PERKIN-ELMER

OS/32
CHARACTER SYNCHRONOUS COMMUNICATIONS

Reference Manual

S29-543 R02

The Perkin-Elmer Corporation, Computer Systems Division 2 Crescent Place, Oceanport, New Jersey 07757

Printed in the United States of America

PREFACE

This manual describes ITAM/32 Character Synchronous Support. Chapters 1 through 4 describe the binary synchronous line drivers. Chapters 5 through 7 describe the binary synchronous terminal manager. It is assumed that the reader has an overview of ITAM/32 as described in the OS/32 Basic Data Communications Reference Manual.

The R02 revision of this manual revises the character synchronous device code section of Chapter 1.

For further information on ITAM/32 consult the following publications:

MANUAL TITLE	PUBLICATION NUMBER
CS/32 Basic Data Communications Reference Manual	S29-541
OS/32 Asynchronous Communications Reference Manual	48-047
OS/32 Bit Synchronous Communications Reference Manual	S29-544
CS/32 System Planning and Configuration Guide	48-024
Synchronous Data Set Adapter Instruction Manual	H29-277
QSA Programming Manual	C29-473
32-Bit System User Documentation Summary	50-003
IBM Binary Synchronous Communications	GA27-3004-2

For further information on the contents of all Perkin-Elmer 32-bit manuals, see the 32-Bit Systems User Documentation Summary.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTERS (Continued)

TABLE OF CONTENTS (Continued)

CHAPTERS (Continued)

TABLE OF CONTENTS (Continued)

CHAPTERS (Continued)

APPENDICES

FIGURES

FIGURES (Continued)

TABLES

CHAPTER 1
BINARY SYNCHRONOUS LINE DRIVER
PROGRAM IDENTIFICATION

1.1 INTRODUCTION

This chapter describes the Perkin-Elmer ITAM Binary Synchronous Line Driver, Program Number 07-070F09.

The following features apply to binary synchronous line driver support:

- ASCII and EBCDIC code set
- Transparent text on ASCII and EBCDIC lines
- Configuration Utility Program (CUP/32) assignment of translation tables. This permits flexibility in specifying line codes for individual lines.
- User task modification of:
  - Error timeout values
  - Translation table selection
  - Output commands to control adapter and modem
  - Number of leading sync characters transmitted
  - Transparent record size indicator
    (default values are listed in Appendix A)

1.2 SUPPORTED DEVICES

The binary synchronous line driver supports these devices:

CODE	DEVICE CODE
Synchronous Set Adapter (SSA)	168
Quad Synchronous Adapter (QSA)	168

The QSA must be strapped for HDX/FDX only, as previously described. Other functions are under software control.

To communicate with terminals, the binary synchronous line driver uses the binary synchronous communications (BSC) protocol, with ASCII (even or odd parity) or EBCDIC line code. Also, the line driver requires a Western Electric 201 or an equivalent modem.

With line driver (SVC 15) access, the user task is responsible for controlling all handshaking procedures of the line protocol. See Chapter 5 for terminal manager (SVC 1) access.

1.3 DEVICE STATEMENTS FOR BINARY SYNCHRONOUS LINES

The device statements for binary synchronous line devices must be included in the devices configuration statement at sysgen time.

See the OS/32 System Planning and Configuration Guide, or the OS/32 System Generation (SYSGEN) Reference Manual.

1.4 RELATION TO OTHER PROGRAMS

The ITAM binary synchronous line driver is an SVC 15 driver. It requires the ITAM option of OS/32.

1.5 COMMAND REPERTOIRE

The command repertoire for the binary synchronous driver is:

XFER	Transfer command chain address
CXFER	Conditional transfer
WAIT	Interval delay
NOP	No operation
EXAMINE	Fetch device status
READ1	READ one byte
READ2	READ two bytes
READ BUFFER	READ with ITAM buffer management
PREPARE	Search for particular character
ANTI-PREPARE1	Search for specified character; on finding it, search for a character not equal to the specified character.
ANTI-PREPARE2	Search for character not equal to specified character.
WRITE1	WRITE one byte
WRITE2	WRITE two bytes
WRITE BUFFER	WRITE with ITAM buffer management
MODE	Set programmable options
RING WAIT	Wait for phone to ring
ANSWER	Answer the phone
DISCONNECT	Hang up the phone

Two options are included for binary synchronous I/O with chained or queued buffers. These options require two extra bits (bits 2 and 3) to be defined in the buffer flag byte. See Table 1-1.

TABLE 1-1 CHAINED/QUEUED BUFFER FLAG BYTE

BIT NO.	0	1	2	3	4	5	6	7
MEANING	BUSY	DONE	ITB	END TRANS			TTB	

BIT 2 ITB	Setting this bit directs I/O to continue with the next buffer of the chain, after the handling of an ITB for this buffer.
BIT 3 END TRANS	Setting this bit indicates that transparent text mode may end in this buffer if the last two characters of the buffer are a proper termination sequence; i.e., DLE, ETX.
BIT 6 TTB	Setting this bit indicates that this buffer contains transparent text.

1.6 DCB FIELDS FOR BINARY SYNCHRONOUS LINE DRIVER AND TERMINAL MANAGER

The list below specifies the DCB fields required by the binary synchronous line driver and terminal manager. These fields are in the ITAM portion of the DCB, which follows the basic DCB.

DCB.XDCD	Extended device code
DCB.RECS	Transparent records size
DCB.SPCR	Special character for read
DCB.SPCW	Special character for write
DCB.XLT	Translate table address
DCB.LDCT	Count of leading sync characters
DCB.PDCT	Count of trailing pad characters
DCB.LCB	LCB Address
DCB.IOBQ	IOB (I/O Block) queue header

With the IOB queue, the binary synchronous terminal manager queues I/O requests previously put on hold because of a busy condition.

1.5 COMMAND REPERTOIRE

The command repertoire for the BISYNC driver is:

XFER	Transfer command-chain address
CXFER	Conditional transfer
WAIT	Interval delay
NOP	No operation
EXAMINE	Fetch device status
READ1	READ one byte
READ2	READ two bytes
READ BUFFER	READ with ITAM buffer management
PREPARE	Search for particular character
ANTI-PREPARE1	Search for specified character; on finding it, search for a character <u>not</u> equal to the specified character.
ANTI-PREPARE2	Search for character <u>not</u> equal to specified character.
WRITE1	WRITE one byte
WRITE2	WRITE two bytes
WRITE BUFFER	WRITE with ITAM buffer management
MODE	Set programmable options
RING WAIT	Wait for phone to ring
ANSWER	Answer the phone
DISCONNECT	Hang-up the phone

Two options are included for BISYNC I/O with chained or queued buffers. These options require two extra bits (bits 2 and 3) to be defined in the buffer flag byte. See Table 1-1.

TABLE 1-1 CHAINED/QUEUED BUFFER FLAG BYTE

BIT NO.	0	1	2	3	4	5	6	7
MEANING	BUSY	DONE	ITB	END TRANS			TTB	

BIT 2	ITB	Setting this bit directs I/O to continue with the next buffer of the chain, after the handling of an ITB for this buffer.
BIT 3	END TRANS	Setting this bit indicates that transparent text mode may end in this buffer if the last two characters of the buffer are a proper termination sequence; i.e., DLE,ETX.
BIT 6	TTB	Setting this bit indicates that this buffer contains transparent text.

1.6 DCB FIELDS FOR BISYNC LINE DRIVER AND TERMINAL MANAGER

The list below specifies the DCB fields required by the BISYNC line driver and terminal manager. These fields are in the ITAM portion of the DCB, which follows the basic DCB.

DCB.XDCD	Extended device code
DCB.RECS	Transparent-records size
DCB.SPCR	Special character for read
DCB.SPCW	Special character for write
DCB.XLT	Translate-table address
DCB.LDCT	Count of leading SYNC characters
DCB.PDCT	Count of trailing pad characters
DCB.LCB	LCB Address
DCB.IOBQ	IOB (I/O Block) queue header

(With the IOB queue, the BISYNC terminal manager queues I/O requests previously put on hold because of a busy condition.)

1.7 STATEMENT SYNTAX CONVENTIONS

These statement syntax conventions are used in all instruction formats:

CONVENTION	USAGE
Capital letters, parentheses, and punctuation marks	Must be entered exactly as shown
Lowercase letters n	Represent parameters or information provided by the user
Underlining PAUSE	Indicates only the underlined portion of the entry is required
Ellipsis ... $param_1,...,param_5$	Represents an indefinite number of parameters or a range of parameters
Lettering with shading n	Represents a default option
Braces { }	Represent required parameters from which one must be chosen
Brackets [ ]	Represent an optional parameter that can be chosen
Commas ,	Separate parameters and substitute missing positional parameters
Braces inside brackets [{ }]	Represent optional parameters from which one can be chosen

Comma preceding braces inside brackets [,{ }]	Must be entered if one of the optional parameters is chosen However, if the parameters are not positional and the first parameter of the statement is not chosen, the parameter specified as the first is not preceded by a comma.
Comma inside brackets [,]	Must be entered if the optional parameter is chosen
Comma outside brackets except last parameter [,[ ],[ ] [,]]	Must be entered in place of missing positional parameters and to separate optional parameters that are chosen. Commas are omitted for trailing parameters and a comma must be entered with the last specified parameter.
Equal sign separating keyword from parameters KEYWORD=param	Must be entered to associate parameter with keyword

CHAPTER 2
INTRODUCTION TO BISYNC

2.1 INTRODUCTION

Following is a general description of the Binary Synchronous Communications (BSC) protocol applicable to the BISYNC line driver. A more detailed description can be found in General Information -- Binary Synchronous Communications, IBM Publication Number GA27-3004-2.

A transmission on a BISYNC line consists of:

- A series of leading SYNC characters. These are needed to place the receiver into bit and character synchronization.

- Data and/or control characters. These are used to convey data and to enter and exit various modes that may add special meaning to any data.

- A trailing pad character. This ensures integrity of the last character of data.

  Example:

```
SSS         P
YYY  DATA   A
NNN         D
```

BISYNC control characters, listed in Table 2-1, inform the receiving end exactly when to expect certain characters, such as a PAD.

TABLE 2-1 BISYNC CONTROL CHARACTERS

CONTROL CHARACTER		ASCII (EVEN PARITY)	ASCII (ODD PARITY)	EBCDIC
SYNC	Synchronization character	96	16	32
NAK	Negative acknowledgement	95	15	3D
EOT	End-of-trans- mission	84	04	37
ENQ	Enquiry	05	85	2D
SCH	Start of header	81	01	01
STX	Start of text	82	02	02
ETX	End-of-text (end-of-message)	03	83	03
DLE	Data link escape (implies special meaning to the next character or characters)	90	10	10

2.2 BISYNC TRANSMISSION MODES

At any given time, a BISYNC transmission can be in one of the following three modes:

- control mode
- normal-text mode
- transparent-text mode

Each mode has a unique purpose and a defined set of conventions for BISYNC transmission. The rest of this chapter describes these conventions.

2.2.1 Control Mode

Following is a description of BISYNC transmission conventions in the control mode:

- A transmission always begins in the control mode.

- A transmission terminates from the control mode via:

  EOT
  ENQ
  NAK
  DLE "stick"

In the EBCDIC character set, "stick" represents any character in the range X'60' thru X'7F'. See Chapter 4.

In the ASCII character set, "stick" represents one of the four characters listed below. For more information on these characters, consult the IBM Binary Synchronous Communications Manual.

ASCII CHARACTER	ASCII CODE
0	30
1	31
;	3B
<	3C

- A transmission enters the normal-text mode from the control mode via:

  STX
  SOH

- A transmission enters the transparent-text mode from the control mode via:

  DLE,STX

Following are examples of transmissions using control mode only:

```
     TRANSMISSION                    MEANING

        SSSEP                 Line Bid
        YYYNA
        NNNQD

          S
       SSSDTP                 ACK0, ACK1, etc.
       YYYLIA
       NNNLCD
          K

       SSS        E           Data
       YYY'ABCDEF'O
       NNN        T

       SSSNP                  Negative Acknowledgement
       YYYAA
       NNNKD
```

2.2.2 Normal-Text Mode

The following describes the conventions of BISYNC transmission in the normal-text mode:

- Normal-text mode is a variation in which a block check character (BCC) is accumulated over an entire block of text. Special characters have meaning. Chapter 4 defines the BCC and the "stick" characters.

- Normal-text mode is entered (and BCC reset to zero) via:

  STX
  SOH

- The end of the particular block and, therefore, the indication that a BCC follows is via:

  ETX followed by BCC and PAD
  ETB followed by BCC and PAD
  ITB followed by BCC (remains in the text mode)

- Transparent-text mode can be entered from the normal-text mode via:

```
DLE,STX
```

Following are examples of transmissions using normal-text mode:

```
SSSS        E B P
YYYTABCDEFGT C A
NNNX        X C D

SSSS    S      E B P
YYY01234TABCD T C A
NNNH    X      B C D

SSSS    IB       I B      E B P
YYYTABCDTC EF GH T C IJKL T C A
NNNX    BC       B C      X C D
```

- A temporary text delay (TTD) is sent as:

```
STX,ENQ
```

2.2.3 Transparent-Text Mode

The following describes the conventions of BISYNC transmission in the transparent-text mode:

- Transparent-text mode, a variation of normal-text mode, allows all 256 8-bit patterns to be sent as data.

- It is entered from control mode or normal-text mode via the 2-character sequence:

```
DLE,STX
```

- Once in the transparent-text mode, all 8-bit characters on the line are treated as data, except for DLE. If a DLE occurs in transparent text, the meaning of the next character is modified as follows:

  - If the character following the DLE is another DLE, the sequence is interpreted as a single data character of DLE.

  - If the character following the DLE is a control character, the action is based on the specific character. Valid sequences are:

DLE,ETX	Terminates transmission when followed by BCC,PAD
DLE,ETB	Terminates transmission when followed by BCC,PAD
DLE,ENQ	Terminates transmission when followed by PAD
DLE,ITB	Continues in normal-text mode when followed by BCC

- When the BISYNC line driver sends special characters over an ASCII line in control mode or normal-text mode, these characters have appropriate (even or odd) parity; the driver also expects received special characters to have appropriate parity. In transparent-text mode, however, the driver sends special characters over the ASCII line without parity and expects received special characters to have no parity.

- When reading any character from an ASCII line, the BISYNC line driver checks and strips the parity bit, and places the stripped character into the line buffer. But on detecting a parity error with a character, the driver places that character--along with its parity bit--into the buffer and sets appropriate error status.

