ITT3290 File Transfer Protocol and Application Programming Guide Reference Manual

ITT 3290

RCSL No. 42-i 2382

ITT 3290 File Transfer Protocol and Application Programming Guide

Reference Manual

A/S REGNECENTRALEN af 1979

Information Department

December 1983

RCSL No. 42-i2382

Author: Claus Terp

Keywords: ITT 3290, File Transfer Protocol (FTP), Reference Manual

Abstract: This is a Reference Manual for the ITT 3290 File Transfer Protocol and the intelligent ITT 3290 workstation FTP program. The manual contains the detailed specifications constituting the File Transfer Protocol layer, a description of the ITT 3290 workstation FTP program in terms of commands and messages, and general programming guidelines for writing host application software utilizing the File Transfer Protocol. In addition, a description of one such host framework application is included:an IBM-CICS implementation (see Appendix C).

(122 printed pages)

TABLE OF CONTENTS PAGE

TABLE OF CONTENTS (continued) PAGE

TABLE OF CONTENTS (continued) PAGE

TABLE OF CONTENTS (continued) **PAGE**

1. INTRODUCTION

The **ITT 3290 File Transfer Protocol** is an application program which makes your intelligent ITT 3290 workstation capable of exchanging files with an IBM or IBM-compatible mainframe.

The fundamental idea behind FTP is to combine the power of your local CP/M-based software packages with the power of the mainframe. This is obtained by allowing you to switch between using your ITT 3290 workstation software packages locally and connecting your local discette storage to the mainframe. During a FTP session you may send your previously prepared files to the mainframe, or you may receive new files from the mainframe, which you may then locally use for any purpose you may wish.

This manual applies to everyone who has an interest in knowing and using FTP. The main purpose of the manual is to descripe the ITT 3290 and the mainframe environment, and to support the mainframe application programmer in his development of the mainframe application.

Combined with the ITT 3290 File Transfer Program, Operating Guide you will be capable to get full utilization of your ITT 3290 FILE TRANSFER PROTOCOL.

2. GENERAL SYSTEM DESCRIPTION

The FTP environment is an intelligent ITT 3290 workstation equipped with one or two 8 inch diskette drives and with a communication connection to a local control unit (e.g. an ITT 328x Cluster Controller supporting the Courrier Level III Coax Protocol) which in turn must have a communication line to the mainframe. Your workstation must have the ITT 3297 Terminal Emulator installed in Proms.

Your ITT 3290 workstation must have a CP/M system under which you can load various programs. This is only mentioned as a formality as FTP is utilizing the CP/M BDOS facilities.

On the mainframe side one or more application programs "understanding" FTP messages must be installed. As FTP is a pure application program this only means that the mainframe side must be able to exchange screen images with the workstation corresponding to the conventions for FTP communication. So no hardware changes are necessary to run FTP.

On your workstation there are no limitations on files which can be used in a file transfer, except that files which you want to send to the host must exist on one of the two diskettes mounted and files in which you want to receive data from the host must not exist. Chapter 5 contains information about ITT 3290 workstation.

On the mainframe side the conventions are slightly different as the files to which you send or from which you receive will normally be permanently installed. (There may be variations from mainframe to mainframe). This means that you must know the names of the files installed on the mainframe and the rules for their usage. Chapter 4 and appendix C of this manual is intended to contain such mainframe information.

The main system components in the FTP environment are the mainframe with some disk storage, the control unit and the workstation with diskette storage and (optionally) a printer.

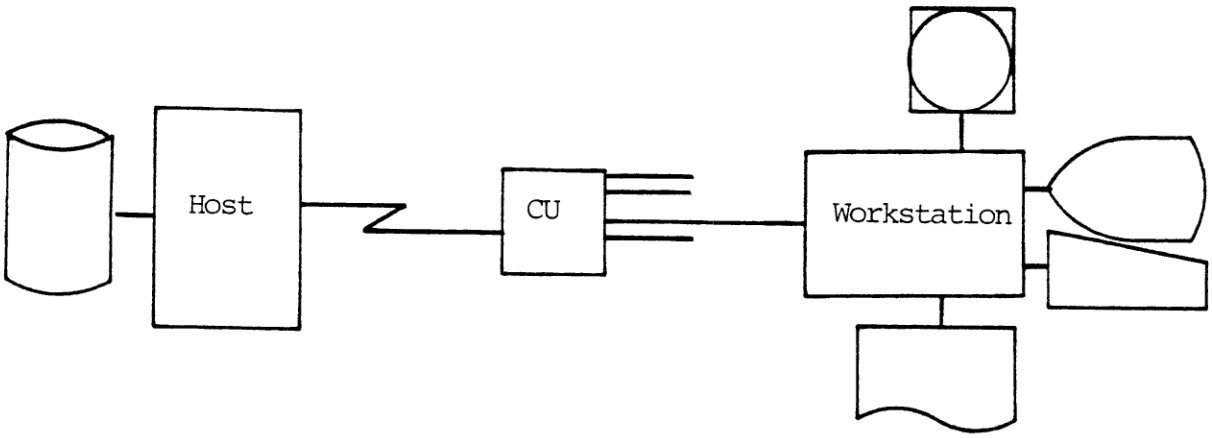


Figure 2.a. Main System Components

To get an overview of the FTP, please find chapter 2 "An overview of the FTP" in the ITT 3290 File Transfer Program, Operating Guide. Detailed information about the File Transfer Protocol may be found in chapter 3 in this manual.

3. FILE TRANSFER PROTOCOL

The File Transfer Protocol defined in this chapter is a communication protocol layer based on the IBM 3270 communication protocol. The IBM 3270 communication protocol defines how IBM or compatible mainframes communicate with terminals by means of device buffers sent between a mainframe and a terminal and the File Transfer Protocol layer is implemented by specifying a fixed structured device buffer containing control information and file data.

In the following sections the File Transfer Protocol device buffer structure and the control information contained in the device buffer is specified.

A file transfer session between a mainframe and an intelligent ITT 3290 workstation may be described as a sequence of steps as shown in figure 3.a.

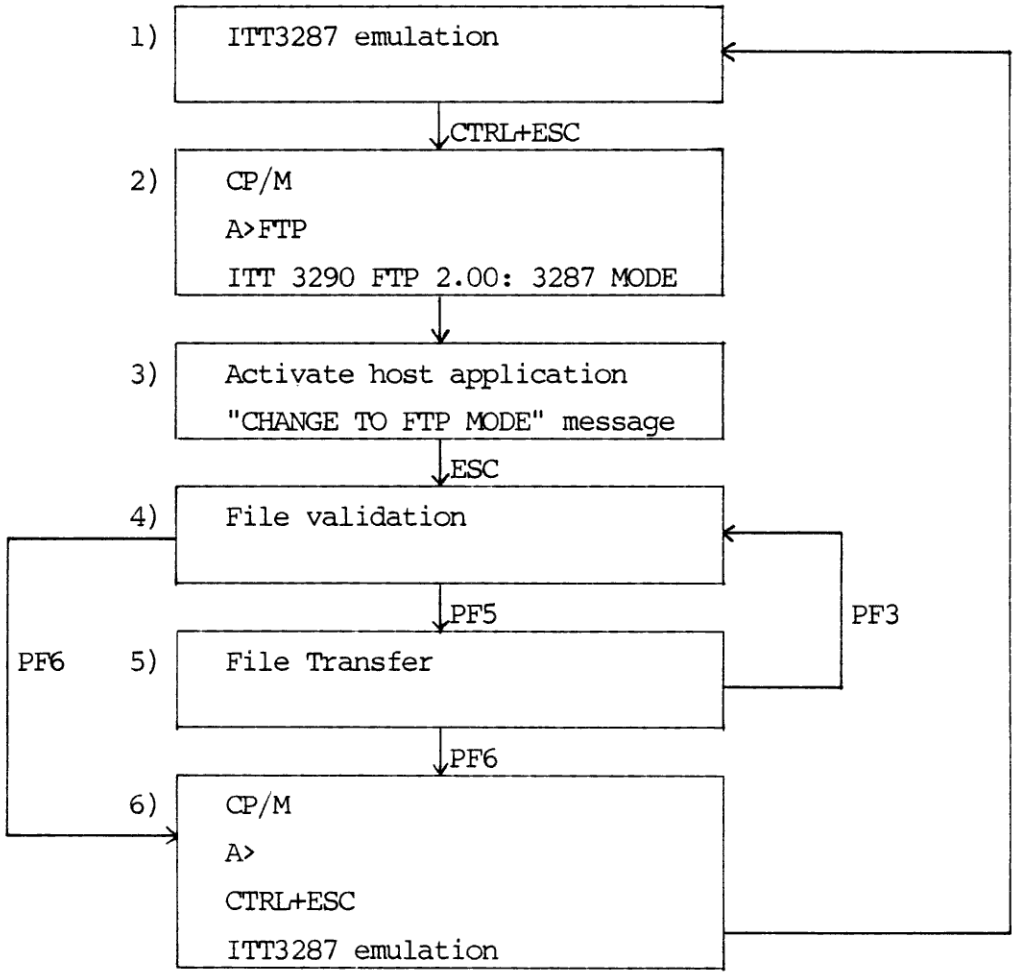


Figure 3.a. File transfer session steps

In the first step standard 3287 emulation takes place between the terminal and the mainframe. In the second step CP/M is booted and FTP (the ITT 3290 workstation program which supports the File Transfer Protocol) is loaded and initially runs in "3287 MODE". In the third step the mainframe File Transfer Application is activated. In the fourth step validation of file control information takes place and in the fifth step the file transfer is executed. In the sixth step CP/M is first booted and then the ITT 3297 Coax Emulator is activated again.

3.1 Device Buffer Format Specification

The host File Transfer Application and the terminal communicate by alternately sending decive buffers to each other, and the device buffers must have precisely the structure described in this section.

A device buffer sent from the 3297 terminal to the host must be structured as if an operator has pressed "send" with the cursor positioned in the upper left corner of the panel.

The device buffer is composed of two fields: the first field is the File Transfer Protocol communication region (corresponding to the first line in the display) and the second field is the file data part (corresponding to the remaining part of the display).

AID	X'7D'	"send"
b1	X'40'	cursor address 1
b2	X'40'	cursor address 2
SBA	X'11'	Set Buffer Address
b1	X'40'	buffer address 1
b2	X'C1'	buffer address 2
<text>	<control line>	78 characters
SBA	X'11'	Set Buffer Address
b1	X'C1'	buffer address 1
b2	C'50'	buffer address 2
(<text>	<file data>	0..1840 characters)

Figure 3.1.a Device buffer structure, from terminal (cu) to host

The device buffer structure for device buffers sent from the terminal (the CU) to the host is shown in figure 3.1.a, using IBM 3270 (BSC) terminology.

A device buffer sent from the host to the terminal is a write command of the type erase/write. The device buffer is composed of two fields: the first field is the File Transfer Protocol communication region (corresponding to the first line in the display) and the second field (corresponding to the remaining part of the display) contains the file data.

The device buffer structure for device buffers sent from the host to the terminal is described with IBM 3270 (BSC) terminology in figure 3.1.b.

CMD	X'F5'	Erase/Write
WCC	X'C6'	Write Control Character
SBA	X'11'	Set Buffer Address
b1	X'40'	buffer address 1
b2	X'40'	buffer address 2
SF	X'1D'	Start Field (1st field)
a	X'40'	Attribute Character
<text>	<control line>	78 characters
SF	X'1D'	Start Field (2nd field)
a	X'40'	Attribute Character
(<text>	<file data>	0..1840 characters)

Figure 3.1.b Device buffer structure, from host to terminal (cu)

3.2 Communication Region

The File Transfer Protocol communication region contains the control information necessary to manage the communication flow between the host File Transfer Application and FTP (the terminal program). The communication region has a fixed structure as described in section 3.3 and is placed as the first 80 characters of the device buffer, corresponding to the first line on the display.

One of the fields in the communication region is the Function Code field

which specifies the meaning of the device buffer. Thus the contents of the Function Code field is a command from the sender to the receiver of the device buffer.

3.2.1 Function Code Survey

In order to explain the File Transfer Protocol structure it is comprehensible to give an overview of the function codes (commands) defined. A detailed description of the logical meaning of each function code may be found in section 3.4.

Function Code Field		
Code	Meaning	Definition
"VAF"	Validate File	3.4.1
"VAE"	Validate File End	3.4.2
"SOF"	Start Of File Transfer	3.4.3
"TFD"	Transfer Data	3.4.4
"EOF"	End of File	3.4.5
"REQ"	Request Next Block (device buffer)	3.4.6
"STP"	Stop FTP Handling	3.4.7
"ENQ"	Enquiry ("What?")	3.4.8
"AB1"	Abort 1 (i.e. delete received file)	3.4.9
"AB2"	Abort 2 (i.e. close received file)	3.4.10

Figure 3.2.1.a Function Code Survey

With a specific function code in the Function Code field the entire device buffer may be regarded as a command with a set of parameters contained in the communication region and possibly some file data in the remaining part of the device buffer.

3.2.2 File Transfer Protocol Survey

The File Transfer Protocol defines that the host File Transfer Application and the terminal must alternately send device buffers to each other (you may think of the device buffer as a baton repeatedly exchanged between the host and the terminal).

A file transfer is always initialized with a validation of the file control information defining the file transfer. The terminal sends a VAF command

(VAlidate File) to the host File Transfer Application and the communication region contains all the information needed to accept or reject the file transfer. As a response to the received VAF command the host application sends a REQ command (REQuest) to the terminal. This command contains a status code indicating to the terminal if the file transfer can be started or not and at the same time the command serves the purpose of allowing the terminal to proceed with the next command.

In general, the sequence is that the terminal sends a command (a message) to the host and the host sends a command (an answer) with a status (a result) to the terminal. The status codes defined in the File Transfer Protocol is described in section 3.6 of this chapter.

The terminal may send one or more VAF commands to the host before a file transfer (or a sequence of transfers) is commenced. This is shown in figure 3.2.2.a where the terminal is initially at point 1.

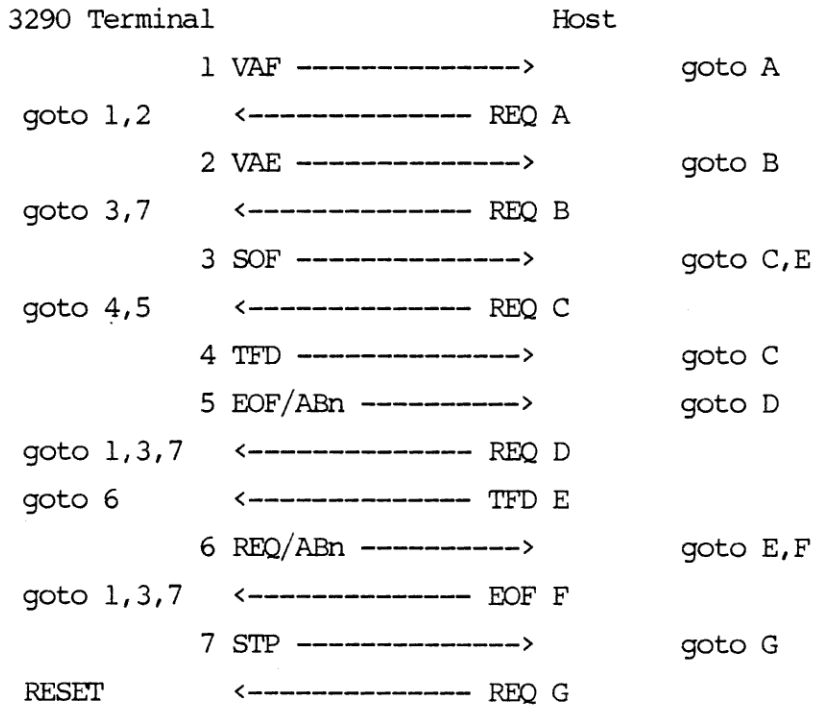


Figure 3.2.2.a File Transfer Protocol Survey

The figure shows that the terminal, being at point 1, sends a VAF command to the host which will go to point A and send a REQ command to the terminal. As described above all answers from the host contain a status and this may be

used by the terminal operator to decide e.g. whether another VAF command or a VAE command (VAlidate End) should be sent.

When the terminal operator decides to start the file transfer(s) the terminal sends the VAE command, instructing the host that a file transfer may soon be started. The host will go to point B and send a REQ command. Now the terminal may go to point 3 and send a SOF command (Start Of File transfer) telling the host the names of the files involved and the transfer direction.

When the host receives the SOF command the transfer direction is controlled. If the transfer is from terminal to host the host will go to point C, otherwise the host will go to point E.

At point C the host sends a REQ command and the terminal will go to point 4 and send a TFD command (Transfer File Data) with file data to the host. The sequence of REQ and TFD commands is repeated until the terminal sends the last part of the file. In this case the terminal goes to point 5 and sends an EOF command (End Of File data) with the last part of the file data to the host. When the host has answered with a REQ command the terminal may either go to point 1, starting another file control information validation, to point 3, starting another (previously validated) file transfer or to point 7, sending a STP command (SToP File Transfer Protocol mode).

If the transfer is from the host to the terminal the host goes to point E and sends a TFD command with file data to the terminal. The terminal goes to point 6 and sends a REQ command. This sequence is similarly repeated until the host sends the last part of the file data with an EOF command. The terminal may then go to points 1, 3 or 7 as described above.

As indicated at the points 5 and 6 the terminal may send an AB1 or AB2 command instructing the host that the file transfer should be terminated. This is always initialized by the terminal operator, normally after an error message has been displayed on the terminal. The reason for the error message may be a status received from the host or local problems on the terminal.

One more command (not shown in figure 3.2.2.a) may be sent from the terminal. If the terminal has not received an answer one minute after the latest command was sent from the terminal, then the terminal sends an ENQ command

(ENQuiry) instructing the host to repeat transmission of the latest device buffer sent from the host. Section 3.7 contains a description of the error recovery strategy used.

3.3 Communication Region Field Specifications

The File Transfer Protocol communication region is a fixed record structure as previously described, and the contents of the File Transfer Protocol communication region fields depend on the type of function code (command) used in the actual operation.

Communication Region Field			
Position	Length	Name	Definition
0 - 0	1	Attribute character position	
1 - 8	8	FTP Image Identification	(A) 3.3.1
9 - 9	1	Space	
10 - 10	1	Transfer Direction	(B) 3.3.2
11 - 11	1	Space	
12 - 19	8	Host File Name	(C) 3.3.3
20 - 20	1	Space	
21 - 34	14	3297 File Name	(D) 3.3.4
35 - 35	1	Space	
36 - 41	6	File Size (in bytes)	(E) 3.3.5
42 - 42	1	Space	
43 - 48	6	No. of Transferred Characters	(F) 3.3.6
49 - 49	1	Space	
50 - 51	2	Status Code	(G) 3.3.7
52 - 52	1	Space	
53 - 55	3	Function Code	(H) 3.3.8
56 - 56	1	Space	
57 - 57	1	"Block No" Identification	(I) 3.3.9
58 - 58	1	Space	
59 - 59	1	Data Format	(J) 3.3.10
60 - 60	1	Space	
61 - 64	4	Current Blocksize	(K) 3.3.11
65 - 65	1	Space	
66 - 67	2	Blocking Factor	(L) 3.3.12
68 - 68	1	Space	
69 - 72	4	Host File Record Length	(M) 3.3.13
73 - 73	1	Space	
74 - 75	2	Timing Factor (in seconds)	(N) 3.3.14
76 - 78	3	Spaces	
79 - 79	1	Attribute Character Position ('*')	
80 - 1919	1840	Data Area	(O) 3.3.15

Figure 3.3.a Communication Region fields

The structure of the communication region is shown in figure 3.3.a and the meaning of the various fields is described in the following subsections.

3.3.1 FTP Image Identification (1-8)

This field contains a unique character sequence enabling the host File Transfer Application and the terminal (FTP) to accept the device buffer as a valid File Transfer Protocol element.

The character-sequence is "FTPIMAGE".

3.3.2 Transfer Direction (10-10)

This field contains information about the file transfer direction.

"T" (Transmit) defines the file transfer direction from the 3297 terminal to the host.
"R" (Receive) defines the opposite direction (host to terminal).

3.3.3 Host File Name (12-19)

This field contains the name of the host file used in the file transfer (e.g. "ABC123XX").

3.3.4 3297 File Name (21-34)

This field contains the ITT 3290 terminal CP/M file name (perhaps the printer identification name: 'FPRINTER'), including diskette volume specification (e.g. "B:MYTEXT.TXT").

3.3.5 File Size (36-41)

This field contains the size of the file to be transferred. The size is

defined as the number of characters, e.g. "001280". Maximum value is 999999 characters.

3.3.6 No. of Characters Transferred (43-48)

This field contains the number of characters transferred in the current file transfer, e.g. "018400", (exclusive of the data contents of the current device buffer).

