IND$FILE Datastream Description

document number: SS-HCS12-1372-00

```
+--------------------------------------------------------------------------------+
| This document is intended to help designers/programmers understand how         |
| IND$FILE program communicates with its partner programs in workstations.        |
|                                                                                |
| The information in this document is not intended as a specification of the     |
| communication interface that is provided by IND$FILE and its partner            |
| programs in workstations.  The information contained in this document has      |
| not been submitted to any formal IBM test and is distributed on an 'AS IS'     |
| basis without any warranty either expressed or implied, including,             |
| but not limited to, the implied warranties of merchantability or               |
| fitness for a particular purpose.  This disclaimer does not apply if           |
| inconsistent with local law.  The use or the implementation of the             |
| information of this document is a reader's responsibility and depends          |
| on the reader's ability to evaluate and integrate it in the reader's           |
| operational environment or programs.                                           |
|                                                                                |
| Information in this document is subject to change without notice.              |
|                                                                                |
| This document has 26 pages in total (begins with page 11 and ends with page 36). |
+--------------------------------------------------------------------------------+
```

6.2 CUT AND ASYNCHRONOUS FILE TRANSFER

6.2.1 OVERVIEW

For CUT or Asynchronous attach, the HFTP uses standard base 3274 datastream to create a screen format (called a frame) that the device recognizes and that can be identified as a file transfer frame. Frame size is limited to that of a 1920 (model 2) screen to make the appearance consistent to the device and to minimize controller loading. Data transfer between the host and the PC is accomplished via the CUT hardware regen buffer. Data is moved to the CUT buffer directly without altering the screen format (attributes and protected data). Only unprotected fields are modified.

The data which is placed in the frame for transfer can undergo several types of transformations from the time it leaves the disk at one end of the transfer and the time it is written to disk at the other end. The transformations through which the data passes are illustrated in the following diagrams.

```
                 +-------------+
                 |ASCII/EBCDIC |
               +>| Translation |
               | +------+------+
               |          |
               |          |         +------------+
 +-------+     |          +-------->|  3274 DS   |
 | DISK  +-----+------------------->|  Encoder   |
 |       |                          +-----+------+
 |       |                                |
 |       |                                |
 +-------+                                |               Host
 - - - - - - - - - - - - - - - - - - - - -|- - - - - - - - - - - -
                                          V
                                  +------------+
                                  | 3278 MOD 2 |
                                  |   Screen   |
                                  +-----+------+
                                        |
 - - - - - - - - - - - - - - - - - - - -|- - - - - - - - - - - - - -
                                        |                PC
                                        V
 +-------+                           +------------+
 | DISK  |<--------------------------+  3274 DS   |
 |       |                           |  Decoder   |
 |       |                           +------------+
 |       |
 +-------+
```

Figure 3. Data Transformations for Async or CUT download

For outbound transfers (from the host to the PC), the PC validates each frame for the proper sequence number and data integrity. It generates an AID to indicate whether or not the frame was accepted. The data received by the PC will be translated from 3274 internal codes sent by the host.

```
+-------+
| DISK  |
|       +-----------+
|       |           |
|       |           |
+-------+           |            +------------+
                    +------------>|  3274 DS   |
                                  |  Encoder   |
                                  +-----+------+
                                        |
                                        |
                                        |               PC
 - - - - - - - - - - - - - - - - - - - -|- - - - - - - - - - - - - -
                                        V
                                +------------+
                                | 3278 MOD 2 |
                                |   Screen   |
                                +-----+------+
                                        |
 - - - - - - - - - - - - - - - - - - - -|- - - - - - - - - - - - - -
            +--------------------+      |                  Host
            |                    |      V
 +-------+  V   +------------+   |  +------------+
 | DISK  |<-----+ASCII/EBCDIC|<--+--+  3274 DS   |
 |       |      |Translation |      |  Decoder   |
 |       |      +------------+      +------------+
 |       |
 +-------+
```

Figure 4. Data Translations for Async or CUT upload

For inbound transfers (from the PC to the host), an empty frame with the proper sequence number is provided by the HFTP. The PC will translate the data to be transferred and then move the inbound data directly to the CUT buffer and generate an enter key to the controller. Positive acknowledgement for inbound frames is provided by the HFTP via a new empty frame.

6.2.2 COMMUNICATION FLOW

A file transfer using CUT or Asynchronous attachment is accomplished by the exchange of data and control frames between the host and the PC. The order in which the various frame types are expected at the host and PC side of the transfer constitutes the communication sequence for a successful file transfer. If a frame is received out of order, or if an unexpected type of frame is received, then an error has occurred in the sequence and must be dealt with. The following sections describe in detail the sequences for both uploads and downloads.

6.2.2.1 Normal (Non-Error) Sequences

The following four figures show the sequences for Cut and Asynchronous Uploads and Downloads. Each figure consists of a set of labelled flows. The label is a brief description of the type of data being transferred in the direction indicated by the direction of flow arrow. The figure references associated with each flow refer to the detailed description of the transmission unit associated with that flow.

```
   PC                                     Host
  +---+      IND$FILE PUT (PTP) fn ft fm        +---+
  | --+-----------------------------------------+-> |
  |   |           Read Partition Query          |   |
  | <-+-----------------------------------------+-- |
  |   |      Query Reply (CUT or Async Attach)  |   |
  | --+-----------------------------------------+-> |
  |   |           Command Acknowledgement       |   |  Figure 11 on page 16
  | <-+-----------------------------------------+-- |
  |   |           Enter Key (Ack)               |   |  Figure 19 on page 21
  | --+-----------------------------------------+-> |
+-+-> |           Data Request Frame            | <-+-+Figure 14 on page 19
| | <-+-----------------------------------------+-- | |
| |   |          Upload Data Frame              |   | |Figure 15 on page 19
| | --+-----------------------------------------+-> | |
+-+-  |          End of Data Frame              |  -+-+Figure 16 on page 20
  | --+-----------------------------------------+-> |
  |   |   Normal Completion Control Frame       |   |  Figure 12 on page 17
  | <-+-----------------------------------------+-- |
  |   |           Enter Key (Ack)               |   |
  | --+-----------------------------------------+-> |
  +---+                                         +---+
```

Figure 5. Sequence for CUT Upload

```
   PC                                     Host
  +---+      IND$FILE PUT (PTP) fn ft fm        +---+
  | --+-----------------------------------------+-> |
  |   |          Read Partition Query           |   |
  | <-+-----------------------------------------+-- |
  |   |      Query Reply (CUT or Async attach)  |   |
  | --+-----------------------------------------+-> |
  |   |     Command Acknowledgement Frame       |   |  Figure 11 on page 16
  | <-+-----------------------------------------+-- |
  |   |           Enter Key (Ack)               |   |  Figure 19 on page 21
  | --+-----------------------------------------+-> |
+-+-> |          Download Data Frame            | <-+-+Figure 17 on page 20
| | <-+-----------------------------------------+-- | |
| |   |           Enter Key (Ack)               |   | |
| | --+-----------------------------------------+-> | |
+-+-  |           End of Data Frame             |  -+-+Figure 18 on page 21
  | <-+-----------------------------------------+-- |
  |   |           Enter Key (Ack)               |   |
  | --+-----------------------------------------+-> |
  |   |     Normal Completion Control Frame     |   |  Figure 12 on page 17
  | <-+-----------------------------------------+-- |
  |   |           Enter Key (Ack)               |   |
  | --+-----------------------------------------+-> |
  +---+                                         +---+
```

Figure 6. Sequence for CUT Download

6.2.2.2 Error Sequences

This section provides details on the communication sequences which follow the detection of an error on either side of the transmission. The errors which will be detected are divided into two categories, data errors and terminating errors. Both error types and their associated communication sequences are described in the following sections.