Following are examples of transmissions using transparent-text mode:

Data Buffer:

```
DS     D   I DE
LTABCDL5HSTGLT (transparent data block of 10 characters)
EX     E   B EX
```

BISYNC Line:

```
SSSDS    DD   I DEBP
YYYLTABCDLL5HSTGLTCA
NNNEX    EE   B EXCD
```

CHAPTER 3
BISYNC LINE DRIVER
FUNCTIONAL DESCRIPTION

3.1 SUPPORTED COMMANDS

The BISYNC line driver supports the following commands:

- XFER
- CXFER
- WAIT
- NOP
- EXAMINE
- RING WAIT
- ANSWER
- DISCONNECT
- READ,READ1,READ2
- PREPARE
- ANTI-PREPARE1
- ANTI-PREPARE2
- WRITE,WRITE1,WRITE2
- MODE Commands

3.2 XFER

This command obtains one data field specifying the address of the next driver command word (DCW). XFER can be used to alter the normal sequential execution of the DCW chain.

3.3 CXFER

This command obtains two data fields. The first data field specifies a fullword with two halfword masks. The first halfword is ANDed with the present ITAM status. The result is compared with the second halfword: if they are equal, a command transfer takes place; otherwise, the next command in sequence is executed. The second data field points to the command to execute for a command transfer.

NOTE

> The XFER and CXFER commands must be chained if they are to serve any useful purpose.

3.4 WAIT

This command obtains one data field specifying the address of a halfword with a timeout count. The timeout count is in units of 100 milliseconds (ms). A delay of this time period occurs before the command goes to completion. The precision interval clock ticks in units of 100 ms. So, depending on when the driver executes the WAIT command within any 100-ms unit, the actual wait period can be shorter than the specified wait period; specifically, the actual timeout is shorter than the specified timeout by a value less than 100 ms.

3.5 NOP

This command obtains one data field, which the driver ignores. NOP is useful for reserving space in the command chain and the data chain. The data field must be a valid address (i.e., within user program space).

3.6 EXAMINE

This command obtains one data area specifying the address of a device-status byte. The driver fetches the latest device status from this byte for the user task: if that byte is nonzero, its contents are returned to the user and are reset to zero; if the byte is zero, a sense status is performed on the device, and its present status is returned to the user.

NOTE

> The driver returns the status byte exactly as received from the adapter. It is the user's responsibility to understand what meaning any bit combinations may have at any given time. See Figure 3-1.

BIT NO.	0	1	2	3	4	5	6	7
201 STATUS	OVER-FLOW	PARITY	SYNC DETECT	RING	BUSY	EXAMINE	CARRIER OFF	DSNR
CSA STATUS	OVER-FLOW	PARITY	SPECIAL	RING	BUSY	EXAMINE	CARRIER OFF CL2S	DSNR

Figure 3-1 Synchronous Adapter Status Byte

3.7 RING WAIT

This command fetches no data fields. Interrupts from the adapter are enabled; however, the data-terminal-ready lead to the modem is not enabled. The command terminates when an interrupt is received with ring status set. If the DCW chain command bit is set, execution continues with the next command; otherwise, the driver terminates. If the DCW timeout bit is set, the command waits as long as the value specified in the write-timer halfword; when this interval expires, timeout error status is set. If the DCW timeout bit is not set, the command waits indefinitely.

3.8 ANSWER

For already connected nonswitched and switched lines this command terminates immediately. For unconnected dial-in lines the data-terminal-ready lead to the modem is enabled, causing the modem to answer the incoming call. When the data set indicates readiness for input/output (I/O), the command terminates. The DCW timeout and chain-command bits are handled as described for the RING WAIT command.

3.9 DISCONNECT

This command disables the data-terminal-ready lead to the modem, causing a disconnect on a switched line. The command then waits for one second and continues to the next command (when the DCW chain-command bit is set) or terminates (when the chain-command bit is reset).

3.10 READ, READ1, READ2

These commands transfer data from the synchronous adapter to main memory. READ1 and READ2 obtain one data field specifying the address of a writable 1- or 2-byte area into which data is read. READ obtains one or two data fields, depending on which one of the three standard ITAM buffer-management techniques are specified in the data code of the first field obtained. One data field is obtained for indirect text and chained buffers. Two fields are obtained for direct text.

The adapter is placed in SYNC-search mode and a read is performed. The first non-SYNC character is the first data character.

Special-character handling is designed to conform to the standard BISYNC line protocol. For a detailed description, refer to the flowcharts in Chapter 4.

A series of internal indicators are maintained to control the progress of the read operation. There are two possible modes of operation once data transfer begins: control mode and text mode. The operations are controlled by the text-mode indicator which, when set, indicates that the driver is in the text mode; when reset, in the control mode. Additionally, the text mode can be in normal text or transparent text, as indicated by the transparent-text indicator. When data transfer begins, after the character phase is established, the driver is placed in the control mode. In this mode, data characters (other than SYN) are stored in memory, without CRC accumulation, as they are received from the adapter. Another indicator is maintained to determine whether characters are to be converted from EBCDIC and stored in ASCII, or stored in EBCDIC as they are received. This indicator can be changed via the MODE TRANSL command, described in Section 3.13. Special characters affecting the operation of the driver in the control mode can be divided into two categories: termination sequences and text-mode initiation sequences.

3.10.1 Termination Sequences

Termination sequences are comprised of any one of the single characters ENQ, NAK, or EOT, or of any 2-character DLE "stick" character sequence. (A "stick" character is a character ranging from X'60' thru X'7F'.) After receiving a termination sequence, the driver reads one more character; if that character is not a valid pad character (i.e., not in the range of X'F0' thru X'FF'), the driver terminates with "bad pad" status. Receipt of ENQ or EOT is indicated to the user task by the SVC 15 status halfword (see Table 3-3). The pad character is not stored.

3.10.2 Text-Mode Initiation Sequences

The text-mode initiation sequences consist of three possible sequences: SOH (one character), STX (one character), or DLE STX (two characters). SOH and STX place the driver into the normal-text mode; DLE STX places the driver into the transparent-text mode.

In the normal-text mode, data characters other than SYN are stored in memory. Starting with the first character after STX or SOH, which initiated normal-text mode, the CRC-16 algorithm accumulates a BCC (block check character); SYN characters are not included in the BCC accumulation. The requirement for EBCDIC-to-ASCII conversion is based on the same indicator used in the control mode. Control characters in the normal-text mode are any of the single-character ITB, ETB, ETX or ENQ sequences or the 2-character DLE STX sequence.

ITB, ETB, or ETX causes the driver to read two additional characters. These characters are combined to form a 16-bit BCC, that is compared to the BCC accumulated over the message; comparison failure causes the driver to set BCC error status.

After ITB, the driver remains in text mode and restarts BCC accumulation.

Setting bit 2 in the link word of chained buffers causes the driver to store the next block into the next chained buffer.

After ETB, ETX, or ENQ, the driver

- reads an additional character after the intervening BCC (no BCC is read after ENQ),
- sets bad-pad status if the pad is invalid,
- resets text mode,
- and terminates.

Receipt of the DLE STX sequence causes the driver to enter the transparent-text mode. If this sequence is received in the control mode, BCC accumulation begins with the next character after the DLE STX. In the text mode, the received sequence is included with the BCC being accumulated. An additional indicator is maintained to determine whether EBCDIC-to-ASCII translation is required in the transparent-text mode. In the transparent-text mode, the only control characters recognized are the 2-character sequences starting with DLE. They are interpreted as follows:

- DLE DLE results in a single DLE character being included in the BCC accumultion and stored in memory.
- DLE SYN is discarded; however, the timer is reset to the read-error time value.
- DLE ITB resets the transparent-text mode, leaving the driver in the normal-text mode. ITB is included in the BCC and stored. The next two characters read are compared with the accumulated BCC, as described above; the BCC is reset; and BCC accumulation begins with the next character.
- DLE ETX and DLE ETB reset the transparent-text and the normal-text modes. ETX or ETB is included in the BCC and stored; a 2-byte BCC and a 1-byte pad are read and validated as described above. The command then terminates.
- DLE ENQ resets the transparent and normal-text modes. A pad is then read and validated as described above. The command terminates. In the SVC 15 status halfword the ENQ bit is set. See Table 3-1.

Any other sequence beginning with DLE resets the transparent and normal-text modes and results in driver termination with the error bit set in the SVC 15 status halfword (see Table 3-1) and with the termination code indicating a bad character sequence. See Table 3-2.

NOTE

> The DLE of all transparent-text terminating sequences is not stored in the buffer; e.g., for DLE ETX, only ETX is stored.

The read-error timer is started when any READ command, with timeout specified, starts. If the read expires before an SOH, ETX, DLE STX, or a block-terminating character is received, the command terminates with a timeout and transfer-not-begun status. The timer is restarted (reset to read-error time) when SOH or STX is received, and thereafter when a SYN (DLE SYN in the transparent-text mode) or a block-terminating character is received.

TABLE 3-1 SVC 15 STATUS HALFWORD FOR BISYNC DRIVER

BIT NO.	0	1	2	3	4	5	6	7
MEANING	ERROR	BSY	XFER NOT BEGUN	TIMEOUT	RESERVED	BCC ERROR	RESERVED	RESERVED

BIT NO.	8	9	10	11	12	13	14	15
MEANING	EOT	ENQ	ENCODED TERMINATION CODE (See Table 3-2)					

STATUS BITS	MEANING	DESCRIPTION
X'8000'	Error	Set for all error conditions. Certain conditions are not errors and do not set the error bit; e.g., special character detection.
X'4000'	Busy	The driver is still busy with SVC 15 call; can be cancelled via SVC 15 halt I/O.
X'2000'	XFER Not Begun	STX, SOH, or proper control-mode terminating sequence never occurred.
X'1000'	Timeout	Set if I/O timesout.
X'0400'	BCC Error	Indicates BCC (CRC or LRC) error.
X'0080'	EOT	EOT detected in data stream.
X'0040'	ENQ	ENQ detected in data stream.

TABLE 3-2 ENCODED TERMINATION CODES

ENCODED BITS	STATUS	MEANING
00	No Errors	No error.
05	Data Check	Terminated by data error; see bits 4 and 5.
06	Buffer Limit	Buffer limits reached without proper ending sequence.
07	Bad Pad	Pad character not received.
0A	Loss of Carrier	Lost carrier on reads.
0B	CL2S Error	Lost clear-to-send on write.
0C	Data Set not Ready	Data set not ready.
0D	Device Unavailable	Adapter not present.
0E	Overflow	Character overflow.
0F	Ring	Ring status detected during data transfer.
10	Buffer Overrun 1	BSY and/or DONE bits in chain buffers bad; may indicate priority too low.
11	NCE Overflow	Number of commands executed is greater than 255.
12	Task Queue Error	Task queue full, initialized, or absent during attempt to trap; transfer questionable.
13	Buffer Overrun 2	ESR did not execute in time; may indicate priority too low.
14	Timeout	Timeout.

TABLE 3-2 ENCODED TERMINATION CODES (Continued)

ENCODED BITS	STATUS	MEANING
15	Halt I/O	Halt I/O request aborted I/O.
16	Trans Block Error	Transparent block size error during write.
17	Bad Character Sequence	Improper BISYNC sequence.
18	Illegal Command	Command or modifier not valid.
19	Memory Fault 1	Memory fault referencing data.
1A	Memory Fault 2	Memory fault referencing buffer.
1B	Illegal LU	Logical unit illegal (not SVC 15, not assigned).
1C	Illogical Status	Device status not valid; may be hardware problem.
1D	Power Fail	Power failure.
1E	Illegal Action	Illegal software condition.
1F	Illegal Translation Table	Translaion table invalid.
23	Queue Empty	Queued buffer list empty.
24	Queue Overflow	Queued buffer list overflow.

3.11 PREPARE, ANTI-PREPARE1, ANTI-PREPARE2

For a PREPARE, the data field points to a byte containing a prepare character. The adapter is placed into the SYNC-search mode, and the first non-SYNC character received is compared with this prepare character. If these two characters are unequal, the adapter is again placed into the SYNC-search mode. This process continues until the command timeout (if specified) expires, or until the received character equals the prepare character. Once the two characters are equal, the command is considered done and the next command, if chained, is executed. The received character is discarded.

NOTE

> The special case of READ after PREPARE results in a lookahead to set up the READ. This allows the user to look for a special character and to read the following text without losing character synchronization. The received prepare character is not stored.

ANTI-PREPARE1 commands the BISYNC line driver to wait for a specified anti-prepare character; the data field associated with the ANTI-PREPARE1 points to a buffer containing this character. On receiving a character that matches the anti-prepare character, the driver places the adapter into the SYNC-search mode.

With the adapter in the SYNC-search mode, the driver then waits for a character that does not match the anti-prepare character. On receiving the first nonmatching character, the ANTI-PREPARE1 is satisfied: the driver stores the character into the line buffer and acts on the next command--which must be a READ-type command--to read the data following the anti-prepare character.

ANTI-PREPARE2 commands the line driver to wait for a character that does not match a specified anti-prepare character; the data field associated with the ANTI-PREPARE2 points to a buffer containing the anti-prepare character. On receiving the first non-matching character, the ANTI-PREPARE2 is satisfied: the driver stores the character into the line buffer and acts on the next command--which must be a READ-type command--to read the data following the non-matching character.

3.12 WRITE, WRITE1, WRITE2

These commands transfer data from main memory to the synchronous adapter. WRITE1 and WRITE2 obtain one data field specifying the address of a 1- or 2-byte area from which data is obtained. WRITE obtains one or two data fields, based on the ITAM buffer management criterion for READ.

These commands first establish character phase by transmitting a fixed number of SYN characters. The number sent can be modified via the MODE SYCT command described in Section 3.13. Characters are obtained from memory and, depending on the setting of the control/normal-text code indicator, are converted from ASCII to EBCDIC or transmitted as is (EBCDIC-to-EBCDIC). The characters are then transmitted to the data set adapter. Transfer begins in the control mode as it does with the READ commands. In control mode, the termination sequences ENQ, NAK, EOT, and DLE "stick" cause a pad character (X'FF') to be sent after the terminating character, causing the command to terminate.

The text-mode initiation sequences SOH, STX or DLE STX, described for the READ commands, place the driver into the normal or transparent-text mode.

In the normal-text mode, characters are obtained from memory and sent to the adapter; ASCII-to-EBCDIC conversion depends on the setting of the control/normal-text code indicator. The BCC is accumulated exactly as with the READ commands, and the same control characters are recognized: ITB, ETB, ETX, ENQ, and DLE STX.

The ITB, ETB or ETX sequence causes the driver to transmit the 16-bit BCC as accumulated. After ITB, the driver remains in text mode, resets the BCC, and restarts BCC accumulation. After ETB, ETX or ENQ, the driver transmits a pad character (X'FF') after the BCC for ETB or ETX, resets the text mode, and terminates. By setting bit 2 in the link word of chained buffers, text continuation after ITB is from the next buffer.

After the DLE STX sequence, the driver enters the transparent-text mode, with BCC accumulation and ASCII-to-EBCDIC conversion taking place under the conditions described for the READ commands. By specifying EBCDIC-to-EBCDIC conversion, binary data can be sent; however, all special characters must be in EBCDIC.