3.3.7 Status Code (50-51)

This field contains information about the current state of a data transfer, e.g. "00" (meaning OK). See section 3.6 for a complete list of defined status codes.

3.3.8 Function Code (53-55)

This field contains information about the interpretation of the current device buffer, implicitly defining which fields in its communication region carry relevant information, e.g. "REQ", "TFD", "ENQ".

3.3.9 "Block No" Identification (57-57)

This field contains a device buffer sequence number modulo 2. The host application and the terminal use this field to number each delivered device buffer (except "ENQ" device buffers) for end to end control reasons, i.e. "0", "1" alternating.

3.3.10 Data Format (59-59)

This field describes the file transfer data format.

"S" defines a file transport using the transparent "split"-code data format,

"A" defines a file transport using the normal ASCII-EBCDIC data format.

3.3.11 Current Blocksize (61-64)

This field contains information about the number of characters which actually are valid in the data part of the device buffer , e.g. "1860", "0087". The number of characters need not be a multiple of the number of 80-character device buffer lines.

3.3.12 Blocking Factor (66-67)

This field describes the number of lines of file data in the device buffer which may be transferred in each device buffer, "01" to "23".

3.3.13 Host File Record Length (69-72)

This field specifies the record length which will be used when the host file is written, e.g. "0080", "1024". All host file parameters are maintained at the host site. Recommended host record length is 128 bytes corresponding to the CP/M record length (sector size).

3.3.14 Timing Factor (74-75)

This field specifies the number of seconds a device buffer is delayed at the terminal before transmission and serves the purpose of preventing a terminal from monopolizing the telecommunication capacity. The value range is 00 to 99.

3.3.15 Data Area (80-1919)

This field carries the file data to be transferred. The field may be viewed as a structured record consisting of 23 sub-fields of 80 characters each, corresponding to the lower 23 lines of the 24-line device buffer.

3.4 Logical Functions

The function code field in the communication region of a device buffer is an essential part of the File Transfer Protocol. The receiver (host or terminal) of a device buffer corresponding to the File Transfer Protocol definition is able to act properly by interpreting the communication region information in accordance with the actual function code used.

The following function codes are defined:

Function Code Field		
Code	Meaning	Definition
"VAF"	Validate File	3.4.1
"VAE"	Validate File End	3.4.2
"SOF"	Start Of File Transfer	3.4.3
"TFD"	Transfer Data	3.4.4
"EOF"	End of File	3.4.5
"REQ"	Request Next Block (device buffer)	3.4.6
"STP"	Stop FTP Handling	3.4.7
"ENQ"	Enquiry ("What?")	3.4.8
"AB1"	Abort 1 (i.e. delete received file)	3.4.9
"AB2"	Abort 2 (i.e. close received file)	3.4.10

Figure 3.4.a Function Codes

In the following subsections the mandatory fields are specified for each function code (all mandatory fields are boldfaced). Fields which are not absolutely necessary may, however, contain relevant information, especially useful during a debugging/trace session.

3.4.1 VAF - Validate File

Information from terminal to host. The communication region contains data from 3297 terminal to host concerning validation of a file transfer (terminal to host or host to terminal).

3.4.1.1 VAF Transfer from Terminal to Host

Information from terminal to host concerning file transfer from terminal to host.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**T**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	**36-41**	**File Size**	**<filesize>**
F	43-48	No of Transferred Characters	<transferred>
G	50-51	Status Code	<status>
H	**53-55**	**Transfer Code**	**VAF**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	59-59	Data Format	S/A
K	61-64	Current Blocksize	<bytes>
L	66-67	Blocking Factor	1..23
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	80-	Data Area	<devicebufferlines>

3.4.1.2 VAF Transfer from Host to Terminal

Information from terminal to host concerning file transfer from host to terminal.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**R**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	36-41	File Size	<filesize>
F	43-48	No of Transferred Characters	<transferred>
G	50-51	Status Code	<status>
H	**53-55**	**Transfer Code**	**VAF**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	59-59	Data Format	S/A
K	61-64	Current Blocksize	<bytes>
L	66-67	Blocking Factor	1..23
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	80-	Data Area	<devicebufferlines>

3.4.2 VAE - Validate File End

Information from terminal to host. The command is sent when file validation is finished.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	10-10	Transfer Direction	T/R
C	12-19	Host File Name	<hostfilename>
D	21-34	3297 File Name	<3297filename>
E	36-41	File Size	<filesize>
F	43-48	No of Transferred Characters	<transferred>
G	50-51	Status Code	<status>
H	**53-55**	**Transfer Code**	**VAE**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	59-59	Data Format	S/A
K	61-64	Current Blocksize	<bytes>
L	66-67	Blocking Factor	1..23
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	80-	Data Area	<devicebufferlines>

3.4.3 SOF - Start Of File Transfer

Information sent from terminal to host when a file transfer is to be started. The necessary fields depend on the direction of file transfer.

3.4.3.1 SOF Transfer from Terminal to Host

Information from terminal to host, concerning the start of a file transfer from terminal to host.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**T**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	**36-41**	**File Size**	**<filesize>**
F	43-48	No of Transferred Characters	<transferred>
G	50-51	Status Code	<status>
H	**53-55**	**Transfer Code**	**SOF**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	**59-59**	**Data Format**	**S/A**
K	61-64	Current Blocksize	<bytes>
L	**66-67**	**Blocking Factor**	**1..23**
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	80-	Data Area	<devicebufferlines>

3.4.3.2 SOF Transfer from Host to Terminal

Information from terminal to host, concerning the start of a file transfer from host to terminal.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**R**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	**36-41**	**File Size**	**<filesize>**
F	43-48	No of Transferred Characters	<transferred>
G	50-51	Status Code	<status>
H	**53-55**	**Transfer Code**	**SOF**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	**59-59**	**Data Format**	**S/A**
K	61-64	Current Blocksize	<bytes>
L	**66-67**	**Blocking Factor**	**1..23**
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	80-	Data Area	<devicebufferlines>

3.4.4 TFD - Transfer Data

The device buffer includes data.

3.4.4.1 TFD from Terminal to Host

Information from terminal to host. The device buffer includes data.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**T**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	**36-41**	**File Size**	**<filesize>**
F	**43-48**	**No of Transferred Characters**	**<transferred>**
G	50-51	Status Code	<status>
H	**53-55**	**Transfer Code**	**TFD**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	**59-59**	**Data Format**	**S/A**
K	**61-64**	**Current Blocksize**	**<bytes>**
L	**66-67**	**Blocking Factor**	**1..23**
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	**80-**	**Data Area**	**<devicebufferlines>**

3.4.4.2 TFD from Host to Terminal

Information from host to terminal. The device buffer includes data.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**R**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	**36-41**	**File Size**	**<filesize>**
F	**43-48**	**No of Transferred Characters**	**<transferred>**
G	**50-51**	**Status Code**	**<status>**
H	**53-55**	**Transfer Code**	**TFD**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	**59-59**	**Data Format**	**S/A**
K	**61-64**	**Current Blocksize**	**<bytes>**
L	66-67	Blocking Factor	1..23
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	**80-**	**Data Area**	**<devicebufferlines>**

3.4.5 EOF - End of File

3.4.5.1 EOF from Terminal to Host

Information from terminal to host.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**T**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	**36-41**	**File Size**	**<filesize>**
F	**43-48**	**No of Transferred Characters**	**<transferred>**
G	50-51	Status Code	<status>
H	**53-55**	**Transfer Code**	**EOF**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	**59-59**	**Data Format**	**S/A**
K	**61-64**	**Current Blocksize**	**<bytes>**
L	66-67	Blocking Factor	1..23
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	**80-**	**Data Area**	**<devicebufferlines>**

3.4.5.2 EOF from Host to Terminal

Information from host to terminal.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**R**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	21-34	3297 File Name	<3297filename>
E	36-41	File Size	<filesize>
F	**43-48**	**No of Transferred Characters**	**<transferred>**
G	**50-51**	**Status Code**	**<status>**
H	**53-55**	**Transfer Code**	**EOF**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	**59-59**	**Data Format**	**S/A**
K	**61-64**	**Current Blocksize**	**<bytes>**
L	66-67	Blocking Factor	1..23
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	**80-**	**Data Area**	**<devicebufferlines>**

3.4.6 REQ - Request Next Block

3.4.6.1 REQ from Terminal to Host

Information from terminal to host, REQuesting the next TFD device buffer from the host.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**R**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	36-41	File Size	<filesize>
F	**43-48**	**No of Transferred Characters**	**<transferred>**
G	50-51	Status Code	<status>
H	**53-55**	**Transfer Code**	**REQ**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	**59-59**	**Data Format**	**S/A**
K	61-64	Current Blocksize	<bytes>
L	66-67	Blocking Factor	1..23
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	80-	Data Area	<devicebufferlines>

3.4.6.2 REQ From Host to Terminal

3.4.6.2.1 REQ Next TFD/EOF

Information from host to terminal, REQuesting the next TFD device buffer from the terminal.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**T**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	36-41	File Size	<filesize>
F	43-48	No of Transferred Characters	<transferred>
G	**50-51**	**Status Code**	**<status>**
H	**53-55**	**Transfer Code**	**REQ**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	**59-59**	**Data Format**	**S/A**
K	61-64	Current Blocksize	<bytes>
L	**66-67**	**Blocking Factor**	**1..23**
M	**69-72**	**Host File Record Length**	**<recordlength>**
N	**74-75**	**Timing Factor**	**<seconds>**
O	80-	Data Area	<devicebufferlines>

3.4.6.2.2 REQ Next After VAE/STP/AB1/AB2

Information from host to terminal, REQuesting next device buffer from the terminal (host answer to a received VAE/STP/AB1/AB2).

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	10-10	Transfer Direction	R
C	12-19	Host File Name	<hostfilename>
D	21-34	3297 File Name	<3297filename>
E	36-41	File Size	<filesize>
F	43-48	No of Transferred Characters	<transferred>
G	**50-51**	**Status Code**	**<status>**
H	**53-55**	**Transfer Code**	**REQ**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	59-59	Data Format	S/A
K	61-64	Current Blocksize	<bytes>
L	66-67	Blocking Factor	1..23
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	80-	Data Area	<devicebufferlines>

3.4.6.2.3 REQ Next VAF/VAE

Information from host to terminal, REQuesting next VAF (or VAE) from the terminal. The REQ is an answer to a VAF describing a file transfer from the host to the terminal.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**R**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	**36-41**	**File Size**	**<filesize>**
F	43-48	No of Transferred Characters	<transferred>
G	**50-51**	**Status Code**	**<status>**
H	**53-55**	**Transfer Code**	**REQ**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	**59-59**	**Data Format**	**S/A**
K	61-64	Current Blocksize	<bytes>
L	**66-67**	**Blocking Factor**	**1..23**
M	**69-72**	**Host File Record Length**	**<recordlength>**
N	**74-75**	**Timing Factor**	**<seconds>**
O	80-	Data Area	<devicebufferlines>

3.4.6.2.4 REQ Next VAF

Information from host to terminal, REQuesting next VAF (or VAE) from the terminal. The REQ is an answer to a VAF describing a file transfer from the terminal to the host.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**T**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	36-41	File Size	<filesize>
F	43-48	No of Transferred Characters	<transferred>
G	**50-51**	**Status Code**	**<status>**
H	**53-55**	**Transfer Code**	**REQ**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	**59-59**	**Data Format**	**S/A**
K	61-64	Current Blocksize	<bytes>
L	**66-67**	**Blocking Factor**	**1..23**
M	**69-72**	**Host File Record Length**	**<recordlength>**
N	**74-75**	**Timing Factor**	**<seconds>**
O	80-	Data Area	<devicebufferlines>

3.4.7 STP - Stop FTP Handling

Information from terminal to host.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	10-10	Transfer Direction	T/R
C	12-19	Host File Name	<hostfilename>
D	21-34	3297 File Name	<3297filename>
E	36-41	File Size	<filesize>
F	43-48	No of Transferred Characters	<transferred>
G	50-51	Status Code	<status>
H	**53-55**	**Transfer Code**	**STP**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	59-59	Data Format	S/A
K	61-64	Current Blocksize	<bytes>
L	66-67	Blocking Factor	1..23
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	80-	Data Area	<devicebufferlines>

3.4.8 ENQ - Enquiry

Information from terminal to host. The terminal requests that the host re-send the last device buffer sent.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	10-10	Transfer Direction	T/R
C	12-19	Host File Name	<hostfilename>
D	21-34	3297 File Name	<3297filename>
E	36-41	File Size	<filesize>
F	43-48	No of Transferred Characters	<transferred>
G	50-51	Status Code	<status>
H	**53-55**	**Transfer Code**	**ENQ**
I	57-57	"Blockno" Identification	0/1
J	59-59	Data Format	S/A
K	61-64	Current Blocksize	<bytes>
L	66-67	Blocking Factor	1..23
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	80-	Data Area	<devicebufferlines>

3.4.9 AB1 - Abort 1

Information from terminal to host. The receiving part should delete the received data as if the file transfer was never started.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**T**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	36-41	File Size	<filesize>
F	43-48	No of Transferred Characters	<transferred>
G	50-51	Status Code	<status>
H	**53-55**	**Transfer Code**	**AB1**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	59-59	Data Format	S/A
K	61-64	Current Blocksize	<bytes>
L	66-67	Blocking Factor	1..23
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	80-	Data Area	<devicebufferlines>

3.4.10 AB2 - Abort 2

Information from terminal to host. The receiving part should close the file with file length equal to number of received characters, preserving the data received.

A	**1- 8**	**FTP Image Identification**	**FTPIMAGE**
B	**10-10**	**Transfer Direction**	**T/R**
C	**12-19**	**Host File Name**	**<hostfilename>**
D	**21-34**	**3297 File Name**	**<3297filename>**
E	36-41	File Size	<filesize>
F	43-48	No of Transferred Characters	<transferred>
G	50-51	Status Code	<status>
H	**53-55**	**Transfer Code**	**AB2**
I	**57-57**	**"Blockno" Identification**	**0/1**
J	59-59	Data Format	S/A
K	61-64	Current Blocksize	<bytes>
L	66-67	Blocking Factor	1..23
M	69-72	Host File Record Length	<recordlength>
N	74-75	Timing Factor	<seconds>
O	80-	Data Area	<devicebufferlines>

3.5 Communication Sequence Samples

This section describes sequences of commands (messages and answers) and associated status codes sent between the terminal and the host. The status codes are described in section 3.6.

File validation

Terminal message	Host answer	and possible status codes
VAF	REQ	00,01,02,03,04,06,07,09 or 99
VAE	REQ	00,07 or 99
STP	REQ	00,07 or 99

Figure 3.5.a File validation sequence

The general host reaction when an unexpected function code is received from the terminal is to answer with function code REQ and status code 09.

File transfer from terminal to host

Terminal message	Host answer	and possible status codes
SOF	REQ	00,01,03,06,07,09 or 99
TFD + data	REQ	00,02,05,07,09 or 99
EOF + data	REQ	00,07 or 99
AB1	REQ	00,02,07 or 99
AB2	REQ	00,02,07 or 99

Figure 3.5.b File transfer from terminal to host

When the host has answered with function code REQ and status code 00 to an EOF, AB1 or AB2 message from the terminal, the terminal may commence with a VAF, VAE, SOF or STP message.

File transfer from host to terminal

Terminal message	Host answer	and possible status codes
SOF	TFD + data	00,01,04,06,07,09 or 99
	or EOF + data	00
REQ	TFD + data	00,01,04,06,07,09 or 99
	or EOF + data	00
AB1	EOF	00
AB2	EOF	00

Figure 3.5.c File transfer from host to terminal

When the host has answered with function code EOF and status code 00 the terminal may commence with a VAF, VAE, SOF or STP message.

3.6 Status Codes

The contents of the Status Code Field in the device buffer communication region informs the terminal about the result of the previous message. The status codes are shown in figure 3.6.a.

Status Code Field	
Code	Meaning
00	Normal response (ok)
01	Hostfile is either undefined or closed
02	Invalid transfer direction
03	Hostfile is not empty
04	Hostfile is empty
05	No more host file space
06	No table entry exists for the file in question
07	Block-id received twice
09	Invalid transfer code
98	No inputdata received (device buffer empty)
99	Unrecoverable error

Figure 3.6.a Status Codes

The status code is generated by the host in each answer sent to the terminal. Only the status code values described above are defined in the File Transfer Protocol.

3.7 Error Recovery

In case of special communication line circumstances (e.g. too many retransmissions) a device buffer exchange between the host FTP application and the 3297 terminal FTP system might fail.

If the host application, due to errors on the line, does not receive a terminal device buffer, or the host answer is not received by the terminal the error recovery procedure is used.

If an answer from the host (a device buffer) is not received by the ter-

minal within a certain time limit (i.e. 1 minute) the terminal will activate the error recovery procedure: The terminal will transmit a device buffer with the special transfer code ENQ (without blocknumber identification) to the host application. If the host application receives this command, it must respond by resending the last device buffer transmitted to the terminal.

The terminal will send up to three ENQ commands before the communication is given up.

4. HOST FRAMEWORK APPLICATION PROGRAMMING GUIDE

The purpose of this chapter is to describe the general structure of a host application utilizing the File Transfer Protocol. This user supplied application program may be implemented in different programming languages on different hosts which may also provide different operating system environments. Thus a general description of the host application should not be too strictly connected to any specific host system.

In this host framework application programming guide we have chosen to describe the host application program structure in a pseudo-pascal language which should be easy to understand. Rather than showing calls of actual operating system function calls we use virtual calls combined with semantic descriptions of the meaning of these calls. It should then be easy for the skilled systems programmer to substitute these virtual calls with the actual operating system function calls to be used on a particular installation.

To simplify programming logic a non-reentrant program servicing only one ITT3297 workstation running FTP (File Transfer Program) is shown in the program listing shown in section 4.2. Re-entrancy or multiple terminal support is supposed to be taken care of by the operating system environment.

4.1 Host Application Program Description

From the introductory chapters and from chapter 3 on the File Transfer Protocol you should have a rather deep understanding of the various tasks which the host application program must take care of. These are mainly to validate the information used to define a file transfer, and to do the file transfer. Thus the host application program may be in state "validation", in state "prepare_transmit" or in state "transmission". The program is constructed as a state-event-action machine in which the following three major actions are repeatedly executed. First, the program waits for a command (an event), which is received as a full or partial screen image with a pre-determined structure and content. Second, the program executes an action in accordance with the current state and the event received, possibly changing the state. And third, the program sends an answer in the shape of a full or partial screen image with the same pre-determined structure, but with a suitably modified content.

The host application program shown in section 4.2 is based on the assumption that a table describing valid host files and attributes associated with the host files is existing and accessible from the application program. The table is supposed to contain at least the following attributes associated with each file. The host file name. The types of transmission which may be performed, where type R means that the host file may be sent, type T means that data may be received in the host file and type B (both) means that both transmission and reception may be performed. The transmission format which may be either T for text transfer mode or S for transparent transfer mode (also called split mode). The record length, in case the host application is storing data in fixed length records (in which case the record length 128 bytes is recommended). The blocking factor, which describes the number of text lines (80 bytes of file data) transmitted in each screen image. The pacing factor or delay which the ITT3297 workstation must obey from the reception of a screen image and until the next screen image is sent.

In the host framework application described in appendix C all host file names and host file attributes are collected in a table denoted the DFT - Destination File Table. A slightly expanded table structure is used in the host application program in section 4.2.

The host application must have a strategy for determining whether a host file is empty or full in order to e.g. prevent data received from one ITT3297 workstation from being overwritten with data from another ITT3297 workstation before the host file data has been used by some host application. The File Transfer Protocol does not define this but provides a set of status codes which may be communicated to the ITT3297 workstation. If for example transmission from an empty host file is attempted the host application may answer with status code "04" meaning "host file empty". A complete list of status codes defined in the File Transfer Protocol may be found in section 3.6.

It is the host systems programmers responsibility to implement a strategy suitable for the host computer operating system and environment. The strategy must define how the state of a host file switches from empty to full, and reverse. In the host application program in section 4.2 the following strategy, which is also the strategy followed by the host framework application described in appendix C, is used: The host file transfer type (T=terminal transmits, R=terminal receives, B=terminal may transmit and receive) is compared to

the transfer direction indicated by the terminal. In case of conflict the terminal request is answered with status "transfer direction invalid". The host file is then checked for data. If data is present then attempts to overwrite the data (T) is answered with status "host file not empty". If data is not present then attempts to read the data (R) is answered with status "host file empty".

The host application accepts to write data only in an empty host file and the state of the file is changed to full upon file transfer termination. The host application accepts to read data only from a non-empty host file and the state of the file is not changed upon file transfer termination. Thus a host file can only be set to empty by some other host application.

