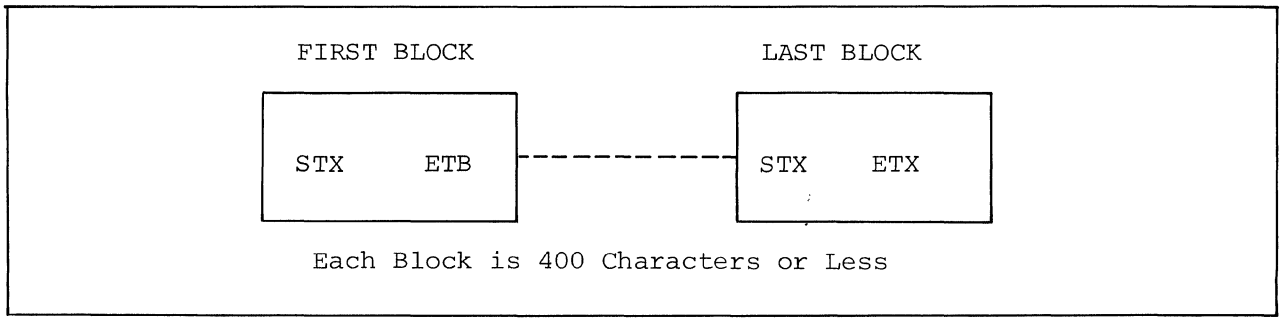


Figure A. Transmission Blocks

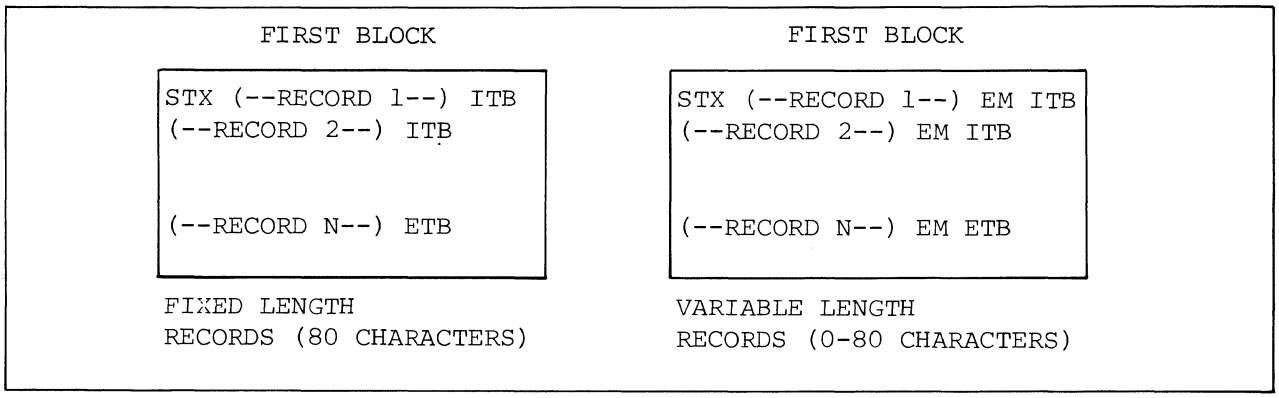


Figure B. Fixed and Variable Length Records

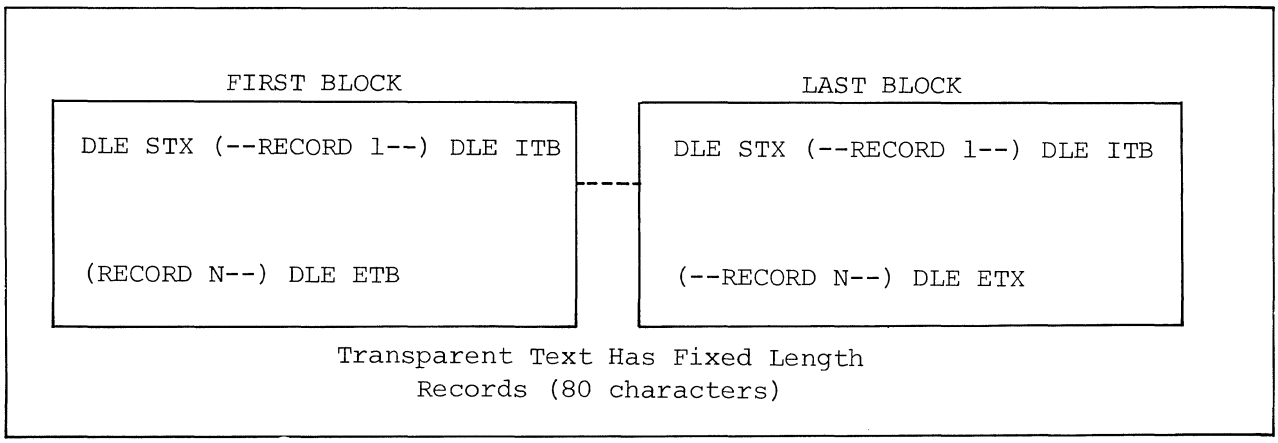


Figure C. Transparent Text Blocks

2.3 TRANSMIT VERB

The TRANSMIT verb structures Reality file items into transmission messages and enters them into the Bisync queue.

The TRANSMIT verb can structure file items or entire files. Each attribute of an item will become a record in the message. Each record is limited to 80 characters. If an attribute is greater than 80 characters, the excess will be truncated and lost.

The general form of the verb is:

```
TRANSMIT file-name item-list (options)
```

The item-list may consist of one or more items within the file, separated by blanks, or an asterisk (*) to indicate all items in the file. The item(s) or file will then be structured into a transmission message in the format specified by the options and entered into the Bisync queue. The options are discussed in detail in Section 2.4.

Each message formed by the TRANSMIT verb is assigned a four-digit identification number in the range 0000 to 9999. After a message is structured, Bisync returns the message:

```
MESSAGE 'NNNN' ENTERED IN QUEUE
```

where 'NNNN' is the identification number assigned to the message. When the Bisync process is activated, messages are transmitted in ascending order of identification number.

Figure A shows the format of the TRANSMIT verb. Figure B shows the function of the TRANSMIT verb. Examples of TRANSMIT verb are given in the next section.

```
TRANSMIT file-name item-list (options)
```

Figure A. TRANSMIT Verb Format

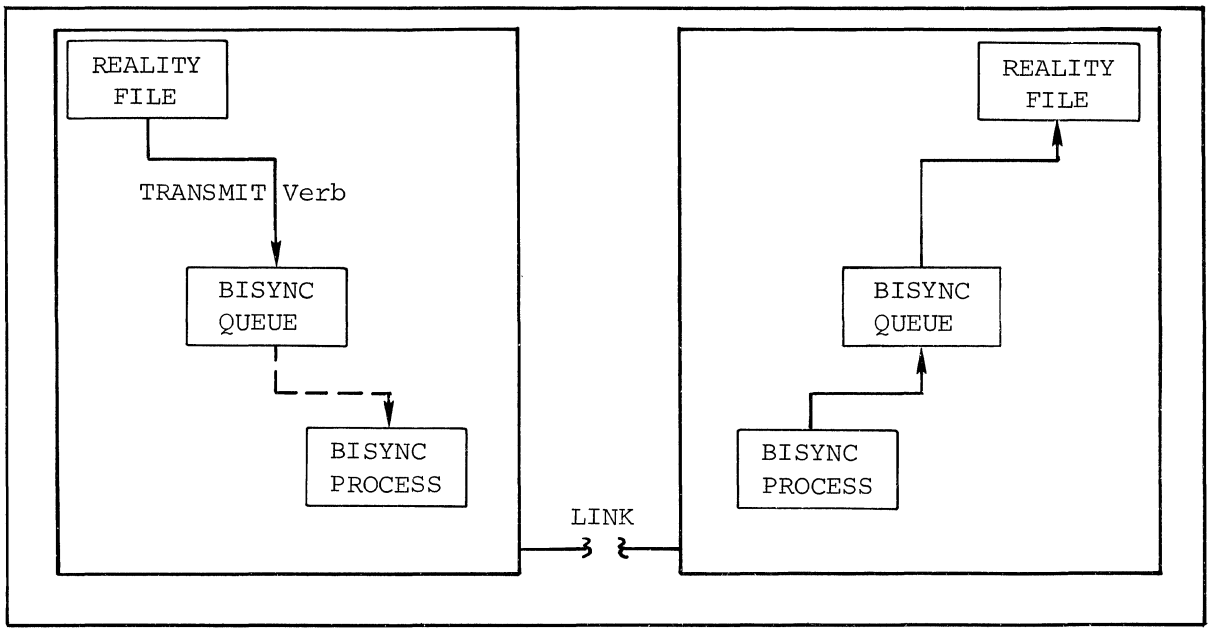


Figure B. Function of TRANSMIT Verb

2.4 TRANSMIT VERB OPTIONS

TRANSMIT verb options allow the selection of optional transmission features. With no options, the default is EBCDIC-coded, variable length records up to 80 characters, and multiple records per block.

C - Transmit Transparent EBCDIC

The ASCII data within the specified item(s) or file is converted to EBCDIC and structured into a transmission message in the transparent text mode. Reality system delimiters will be stripped from the data. Selecting the transparent text mode automatically selects fixed length records (F option), and any short records will be padded with EBCDIC blanks to 80 characters.