Termination sequences DLE ITB, DLE ETB, DLE ETX, and DLE ENQ are allowed. Their effects are the same as those described for ITB, ETB, ETX and ENQ. DLE is not included in the BCC accumulation. Any sequence other than these four will have unpredictable consequences.

For multiple block messages, a SYNC-idle sequence (DLE SYN in the transparent-text mode; otherwise, SYN SYN) is inserted between blocks. Ensure that the transmission time for a single block does not exceed one second; i.e., that block-size/line-rate does not exceed 0.125 bytes/baud.

Transparent text in BISYNC allows transmission of all 256 8-bit patterns. Transparent text terminates via DLE ETX, DLE ETB, or another appropriate DLE sequence.

Once in the transparent-text mode (via the DLE STX sequence) during writes, the driver has no way of knowing where to end the transparent-text block unless there is a prearranged block size. Therefore, when using transparent data blocks, indicate to the driver when to exit from transparent text. This tells the driver that a particular DLE is not transparent data (which would be sent down the line as DLE DLE), but is a valid transparent-text terminating sequence. There are two basic methods for notifying the driver that transparent text is to end:

1. Transparent text can be transmitted in fixed-size blocks. The size of transparent records is generally a defined constant on any particular link, and is defined as a MODE-changeable parameter; default is 80 bytes. The MODE TRECS command, described in Section 3.13, can define the size of the actual transparent data. This size does not include either of the 2-character sequences necessary to enter and exit transparent mode. See Figure 3-2.

2. Alternately, the user can place the 2-character terminating sequence as the last two characters of a buffer. Chained buffers must have bit 3 set in the flag field of the appropriate buffer.

	80 BYTES		80 BYTES
D S L T E X	TRANSPARENT DATA	D I S S S D S L T Y Y Y L T E B N N N E X	TRANSPARENT DATA D E L T E X

Figure 3-2 Transparent Record Size of 80 Bytes

3.13 MODE COMMANDS

The MODE commands give the user the option to modify certain control parameters of the BISYNC driver.

MODE commands take binary (hexadecimal) data from the user buffer and store this data within the DCB for later use by the driver. Since modification of DCB control fields can significantly impact the system, the user of a MODE command must be familiar with:

- the structure of any control field to be modified
- the use of the control field by the BISYNC driver
- the impact any modification might have on overall system operation

The use of MODE commands is optional. With some MODE commands, knowledge of the compatibility between the modified control field and the adapter capabilities or any special adapter-strapping requirements and modem characteristics might be useful. Default control-field values are provided within the DCB; normal SVC 15 I/O operations can be satisfactorily executed with these default values. Only the exceptional case requires a control field modification prior to I/O. Table 3-3 lists the MODE-changeable control fields and their default values.

TABLE 3-3 MODE CHANGEABLE CONTROL FIELDS

MODE COMMAND	COMMAND/ MODIFIER VALUE	DESCRIPTION	201 DEFAULT	QSA DEFAULT
TOUT	XX06'	Modify timeout periods	SEC READ SEC WRITE	6-SEC READ 5-SEC WRITE
CMD2	XX0E'	Modify Programmable Adapter Initialization Command	N/A	X'30'
RCMD	XX16'	Modify Read Command	X'49'	X'59'
WCMD	XX1E'	Modify Write Command	X'4A'	X'58'
RDIS	XX26'	Modify Read-Disable Command	X'89'	X'C9'
WDIS	XX2E'	Modify Write-Disable Command	X'89'	X'D9'
DISC	XX36'	Modify Line-Disconnect Command	X'81'	X'C1'
SYCT	XX3E'	Modify Count of Leading SYN Characters	4	4
TRNSL	XX46'	Modify Translation Table Use	All ASCII	All ASCII
TRECS	XX56'	Modify Transparent Record Size	80 bytes	80 bytes

TABLE 3-3 MODE CHANGEABLE CONTROL FIELDS (Continued)

TOUT	Set Timeout Values The data field points to the first of two halfwords. The first halfword contains an error timeout interval for read-type operations; the second, an error timeout interval for write-type operations. Both intervals are in units of one second.	|
CMD2	Set Programmable Adapter Options Applicable only to the QSA; the data field specifies the command byte to be sent to initialize the adapter to the correct data mode before any Read or Write operation.	
RCMD	Enable READ Command Whenever the driver enters the read mode, the data field points to a byte for the output command. Bit combinations should correspond to adapter strappings, modems, etc.	|
WCMD	Enable WRITE Command Whenever the driver enters the write mode, the data field points to a byte for the output command. Bit combinations should correspond with adapter strappings, modems, etc.	
RDIS	Disable READ Command The data field points to a byte for disabling all READ commands. This disable occurs at normal completion or on any error condition.	
WDIS	Disable WRITE Command The data field points to a byte for disabling all WRITE commands. This disable occurs at normal completion or on any error condition.	|
DISC	Disconnect Switched Line The data field points to a byte for the line disconnect command. This command is issued when an SVC 15 control command is received with disconnect modifier (X'0018') or when a final SVC 7 closes a communication line.	
SYCT	Set Number of Leading SYNC Characters The data field specifies a byte conaining a right-justified, single hexadecimal digit. This is the number, minus one (1), of the leading SYNC characters transmitted before each write. Its value ranges from X'0' through X'F'.	|

TABLE 3-3 MODE CHANGEABLE CONTROL FIELDS (Continued)

TRANSL	Set Translation Table Options The data field points to a byte specifying the type of translation for the control mode and the transparent-text mode. The first 4-bit hexadecimal digit of the data byte indicates translation for input; the second hexadecimal digit, for output. The driver supports only the EBCDIC character set on the line. The internal code can be ASCII or EBCDIC. Once a code is assigned for a communications line, that code remains in effect until the next MODE command is executed for that line. The read and write digits do not have to be equal. For downline loads or for transmission of transparent 8-bit binary data, EBCDIC transparent text must be used. This implies that only EBCDIC special characters (DLE, ETB, ETX, ITB, and SYNC) are recognized within the transparent block. Table 3-4 lists the possible translation options.
TRECS	Transparent-Record Size The data field points to a halfword containing the wanted transparent-record size. The halfword replaces the original value assembled into DCB.RECS.

TABLE 3-4 MODE CONTROLLABLE TRANSLATION OPTIONS

MODE	CONTROL CODE	DESCRIPTION
AA	0X	INPUT: control, normal-text, and transparent-text characters are converted from EBCDIC to ASCII.
	X0	OUTPUT: control, normal-text and transparent-text characters are converted from ASCII to EBCDIC.
AE	1X	INPUT: control and normal-text characters are converted from EBCDIC to ASCII. Transparent-text characters are not converted.
	X1	OUTPUT: control and normal-text characters are converted from ASCII to EBCDIC. Transparent-text characters are not converted.
EA	4X	INPUT: control and normal-text characters are not converted. Transparent-text characters are converted from EBCDIC to ASCII.
	X4	OUTPUT: control and normal-text characters are not converted. Transparent-text characters are converted from ASCII to EBCDIC.
EE	5X	INPUT: no conversion takes place.
	X5	OUTPUT: no conversion takes place.

3.13.1 Timeout Values

There are two halfwords (READ and WRITE) that specify values, in seconds, for error timeouts. When a command specifies timeout, this time value is placed into DCB.TOUT and is decremented every second by the system clock. If the particular command is not completed within the allotted time, the entire SVC 15 call is aborted, and the timeout status bit is set. If there are no other encoded errors, the timeout code is also placed into the encoded portion of the status. If the timeout status bit (X'1000') is set, and the encoded error is not timeout, the encoded error occurred first and might be the reason for the timeout.

There are separate time values for READ and WRITE. The data field of the MODE TOUT command specifies a fullword--the first halfword is the READ-error time value and the second halfword is the WRITE-error time value. Zero is not a valid time value. The actual timeout period can be shorter than that specified; specifically, the actual timeout is shorter than the specified timeout by a value less than 100 ms.

3.13.2 Adapter Commands

The BISYNC driver communicates with the synchronous adapter, 201 adapter or Quad Synchronous Adapter (QSA), through a series of I/O commands. To maintain flexibility, all hardware commands the driver issues are obtained from bytes reserved in the DCB. Each command byte can be modified by issuing an appropriate MODE command.

Figure 3-3 depicts the structure of the command bytes for the QSA and 201 Synchronous Adapters. Details concerning the use of these commands can be found in the M47-001 Synchronous Data Set Adapter Instruction Manual, (for the 201 Data Set Adapter) and the QSA Programming Specification Manual (for the QSA). Thoroughly review these publications before modifying any hardware commands. Of particular interest to the user of a QSA might be "local loop back" for back-to-back system testing.

201 ADAPTER COMMAND BYTE

BIT NO.	0	1	2	3	4	5	6	7
DESCRIPTION	DISABLE	ENABLE	PARITY MODE	SYNC SEARCH (READ ONLY)	DATA TERMINAL READY (DTR)	DISCONNECT	WRITE	READ

$\underbrace{\text{bits 0, 1}}$

00 = No change

01 = Enable interrupts

10 = Queue interrupts
but disable execution

11 = disarm device from
generating or queueing interrupts

QSA COMMAND BYTE

BIT NO.	0	1	2	3	4	5	6	7
DESC FOR RECEIVE MODE	DISABLE ENABLE SEE 201 ENCODING		LOCAL LOOP BACK	SYNC SEARCH	DTR	SPECIAL	0 REQUEST TO SEND	1 I/O MODE
DESC FOR TRANSMIT MODE	DISABLE ENABLE SEE 201 ENCODING		LOCAL LOOP BACK	RESET DATA MODE	DTR	SPECIAL	0 REQUEST TO SEND	1 I/O MODE
DESC FOR PROGRAMMABLE OPTIONS MODE			NUMBER OF DATA BITS SELECTED			LINE CONTROL MODE		0 PROG OPTION SET MODE

00= 5-Bit Char	00 = No Parity
01 = 6-Bit Char	01 = ZBID
10 = 7-Bit Char	10 = Odd Parity
11 = 8-Bit Char	11 = Even Parity

Figure 3-3 201 and QSA Command Bytes

3.14 DCW OPTIONS

The user can select four options by setting bits in the flag field of the DCW, as shown in Table 3-5.

TABLE 3-5 DCW (DRIVER COMMAND WORD) HALFWORD

BIT NO.	0	1	2	3	4 5 6 7	8 9 10 11 12	13 14 15
COMMAND HALFWORD	CC	CT	BT	TO	RESERVED (4 BITS)	MODIFIER (5 BITS)	COMMAND (3 BITS)

Bit 0	Chain command	If this bit is set, after execution of the current DCW command, the next DCW command in sequence is executed. Otherwise, the driver terminates.
Bit 1	Command trap	If set and enabled, this bit generates a command trap to the calling task before DCW execution.
Bit 2	Buffer trap	If set and enabled, this bit generates a buffer trap when the first character is transferred and after each buffer is transmitted or filled.
Bit 3	Timeout	If set, this bit initializes an error timer before DCW execution. If the timer expires before the command completes, the SVC 15 aborts with a timeout status. If reset, this bit stops the timer. The command does not timeout. There are separate error-time values for read and write.

3.15 SVC 15 FUNCTION CODE OPTIONS

The user can select four options by setting the appropriate bits in the function code of the SVC 15 parameter block, as shown in Table 3-6.

TABLE 3-6 FUNCTION BYTE

BIT NO.	0	1	2	3	4	5	6	7
FUNCTION BYTE	HI/O	CT	BT	TT	RESERVED			

Bit 0	Halt I/O	If set, this bit initiates a halt I/O sequence when the driver is currently connected to this task, or when a condition code (CC=1) is returned indicating that the driver is not connected to this task.
Bit 1	Command queue	If set, this bit causes command-queue entries to be made as specified in the DCW; if reset, no entries are made.
Bit 2	Buffer queue	If set, this bit causes buffer-queue entries to be made as specified in the DCW; if reset, no entries are made.
Bit 3	Termination	If set, this bit causes a queue entry to be made (and a trap to be taken if enabled in the TSW) when the driver terminates.

CHAPTER 4
BISYNC LINE DRIVER
PRINCIPLES OF OPERATION

4.1 INTRODUCTION

The BISYNC line driver uses CCB.BMOD to maintain various pointers. Only the low-order 8 bits, bits 8 through 15, are used. See Table 4-1.

TABLE 4-1 CCB.BMOD FLAGS

0 7	8	9	10	11	12	13	14	15
RESERVED	XFER NOT BEGUN	TEXT MODE	TRANS TEXT	END TRANS	NORMAL TEXT XLATE		TRANS TEXT XLATE	

Bit 8	XFER NOT BEGUN	If set, this bit indicates input has not begun. It is reset by: EOT if in control mode NAK if in control mode ENQ if in control mode SOH if in control mode STX if in control mode Any proper "stick"
Bit 9	TEXT MODE	If set, this bit indicates the line is not in control mode, but in normal-text or transparent-text mode as determined by bit 10.
Bit 10	TRANSPARENT TEXT	If set, this bit indicates the line is in transparent-text mode.
Bit 11	END TRANSPARENCY	If set, this bit indicates that the previous character was a DLE and that the line was in transparent-text mode at the time.

Bits 8 through 11 are flags indicating the mode for the line protocol; i.e., control, normal-text, or transparent-text mode. These bits are manipulated by the BISYNC special-character routines.

Bits 12 through 15 are translation table indicators for normal and transparent-text modes. These bits are assembled to indicate the translation required. DCB.XLT points to a table containing the addresses of several translation tables. At the beginning of any read or write, bits 12 and 13 are used as an index into the table; writes add 4 to this index. The address obtained is placed into CCB.XLT. For transparent-text mode, bits 14 and 15 are used as the index. Thus, by specifying different bit patterns for bits 12 through 15, different translations can be set up for reads and writes.

4.2 WRITE OPERATIONS

The WRITE command (WRITE, WRITE1, or WRITE2) performs these operations before branching to ITGETMOD:

- loads register 2 with the write-device number
- terminates call with "illegal instruction", if zero
- loads register C with the address of the write CCB
- initializes mode byte in CCB and sets CCB.XLT to the normal translation table
- loads register 7 with the address of the command table
- loads register 6 with the maximum modifier value (3)
- branches on register A to ITGETMOD

Address ITGETMOD branches to one of the three supported write routines: WRITE1, WRITE2, or WRITEB. WRITE2 adds 1 to the options register U4 and enters WRITE1. WRITE1 adds 1 to the options register and enters WRITEB.

Routine WRITEB calls ITGETBUF to completely set up the CCB for output with the appropriate buffer type. CCB.SUBA and CCB.CCW are now initialized. If the WRITE command is chained, a call to RAWCHR checks for read-after-write and, if necessary, sets up the read. The left digit of the SYNC counter byte is initialized at this time. The first driver ISR is now entered by a SINT.