The host application in section 4.2 assumes that the ITT3297 workstation obeys the File Transfer Protocol, both in respect to single screen image structures and contents, and in respect to sequences of screen images. Thus only a limited validation is performed on received screen images and screen images to be sent to the terminal are created by modifying the received screen images. In a real host file transfer application a more detailed validation ought to be carried out in order to protect the host application against erroneous data. Similarly, the host application should fill in all fields in the command region part of the screen images to ensure that all field contents are legal. The command region structure is shown in detail in chapter 3.

The host application program works with the EBCDIC alphabet while the ITT3297 workstation works with the ASCII alphabet and the necessary code conversion is carried out in the CU. As the File Transfer Protocol allows transmission of binary data in the Transparent Transfer Mode files with binary data may exist on both the host side and on the workstation side. This is no problem as long as the binary data is transmitted in Transparent Transfer Mode as each binary byte is carried in two bytes which both take legal values with respect to the EBCDIC and ASCII alphabets. Thus no conflicts with the IBM 3270 protocol control codes are possible. But, if an operator attempts to transfer a binary file in Text Transfer Mode conflicts are possible. Therefore, the host application must prevent problems by filtering out all illegal characters before screen images are sent to a workstation. A similar process takes place in the ITT3297 workstation.

Except for the characters 0B, 0C, 0D, 15, 19, 1C, 1E, 27 and 3F (hexadecimal values) all character values in the range 00 to 3F should be converted to spaces (hexadecimal 40), before they are transmitted.

4.2 Host Application Program Listing

```
0010        program host_application;
0020        const
0030
0040           line_10_text   = "****************************************";
0050           line_11_text   = "*                                      *";
0060           line_12_text   = "*        PLEASE ENTER FTP-MODE         *";
0070           line_13_text   = "*      BY PRESSING THE <ESC> KEY       *";
0080           line_14_text   = "*                                      *";
0090           line_15_text   = "****************************************";
0100
0110           ftpimage       = "FTPIMAGE";
0120
0130           host_receive   = "T"; <* terminal transmits          *>
0140           host_transmit  = "R"; <* terminal receives           *>
0150           host_rec_tr    = "B"; <* both directions allowed     *>
0160
0170           vaf            = "VAF"; <* validate file            *>
0180           vae            = "VAE"; <* validate end             *>
0190           sof            = "SOF"; <* start of file transfer   *>
0200           req            = "REQ"; <* request                 *>
0210           tfd            = "TFD"; <* transfer file data       *>
0220           eof            = "EOF"; <* end of file transfer     *>
0230           stp            = "STP"; <* stop ftp mode           *>
0240           enq            = "ENQ"; <* enquiry, after timeout   *>
0250           ab1            = "AB1"; <* abort 1, close and delete *>
0260           ab2            = "AB2"; <* abort 2, close           *>
0270
0280           text_format    = "T"; <* text transfer mode         *>
0290           split_format   = "S"; <* transparent transfer mode  *>
0300
0310           status_00      = "00"; <* normal response           *>
0320           status_01      = "01"; <* host file undefined/closed *>
0330           status_02      = "02"; <* transfer direction invalid *>
0340           status_03      = "03"; <* host file not empty       *>
0350           status_04      = "04"; <* host file is empty        *>
0360           status_05      = "05"; <* no more space on host file *>
0370           status_06      = "06"; <* host file unknown         *>
0380           status_07      = "07"; <* block id received twice   *>
0390           status_09      = "09"; <* invalid transfer code     *>
0400           status_98      = "98"; <* no input data received    *>
0410           status_99      = "99"; <* unrecoverable error       *>
0420
```

```
1000        type
1010
1020           states          = (validation,prepare_transmit,
1030                              transmission,termination);
1040
1050           bytes_2         = array (.1..2.)  of byte;
1060           bytes_3         = array (.1..3.)  of byte;
1070           bytes_4         = array (.1..4.)  of byte;
1080           bytes_6         = array (.1..6.)  of byte;
1090           bytes_8         = array (.1..8.)  of byte;
1100           bytes_14        = array (.1..14.) of byte;
1110
1120           text_field      = array (.1..40.)  of byte;
1130           fill_1_type     = array (.1..739.) of byte;
1140           fill_2_type     = array (.1..40.)  of byte;
1150           fill_3_type     = array (.1..740.) of byte;
1160
1170           login_message   = record
1180                                start_attr:     byte;
1190                                fill_1:         fill_1_type;
1200                                line_10:        text_field;
1210                                fill_2:         fill_2_type;
1220                                line_11:        text_field;
1230                                fill_3:         fill_2_type;
1240                                line_12:        text_field;
1250                                fill_4:         fill_2_type;
1260                                line_13:        text_field;
1270                                fill_5:         fill_2_type;
1280                                line_14:        text_field;
1290                                fill_6:         fill_2_type;
1300                                line_15:        text_field;
1310                                fill_7:         fill_3_type;
1320                             end;
1330
1340           protocol        = record
1350                                start_attr:     byte;
1360                                ftpimage_id:    bytes_8;
1370                                space_9:        byte;
1380                                transfer_dir:   byte;
1390                                space_11:       byte;
1400                                host_file_name: bytes_8;
1410                                space_20:       byte;
1420                                term_file_name: bytes_14;
1430                                space_35:       byte;
1440                                file_size:      bytes_6;
1450                                space_42:       byte;
1460                                xferred_chars:  bytes_6;
1470                                space_49:       byte;
1480                                status_code:    bytes_2;
1490                                space_52:       byte;
1500                                function_code:  bytes_3;
1510                                space_56:       byte;
1520                                block_no:       byte;
1530                                space_58:       byte;
1540                                data_format:    byte;
1550                                space_60:       byte;
1560                                curr_blocksize: bytes_4;
1570                                space_65:       byte;
```

```
1580                                blocking_factor:bytes_2;
1590                                space_68:       byte;
1600                                host_file_recl: bytes_4;
1610                                space_73:       byte;
1620                                timing_factor:  bytes_2;
1630                                space_76:       bytes_3;
1640                                attr_char_2:    byte;
1650                            end;
1660
1670        data                = array (.1..1840.) of byte;
1680
1690        screen_image        = record
1700                                command_region: protocol;
1710                                data_region:    data;
1720                            end;
1730
1740        screen_type         = (ok,not_ftpimage,block_no_error);
1750
1760        dft_entry_type      = record
1770                                host_name:      bytes_8;
1780                                directions:     byte;
1790                                format:         byte;
1800                                rec_length:     bytes_4;
1810                                block_factor:   bytes_2;
1820                                delay:          bytes_2;
1830                                file_size:      bytes_6;
1840                                data_present:   boolean;
1850                                status:         bytes_2;
1860                            end;
1870
```

```
2010        var
2020
2030           state:           states;
2040           login_buf:       login_message;
2050           buffer:          screen_image;
2060           previous:        screen_image;
2070           screen_length:   integer;
2080           previous_lgt:    integer;
2090           next_block:      byte;
2100           received:        screen_type;
2110           dft_entry:       dft_entry_type;
2120           bytes_sent:      integer;
2130           file_opened:     boolean;
2140
3010        function wait_buffer(var buf: screen_image): screen_type;
3020        begin
3030           <* call an operating system function to wait for a screen  *>
3040           <* image from a terminal.                                   *>
3050           with buf.command_region do
3060           begin
3070              if ftpimage_id <> ftpimage then
3080                 wait_buffer:= not_ftpimage
3090              else
3100              if function_code<>enq and block_no <> next_block then
3110                 wait_buffer:= block_no_error
3120              else
3130                 wait_buffer:= ok
3140           end
3150        end;
3160
3170        procedure send_buffer(var buf: screen_image;var lgt: integer);
3180        begin
3190           previous:= buf;       <* save a copy for recovery *>
3200           previous_lgt:= lgt;
3210           <* call an operating system function to send the screen    *>
3220           <* image to a terminal. The number of bytes in the screen  *>
3230           <* image has been calculated in advance.                    *>
3240        end;
3250
3260        procedure look_in_dft(var entry: dft_entry_type;
3270                              var file_name: bytes_8);
3280        begin
3290           <* the procedure looks in the file table to see if the     *>
3300           <* file name is known; entry.status is set to status_06     *>
3310           <* if the file is unknown. if the file is known but          *>
3320           <* presently not accessible then entry.status is set to     *>
3330           <* status_01; otherwise entry.status is set to status_00     *>
3340           <* and all fields in entry are filled in with the file       *>
3350           <* attributes.                                               *>
3360        end;
3370
```

```
3380        procedure get_host_file_data;
3390        var
3400           dest,last_dest,bytes: integer; data_byte,wrk: byte;
3410        begin
3420           <* the procedure reads data from the currently open host   *>
3430           <* file and inserts the data in the buffer. If transparent *>
3440           <* mode is specified each byte is converted to two bytes   *>
3450           <* before the data is inserted in the buffer. Each byte    *>
3460           <* inserted in the buffer is converted in order to avoid   *>
3470           <* illegal values with respect to the IBM 3270 Protocol.   *>
3480           dest:=1;
3490           bytes:=0;
3500           last_dest:= blocking_factor*80;
3510           while not end_of_file and dest<=last_dest do
3520           begin
3530              data_byte:= get_char_from_host_file;
3540              if data_format=text_format then
3550              begin
3560                 data_byte:= convert_text_char(data_byte);
3570                 data_region(dest):= data_byte;
3580                 dest:= dest+1;
3590                 bytes:=bytes+1;
3600              end
3610              else
3620                 wrk:= convert_left_hex(data_byte);
3630                 data_region(dest):=wrk;
3640                 wrk:= convert_right_hex(data_byte);
3650                 data_region(dest+1):= wrk;
3660                 dest:=dest+2;
3670                 bytes:=bytes+1;
3680              end;
3690           end;
3700           if end_of_file then function_code:= eof;
3710           bytes_sent:= bytes_sent+bytes;
3720           xferred_chars:=binary_to_decimal(bytes_sent);
3730           curr_blocksize:=binary_to_decimal(dest);
3740           screen_length:= 80+dest;
3750        end;
3760
```

```
3770          procedure put_host_file_data;
3780          var
3790             src,last_src: integer; data_byte,wrk: byte;
3800          begin
3810             <* the procedure extracts data from the buffer and writes  *>
3820             <* the data in the currently open host file. If transpar-  *>
3830             <* ent mode is specified each byte in the host file is     *>
3840             <* formed by concatenating two bytes from the buffer.      *>
3850             src:= 1;
3860             last_src:= decimal_to_binary(curr_blocksize);
3870             while src<=last_src do
3880             begin
3890                if data_format=text_format then
3900                begin
3910                   data_byte:= data_region(src);
3920                   put_char_in_host_file(data_byte);
3930                   src:= src+1;
3940                end
3950                else
3960                begin
3970                   wrk:=convert_from_left_hex(data_region(src));
3980                   data_byte:=convert_from_right_hex(data_region(src+1));
3990                   data_byte:=wrk*16+data_byte;
4000                   put_char_in_host_file(data_byte);
4010                   src:= src+2;
4020                end
4030             end;
4040          end;
4050
```

```
4060          procedure answer_vaf;
4070          begin <* check the vaf and build a req *>
4080             state:= validation;
4090             function_code:= req;
4100             look_in_dft(dft_entry,host_file_name);
4110             status_code:= dft_entry.status;
4120             <* the information answered to the terminal is stored in   *>
4130             <* the terminal and will be used during the file transfer  *>
4140             <* if status_00 and the transfer is started.               *>
4150             data_format:= dft_entry.format;
4160             host_file_recl:= dft_entry.rec_length;
4170             blocking_factor:= dft_entry.block_factor;
4180             timing_factor:= dft_entry.delay;
4190             if status_code=status_00 then
4200             begin <* host file is known *>
4210                if dft_entry.directions<>host_rec_tr and
4220                   dft_entry.directions<>transfer_dir then
4230                   status_code:= status_02 <* illegal direction *>
4240                else
4250                if dft_entry.data_present and
4260                   transfer_dir=host_receive then
4270                   status_code:= status_03 <* not empty *>
4280                else
4290                if not dft_entry.data_present and
4300                   transfer_dir=host_transmit then
4310                   status_code:= status_04 <* empty *>
4320             end;
4330             screen_length:= 80;
4340             send_buffer(buffer,screen_length);
4350          end;
4360
4370          procedure answer_vae;
4380          begin <* build a req *>
4390             <* the terminal has terminated a sequence of vaf commands  *>
4400             <* and is expected to send a sof command after this answer.*>
4410             state:= prepare_transmit;
4420             function_code:= req;
4430             status_code:= status_00;
4440             screen_length:= 80;
4450             send_buffer(buffer,screen_length);
4460          end;
4470
```

```
4480          procedure answer_sof;
4490          begin <* check sof and build req or tfd *>
4500             <* only information which has been previously been re-     *>
4510             <* ceived in a vaf, and has been accepted, is expected to  *>
4520             <* be received in a sof.                                   *>
4530             state:= transmission;
4540             look_in_dft(dft_entry,host_file_name);
4550             if transfer_dir=host_receive then
4560                function_code:= req
4570             else
4580                function_code:= tfd;
4590             status_code:= dft_entry.status;
4600             <* as some time may have passed since the vaf was received *>
4610             <* the state of the host file may have changed and thus a  *>
4620             <* secondary check is necessary.                           *>
4630             if status_code=status_00 then
4640             begin <* host file is known *>
4650                if dft_entry.directions<>host_rec_tr and
4660                   dft_entry.directions<>transfer_dir then
4670                   status_code:= status_02 <* illegal direction *>
4680                else
4690                if dft_entry.data_present and
4700                   transfer_dir=host_receive then
4710                   status_code:= status_03 <* not empty *>
4720                else
4730                if not dft_entry.data_present and
4740                   transfer_dir=host_transmit then
4750                   status_code:= status_04 <* empty *>
4760             end;
4770             if status_code<>status_00 then
4780             begin <* the transfer can not be started *>
4790                function_code:= req;
4800                <* status_code is already initialized *>
4810             end
4820             else
4830             begin <* the transfer may be started *>
4840                bytes_sent:= 0;
4850                if transfer_dir=host_transmit then
4860                begin
4870                   <* open host file for reading, and position to start *>
4880                   <* of file.                                          *>
4890                   get_host_file_data;
4900                end
4910                else
4920                begin
4930                   <* open host file for writing, and position to start *>
4940                   <* of file.                                          *>
4950                end;
4960                file_opened:= true;
4970             end;
4980             if function_code=req then screen_length:= 80;
4990          end;
5000
```

```
5010        procedure answer_req;
5020        begin <* build tfd or eof with data *>
5030           state:= transmission;
5040           function_code:= tfd;
5050           get_host_file_data;
5060           if function_code=eof then
5070           begin
5080              <* close the host file *>
5090              file_opened:= false;
5100           end;
5110           send_buffer(buffer,screen_length);
5120        end;
5130
5140        procedure answer_tfd;
5150        begin
5160           <* data is received from the terminal. extract the data    *>
5170           <* from the buffer. if transmission mode is transparent    *>
5180           <* then compose each host file data byte by concatenating  *>
5190           <* two bytes from the buffer.                              *>
5200           put_host_file_data;
5210           state:= transmission;
5220           function_code:= req;
5230           screen_length:= 80;
5240           send_buffer(buffer,screen_length);
5250        end;
5260
5270        procedure answer_eof;
5280        begin
5290           <* this is the last part of the data from the terminal.    *>
5300           <* extract data from the buffer and write the data in the  *>
5310           <* host file. if transparent mode is used then compose     *>
5320           <* each byte in the host file by concatenating two bytes   *>
5330           <* from the buffer to one host file data byte.             *>
5340           put_host_file_data;
5350           <* close the host file. update the host file description   *>
5360           <* with respect to number of data bytes.                   *>
5370           file_opened:= false;
5380           state:= prepare_transmit;
5390           function_code:= req;
5400           screen_length:= 80;
5410           send_buffer(buffer,screen_length);
5420        end;
5430
5440        procedure answer_stp;
5450        begin
5460           if file_opened then
5470           begin
5480              <* close the host file. if transfer direction was host  *>
5490              <* receive then set file size to number of bytes recei- *>
5500              <* ved.                                                  *>
5510              file_opened:= false;
5520           end;
5530           state:= termination;
5540           function_code:= req;
5550           status_code:= status_00;
5560           screen_length:= 80;
5570           send_buffer(buffer,screen_length);
5580        end;
```

```
5590
5600          procedure answer_enq;
5610          begin
5620             buffer:= previous;
5630             screen_length:= previous_lgt;
5640             next_block:= block_no;
5650             send_buffer(buffer,screen_length);
5660          end;
5670
5680          procedure answer_ab1;
5690          begin
5700             <* the terminal wants to terminate the transfer and delete *>
5710             <* the received data. close the host file.                  *>
5720             if transfer_dir=host_receive then
5730             begin
5740                <* set the file size to 0 bytes *>
5750             end;
5760             close_the_host_file;
5770             file_opened:= false;
5780             state:= prepare_transmit;
5790             function_code:= req;
5800             status_code:= status_00;
5810             screen_length:= 80;
5820             send_buffer(buffer,screen_length);
5830          end;
5840
5850          procedure answer_ab2;
5860          begin
5870             <* the terminal wants to terminate the transfer. close the *>
5880             <* host file.                                               *>
5890             close_the_host_file;
5900             file_opened:= false;
5910             state:= prepare_transmit;
5920             function_code:= req;
5930             status_code:= status_00;
5940             screen_length:= 80;
5950             send_buffer(buffer,screen_length);
5960          end;
5970
```

```
7010        begin
7020
7030           <* the application has just been activated by the operat-  *>
7040           <* ing system. No transaction is expected at this point.   *>
7050           <* send the login message.                                 *>
7060
7070           login_buf.line_10:= line_10_text;
7080           login_buf.line_11:= line_11_text;
7090           login_buf.line_12:= line_12_text;
7100           login_buf.line_13:= line_13_text;
7110           login_buf.line_14:= line_14_text;
7120           login_buf.line_15:= line_15_text;
7130           screen_length:= 1920;
7140           send_buffer(login_buf,screen_length);
7150
7160           state:= validation;
7170           next_block:= 1;
7180
7190           repeat
7200              next_block:= 1-next_block;
7210              received:= wait_buffer(buffer);
7220              case received of
7230              ok:
7240                 begin
7250                    case state of
7260                    validation:
7270                       with buffer.command_region do
7280                       begin
7290                          case function_code of
7300                          vaf: answer_vaf;
7310                          vae: answer_vae;
7320                          stp: answer_stp;
7330                          enq: answer_enq;
7340                          otherwise
7350                             state:= validation;
7360                             function_code:= req;
7370                             status_code:= status_09;
7380                             screen_length:= 80;
7390                             send_buffer(buffer,screen_length);
7400                          end;
7410                       end;
7420                    prepare_transmit:
7430                       with buffer.command_region do
7440                       begin
7450                          case function_code of
7460                          sof: answer_sof;
7470                          vaf: answer_vaf;
7480                          stp: answer_stp;
7490                          enq: answer_enq;
7500                          otherwise
7510                             state:= prepare_transmit;
7520                             function_code:= req;
7530                             status_code:= status_09;
7540                             screen_length:= 80;
7550                             send_buffer(buffer,screen_length);
7560                          end;
7570                       end;
```

```
7580                    transmission:
7590                       with buffer.command_region do
7600                       begin
7610                          case function_code of
7620                          tfd: answer_tfd;
7630                          req: answer_req;
7640                          eof: answer_eof;
7650                          abl: answer_abl;
7660                          ab2: answer_ab2;
7670                          stp: answer_stp;
7680                          enq: answer_enq;
7690                          otherwise
7700                             state:= transmission;
7710                             function_code:= req;
7720                             status_code:= status_09;
7730                             screen_length:= 80;
7740                             send_buffer(buffer,screen_length);
7750                          end;
7760                       end;
7770                    end;
7780                 end;
7790              block_no_error:
7800                 begin
7810                    <* this situation is not expected to occur *>
7820                    state:= state; (* try to re-synchronize *)
7825                    next_block:= 1-next_block;
7830                    function_code:= req;
7840                    status_code:= status_07;
7850                    screen_length:= 80;
7860                    send_buffer(buffer,screen_length);
7870                 end;
7880              not_ftpimage:
7890                 begin
7900                    state:= termination;
7910                    function_code:= req;
7920                    status_code:= status_99;
7930                    screen_length:= 80;
7940                    send_buffer(buffer,screen_length);
7950                 end;
7960              end;
7970           until state=termination;
7980           <* call an operating system function to have the host      *>
7990           <* application program removed.                           *>
8000        end.
```

4.3 Comments to the Program Listing

In lines 40 to 410 is described the fundamental constants in the host framework application. Lines 40 to 90 describe the message sent from the host framework application when the host framework application is invoked. Line 110 describes the File Transfer Protocol identification "FTPIMAGE" which must be present in all screen images exchanged between the host File Transfer Application and the terminal, except for the first one sent to the terminal.