DATA ERRORS

Data errors are errors which are detected in the transmitted data itself. They are detected when the Check Sum calculated against the data received does not match the Check Sum calculated against the sent data. Data errors are dealt with by responding to the bad frame with a Retransmit negative acknowledgement. The protocols for retransmit responses from both the host and the PC are depicted in the figure below.

```
 +---+          Bad Data Frame         +---+
 | <-+---------------------------------+-- |
 |   |     Re-Transmit Negative Ack    |   | Figure 20 on page 21
 | --+---------------------------------+-> |
 |   |Re-Transmission of Bad Data Frame|   |
 | <-+---------------------------------+-- |
 +---+                                 +---+
```

Figure 7. Data Error Sequence (CUT) Initiated by PC

```
 +---+          Bad Data Frame         +---+
 | --+---------------------------------+-> |
 |   | Re-Transmit Data Request Frame  |   |  *
 | <-+---------------------------------+-- |
 |   |Re-Transmission of Bad Data Frame|   |
 | --+---------------------------------+-> |
 +---+                                 +---+

  * 6.2.4.5, "Retransmit Data Request Frame" on page 19
```

Figure 8. Data Error Sequence (CUT) Initiated by the Host

TERMINATING ERRORS

Terminating errors cover any errors detected on either side of the transmission which are not Data Errors. Terminating Errors result in the termination of the file transfer. The communication sequences which result from a transmission error being detected on the host and PC sides of the transfer are shown in the figure below.

```
 +---+     Control or Data Frame       +---+
 | <-+---------------------------------+-- |
Terminating                            |   |
Error Detected                         |   |
 |   |     Abort Acknowledgement       |   | Figure 23 on page 22
 | --+---------------------------------+-> |
 |   |          Abort Frame            |   | Figure 13 on page 18
 | <-+---------------------------------+-- |
 |   |          Enter Key (Ack)        |   |
 | --+---------------------------------+-> |
 +---+                                 +---+
```

Figure 9. Terminating Error Sequence (CUT) Initiated in PC

```
 +---+  HFTP Invocation or Data Frame  +---+
 | --+---------------------------------+-> |
 |   |                              Terminating
 |   |                              Error Detected
 |   |          Abort Frame            |   |  Figure 13 on page 18
 | <-+---------------------------------+-- |
 |   |          Enter Key (Ack)        |   |
 | --+---------------------------------+-> |
 +---+                                 +---+
```

Figure 10. Terminating Error Sequence (CUT) Initiated by the Host

6.2.3 PC/HOST MESSAGE INTERFACE

For CUT Attachment, data transfer is accomplished through the screen image buffer in the adapter. The buffer will be divided into two 3270 fields, one for host to 3270 Emulator flow (attribute byte '7C'x - protected, non-display, numeric) and the other an unprotected field for 3270 Emulator to host flow. The relative size of these fields will depend upon the direction of data flow, as opposed to response or control only flow. PF key AID's will be used for simple responses to the host. The 3270 Emulator will recognize the '7C'x attribute byte, as translated by the 3274 controller, in the last position to determine that the logical refresh buffer contains data for the 3270 Emulator application, rather than the actual screen.

6.2.4 FRAME DESCRIPTIONS

Information is exchanged between the PC and the Host using the CUT attachment method using 3270 messages. This section describes the messages that are encoded by the HFTP and send to the controller. The next section (Logical Refresh Buffer Formats) describes how these messages appear in the buffer on the PC.

6.2.4.1 Command Acknowledgment Frame

```
+-------+-------+-------+-------+---------+-------+---------+-----
| x'27' | x'F5' | x'C2' | x'11' | x'5D7F' | x'11' | x'4040' |
|       |       |       |       |         |       |         |
|  ESC  |  EW   |  WCC  |  SBA  | TCAM    |  SBA  |  (0)    |
|  (TSO ONLY)   |       |       | Signal  |       |         |
|       |       |       |       |         |       |         |
+-------+-------+-------+-------+---------+-------+---------+--
        --+-------+-------+---------+-------------+-------+---------+-----
          | x'C3' | x'ss' | x'8181' |   x'40'     | x'11' | x'5D7A' |
          |       |       |         |             |       |         |
          |Control|Frame  | Status  |  Message    |  SBA  | (1914)  |
          |Code   |Sequence Code    |             |       |         |
          |       |       |         |             |       |         |
     -----+-------+-------+---------+-------------+-------+---------+--
        --+-------+-------+-------+-------------+-------+-------+
          | x'1D' | x'C1' | x'13' | X'00000000' | x'1D' | x'7C' |
          |       |       |       |             |       |       |
          |  SF   |Unpro- |Set    | Response    |  SF   |Prot   |
          |       |tected |Cursor | Area        |       |Nondisp|
          |       |MDT on |       |             |       |Numeric|
     -----+-------+-------+-------+-------------+-------+-------+
```

Figure 11. Command Acknowledgement Frame Format.

The command acknowledgement frame is used to signal successful completion of the HFTP invocation parsing and initialization. This frame format is identical to the format of the Normal Completion, and Abort Frames, except for the contents of the Status Code and Message fields. The status code of aa (x'8181') indicates that this is a Command Acknowledgement frame and that the message field will be blank.

6.2.4.2 Normal Completion Frame

```
+-------+-------+-------+-------+---------+-------+---------+-----
| x'27' | x'F5' | x'C2' | x'11' | x'5D7F' | x'11' | x'4040' |
|       |       |       |       |         |       |         |
|  ESC  |  EW   |  WCC  |  SBA  | TCAM    |  SBA  |  (0)    |
|  (TSO ONLY)   |       |       | Signal  |       |         |
|       |       |       |       |         |       |         |
+-------+-------+-------+-------+---------+-------+---------+--
        --+-------+-------+---------+--------------+-------+---------+-----
          | x'C3' | x'ss' | x'8189' | See Note 1   | x'11' | x'5D7A' |
          |       |       |         |              |       |         |
          |Control|Frame  | Status  |  Message     |  SBA  | (1914)  |
          |Code   |Sequence Code    |              |       |         |
          |       |       |         |              |       |         |
     -----+-------+-------+---------+--------------+-------+---------+--
        --+-------+-------+-------+-------------+-------+-------+
          | x'1D' | x'C1' | x'13' | X'00000000' | x'1D' | x'7C' |
          |       |       |       |             |       |       |
          |  SF   |Unpro- |Set    | Response    |  SF   |Prot   |
          |       |tected |Cursor | Area        |       |Nondisp|
          |       |MDT on |       |             |       |Numeric|
      ----+-------+-------+-------+-------------+-------+-------+
```

Note 1: The message field is 80 characters (bytes) long and contains
          the following EBCDIC text: TRANS03   File Transfer Complete
          The message is padded with blanks (x'40')

Figure 12. Normal Completion Frame format.

The Normal Completion Frame is sent from the Host to the PC to indicate successful completion of the file transfer operation.