The ISR puts the adapter into write mode using DCB.MOCW and waits until the status is satisfactory for output. SYN characters are written until the SYNC counter overflows. The executive bit is set; the subroutine address is changed to BWISR2; and a buffer trap is generated if requested in the command. The ISR then exits by a load PSW instruction, and the output proceeds under control of the auto-driver microcode. The BQSAINIT is entered to set QSA programmable options, if required.

AWISR2 is entered for one of two reasons:

- error-status interrupt
- buffer-limit interrupt

Error-status interrupts are handled by BWSTAT. This routine analyzes the device status and terminates the call with the appropriate status.

Buffer limit interrupts are handled by:

- changing CCB.SUBA to send two SYNC characters or, for transparent text, the DLE SYNC sequence
- calling ITXFRISR to schedule the next buffer, if available

A return from ITXFRISR indicates that no buffers are available and that a "buffer limit" error occurred. The buffer-select bit is complemented and the status is set to "buffer limit" by calling ITISSTAT.

After returning from ITISSTAT, the write is aborted by sending:

- EOT, if in control mode
- ENQ, if in normal-text mode
- DLE ENQ, if in transparent-text mode

After the above characters and any valid terminating characters (ETX or BCC) are sent, two pad characters are written.

The BISYNC protocol requires the first pad character and needs the second pad character to ensure that the first one clears the adapter. After the last character is accepted by the adapter, an "end buffer" routine is scheduled by a call to ITSRABS. The adapter is disabled using DCB.DOCW, and the ISPTAB is pointed to III, an OS subroutine that ignores interrupts. If read-after-write is pending, the RAW bit is reset, and the DCW command pointer is adjusted, along with the number of commands executed, by calling ITWR.RD. The read ISR is now entered to begin the READ.

If a read-after-write is not pending, calling ITSRABS schedules the driver ESR.

4.3 READ OPERATIONS

The beginning portion of a read operation is like that of a write. The read device number and the CCB address are loaded into registers; the modifier is fetched by branching to ITGETMOD using register UA. READ1 and READ2 add 1 and 2, respectively, to the options register and enter the normal code for READ BUFFER. ITGETBUF sets up the CCB just as in a write. The CCW is initialized to indicate write, translate, and ignore SYN detect. A check is then performed to see if this has been done on behalf of a read-after-write lookahead and, if so, control returns to the caller. Otherwise, CCB.SUBA and the ISPTAB entry are set up, and the driver READ ISR0 is entered via a SINT instruction.

The read ISR enables the adapter, placing it into read mode via DCB.MOCR. This command also places the adapter in the SYNC-search mode. All interrupts are then thrown away until a SYN arrives. The next character is checked for SYN and, if the character is a SYN, the execute bit is set; a buffer trap is scheduled; and input proceeds through the auto-driver channel. The BQSAINIT routine is entered to set QSA programmable options, if required.

The driver is now entered only for:

- error-status interrupts
- buffer-limit interrupts

For either interrupt, entry to the driver is at BRISR3.

Status errors are handled by BRISRSTA, which disables the adapter and returns the appropriate status. Buffer limits are again handled by calling ITXFRISR. If more buffers are available, the "next buffer" routine is scheduled; otherwise, ITXFRISR returns. If the subroutine returns, the I/O is aborted with a "buffer limit" error.

When reset and started by the proper special character routines, a CRC or LRC is accumulated during the read. An LRC is accumulated during normal text mode on an ASCII line; during transparent mode on an ASCII line, a CRC is accumulated. A CRC or LRC error results in "BCC or LRC error status"; however, if the CRC or LRC is followed by an ITB, the input continues until normal termination. An LRC is accumulated during normal text mode on an ASCII line; during transparent mode on an ASCII line, a CRC is accumulated.

The basic register convention for special-character handling in these routines is:

E0	PSW status
E1	PSW location
E2	Device number
E3	Device status/characters
E4	CCB address
E5	DCB address, if needed
E6	Mode flags
E7	Work

The routines that handle the special characters are written for general use. They are entered with the translated and untranslated characters in registers. Reads require the untranslated character for inclusion in the CRC or LRC; thus, E3 contains the untranslated character, while E2 is loaded with the translated character. This register loading does destroy the device number; however, reads do not need it.

Since the translated character is used for the CRC and since the device number is required to write it out to the line, write routines for special characters put the translated character in register 3.

In front of the translation tables is a list of special characters as they appear on the line. All routines that require checking characters must check for the value on the line. For example, all writes in EBCDIC must perform a software translation to find out if a special character is an STX.

4.4 PREPARE OPERATIONS

The routine for the PREPARE is like the read routine. It fetches the data field by calling ITGETDAT. If not called by a TET E-task, it relocates the address and fetches the prepare character. This character is stored in DCB.CHAR; a prepare-pending bit is set; and a read-after-write lookahead is performed by calling RAWCHKR. The read is then begun by branching into the read code at BRSTRT, and the normal-read ISR takes control. However, when detecting an SYN, CCB.SUBA is changed to the prepare ISR that ignores all further SYN characters. The first non-SYN character is translated and compared with the character in DCB.CHAR. If those characters are unequal, the adapter is again placed into SYNC-search mode, and the cycle repeats. If the non-SYN character is equal to the prepare character, the prepare is satisfied, and the prepare-pending bit is reset. If read-after-write (actually, a read-after-prepare) is pending, the execute bit is turned on, and the input continues as a read. Otherwise, the adapter is disabled and the driver ESR is scheduled.

The routines for the ANTI-PREPARE1 and ANI-PREPARE2 are functionally equivelent to the PREPARE routine, with the exception that the driver searches for non-matching characters. See Section 3.11 for a further description.

4.5 MODE OPERATIONS

Most of the MODE commands are handled by the standard ITAM MODE-command executors. The MODE TRANSL command is unique to the BISYNC driver. This modifier returns to BMTRNSL after obtaining the relocated address of a data area (a byte consisting of two hexadecimal digits). The leftmost hexadecimal digit replaces bits 12 through 15 of CCB.BMOD in the read CCB. The rightmost hexadecimal digit replaces the same bits in the write CCB.

4.6 CONTROL OPERATIONS

EXAMINE is handled by the standard ITAM EXAMINE command executor.

RING WAIT, ANSWER, and DISCONNECT are unique to the BISYNC driver because the required control bit and status bit are not compatible with other adapters. The logical flow is the same in all cases; however, the synchronous adapter provides the needed data-set-ready indication.

4.7 NULL OPERATIONS

NULL commands are handled by the standard ITAM NULL command executors.

4.8 TRANSLATION TABLES

The BISYNC line driver can be configured (via CUP/32) to work with one, two, or three sets of translation tables described belcw:

- Even-Parity ASCII Translation Tables
  - for writing: internal ASCII to even-parity ASCII
  - for reading: even-parity ASCII to internal ASCII
  - for writing and reading: 8-bit to 8-bit (any DLE sequences are written without parity and are expected to be received without parity)

- Odd-Parity ASCII Translation Tables
  - for writing: internal ASCII to odd-parity ASCII
  - for reading: odd-parity ASCII to internal ASCII
  - for writing and reading: 8-bit to 8-bit (any DLE sequences are written without parity and are expected to be received without parity)

- EBCDIC Translation Tables
  - for writing: ASCII-to-EBCDIC
  - for reading: EBCDIC-to-ASCII
  - for writing and reading: 8-bit to 8-bit (any DLE sequences are written with characters of the EBCDIC set; received DLE sequences are also expected to be EBCDIC)

Appendix C contains the values for the ASCII-to-EBCDIC and the EBCDIC-to-ASCII translation tables. As shown in these tables, the ASCII character set has 128 7-bit characters directly corresponding to 8-bit characters of the EBCDIC set. But to identify the other 127 characters of the 256-character EBCDIC set, the ASCII code must have the parity bit set.

An applications program must exercise caution when reading and writing data for a mix of EBCDIC and ASCII lines: the program must select only those 128 ASCII values that directly correspond to the EBCDIC characters. For example, suppose a task reads characters from an EBCDIC line and, via the EBCDIC-to-ASCII translation table, writes the translated characters over an ASCII line. For the 128 EBCDIC characters corresponding to the ASCII character set, the program encounters no difficulties in writing the translated ASCII characters over the line. But on receiving any EBCDIC character that does not correspond to the ASCII character set, the EBCDIC-to-ASCII translation table presents the program with an internal ASCII character with the parity bit set. When the program attempts to feed this character to the BISYNC line driver, which expects all parity bits reset, a parity error occurs.

4.9 PROGRAM INFORMATION

The ITAM/32 BISYNC Line Driver occupies approximately 5.5kb of system space that includes three 512-byte translation tables. Each line requires a DCB of 196 bytes, and 2 CCBs of 56 bytes.

Standard default block size for transparent data is 80 bytes.

Standard default for line code is EBCDIC; to SYSGEN a line for ASCII, refer to the ITAM/32 Reference Manual.

Refer to Appendix A for default values of all MODE parameters.

4.10 NOTES ON BISYNC LINE DRIVER

The BISYNC line driver is assembled to handle BISYNC lines that pass both ASCII and EBCDIC character sets.

In addition, the driver maintains separate indicators for transparent text as opposed to normal or control text. Thus, transparent text can be in EBCDIC (no translation), while control or normal text can go through the ASCII/EBCDIC translation. The option to allow ASCII-to-EBCDIC translation in transparent text is desirable because Perkin-Elmer machines are internally ASCII, whereas the line code is ASCII or EBCDIC. An ASCII-to-EBCDIC conversion is required to send alphanumeric and graphic data (ASCII in memory) as a transparent record (required by certain RJE protocols). This conversion translates ASCII graphics to the equivalent EBCDIC graphics. All other bit combinations translate to a unique 8-bit character. Thus, a one-to-one transformation exists between the ASCII and EBCDIC translation tables. A conversion of any 8-bit pattern from ASCII to EBCDIC and back to ASCII results in a null translation.

In the ASCII character set, a "stick" character is one of the four characters shown in Table 4-2. The BISYNC line driver writes all ASCII stick characters with appropriate (even or odd) parity and expects to receive ASCII stick characters with appropriate parity.

TABLE 4-2 ONLY STICK CHARACTERS FOR ASCII LINE

FUNCTION	ASCII LINE			INTERNAL ASCII	
		CODE (HEX)			
	GRAPHIC	EVEN PARITY	ODD PARITY	GRAPHIC	CODE (HEX)
ACKO	DLE,0	90,30	80,B0	DLE,0	10,30
ACK1	DLE,1	90,B1	80,31	DLE,1	10,31
WAK	DLE,;	90,BB	80,3B	DLE,;	10,3B
RVI	DLE,<	90,3C	80,BC	DLE,<	10,3C

In the EBCDIC character set, a "stick" character is any character having the value of X'60' through X'7F'. Table 4-3 shows the more frequently used stick characters for an EBCDIC line.

Table 4-4 lists all of the possible EBCDIC stick values and their corresponding internal ASCII equivalents.

TABLE 4-3 FREQUENTLY USED STICK CHARACTERS FOR EBCDIC LINE

FUNCTION	EBCDIC LINE		INTERNAL ASCII	
	GRAPHIC	CODE (HEX)	GRAPHIC	CODE (HEX)
ACK0	DLE,X'70'	10,70	DLE,X'F0'	10,F0
ACK1	DLE,/	10,61	DLE,/	10,2F
WAK	DLE,,	10,6B	DLE,,	10,2C
RVI	DLE,@	10,7C	DLE,@	10,40

Table 4-4 lists all of the possible EBCDIC stick values and their corresponding internal ASCII equivalents.

TABLE 4-4 EBCDIC STICK CHARACTERS WITH INTERNAL ASCII EQUIVALENTS

EBCDIC (LINE)	ASCII (INTERNAL)	EBCDIC (LINE)	ASCII (INTERNAL)
60	2D	70	F0
61	2F	71	F1
62	F2	72	F2
63	D3	73	F3
64	E4	74	F4
65	E5	75	F5
66	E6	76	F6
67	E7	77	F7
68	E8	78	F8
69	E9	79	60
6A	7C	7A	3A
6B	2C	7B	23
6C	25	7C	40
6D	5F	7D	27
6E	3F	7E	3D
6F	3F	7F	22

NOTE

Internal ASCII code for stick characters is different from the ASCII or EBCDIC line code. All tasks working with BISLNC protocol should be aware of this difference.

4.11 DEFINITIONS OF TERMS

- DCW CHAIN — Device Command Word Chain. Consists of sequential halfwords, each being one distinct command. ITAM drivers execute each command one at a time. The format is illustrated in Figure 4-1.

CC	CT	BT	TO	NOT USED BY DRIVER	MODIFIER (5 BITS)	COMMAND (3 BITS)

Figure 4-1 Device Command Word Chain Format

- "stick" — A special 2-character sequence used in BISYNC. The first character is always DLE. For an EBCDIC line, the second character must be in the range of X'60' through X'7F'. See Tables 4-3 and 4-4. For an ASCII line, the second character must be one of the four hexadecimal values shown in Table 4-2.

- BCC — Block Check Character. Used for error detection in BISYNC. Cyclic redundancy checking (CRC) is used on EBCDIC BISYNC lines. Longitudinal redundancy check (LRC) is used on ASCII BISYNC lines during a read or write of normal text; but a read or write of transparent text uses CRC.

CHAPTER 5
BISYNC TERMINAL MANAGER
PROGRAM IDENTIFICATION

5.1 INTRODUCTION

This chapter describes the ITAM/32 BISYNC Terminal Manager, Program Number 07-070F11. The ITAM/32 BISYNC Terminal Manager is part of the ITAM/32 Package.

This buffered BISYNC terminal manager supports medium speed (1200 to 9600 baud) controlled communications lines that use the IBM Binary Synchronous Communications Protocol for data transfer, error detection and error correction. Data can be formatted for the IBM 2780 or 3780 Remote Job Entry (RJE) terminals, or for a Perkin-Elmer processor-to-processor application using selected features of both the 2780 and 3780 systems.

Features of the BISYNC terminal manager include:

- SVC 1 device-independent access
- Fully automatic recovery following line transmission errors
- Built-in reduction of line-block transmission size during periods of excessive line errors
- Data record blocking/deblocking for the user task
- Space compression
- Horizontal tabulation
- Translation of selected printer-escape character sequences

Data transfer assumes ASCII internal code and EBCDIC line code; the BISYNC terminal manager does not support ASCII line code. | Limited conversational mode, automatic dial-out, and multipoint operation are not currently supported.

5.2 REQUIRED LINE DRIVER

The ITAM BISYNC terminal manager requires the ITAM/32 BISYNC Line Driver, Program Number 07-070F09.