Lines 130 to 150 describe the allowed values for transfer directions. The normal transfer direction specifications will be "T" or "R", but in test environments "B" may be used to specify that transfer may take place in both directions.

Lines 170 to 260 decribe the valid function codes used in the File Transfer Protocol.

Lines 280 to 290 describe the two possible formats of data transmission, either "T" for text transfer mode or "S" for (split) transparent transfer mode. The term "S" means that each data byte is transferred in two bytes where only the least significant four bits in each byte are regarded as data.

Lines 310 to 410 describe the possible status codes which may be answered by the host File Transfer Application. Only the host File Transfer Application is utilizing the status field as the protocol indicates that the terminal is taking all initiatives to issuing file transfers.

In lines 1010 to 1870 is described the various scalar types and data structures.

Lines 1020 to 1030 describe the states of the host framework application program. Initially the program is in state validation, and in this state the program normally receives vaf commands. After the last vaf command the program receives a vae command and changes to state prepare_transmit; In this state the program receives either a sof command or another vaf command. The sof command indicates the start of a file transfer and upon reception of the sof command the program changes to state transmission. If a vaf command is received the program returns to state validation. When the program is in state transmission req or tfd commands are received, depending on the direction of the file transfer. Upon termination of the file transfer the program receives or transmits an eof command, depending on the direction of the file transfer, and changes to state prepare_transmit. When the program receives a stp command then the program terminates.

Lines 1170 to 1320 describe the login message. When the host file transfer application is invoked it must send exactly one screen image not confirming to the File Transfer Protocol instructing the terminal operator to switch the terminal into FTP Mode.

Lines 1340 to 1650 describe the File Transfer Protocol part of the screen images sent between the host program and the terminal program. The protocol part is exactly 80 bytes, containing all the necessary information to make the file transfer possible. The remaining part of the screen image may contain data. Only the part of the screen image which is filled in with control information or data is transmitted, i.e. a minimum of 80 bytes are transmitted and 80+n*80 bytes may be transmitted, where 0<=n<=23, indicating the number of data carrying text lines in the screen image.

Line 1670 describes the data carrying part of the screen image as an array of 1840 bytes, corresponding to 23 lines of 80 bytes.

Line 1690 to 1720 describe a screen image as a composite record consisting of a protocol part (80 bytes) and a data part (0 to 1840 bytes).

Line 1740 describes the qualification of a received screen image. The screen type may be ok, not_ftp_image or block_no_error. Type ok means that the received screen image contains the expected identification "ftpimage" and that the block number contained in the screen image is equal to the expected block number; This is the normal case. Type not_ftp_image indicates that the terminal does not operate in FTP mode as the identification "ftpimage" was not contained in the screen image; In this case the host program may terminate its operation. Type block_no_error is not expected to occur as it indicates a serious problem in the File Transfer Protocol communication; The type may be considered a reminiscence from the days of program development and test.

Lines 1760 to 1860 describe an instance of a host file description. Depending on the kind of host application environment the structure may be of varying appearance as long as the fundamental informations described in section 4.1 are present.

In the lines 2010 to 2130 the fundamental variables are described. The variable state of type states describes the state of the host program. The variable login_buf is used for the transmission of the first (not File Transfer Protocol format) informative message to the terminal. The variable buffer is used for transmission and reception of File Transfer Protocol screen images and contains a command region and a data part. Number of bytes to be sent from the variable buffer is contained in variable screen_length. In order to sup-

port error recovery each buffer is copied to the variable previous and the variable screen_length is copied to the variable previous_lgt, before the buffer is transmitted. The variable next_block contains the next block number id expected to be received. The variable received determines the kind of screen image received, and may take the values ok, not_ftp_image or block_no_error. The variable dft_entry describes the host file which has latest been involved in a vaf command or is actually being used during a file transfer. The variable bytes_sent expresses the current number of bytes sent during a file transfer from the host to a terminal. The variable file_opened determines whether a host file is opened or not.

In the lines 3010 to 5960 the various functions and procedures are described.

Lines 3010 to 3150 describe the function wait_buffer. The purpose of this function is to wait for a screen image from the terminal and to determine the kind of the screen image, which may be ok, not_ftp_image or block_id_error. Received screen images are expected to contain at least the command region (80 bytes) and possibly data.

Lines 3170 to 3240 describe the procedure send_buffer. This procedure is used when a screen image with or without data (but always with the command region) is to be sent to the terminal. The actual size or amount of bytes to be sent is determined by the variable screen_length.

Lines 3260 to 3360 describe the procedure look_in_dft, which finds the characteristics of a specified host file.

Lines 3380 to 3750 describe the procedure get_host_file_data. This procedure reads data from the currently open host file and inserts the data in the screen image buffer. The procedure handles the possible splitting of data bytes when Transparent Transfer Mode is used as well as the code conversion necessary in order to avoid conflicts with IBM 3270 Protocol function codes.

Lines 3770 to 4040 describe the procedure put_host_file_data. This procedure extracts data from the screen image buffer and writes the data in the currently open host file. The procedure handles the possible concatenation of data byte parts when Transparent Transfer Mode is used.

In the lines 4060 to 5950 a set of procedures are described, which each handle the reception of a specific command from the terminal.

Lines 4060 to 4350 describe the procedure answer_vaf. When a vaf command is received the host program must control the presence and state of the host file specified in the vaf command. The procedure builds and sends a req command with the result of the validation.

Lines 4370 to 4460 describe the procedure answer_vae. When this command is received the host program knows that the state should be changed from validation to prepare_transmit. The procedure builds and sends a req command.

Lines 4480 to 4990 describe the procedure answer_sof. This command is received from the terminal when the terminal wants to start a file transfer. As the state of the host file may have changed since the earlier vaf command was received another validation of the host file state is performed. If the host file condition is proper the procedure builds and sends either a req command (when the terminal wants to transmit) or a tfd command (when the terminal wants to receive). If the host file state is improper a req command with the cause for rejecting the file transfer is built and sent.

Lines 5010 to 5120 describe the procedure answer_req. This command occurs only when a file transfer from host to terminal has been started and means that the host should send more data. The host program builds and sends a tfd command with associated data. If the host program has determined this part as the last of the host file the command is changed to eof (with data).

Lines 5140 to 5250 describe the procedure answer_tfd. This command indicates that the terminal has sent data. The data is extracted from the screen image buffer and is written in the host file. The procedure builds and sends a req command to the terminal insructing the terminal to send more data.

Lines 5270 to 5420 describe the procedure answer_eof. When the terminal is sending the last part of a file the command is eof instead of tfd. The data is extracted from the screen image buffer and is written in the host file which is then closed. The procedure builds and sends a req command.

Lines 5440 to 5580 describe the procedure answer_stp. On the reception of

this command the host program should terminate its function. If any host file is open it should be closed. Before terminating the procedure builds and sends a req command to the terminal.

Lines 5600 to 5660 describe the procedure answer_enq. If for some reason the host program did not receive (and answer) a command from the terminal within a time period specified by the terminal then the terminal will send an enq command. The procedure will then build and send a screen image similar to the one latest sent. This is the reason why the procedure send_buffer always saves a copy of the screen image to be sent.

Lines 5680 to 5830 describe the procedure answer_abl. When this command is received then the host program must detect that the terminal wants to terminate the file transfer and to ignore transmitted data. This means that if the host program is receiving then the host file size should be set to zero as if the file transfer was never started. If the host program is transmitting then the state of the host file should not be changed. The procedure builds and sends a req command.

Lines 5850 to 5960 describe the procedure answer_ab2. Reception of this command means that the terminal wants to terminate the file transfer currently in progress. If the host program is receiving then the host file should be closed with the data contained so far. If the host program is transmitting then the host file state should not be changed. The procedure builds and sends a req command.

In the lines 7010 to 8000 the program main loop is described. When the program is activated the "change to FTP mode" message is sent to the terminal. Then the host program performs a loop where a screen image from the terminal is waited for and a proper action corresponding to the state of the host program is issued. The loop is repeated until the state of the host program is changed to "termination".

Note that if an enq is received as the first screen image from the terminal the variable previous is not properly initialized. Thus the variable should be initialized as if a vaf had been received earlier.

5. ITT 3290 FILE TRANSFER PROTOCOL PROGRAM

This chapter describes the File Transfer Protocol program for the ITT 3290 intelligent workstation. A brief description of the program and its structure is given, followed by a description of the data files which may be transmitted. Then follows a description of the message text file used to make the program function with various Control Unit conversion tables and various local languages.

The chapter does not describe how to operate the program. Operating instructions may be found in ITT 3290 File Transfer Program, Operating Guide (Ref (2)).

5.1 What is the File Transfer Program

The File Transfer Program (FTP) is a 'menu-driven' application program which you may locally load under CP/M on your ITT 3290 intelligent workstation whenever you wish to deliver files to your mainframe or receive files from your mainframe. The File Transfer Program will use your local language whenever a menu (or other message) is presented to you.

The File Transfer Program is a pure application program. This means that no changes in your existing telecommunication facilities are necessary, no matter if you are utilizing BSC or SNA telecommunication facilities.

The File Transfer Program is capable of exchanging any type of file you may wish. Some of the files you are using may consist of pure ASCII characters and other files may consist of pure binary data (e.g. a program file), or perhaps your files consist of a mixture of these types.

You may use FTP for transmission in either Text Mode or Transparent Mode depending on the type of file data you want to exchange.

When FTP works in Text Mode then a conversion between your local ASCII character representation and the host end EBCDIC character representation automatically takes place. You may also wish to move the files without conversion (the Transparent Mode) and this also gives you the freedom to exchange

files which contain non-ASCII characters. In addition to offering you a file to file transmission facility between the host and the workstation FTP is also capable of receiving files from the host and directly print the received data, provided your workstation has a printer attached locally.

As an extra facility the most commonly used directory and file handling functions are made available within FTP.

5.2 The Structure of FTP

The File Transfer Program communicates with the workstation user through 7 panels. Each of the panels have a fixed structure and the leading texts in each panel have a pre-determined meaning although the actual appearance of a text may occur in local language. In this manual all texts occur in english.

One of the 7 panels is named the Read Transfer Specification File panel and is used to read a previously prepared diskette file containing a set of file transfer specifications. This will normally be the first task to do with FTP when FTP has been loaded on the workstation.

After having read a Transfer Specification File FTP automatically proceeds to the Transfer Specifications panel which will display the transfer specification data just read from the file. The purpose of this panel is to let you enter, delete or change file names and the other necessary informations. The list of file name pairs and transfer directions is handled as a entirety when you activate the execution of the file transfer and the accomplishment of the specified set of file transfers is denoted a FTP session.

You may choose to perform a validation of the information on the list before actually activating the FTP session. This is done by selecting the Transfer Specification Validation panel.

After having validated the Transfer Specification list you may proceed to start the FTP session by selecting the (FTP) Execution panel. During the FTP session FTP will execute all the file transfers specified on the list, one by one, and while a specific file transfer takes place the file names involved and the transfer direction together with file size and bytes transferred are

displayed. When the FTP session is carried through FTP automatically returns to the Main Menu panel, ready to start another transfer specification and FTP session.

When the terminal or workstation is running the ordinary ITT 3297 Terminal Emulator Program the terminal functions according to the block diagram showed below.

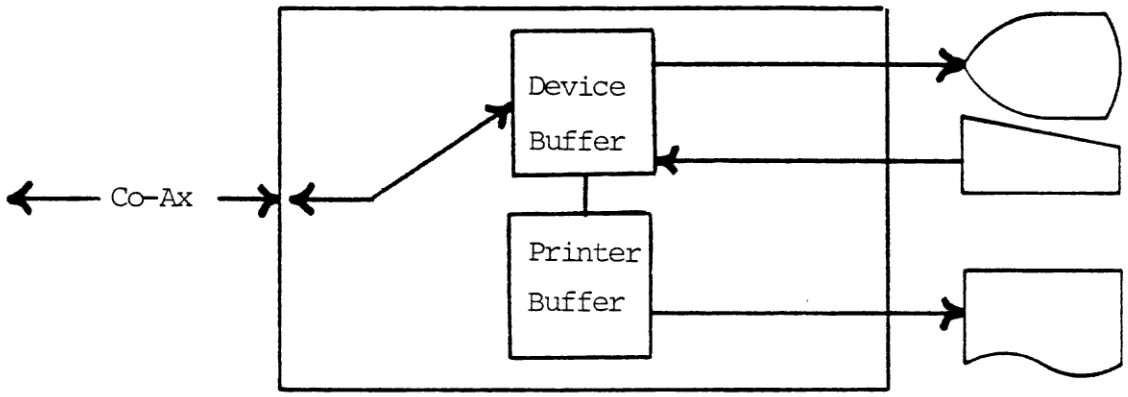


Figure 5.2.a Internal structure of the ITT 3297 Terminal Emulator

On the Co-Ax communication line the terminal receives screen images in the device buffer and the contents of the device buffer is automatically shown on the display. Keyboard input causes an update of the device buffer (which may be seen on the display) and by means of attention keys the operator instructs the terminal emulator to send the device buffer on the Co-Ax communication line (figure 5.2.a).

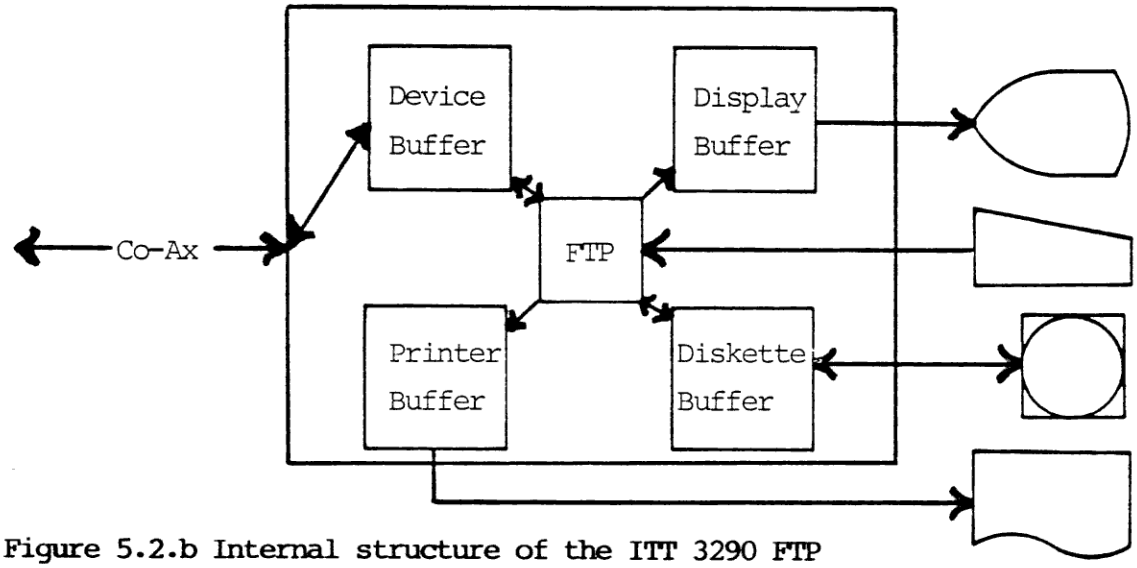


Figure 5.2.b Internal structure of the ITT 3290 FTP

When the workstation is running FTP you might say that the workstation still looks like an ITT 3297 Terminal Emulator to the CU. The CU still writes and reads in the device buffer according to the Courrier Level III protocol but the device buffer is no longer automatically shown on the display and the keyboard input is no longer automatically inserted in the device buffer (figure 5.2.b on previous page).

The FTP generates its own panels, which are shown i appendix B.2, and FTP changes the display according to keyboard input without changing the device buffer. When FTP wants to communicate with the mainframe data is generated by FTP and inserted in the device buffer and a simulation of a keyboard attention key having been pressed is performed. This causes a transfer of the device buffer to the mainframe without disturbing the display shown to you. Similarly, screen images from the mainframe are received in the device buffer and are then interpreted by FTP.

The File Transfer Protocol defines the structure of screen images sent back and forth and the mainframe application and FTP are "talking" to each other by alternately sending screen images to each other. During a file transfer from the workstation to the mainframe the folowing sequence of events are repeated until the file has been completely transferred:

The mainframe sends a screen image to the workstation. This screen image contains among other necessary information the command REQ (request for data) instructing FTP to deliver more data. FTP will then read data from the current diskette file and insert the data in the device buffer (still without changing your display image), and the command in the device buffer is changed to TFD (transfer file data) along with other necessary information. Finally, FTP simulates that the SEND key were pressed on the keyboard causing the device buffer to be sent to the mainframe. The mainframe extracts the data part from the device buffer and writes the data in the specified mainframe file. Then the mainframe changes the command to REQ again and sends the screen image to the terminal (FTP) and so on.

5.3 The Data Files

FTP accepts to transmit any diskette file you may specify and to receive

and write on the diskette any sequence of characters sent from the host. However, you must be aware of the two different principles of file transfer which FTP may use. The two principles are called Transparent Transfer Mode and Text Transfer Mode.

The principle of file transfer actually used depends on the mainframe which has a description of all its files installed for file transfer usage. When you send a file to the host then FTP receives REQ commands which also specify whether FTP should deliver the data in Transparent Transfer Mode or in Text Transfer Mode.

The difference between the two principles are that a file transfer in Text Transfer Mode always performs a conversion of each single character in the file transfer as the mainframe side normally handles text written in the EBCDIC alphabet whereas the workstation is working with the ASCII alphabet. In contrast to this the Transparent Transfer Mode simply copies each single character without changing anything. This means that if you are sending an ASCII character file in transparent transfer mode then the mainframe file will contain ASCII characters when the file transfer is finished. Similarly, an EBCDIC file received from the mainframe in Transparent Transfer Mode will cause your diskette file to be filled with EBCDIC characters.

Although the transfer mode used is determined by the host file used you may still have the possibility to make your own choice on the transfer mode to be used. One possibility is that files on the mainframe are divided in two groups with each their transfer mode. Another possibility might be to alter the mainframe characteristics to suit your purpose, from your own workstation. This second possibility, however, depends on the availability of mainframe applications capable of performing the desired changes.

The Transparent Transfer Mode should always be used in cases where you want a precise copy of a file to be transferred without any code conversion. If you want to transfer a CP/M program (a .COM file) to the mainframe e.g. for distribution to other workstations then you should use the Transparent Transfer Mode as the file is what could be denoted a binary file.

Transparent Transfer Mode should also be used if your text file contains non-ASCII characters (which is the case with e.g. WordStar files) in order to avoid "loss" of the non-ASCII characters.

When you have a text file containing pure ASCII characters then you can use the Text Transfer Mode. This transfer mode also implies that your ASCII characters are stored as EBCDIC characters in the host file. If you transfer the file back again you will receive ASCII characters on your diskette.

If your file contains non-ASCII characters then these are converted to spaces before the transmission. The CU will then convert the ASCII characters to EBCDIC according to the standard or alternate conversion table used.

On the mainframe side those EBCDIC characters conflicting with the 3270 protocol are converted to spaces before they are sent to the CU, and the CU will then convert the EBCDIC characters to ASCII characters according to the standard or alternate conversion table used before the characters are sent to the workstation.

One more code conversion should be mentioned. As your diskette data is containing ASCII characters conforming to the ISO standard which is slightly different from the ITT Standard and Alternate ASCII character sets a conversion between these two standards is performed before diskette data is sent to the mainframe and before data received from the mainframe is written on your diskette.

The Text Transfer Mode is best suited for transmission of program text files which can always be assumed to consist of ASCII characters only. Also data generated on the mainframe is well suited for Text Transfer Mode as the mainframe normally will use only Carriage-Return, Line-feed and Form-feed characters as text formatting characters, as opposed to various text-editing packages on your workstation. The mainframe kind of data is typically reports and other statistical material meant to be printed on fast drum or chain printers without letter-quality facilities.

5.4 Message File

When FTP is loaded on your workstation a Message File on the diskette is opened and among other information FTP reads in all text strings necessary to build the various panels and other messages which is needed in the communication on the screen.

The use of a Message File makes it possible to present all FTP panels and messages in various local languages. The Message File includes all the panel titles, menu texts, error/status/progress messages etc. used during the work with FTP. Furthermore, the Message File includes a description of character code differences between the CP/M ASCII and the ITT ASCII (standard/alternate) character sets.

The Message File is an ordinary CP/M file containing pure ASCII characters. The Message File must be situated on the same diskette from which FTP is loaded (always on drive A) and must have the name FTPMSGS.TXT.

As long as the conventions for the Message File structure are followed the contents of the text strings may be altered, but the meaning of the sentences should, however, not be altered. An ordinary text editor, e.g. WordStar, may be used for this purpose.