F - Transmit Fixed Length Records

Records are 80 characters in size. Attributes shorter than 80 characters will be padded with blanks to 80 characters.

M - Transmit Multiple Records Per Block (Default)

In this mode, records will be placed in transmission blocks until either seven records are in the block or the block size exceeds 400 characters (maximum block size for 2780). When fixed length or transparent records are being transmitted, only four records will fit in a block.

N - Transmit Normal EBCDIC (Default)

The ASCII data within the specified item(s) or file is converted to EBCDIC and structured into a transmission block in the normal text mode.

S - Transmit Short (Variable Length) Records (Default)

Records may be variable length, up to a maximum of 80 characters. Short records (less than 80 characters) will have an EM character appended to them. This mode is recommended when possible as it will reduce transmission time.

T - Transmit Transparent ASCII

The ASCII data within the specified item(s) or file is structured into a transmission message in the transparent text mode. Transparent text mode automatically selects fixed length records (F option), and any short records will be padded with ASCII blanks to 80 characters. This mode should be selected when transmitting to another Reality or transmitting Reality assembler output because the system delimiters will not be stripped and no conversion to and from EBCDIC is required. All data link control characters will be in EBCDIC.

X - Transmit Two Records Per Block

Each transmission block, normal or transparent text, will contain two short or fixed length records.

Default options are N, S, and M. Therefore, if the only option specified is the F option, the message will be structured in normal text (N option), fixed length records (F option), and with multiple records per block (M option).

C	Transmit Transparent EBCDIC
F	Transmit Fixed Length Records
M	Transmit Multiple Records Per Block*
N	Transmit Normal EBCDIC*
S	Transmit Short Records*
T	Transmit Transparent ASCII
X	Transmit Two Records Per Block
*	Indicates A Default Option

Figure A. Transmit Options

:TRANSMIT BP ACCTS (CR) MESSAGE '0002' ENTERED IN QUEUE	Item 'ACCTS' formed into Message using Default Options of Normal EBCDIC (N), Short Records (S), Multiple Records per Block (M)
:TRANSMIT ACCTS * (C) (CR) MESSAGE '0003' ENTERED IN QUEUE	ACCTS File formed into Message in Transparent EBCDIC (C), Fixed Length Records (F), Multiple Records per Block (M)
:TRANSMIT SYSTEM - OBJECT BMAP (T) (CR) MESSAGE '0010' ENTERED IN QUEUE	Item 'BMAP' formed into Message in Transparent ASCII (T), Fixed Length Records (F), Multiple Records per Block (M)

Figure B. Examples of TRANSMIT Verb

2.5 START-BSC VERB

The START-BSC verb activates the Bisync process and allows selection of the type of station (i.e., primary or secondary) and mode of operation (attended or unattended).

The START-BSC verb is used to activate the Bisync Process. The Bisync Process runs on its own terminal and occupies one logical process (similar to the Reality output spooler). The Process is assigned to a line which is not logged on. Bisync may be active on only one line at a time. The terminal is used as the Bisync console for displaying messages concerning the status of the Process.

The general form of the verb is shown in Figure A. The line number is the line to which the Bisync Process will be assigned. The message "BISYNC PROCESSOR INITIATED" is returned to the user and to the Bisync Process's console. If the Bisync Process has previously been started, the message "BISYNC PROCESSOR ALREADY ACTIVE" is returned to the user's terminal. The valid options for the verb are described in the following paragraphs.

P - Primary Station

Designates the Reality system as the primary station in a communications link. Designated as a primary station, the Reality system will send out ENQ characters every second when bidding for the communications line and will wait one second for the other station to reply.

S - Secondary Station

Designates the Reality system as the secondary station in a communications link. Designated as a secondary station, the Reality system will send out ENQ characters every three seconds when bidding for the communications line and will wait three seconds for the other station to reply.

A - Attended Mode

The attended option is set when operator intervention is desired if an abnormal condition occurs while transmitting or receiving a message. If an abnormal condition (such as line failure) occurs, the Process will request operator intervention (via a message on Bisync's console). The operator must restart Bisync via the RESTART-BSC verb (see Section 2.6), at which time the operator may select to restart the transmission or reception of a message either from the point of interruption or from the beginning. This option should not be specified when using the Process in an unattended, autoanswer mode, as operator intervention is then required to restart the Process.

Unattended Mode

The unattended mode is the default mode. If an abnormal condition occurs in this mode, a partially received message is deleted from the queue and a partially transmitted message is retransmitted from the beginning of the message.

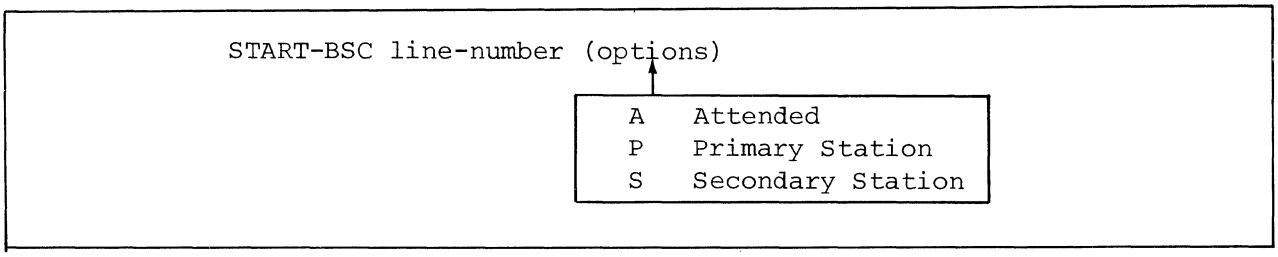
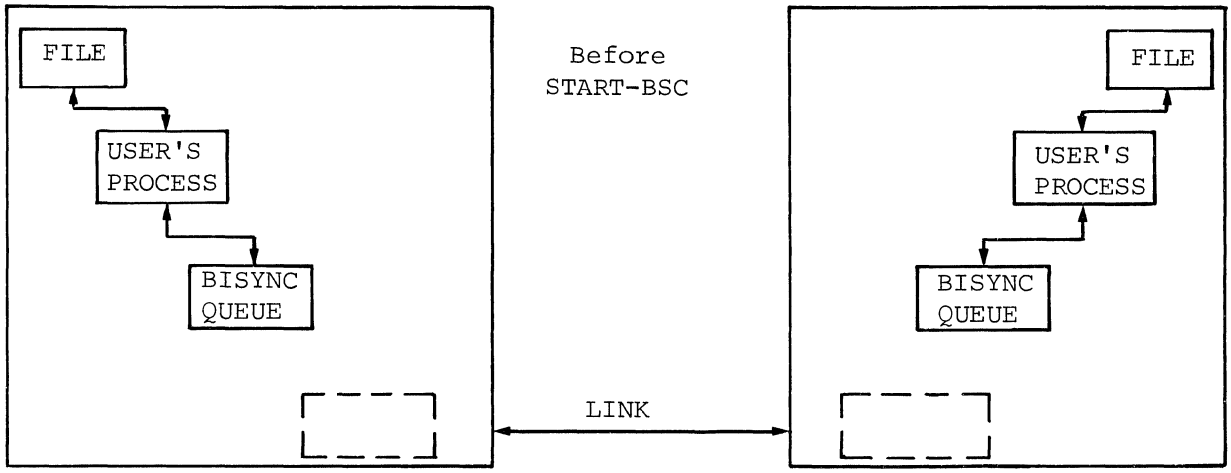
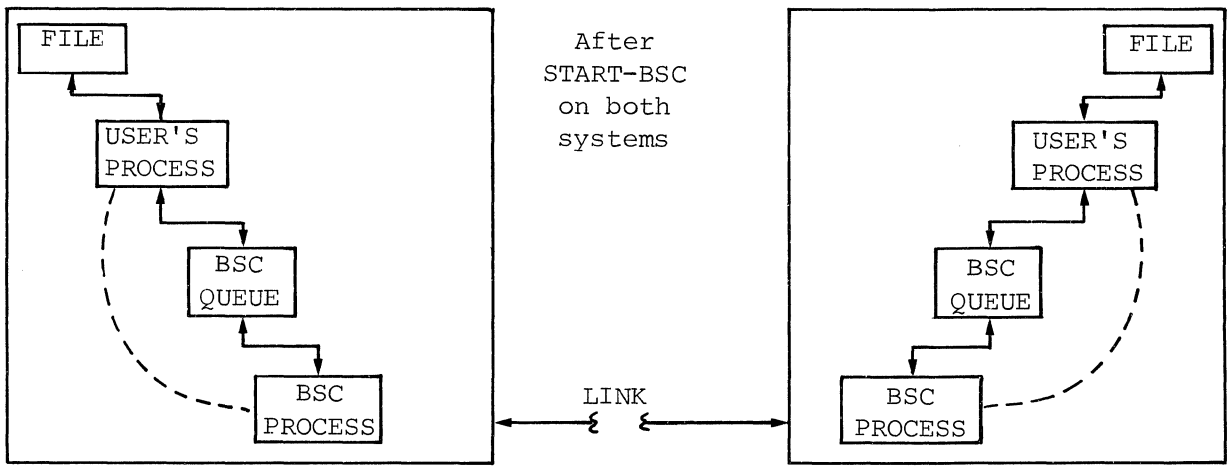


Figure A. START-BSC Verb Format





(Reality to Reality)

Figure B. Function of START-BSC

:START-BSC 0 (P) CR

BISYNC PROCESSOR INITIATED

:START-BSC 0 (S) CR

BISYNC PROCESSOR INITIATED

2.6 RESTART-BSC VERB

The RESTART-BSC verb restarts the Bisync Process after an abnormal condition occurs when operating in the attended mode (A option used with START-BSC).