6.2.4.3 Abort Frames

The Abort Frames are used to indicate that an error has occurred in the file transfer operation and that the operation is to be aborted. The frames will all have the same format, the different error types will be specified by the Status Code and the message text in the Message field. The basic format is shown in Figure 13 on page 18.

```
+-------+-------+-------+-------+---------+-------+---------+-----
| x'27' | x'F5' | x'C2' | x'11' | x'5D7F' | x'11' | x'4040' |
|       |       |       |       |         |       |         |
|  ESC  |  EW   |  WCC  |  SBA  | TCAM    |  SBA  |  (0)    |
|  (TSO ONLY)   |       |       | Signal  |       |         |
|       |       |       |       |         |       |         |
+-------+-------+-------+-------+---------+-------+---------+--
        --+-------+-------+---------+--------------+-------+---------+-----
          | x'C3' | x'ss' | x'rrrr' |              | x'11' | x'5D7A' |
          |       |       |         |              |       |         |
          |Control|Frame  | Status  |  Message     |  SBA  | (1914)  |
          |Code   |Sequence Code    |              |       |         |
          |       |       |         |              |       |         |
     -----+-------+-------+---------+--------------+-------+---------+--
        --+-------+-------+-------+-------------+-------+-------+
          | x'1D' | x'C1' | x'13' | X'00000000' | x'1D' | x'7C' |
          |       |       |       |             |       |       |
          |  SF   |Unpro- |Set    | Response    |  SF   |Prot   |
          |       |tected |Cursor | Area        |       |Nondisp|
          |       |MDT on |       |             |       |Numeric|
     -----+-------+-------+-------+-------------+-------+-------+
```

Status Codes: am (x'8194') - File error (See text for Message contents)
            aq (x'8198') - Transmission error (See text for Message
                         contents)

Figure 13. Abort Frame from the Host to the PC

The contents of the Message field will be dependent on the Status Code value. For a list of the messages which can appear in the Message area, see 6.5, "Error Messages and Codes" on page 36. For all messages, the message area is 80 bytes (characters) long. The messages are transmitted in EBCDIC and are padded on the right with blanks (x'40').

For the aq (x'8198') Status Code, the message area will contain a single blank (x'40').

6.2.4.4 Data Request Frame (Screen format for PC to Host Transmission)

The data request frame constitutes the empty frame sent from the Host to the PC to be filled with data for transfer to the host. This empty frame also serves as positive acknowledgement to the PC that the previous frame of data was successfully received.

```
+-------+-------+-------+-------+---------+-------+---------+-------+-----
| x'27' | x'F5' | x'C2' | x'11' | x'5D7F' | x'11' | x'4040' | x'C2' |
|       |       |       |       |         |       |         |       |
|  ESC  |  EW   |  WCC  |  SBA  | TCAM    |  SBA  |  (0)    |Data   |
|  (TSO ONLY)   |       |       | Signal  |       |         |Request|
|       |       |       |       |         |       |         |       |
+-------+-------+-------+-------+---------+-------+---------+-------+--
        --+-------+-------+-------+-------+-------+-----
          | x'1D' | x'C1' | x'C1' | x'ss' | x'13' |
          |       |       |       |       |       |
          |  SF   |Unpro- | Data  |Frame  | Set   |
          |       |tected | Code  |Sequence Cursor|
          |       |MDT on |       |Note 1 |       |
     -----+-------+-------+-------+-------+-------+--
        --+-------+---------+-------+-------+
          | x'11' | x'5D7F' | x'1D' | x'7C' |
          |       |         |       |       |
          |  SBA  | (1919)  |  SF   |Prot   |
          |       |         |       |Nondisp|
          |       |         |       |Numeric|
     -----+-------+---------+-------+-------+
```

Figure 14. Data Request Frame

Note 1: This count is modulo 64 and is encoded according to Figure 47 on page 36.

6.2.4.5 Retransmit Data Request Frame

The Retransmit Data Request Frame is sent to the PC as negative acknowledgement of the previous data frame sent on an upload. The format of the frame is identical to that of the Data Request Frame (see Figure 14 on page 19) except the Data Request value of x'C2' is replaced by the Retransmit code x'4C'.

6.2.4.6 Upload Data Frame Format

The following diagram show the format of a Data Frame as it is received at the host. (See the Logical Refresh Buffer Format section for the format of the frame in the PC buffer.)

```
+-------+---------+-------+---------+-------+-------+-----
| x'7D' | x'cccc' | x'11' | x'40C2' | x'C1' | x'ss' |
|       |         |       |         |       |       |
| AID   | Cursor  | SBA   | Buffer  | Data  |Frame  |
|(ENTER)| Pos.-   |       | Address | Code  |Sequence
|       | Ignored |       | for Data|       |       |
+-------+---------+-------+---------+-------+-------+--
        --+-------+---------+---------------+
          | x'cc' | x'llll' |               |
          |       |         |               |
          | Check | Data    |  Data         |
          | Sum   | Length  |               |
          |       |         |               |
     -----+-------+---------+---------------+
```

Figure 15. Upload Data Frame Format

6.2.4.7 Upload End of Data Frame

The End of Data Frame is a special data frame used to signify that all of the file data has been sent.   The format of the End of Data Frame used during Uploads is shown in the figure below.

```
+-------+---------+-------+---------+-------+-------+-----
| x'7D' | x'cccc' | x'11' | x'40C2' | x'C1' | x'ss' |
|       |         |       |         |       |       |
| AID   | Cursor  | SBA   | Buffer  | Data  |Frame  |
|(ENTER)| Pos.-   |       | Address | Code  |Sequence
|       | Ignored |       | for Data|       |       |
+-------+---------+-------+---------+-------+-------+--
        --+-------+---------+---------------+
          | x'cc' | x'8183' |  x'5CA9'      |
          |       |         |               |
          | Check | Data    |  CR EOF       |
          | Sum   | Length  |               |
          |       |         |               |
     -----+-------+---------+---------------+
```

Figure 16. Upload End of Data Frame Format

6.2.4.8 Download Data Frame Format

The following diagram shows the format of the Download Data Frame. This frame is used during a download to move actual file data from the host to the PC. The frame shown is as it is build in the HFTP.

```
+-------+-------+-------+-------+---------+-------+---------+-------+-----
| x'27' | x'F5' | x'C2' | x'11' | x'5D7F' | x'11' | x'4040' | x'C1' |
|       |       |       |       |         |       |         |       |
|  ESC  |  EW   |  WCC  |  SBA  | TCAM    |  SBA  |  (0)    | Data  |
|  (TSO ONLY)   |       |       | Signal  |       |         | Code  |
|       |       |       |       |         |       |         |       |
+-------+-------+-------+-------+---------+-------+---------+-------+--
        --+-------+-------+---------+--------------+-------+---------+-----
          | x'ss' | x'cc' | x'llll' |              | x'11' | x'5D7A' |
          |       |       |         |              |       |         |
          |Frame  | Check | Data    |  Data        |  SBA  | (1914)  |
          |Sequence Sum   | Length  |              |       |         |
          |       |       |         |              |       |         |
     -----+-------+-------+---------+--------------+-------+---------+--
        --+-------+-------+-------+-------------+-------+-------+
          | x'1D' | x'C1' | x'13' | X'00000000' | x'1D' | x'7C' |
          |       |       |       |             |       |       |
          |  SF   |Unpro- |Set    | Response    |  SF   |Prot   |
          |       |tected |Cursor | Area        |       |Nondisp|
          |       |MDT on |       |             |       |Numeric|
      ----+-------+-------+-------+-------------+-------+-------+
```