5.4.1 Message File Structure

The Message File is composed of the following three parts:

- CP/M ASCII to ITT Standard ASCII conversion table corrections
- CP/M ASCII to ITT Alternate ASCII conversion table corrections
- FTP message texts

The first part starts with the keyword STD followed by a number of lines each specifying one correction to the standard conversion table and terminated with a line containing a single slash.

The second part starts with the keyword ALT followed by a number of lines each specifying one correction to the alternate conversion table and terminated with a line containing a single slash.

The third part is a number of lines each containing one text string. Below is given an example of how the Message File might look. Note, that the comments in the right column are not included in the Message File. Also note that a structure like e.g. 34,- means that the ITT table has no character corresponding to the CP/M character with value 34. In appendix B.1 ASCII TABLE SAMP-

LES the same structure is expressed as 34,SP indicating that the CP/M character with decimal value 34 (the character ") is converted to space as the character is not included in the ITT table.

Danish version of FTPMSGS.TXT (main structure):

```
STD (cr,lf)                  Start of standard table corrections
33,93 (cr,lf)                Exclamation mark
35,91 (cr,lf)                Number sign
 -                            -
 -                            -
 -                            -
126,92 (cr,lf)               Back slash
/                            End of standard table corrections

ALT (cr,lf)                  Start of alternate table corrections
33,93 (cr,lf)                Exclamation mark
34,- (cr,lf)                 Quotation mark (doesn't exist with ITT)
 -                            -
 -                            -
 -                            -
125,33 (cr,lf)               The character å
126,- (cr,lf)                Back slash (doesn't exist with ITT)
/                            End of alternate table corrections

5,5,MT000 (cr,lf)            First message text item
15,11,ITT3297 FTP (cr,lf)     -
 -                            -
 -                            -
 -                            -
30,20,PRESS <ESC> TO RETURN (cr,lf) Last message text item
```

Each message text item defined in the Message File has a specific meaning and may be used on various places in several panels. As the panels have a fixed structure with respect to the screen position of the first character in the text a maximum length is defined with each text. The maximum length is chosen such that the meaning of the text can be expressed in all languages.

As mentioned above appendix B.1 contains a survey of code conversion table corrections for various languages. For more detailed table descriptions, please consult the ITT 3297 Display Station, Reference Manual (ref. (1)).

Each message text item in the Message File has the following structure:

<ml>,<al>,<text><cr><lf>

where

<ml> : maximum message text item length
<al> : actual message text item length, <= <ml>
<text> : message text item, no of characters = <al>
<cr> : carriage return (not included in <ml>, <al>)
<lf> : line feed (not included in <ml>, <al>)

5.4.2 Message File Contents

The message text items contained in the Message File are (the message number is not included in the Message File):

Message no.	Max length	Act.length	Text
00	5	5	MT000
01	15	11	ITT3290 FTP
02	40	9	MAIN MENU
03	40	32	READ TRANSFER SPECIFICATION FILE
04	40	22	TRANSFER SPECIFICATION
05	40	33	TRANSFER SPECIFICATION VALIDATION
06	40	22	VOLUME/FILE FACILITIES
07	40	13	FTP EXECUTION
08	40	15	FTP TERMINATION
09	40	10	TRACE MODE
10	40	9	STEP MODE
11	40	32	TRANSFER SPECIFICATION FILE NAME
12	30	28	HOST FILE CP/M FILE T/R
13	30	17	VALIDATION RESULT
14	40	16	DELETE CP/M FILE
15	40	16	RENAME CP/M FILE
16	40	14	CP/M FILE SIZE
17	40	21	CURRENT LOGGED VOLUME
18	40	13	VOLUME STATUS
19	40	14	DIRECTORY LIST
20	40	19	CLEAR RECEIVED DATA
21	40	19	CLOSE RECEIVED FILE
22	30	14	ILLEGAL VOLUME

Message no.	Max length	Act.length	Text
23	40	16	RESTART PRINTING
24	40	08	DELETED
25	15	12	RENAMED TO
26	30	13	NOT DIRECTION
27	30	22	NEXT STEP PRESS <ESC>
28	30	12	PRINTER BUSY
29	30	15	PRINTER OFFLINE
30	30	17	PRINTER NOT READY
31	30	21	INVALID DISC RESPONSE
32	30	16	END OF DISC FILE
33	30	22	ILLEGAL DISC OPERATION
34	30	25	DISC OFFLINE (I/O PARITY)
35	30	12	DISC OFFLINE
36	30	20	DISC WRITE-PROTECTED
37	30	22	ILLEGAL DISC FILE-NAME
38	30	16	DISC FILE EXISTS
39	30	17	DISC FILE UNKNOWN
40	30	09	DISC FULL
41	30	17	DISC STREAM ERROR
42	30	23	REJECTED DISC OPERATION
43	30	25	DISC FILE WRITE-PROTECTED
44	30	18	DISC INPUT OVERRUN
45	30	02	OK
46	30	26	HOST FILE UNDEFINED/CLOSED
47	30	26	INVALID TRANSFER DIRECTION
48	30	19	HOST FILE NOT EMPTY
49	30	15	HOST FILE EMPTY
50	30	14	HOST FILE FULL
51	30	22	HOST FILE NAME UNKNOWN
52	30	22	BLOCKID RECEIVED TWICE
53	30	13	HOST ERROR 08
54	30	21	INVALID TRANSFER CODE
55	30	19	INVALID STATUS CODE
56	20	18	USED/FREE ENTRIES:
57	15	15	USED/FREE DISK:
58	30	20	PRESS <ESC> TO RETURN

5.4.3 Message References

A precise description of the usage of the Message File texts, including for each text the panel numbers and the screen coordinates, is provided below. The names of the panels, which are shown in appendix B.2, are:

1. Main Menu Panel
2. Read Transfer Specification File Panel
3. Transfer Specification Panel
4. Transfer Specification Validation Panel
5. Volume/File Facilities Panel
6. Directory List Panel
7. FTP Execution Panel
8. FTP Termination Panel

Message no.	Panel no.	Line no.	Position (1-80)	Comments
00	1,2,3,4,5,6,7,8	1	76	Text file id.
01	1,2,3,4,5,6,7,8	1	1	
02	1	1	20	
02	2,3,4,5	3	20	
03	1	3	20	
03	2	1	20	
03	3,4,5	4	20	
04	1,2	4	20	
04	3	1	20	
04	4,5	5	20	
05	1,2,3	5	20	
05	4	1	20	
05	5	6	20	
06	1,2,3,4	6	20	
06	5	1	20	
07	1,2,3,4,5	7	20	
07	7	1	20	
08	1	8	20	
08	8	1	20	
09	1	9	20	
09	7	3	20	
10	1	10	20	
10	7	4	20	
11	2	9	20	
12	3,4	9	20	
13	4	9	52	
14	5	9	20	
15	5	11	20	
16	5	13	20	
17	5	15	20	
18	5	17	20	
19	5	19	20	
19	6	1	20	DIRECTORY LIST
20	7	5	20	
21	7	6	20	
22	2,5	24	3	ILLEGAL VOLUME
22	4	10-19	50	
23	7	7	20	RESTART PRINTING
24	5	24	19	DELETED
25	5	24	19	RENAMED TO
26	4	10-19	50	NOT DIRECTION
27	7	24	3	NEXT STEP PRESS <ESC>
28	1,2,3,4,5,6,7,8	24	3	PRINTER BUSY
29	1,2,3,4,5,6,7,8	24	3	PRINTER OFFLINE
30	1,2,3,4,5,6,7,8	24	3	PRINTER NOT READY
31	2,5,7	24	3	INVALID DISK RESPONSE
31	4	10-19	50	
32	4	10-19	50	END OF DISC FILE
33	2,5,7	24	3	ILLEGAL DISK OPERATION
34	2,5,7	24	3	DISK OFFLINE (I/O PARITY)
35	2,5,7	24	3	DISK OFFLINE
36	2,5,7	24	3	DISK WRITE-PROTECTED
37	2,5,7	24	3	ILLEGAL DISK FILE-NAME
38	2,5,7	24	3	DISK FILE EXISTS
39	2,5,7	24	3	DISK FILE UNKNOWN

Message no.	Panel no.	Line no.	Position (1-80)	Comments
40	2,5,7	24	3	DISK FULL
41	2,5,7	24	3	DISK STREAM ERROR
42	2,5,7	24	3	REJECTED DISK OPERATION
43	2,5,7	24	3	DISK FILE WRITE-PROTECTED
44	2,5,7	24	3	DISK INPUT OVERRUN
45	4	10-19	50	OK
46	4	10-19	50	HOST FILE UNDEFINED/CLOSED
47	4	10-19	50	INVALID TRANSFER DIRECTION
48	4	10-19	50	HOST FILE NOT EMPTY
49	4	10-19	50	HOST FILE EMPTY
50	7	24	3	HOST FILE FULL
51	4	10-19	50	HOST FILE-NAME UNKNOWN
52	4	10-19	50	BLOCKID RECEIVED TWICE
52	7	24	3	
53	4	10-19	50	HOST ERROR 08
53	7	24	3	
54	4	10-19	50	INVALID TRANSFER-CODE
54	7	24	3	
55	4	10-19	50	INVALID STATUS-CODE
55	7	24	3	
56	5	18	22	USED/FREE ENTRIES:
57	5	18	49	USED/FREE DISC:
58	6	24	3	PRESS <ESC> TO RETURN

A. REFERENCES

(1) RCSL No. 42-i2352:
ITT 3297 Display Station, Reference Manual
June 1983
Abstract: This manual describes the operational characteristics of the ITT 3297 Display Terminal. Includes: printer attachments, character codes and translation tables.

(2) RCSL No. 42-i2381:
ITT 3290 File Transfer Program, Operating Guide
Claus Terp, December 1983
Abstract: This manual is an operating guide for the ITT 3290 FTP program. The manual describes the various capabilities of the File Transfer Program, how the program is operated, how the various functions are activated and how the program reacts to the operator commands. The manual also describes how error conditions are reported to the operator and how the operator may overcome the various problems. In addition, the manual contains all necessary information on how to create transfer specification files and how different types of data files should be transferred. The manual also describes the principles of code conversions used.

B.1 ASCII TABLE SAMPLES

This section describes the necessary code conversions to be performed between the ITT 3290 CP/M ASCII character set and the ITT Courier Control Unit standard and alternate character set. In correspondence with the local language in question one of the below listed conversion specifications have been included in the FTPMSGS.TXT file delivered with FTP.

Please note the notational method used in the tables below. As an example we take a look at the belgian conversions which show e.g. that the exclamation mark has decimal value 33 in CP/M and decimal value 93 in the ITT standard (and alternate, by occasion) table(s), and that the right square bracket has value 93 in CP/M and value 33 in the ITT tables. Thus xx:yy(char) means that (char), which has the value xx, is converted to yy, and you will often find the expression yy:xx(char) in the opposite conversion table.

For other languages you will see differences between the standard and alternate tables.

The symmetry exposed by the belgian conversions is not present in e.g. the german conversion tables. The notation 34:SP(") found in the third column means that the quotation mark is present in the CP/M alphabet but not in the ITT alternate alphabet. Thus the opposite table shows no conversion from something to 34.

For nationalities not mentioned below the necessary code conversions may be specified by editing the file FTPMSGS.TXT with an ordinary word processor, e.g. WordStar.

NATIONALITY	CP/M TO ITT (STD)	ITT (STD) TO CP/M	CP/M TO ITT (ALT)	ITT (ALT) TO CP/M
BELGIAN	33:93 (!) 93:33 (])	33:93 (]) 93:33 (!)	33:93 (!) 93:33 (])	33:93 (]) 93:33 (!)

NATIONALITY	CP/M TO ITT (STD)	ITT (STD) TO CP/M	CP/M TO ITT (ALT)	ITT (ALT) TO CP/M
ENGLISH	36:91 ($) 91:124([) 93:96 (]) 96:36 (£) 124:93(|)	36:96 (£) 91:36 ($) 93:124(|) 96:93 (]) 124:91([)	36:91 ($) 91:124([) 93:96 (]) 96:36 (£) 124:93(|)	36:96 (£) 91:36 ($) 93:124(|) 96:93 (]) 124:91([)
DANISH	33:93 (!) 35:91 (#) 36:33 ($) 64:126(ü) 91:35 (Æ) 92:64 (Ø) 93:36 (Å) 126:92(\)	33:36 ($) 35:91 (Æ) 36:93 (Å) 64:92 (Ø) 91:35 (#) 92:126(\) 93:33 (!) 126:64(ü)	33:93 (!) 34:SP (") 35:SP (#) 36:SP ($) 64:SP (ü) 91:35 (Æ) 92:64 (Ø) 93:36 (Å) 96:SP (\) 123:34(æ) 124:91(ø) 125:33(å) 126:SP(\)	33:125(å) 34:123(æ) 35:91 (Æ) 36:93 (Å) 64:92 (Ø) 91:124(ø) 93:33 (!)
GERMAN	33:93 (!) 93:33 (ü)	33:93 (ü) 93:33 (!)	33:93 (!) 34:SP (") 35:SP (#) 36:SP ($) 64:SP (§) 91:35 (Ä) 92:64 (Ö) 93:36 (ü) 123:34(ä) 124:91(ö) 125:33(ü) 126:124(ß)	33:125(ü) 34:123(ä) 35:91 (Ä) 36:93 (ü) 64:92 (Ö) 91:124(ö) 93:33 (!) 124:126(ß)

NATIONALITY	CP/M TO ITT (STD)	ITT (STD) TO CP/M	CP/M TO ITT (ALT)	ITT (ALT) TO CP/M
SWEDISH	33:93 (!)	33:36 (¤)	33:93 (!)	33:125(å)
	35:91 (§)	35:91 (Ä)	34:124(")	34:123(ä)
	36:33 (¤)	36:93 (Å)	35:125(§)	35:91 (Ä)
	64:92 (É)	64:92 (ö)	36:123(¤)	36:93 (Å)
	91:35 (Ä)	91:35 (§)	64:92 (É)	64:92 (Ö)
	92:64 (ö)	92:64 (É)	91:35 (Ä)	91:124(ö)
	93:36 (Å)	93:33 (!)	92:64 (Ö)	92:64 (É)
	94:126(ü)	94:126(^)	93:36 (Å)	93:33 (!)
	126:94(^)	126:94(ü)	94:126(ü)	94:126(^)
			123:34(ä)	123:36(¤)
			124:91(ö)	124:34(")
			125:33(å)	125:35(§)
			126:94(^)	126:94(ü)

B.2 ITT 3290 FTP PANELS AND COMMANDS

This section describes the panels and commands (Program Function keys) used in the operation of the ITT 3290 FTP. The section is not meant to be an exhaustive operating guide as the operation of FTP is fully described in (2), ITT 3290 File Transfer Program, Operating Guide.

```
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.. ITT 3290 FTP 2.00: 3287 Mode .....
```

Figure B.2.a Panel shown when FTP is loaded

When FTP is loaded the screen is cleared and in the status line (line 25) is displayed the text "ITT 3290 FTP 2.00: 3287 Mode" in order to distinguish FTP from the standard 3297 Coax Emulator, as FTP contains only a limited set of the functions provided with the standard 3297 Coax Emulator.

In the 3287 mode FTP can do editing of the screen image, send the screen image according to the Program Function key pressed and display received screen images.

While in the 3287 mode the operator may do logon operations to the host operating system and activate the host File Transfer Application.

Figure B.2.b is an example of the first panel which the Host File Transfer Application must send when invoked. The panel is the only one which is not sent in accordance with the File Transfer Protocol, and serves the purpose of instructing the terminal user to change the terminal mode from pseudo ITT 3297 Coax Emulator mode (denoted 3287 mode in the status line) to FTP mode.

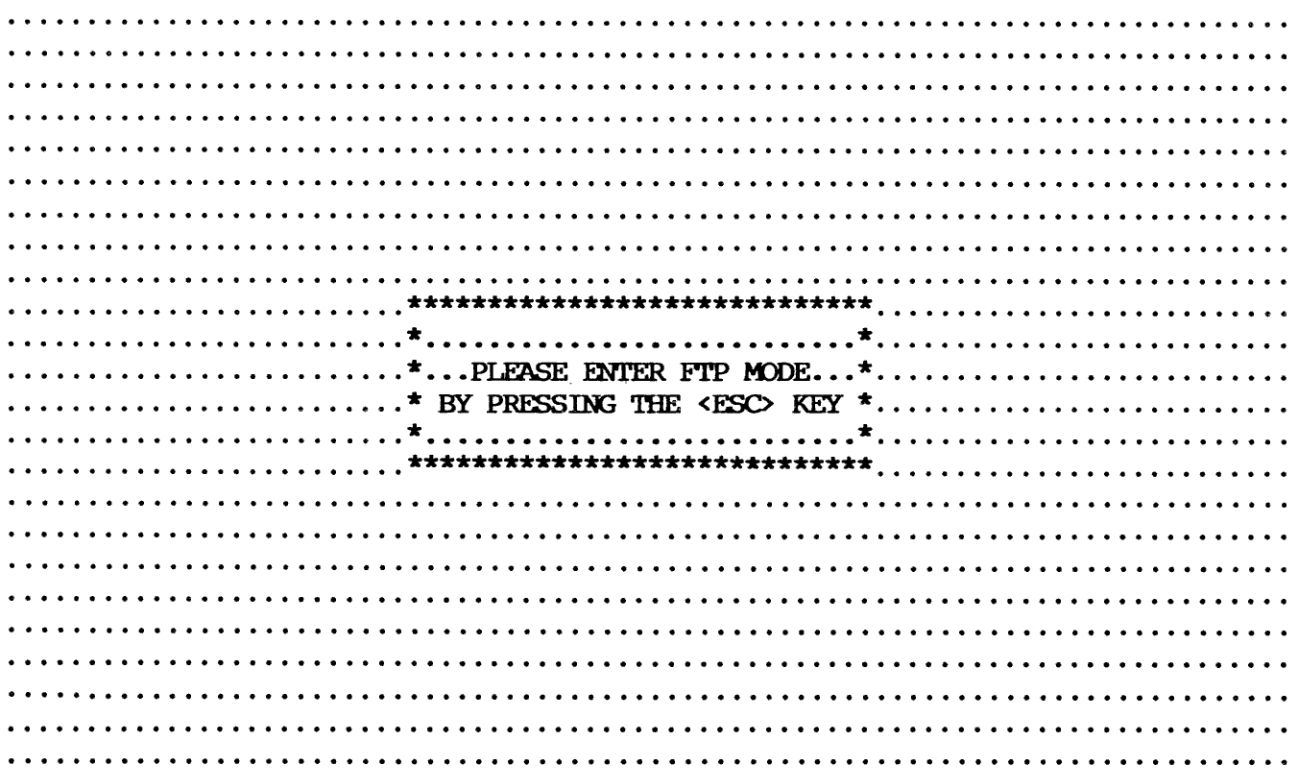


Figure B.2.b Panel sent from Host File Transfer Application

The operator may now enter the FTP mode by pressing the ESC key. This instruction may be contained in the screen image. See also (2), chapter 4, on FTP operation.

When FTP is switched into FTP mode the "3287 Mode" text in the status line is removed and FTP displays the Main Menu panel.

Figure B.2.c is the first panel displayed by FTP after the ITT 3290 terminal has been switched into FTP mode, and is denoted the Main Menu panel.

```
ITT3297.FTP.....MAIN.MENU.....REL.2.00..XXXXX
.....
.....PF1 READ TRANSFER SPECIFICATION FILE.....
.....PF2 TRANSFER SPECIFICATION.....
.....PF3 TRANSFER SPECIFICATION VALIDATION.....
.....PF4 VOLUME/FILE FACILITIES.....
.....PF5 FTP EXECUTION.....
.....PF6 FTP TERMINATION.....
.....PF7 TRACE MODE.....
.....PF8 STEP MODE.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
```

Figure B.2.c Main Menu Panel

The general structure of all FTP panels is as follows: Line 1 contains four fields, where field 1 containing the text "ITT3297 FTP" serves as program identification, field 2 is the panel name e.g. "Main Menu", field 3 is the FTP release number e.g. "REL 2.00" and field 4 (shown as "XXXXX" in the panels) is an identification of the message text file used with FTP.

Lines 2 to 24 is the communication area. In the upper part of the panel is shown other panels or functions which may be selected, one possible choice per line, where the line shows the name of a key and the panel/function which is selected if the key is pressed. Below the menu may be shown various leading texts and input fields as explained in the following. Status messages may occur to the right on the same line as an input field or on line 24.

From the Main Menu panel, and from any other panel, other panels or functions are selected by pressing the Program Function key associated with the panel or function name.

If any other than the Program Function keys (or ESC) displayed in the panel is pressed the beeper will sound.