The RESTART-BSC verb is effective only when operating in the attended mode of Bisync (see Section 2.5 - START-BSC verb). It has no effect when operating in an unattended mode. The verb is used to restart the Bisync Process if an abnormal condition, such as a line failure, occurs while transmitting or receiving a message. When Bisync is in the attended mode, a condition such as line failure will cause the message:

OPERATOR INTERVENTION REQUIRED

to be displayed on the Bisync console. The user must then restart the Bisync via the RESTART-BSC verb. When Bisync is restarted, the message:

BISYNC PROCESS RESTARTED

is returned to the user, and Bisync is restarted with the message that was being processed when the abnormal condition occurred.

The general form of the verb is:

RESTART-BSC (N)

Figure B shows an example of the RESTART-BSC verb. The N option is discussed in the following paragraph.

N - New Start

The N option is the only valid option for the RESTART-BSC verb. If the N option is specified, a partially received message is deleted and the next reception starts a new message. During message transmission, the N option will cause the partially transmitted message to be retransmitted from the beginning during the next transmission.

When Bisync is restarted without the N option while transmitting, the block of data that was being transmitted when the problem occurred is sent as the first block of the new transmission. If Bisync is restarted (without the N option) while receiving a message, the first block of data received after restart is appended to the end of the partially received message.

Using the RESTART-BSC verb without the N option may lead to duplication of a block of data, although usually not. Duplication will occur if the receiving station received a block of data correctly but was unable to respond with a positive acknowledgment before the abnormal condition occurred. The transmitting station restarts transmission with the first block of data that it did not receive a positive acknowledgment for, thus causing duplication of one block of data.

```
RESTART-BSC (N)
                N = New Start
```

Figure A. RESTART-BSC Verb Format

:RESTART-BSC (CR)	A partially transmitted message will be retransmitted starting with the block following the last block transmitted. A partially received message is held in the queue and the next block received will be appended to the partially received message.
BISYNC PROCESS RESTARTED	
:RESTART-BSC (N) (CR)	A partially transmitted message will be retransmitted from the beginning. A partially received message will be deleted from the queue and the next block received will start a new message.
BISYNC PROCESS RESTARTED	

Figure B. Example of RESTART-BSC Verb

2.7 ABORT-BSC AND STOP-BSC VERBS

The ABORT-BSC verb is used to terminate the transmission or reception of a message. The STOP-BSC verb is used to terminate the Bisync Process after all transmission and reception is complete.

ABORT-BSC is used when it is desired to deactivate Bisync during transmission or reception of a message. The general form of the verb is:

```
ABORT-BSC
```

ABORT-BSC immediately terminates the currently active transmission or reception of a message. A message will be displayed on the user's terminal:

```
BISYNC PROCESSOR TERMINATED
```

The communications line will be disconnected and a message displayed on the Bisync console:

```
BISYNC COMMUNICATIONS LINE DISCONNECTED
```

If a message is currently being received, further reception is stopped and the partially received message is deleted from the queue. During message transmission, the transmission is stopped and the partially transmitted message is returned to the queue.

If Bisync is not active (transmitting or receiving a message), the ABORT-BSC verb has no effect. The message:

```
BISYNC PROCESS INACTIVE
```

will be returned to the user's terminal.

The STOP-BSC verb is used to terminate Bisync after all transmission and reception is complete. STOP-BSC causes the Bisync Process to be terminated. The general form of the verb is:

```
STOP-BSC
```

The Bisync Process is terminated and a message is returned to the user's terminal:

```
BISYNC PROCESSOR TERMINATED
```

The terminal that was assigned as the Bisync console is returned to LOGON and is freed for other use.

Figure B shows an example of the ABORT-BSC verb. Figure C shows an example of the STOP-BSC verb.

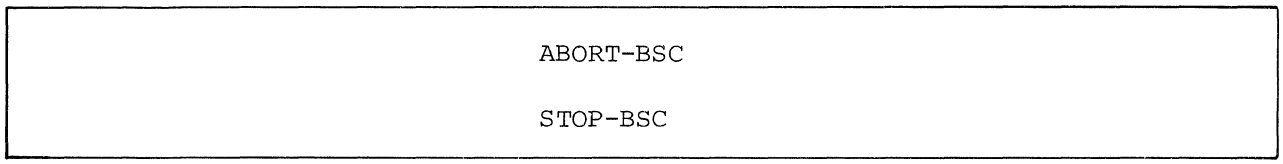


Figure A. ABORT-BSC and STOP-BSC Verb Formats

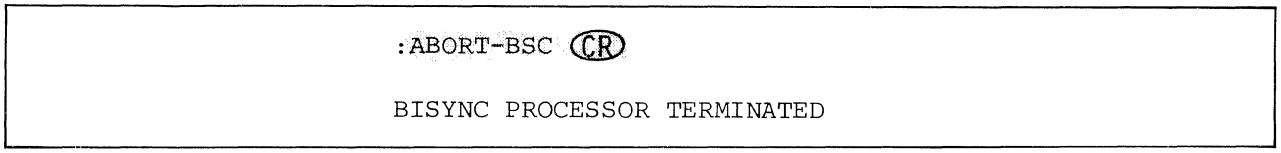


Figure B. Example of ABORT-BSC Verb

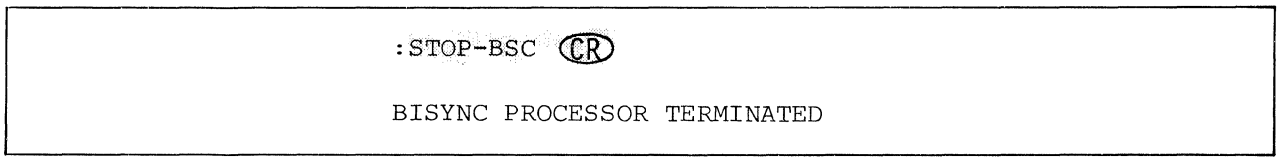


Figure C. Example of STOP-BSC Verb

2.8 THE FILE-MESSAGE VERB

The FILE-MESSAGE verb moves messages out of the Bisync queue, and makes them into items in a specified file.

The verb usage format is

```
:FILE-MESSAGE message-number (options)
TO:  file-name [item-name]
```

Normally each record in the message becomes a different item in the file. Item-id's are assigned numerically from ØØØ1. No "item-name" need be entered (indicated by brackets above).

Alternately, the 'I' option may be used to convert the message into one item. Each line becomes one record.

Valid options are listed below; they are described in detail in the following section.

T	Process horizontal tabs
F	Process vertical forms control
I	File into one item
R	Format Reality object code (I option gets set)
C	Convert EBCDIC to ASCII (for transparent text)

The FILE-MESSAGE verb is normally used on received messages. It may, however, be used on a message which has been placed in the Bisync queue of the transmitting system by the transmit verb.

Note that a message which is filed into an item cannot exceed the limit of 32,267 bytes.

```
FILE-MESSAGE message-number (Options)

TO:  file-name [item-name]
```