Figure 17. Download Data Frame Format

6.2.4.9 Download End of Data Frame

The End of Data Frame is a special Download Data Frame which is used during Downloads to signify that all of the file data has been transferred. The format of the Host to PC End of Data Frame is shown in the following figure.

```
+-------+-------+-------+-------+---------+-------+---------+-------+-----
| x'27' | x'F5' | x'C2' | x'11' | x'5D7F' | x'11' | x'4040' | x'C1' |
|       |       |       |       |         |       |         |       |
|  ESC  |  EW   |  WCC  |  SBA  | TCAM    |  SBA  |  (0)    | Data  |
|  (TSO ONLY)   |       |       | Signal  |       |         | Code  |
|       |       |       |       |         |       |         |       |
+-------+-------+-------+-------+---------+-------+---------+-------+--
        --+-------+-------+---------+--------------+-------+---------+-----
          | x'ss' | x'cc' | x'8183' |  x'5CA9'     | x'11' | x'5D7A' |
          |       |       |         |              |       |         |
          |Frame  | Check | Data    |  CR EOF      |  SBA  | (1914)  |
          |Sequence Sum   | Length  |              |       |         |
          |       |       |         |              |       |         |
     -----+-------+-------+---------+--------------+-------+---------+--
        --+-------+-------+-------+-------------+-------+-------+
          | x'1D' | x'C1' | x'13' | X'00000000' | x'1D' | x'7C' |
          |       |       |       |             |       |       |
          |  SF   |Unpro- |Set    | Response    |  SF   |Prot   |
          |       |tected |Cursor | Area        |       |Nondisp|
          |       |MDT on |       |             |       |Numeric|
     -----+-------+-------+-------+-------------+-------+-------+
```

Figure 18. Download End of Data Frame Format

6.2.4.10 PC Acknowledgments to Transmissions from Host

The following figures show the formats for the various types of acknowledgements which are sent by the PC to the Host to indicate receipt of a transmission from the host. NOTE : For the single AID key acknowledgements, the controller will always add the cursor address and a Set Buffer Address to the end of the AID key transmission.

```
+-------+
| x'7D' |
|       |
| AID   |
|(ENTER)|
|       |
+-------+
```

Figure 19. Enter Key - Positive Acknowledgment to Host Transmission

```
+-------+---------+-------+---------+-------+-------+---------+
| x'F1' | x'cccc' | x'11' | x'5D7B' | x'4C' | x'ss' | X'rrrr' |
|       |         |       |         |       |       |         |
| AID   | Cursor  | SBA   | Buffer  | Re-   |Frame  | Reason  |
| (PF1) | Pos.-   |       | Address | Xmit  |Sequence Code    |
|       | Ignored |       | for Resp|       |       |         |
+-------+---------+-------+---------+-------+-------+---------+
```

Figure 20. Negative Acknowledgment to Host Transmission (Re-Transmit)

Note : Receipt of a Negative Acknowledgement (Re-Transmit) from the PC will cause the HFTP to re-transmit the previously sent message.

```
+-------+
| x'6D' |
|       |
| AID   |
|(CLEAR)|
|       |
+-------+
```

Figure 21. Resynchronize Acknowledgement (VM)

```
+-------+
| x'6E' |
|       |
| AID   |
| (PA2) |
|       |Resynchronize - TSO
+-------+
```

Figure 22. Resynchronize Acknowledgment (TSO)

```
+-------+---------+-------+---------+-------+-------+---------+
| x'F2' | x'cccc' | x'11' | x'5D7B' | x'C3' | x'ss' | X'rrrr' |
|       |         |       |         |       |       |         |
| AID   | Cursor  | SBA   | Buffer  |Control|Frame  | Reason  |
| (PF2) | Pos.-   |       | Address | Code  |Sequence Code    |
|       | Ignored |       | for Resp|       |       |         |Abort
+-------+---------+-------+---------+-------+-------+---------+
```

Figure 23. Abort Acknowledgement

NOTE : The Reason code field in the above Acknowledgements is not used in the actual implementation.

An Abort Acknowledement from the PC results in the beginning of the abort sequence from the host. (See Error Sequences).

6.2.4.11 The DDM Query Reply

The HFTP determines the attachment type via the Query Reply sent by the PC. A CUT attachment is signified by the lack of a TCA or PC3270 attachment. The TCA attachment is signified by the query reply code x'95' where the PC3270 attachment is signified by a query reply code x'93'. If neither of these are present in the query reply transmission, then the HFTP assumes a CUT or Asynchronous attachment. (The verbs GET or GTP and PUT or PTP will be used to distinguish between Asynchronous and CUT.)

6.2.5 LOGICAL REFRESH BUFFER FORMATS

The Logical Refresh Buffer Formats represent the format of the data that will appear in the buffer on the PC. For Host to PC Transmissions the format of the data that is written to the regen buffer on the PC is shown in the figure below.

```
+-------+-------+-------+---------+----------------------------+
| x'A0' | x'ss' | x'cc' | x'llll' |                            |
|       |       |       |         |                            |
| 'A'   |Frame  | Check | Data    |  Data ...                  |
| Data  |Sequence Sum   | Length  |                            |
| Code  |       |       |         |                            |
+-------+-------+-------+---------+--/                         |
|0      1       2       3           5                          |
|                                                              |
|                                                              |
|                                                              |
|                                                              |
|                              /-+-------+-------------+-------+
|                                | x'C1' | x'rrrrrrrr' | x'FC' |
|                                |       |             |       |
|                                | Attr. | Response    |Attr.  |
|                                | Unpro-| Area        |Numeric|
|                                | tected|             |Protect|
+--------------------------------+-------+-------------+-------+
                                  1914     1915          1919
```

Figure 24. Host to PC Transmission

The format of the buffer to be uploaded to the host is shown in the following figure.

```
+-------+-------+-------+-------+-------+---------+------------+
| x'A1' | x'C1' | x'A0' | x'ss' | x'cc' | x'llll' |            |
|       |       |       |       |       |         |            |
| 'B'   | Attr. | 'A'   |Frame  | Check | Data    |  Data ...  |
| Resp. | Unpro-| Data  |Sequence Sum   | Length  |            |
| Code  | tected| Code  |       |       |         |            |
+-------+-------+-------+-------+-------+---------+--/         |
|0      1       2       3       4       5           7          |
|                                                              |
|                                                              |
|                                                              |
|                                                              |
|                                                   /-+-------+
|                                                     | x'FC' |
|                                                     |       |
|                                                     |Attr.  |
|                                                     |Nondisp|
|                                                     |Protect|
+-----------------------------------------------------+-------+
```

Figure 25. PC to Host Transmission

6.2.6 ASYNCHRONOUS CONSIDERATIONS

Asynchronous attachment is implemented in the same manner as CUT attachment with the following exceptions.

+ In the Command Acknowledgement (Figure 11 on page 16), Normal Completion (Figure 12 on page 17), Abort (Figure 13 on page 18), Data Request (Figure 14 on page 19) and Download Data Frames (Figure 17 on page 20), the x'7C' is the last field (Protected, Nondisplay, Numeric) is replaced by a x'F0' (Protected, Numeric).

+ The single byte check sum field is replaced by a three byte extended Frame Check Sequence. (see section 6.4.2, "CRC Computation - CUT and Asynch" on page 34 for details on its calculation).

6.3 DFT FILE TRANSFERS

For DFT attach, data transfer between the host and the PC is accomplished by Structured Fields which are used to send and receive messages within buffer space provided by the PC utility. In the 3270 Emulators, the messages are sent using the MFI (Main Frame Interface) Read and Write Structured Field APIs. These APIs move data to/from a buffer provided by the caller and ship it to the host via the TCA/DCA card buffer.

For outbound transmissions (from the host to the PC), the data transfer structured fields described in this document will be recognized by the PC communication code, moved from the TCA/DCA card buffer into the application specified buffer, and the application signalled that a buffer has been received.