The Main Menu panel expresses all the possible functions or panels which the operator may choose. When a new panel is selected a new list of choices is presented with the new panel, and if this new panel does not allow selection of a desired panel the operator may always go back to the Main Menu panel by pressing the ESC key and then select the panel.

The Read Transfer Specification File panel (figure B.2.d) is always selected with the PF1 key. The panel is used when a file transfer session (a set of file transfers) specification is to be read in from a previously prepared file rather than being keyed in directly to FTP.

```
ITT3297.FTP.....READ.TRANSFER.SPECIFICATION.FILE.....REL.2.00..XXXXX
.....
.....ESC MAIN MENU.....
.....PF2 TRANSFER SPECIFICATION.....
.....PF3 TRANSFER SPECIFICATION VALIDATION.....
.....PF4 VOLUME/FILE FACILITIES.....
.....PF5 FTP EXECUTION.....
.....
.....TRANSFER SPECIFICATION FILE NAME.....
.....
```

Figure B.2.d Read Transfer Specification File panel

The operator should key in the drive name (A or B) followed by the file name (1 to 8 characters) and the file extension name (1 to 3 characters) according to ordinary CP/M file name conventions. When the SEND key is pressed FTP will read from the specified file and display the Transfer Specification panel (figure B.2.e) with the file data inserted.

The Transfer Specification panel (figure B.2.e) is always selected with the PF2 key. The panel also appears when a file transfer specification file has been read. The panel is used for editing of file transfer specifications.

```
ITT3297.FTP.....TRANSFER.SPECIFICATION.....REL.2.00..XXXXX
.....
.....ESC MAIN MENU.....
.....PF1 READ TRANSFER SPECIFICATION FILE.....
.....PF3 TRANSFER SPECIFICATION VALIDATION.....
.....PF4 VOLUME/FILE FACILITIES.....
.....PF5 FTP EXECUTION.....
.....
.....HOST FILE   CP/M FILE    T/R.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....
.....
.....
.....
```

Figure B.2.e Transfer Specification panel

The panel shows ten lines of input fields where a file transfer may be specified in each line. The data in a file transfer specification line is composed of (from left to right) host file name, workstation drive, workstation file name, workstation file extension name and transfer direction.

When the editing has been terminated the operator may press PF3 to start the validation of the information, i.e. the files and the transfer directions.

The Transfer Specification Validation panel is always selected with the PF3 key. The panel is used when the keyed in transfer specification information is supposed to be correct and the actual file transfer is to be started.

The panel will also appear if PF5 is pressed. PF5 means that the file transfer(s) should be started, but before the file transfers can take place FTP will automatically validate the actual file transfer specifications. In this case FTP will also automatically proceed to the FTP Execution panel if no errors were found during the validation.

During the validation phase the validity of each transfer is shown with a message to the right in each transfer specification line.

```
ITT3297.FTP.....TRANSFER.SPECIFICATION.VALIDATION.....REL.2.00..XXXXX
.....
.....ESC MAIN MENU.....
.....PF1 READ TRANSFER SPECIFICATION FILE.....
.....PF2 TRANSFER SPECIFICATION .....
.....PF4 VOLUME/FILE FACILITIES.....
.....PF5 FTP EXECUTION.....
.....
.....HOST FILE   CP/M FILE    T/R...VALIDATION RESULT.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....________  _ ________ ___  _.....
.....
.....
.....
.....
```

Figure B.2.f Transfer Specification Validation panel

If PF3 was pressed or if PF5 was pressed and one or more errors were found then the Transfer Specification Validation panel is also shown when the validation has been finished. The messages associated with each file transfer specification will show if further editing is necessary or if the file transfers may be started.

If PF5 was pressed and no errors was found during the validation FTP will display the FTP Execution panel and start the file transfers.

The Volume/File Facilities panel (figure B.2.g) is always selected with the PF4 key. The panel is used when the operator wants to do some local file or volume operations.

```
ITT3297.FTP.....VOLUME/FILE FACILITIES.....REL.2.00..XXXXX
.....
.....ESC MAIN MENU.....
.....PF1 READ TRANSFER SPECIFICATION FILE.....
.....PF2 TRANSFER SPECIFICATION.....
.....PF3 TRANSFER SPECIFICATION VALIDATION.....
.....PF5 FTP EXECUTION.....
.....
.....1 DELETE 3290 FILE.....
.....  _ ________ ___.....
.....2 RENAME 3290 FILE.....
.....  ________ ___  _ ________ ___.....
.....3 CP/M FILE SIZE.....
.....  ____________ ._______.....
.....4 CURRENT LOGGED VOLUME.....
.....  _.....
.....5 VOLUME STATUS.....
.....  __ __ __ ____ _______ _______.....
.....6.DIRECTORY LIST.....
.....  _.....
.....
.....
.....
.....
```

Figure B.2.g Volume/File Facilities panel

The panel exposes 6 different volume/file operations: Delete file, rename file, get file size, get/set current logged volume, get volume status and get directory list. The function to be executed is selected with the numeric keys and the cursor will as a result be placed in the (first) input field associated with the function, and the operator may then key in the necessary parameters.

Deleting a file requires the volume and the file name of the file to be deleted. Renaming of a file requires the volume and the old and the new file name. Getting the size of a file requires the volume and the file name in question. Current logged volume is specified by keying A or B. Volume status is obtained by specifying the volume. Directory list is in the same way obtained by specifying the volume.

If a display of a directory list (volume A or B) is specified then a special panel (figure B.2.h) is shown. This panel is only showed when the Volume-/File panel has been selected and the directory list (function 6) has been specified.

```
ITT3297.FTP.......DIRECTORY LIST.............................REL.2.00..XXXXX
..PRESS <ESC> TO RETURN.....
```

Figure B.2.h Directory List panel

The panel shows up to 128 file names from the specified directory (volume A or B).

The FTP Execution panel (figure B.2.i) i.e. the start of file transfers is always selected with the PF5 key. As described above the Transfer Specification Validation panel will appear first and only if no errors were found will FTP proceed to the FTP Execution panel.

```
ITT3297.FTP.....FTP.EXECUTION.....REL.2.00..XXXXX
.....
.....PF7 TRACE MODE.....
.....PF8 STEP MODE.....
.....PF9 CLEAR RECEIVED DATA.....
.....PF10 CLOSE RECEIVED FILE.....
.....PF11 RESTART PRINTING.....
```

Figure B.2.i FTP Execution panel

During the file transfer session the panel will show some progress information which will at least consist of the file names involved. With the trace function on more information from the command region is shown along with the data transferred.

Termination of FTP operation is always selected with the PF6 key. FTP will for a few seconds display the FTP Termination panel (figure B.2.j) and then reboot the CP/M system.

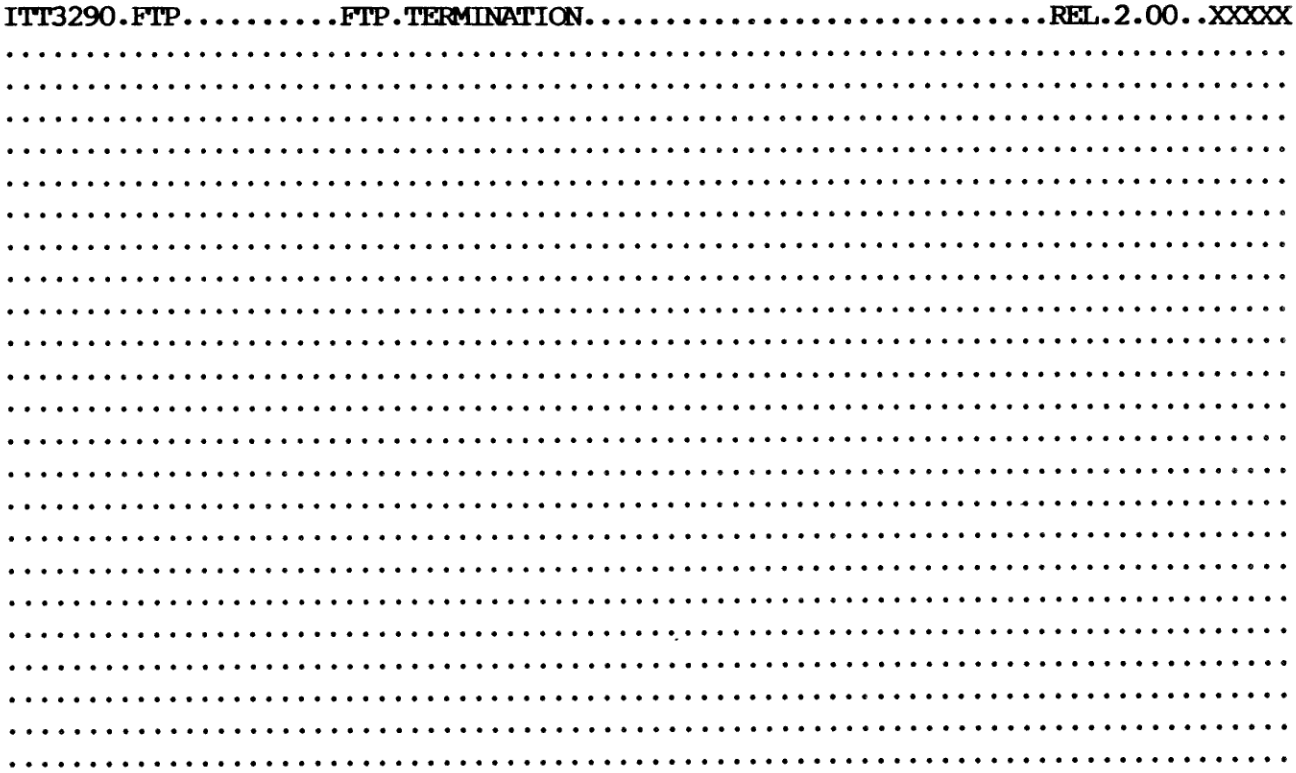


Figure B.2.j FTP Termination panel

C. A File Transfer Protocol Frame Work Application (FTPFWA)

This appendix serves as an illustration of how the FWA is implemented under CICS/VS.

C.1 General System Description

The Host Framework Application, FWA, is designed as a subsystem to run under CICS/VS using the 3270 Information Display System.

The FWA provides for sending or receiving user-generated files between the host and any ITT3297 with CP/M.

The FWA should principally be regarded as a framework, which can be changed in accordance with the individual user's needs. However, it is possible to install and run the FWA without any changes at all.

C.1.1 System Objectives

The FWA consists of the following two main procedures:

- Creation & maintenance of Destination File Table
- Management of the file transfer

The destination file table (DFT) contains one entry for each of the files, which forms part of the file transfer.

These entries, which are maintained by the CICS/VS system administrator, are required in order to secure proper treatment of files and correct communication between the host and the ITT3297.

The FTP package contains a set of panels for the purpose of creation and maintenance of the DFT.

The procedure for management of the file transfer is divided into the following four sub-procedures:

- Initiation of the file transfer
- Validation of files to be transferred
- Control of the transfer
- Transfer disk interface

C.1.2 System Relationships

Figure C.1 shows the logical relationships of the system. The illustration only shows relations between the FWA & CICS/VS as it is insignificant whether the FWA runs under DOS/VS or OS/VS.

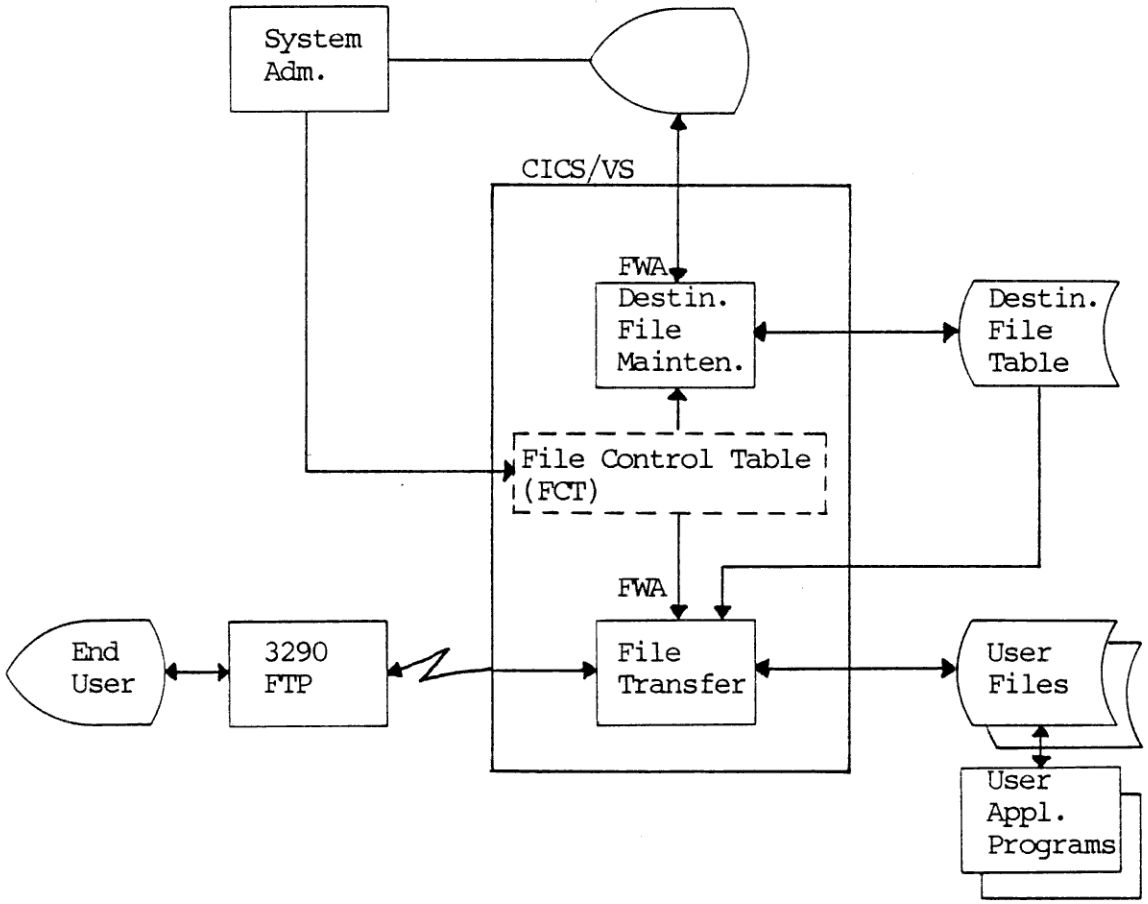


Figure C.1 FTP-CICS/VS Logical System Relationship

C.2 Programs and Modules

This section describes the programs and modules which together form the FWA software.

C.2.1 Program Structure

Figure C.2 shows the structure of the FWA, with each box representing a program or a module. All programs are written in COBOL.

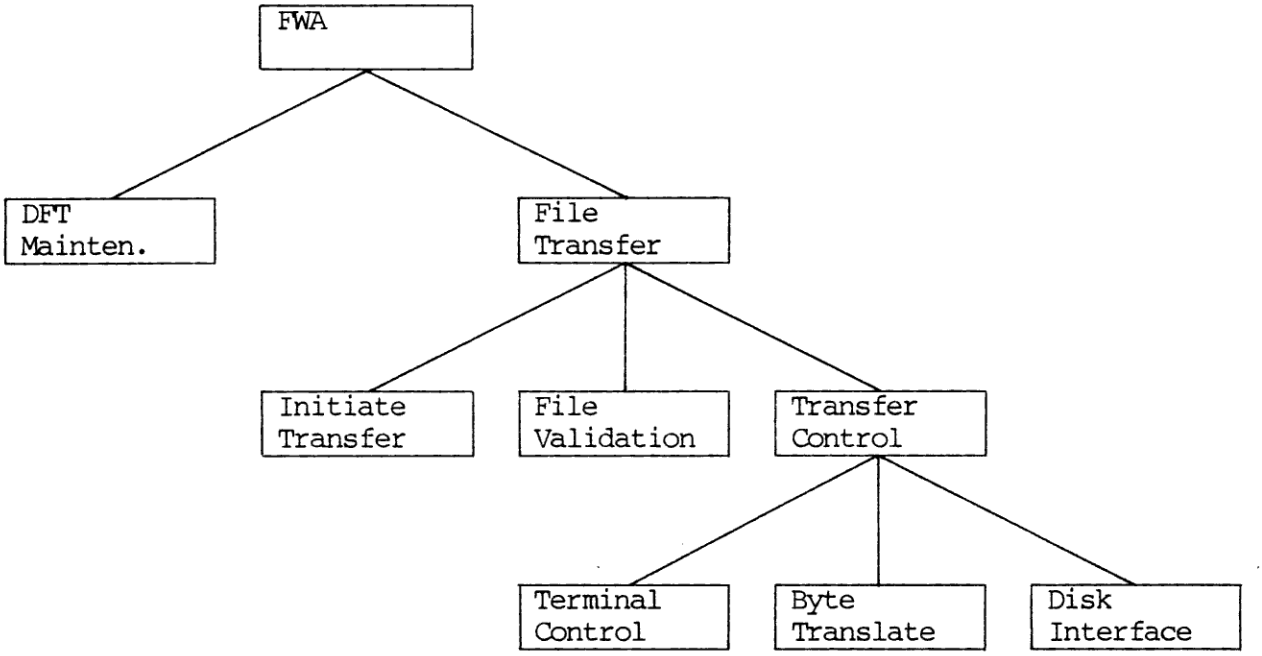


Figure C.2 FWA Program Structure

C.2.2 General Description

The Destination File Table procedure is designed as an aid to the CICS/VS system administrator in maintaining the various information about the files transferred which are necessary for the FCT facility.

The procedure, which simply consists of two panels, offers the possibility of adding, changing or deleting entries in the DFT.

In addition to the two above panels, some help panels are included, which in detail describe all fields in a DFT-entry and their allowable values.

When a file transfer is to be started, the ITT3290 terminal operator keyes in a transaction-id, which initiates the FTP-procedure (host FTP).

The procedure returns a panel telling the operator to start up the file transfer program on the ITT3290 (3290-FTP).

After that, the operator one at the time keyes in, the names of the files to be transferred.

The 3290-FTP sends the file names to the host-FTP, which validates each file name against the corresponding DFT-entry and the CICS/VS File Control Table.

A response is returned to the 3290-FTP, indicating whether the request was erroneous, or otherwise includes information necessary from the DFT-entry for the later transfer.

When all files are validated, the actual transfer takes place. Each individual transmission takes the form of a logical IBM 3270 panel.

Before the transmission takes place, the panel is built and formatted in accordance with the File Transfer Protocol and the information from the DFT-entry for the current file.

When the panel is received, it is "decoded" in a similar way.

Regardless of the transfer direction, the receiving destination always returns a receipt before the next panel can be sent.

On the host data is written to or read from fixed pre-defined files by means of a disk interface module, using the VSAM access-method.

If necessary this module can easily be either changed to another access-method or replaced by a user-written CICS/VS-module.

During a transfer session, the ITT3290 terminal operator has the possibility of interrupting the transfer, forcing it to stop before normal end-of-file.

In this case the operator likewise will instruct the receiving destination to keep or to delete the file just received. Only target files can be deleted in this way.

C.2.3 Destination File Table Maintenance

The procedure provides the ability of maintaining DFT-entries for files to be transferred.

An entry in the DFT as well as in the CICS/VS FCT, is required for each file participating in the file transfer.

The CICS/VS FCT is maintained in its normal way, while maintenance of the DFT requires a non-CICS/VS supported function.

This function, which is developed for that specific purpose, covers the following functions:

- Add an entry to the DFT
- Change an entry in the DFT
- Delete an entry in the DFT
- Summarise all entries in the DFT

When changes to the DFT are required, care must be taken not to execute the function while one or more transfer sessions are running, as this could cause unpredictable results on the files being transferred. Figure C.3 shows the principal work of the DFT maintenance procedure.