Figure A. General Form of the FILE-MESSAGE Verb

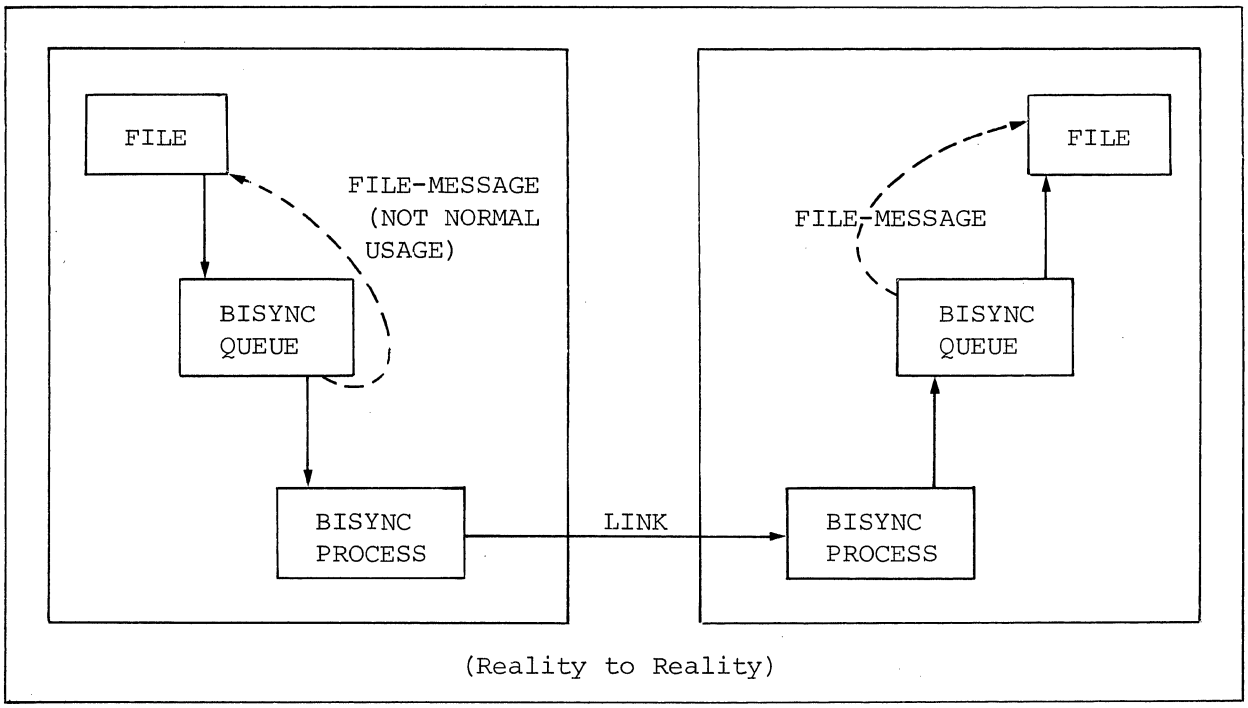


Figure B. Function of the FILE-MESSAGE Verb

2.9 OPTIONS OF THE FILE-MESSAGE VERB

FILE-MESSAGE verb options allow tabs and vertical forms control, single-item filing, EBCDIC to ASCII conversions, and formatting of items containing Reality assembler output.

T - Process Horizontal Tabs

If the message contains a tab format record as its first record and the message records contain horizontal tab characters, tabbing will be done on the records as they are filed by inserting an appropriate number of blanks between fields in the record. If not specified, the tab characters will be left in the records.

F - Process Vertical Forms Control

If the records in the message begin with forms control characters, the appropriate form and line feeds are inserted in the filed message. However, 'skip to channel' commands will be treated as 'top of form'. If the F option is not specified, the forms control characters are left at the beginning of the filed records.

I - File Into One Item

When the I option is specified, the message is placed in one item in the file, each record of the message becoming an attribute of the item. The records of the message must not exceed a total of 32K bytes, the maximum item size. If the message is too long, an error message is given to the operator, the item being built is deleted from the file, and the message remains in the message queue. This option will be used when filing received Reality object code. If this option is not specified, the records in the message will be filed one to an item within the file. The records will be given sequential numeric item-id's as they are filed, starting from 0001.

R - Format Reality Assembler Output into File Item (also sets I option)

The R option is used when a frame of Reality assembler output has been received as a message. The I option (file message into one item) is set whenever the R option is specified. The Reality assembler output will be properly formatted into an item so that it may be loaded into the system. This entails removing pad characters (see Figure B).

C - Convert EBCDIC to ASCII (used when filing transparent text)

The EBCDIC data in the received message is converted to ASCII before the data is filed. This option should normally be specified unless it is known that the message consists of ASCII data from another Reality system or hexadecimal data.

```
T     Process horizontal tabs
F     Process vertical forms control
I     File into one item
R     Format Reality assembler output into file item
C     Convert EBCDIC to ASCII
```

Figure A. FILE-MESSAGE Options

```
:TRANSMIT USER-MODES FORMAT2 (T) CR
:                                                    } Done on
:                                                    } Transmitting
:FILE-MESSAGE 0010 (R) CR (Note:  I not needed)      } Reality System
TO: MODES FORMAT2 CR                                 }
                                                     }
10 MAR 1977 12:30:04 MESSAGE '0010' DELETED          } Done on
                                                     } Receiving
:MLOAD MODES FORMAT2 CR                              } Reality System
MODE 'FORMAT2' LOADED; . . . . . .                   }
```

Figure B. Sequence Showing Options for Transmitting and Filing Assembler Output

```
:FILE-MESSAGE 1846 (T,F,C) CR        The message 1846 is placed
                                     into the file INVOICES.
TO: INVOICES CR                      Tabbing, forms control, and
                                     EBCDIC to ASCII conversion
DATE TIME MESSAGE '0004' DELETED     are specified as options.


:FILE-MESSAGE 0006 (I) CR            The message 0006 is placed
                                     in the item BOSTON in the
TO: ADDRESS-LIST BOSTON CR           ADDRESS-LIST file.
```

Figure C. Examples of the FILE-MESSAGE Verb

2.10 SPOOL-MESSAGE VERB

The SPOOL-MESSAGE verb is used to output a message via the Reality spooler.

The SPOOL-MESSAGE verb sends messages to the Reality output spooler a line at a time. The general form is:

SPOOL-MESSAGE message-number (options)

Recognized options are S, H, C, and Z.

S - Suppress

The S option suppresses tabs and forms control, and allows automatic paging. This option should be used if the received message does not contain imbedded forms control characters. The message will then be automatically paged.

H - Hold Message in Message Queue

With the H option, the message is printed but is not subsequently deleted from the message queue; otherwise the message is automatically deleted. This option may be used for printing multiple copies of the message or for printing and filing the same message. *Note that this option is independent of the 'H' in the SP-ASSIGN verb.*

C - Convert EBCDIC to ASCII

The C option is available for use when printing a transparent text message. This option is normally specified when printing transparent text.

Z - Spool Immediate

The Z option is used to begin output of a message to the Reality output spooler as soon as reception begins. It is advisable to assign the spooler via the SP-ASSIGN verb for instant printing (I option) or direct output (N option) to cause simultaneous spooling and despooling of the message.

SPOOL-MESSAGE message-number (options)

S	Suppress tabs and forms control
H	Hold message in message queue
C	Convert EBCDIC to ASCII
Z	Instant spooling

Figure A. SPOOL-MESSAGE Verb and Options

```
:SP-ASSIGN  IH (CR)
:SPOOL-MESSAGE 0256 (S,C,Z) (CR)

MESSAGE '0256' DELETED
```

Message 0256, a transparent text message, was printed after converting the EBCDIC text to ASCII. Since the H option was not specified, the message was deleted from the Bisync queue after printing. The message started to print as it was being received. It will be held in the spooler's queue due to the 'H' in the SP-ASSIGN.

Figure B. Example of SPOOL-MESSAGE Verb

2.11 DISPLAY-MSG-QUE

The DISPLAY-MSG-QUE verb may be used to display the status of all messages currently in the Bisync message queue.

The DISPLAY-MSG-QUE verb generates a list to the terminal of all messages in the Bisync queue. The general form of the verb is:

```
DISPLAY-MSG-QUE
```

A line of information is generated for each message in the queue. This generates a display with the following headings:

```
NAME      ACCOUNT      CHANNEL      TIME      DATE      STATUS
```

The NAME is the identification number of the message which is automatically assigned by the system. The identification number is in the range 0000 to 9999.

The ACCOUNT is the account the user was logged onto when the message was entered in the queue via the TRANSMIT verb. A received message will not have an ACCOUNT field.

The CHANNEL is the line number of the terminal from which the message was entered into the queue via the TRANSMIT verb. A received message will not have a CHANNEL field.

TIME is the time the message was entered into the queue via the TRANSMIT verb or the time the message was received.

DATE is the date the message was put into the queue via the TRANSMIT verb, or the date the message was received.

STATUS is displayed as a code indicating the current status of the message, such as received, waiting to be transmitted, currently being transmitted, etc. Figure B shows the possible status codes.

Figure C shows an example of the DISPLAY-MSG-QUE.

```
DISPLAY-MSG-QUE
```

Figure A. DISPLAY-MSG-QUE Verb Format

```
T     Message is waiting to be transmitted
T*    Message is currently being transmitted
R     Message has been received
R*    Message is currently being received
TH    Message was partially transmitted when a condition
      requiring operator intervention occurred (attended mode)
RH    Message was partially received when a condition
      requiring operator intervention occurred (attended mode)
```

Figure B. Status Codes

Example

```
:DISPLAY-MSG-QUE (CR)
BSC MESSAGE QUEUE              10:03:30     11 MAR 1977     PAGE 1

NAME  ACCOUNT  CHANNEL          TIME           DATE          STATUS

0154  SYSPROG      0           9:46:05     11 MAR 1977        T*

0155                           9:51:16     11 MAR 1977        R

0156  ALICE        5           10:01:47    11 MAR 1977        T

END OF QUEUE
```

Description

Message 0154 was entered into the queue from the terminal attached to line 0 and logged onto the SYSPROG account at 9:46:05 on 11 MAR 1977, and is currently being transmitted. Message 0155 was received at 9:51:16 on 11 MAR 1977, and has no ACCOUNT or CHANNEL associated with it. Message 0156 was entered in the queue by user ALICE from line 5 at 10:01:47 on 11 MAR 1977, and is waiting to be transmitted.

Figure C. Example of DISPLAY-MSG-QUE Verb

2.12 DUMP-MESSAGE, DISPLAY-MESSAGE AND KILL-MESSAGE VERBS

The DUMP-MESSAGE verb is used to examine a message on the Bisync queue in hexadecimal. The DISPLAY-MESSAGE verb is used to display a message on the Bisync queue in character format. The KILL-MESSAGE verb is used to delete a message from the Bisync queue.