For inbound transmissions (from the PC to the host), the data transfer structured fields should be constructed and moved to the TCA/DCA card buffer for transmission. The HFTP will recognize the data transfer structured field and process them accordingly.

Unlike Asynchronous and CUT attach, DFT transmissions do not require encoding before they are sent. Therefore, the only translation that the transmitted data will go through is EBCDIC/ASCII translation if that option is specified on the HFTP invocation. The flow of information is shown in the figure below.

```
  +------+           +------------+
  | DISK |<----+---->|ASCII/EBCDIC|
  |      |     |     |Translation |
  |      |     |     +------------+
  |      |     |           A
  +------+     +----------->|                               HOST
 - - - - - - - - - - - - - -|- - - - - - - - - - - - - - - - - - - -
                            V
                     +------------+
                     | Read/Write |
                     |Struc. Field|
                     +------------+
                          A
 - - - - - - - - - - - - -|- - - - - - - - - - - - - - - - - - - - -
                          |                    +------+      PC
                          +------------------->| DISK |
                                               |      |
                                               |      |
                                               +------+
```

Figure 26. Data Transformations for DFT File Transfers

NOTE : The EBCDIC/ASCII translations are all done on the host.

6.3.1 COMMUNICATION FLOW

A file transfer between the PC and the HFTP using DFT attach is accomplished by the exchange of data and control information using 3270 DS structured fields. The order in which the structured fields are exchanged during the file transfer constitutes the communication sequence for a successful file transfer. If an unexpected structured field arrives, then an error has occurred and must be dealt with.

DFT attachment type supports buffer sizes ranging from 2K to 32K. What this buffer size refers to is the amount of data which will be sent in a single data structured field. (This is equivalent to the amount of data which is sent to or received from the host before an application level acknowledgement is expected.)

The following two sections describe the Normal and Error sequences supported by the HFTP.

6.3.1.1 Normal (Non-Error) Sequences

The following figures show the communication sequences for DFT Uploads and Downloads. Each figure consists of a set of labelled flows. The label is a brief description of the type of data being transferred in the direction indicated by the direction of the flow arrow. The figure references associated with each flow refer to the detailed description of the transmission unit associated with that flow.

```
   PC                                     Host
  +---+          IND$FILE PUT   .....           +---+
  | --+-----------------------------------------+-> |
  |   |          Read Partition Query           |   |
  | <-+-----------------------------------------+-- |
  |   |            Query Reply                  |   |
  | --+-----------------------------------------+-> |
  |   |          Open for Upload                |   |  Figure 33 on page 28,
  | <-+-----------------------------------------+-- |     Figure 34 on page 28
  |   |          Open Acknowledge               |   |  Figure 36 on page 29
  | --+-----------------------------------------+-> |
+-+-> |           Set Cursor and Get            | <-+-+Figure 37 on page 29
| | <-+-----------------------------------------+-- | |
| |   |           Upload Data Buffer            |   | |Figure 38 on page 30
| | --+-----------------------------------------+-> | |
+-+-  |       Get Past End of File Error        |  -+-+Figure 43 on page 32
  | --+-----------------------------------------+-> |
  |   |               Close Request             |   |  Figure 41 on page 31
  | <-+-----------------------------------------+-- |
  |   |            Close Reply                  |   |  Figure 42 on page 31
  | --+-----------------------------------------+-> |
  |   |               Open for Messages         |   |  Figure 35 on page 29
  | <-+-----------------------------------------+-- |
  |   |          Open Acknowledgment            |   |
  | --+-----------------------------------------+-> |
  |   |       MSG : Transfer Complete           |   |  *
  | <-+-----------------------------------------+-- |
  |   |          Data Acknowledgement           |   |  Figure 40 on page 30
  | --+-----------------------------------------+-> |
  +---+                                         +---+

  * 6.3.2.10, "Message Structured Field Format" on page 30
```

Figure 27. Sequence for DFT Upload

```
     PC                                     Host
   +---+          IND$FILE GET   ......          +---+
   | --+-----------------------------------------+-> |
   |   |           Read Partition Query          |   |
   | <-+-----------------------------------------+-- |
   |   |             Query Reply                 |   |
   | --+-----------------------------------------+-> |
   |   |               Open for Download         |   |  Figure 31 on page 27,
   | <-+-----------------------------------------+-- |   Figure 32 on page 28
   |   |          Open Acknowledgment            |   |  Figure 36 on page 29
   | --+-----------------------------------------+-> |
 +-+-> |           Download Data Buffer          | <-+-+Figure 39 on page 30
 | | <-+-----------------------------------------+-- | |
 | |   |          Data Acknowledgment            |   | |Figure 40 on page 30
 | | --+-----------------------------------------+-> | |
 +-+-  |            Close Request                |  -+-+Figure 41 on page 31
   | <-+-----------------------------------------+-- |
   |   |           Close Acknowledgment          |   |  Figure 42 on page 31
   | --+-----------------------------------------+-> |
   |   |            Open for Messages            |   |  Figure 35 on page 29
   | <-+-----------------------------------------+-- |
   |   |          Open Acknowledgment            |   |
   | --+-----------------------------------------+-> |
   |   |      MSG : File Transfer Complete       |   |  *
   | <-+-----------------------------------------+-- |
   |   |          Data Acknowledgment            |   |  Figure 40 on page 30
   | --+-----------------------------------------+-> |
   +---+                                         +---+
```

* 6.3.2.10, "Message Structured Field Format" on page 30

Figure 28. Sequence for DFT Download

6.3.1.2 Error Sequences

This section provides details on the communication sequences which follow the detection of an error on either side of the transmission. For DFT file transfers, only terminating errors are detected (as no Check Sum is calculated for the data transmitted.) The Terminating Error sequences for DFT transfers are described by the following diagrams.

```
 +---+     Transmission From Host      +---+
 | <-+---------------------------------+-- |
Terminating                            |   |
Error Detected                         |   |
 |   |           Error Response        |   |  Figure 43 on page 32
 | --+---------------------------------+-> |
 |   |          Open for Messages      |   |  Figure 35 on page 29
 | <-+---------------------------------+-- |
 |   |       Open Acknowledgment       |   |  Figure 36 on page 29
 | --+---------------------------------+-> |
 |   |      MSG : TRANSxx Message      |   |  *
 | <-+---------------------------------+-- |
 |   |       Data Acknowledgment       |   |
 | --+---------------------------------+-> |
 +---+                                 +---+
```

* 6.3.2.10, "Message Structured Field Format" on page 30

Figure 29. Terminating Error Sequence (DFT) Initiated by PC

```
 +---+ HFTP Invocation or Data Buffer  +---+
 | --+---------------------------------+-> |
 |   |                             Terminating
 |   |                             Error Detected
 |   |          Open for Messages      |   | Figure 35 on page 29
 | <-+---------------------------------+-- |
 |   |       Open Acknowledgment       |   | Figure 36 on page 29
 | --+---------------------------------+-> |
 |   |      MSG : TRANSxx Message      |   | *
 | <-+---------------------------------+-- |
 |   |       Data Acknowledgment       |   |
 | --+---------------------------------+-> |
 +---+                                 +---+

 * 6.3.2.10, "Message Structured Field Format" on page 30
```

Figure 30. Terminating Error Sequence (DFT) Initiated by Host

6.3.2 STRUCTURED FIELD FORMATS

Data and Acknowledgements between the 3270 Emulator and the host using the DFT attachment method are encoded as a subset of the Distributed Data Management (DDM) architecture, enumerated below:

6.3.2.1 Open for Download

x'0023'	x'D00012'	x'010601010403'	x'0A0A0000000011010100'	x'50055203F0'	x'0309'	c'FT:DATA'
Struct. Field Length	Open Request	Fixed Parameter Changes File, etc.	Functions Required Sequential Insert	Data Not Compressed	File Nm. Header w/length	Fixed Name for File Xfr. Data

Figure 31. Open for Download Structured Field Format

Note - The four Open Structured fields here are based on the DDM Open Structured Field format.