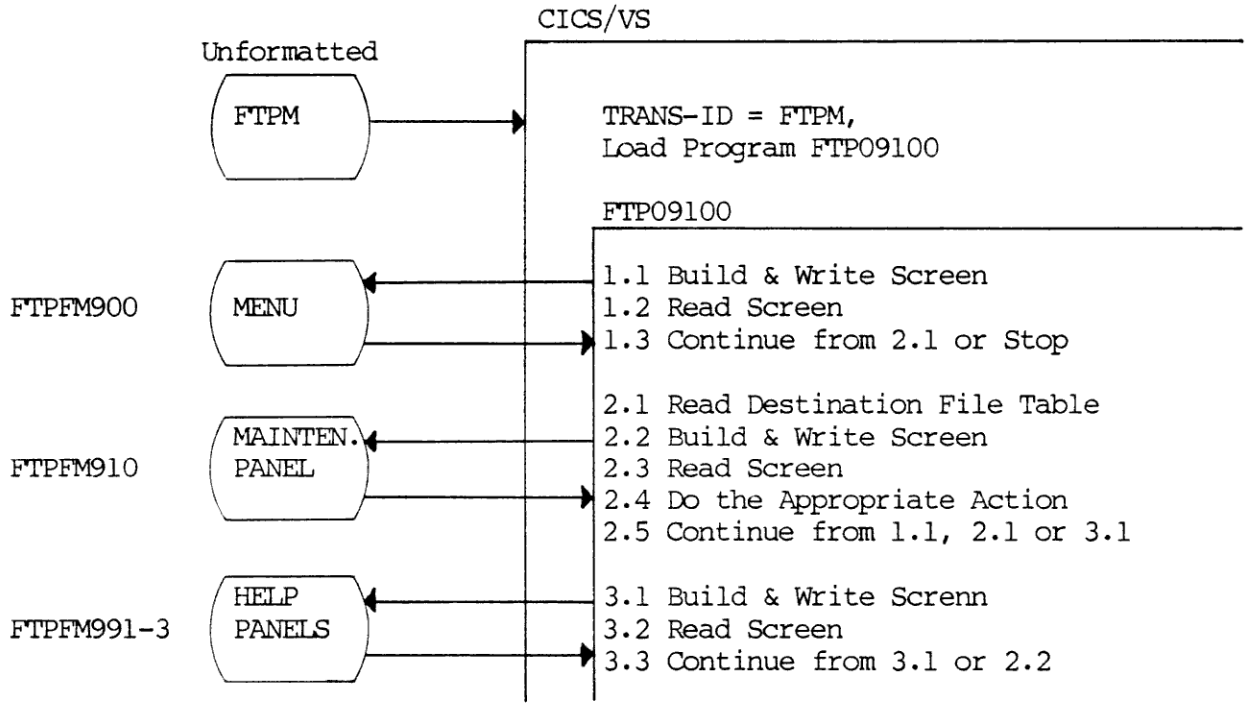


Figure C.3 Principal Work of the DFT Maintenance Procedure

The contents of the procedure are as follows:

- program FTPMAIN - initiated by trans-id FTPE

- panel FM000 - Menu
- panel FM001 - DFT-entry summary
- panel FM100 - Help-panel 1
- panel FM101 - Help-panel 2
- panel FM102 - Help-panel 3

By initiating the procedure, trans-id FTPM is entered from an unformatted screen.

In addition to this all of the above mentioned screens are read and written in CICS/VS Basic Mapping Support (BMS).

The Destination File Table is treated as a key-sequenced VSAM file.

Refer to section C.3.2 for a detailed description of the Destination File Table.

Refer to section C.6.3 for guidelines on how to installate the procedure.

When running the DFT maintenance procedure, the following rules must be observed:

- the procedure should not be executed, while one or more file transfer sessions are running
- when creating a DFT-entry, the associated CICS/VS file name must be pre-defined in the CICS/VS File Control Table (FCT)
- all fields in the DFT-entry are mandatory
- special care must be taken when values of the byte translation code and the time elapse code is inserted in order to minimize transmission time

C.2.4 Initiation Function

The purpose of this function is to initiate a file transfer session by writing a start message to the 3290 terminal operator.

The program is initiated by a transaction-id keyed in by the operator.

The program builds a panel containing the start-message and writes it to the terminal.

After this has been completed, the program terminates with a new transaction-id telling CICS/VS to start up the file validation function, next time a message is received from that particular terminal.

Figure C.4 shows the principal work of the functon.

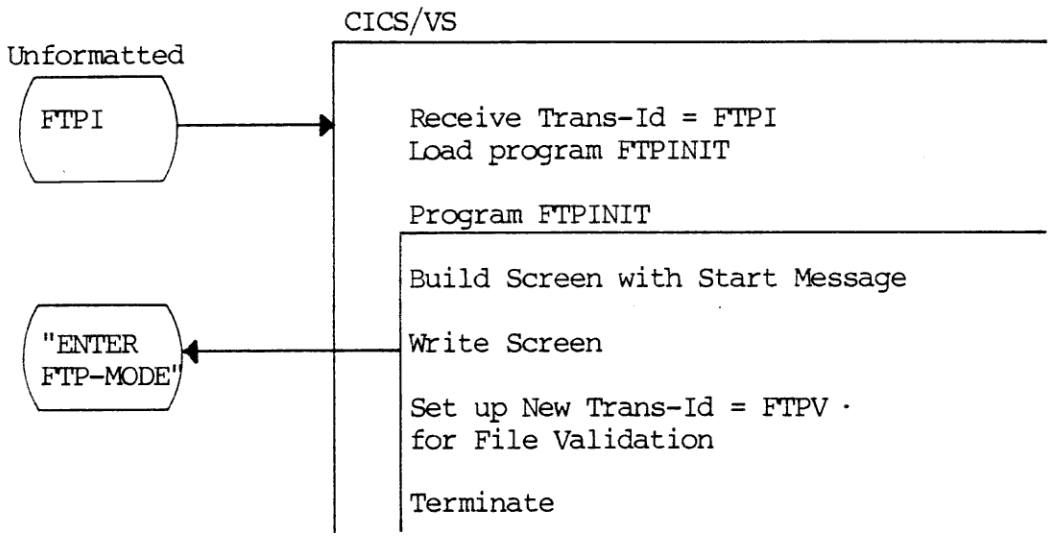


Figure C.4 Principal work of the initiation function

C.2.5 File Validation

The purpose of this function is to control the validity for requested files to be transferred between a 3290 terminal and the host.

It is initiated when the 3290 FTP sends a message telling the file validation function to start.

Initiation takes place by means of logical screens, which however never will be shown on the terminal.

The file validation is divided into the following steps:

- initiation of file validation
- validation of files against the DFT
- validation of files against the FCT
- response to the 3290 FTP
- termination of the file validation

Which step is to be performed depends on the Function Code received from the 3290 FTP.

If an invalid Function Code is received, an error message will be sent to the terminal.

If the Function Code asks for validation, the DFT will be read with the received file name as key.

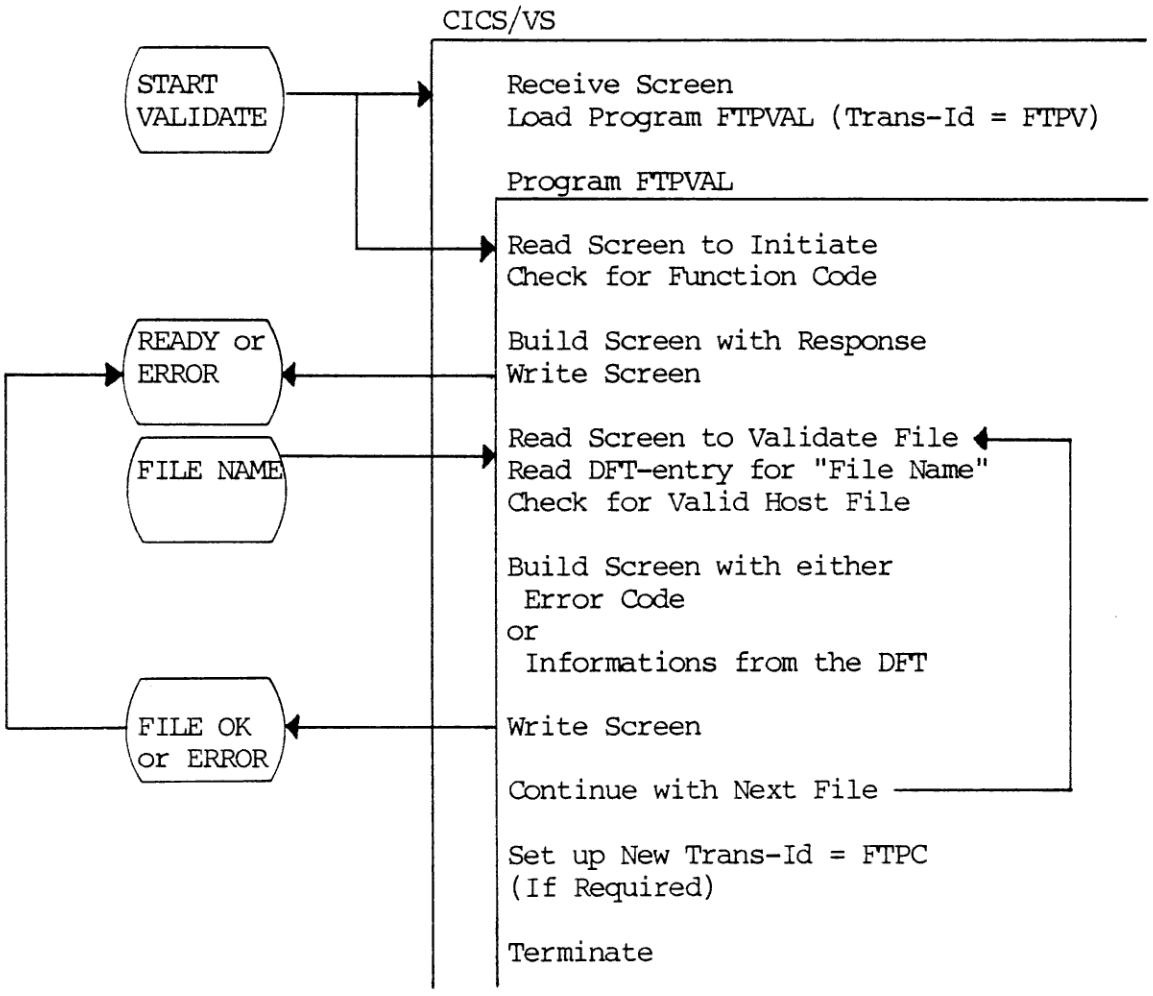


Figure C.5 Principal work of file validation function.

For valid file names, the corresponding host file is checked for availability.

After this, a panel is built containing either an error code if the request was erroneus, or all information stored in the appropriate DFT-entry together with a status code indicating a correct request.

These functions are performed once for each file to be validated.

When there are no more files to be validated, the function terminates with a trans-id telling CICS/VS to start up either the transfer function or to terminate the session, depending on the Function Code received from the terminal.

Figure C.5 shows the principal work of the file validation function.

C.2.6 Transfer Control

This function, which is subdivided into minor procedures, manages the actual file transfer.

The primary reason for subdividing is to ease changes to the FWA in case the individual user has special requirements to access methods, file specifications etc.

However, special care must be taken when changes are made to the modules, which are using and preparing information used by the File Transfer Protocol.

The transfer control is divided into the following procedures:

- session control
- terminal control
- byte translation
- file interface

A transfer session, which all the time it is running is managed by the 3290 FTP, can transfer any number of files in both directions, but only one at the time.

If a file transfer fails before completion, it is brougth to the operator's notice whereafter he is offered the possibilty of either deleting the receiving file or keeping it even if it is incomplete.

Like the file validation rutine, the transfer control operates due to a sequence of Function Codes. Furthermore each logical panel is identified with a special label so that only valid screens will be treated by the transfer control.

As there may be some elapse of time between the file validation and the actual file transfer, the transfer control once again controls the validity of the file to be transferred. This is done to make sure that the file still exists, and that no other task or transfer session is using it.

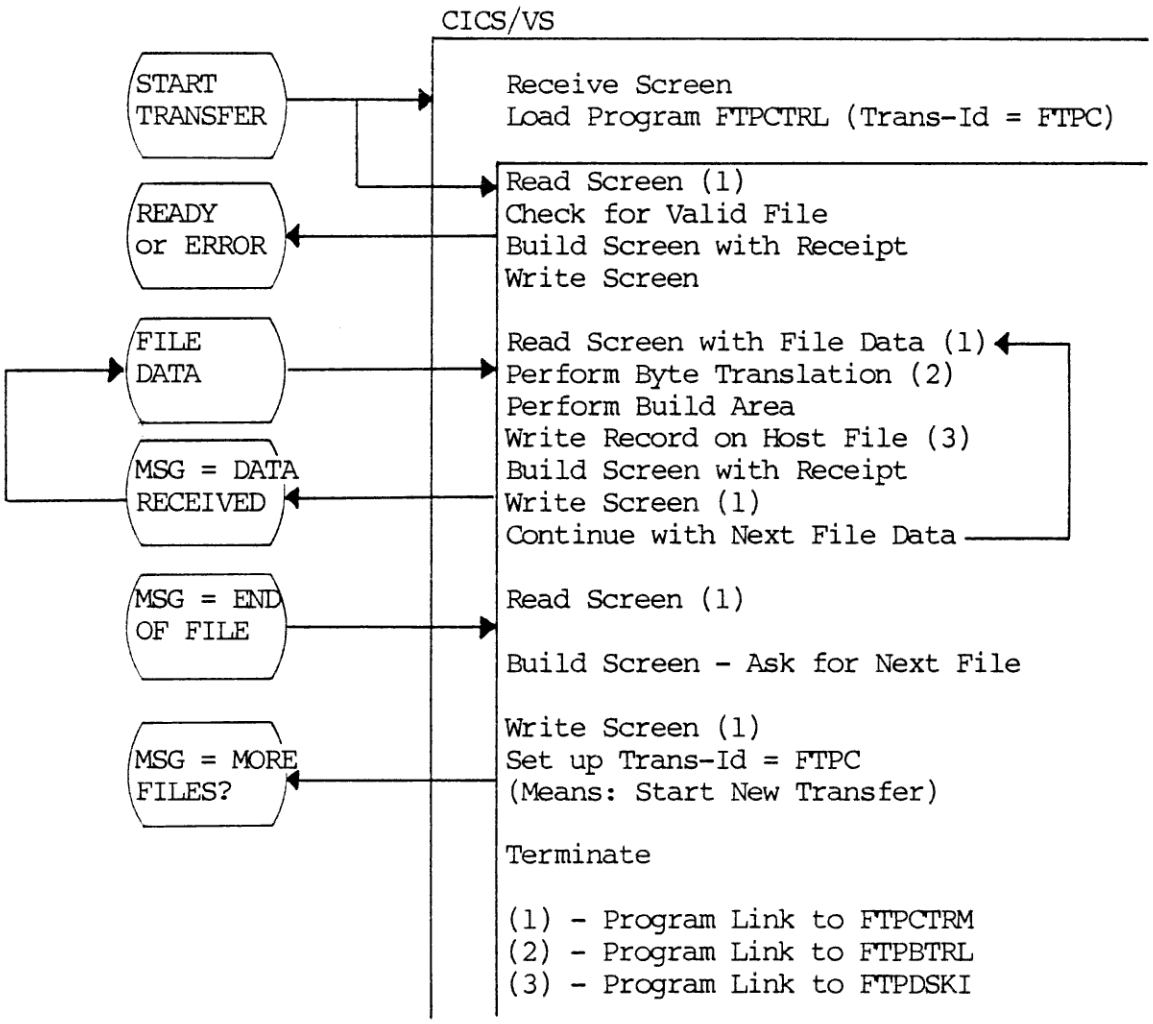


Figure C.6 Principal work of the transfer control

Figure C.6 on previous page shows file transfer from the 3290 terminal to the host. Transfer in the opposite direction takes place in the same manner, except that it is the FWA, which issues the EOF message.

CODE	MEANING
00	Normal response.
01	Host file is either undefined or closed. The CICS/VS system administrator must be consulted to solve the problem.
02	Transfer direction is invalid. The specified transfer direction does not match with that specified in the corresponding DFT-entry.
03	Host file is not empty. A transfer is requested from the terminal to the specified host file, but the file does already contain data. The CICS/VS system administrator must be consulted in order to solve the problem.
04	Host file is empty. A transfer is requested from a host file to terminal, but the file does not contain any data.
05	No more space on host file. Occurs when no space is available on the host file for additional records. The CICS/VS system administrator must be consulted.
06	No entry found for the requested file. A file name, for which no corresponding host file could be found, was recieved. The CICS/VS system administrator must be consulted in order to solve the problem.
07	Block-id received twice. The same block-id (0 or 1) is received two succeding times. Probably a transmission error or a program error.
09	Invalid transfer code. A transfer code, which is not valid in relation to the previous transfer code, was recieved. This is probably caused by a system or programming error. Stop the transfer session and try to run it again. If the problem occours after this, the distributor of the 3290 FTP should be consulted.
98	No input data recieved. A message was recived from the terminal, but there was no data in it (i.e. the ENTER - key or a PF-key was depressed. The message code issued, when the host FWA expects a transfer code. This could be caused by either a system or a program error. Stop the transfer session and try to run it again. If the problem recours after this, the distributor of the 3290 FTP should be consulted.
99	An unrecoverable error is encountered. There is no possibility for the host FWA to solve this problem. The appropriate program terminates abnormally with a CICS/VS transaction dump. The dump must be analyzed by CICS/VS system administrator to determine the nature of the error and take the necessary action in order to eliminate the cause of the error.

Figure C.7 FTP status codes returned from the FWA

During a transfer session Function Codes are only accepted according to the protocol described i chapter 3. Each time the transfer control program receives a Function Code from the terminal, this code is checked against the previous received code. If there is any kind of illogic between the current and the previous Function Code received by the terminal, an error code (i.e. a nonzero status code) is returned to the terminal. Figure C.7 on previous page shows status codes returned to the terminal by the FWA.

The following describes in short the major process of the procedures, which are included in the transfer control procedure.

Transfer Control (FTPCRTL)

This is the main procedure, which controls all functions to be performed through a transfer session by issuing program links to the subroutines, depending on the required action.

All communication with these procedures takes place by means of a CICS PROGRAM LINK, where data to and from the procedures are passed through the CICS Commarea (DFHCOMAREA)

In addition to this, certain function and returncodes are passed between the procedures. These codes will not be described in this section, but will appear in the source listings of the procedures.

Terminal Control (FTPCTRM)

The procedure is called each time a message (equals a logical panel) is to be written to or read from the terminal.

When a terminal write is to be performed, data is received in the TWA in a ready-to-use format. On the basis of this, the Terminal Input/Output area (TIOA) is built.

The procedure can receive either a normal panel or a "request for answer" from the terminal. In the latter case the last sent panel will be written to the terminal once more.

When a normal panel is received, data is moved from the TIOA to the TWA for further treatment, whereafter the procedure returns to the invoking procedure.

Byte Translate (FTPBTRL)

The procedure receives an area of data in the TWA and translates each individual byte to a new format in accordance with the content of the field in the DFT-entry for the appropriate file.

Before writing to the terminal, one byte is divided into half-bytes, which are expanded to one full byte, each in order to meet ASCII/EBCDIC standards.

When receiving data from the terminal, one byte is created for each two bytes that are received. On the terminal the same thing happens in reverse order.

C.2.7 File Interface

This is the procedure handling all access to and from the files being transferred.

As the procedure only supports the VSAM access method using the Relative Record Organization, this is the primary procedure to be changed if the user has special requirements (see section C.3.1 - Source and Target files - for a detailed explanation of the access method used).

Depending on the requested action the procedure handles the following :

- one time prior to the transfer of each individual file, it checks for the presence of the file and secures that the file is not being used by any other task. Yet, the latter will only be done for target files. Besides that, it is controlled that the appropriate DFT-entry is present as it has to be used for later transfer.
- on the basis of the action, requested records are either read from or written to the file. To manage this, information about the file (i.e. record length, file name) is fetched from the DFT-entry.

Records are received from and sent to a work area, which is used by the Transfer Control Procedure (FTPCTRL).

As the procedure is invoked by a CICS PROGRAM LINK command, it could easily be replaced by a user-written routine, if the user has special requirements for the further treatment of data.

Moreover, it is possible to change the organization to key-sequenced organization or quite a different access method.

C.3 File Support and Handling

The FWA as present supports only VSAM files. However, the FWA is designed modular in order to ease posible changes in access methods.

Each file participating the file transfer, must be defined on the host. This is accomplished by means of the Destination File Table (DFT).

The DFT is maintained by the CICS/VS system administrator. He also maintains the CICS/VS File Control Table (FCT), which must include at least two entries for the following:

- Destination File Table (DFT)
- the file which is source or target for the transfer session

For each 3290 file there must be a corresponding entry in both the FCT as well as in the DFT.

C.3.1 Source and Target Files

These are the files residing on the host to be used in the transfer session. Source files are files used for transfer to the 3290 terminal. Target files are files used for transfer from the 3290 terminal to the host. In some cases a file can be used in both directions.

The files being used for the transfer sessions are all accessed by Virtual Storage Access Method (VSAM). However, it is possible for the individual user to change this to either Direct Access Method (DAM) or any other access method. For this purpose the file access procedure is placed as a separate

module in the FWA-structure. This module can easily be replaced by a user-written module covering the required access method. In addition to this, it is possible to replace the module by a user-written module which simply issues a program-link to another user-written module.

Records in above mentioned files are of fixed length and are stored, using the VSAM Relative Data Organization. This means that logical records are stored and retrived on the basis of their record numbers, relative to the beginning of the file starting with one. Records are always processed by keyed-sequential processing.