The general form of the DUMP-MESSAGE verb is:

```
DUMP-MESSAGE message-number {(P)}
```

Output is in hexadecimal to the screen. If the (P) is included, output is in hexadecimal to the printer. All line control characters are displayed. Figure B shows an example of the DUMP-MESSAGE verb.

The general form of the DISPLAY-MESSAGE verb is:

```
DISPLAY-MESSAGE message-number
```

Output is in character format to the screen. If the message was transmitted in transparent EBCDIC, the characters will be displayed in their EBCDIC equivalent, and the pad characters (for records less than 80 characters) will display as @'s. Figure C shows examples of the DISPLAY-MESSAGE verb.

The general form of the KILL-MESSAGE verb is:

```
KILL-MESSAGE message-number
```

The specified message is delete from the Bisync queue. Figure D shows an example of the KILL-MESSAGE verb.

```
DUMP-MESSAGE    message-number (P)
DISPLAY-MESSAGE message-number
KILL-MESSAGE    message-number
```

Figure A. DUMP-MESSAGE, DISPLAY-MESSAGE, and KILL-MESSAGE Verb Formats

```
:DUMP-MESSAGE 0025 (CR)       Message 0025 is dumped to the terminal
                              in hex, including all line control
MESSAGE '0025'                characters.

3232

02C1C2C3C4191F

3232

02C5C6C7C81903
```

Figure B. Example of DUMP-MESSAGE Verb

```
:DISPLAY-MESSAGE 0004 (CR)
QVHU@HK@BTICH@@@@@@@@@@@@@@——┤ ├—— @@@@
RRWUP@FVYEBC@SAUE@@@@@@@@@@——┤ ├—— @@@@
IYEIUE@CAK@@@@@@@@@@@@@@@@@——┤ ├—— @@@@
YRVXP@@@@@@@@@@@@@@@@@@@@@@——┤ ├—— @@@@

:DISPLAY-MESSAGE 0005
JOHN H. SMITH
22750 FOREST LANE
IRVINE CA.
92680

Message number 0004 was transmitted in transparent EBCDIC.
Message number 0005 was transmitted in normal EBCDIC.
```

Figure C. Examples of DISPLAY-MESSAGE Verb

```
:KILL-MESSAGE 0005 (CR)

30 MAR 1977  15:52:41  MESSAGE '0005' DELETED
```

Figure D. Example of KILL-MESSAGE Verb

2.13 DATA LINK CONTROL CHARACTERS

> Knowledge of the data link control characters and timeouts is not essential to the use of Bisync. You may wish to resume reading with Chapter 3.

Bisync uses a set of control character sequences to maintain control of the data link. These control character sequences are discussed in the following pages with respect to their usage in 2780 Protocol.

Line control is maintained via a set of control character sequences that have a predefined meaning for Bisync. The control character sequences may consist of a single data link control character or a two-character data link control sequence. A single control character's meaning may change when it is combined with another character, a DLE. These control characters and sequences are described in detail in this and the following pages. Figure A shows a list of the line control characters.

Transparent Mode Control Characters

When transmitting data in the transparent mode, all data link control characters are preceded with a DLE control character to identify them as being control characters. Figure B shows some examples of transparent control characters.

BCC - Block Check Character

Bisync uses a cyclic redundancy check (CRC) as a means of detecting transmission errors in data. Reality Bisync uses the CRC-16 for 8-bit transmission codes. The message block that is to be transmitted is treated as one long binary number, which is divided by a constant 16-bit binary number. The 16-bit remainder is referred to as the BCC and is transmitted following an ITB, ETB, or ETX. The receiving station divides the received block by the known 16-bit number. If the remainder it generates is the same as the BCC received, the block is assumed to have been received correctly. The BCC is transmitted as two characters (16 bits) but is functionally one sequence. The polynomial used as the divisor is:

$$1^{16} + 1^{15} + 1^{2} + 1$$

Pad Character (FF)

Pad characters are used in Bisync to ensure that the last characters of a transmission are properly transmitted by the data set. A pad character also follows a NAK (negative acknowledgment) to ensure that a transmission line error doesn't change a positive response (ACK characters) to a negative response. The trailing pad character must be all 1 bits (a hex FF).

CONTROL CHARACTER (EBCDIC STANDARD)	CODE (HEXADECIMAL)	DESCRIPTION
ACK0	1070	Positive Acknowledgment (0)
ACK1	1061	Positive Acknowledgment (1)
DLE	10	Data Link Escape
DLE EOT	1037	Disconnect
ENQ	2D	Enquiry
EM	19	End of Media
EOT	37	End of Transmission
ESC	27	Escape
ETB	26	End of Text Block
ETX	03	End of Text
HT	05	Horizontal Tab
ITB	1F	Intermediate Text Block
NAK	3D	Negative Acknowledgment
PAD	FF	Pad Character
RVI	107C	Reverse Interrupt
SOH	01	Start of Heading
STX	02	Start of Text
SYN	32	Synchronous Idle
TTD	022D	Temporary Text Delay
WACK	106B	Wait Before Transmit Positive Acknowledgment

Figure A. Summary of Data-Link Control Characters

CONTROL CHARACTER	CODE (EBCDIC)	DESCRIPTION
DLE ETB	1026	End of Text Block
DLE ITB	103D	Negative Acknowledgment
DLE STX	1002	Start of Text

Figure B. Examples of Transparent Text Control Characters

2.14 DATA-LINK CONTROL CHARACTERS - TRANSMISSION CONTROL

Data-link control characters are used to initiate a transmission enquiry, disconnect, and control various other transmission activities.

SYN - Synchronous Idle

The SYN character is used to establish and maintain synchronization of the data link. Each transmission must be preceded with at least two SYN characters. Three SYN characters will precede a 2780 transmission, and a minimum of two are required to obtain synchronization when in receive mode.

ENQ - Enquiry

The ENQ character is used to bid for the line, to request a response, to indicate an I/O error, or to obtain a repeat transmission of a reply if one was not received when expected.

ACK0/ACK1 - Affirmative Acknowledgment

These replies, in proper sequence, indicate that the previous block of data was received without error and the receiver is ready to accept the next block. ACK0 and ACK1 are alternated for each block of data. ACK0 is also the positive response to selection (multipoint) or line bid (point-to-point).

NAK - Negative Acknowledgment

NAK is used to indicate that an error was detected while receiving the last block of data and requests a retransmission of that block.

WACK - Wait Before Transmit Positive Acknowledgment

WACK is used as a positive acknowledgment to a block of data, but indicates that the receiving station is not yet ready for another block of data. This response causes a reset of the ENQ count at the transmitting terminal so that a timeout (usually after three ENQ's) does not occur. A 2780 will accept a WACK but will not transmit a WACK.

RVI - Reverse Interrupt

RVI is a positive response used in place of an ACK0/ACK1 by a receiving station. It is used by a receiving station to request termination of the present transmission due to a high priority message it must transmit to the sending station. The transmitting station responds by transmitting any data in its I/O buffer that prevents it from becoming a receiving station and goes into the receive mode. In a multipoint environment, an RVI may indicate that the control station (when receiving) wishes to communicate with another station on the line. A 2780 will accept but not transmit an RVI.

TTD - Temporary Text Delay

The TTD control sequence is sent by a sending station while transmitting when it wishes to retain control of the line but is not ready to transmit. The receiving station replies with a NAK and waits for transmission to begin. This sequence can be repeated. TTD is also sent by a transmitting station to indicate to the receiver that it is aborting the current transmission. After the receiving station responds with a NAK, the transmitting station sends an EOT (end of transmission).

DLE EOT - Disconnect

The DLE EOT character is sent by a terminal on a switched (dial-up) line to indicate that it is "hanging-up" the line.

CONTROL CHARACTER	CODE (EBCDIC)	DESCRIPTION
SYN	32	Synchronous Idle
ENQ	2D	Enquiry
ACK0/ACK1	1070/1061	Affirmative Acknowledgment
NAK	3D	Negative Acknowledgment
WACK	106B	Wait Before Transmit Positive Acknowledgment
RVI	107C	Reverse Interrupt
TTD	022D	Temporary Text Delay
DLE EOT	1037	Disconnect

Figure A. Data-Link Transmission Control Characters

2.15 DATA-LINK CONTROL CHARACTERS - TEXT CONTROL

Data-link control characters are used to signal beginnings and sendings of data text.

SOH - Start of Heading

SOH precedes a block of heading characters. A heading consists of auxiliary information such as routine and priority. Although 2780 does not use headings, it will accept an SOH and treat it as an STX (start of text) to allow operation on the same line with a terminal using headings.

STX - Start of Text

The STX character precedes a block of text (data) characters and signals the receiving terminal that the text portion of the message will follow.

ETB - End of Transmission Block

ETB is transmitted at the end of the last record of a block of text data, except for the last block. The last block is ended with an ETX (end of text). ETB causes block checking with a line turnaround and a response from the receiving terminal. A block check character sequence immediately follows the ETB. If retransmission of the block is required, all data framed by the ETB and the previous STX is retransmitted.