5.3 REQUIRED HARDWARE

The ITAM BISYNC terminal manager requires the following minimal hardware:

- Synchronous Data Set Adapter 201 (201 DSA)
  Product Number M47-000

- Quad Synchronous Adapter (QSA)
  Product Number M47-002 or M47-003

- Western Electric 201 Modem, or equivalent

5.4 DATA SET ADAPTER STRAPPING REQUIREMENTS

- QSA

- Strap SYNC character to X'32' (EBCDIC SYNC).

- Leading SYNC-character suppression can be selected. System operates with or without this option.

- Special status bit must not be activated.

- 201 DSA

- Strap line to 2-wire or 4-wire, as required.

- Strap SYNC character to X'32' (EBCDIC SYNC).

- Strap character size to eight bits, no parity.

5.5 SUPPORTED MODES OF OPERATION

The ITAM/32 BISYNC Terminal Manager supports these operating modes:

- BISYNC IBM 3780 Remote Job Entry Data Format
  (OS/32 device codes 161 and 169)

- BISYNC IBM 2780 Remote Job Entry Data Format
  (OS/32 device codes 162 and 170)

- BISYNC Processor-to-Processor Interface
  (OS/32 device codes 163 and 171)

These supported operation modes are described in the following sections.

5.5.1 BISYNC IBM 3780 Remote Job Entry Data Format

This operational mode permits a user task to function like an IBM 3780 RJE terminal while transferring data to or from an IBM 360/370 or equivalent processor.

Standard format features for the user include horizontal tabulation, trailing space truncation, internal space compression and expansion, and data record blocking/deblocking.

5.5.2 BISYNC IBM 2780 Remote Job Entry Data Format

This operating mode permits a user task to function like an IBM 2780 RJE terminal while transferring data to or from an IBM 360/370 or equivalent processor.

Standard format features for the user include horizontal tabulations, trailing space truncation, and data record blocking/deblocking.

5.5.3 BISYNC Processor-to-Processor Interface

This operational mode permits a user task to transfer data to and from a second processor configured with ITAM BISYNC support or with other similar BISYNC support.

Standard format features for the user include horizontal tabulations, trailing space truncation, and data record blocking/deblocking.

5.6 EXTENDED DEVICE CODE

The ITAM extended device code (DCB.XDCD) must be specified at system configuration time for the Configuration Utility Program (CUP/32). Consult the OS/32 Program Configuration Manual for operation of this program.

Table 5-1 describes the ITAM extended device-code halfword and the options applicable to each device code.

When a processor-to-processor connection is made, one end facility must be designated as the master station, and the other as its slave. This technique prevents clashes when the two connected facilities simultaneously bid for line write initialization.

TABLE 5-1 EXTENDED DEVICE CODES*

0	1 2 3	4 5	6 7
MASTER SLAVE	RESERVED	LINE STATUS CODE	LINE PROTOCOL CODE

	EXTENDED CODE (DECIMAL VALUE)	DEVICE CODES		
		161/169	162/170	163/171
MASTER/SLAVE BIT				
MASTER	32768	NO	NO	YES
SLAVE	0	YES	YES	YES
LINE CODE				
LEASED LINE	1024	YES	YES	YES
MANUAL DIAL OUT	2048	YES	YES	YES
& AUTO DIAL IN				
LINE PROTOCOL				
HALF DUPLEX 4-WIRE	0	YES	YES	YES
HALF DUPLEX 2-WIRE	768	YES	YES	YES

* To formulate the extended device code, add the appropriate supported option within each of the three optional categories and use that value for the CUP extended-device option.

5.7 SUPPORTED ATTRIBUTES

The BISYNC terminal manager supports SVC 1 data-transfer requests and command requests for read, write, wait I/O, write/read binary, unconditional proceed, image, write filemark, and proceed I/O. Other data transfer requests return an error status; other command requests are ignored. The device-independent extended ITAM options for connect, disconnect, and format are supported. Device-dependent extended options include capabilities to transmit transparent text, to receive sequences for escape control characters, and to direct specific data buffering procedures.

The BISYNC terminal manager is buffered; it employs internal data transfers within a line control block (LCB) for I/O over the communications line. To facilitate this operation, SVC 7 allocate requests are required to structure an LCB within available system space and to permit later line assignments. Two internal buffers are generated within each LCB. Default sizes for the logical record length and buffer size are described in Table 5-2.

TABLE 5-2 DEFAULT SVC 7 ALLOCATION PARAMETERS

DEVICE CODES	LOGICAL RECORD LENGTH	BUFFER SIZE
161,169	80 bytes	516 bytes
162,170	80 bytes	404 bytes
163,171	80 bytes	516 bytes

5.8 SOURCE SYSGEN: HOW TO SUPPORT MPBSR

All 32-bit processors equipped with the high-speed communication options provide the capability to use the move and process byte-string register (MPBSR) instruction. The MPBSR instruction improves performance in moving characters between the user-task buffer (defined by the SVC 1 parameter block) and the terminal manager internal buffer (part of the LCB), resulting in substantial throughput improvement.

The high-speed option is selected for system generation by modifying the source-equate MOVEINST from "MOVEINST EQU 0" to "MOVEINST EQU 1." This equate statement is found within the first 20 lines of the BISYNC terminal manager program. Reassembly of the BISYNC terminal manager is then required, and the OS/32 library loader must merge the new terminal manager into the combined driver library before using this library as input to CUP/32. It is necessary only that the new terminal manager precede the supplied object version in the library and follow all appropriate DCBs. It is not necessary to delete the old version. See the ITAM/32 Packaging Information Document and the 32-Bit Loader Description Manual for further information on merging system libraries.

CHAPTER 6
BISYNC TERMINAL MANAGER
FUNCTIONAL DESCRIPTION

6.1 INTRODUCTION

The BISYNC terminal manager processes all SVC 1 requests from the user task by converting the SVC 1 functions into parameters processed by the SVC 15 BISYNC line driver. The BISYNC line driver accesses SVC 15 routines to drive the line hardware.

The terminal manager, without specific direction from the user task, processes all protocol-required acknowledgements and general line controls.

All reads and writes assume internal ASCII code for the processor and external EBCDIC code for the line.

To prevent loss of communications line control, the line is considered busy during an entire read or write sequence. This busy condition is established by maintaining a connection between the task TCB and the driver leaf during the entire I/O sequence, and by setting busy flags within the line control block (LCB) and data control block (DCB). Before I/O, the line is considered idle. When the first I/O request is received, a busy condition is established, and the appropriate read or write BISYNC line initialization sequence takes place. Subsequent I/O requests from other tasks are deferred pending line initialization. After line initialization, data reads or writes can be accomplished. The busy condition terminates when an EOT character (EOF condition on read), a write filemark, a write with the extended-option transmission bit set (see Section 6.9), an SVC 7 close, or certain error conditions (see Section 6.7) is received.

Since BSC is a half-duplex line protocol, the line stays busy during a single read-only or write-only transmission.

Any read attempt during a write sequence or any write attempt during a read sequence results in an error. The one exception to this is FORMAT READ, described in Section 6.3.

There are two BISYNC control character sequences: transparent and nontransparent. To prevent control errors (as opposed to text errors) during data transfers, certain control characters are not permitted in the text of a nontransparent data transfer. Transparent text, however, uses a data link escape (DLE) character to signify entrance or exit from a control character state, and thus permits any 8-bit character sequence within the data field. To facilitate proper DLE insertion, all transparent writes must contain a record size equal to the logical record length specified during line allocation. Due to different means of control, interlacing transparent and nontransparent write requests is illegal. Failure to follow these procedures results in the return of an illegal function-code error to the user task.

6.2 FORMAT WRITE

When a write request is received, an idle line becomes busy and the terminal manager initializes the line. The standard terminal-manager line initialization procedure consists of transmitting an ENQ character on the line and receiving the expected ACK0 response. Failure to receive the correct response (or a line error) results in a predetermined number of retries and a final error return to the user task.

After successful line initialization, user-task records are buffered into dynamic output buffers processed by the terminal manager. Formatting and buffering considerations are described in Section 6.10.

6.3 FORMAT READ

When a read request is received, an idle line becomes busy and the terminal manager initializes the line. The standard terminal-manager line initialization procedure consists of an SVC 15 PREPARE that looks for a received ENQ. A line error results in a predetermined number of retries and a final error return to the user task.

Because no timeout is provided during the PREPARE, it is possible to remain in a read-initialization sequence for an indefinite period. If the line is in a read-initial PREPARE sequence, an ENQ character was not received. If a write request is received, the read-initialization sequence is halted, and a write-initialization sequence is initiated. Similarly, an SVC 1 halt I/O can be used to halt a read-initialization sequence.

These capabilities permit a user task anticipating input or output to constantly leave a read-initialization sequence on the line to accept possible input, and yet be capable of breaking that sequence should an output be desired.

After successful line initialization, records from dynamically controlled input buffers are moved into the user read buffer on each successive read.

6.4 IMAGE READ/WRITE

These requests are processed with the user-program buffer for actual I/O. Line initialization procedures are identical to those of format read and format write.

In image-write mode, the user program is responsible for providing correct BISYNC control-character sequences to frame output data. Detailed descriptions of BISYNC control-character sequences can be found in General Information -- Binary Synchronous Communication, IBM Publication Number GA27-3004-2. A cursory review of BISYNC control characters is provided in Section 6.13.

6.5 WRITE FILEMARK

This command directs the terminal manager to transmit an EOT character. For image write, this causes immediate scheduling of the single character write. For format writes the request is similar to a write request with the extended option bit transmission (bit 17) set.

6.6 WRITE/READ BINARY

Write or read binary is supported. For processor-to-processor operation (device codes 163 and 171), data is transmitted as transparent text, processed as ASCII data without any format controls, converted to EBCDIC on output to the line, and reconverted to ASCII input to the remote terminal. The remote terminal should be a similarly equipped Perkin-Elmer processor. For 2780 or 3780 operation (device codes 161, 162, 169, and 170), data is processed as binary transparent text and not converted on input or output.

6.7 STATUS CODE PROCESSING

The BISYNC terminal manager uses the standard SVC 1 device-independent status codes as well as extended status information provided for the user program and placed in the device-dependent status field of the SVC 1 parameter block. This allows the user program to employ error-analysis routines identical to those for non-ITAM SVC 1 status analysis. For programs requiring further information, the user program can employ extended BISYNC status analysis.

Table 6-1 illustrates the device-independent SVC 1 status codes used by the BISYNC terminal manager.

TABLE 6-1 SVC 1 DEVICE-INDEPENDENT STATUS CODES

HEXADECIMAL CODE	TYPE STATUS
X'00'	Successful completion
X'C0'	Illegal function
X'A0'	Device unavailable
X'88'	End-of-file
X'84'	Unrecoverable error
X'81'	Illegal or unassigned
X'82'	Recoverable error (RVI)

A busy condition on a line terminates whenever the terminal manager returns an end-of-file or unrecoverable error status to the user task. No other type status affects a busy line condition.

A recoverable error status is returned only after a reverse interrupt (RVI) on an SVC 1 write is received. Similar to a break on a teletype, RVI denotes successful transmission of data to a remote facility with the added indication that the remote facility wishes to terminate transmission, to reverse data-transmission direction, and to send a priority message. When a user task receives an RVI, it can either reverse the line or ignore the RVI by retrying the errored request. To reverse the line, the user task must issue a Write Filemark command to cease writing, or write a final data record with the extended-option transmission bit (bit 17) and issue an SVC 1 read request.

NOTE

> No other recoverable error status returns are provided by the SVC 1 BISYNC terminal manager. The terminal manager automatically retries the normally recoverable errors according to BISYNC protocol conventions. Only if repeated retries fail is an unrecoverable status returned to the user program.

6.8 DEVICE-DEPENDENT STATUS CODES

Table 6-2 lists the device-dependent status codes returned to the SVC 1 parameter block.

The user program can ignore device-dependent statuses and depend solely on the standard device-independent status, or further analyze the extended status byte. Since the extended status is device-dependent on the BISYNC terminal manager, common routines that also analyze the extended status byte for other drivers may not be practical.

TABLE 6-2 ITAM/32 BISYNC DEVICE-DEPENDENT STATUS CODES

DEVICE-INDEPENDENT HEXADECIMAL CODE	EXTENDED STATUS CODES	TYPE STATUS (EXTENDED)
X'C0'	01	Read request during writes or write request during reads
X'84'	02	Line error
X'84'	03	Hardware error
X'84'	04	Message forward abort
X'84'	05	Message reject
X'84'	06	Halt I/O request
X'84'	07	Read image buffer too small
X'84'	08	Invalid image control characters
X'84'	09	Communications protocol error

Code 01 (Read Request during Writes
or Write Request during Reads)

Provided as an extension to the standard legal function code, code 01 indicates that a write request was attempted while the line was set busy with a read sequence, or vice versa.

Code 02 (Line Error)

Provided as an extension to the standard unrecoverable error status, code 02 indicates that garbled data or an excessive number of negative acknowledgements are being received, or that there is a similar problem with the communications facility.

Code 03 (Hardware Error)

Provided as an extension to the standard unrecoverable error status, code 03 indicates loss of carrier, a possible inoperative adapter or modem, or a similar hardware problem.

Code 04 (Message Forward Abort)

Provided as an extension to the standard unrecoverable error status, code 04 indicates that the remote facility terminated message transmission before the normal end-of-message.

Code 05 (Message Reject)

Provided as an extension to the standard unrecoverable error status, code 05 indicates that the remote facility terminated receipt of message.

Code 06 (Halt I/O Request)

Provided as an extension to the standard unrecoverable error status, code 06 indicates that the user task terminated an outstanding I/O request. This error status terminates a read request -- still in the initialization phase -- in favor of a subsequent write request.

Code 07 (Read Image Buffer too Small)

Provided as an extended status to the standard unrecoverable error status, code 07 indicates that a read buffer was too small, preventing successful data input. This status might occur if an image read was attempted or if a format buffer generated during line allocation was too small.

Code 08 (Invalid Image Control Characters)

Provided as an extended status to the standard unrecoverable error status, code 08 indicates that a mismatch between sending and receiving protocols prevented proper data passage. This problem results from a connection with an untested BISYNC facility or from an unusual line condition.

Code 09 (Communications Protocol Error)

Provided as an extended status to the standard unrecoverable error status, code 09 indicates that a mismatch between sending and receiving site protocol prevented proper passage of data traffic. This could result from a connection with an untested BISYNC facility or from a very unusual line condition.

Code 00 (All Other Errors)

All errors for which no specific extended status is reserved receive an extended status of 00. For example, a timeout for nonreceipt of data from a remote terminal receives a code of 00.

6.9 ITAM EXTENDED OPTIONS

The ITAM extended options are connect, disconnect, format, transparent, transmission, message, block, escape, receipt, and binary transparent. The following paragraphs describe each of these options:

- Connect - Bit 0 (the high-order or most-significant bit)

  Used as an extended option for a read or write during the line-initialization sequence (see Section 6.3). Setting this bit directs the terminal manager to answer a telephone ring on a dial-in line.