6.3.2.2 Open for Download (with Record Size)

This alternate Open Structured field is used when either DDM Query Reply indicates an outbound limit greater than 2048 bytes or the size field on the DFT option is specified and is greater than 2048.

```
+---------+-----------+-----------------+-------------------------+-----
| x'0029' | x'D00012' | x'010601010403' | x'0A0A0000000011010100' |
|         |           |                 |                         |
| Struct. | Open      | Fixed Parameter | Functions Required      |
| Field   | Request   | Changes File,   | Sequential Insert       |
| Length  |           | etc.            |                         |
+---------+-----------+-----------------+-------------------------+--
    --+---------------+-------------+---------+---------+------------+
      | x'50055203F0' | x'08062704' | x'llll' | x'0309' | c'FT:DATA' |
      |               |             |         |         |            |
      | Data Not      | Record Size | Size    | File Nm.| Fixed Name |
      | Compressed    | Header      | (LIMIN  | Header  | for File   |
      |               | w/length    | -17)    | w/length| Xfr. Data  |
 -----+---------------+-------------+---------+---------+------------+
```

Figure 32. Open for Download (with Buffer Size) Structured Field Format

6.3.2.3 Open for Upload

```
+---------+-----------+-----------------+-------------------------+-----
| x'0023' | x'D00012' | x'010601010403' | x'0A0A0001000000000100' |
|         |           |                 |                         |
| Struct. | Open      | Fixed Parameter | Functions Required      |
| Field   | Request   | Changes File,   | Sequential Get          |
| Length  |           | etc.            |                         |
+---------+-----------+-----------------+-------------------------+--
        --+---------------+---------+------------+
          | x'50055203F0' | x'0309' | c'FT:DATA' |
          |               |         |            |
          | Data Not      | File Nm.| Fixed Name |
          | Compressed    | Header  | for File   |
          |               | w/length| Xfr. Data  |
     -----+---------------+---------+------------+
```

Figure 33. Open for Upload Structured Field Format

6.3.2.4 Open for Upload (with Record Size)

This alternate Open Structured field is used when either DDM Query Reply indicates an inbound limit greater than 2048 bytes or the size field on the DFT option is specified and is greater than 2048.

```
+---------+-----------+-----------------+-------------------------+-----
| x'0029' | x'D00012' | x'010601010403' | x'0A0A0001000000000100' |
|         |           |                 |                         |
| Struct. | Open      | Fixed Parameter | Functions Required      |
| Field   | Request   | Changes File,   | Sequential Get          |
| Length  |           | etc.            |                         |
+---------+-----------+-----------------+-------------------------+--
    --+---------------+-------------+---------+---------+------------+
      | x'50055203F0' | x'08062704' | x'llll' | x'0309' | c'FT:DATA' |
      |               |             |         |         |            |
      | Data Not      | Record Size | Size    | File Nm.| Fixed Name |
      | Compressed    | Header      | (LIMIN  | Header  | for File   |
      |               | w/length    | -17)    | w/length| Xfr. Data  |
 -----+---------------+-------------+---------+---------+------------+
```

Figure 34. Open for Upload (with Buffer Size) Structured Field Format

6.3.2.5 Open for Message

Messages from the Host to the PC are sent as if they were a data file. When messages are going to be sent the file identifier 'FT: ' replaces 'FT:DATA' in the OPEN request. In abort situations the message is sent without first closing the data file.

```
+---------+-----------+-----------------+------------------------+-----
| x'0023' | x'D00012' | x'010601010403' | x'0A0A0000000011010100' |
|         |           |                 |                        |
| Struct. | Open      | Fixed Parameter | Functions Required     |
| Field   | Request   | Changes File,   | Sequential Insert      |
| Length  |           | etc.            |                        |
+---------+-----------+-----------------+------------------------+--
        --+---------------+---------+------------+
          | x'50055203F0' | x'0309' | c'FT:     ' |
          |               |         |            |
          | Data Not      | File Nm.| Fixed Name |
          | Compressed    | Header  | for File   |
          |               | w/length| Xfr. Data  |
     -----+---------------+---------+------------+
```

Figure 35. Open for Message Structured Field Format

6.3.2.6 Open Acknowledgment

```
+---------+-----------+
| x'0005' | x'D00009' |
|         |           |
| SF      | Open      |
| Length  | Reply     |
|         |           |
+---------+-----------+
```

Figure 36. Open Acknowledgement Structured Field Format

6.3.2.7 Set Cursor and Get

```
+---------+-----------+---------------+---------------+-----
| x'000F' | x'D04511' | x'0105000600' | x'0905010300' |
|         |           |               |               |
| Struct. | Set       | Fixed         | Rel-Pos.      |
| Field   | Cursor    | Parameter     | Parameter     |
| Length  | Request   | 'Rel-Pos.'    | 'Next'        |
+---------+-----------+---------------+---------------+--
        --+---------+-----------+-------------+
          | x'0009' | x'D04611' | x'01040080' |
          |         |           |             |
          | Struct. | Get       | Feed-Back   |
          | Field   | Request   | Requested   |
          | Length  |           |             |
     -----+---------+-----------+-------------+
```

Figure 37. Set Cursor and Get Structured Field Format

6.3.2.8 Upload Data Buffer

```
+---------+-----------+---------+-------------+-----
| x'llll' | x'D04605' | x'6306' | x'nnnnnnnn' |
|         |           |           |           |
| SF len. | Data for  | Record  | Record      |
| Incl.   | Get       | Number  | Number      |
| len fld |           | Header  |             |
+---------+-----------+---------+-------------+--
        --+---------+-------+---------+--------------+
          | x'C080' | x'61' | x'dddd' |              |
          |         |       |         |              |
          | Data    | Begin | Data    |  Data        |
          | Not Com-| Data  | Length  |              |
          | pressed | Code  | plus 5  |              |
     -----+---------+-------+---------+--------------+
```

Figure 38. Upload Data Buffer Structured Field Format

6.3.2.9 Download Data Buffer

```
+---------+-----------+---------------+-----
| x'000A' | x'D04711' | x'0105008000' |
|         |           |               |
| SF      | Insert    | Feedback      |
| Length  | Request   | Requested     |
|         |           |               |
+---------+-----------+---------------+--
     --+---------+-----------+---------+-------+---------+--------------+
       | x'llll' | x'D04704' | x'C080' | x'61' | x'dddd' |              |
       |         |           |         |       |         |              |
       | SF      | Data to   | Data    | Begin | Data    |  Data        |
       | Length  | Insert    | Not Com-| Data  | Length  |              |
       |         |           | pressed | Code  | plus 5  |              |
  -----+---------+-----------+---------+-------+---------+--------------+
```

Figure 39. Download Data Buffer Structured Field Format

6.3.2.10 Message Structured Field Format

Messages sent from the host have the same structured field format as Download Data Buffers, but are interpreted as messages because they follow an open for messages. The format of a Download Data Buffer Structured Field is shown in Figure 39 on page 30.

See 6.5, "Error Messages and Codes" on page 36 for the messages which can appear in the data area of a Message structured field.