ETX - End of Text

ETX ends the last block of data in a message. It indicates the end of a complete message. The block check character sequence is sent immediately following the ETX. If retransmission is required, all data framed by the ETX and the preceding STX will be retransmitted.

EOT - End of Transmission

EOT is sent by a transmitting terminal to indicate that it has no more messages to transmit. EOT is sent after a positive response to the last block of data is received. An ETX must have ended the last block of data sent, or the receiving terminal will indicate an error when it receives the EOT.

EOT is also used as a response to a poll when the polled station has nothing to transmit, and as an abort signal to indicate a system malfunction or operational situation that precludes further transmission or reception.

CONTROL CHARACTER	CODE (EBCDIC)	DESCRIPTION
SOH	01	Start of Heading
STX	02	Start of Text
ETB	26	End of Transmission Block
ETX	03	End of Text
EOT	37	End of Transmission

Figure A. Data-Link Text Control Characters

2.16 DATA-LINK CONTROL CHARACTERS -- MISCELLANEOUS

The ITB and DLE characters described in this section are data-link control characters. The other characters described are not actual data-link control characters, but have special meanings when used in a specific sequence.

ITB - End of Intermediate Text Block

The ITB character is transmitted at the end of each record when transmitting fixed length records, except for the last record in a block. The last record is ended with an ETB or ETX. ITB is used to divide a message for error checking purposes without causing line turnaround and an answerback to the transmitting terminal. A block check character sequence is sent immediately following the ITB.

EM - End of Media

EM is used to indicate the end of a record when transmitting variable length records. The EM is immediately followed by an ITB and a block check character, except for the last record of a block. The last record of a block will have an EM followed by an ETB or ETX.

DLE - Data Link Escape

DLE is a control character used to provide supplementary line control characters (e.g., WACK, ACK0/ACK1) and transparent mode control characters. When transmitting in the transparent mode, control characters are preceded by a DLE (e.g., DLE STX, DLE ITB, DLE ENQ). If a data character of DLE is sent in the transparent mode, it is sent as DLE DLE. One of the DLE's is disregarded and the other is treated as data.

ESC - Escape

The ESC character and the character following form a two-character component selection or printer forms control sequence. Reality accepts the ESC sequence as printer forms control. The ESC sequence must be the first two characters in the record when transmitting or receiving.

HT - Horizontal Tab

The HT character is a tab character and is used in three ways. When HT follows the ESC character at the beginning of a record, it signifies that the record is a printer horizontal format control record. An HT within a printer horizontal format control record sets a tab stop for the printer. An HT in records following the printer control record causes a tab to the next tab stop that was set by the printer control record.

CONTROL CHARACTER	CODE (EBCDIC)	DESCRIPTION
ITB	1F	End of Intermediate Text Block
EM	19	End of Media
DLE	10	Data Link Escape
ESC	27	Escape
HT	05	Horizontal Tab

Figure A. Data-Link Miscellaneous Control Characters

2.17 TIMEOUTS

Timeouts are used to prevent indefinite data-link tie-ups due to false sequences or missed turnaround signals by providing a fixed time within which any particular operation must occur. The four timeout functions provided are transmit, receive, disconnect and continue.

Transmit Timeout

This is a nominal one-second timeout that establishes the rate at which sync idle characters are inserted into text data. In normal data, two consecutive sync idle characters (SYN SYN) are inserted every second. In transparent mode, one transparent sync idle sequence (DLE SYN) is inserted every second. Sync idle characters are inserted in the message for timing purposes only and have no effect on the message format.

Receive Timeout

This is a three-second timeout and is used to limit the waiting time tolerated for a transmitting station to reply. It also permits the receiving station to check the line for sync idle signals which indicate that the transmission is continuing. In a multipoint network, the receive timeout is also used to limit the time a secondary station remains in control mode while monitoring the line for its address code.

Disconnect Timeout

This timeout is used on switched network data links. It is a 20 second timeout used to prevent a station holding a connection for prolonged periods of inactivity. After 20 seconds of inactivity, the station will disconnect from the switched network. If one of the stations on the data link is extremely busy with other processes, it may not respond within the given 20 seconds. An option is available on some devices (i.e., 2780) to extend this timeout to 45 seconds. Reality has a standard disconnect timeout of 45 seconds, and may be set higher if required.

Continue Timeout

This is a two second timeout associated with the transmission of TTD and WACK. It is used by stations where the speed of input or output devices affect buffer availability and may cause transmission delays. The purpose of the continue timeout is to permit the stations to send an appropriate affirmative reply if it is not able to receive within the two second interval.

This page intentionally blank

3.1 MODEMS

Modems (data sets) are required at each end of a communications network to interface with the data link.

An important consideration in installing Bisync on Reality is the choice of a modem with the options Reality Bisync requires. A variety of modems are available from telephone companies and independent suppliers.

Reality Bisync, operating in a 2780 emulation mode, does not support true full duplex (simultaneous send/receive). The communications link may be set up for 2-wire or 4-wire half-duplex.

Regardless of the modem supplier or the physical arrangement of the communications link, the following modem options are required:

1. Grounding option - AB connected to AA.

2. Transmitter timing provided by the data set (internal).

3. Automatic answer (not required for private line, but desirable if offered with modem).

4. Interface of data terminal ready - EIA.

5. Electrical interface of ring indicator - EIA.

6. Line turnaround (request-to-send, clear-to-send) delay MUST be 150 milliseconds.

7. Carrier control - for a private line setup, it may be necessary to select switched carrier in order to get the required 150 millisecond delay.

The 150 millisecond line turnaround delay is essential. In many cases, this will determine whether or not a particular modem is acceptable or what other options are available or required on the modem. Figure A shows some of the Bell modems that may be used with Reality Bisync. It should be noted that standard 2780 protocol will not support baud rates higher than 4800 bps. This is not a limitation for Reality, and may be desirable for Reality-to-Reality communications.

There are two options offered on some of the BELL modems that Reality Bisync will not support. These are:

1. New Sync - usually applicable to a multipoint environment only.
2. Automatic calling unit.

If using a modem supplier other than the telephone companies, a DAA (Data Access Arrangement) must be ordered from the telephone company. There are two types of DAA's:

1. CBS - uses digital logic.
2. CBT - uses relays and contact closures.

Either of these are acceptable, but the CBS is preferable and there is very little difference in price.

Model No.	BPS	Description
201A	2000	Dial-up
201B	2400	Private
201C	2000	Dial-up
	2400	Dial-up or Private
208B	4800	Dial-up
209A	9600	Private

Figure A. Some Bell Modems

CBS	Digital Logic
CBT	Relays and Contact Closures

Figure B. Data Access Arrangements (DAA's)

4.1 EXAMPLE: TRANSMITTING A REALITY FILE

STEP 1 The Bisync Processor was started on line one in the unattended mode of operation.

STEP 2 The initiation message was returned on the terminal connected to line one.

STEP 3 All the items (five of them) in the ACCOUNTS file were structured into a message and entered in the message queue as message 0001.

STEP 4 The operator called the computer which will receive the message and placed the data phone in DATA after the connection had been made. The communications-line-enabled message was returned to the Bisync terminal.

STEP 5 After transmission of message 0001 had begun, the transmission message was returned to the Bisync terminal.

STEP 6 While the message was being transmitted, the operator entered the STOP-BSC verb, terminating the Bisync Process at the end of the job.

STEP 7 After completion of the message transmission, the transmission statistics were returned to the Bisync terminal. The number of blocks transmitted and received, and the number of retransmissions were listed. Message 0001 was then deleted from the message queue.

STEP 8 The Bisync Process, seeing that the STOP-BSC verb had been executed, disconnected the line and deactivated itself after transmitting message 0001.

```
STEP 1

     :START-BSC 1 (CR)
     01 NOV 1975 13:49:09 BISYNC PROCESSOR INITIATED

STEP 3

     :TRANSMIT ACCOUNTS* (CR)
     1267
     54862
     7914
     7902
     980002
     MESSAGE '0001' ENTERED IN QUEUE


STEP 6

     :STOP-BSC (CR)
     01 NOV 1975 14:02:15 BISYNC PROCESSOR TERMINATED
```

Figure A. Operator's Display

```
STEP 2

     :01 NOV 1975 13:49:09 BISYNC PROCESSOR INITIATED

STEP 4

     01 NOV 1975 14:01:54 BISYNC COMMUNICATIONS LINE ENABLED

STEP 5

     01 NOV 1975 14:02:04 0001 SYSPROG 00 13:50:05 01 NOV 1975 T*

STEP 7

     01 NOV 1975 14:03:21 11 BLOCKS TRANSMITTED
     01 NOV 1975 14:03:21 11 BLOCKS RECEIVED
     01 NOV 1975 14:03:21 0 TRANSMISSION ERRORS
     01 NOV 1975 14:03:22 MESSAGE TRANSMISSION COMPLETE
     01 NOV 1975 14:03:23 MESSAGE '0001' DELETED

STEP 8

     01 NOV 1975 14:03:23 BISYNC COMMUNICATIONS LINE DISCONNECTED
     BISYNC PROCESSOR INACTIVE

     LOGON PLEASE:
```

Figure B. Bisync Console Display

4.2 EXAMPLE: RECEIVING A REALITY FILE

STEP 1 The Bisync Process was started on line one in the unattended mode.