- Disconnect - Bit 1

  Used as an extended option for a read or write. Setting this bit directs the terminal manager to disconnect a switched line following final data transfer.

- Format - Bit 2

  Used as an extended option for a read or write. Setting this bit directs the terminal manager to perform its normal record buffering, to insert or delete line control characters, and to recognize appropriate data-format control characters. Resetting this bit corresponds to SVC 1 image I/O.

- Transparent - Bit 16

  Used as an extended option for a format write. Setting this bit directs the terminal manager to use transparent-text line control characters in message formatting. Transparent data is converted from ASCII to EBCDIC on output, and from EBCDIC to ASCII on input.

- Transmission - Bit 17

  Used as an extended option for a format or image-write. Setting this bit directs the terminal manager to take all actions normally taken when the message bit (bit 18) is set.

In addition, an EOT character is transmitted after the final message block. The terminal manager idles the line after successful request completion. When the transmission and message bits are used together, the transmission bit has precedence over the message bit which, in turn, has precedence over the block bit.

- Message - Bit 18

  Used as an extended option for a format-write. Setting this bit directs the terminal manager to transmit all existing buffered records and to place an ETX as the terminating character of the final record. Successful completion is indicated when the final message block is transmitted and properly acknowledged by the receiving facility. Therefore, setting this bit does not cause a line to be idled. The bit is usually set to distinguish each message among a series of messages sent on a single transmission.

- Block - Bit 19

  Used as an extended option for a format write. Setting this bit directs the terminal manager to complete its current buffer with the provided record, to place an ETB character as the terminating character of that record, and to transmit the buffered block.

- Escape Receipt - Bit 20

  Used as an extended option of a format read. Normally, the BISYNC (escape) ESC character is removed from a received record and translated into an appropriate series of ASCII characters if possible. Setting this bit allows the ESC character sequence to be passed to the user buffer.

- Binary Transparent - Bit 21

  Used as an extended option for format read or write. On writes, setting this bit directs the terminal manager to use transparent-text line control characters in message formatting and to inhibit ASCII-to-EBCDIC conversion of transparent text during output. On reads, setting this bit directs the terminal manager to inhibit EBCDIC-to-ASCII conversion of transparent text during input; conversion of received normal text still takes place. When the binary transparent and the transparent bits are used together, the binary transparent bit takes precedence over the transparent bit.

6.9.1 Default Extended Options

Table 6-3 shows the extended options the terminal manager defaults to when no options are selected from the SVC 1 ITAM extended-options fullword.

TABLE 6-3 DEFAULT EXTENDED OPTIONS

FUNCTION	DEVICE CODE	DEFAULT EXTENDED OPTIONS*
ASCII read/write	161/169	Format, escape receipt
ASCII read/write	162/170	Format, escape receipt
ASCII read/write	163/171	Format
Binary read/write	161/169	Format, escape receipt, binary transparent
Binary read/write	162/170	Format, escape receipt, binary transparent
Binary read/write	163/171	Format, transparent

*Format = bit 2; escape receipt = bit 20;
binary transparent = bit 21; transparent = bit 16.

6.10 FORMAT AND LINE CONTROL

The BISYNC terminal manager has format capabilities for various device-code read/write combinations. As an assembly option, it is possible to modify processor-to-processor capabilities for system needs. The following sections describe each of the available format capabilities.

6.11 FORMAT CAPABILITIES

6.11.1 Space Suppression

Space suppression processes the ASCII group separator (GS) character as a substitute for an internal string consisting of 2 to 63 sequential spaces. On output, when two or more spaces are found within the text, a 2-character sequence consisting of the GS character plus an encoded space count is substituted for the space string. On input, when a GS character is found, the subsequent encoded space-count character is checked, and the appropriate number of spaces is inserted.

6.11.2 Trailing Space Suppression

Trailing space suppression truncates trailing space characters in an output record and replaces them with an appropriate record-termination character (ETB, ETX, ITB, or EM).

6.11.3 Horizontal Tabulation

Horizontal tabulation processes the horizontal tabulation (HT) character as a tab indicator for input records. To set tabulation, the first record of an incoming message must be flagged with a starting ESC character, followed by an HT character. Subsequent character positions should contain an HT wherever a horizontal tab is needed. Horizontal-tab analysis is terminated when a new line (NL) character is received. All other character input is ignored. The terminal manager develops a horizontal-tabulation bit map from this record and skips to the next record. For subsequent records, a received HT directs the user record-character input pointer to be updated, as indicated in the bit map. Intervening positions are filled with spaces.

6.11.4 Escape Character Recognition

Escape-character recognition recognizes the ASCII ESC character as the first character of a record. If the escape-receipt bit (bit 20) is set in the ITAM extended options, the 2-character ESC sequence is moved into the record. Otherwise, the ESC sequence is converted to an equivalent ASCII control character, if possible, or it is bypassed.

For ESC character sequences converted to equivalent ASCII character sequences, the escape-printer control sequence is provided after the record containing the escape control. This directs the printer-control sequence, a vertical tab for example, to be executed after the record to be printed. The actual data record and control sequence are passed to the user program through two sequential SVC 1 reads. The data record is provided for the first read. The ASCII printer control sequence is provided for the second read. See Table 6-4 for a description of escape character sequences and their equivalent ASCII control characters.

TABLE 6-4 ESCAPE CHARACTER SEQUENCES

	CODE SEQUENCE	MEANING	SECOND RECORD ASCII CONVERSION
3	ESC	SINGLE SPACE	N/A
7	ESC S	DOUBLE SPACE	SPACE,CR
8	ESC T	TRIPLE SPACE	LF,CR
0	ESC A	SKIP TO TRACK 1	FF,CR
	ESC B	2	VT,CR
E	ESC C	3	N/A
M	ESC D	4	N/A
U	ESC E	5	N/A
L	ESC F	6	N/A
A	ESC G	7	N/A
T	ESC H	8	N/A
I	ESC I	9	N/A
O	ESC J	10	N/A
N	ESC K	SKIP TO TRACK 11	N/A
	ESC L	SKIP TO TRACK 12	N/A
	ESC M	SPACE SUPPRESS	N/A

	CODE SEQUENCE	MEANING	SECOND RECORD ASCII CONVERSION
2	ESC	SINGLE SPACE	N/A
7	ESC S	DOUBLE SPACE	SPACE,CR
8	ESC T	TRIPLE SPACE	LF,CR
0	A	SKIP TO TRACK 1	FF,CR
	B	2	VT,CR
E	C	3	N/A
M	D	4	N/A
U	E	5	N/A
L	F	6	N/A
A	G	7	N/A
T	ESC H	SKIP TO TRACK 8	N/A
I	ESC 4	PUNCH SELECTION	N/A
O			
N			

6.11.5 Carriage Return

The carriage return (CR) terminates a record at the recognition of an ASCII CR character. For reads, all short blocks are terminated when a CR is inserted in the user buffer.

6.11.6 End-of-Medium

End-of-medium terminates a record when an ASCII EM character is recognized. The EM character is processed the same as a CR on write. For read, a CR is passed to the user buffer in lieu of an EM character.

6.11.7 Unit Separator

Unit separator inserts or deletes the ASCII US character as a record separator within a block. This insertion allows more than one record to be included within a transmission block.

6.11.8 Transparent Text

Transparent text allows transparent-text control characters to be recognized and inserted for data buffering. The following is implied by the use of transparent text:

- Any 8-bit character can be provided by the user program within a data record. Data transmission control is totally transparent to data content.
- All data records must have a record length equal to the record length specified at line allocation time.
- The terminal manager does not recognize escape characters, horizontal tabulation, trailing space suppression, or space compression.
- Once specified for a data message, transparent text is assumed for all message transmission or reception. If transparent text is not specified for the first record of a message, it cannot be specified later.
- Cn reads, control characters within the received data determine transparent-text processing.
- Multiple-record block buffering is provided.

6.12 RECORD BUFFERING

Record buffering is provided for all format read/writes. Image read/writes have no record buffering. Following is a definition of buffering terms:

- Record

  User data contained in a single read or write request usually consist of a single 80-character card image or a single 80- to 140-character print line. LRECL specifies maximum record size at allocation time.

- Block

  A block consists of one or more records buffered for a single BISYNC line communication. Each record is expanded to include BISYNC record separators and block delimiters. Each block is terminated by an ASCII ETB or ETX character. BSIZE specifies maximum block size at line allocation time.

- Message

  A message consists of one or more blocks transmitted in a BISYNC line communication. An ETX character terminates the last message block.

- Transmission

  A transmission consists of one or more messages transmitted in BISYNC line communication. An ASCII EOT character normally terminates a transmission.

6.12.1 Transmission, Message, Block and Record Relationships

In BISYNC, several records can be combined into a single block. Blocks can then be combined into a single logical message, several of which make up a physical transmission.

The terminal manager buffers user records into a block. The number of records buffered into each block depends on block size and protocol characteristics. If the user sets the extended-option block bit (bit 19), buffering of that block ceases, and the block is transmitted. If the user sets the extended-option message bit (bit 18), buffering of that block ceases; the protocol-dependent end-of-message indicator is set in the block, and the block is transmitted. If the user sets the extended-option transmission bit (bit 17), all message-termination actions, as above, are taken, and a protocol-dependent end-of-transmission indicator is also transmitted. An SVC 1 write filemark also directs transmission of all buffered records and an end-of-message indicator in a manner similar to setting the extended-option transmission bit within a write request.

Dynamic line buffers are provided for record buffering at SVC 7 line allocation time. All pointers to these blocks, along with the counters and internal indicators to control them, are provided within the LCB. Only a single set of these buffers is required for input or output.

For writes, when a record is received from the user program, it is first formatted or compressed in accordance with device conventions and then moved into the next available position within the active output block buffer. The terminal manager determines whether to hold the block buffer to receive subsequent writes and immediately return to the user or to queue it for write. A transmission is scheduled if the remaining space in the block prevents the insertion of a subsequent record or if the number of records within the block reaches a predefined point.

The 3780 protocol subset, for example, permits only seven records to be inserted within a block. Setting the extended-option transmission bit, message bit, or block bit also causes transmission scheduling. When a transmission is scheduled, immediate return to the user takes place if a second internal buffer block is available to receive future writes. If a second block is not available because both blocks are currently busy with previously scheduled transmission requests, return to user is delayed until a previous transmission is completed.

For reads, the terminal manager attempts to read ahead into available input block buffers. When a read request is received from the user program, the terminal manager determines whether an active input deblocking buffer is available. If one is not available, processing delays until a successful read completion makes such a buffer available. Otherwise, the next sequential record is removed from the deblocking buffer, formatted or expanded as required, and moved to the user-program read buffer. Return to the user then takes place. When a deblocking buffer is emptied, it is flagged as available for additional reads.

ITAM extended-option settings for block, message, or transmission bits, or the receipt of a read ETX character takes precedence over normal record buffering. The user program might be behind the terminal manager on reads, or ahead of it on writes.

6.13 USE OF CONTROL CHARACTERS FOR DATA FRAMING

Before each transmitted block, the line driver includes line synchronization characters. It also terminates each block or US character with a cycle redundancy check (CRC) block-validation sequence. No CRC is provided after an RS character.

Any character can be included within transparent text since only those control characters preceded by a DLE are recognized. The line driver modifies each DLE character within the text to a DLE sequence.

Within nontransparent text, only the GS, ESC, HT, EM, CR, FF, and VT control characters can be included in the record text for their defined purpose. All other control characters, listed in Table 6-5, cannot be in data text.

TABLE 6-5 BISYNC CONTROL CHARACTERS

CHARACTER	STANDARD ABBREVIATION	HEX ASCII	HEX EBCDIC
Start of header	SOH	01	01
Start of text	STX	02	02
Enquire	ENQ	05	2D
Data link escape	DLE	10	10
Negative acknowledgement	NAK	15	3D
End-of-transmission block	ETB	17	26
End-of-text	ETX	03	03
Intermediate transmission block	ITB	(See US)	
Unit separator	US or IUS	1F	1F
Record separator	RS or IRS	1E	1E
Line sync	SYN	16	32
End of medium	EM	19	19
Horizontal tab	HT	09	05
End-of-transmission	EOT	C4	37
Escape	ESC	1B	27
Generate spaces	GS	1D	1D
Carriage return	CR	0D	0D
Form feed	FF	0C	0C
Vertical tab	VT	0B	0B

6.14 TRANSPARENT VS BINARY TRANSPARENT TEXT

There are subtle differences between transparent text and binary transparent text. In both cases, identical control-character sequences permit any binary character to be transmitted; all format characteristics apply. However, user-data translation procedures are different.

6.14.1 Transparent Text

Data is converted from ASCII to EBCDIC on output and from EBCDIC to ASCII on input. Complete 256-character ASCII and EBCDIC translation tables provide a one-to-one correspondence between each ASCII character and its equivalent EBCDIC character. Special 8-bit ASCII characters above 127 have been invented to facilitate translation of the complete EBCDIC set into ASCII. See Appendix C. If transmission is between two Perkin-Elmer ITAM facilities, data is translated from ASCII to EBCDIC on output and back to original ASCII on input. For Perkin-Elmer to Perkin-Elmer communication, data translation is transparent to the user. If the communication is between a Perkin-Elmer ITAM facility and an EBCDIC facility such as an IBM System 360 or 370, a translated data record is received. This translation is useful if it is necessary to transmit the graphic character set to an EBCDIC facility, with the understanding that the data is interpreted as the graphic alphanumeric or special character EBCDIC set. If the data transmitted to the EBCDIC facility were binary, this conversion could be detrimental. One solution would be to retranslate the data back to normal state within the user task. A second solution would be to use the binary-transparent format mode described below. Transparent text mode is the default for binary writes using device code 163 or 171.

6.14.2 Binary Transparent Text

No translation of binary transparent data takes place on input or ouput. This feature is most useful for the passage of binary data to an EBCDIC-oriented facility. It is the default for binary writes with device codes 161, 162, 169, or 170.

Because of different buffering techniques, interlacing image and format I/O requests is illegal. A violation results in an illegal function error status.

Multi-record buffering is provided for transparent I/O. To allow for this feature, interlacing transparent and nontransparent I/O requests is illegal. Also, all transparent record buffering is performed with a constant size for all user records. This constant size is specified during line allocation. Improper size for a transparent buffer results in an illegal function error status.

The LCB internal buffers are not used for SVC 1 image reads or writes. Hence, if only image I/O is being performed, minimum-sized internal buffers should be specified during line allocation.