6.3.2.11 Data Acknowledgement

```
+---------+-----------+---------+-------------+
| x'000B' | x'D04705' | x'6306' | x'nnnnnnnn' |
|         |           |         |             |
| SF      | Insert    | Record  | Record      |
| Length  | Normal    | Number  | Number      |
|         | Reply     | Header  |             |Positive
+---------+-----------+---------+-------------+
```

Figure 40. Data Acknowledgement Structured Field Format

CLOSE REQUEST

```
+---------+-----------+
| x'0005' | x'D04112' |
|         |           |
| SF      | Close     |
| Length  | Request   |
|         |           |
+---------+-----------+
```

Figure 41. Close Request Structured Field Format

6.3.2.12 Close Acknowledgement

```
+---------+-----------+
| x'0005' | x'D04109' |
|         |           |
| SF      | Close     |
| Length  | Reply     |
|         |           |
+---------+-----------+
```

Figure 42. Close Acknowledgement Structured Field Format

ERROR RESPONSE

```
+---------+-----------+---------+---------+
| x'0009' | x'D0tt08' | x'6904' | x'nnnn' |
|         |           |         |         |
| SF      | Type of   | Error   | Error   |
| Length  | Request   | Code    | Code    |
|         |           | Header  |         |Negative
+---------+-----------+---------+---------+
```

The error is indicated by the following substitutions for tt and nnnn.

Request Type	Error Code
00 Open	0100 Open Failed Exception
	0200 Arrival Sequence Not Allowed
	6000 Command Syntax Error
47 Insert	0200 Arrival Sequence Not Allowed
	3E00 Operation Not Authorized
	6000 Command Syntax Error
45 Set Cursor	0200 Arrival Sequence Not Allowed
	6000 Command Syntax Error
46 Get	0200 Arrival Sequence Not Allowed
	2200 Get Past End of File
	6000 Command Syntax Error
41 Close	6000 Command Syntax Error

Figure 43.   Error Response Codes

6.4 DATA CONVERSIONS AND CHECKS

6.4.1 DATA STREAM CONVERSIONS - CUT AND ASYNCH

For transmission through the screen image buffer with CUT attachment, binary data must first be converted into a string of displayable characters. To accomplish this conversion, the 256 possible bit combinations are divided into quadrants of up to 95 combinations each. Displayable characters are chosen to identify the code within the quadrant, and an additional character code to identify each quadrant. The converted data stream will consist of quadrant identifying characters followed by a string of code characters for codes within that quadrant until a code in a different quadrant is encountered in the unconverted string. This conversion scheme results in, at worst, a 2-for-1 expansion in the data, and by choosing the quadrant such that codes likely to appear adjacent to each other (i.e., alphabetical character codes) within the data stream are in the same quadrant, a much lower expansion will result. In the host TR and TRT may be used in the transformation to avoid a byte by byte scan of characters.

The table works as follows. For downloads, the quadrant identifier is set to the quadrant in which a value for the first character is found. The quadrant identifier and the translated first character are written to the output buffer. Each subsequent character's EBCDIC value is used as an index into each table in order. If the value in the table is non-zero then that value is the 3274 code sent for the character. If the entry is zero, then the next table is tried. If none of the tables have an entry, then an error has occurred. Once the 3274 value for the character is determined, then the quadrant that it's value was found in is compared to the quadrant of the previous character. If they are the same, then the character value is written to the character buffer. If they are different, then the quadrant identifier is written to the buffer followed by the character value.

For uploads, the process is reversed. When a quadrant identifier is recognized, all subsequent characters up to the next quadrant identifier are converted using that quadrant of the table. A value is taken from the buffer and mapped to the value it had when it left the PC. The conversion to ASCII takes place later depending on if the ASCII option was specified on the HFTP invocation.

EBCDIC	ASCII	OTHER 1	OTHER 2	HOST	3274
; (5E/BE)	= (7E/11)	* (5C/BF)	' (7D/12)	(EBCDIC/3274 CODE)	
SP (40)	SP (20)	-	-	40	10
A (C1)	A (41)	(00)	(A0)	C1	A0
B (C2)	B (42)	(01)	(A1)	C2	A1
C (C3)	C (43)	(02)	(EA)	C3	A2
D (C4)	D (44)	(03)	(EB)	C4	A3
E (C5)	E (45)	(04)	(EC)	C5	A4
F (C6)	F (46)	(05)	(ED)	C6	A5
G (C7)	G (47)	(06)	(EE)	C7	A6
H (C8)	H (48)	(07)	(EF)	C8	A7
I (C9)	I (49)	(08)	(E0)	C9	A8
J (D1)	J (4A)	(09)	(E1)	D1	A9
K (D2)	K (4B)	(0A)	(AA)	D2	AA
L (D3)	L (4C)	(0B)	(AB)	D3	AB
M (D4)	M (4D)	(0C)	(AC)	D4	AC
N (D5)	N (4E)	(0D)	(AD)	D5	AD
O (D6)	O (4F)	(0E)	(AE)	D6	AE
P (D7)	P (50)	(0F)	(AF)	D7	AF
Q (D8)	Q (51)	(10)	(B0)	D8	B0
R (D9)	R (52)	(11)	(B1)	D9	B1
S (E2)	S (53)	(12)	(B2)	E2	B2
T (E3)	T (54)	(13)	(B3)	E3	B3
U (E4)	U (55)	(14)	(B4)	E4	B4
V (E5)	V (56)	(15)	(B5)	E5	B5
W (E6)	W (57)	(16)	(B6)	E6	B6
X (E7)	X (58)	(17)	(B7)	E7	B7
Y (E8)	Y (59)	(18)	(B8)	E8	B8
Z (E9)	Z (5A)	(19)	(B9)	E9	B9
a (81)	a (61)	note	(80)	81	80
b (82)	b (62)		-	82	81
c (83)	c (63)		(CA)	83	82
d (84)	d (64)		(CB)	84	83
e (85)	e (65)		(CC)	85	84
f (86)	f (66)		(CD)	86	85
g (87)	g (67)		(CE)	87	86
h (88)	h (68)		(CF)	88	87
i (89)	i (69)		(C0)	89	88
j (91)	j (6A)		-	91	89
k (92)	k (6B)		(8A)	92	8A
l (93)	l (6C)		(8B)	93	8B
m (94)	m (6D)		(8C)	94	8C

Figure 44. Data stream conversion chart (part 1 of 2)