STEP 2 The initiation message was returned on the Bisync Process' terminal.

STEP 3 Assuming that the data phone was in AUTO, several minutes later a call was received and the communication line enabled.

STEP 4 Message 0002 was received. There were 113 blocks received with two retransmissions.

STEP 5 After waiting 48 seconds without receiving anything else, the Bisync Process disconnected the line.

STEP 6 The operator deactivated the Bisync Process using the STOP-BSC verb.

STEP 7 The STOP-BSC verb cleared wait for a call state in the Bisync Process. The Process saw that the line was disconnected and that the STOP command had been issued, so the Process deactivated itself, and returned to logon.

STEP 8 The operator then filed the received message in the RECEIVED-INVOICES file.

```
STEP 1

     :START-BSC 1 CR
     01 NOV 1975 14:55:02 BISYNC PROCESSOR INITIATED

STEP 6

     :STOP-BSC CR
     01 NOV 1975 15:16:23 BISYNC PROCESSOR TERMINATED

STEP 8

     :FILE-MESSAGE 0002 CR
     TO: RECEIVED-INVOICES CR
     MESSAGE '0002' DELETED
```

Figure A. Operator's Display

```
STEP 2

     :01 NOV 1975 14:55:02 BISYNC PROCESSOR INITIATED

STEP 3

     01 NOV 1975 15:01:36 BISYNC COMMUNICATIONS LINE ENABLED

STEP 4

     01 NOV 1975 15:01:42 0002          15:01:42 01 NOV 1975 R*
     01 NOV 1975 15:07:16 112 BLOCKS TRANSMITTED
     01 NOV 1975 15:07:16 113 BLOCKS RECEIVED
     01 NOV 1975 15:07:16 2 TRANSMISSION ERRORS
     01 NOV 1975 15:07:17 MESSAGE TRANSMISSION COMPLETE

STEP 5

     01 NOV 1975 15:08:05 BISYNC COMMUNICATIONS LINE DISCONNECTED

STEP 7

     01 NOV 1975 15:16:23 BISYNC COMMUNICATIONS LINE ENABLED
     01 NOV 1975 15:16:23 BISYNC COMMUNICATIONS LINE DISCONNECTED

     BISYNC PROCESSOR INACTIVE

     LOGON PLEASE:
```

Figure B. Bisync Console Display

4.3 EXAMPLE: TRANSMITTING REALITY ASSEMBLER OUTPUT

STEP 1 The STRIP verb was used to strip off the source code from the frame of code DBI. When transmitting Reality code, the operator should insure that the first five lines of the frame are comments to preclude the possibility of source code being lost when the lines of code are truncated to 80 characters, since the STRIP verb does not affect the first five lines of code in the frame.

STEP 2 The stripped frame of code was then structured into a transparent ASCII text transmission message.

STEP 3 The operator then started the Bisync Process with the START-BSC verb.

STEP 4 The initialization message was returned to the Bisync Process terminal.

STEP 5 After the line connection was made, the message was transmitted.

STEP 6 After transmitting its message, the Bisync Process went into the receive state. Since nothing was received within 48 seconds, the Process disconnected the line.

```
STEP 1

     :STRIP SYSTEM-MODES DBI (CR)
     DESTINATION: HOLD

STEP 2

     TRANSMIT HOLD DBI (T)
     MESSAGE '0003' ENTERED IN QUEUE

STEP 3

     :START-BSC 1 (CR)
     01 NOV 1975 15:30:00 BISYNC PROCESSOR INITIATED
```

Figure A. Operator's Display

```
STEP 4

     :01 NOV 1975 15:30:00 BISYNC PROCESSOR INITIATED

STEP 5

     01 NOV 1975 15:32:56 BISYNC COMMUNICATION LINE ENABLED
     01 NOV 1975 15:33:01 0003 SYNPROG 00 15:25:13 01 NOV 1975 T*
     01 NOV 1975 15:36:50 52 BLOCKS TRANSMITTED
     01 NOV 1975 15:36:50 52 BLOCKS RECEIVED
     01 NOV 1975 15:36:50 0 TRANSMISSION ERRORS
     01 NOV 1975 15:36:51 MESSAGE TRANSMISSION COMPLETE

STEP 6

     01 NOV 1975 15:37:39 BISYNC COMMUNICATIONS LINE DISCONNECTED
```

Figure B. Bisync Console Display

4.4 EXAMPLE: RECEIVING REALITY ASSEMBLER OUTPUT

STEP 1 The Bisync Process was started on line one by the operator.

STEP 2 The initialization message was returned to the Bisync Process' terminal.

STEP 3 After the line connection was made, message 0004 was received.

STEP 4 Since the operator knew the frame DBI was received, he first deleted the old DBI from the SYSTEM-MODES file using the EDITOR.

STEP 5 The message was then filed into SYSTEM-MODES DBI using the R option (file Reality object code) of the FILE-MESSAGE verb.

```
STEP 1

     :START-BSC 1 (CR)
     01 NOV 1975 15:52:26 BISYNC PROCESSOR INITIATED

STEP 4

     :ED SYSTEM-MODE DBI (CR)
     .TOP (CR)
     .FD (CR)
     DBI DELETED

STEP 5

     :FILE-MESSAGE 0004 (R) (CR)
     TO: SYSTEM-MODES DBI (CR)
```

Figure A. Operator's Display

```
STEP 2

     :01 NOV 1975 15:56:26 BISYNC PROCESSOR INITIATED

STEP 3

     01 NOV 1975 15:59:30 BISYNC COMMUNICATIONS LINE ENABLED
     01 NOV 1975 15:59:45 0004    15:59:45 01 NOV 1875 R*
     01 NOV 1975 16:05:02 52 BLOCKS TRANSMITTED
     01 NOV 1975 16:05:02 53 BLOCKS RECEIVED
     01 NOV 1975 16:05:03 0 TRANSMISSION ERRORS
     01 NOV 1975 16:05:03 MESSAGE TRANSMISSION COMPLETE
     01 NOV 1975 16:05:25 BISYNC COMMUNICATIONS LINE DISCONNECTED
```

Figure B. Bisync Console Display

4.5 EXAMPLE: OPERATING WITH JCL: Transmitting a Ditto Job to an IBM 360 Operating Under DOS/Power

Assume there is a file called JOB-CONTROL and this file contains two items, LOGON and LOGOFF:

LOGON

```
001 * .. RJSTART MICRO
002 * .. LOGON MICRO
003 * $$ JOB DITTO
004 * $$ PRT H
005 * $$ PUN H
006 // JOB DITTO
007 // UPSI 1
008 // EXEC DITTO
009 $$DITTO CPU
```

LOGOFF

```
001 /*
002 $$DITTO EOJ
003 * $$ EOJ
004 * .. LOGOFF
005 * .. RJEND
```

STEP 1 The operator started the Bisync Process on line one.

STEP 2 The initialization message was returned on the Bisync Process' terminal.

STEP 3 The logon job control cards were entered in the message queue.

STEP 4 All the items in the DATA file, the data which is to be dittoed, was entered in the message queue.

STEP 5 The logoff job control cards were entered in the message queue.

STEP 6 The job was sent to the IBM 360. The operator disconnected the line, waiting for the job to be run.

```
STEP 1

     :START-BSC 1(CR)
     01 NOV 1975 16:20:02 BISYNC PROCESSOR INITIATED

STEP 3

     :TRANSMIT JOB-CONTROL LOGON (F)(CR)
     MESSAGE '0005' ENTERED IN QUEUE

STEP 4

     :TRANSMIT DATA * (F)(CR)
     1034
     92685
     .
     .
     .
     28
     4678
     MESSAGE '0006' ENTERED IN QUEUE

STEP 5

     :TRANSMIT JOB-CONTROL LOGOFF (F)(CR)
     MESSAGE '0007' ENTERED IN QUEUE
```

Figure A. Operator's Display

```
STEP 2

     :01 NOV 1975 16:20:02 BISYNC PROCESSOR INITIATED

STEP 6

     01 NOV 1975 16:27:15 BISYNC COMMUNICATIONS LINE ENABLED
     01 NOV 1975 16:27:20 0005 SYSPROG 00 16:20:56 01 NOV 1975 T*
     01 NOV 1975 16:28:32 3 BLOCKS TRANSMITTED
     01 NOV 1975 16:28:32 3 BLOCKS RECEIVED
     01 NOV 1975 16:28:32 0 TRANSMISSION ERRORS
     01 NOV 1975 16:28:33 MESSAGE TRANSMISSION COMPLETE
     01 NOV 1975 16:28:35 MESSAGE '0005' DELETED
     01 NOV 1975 16:28:40 0006 SYSPROG 00 16:23:10 01 NOV 1975 T*
     01 NOV 1975 16:40:16 242 BLOCKS TRANSMITTED
     01 NOV 1975 16:40:16 243 BLOCKS RECEIVED
     01 NOV 1975 16:40:16 12 TRANSMISSION ERRORS
     01 NOV 1975 16:40:17 MESSAGE TRANSMISSION COMPLETE
     01 NOV 1975 16:40:26 MESSAGE '0006' DELETED
     01 NOV 1975 16:40:31 0007 SYSPROG 00 16:28:14 01 NOV 1975 T*
     01 NOV 1975 16:41:02 2 BLOCKS TRANSMITTED
     01 NOV 1975 16:41:02 2 BLOCKS RECEIVED
     01 NOV 1975 16:41:02 0 TRANSMISSION ERRORS
     01 NOV 1975 16:41:03 MESSAGE TRANSMISSION COMPLETE
     01 NOV 1975 16:41:05 MESSAGE '0007' DELETED
     01 NOV 1975 16:41:26 BISYNC COMMUNICATION LINE DISCONNECTED
```

Figure B. Bisync Console Display

4.6 EXAMPLE: OPERATING WITH JCL: Receiving a Ditto Job From an IBM 360 Operating Under DOS/Power

Assume there is a file called JOB-CONTROL and this file contains two items, OUTPUT and LOGOFF.