CHAPTER 7
BISYNC TERMINAL MANAGER
PROGRAM INFORMATION

7.1 INTRODUCTION

The ITAM BISYNC Terminal Manager occupies 8000 bytes; the device control block (DCB) and channel control block (CCB), 308 bytes/line; the line control block (LCB), 176 bytes/line. The LCB, in addition to the 176 bytes/line, must contain two line buffers. The LCB receives support from the BISYNC SVC 15 line driver and the common ITAM module. For line allocation, 516-byte buffers are required for 3780 emulation (device codes 161,169); 404-byte buffers are required for 2780 emulation (device codes 162,170). The size of buffers for processor-to-processor (device codes 163,171) data depends on implementation. For device codes 163 and 171, both ends of a processor-to-processor link must use buffers of the same size. To determine the best size, calculate the maximum size of each data record; add 5 control characters for nontransparent text or 7 control characters for transparent text; multiply that figure by the number of records to be included within each transmission block; and add the 8 bytes required by terminal-manager overhead.

7.2 PRINCIPLES OF OPERATION

7.2.1 Table Structure and Defined Constants

Refer to the ITAM Reference Manual for a description of the LCB, DCB, and block descriptor as well as the SVC 15 commands, command modifiers, data buffer control, and error status.

Whether or not transparent text is used, device-code indices are:

INDEX	DEVICE CODE
0	163 or 169
2	163 or 169 (transparent)
4	161 or 170
6	161 or 170 (transparent)
8	162 or 171
10	162 or 171 (transparent)

Read and write format options are coded in a table accessed by the device-code index. Each halfword is generated by adding the appropriate buffer-format options in the following list:

```
BFO.SSM    EQU   X'8000'   Internal space suppression
BFO.SSB    EQU   0

BFO.TSM    EQU   X'4000'   Trailing space suppression
BFO.TSB    EQU   1

BFO.HTM    EQU   X'2000'   Horizontal tab
BFO.HTB    EQU   2

BFO.EMM    EQU   X'1000'   End-of-medium character
BFO.EMB    EQU   3

BFO.CRM    EQU   X'0800'   Carriage return character
BFO.CRB    EQU   4

BFO.USM    EQU   X'0400'   US character as ITB
BFO.USB    EQU   5

BFO.RSM    EQU   X'0200'   RS character as ITB
BFO.TSB    EQU   6

BFO.ESCM   EQU   X'0100'   Process ESC character
BFO.ESCB   EQU   7

BFO.HTSM   EQU   X'0080'   Initialize HT map
BFO.HTSB   EQU   8

BFO.NLM    EQU   X'0040'   Recognize new line character
BFO.NLB    EQU   9
```

Certain special characters and special-character sequences are converted into special-character indices by the SPC.CHK subroutine for simplified mainline character determination. Special-character indices are:

INDEX	SPECIAL CHARACTER/CHARACTER-SEQUENCE
1	ACK0
2	ACK1
3	WACK
4	RVI
5	NAK
6	STX, DLE STX
7	SOH, DLE SOH
8	ETB, DLE ETB
9	ETX, DLE ETX
10	US
11	EOT, DLE EOT
12	ENQ
13	TTD

An error-control table (UERTAB) determines whether an error is recoverable and what status and extended status to return to the user task. One halfword is generated within the table for each possible error listed in the ITAM error table (ITE.TAB). Each halfword can be indexed by ITE entries. The structure of each entry is:

0 7	8	9 15
EXTENDED ITAM STATUS	RETRY FLAG	STANDARD SVC 1 STATUS

Standard mainline register assignment is:

U0	Work--may be destroyed by any subroutine
U1	DCB address
U2	LCB address
U3	BLK address
U4	SVC 1 function code
U5	SVC 1 extended options
U6	Line status code
U7	Level 0 subroutine linkage
U8	Level 1 subroutine linkage
U9	Level 2 subroutine linkage
U10-U15	Work--may be destroyed by any subroutine

7.2.2 Individual Routine Descriptions

Following is a description of the BISYNC terminal manager (INITMBSC) routines:

- INITMBSC

  Entrance from SVC 1 with LCB address. Sets up registers U1 and U2 to contain LCB and DCB addresses. Calls PCINIT for register and LCB initialization. Depending upon whether task is already connected, exits to either BSIDLE or BSBUSY.

- BSIDLE

  Moves LCB parameters, device number, record length, etc., into the DCB by calling the SHIFT subroutine. Also, if the switched line needs connection, BSIDLE connects it. Exits to WRITINIT or READINIT.

- READINIT

  Clears horizontal-tab bit map in the LCB. Builds command and data-chain pointers to perform an SVC 15 PREPARE. Enters SVC 15 driver through SVC15GO. On termination, returns from SVC 15; enters RESETREG to restore environment; enters TOCHOFF to remove DCB from timer chain; and enters ERRORSET to check for any possible error. It sets line status to busy (read side and ACK required), sets a timer to send a WACK, and exits to BSBUSY3.

- WRITINIT

  Sets format records for block to a maximum of seven. Builds into a 2-character buffer (LCB.BF2) the data chain and command pointers to write an ENQ and to read, and enters SVC 15 driver through SVC15GO. On return from SVC 15 driver, enters RESETREG to restore environment; enters TOCHOFF to remove DCB from driver chain; enters ERRORSET to check for possible error; and enters SPC.CHK to analyze the input character. An ACK0, the expected response, sets a TTD timer and causes an exit to BSBUSY3. An ENQ input during the slave mode causes a line turnaround. An EOT will cause an "unavailable device" error return. All other special-character inputs cause a retry.

- BSBUSY

  Puts a request into wait if the LNS.HLD bit is set. Checks for interlace of image and format traffic, incorrect transparent-read buffers, read/write interlace, and outstanding errors. Shifts parameter block information from LCB to DCB. Exits to READIMIG, READFORM, WRITIMIG, or WRITFORM.

- WRITIMIG

  Resets TTD timer and sets LNS.HLD bit. Checks for expired timer, I/O in progress, invalid character ending, and EOF command. Builds command and data chain to WRITE BUFFER and READ into a two-word buffer. Exits to SVC 15 driver through SVC15GO, with termination return to WFTERM.

- READIMIG

  Resets WACK timer and sets LNS.HLD bit. Checks for expired timer and I/O in progress. Builds command and data chain to WRITE ACK and READ data into user buffer. Exits to SVC 15 driver through SVC15GO with termination return to RFTERM.

- READFORM

  Obtains a read-deblocking buffer. If none is available, exits to READER to schedule a READ, sets the LNS.HLD bit, and exits to IRLOUT. If buffer available, enters FORMATR to the deblock buffer; checks for emptied buffer; enters READER to try to READ ahead; and exits to DONE.

- READER

  First-level subroutine. Exits without a READ if I/O is in progress, if no buffer is available, or if ACK is not ready on a WACK timer expiration. For a READ, it builds a command and data chain into internal buffer to WRITE ACK and READ. Exits to SVC 15 driver through SVC15GO, with continuation return to caller and termination return to RFTERM.

- WRITFORM

  Obtains a write-blocking buffer and enters FORMATW to block buffer. If buffer is not full, exit to DONE is scheduled; otherwise, a WRITE is scheduled. No WRITE is made if an I/O is in progress or a TTD timer is expired. For a WRITE, it builds into a 2-word buffer (LCB.BF2) the SVC 15 command and the data chains to WRITE BUFFER and READ. Exit is to SVC 15 driver through SVC15GO, with termination return to WFTERM and continuation return to WRITFCNT. Depending upon line status, WRITFCNT exits to IRLOUT, DONE or CPT.DONE.

- WFTERM

Entered at termination of WRITE BUFFER. Enters RESETREG to restore environment; enters TOCHOFF to remove DCB from timer chain; enters ERRORSET to check for errors; and enters SPC.CHK to validate line input character. Normal path is to WFTGDACK, which complements ACK bit, clears any error status, releases writer buffer, and checks for a second buffer ready for output or checks for an EOT to be sent. On receipt of a different special character, these responses will be sent:

INPUT	ACTION
Wrong ACK	Send ENQ, unless following timeout error (ENQ was sent following timeout); in that case, the buffer is retransmitted.
RVI	Set recoverable-error flag and process as good ACK
WACK	Send an ENQ
NAK	Check retry count. If exhausted, set unrecoverable error; otherwise, retransmit buffer.
EOT	Set "message reject" status and process as unrecoverable error.
Other	Set "protocol error" status and process as unrecoverable error.
Error on Read	Send ENQ
Error on Write	Retry

- RFTERM

  Entered at termination of a READ BUFFER. Enters RESETREG to restore environment; enters TOCHOFF to remove DCB from timer chain; enters ERRORSET to check for possible errors; and enters SPC.CHK to check input character. Normal path (STX or SOH) dictates clearing all statuses, placing the buffer into the deblocking queue, and setting a NACK timer. Normal exit is to IRLOUT or READFORM, depending on the LNS.HLD bit. For other input, the following action will be taken:

INPUT	ACTION
EOT	Handle as a normal buffer and let FORMATR later set an EOF error
ENQ	Retry
TTD	Send NACK
Buffer Ending with ENQ	NACK
Error on Read	NACK
Error on Write	Retry

- TIMERON

  Second-level subroutine. Enter with U15 containing time address and U9 containing termination address. Builds SVC 15 wait command and exits to SVC 15 driver through SVC15GO2, with continuation return to caller.

- TIMEROFF

  Second-level subroutine. Exit with zero status if expired, or with positive status if actually removed. Enters IS State, unlinks DCB, clears ISP table entry, and enters TOCHOFF to remove DCB from timer chain.

- PCINIT

  Second-level subroutine. Loads extended ITAM options (specified or defaulted) and sets memory range to all of memory.

- ERRORSET

  Second-level subroutine. Performs journal if SGN.JRNL not zero. Returns with U15 equal to zero if no error occurred. If error occurred, determines error index, saves error index in LCB.LSTE, increments error count, and returns with either retry or unrecoverable flag.

Whenever possible, all line and hardware errors are retried. For example, if the line carrier is lost, the transmission is retried a specified number of times before a return to the user task with an "unrecoverable error" status.

- ERROR

  Determines the error status from UERTAB and the ITE index from LCB.LSTE, stores status into DCB, and exits to DONE or CPT.DONE. Clears line status, resets all timers and releases all buffers for an unrecoverable error, an EOF, or a zero error status (final transmission completion).

- SVC15GO (SVC 15 Driver Interface)

  Entered with termination address in U9 and continuation address in U10. User of this routine must previously establish data chain with LCB/DCB SCN words, put address of data chain into DCB.NDA, and put address of command chain into DCB.DCW. SVC15GO sets LNS.IO bit; saves DCB.DCW contents in LCB.DCW to facilitate retries; clears DCB.NCE, DCB.ITB, DCB.IFC, DCB.ISTA, DCB.EXST, and DCB.DVST; and puts DCB on timer chain. User byte of continuation address must always have the CPNORL bit set if continuation return is wanted. Exit to SVC 15 driver is to address located in DCB.SVCF, with DCB address in U13.

- IRLOUT

  Continuation exit. Loads DCB address into U13 and exits to ISSEXEC whenever more ITB bits are set. Otherwise, branches to EVRTE.

- RETRY

  Puts address of LCB.SCN1 into DCB.NDA; reloads DCB.DCW from LCB.DCW, as saved by SVC15GO; and exits to SVC15GO.

- DONE

  Resets DCB.HLDB and LCB.HLDB. Exits to IODONE; for a disconnect in progress, waits 100 milliseconds before exiting to IODONE.

- SENDEOT

  Resets any timers and builds command and data chain to write the one-byte EOT. At termination, it exits to ERROR or IRLOUT, depending on LNS.HLDB.

- BUFFER

  Series of second-level subroutines used to perform the following functions:

BUFFER 0	Release Buffer
BUFFER 1	Obtain Write-Blocking Buffer
BUFFER 2	Obtain Read-Deblock Buffer
BUFFER 3	Obtain Free Buffer
BUFFER 4	Obtain Write Buffer on Queue for Output
BUFFER 5	Check for Free Buffer
BUFFER 6	Check for Buffer with I/O in Progress
BUFFER 8	Free All Buffers
BUFFER 9	Put Buffer on Queue

- IOHMBSC

  This module interfaces the BISYNC terminal manager with the SVC 1 and SVC 3 (end-of-task) executors. IOHMBSC consists of I/O handler routines, along with an IOH list; the IOH list has pointers to each of the I/O handler routines.

  The address of the IOH list is assembled into the DCB.IOH field. Consequently, when the user task issues an SVC 1 or an SVC 3, the executor identifies the function request and vectors to the appropriate I/O handling routine, via the IOH list addressed in DCB.IOH.

  The I/O handling routines within IOHMBSC are described below.

ROUTINE NAME	FUNCTION
MBS.READ	SVC 1 Read
MBS.WRIT	SVC 1 Write
MBS.WFM	SVC 1 Write Filemark
MBS.HALT	SVC 1 Halt
MBS.WAIT	SVC 1 Wait
MBS.TEST	SVC 1 Test
MBS.EOT	SVC 3 End-of-Task
MBS.INIT	System Initialization

  MBS.READ, MBS.WRIT, and MBS.WFM share coding. If the terminal manager is not busy, these routines direct it to perform the requested I/O by scheduling ISSEXEC.

  If the terminal manager is busy, these routines queue the request to the IOB queue or to the leaf queue, depending on the following two conditions:

1. If the terminal manager is already busy with the calling task and the same LU, these routines queue the I/O request to the IOB queue addressed by DCB.IOBQ.

2. If the terminal manager is busy with the calling task and a different LU, or with another task, these routines queue the I/O request to the leaf queue via a call to the LEAFQ routine.

Once the terminal manager finishes the ongoing I/O, the routines schedule the ISSEXEC to have the terminal manager execute the queued I/O request.

MBS.HALT, after ensuring no previous I/O requests are pending on the IOB queue, calls HALTITAM, clears statuses, and exits to the operating system.

MBS.WAIT calls the task manager to unchain a task from the ready chain when these conditions are met: the wait request is on behalf of the current task and the current LU; the IOB queue has an entry. Without these conditions, MBS.WAIT passes control to SVC1WAIT.

MBS.TEST returns a condition code of zero if the leaf is disconnected or if the IOB queue is empty. If the leaf is connected and the IOB queue has an entry, MBS.TEST first searches the IOBs to find the IOB for the calling task and then returns a condition code of X'F' to indicate I/O in progress.

MBS.EOT, entered from the SVC 3 (end-of-task) executor, ends task execution gracefully or immediately.

For graceful, protocol-controlled EOT, this routine executes an SVC 1 wait (allowing the I/O to complete) and issues an SVC 7 checkpoint (for a memory-resident task) or an SVC 7 close (for a nonresident task). If issuing an SVC 7 close, the routine also sets the "close in progress" bit to alert the terminal manager.

For an immediate EOT, MBS.EOT executes a HALTITAM, instead of the SVC 1 wait, before issuing the SVC 7 checkpoint or SVC 7 close, as discussed above.