EBCDIC	ASCII	OTHER 1	OTHER 2	HOST	3274
n (95)	n (6E)		(8D)	95	8D
o (96)	o (6F)		(8E)	96	8E
p (97)	p (70)		(8F)	97	8F
q (98)	q (71)		(90)	98	90
r (99)	r (72)		-	99	91
s (A2)	s (73)		(DA)	A2	92
t (A3)	t (74)		(DB)	A3	93
u (A4)	u (75)		(DC)	A4	94
v (A5)	v (76)		(DD)	A5	95
w (A6)	w (77)		(DE)	A6	96
x (A7)	x (78)		(DF)	A7	97
y (A8)	y (79)	abort	(D0)	A8	98
z (A9)	z (7A)	eof	-	A9	99
0 (F0)	0 (30)	(3C)	-	F0	20
1 (F1)	1 (31)	(3D)	(21)	F1	21
2 (F2)	2 (32)	(3E)	(22)	F2	22
3 (F3)	3 (33)	-	(23)	F3	23
4 (F4)	4 (34)	(FA)	(24)	F4	24
5 (F5)	5 (35)	(FB)	(5B)	F5	25
6 (F6)	6 (36)	(FC)	(5C)	F6	26
7 (F7)	7 (37)	(FD)	-	F7	27
8 (F8)	8 (38)	(FE)	(5E)	F8	28
9 (F9)	9 (39)	(FF)	(5F)	F9	29
% (6C)	% (25)	(7B)	-	6C	2E
& (50)	& (26)	(7C)	(9C)	50	30
_ (6D)	' (27)	(7D)	(9D)	6D	2F
((4D)	((28)	(7E)	(9E)	4D	0D
) (5D)	) (29)	(7F)	(9F)	5D	0C
< (4C)	* (2A)	(1A)	(BA)	4C	09
+ (4E)	+ (2B)	(1B)	(BB)	4E	35
, (6B)	, (2C)	(1C)	(BC)	6B	33
- (60)	- (2D)	(1D)	(BD)	60	31
. (4B)	. (2E)	(1E)	(BE)	4B	32
/ (61)	/ (2F)	(1F)	(BF)	61	14
: (7A)	: (3A)	-	(9A)	7A	34
> (6E)	; (3B)	-	(9B)	6E	08
? (6F)	? (3F)	-	-	6F	18

Note: Lower case alpha codes in OTHER 1 quadrant are reserved for control codes.

Figure 45. Data stream conversion chart (part 2 of 2)

6.4.2 CRC COMPUTATION - CUT AND ASYNCH

For CUT attach, the Check Sum value is calculated by a running Exclusive-OR on each byte in the Data portion of the frame. The Check sum is the lower six bits of the result coded as shown described under 6.4.2.1, "Encoding for Frame Sequence, Length, and Check Sum" on page 35.

For Asynchronous attachment, the Check Sum (or Cyclic Redundancy Check) value is a 32-bit sequence based on the 802.5 standard generator polynomial of degree 32. The polynomial is shown below.

```
G(X) =   X**3+ X**26 + X**23 + X**22 + X**16 + X**12 + X**11
                 + X**10 + X**7  + X**5  + X**4  + X**2  + X + 1
```

Figure 46. Check Sum generator polynomial

The Check Sum is calculated only for the information which appears in the data fields of the download (10 - 1913) and upload (12 - 1918) screens.

The Check Sum is calculated on the EBCDIC equivalent of the encoded file transfer data rather than on the actual binary or ASCII data itself. In the terminal this requires a conversion from ASCII to EBCDIC. In IND$FILE the encoded file transfer data is already in EBCDIC.

For purposes of implementation of the referenced 802.5 standard only, the order of transmission of the data field and FCS is taken to be as follows:

+   The data field is transmitted first, followed by the Check Sum.

+   The bytes in the data field and Check Sum are transmitted smallest address first, largest address last.

+   The bits in a given byte are transmitted most significant bit first, least significant bit last.

The Check Sum deviates from the 802.5 standard in that it is sent as a separate field from the data. The 32 bit field is padded on the right with four binary ones to make a 36 bit field, which is then divided into six subfields of six bits each. These six subfields are then converted to valid 3270 graphic codes for transmission to the terminal, using the method described in the next section.

6.4.2.1 Encoding for Frame Sequence, Length, and Check Sum

The Frame Sequence, Length and Check Sum Fields of CUT and Asynchronous frames are encoded for transmission as follows. The number is first broken into six bit fields. For the Frame Sequence number this is done simply by truncating the two most significant zeroes (the Frame Sequence count is modulo 64). For the Length field, the division takes places as follows.

```
Byte 1       Byte 2           The Length is a 12-bit integer
|0000 xxxx|  |yyzz zzzz|          so the high order 4 bits are
     \      /\   |           always 0.
     xxxxyy  zzzzzz
```

The Check Sum for CUT is encoded using just the low order six bits. The Check Sum for Async is broken up into six six bit fields.

For each six bit field, the value of the field is used as an index into the table shown in Figure 47 on page 36 to get the desired code.

Code	Char	Host	3270
0	a	81	80
1	b	82	81
2	c	83	82
3	d	84	83
4	e	85	84
5	f	86	85
6	g	87	86
7	h	88	87
8	i	89	88
9	j	91	89
10	k	92	8A
11	l	93	8B
12	m	94	8C
13	n	95	8D
14	o	96	8E
15	p	97	8F
16	q	98	90
17	r	99	91
18	s	A2	92
19	t	A3	93
20	u	A4	94
21	v	A5	95
22	w	A6	96
23	x	A7	97
24	y	A8	98
25	z	A9	99
26	&	50	30
27	-	60	31
28	.	4B	32
29	,	6B	33
30	:	7A	34
31	+	4E	35
32	A	C1	A0
33	B	C2	A1
34	C	C3	A2
35	D	C4	A3
36	E	C5	A4
37	F	C6	A5
38	G	C7	A6
39	H	C8	A7
40	I	C9	A8
41	J	D1	A9
42	K	D2	AA
43	L	D3	AB
44	M	D4	AC
45	N	D5	AD
46	O	D6	AE
47	P	D7	AF
48	Q	D8	B0
49	R	D9	B1
50	S	E2	B2
51	T	E3	B3
52	U	E4	B4
53	V	E5	B5
54	W	E6	B6
55	X	E7	B7
56	Y	E8	B8
57	Z	E9	B9
58	0	F0	20
59	1	F1	21
60	2	F2	22
61	3	F3	23
62	4	F4	24
63	5	F5	25

Figure 47.   Encoding for Frame Sequence, Length, and Check Sum.

6.5 ERROR MESSAGES AND CODES

The Host File Transfer Program communicates return codes to the PC using messages. All messages from the host have the following format

TRANSxx: <message text>

xx is the return code value. The messages which will be returned by the HFTP are listed below.

+ TRANS03   File transfer complete

+ TRANS04   File transfer complete, with records segmented (See note below)

+ TRANS13   Error writing file to host: file transfer canceled

+ TRANS14   Error reading file from host: file transfer canceled

+ TRANS15   Required host storage unavailable: file transfer canceled

+ TRANS16   Incorrect request code: file transfer canceled

+ TRANS17   Missing or incorrect TSO data set name: file transfer canceled
  (This message for TSO only)

+ TRANS17   Missing or incorrect CMS file identifier: file transfer canceled
  (This message for VM only)

+ TRANS18   Incorrect option specified: file transfer canceled

+ TRANS19 Error reading or writing to host disk: file transfer canceled

+ TRANS28 Invalid option XXXXXXXX: file transfer canceled

+ TRANS29 Invalid option XXXXXXXX with RECEIVE: file transfer canceled

+ TRANS30 Invalid option XXXXXXXX with APPEND: file transfer canceled

+ TRANS31 Invalid option XXXXXXXX without SPACE: file transfer canceled

+ TRANS32 Invalid option XXXXXXXX with PDS: file transfer canceled

+ TRANS33 Only one of TRACKS, CYLINDERS, AVBLOCK allowed: file transfer canceled

+ TRANS34 CMS file not found: file transfer canceled

+ TRANS35 CMS disk is Read-Only: file transfer canceled

+ TRANS36 CMS disk is not accessed: file transfer canceled

+ TRANS37 CMS disk is full: file transfer canceled

+ TRANS99 Host program error code XX XXXXXXXX: file transfer canceled

NOTE - The TRANS04 message only applies to PUT (or PTP) commands when the PC attempts to upload a record with a length greater than the maximum length given for the host file at initialization time. The record is truncated to the maximum length given at initialization time. The user is advised at the completion of the file transfer that this has been done.