OUTPUT	LOGOFF
001 * .. RJSTART MICRO	001 * .. LOGOFF
002 * .. LOGON MICRO	002 * .. RJEND
003 * .. OUTPUT ALL	

STEP 1 The operator started the Bisync Process on line one.

STEP 2 The initialization message was returned on the Bisync Process' terminal.

STEP 3 The operator then entered the output job control cards in the message queue.

STEP 4 The line connection was then made and the output cards sent to the 360.

STEP 5 Upon receiving the output cards, the 360 transmitted the results of the previous ditto job.

STEP 6 While receiving the ditto job, the operator entered the logoff cards in the message queue. This must be done while receiving the output. Otherwise, if the logoff is placed in the queue at the same time as the logon, the logoff cards would be transmitted before receiving the output, aborting the job on the 360.

STEP 7 The logoff cards were then transmitted after receiving the output.

STEP 8 The operator then filed the received message in the ACCOUNTS file.

```
STEP 1

     :START-BSC 1(CR)
     01 NOV 1975 17:35:53 BISYNC PROCESSOR INITIATED

STEP 3

     :TRANSMIT JOB-CONTROL OUTPUT (F)(CR)
     MESSAGE '0008' ENTERED IN QUEUE

STEP 6

     :TRANSMIT JOB-CONTROL LOGOFF (F)(CR)
     MESSAGE '0010' ENTERED IN QUEUE

STEP 8

     :FILE-MESSAGE 0009(CR)
     TO: ACCOUNTS(CR)
```

Figure A. Outgoing Display

```
STEP 2

     :01 NOV 1975 17:35:53 BISYNC PROCESSOR INITIATED

STEP 4

     01 NOV 1975 17:38:10 BISYNC COMMUNICATIONS LINE ENABLED
     01 NOV 1975 17:38:15 0008 SYSPROG 00 17:36:15 01 NOV 1975 T*
     01 NOV 1975 17:38:58 1 BLOCKS TRANSMITTED
     01 NOV 1975 17:38:58 1 BLOCKS RECEIVED
     01 NOV 1975 0 TRANSMISSION ERRORS
     01 NOV 1975 17:38:58 MESSAGE TRANSMISSION COMPLETE
     01 NOV 1975 17:39:00 MESSAGE '0008' DELETED

STEP 5

     01 NOV 1975 17:39:07 0009    17:39:07 01 NOV 1975 R*
     01 NOV 1975 17:51:36 236 BLOCKS TRANSMITTED
     01 NOV 1975 17:51:36 237 BLOCKS RECEIVED
     01 NOV 1975 17:51:37 6 TRANSMISSION ERRORS
     01 NOV 1975 17:51:38 MESSAGE TRANSMISSION COMPLETE

STEP 7

     01 NOV 1975 17:51:42 0010 SYSPROG 00 17:42:26 01 NOV 1975 T*
     01 NOV 1975 17:52:03 1 BLOCKS TRANSMITTED
     01 NOV 1975 17:52:03 1 BLOCKS RECEIVED
     01 NOV 1975 17:52:03 0 TRANSMISSION ERRORS
     01 NOV 1975 17:52:03 MESSAGE TRANSMISSION COMPLETE
     01 NOV 1975 17:52:05 MESSAGE '0010' DELETED
     01 NOV 1975 17:52:26 BISYNC COMMUNICATIONS LINE DISCONNECTED
```

Figure B. Incoming Display

4.7 EXAMPLE: OPERATING IN THE ATTENDED MODE

STEP 1 The Bisync Process was started on line 1 in the attended mode.

STEP 2 The communications line was enabled and reception of message 0009 began. A line failure occurred and the line was disconnected. Operator intervention was requested because Bisync was started in the attended mode.

STEP 3 Bisync was restarted without the N option (new start) so message 0009 was not deleted.

STEP 4 The communications line was enabled and reception of message 0009 restarted. The blocks received were appended to the first 4 blocks of message 0009 that were received.

STEP 5 After message reception was complete, the Bisync Process disconnected the line after 48 seconds as no further reception or transmission occurred. The terminal attached to line 1 was returned to logon.

```
STEP 1
     :START-BSC 1 (A) CR

STEP 3
     :RESTART-BSC CR
     01 APR 1977 17:45:02 BISYNC PROCESS RESTARTED
```

Figure A. Operator's Display

```
     01 APR 1977 17:35:53 BISYNC PROCESSOR INITIATED

STEP 2
     01 APR 1977 17:39:05 BISYNC COMMUNICATIONS LINE ENABLED
     01 APR 1977 17:39:09 0009      17:39:09 01 APR 1977 R*
     01 APR 1977 17:40:10  *** LINE FAILURE ***
     01 APR 1977 17:40:10 20 BLOCK TRANSMITTED
     01 APR 1977 17:40:11 19 BLOCKS RECEIVED
     01 APR 1977 17:40:12  1 TRANSMISSION ERRORS
     01 APR 1977 17:40:13 BISYNC COMMUNICATIONS LINE DISCONNECTED
     01 APR 1977 17:40:14 OPERATOR INTERVENTION REQUIRED

     01 APR 1977 17:42:02 BISYNC PROCESSOR RESTARTED

STEP 4
     01 APR 1977 17:44:02 BISYNC COMMUNICATIONS LINE ENABLED
     01 APR 1977 17:44:06 0009       17:46:06 01 APR 1977 R*
     01 APR 1977 17:51:02 92 BLOCKS RECEIVED
     01 APR 1977 17:52:02 92 BLOCKS TRANSMITTED
     01 APR 1977 17:52:02 0 TRANSMISSION ERRORS
     01 APR 1977 17:52:05 MESSAGE TRANSMISSION COMPLETE

STEP 5
     01 APR 1977 17:52:53 BISYNC COMMUNICATIONS LINE DISCONNECTED

     BISYNC PROCESSOR INACTIVE
     LOGON PLEASE:
```

Figure B. Bisync Console Display

5.1 TROUBLESHOOTING

When transmitting or receiving, the Bisync Process returns various messages on the Bisync console. The information contained in these messages can be used to troubleshoot if problems occur with Bisync.

When transmitting data, the number of blocks received is a count of all blocks transmitted and any retrys that were necessary. The number of blocks received shows the count of all responses, whether positive (ACK0/ACK1) or negative (NACK). Also contained in this count is the number of timeouts.

When in the receive mode, the number of blocks received shows the count of blocks received and any timeouts. The transmit count shows the count of all responses.

If the receive count is higher than the transmit count, timeouts are occurring. This could be caused by line turnaround occurring before one of the stations is ready. The most common cause of this is the installation of modems that do not have a 150 millisecond delay in turnaround time (see Section 3.1 on Modems).

If it appears that Reality is loosing data when receiving, the user should note any error conditions that occurred, number of blocks transmitted and received, number of transmission errors and any other pertinent information. If Bisync was running in the attended mode and an abnormal condition (such as a line failure) interrupted transmission, the user should check with the other station to see if retransmission was from the beginning of the message or from the point of interruption. If it was from the point of interruption, it will be necessary to run Bisync in the attended mode. When Bisync is running in the unattended mode and an abnormal condition interrupts transmission, a partially received message will be retransmitted from the beginning. When Bisync is running in the attended mode, the user has the option of saving a partially received message and restarting transmission of a partially transmitted message from the point of interruption.

```
:01 APR 1977 16:20:02 BISYNC PROCESSOR INITIATED
 01 APR 1977 16:27:15 BISYNC COMMUNICATIONS LINE ENABLED
 01 APR 1977 16:28:40 0006 SYSPROG 00 16:23:10 01 APR 1977 T*
 01 APR 1977 16:40:16 242 BLOCKS TRANSMITTED
 01 APR 1977 16:40:16 265 BLOCKS RECEIVED
 01 APR 1977 16:40:16 12 TRANSMISSION ERRORS
 01 APR 1977 16:40:17 MESSAGE TRANSMISSION COMPLETE
 01 APR 1977 16:40:26 MESSAGE '0006' DELETED
 01 APR 1977 16:41:26 BISYNC COMMUNICATION LINE DISCONNECTED

In this transmission sequence, 12 blocks had to be retransmitted and
there were 23 timeouts.
```

Figure A. Transmission Counts

Microdata Corporation
17481 Red Hill Avenue, Irvine, California 92714
Post Office Box 19501, Irvine, California 92713
Telephone: 714/540-6730 · TWX: 910-595-1764