MBS.INIT clears status fields in the DCB and the LCB.

APPENDIX A
SUMMARY DATA DRIVER COMMAND WORDS

	COMMAND	MODIFIER/ COMMAND BYTE HEX	VALID COMMAND BITS	NO. DATA FIELDS	DATA FIELD SPECIFIES
NULL					
	NOP	XX00	CC CT X X XXXX 00000 000	1	Any valid address
	WAIT	XX08	CC CT X 0 XXXX 00001 000	1	Halfword
	XFER	XX10	CC CT X X XXXX 00010 000	1	Halfword
	CXFER	XX18	CC CT X X XXXX 00011 000	2	2 halfwords valid address
CONTROL					
	EXAMINE	XX01	CC CT X TO XXXX 00000 001	1	Byte
	RING WAIT	XX09	CC CT X TO XXXX 00001 001		None
	ANSWER	XX11	CC CT X TO XXXX 00010 001		None
	DISCONNECT	XX19	CC CT X TO XXXX 00011 001		None
READ					
	READ BUFFER	XX02	CC CT BT TO XXXX 00000 010	1,2	Buffer
	READ1	XX0A	CC CT BT TO XXXX 00001 010	1	Byte
	READ2	XX12	CC CT BT TO XXXX 00010 010	2	Byte
PREPARE					
	PREP	XX03	CC CT X TO XXXX 00000 011	1	Byte
	ANTI-PREPARE1	XX0B	CC CT X TO XXXX 00001 011	1	Byte
	ANTI-PREPARE2	XX13	CC CT X TO XXXX 00010 011	1	Byte
WRITE					
	WRITE BUFFER	XX04	CC CT BT TO XXXX 00000 100	1,2	Buffer
	WRITE1	XX0C	CC CT BT TO XXXX 00001 100	1	Byte
	WRITE2	XX14	CC CT BT TO XXXX 00010 100	2	Byte

APPENDIX A (Continued)
SUMMARY DATA DRIVER COMMAND WORDS

	COMMAND	MODIFIER/ COMMAND BYTE HEX	VALID COMMAND BITS	NO. DATA FIELDS	DATA FIELD SPECIFIES
HOLD					
	BREAK	XX05	CC CT X TO XXXX 00000 101	1	Halfword
MODE					
	TOUT	XX06	CC X X X XXXX 00000 110	1	Fullword
	CMD2	XX0E	CC X X X XXXX 00001 110	1	Byte
	RCMD	XX16	CC X X X XXXX 00010 110	1	Byte
	WCMD	XX1E	CC X X X XXXX 00010 110	1	Byte
	RDIS	XX26	CC X X X XXXX 00100 110	1	Byte
	WDIS	XX2E	CC X X X XXXX 00101 110	1	Byte
	DISC	XX36	CC X X X XXXX 00110 110	1	Byte
	SYCT	XX3E	CC X X X XXXX 00111 110	1	Byte
	TRNSL	XX46	CC X X X XXXX 01000 110	1	Byte
	SPEC CHAR	XX4E	CC X X X XXXX 01001 110	1	Fullword
	TRECS	XX56	CC CT X X XXXX 01010 110	1	Halfword

The default values assembled in DCB160, DCB161, DCB162, and DCB163 for the above parameters are:

TOUT	DC H'5',H'5'	5 seconds for read and write
CMD2	DB 0	Not applicable for 201 adapter
RCMD	DB X'49'	Enable, DTR, read
WCMD	DB X'4A'	Enable, DTR, write
RDIS	DB X'89'	Disable, DTR, read
WDIS	DB X'89'	Disable, DTR, read (drops RQ2S for HDX)
DISC	DB X'81'	Disable, read
SYCT	DB X'04'	5 leading syncs
TRNSL	DB 0	ASCII for control, normal, and transparent
TRECS	DC H'80'	80-Byte transparent records
XLT	DAC BEBC.TOP	EBCDIC on the line

The default values for DCB168, DCB169, DCB170, and DCB171 are:

TOUT	DC H'5',H'5'	5 seconds for read and write
CMD2	DB X'30'	QSA program command
RCMD	DB X'59'	Enable read
WCMD	DB X'5B'	Enable write
RDIS	DB X'D9'	Disable read
SYCT	DB X'04'	5 leading syncs
TRNSL	DB 0	ASCII for control, normal, and transparent
TRECS	DC H'80'	80-Byte transparent records
XLT	DAC BEBC.TOP	EBCDIC on the line

APPENDIX B
PARAMETER BLOCK FORMATS
SVC 15 PARAMETER BLOCK

Function Code	Logical Unit	Status
No. Cmd Exec	A(DCW)	
Length of Last Read		Length of Last Write
Data Code 1	A(Data Area 1)	
Data Code 2	A(Data Area 2)	
• • •	• • •	
Data Code n	A(Data Area n)	

FUNCTION CODE BITS

0	Halt I/O Call
1	Command Trap Enable
2	Buffer Trap Enable
3	Termination Trap
4-7	Reserved

DATA CODES

X'00'	Direct Text
X'04'	Indirect Text
X'08'	Chained Buffers
X'80'	Transfer in Data
X'01'	DCW Parameter

BIT NO.	0	1	2	3	4 5 6 7	8 9 10 11 12	13 14 15
(DCW)	C C	C T	B T	T O	Not Used	Command Modifier	Driver Command

BIT NO.
0=Command Chained
1=Command Trap
2=Buffer Traps
3=Time-Out

BIT 13-15

000	Null
001	Control
010	Read
011	Prepare
100	Write
101	Hold
110	Mode

SVC 1 PARAMETER BLOCK

0(0) Function Code	1(1) Logical Unit	2(2) Device Independent Status	3(3) Device Dependent Status
4(4) Buffer Start Address			
8(8) Buffer End Address			
12(C) Random Address			
16(10) Length of Transfer			
20(14) Extended Options			

APPENDIX C
TRANSLATION TABLES

The ITAM BISYNC Line Driver supports American Standard Code for Information Interchange (ASCII) and Extended Binary Coded Decimal Interchange Code (EBCDIC) for transmission over the communications line. Using the translation feature of the auto driver channel, data can be maintained in memory in either EBCDIC or ASCII. The following pages contain these translation tables used by the BISYNC line driver:

- EBCDIC-to-ASCII
- ASCII-to-EBCDIC

NOTE

> This appendix does not include the ASCII-to-ASCII translation tables for even and odd parity because these tables merely add and delete parity bits for writes and reads.

APPENDIX C (Continued)

TABLE C-1 ASCII-to-EBCDIC TRANSLATION

ASCII CODE	CHAR-ACTER	EBCDIC CODE	ASCII CODE	CHAR-ACTER	EBCDIC CODE	ASCII CODE	CHAR-ACTER	EBCDIC CODE
00	NUL	00	2B	+	4E	56	V	E5
01	SOH	01	2C	,	6B	57	W	E6
02	STX	02	2D	–	60	58	X	E7
03	ETX	03	2E	.	4B	59	Y	E8
04	EOT	37	2F	/	61	5A	Z	E9
05	ENQ	2D	30	0	F0	5B	[	4A
06	ACK	2E	31	1	F1	5C	\	E0
07	BEL	2F	32	2	F2	5D	]	4F
08	BS	16	33	3	F3	5E	↑	5F
09	HT	05	34	4	F4	5F	←	6D
0A	LF	25	35	5	F5	60	`	79
0B	VT	0B	36	6	F6	61	a	81
0C	FF	0C	37	7	F7	62	b	82
0D	CR	0D	38	8	F8	63	c	83
0E	SO	0E	39	9	F9	64	d	84
0F	SI	0F	3A	:	7A	65	e	85
10	DLE	10	3B	;	5E	66	f	86
11	DC1	11	3C	<	4C	67	g	87
12	DC2	12	3D	=	7E	68	h	88
13	DC3	13	3E	>	6E	69	i	89
14	DC4	14	3F	?	6F	6A	j	91
15	NAK	3D	40	@	7C	6B	k	92
16	SYN	32	41	A	C1	6C	l	93
17	ETB	26	42	B	C2	6D	m	94
18	CAN	18	43	C	C3	6E	n	95
19	EM	19	44	D	C4	6F	o	96
1A	SUB	3F	45	E	C5	70	p	97
1B	ESC	27	46	F	C6	71	q	98
1C	FS	1C	47	G	C7	72	r	99
1D	GS	1D	48	H	C8	73	s	A2
1E	RS	1E	49	I	C9	74	t	A3
1F	US	1F	4A	J	D1	75	u	A4
20	SP	40	4B	K	D2	76	v	A5
21	!	5A	4C	L	D3	77	w	A6
22	"	7F	4D	M	D4	78	x	A7
23	#	7B	4E	N	D5	79	y	A8
24	$	5B	4F	O	D6	7A	z	A9
25	%	6C	50	P	D7	7B	{	C0
26	&	50	51	Q	D8	7C	¦	6A
27	'	7D	52	R	D9	7D	}	D0
28	(	4D	53	S	E2	7E	~	A1
29	)	5D	54	T	E3	7F	DEL	07
2A	*	5C	55	U	E4	80	PF	04

TABLE C-1 ASCII-to-EBCDIC TRANSLATION (Continued)

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

TABLE C-2 EBCDIC-to-ASCII TRANSLATION

EBCDIC CODE	CHAR-ACTER	ASCII CODE	EBCDIC CODE	CHAR-ACTER	ASCII CODE	EBCDIC CODE	CHAR-ACTER	ASCII CODE
00	NUL	00	2B		98	56		D6
01	SOH	01	2C		99	57		D7
02	STX	02	2D	ENQ	05	58		D8
03	ETX	03	2E	ACK	06	59		D9
04	BF	80	2F	BEL	07	5A	!	21
05	HT	09	30		A0	5B	$	24
06	OC	81	31		A1	5C	*	2A
07	DEL	7F	32	SYN	16	5D	)	29
08		82	33		A2	5E	;	3B
09	RLF	83	34	BM	A3	5F	↑	5E
0A	SMM	84	35	RS	A4	60	—	2D
0B	VT	0B	36	DC	A5	61	/	2F
0C	FF	0C	37	EOT	04	62		E2
0D	CR	0D	38		A6	63		E3
0E	SO	0E	39		A7	64		E4
0F	SI	0F	3A		A8	65		E5
10	DLE	10	3B		A9	66		E6
11	DC1	11	3C	DC4	14	67		E7
12	DC2	12	3D	NAK	15	68		E8
13	DC3	13	3E		F9	69		E9
14	RES	85	3F	SUB	1A	6A	¦	7C
15	NL	86	40	SP	20	6B	,	2C
16	BS	08	41		C1	6C	%	25
17	IL	87	42		C2	6D	←	5F
18	CAN	18	43		C3	6E	>	3E
19	EM	19	44		C4	6F	?	3F
1A	CC	88	45		C5	70		F0
1B		89	46		C6	71		F1
1C	IFS	1C	47		C7	72		F2
1D	IGS	1D	48		C8	73		F3
1E	IRS	1E	49		C9	74		F4
1F	IUS	IF	4A	[	5B	75		F5
20	DS	90	4B	.	2E	76		F6
21	SOS	91	4C	<	3C	77		F7
22	BS	92	4D	(	28	78		F8
23		93	4E	+	2B	79	\	60
24	BYP	94	4F	]	5D	7A	:	3A
25	LF	0A	50	&	26	7B	#	23
26	ETB	17	51		D1	7C	@	40
27	ESC	1B	52		D2	7D	'	27
28		95	53		D3	7E	=	3D
29		96	54		D4	7F	"	22
2A	SM	97	55		D5	80		C0

TABLE C-2 EBCDIC-to-ASCII TRANSLATION (Continued)

EBCDIC CODE	CHAR-ACTER	ASCII CODE	EBCDIC CODE	CHAR-ACTER	ASCII CODE	EBCDIC CODE	CHAR-ACTER	ASCII CODE
81	a	61	AC		AC	D7	P	50
82	b	62	AD		AD	D8	Q	51
83	c	63	AE		AE	D9	R	52
84	d	64	AF		AF	DA		DA
85	e	65	B0		B0	DB		DB
86	f	66	B1		B1	DC		DC
87	g	67	B2		B2	DD		DD
88	h	68	B3		B3	DE		DE
89	i	69	B4		B4	DF		DF
8A		8A	B5		B5	E0	\	5C
8B		8B	B6		B6	E1		E1
8C		8C	B7		B7	E2		53
8D		8D	B8		B8	E3	T	54
8E		8E	B9		B9	E4	U	55
8F		8F	BA		BA	E5	V	56
90		D0	BB		BB	E6	W	57
91	j	6A	BC		BC	E7	X	58
92	k	6B	BD		BD	E8	Y	59
93	l	6C	BE		BE	E9	Z	5A
94	m	6D	BF		BF	EA		EA
95	n	6E	C0	{	7B	EB		EB
96	o	6F	C1	A	41	EC		EC
97	p	70	C2	B	42	ED		ED
98	q	71	C3	C	43	EE		EE
99	r	72	C4	D	44	EF		EF
9A		9A	C5	E	45	F0	0	30
9B		9B	C6	F	46	F1	1	31
9C		9C	C7	G	47	F2	2	32
9D		9D	C8	H	48	F3	3	33
9E		9E	C9	I	49	F4	4	34
9F		9F	CA		CA	F5	5	35
A0		E0	CB		CB	F6	6	36
A1	~	7E	CC		CC	F7	7	37
A2	s	73	CD		CD	F8	8	38
A3	t	74	CE		CE	F9	9	39
A4	u	75	CF		CF	FA		FA
A5	v	76	D0	}	7D	FB		FB
A6	w	77	D1	J	4A	FC		FC
A7	x	78	D2	K	4B	FD		FD
A8	y	79	D3	L	4C	FE		FE
A9	z	7A	D4	M	4D	FF		FF
AA		AA	D5	N	4E			
AB		AB	D6	O	4F			

INDEX

CUT ALONG LINE

PUBLICATION COMMENT FORM

Please use this postage-paid form to make any comments, suggestions, criticisms, etc. concerning this publication.

From ______________________ Date ______________________

Title ______________________ Publication Title ______________________

Company ______________________ Publication Number ______________________

Address ______________________

____

____

FOLD FOLD

Check the appropriate item.

☐ Error Page No. ________ Drawing No. ______________

☐ Addition Page No. ________ Drawing No. ______________

☐ Other Page No. ________ Drawing No. ______________

Explanation:

FOLD FOLD

Fold and Staple

No postage necessary if mailed in U.S.A.

6434

STAPLE STAPLE

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 22 OCEANPORT, N.J.

POSTAGE WILL BE PAID BY ADDRESSEE

PERKIN-ELMER

Computer Systems Division
2 Crescent Place
Oceanport, NJ 07757

TECH PUBLICATIONS DEPT. MS 322A

FOLD FOLD

STAPLE STAPLE