When user applications create these files for transfer to the 3290 terminals, care must be taken not to create files, which are unable to reside on the floppy disks on the 3290.

Refer to section C.6 and C.7 for a detailed description on how to install and maintain the files and their appropriate entries.

C.3.2 Destination File Table

This is the file containing all the entries which link together the files residing on the 3290 and the files on the host. It includes one entry for each file to be transferred. This means that all changes concerning host files, which are used for transfer, must be done for both the CICS/VS FCT and the DFT.

This could cause minor problems as the DFT can be updated on-line when the CICS/VS FCT must be updated off-line to CICS/VS.

Yet files to be deleted will cause no problems, while it is the DFT entries which control the transfer, i.e. a file, which is defined in the FCT but which has been deleted from the DFT, would cause an error code to be written to the terminal if any attempt is made to transfer the file. The same would occur for a file, which is defined in the DFT, but not in the FCT.

The DFT file is accessed using the Virtual Storage Access Method (VSAM). Records are of fixed length and are stored using the VSAM Key-sequenced Data Organization.

This means that logical records are stored and retrieved on the basis of a collating sequence, determined by the content of the prime key of those records (this organization is basically similar to the organization of an ISAM file).

If the user desires to change the access method, changes must be done to as well the DFT maintenance procedure as the FWA-procedure.

Refer to section C.6 and C.7 for a detailed description on how to install and maintain the DFT.

FIELD NAME	LENGTH	FORMAT	DESCRIPTION
HOSTNME	8	C	File name defined in the CICS/VS FCT key field.
TRANSFER	1	C	Code, which indicates the transfer direction.
FORMAT	1	C	Code, which indicates ASCII or split-code transfer.
RECLEN	2	B	Record length for the host file.
BLOCKING	2	P	Block factor used for building a logical screen.
TIMING	2	P	Value, which indicate delay on the terminal before next transmission.

Figure C.8 Description of the contents of a DFT-entry

Note: As the CICS/VS requires at least one record in every VSAM file before it can be processed, the user must run a one-time task of two steps, where step one simply creates a dummy record and step two deletes it.

This is the most easily done by using a card-to-disk utility.

C.4 Name Conventions

The user must establish name conventions to ensure that duplicate names are not given to entries in libraries.

The following information should be considered when establishing local naming conventions:

C.4.1 Programs

Program names should be installated by the names mentioned in section C.6, Installation.

It should be noticed that all programs belonging to the FWA have the same prefix, namely FTP. The same convention applies to transaction IDs.

In case the user decides to change these names in order to meet local and different conventions, this must be done with regards to the conventions for CICS/VS.

C.4.2 Files

For file names the same rules apply as for program names. Yet it must be noticed that the file name for the file including the Destination File Table (DFT) is used explicit in the DFT maintenance program as well in the file validation and transfer control programs.

Furthermore the file names must be given to avoid conflict with other application labels in the CICS/VS FCT.

Each time a FTP-session is initiated, the FWA will check for that correspondance for each file to be transferred. If this correspondance is not present, the transfer for that particular file will not take place.

C.4.3 Screens/Panels

The DFT maintenance procedure communicates by means of BMS and uses mapsets, which are defined for COBOL use. If names in these mapsets are changed, the corresponding names in the appropriate program should be changed in accordance to this.

The transfer part of the FWA communicates with the terminal in native mode and as so, there will be no conventions to observe.

However, the FWA, as well as the 3290, will make certain checks to ensure proper communication while a transfer session is running, as each screen in a particular transfer is identified by a screen name.

This should be kept in mind when developing local on-line applications.

Refer to chapter 3, File Transfer Protocol, for further information about this.

C.5 System Requirements

This section describes system configuration requirements, storage requirements, communication requirements and CICS/VS requirements.

C.5.1 Programming Systems Requirements

The Host Framework Application (FWA) is written mainly in S/370 COBOL and operates under control of IBM Customer Information Control System (CICS/DOS/VS 5746 or CICS/OS/VS 5740).

The FWA is using the CICS Command Level Interface, but can be easily changed to CICS Macro Level if required.

C.5.2 System Configuration

The machine configuration for CICS/VS varies accordingly to the user's application needs.

However, the FWA will be able to execute under any CICS/VS environment with sufficient storage to meet the combined operating requirements of CICS/VS, user required applications, and the FWA.

Sufficient DASD space must be included to support the requirements for files and their appropriate entries for the FWA.

C.5.3 Storage Requirements

The storage requirements for the FWA written to run under CICS/DOS/VS are approximately as follows:

- DFT maintenance routine 15000 bytes
- Host-FTP (with Disk Interface) 36000 bytes

Beyond this, FWA demands no special requirements for storage allocation.

However the above mentioned storage requirements is slightly affected by modifications made to the FWA by the user.

C.5.4 Communication Requirements

The FWA is designed to run under CICS/VS using the Basic Telecommunications Access Method (BTAM) for the IBM3270 Information Display System and as so, it takes no special regards to other communication methods as VTAM or TCAM for Systems Network Architecture (SNA).

All communication to and from the FWA is done in native mode, and should not be altered to use Basic Mapping Support (BMS).

C.5.5 CICS/VS Requirements

The following optional CICS/VS components are required by the FWA:

- Dump Management (dump control program)
- File Management (file control program)
- Temporary Storage Management (temporary storage control program)

Additionally there are the following CICS/VS table requirements:

- Terminal Control Table (TCT)

  the terminal control table must have appropriate entries for each device with which the FWA communicates during transfer sessions.

- Processing Program Table (PPT)

  the PPT must have an entry for each of the FWA programs including the DFT maintenance program. All programs are quasi-reentrant.

- Program Control Table (PCT)

  the PCT must have an entry for each of the FWA transactions.

  The transaction ID's are pre-defined for all the required entries but they could be user-defined.

  In the latter case care must be taken to ensure proper change as some of the FWA-programs returns with TRANS-ID in order to automatically initiate the next program.

- Destination Control Table (DCT)

  is not required for this version, but could be requisite if the user desires to change the Disk Interface program to use another access method than VSAM.

- File Control Table (FCT)

  the user must code a file control table entry for the file including the Destination File Table and each of the files being used in connection with the FTP.

  However, only one entry is needed in case all the 3290-files are to be directed to the same host file.

  In case the access method has to be changed to alternatively use files defined in the DCT, the entries in the FCT can be omitted.

See section C.6, Installation, for a detailed description of names and entries to be created in the various CICS/VS tables.

C.6 Installation

This section describes the contents of the distribution medium and how to install the programs.

C.6.1 Contents of the Distribution Medium

The distribution medium for the FTP-FWA consists of an unlabeled tape written in 9-track mode with a density of 1600 bpi. Record length is 80 bytes and blocksize is 3440 bytes.

The tape consists of all the programs and modules, which are used by the FWA, including the DFT maintenance procedure.

Each program/module is written on the tape as separate files. This means, that the tape has the following layout:

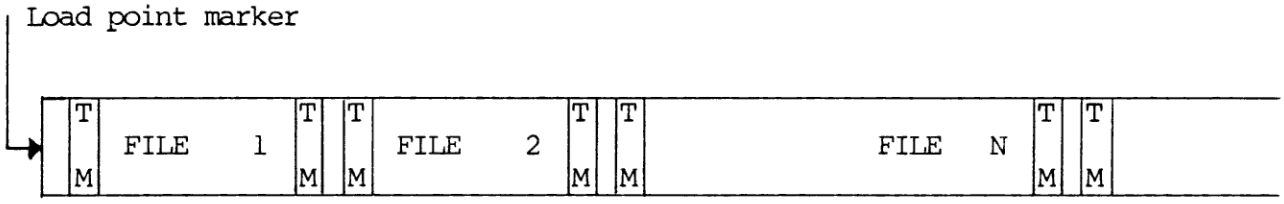


Figure C.9 Magnetic tape structure

The tape does not contain any utility program or other installation tools, as all programs and modules are distributed in source format with no regards to the use under DOS/VS or OS/VS.

File	Name	Source st.mts.	Description
1	FTPINIT	100	Initiation Procedure
2	FTPVAL	368	File Validation
3	FTPCTRL	648	Transfer Control Program
4	FTPCTRM	240	Controlling the Terminal Read and Write
5	FTPBTRL	286	Byte Translate
6	FTPDSKI	192	Disk Interface Module
7	FTPMAIN	632	DFT Maintenance Procedure
8	FTPMS00	101	BMS Mapset for the DFT Mainten.
9	FTPMS01	113	BMS Mapset for the DFT Mainten.
10	FTPMS10	56	BMS Mapset for the DFT Mainten.

Figure C.10 Contents of distributed medium

Note: File no 1-6 concerns the actual file transfer, while file no. 7-10 is used for DFT maintenance. The latter could be omitted if the file entries and their corresponding values are placed as litterals in the file transfer program. Number of source statements are only approximate values.

Thus, it is the user's responsibility to load the programs in source libraries using utility programs developed for use under the appropriate operating system.

For DOS/VS, this could be done by using the DOS/VSE Data Interfile Transfer, Testing and Operations utility (DITTO).

For OS/VS, it could be done by using either the IEBCOPY, IEBGENER or IEBUPDTE utility.

Refer to Figure C.10 for description of the contents of the tape.

C.6.2 Installation Procedure

When installing the FWA, the following steps should be followed:

1. Review documentation

Be sure you did not miss anything at your first reading.

2. Load received material

Use the proper utility program to load all the programs received on the tape.

3. Prepare programs for use

All the loaded programs should be compiled or assembled before they can be loaded into the CICS/VS.

When compiling the programs it should be noticed that they all communicate with CICS/VS using CICS/VS Command Level.

Futhermore all programs must be compiled with option NOOPT.

This is an absolute must, as programs compiled with option OPT requires the CICS command SERVICE RELOAD after each EXEC CICS command. The SERVICE RELOAD command is not included in the present version of the FWA.

Refer to the tables below for compilation and assembly of programs.

Program name	Language
FTPMAIN	COBOL
FTPINIT	COBOL
FTPVAL	COBOL
FTPCTRL	COBOL
FTPCTRM	COBOL
FTPBTRL	COBOL

Mapset name	Language
FTPMS00	COBOL
FTPMS01	COBOL
FTPMS09	COBOL

Figure C.11 Program names and program languages

4. Perform required changes

If any changes are to be made to the the programs, they should be done at this time.

All changes must be made with regards to the structure of the FWA, the File Transfer Protocol specifications, and CICS/VS.

Special care must be taken when changes concern program names, transaction-id's and areas, which are used for communication between modules.

5. Update CICS/VS tables

Refer to section C.5, System Requirements, for a description of which components are required. In addition to this, the following updates must be made to CICS/VS.

- no changes are required for the CSA or the CSWA.

- the following components are required:

  Dump Management (dump control program)
  File Management (file control program)
  Temporary Storage Management (temporary storage control program)
  Basic Mapping Support (BMS)

In addition to this, entries are required in certain CICS tables for proper operation of the FWA.

Journal Control Table (JCT)

Not required.

Terminal Control Table (TCT)

The terminal control table must have appropriate entries for all terminals, which use the FWA during on-line processing.

File Control Table (FCT)

The user must code a file control table entry for each data file used in transfer sessions together with an entry for the Destination File Table (DFT).

If the user changes the access method from VSAM to sequential access method, entries are instead required in the destination control table.

The destination file table (DFT) is specified as a key-sequenced VSAM file with a fixed record length of 16 bytes.

The file name for DFT must explicit be defined as "PFJFTDA". Refer to figure C.8 for a detailed description of the contents of each individual record.

The source and target files are specified as VSAM files using the Relative-record Data Organization with fixed record length.

When many files are involved and frequent transfer may occur, special regards must be taken when defining the FCT-entries in order to minimize overhead. These regards consider the following fields:

- STRNO (string number)

  this value, which highly will affect the access-time, depends on the number of terminal users.

- CI (control interval size)

  must be lowest possible value in order to minimize physical access.

- CA (control area size)

  must be highest possible value in order to minimize physical access.

- depending on the nature of the individual file, it may be considered if, and how much, secondary allocation is needed.

Processing Program Table (PPT)

The PPT must have an entry for each of the on-line FTP programs. The programs are:

- FTPMAIN
- FTPINIT
- FTPVAL
- FTPCTRL
- FTPCTRM
- FTPBTRL
- FTPDSKI

The mapsets are:

- FTPMS00
- FTPMS01
- FTPMS09

If changes are to be made in accordance to local naming conventions, the above mentioned names could be changed.

Refer to section C.4, Name Conventions, before any changes are made.

PROGRAM CONTROL TABLE (PCT)

An entry is required in the PCT for each of the FWA transactions. The required entries are as follows:

```
DFHPCT  TYPE     = ENTRY
        TRANSID  = FTPM
        TRANSREC = mmm
        TRNPRTY  = nn
        TWASIZE  = 0
        TPURGE   = YES
        SPURGE   = YES
        CLASS    = SHORT
        PROGRAM  = FTPMAIN

DFHPCT  TYPE     = ENTRY
        TRANSID  = FTPI
        TRANSREC = mmm
        TRNPRTY  = nn
        TWASIZE  = 0
        TPURGE   = YES
        SPURGE   = YES
        CLASS    = SHORT
        PROGRAM  = FTPINIT

DFHPCT  TYPE     = ENTRY
        TRANSID  = FTPV
        TRANSREC = mmm
        TRNPRTY  = nn
        TWASIZE  = 0
        TPURGE   = YES
        SPURGE   = YES
        CLASS    = LONG
        PROGRAM  = FTPVAL

DFHPCT  TYPE     = ENTRY
        TRANSID  = FTPC
        TRANSREC = mmm
        TRNPRTY  = nn
        TWASIZE  = 200
        TPURGE   = YES
        SPURGE   = YES
        CLASS    = LONG
        PROGRAM  = FTPCTRL
```

If the above mentioned trans-id's conflict with earlier defined trans-id's, they could be changes to meet local conventions.

In this case the FTP programs, which return with trans-id, must be changed. These programs are as follows:

- FTPINIT
- FTPVAL
- FTPCTRL

Destination Control Table (DCT)

Only required if VSAM files are changed to sequential files.

6. Create Destination File Table (DFT)

For each source and target file, which is defined in the CICS FCT, the user must create an entry in the destination file table. Refer to section C.6.3, Destination File Table, for details about the installation.

Below is given an example of how to define a hostfile to be used for the FTP file transfer.

In the following is given an example of how to define a host file to be used for FTP file transfer:

Definition of DFT-entry

HOST-FILE	TYPE	FORMAT	RECLEN	BLOCK-FACT	TIME-FACT
PFJFT1C	T	A	80	20	00

Definition of VSAM-cluster

```
// EXEC IDCAMPS
     DEFINE CLUSTER                                -
         (NAME(PFDFT1C)                            -
         RECORDS(100 10)                           -
         BUFFERSPACE(1024)                         -
         SHAREOPTIONS(2)                           -
         VOLUMES(PVDS10)                           -
         NUMBERED                                  -
         )                                         -
         DATA                                      -
           (NAME(PFDFT1C)                          -
           CONTROLINTERVALSIZE(512)                -
           RECORDSIZE(128 128)                     -
           )
           CATALOG(PCDDADA)
```

Note: Due to the RRN organization the file must be initialized with at least one record, before the file could be accessed. The initialization record(s) need not be present at the time where the file has to be used.

Definition of CICS/VS FCT-entry

```
PFJFT1C   DFHFCT TYPE=DATASET,                                    *
            DATASET=PFJFT1C,                                      *
            ACCMETH=(VSAM,PRDS),                                  *
            SERVREQ(GET,PUT,UPDATE,NEWREC,BROWSE,DELETE),         *
            RECFORM=(FIXED,UNBLOCKED),                            *
            STRNO=1,                                              *
            OPEN=INITIAL
```

Note: According to individual needs there may be different values for the fields DATASET, STRNO and OPEN. All other fields must be defined as stated above.

7. Verify installation

Verification of the installed FWA package could be done in one of the following two ways:

1. Create a file on a specific 3290 terminal as it would be in a normal running environment. Start up a transfer session to transfer the file to a host file.
   Use a print utility program to make sure that the file has been received with proper content. After this, start up again the file transfer to transfer back the file to the terminal. When the session is completed, show or print the contents of the file on the terminal. If the received file is exactly equal to the file sent to the host, the FWA should be considered to work correctly.

2. Create a file on the host by using a card-to-disk utility. Start up the file transfer to transfer the file to the 3290 terminal. List or show the received file on the terminal. Start up again the file transfer to transfer the file to host. Upon completion, use a print utility to list the contents of the received file. If the contents of the received file is exactly equal to the file, which was sent to the terminal, the FWA should be considered to work correctly.

In case of any malfunction for during file transfer, the following action should be performed:

- reveiw the documentation and make sure that installation has been done correctly.

- perform the necessary changes and run the file transfer again as described above.

- if the problem still exists, save all listings and contact the distributor of the FWA package.

C.6.3 Destination File Table

Before a file transfer can be executed, the DFT must be created with an entry for each file being used in the transfer sessions. There must be an entry in the DFT for each file decribed in the CICS FCT.

The DFT maintenance procedure is developed for this purpose.

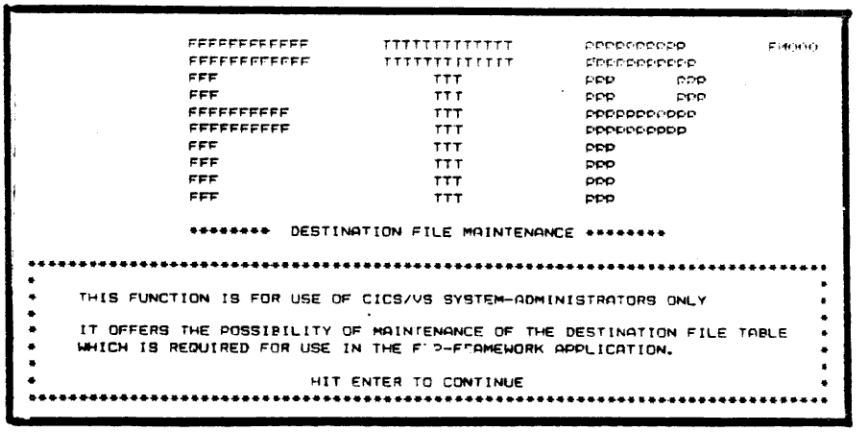
The procedure, which should only be used by the CICS/VS system administator, contains the following screens:

- DFT MAINTENANCE - MENU
- DFT MAINTENANCE - OVERVIEW AND CHANGE
- DFT MAINTENANCE - HELP-PANELS 1-3

To invoke the procedure, do the following:

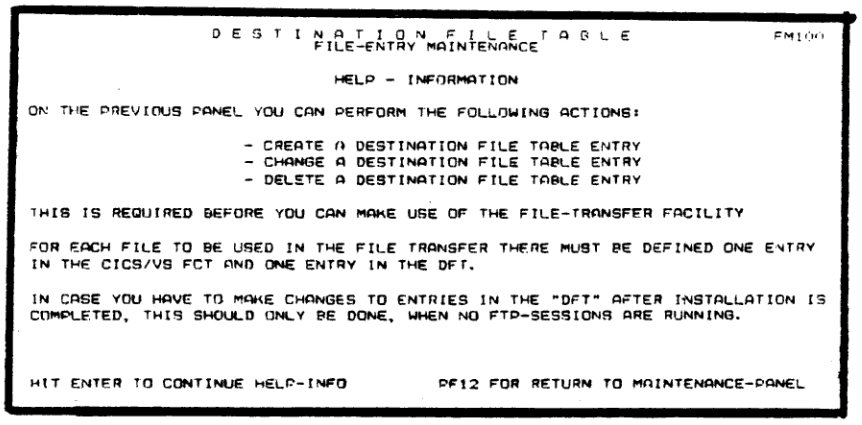
- sign-on to CICS/VS if required
- enter FTPM as the first 4 characters in the top left portion of the panel and press the ENTER key.

This will cause the following panel FM000, to be displayed



To continue the procedure, press the ENTER key.

After depressing the ENTER key on panel FM000, the following panel FM001, will be displayed:



On the panel files are shown previously defined in the DFT.

On the point of installation, the panel has no entries.

The user will now be able to create entries by keying in the required information.

If there are errors in any of the fields keyed in, an error message is displayed on the bottom line of the panel.

When all the fields keyed in are correct, the DFT is updated and a new panel containing all the entries is displayed.

If there is any doubt about the content and the allowable values for the individual fields, a help-panel can be displayed by de-pressing PF9.

The help-panel, which leads to two more help-panels, contains fully detailed information on how to fill in the individual fields and their allowable values.

Upon return from any of the help-panels, the original maintenance panel (FM001) is redisplayed without any loss of previously keyed in data.

As the help-screens are fully detailed about fields and values, no further information hereof is given.

After depressing the PF9 key on panel FM001, the following panel, FM100, is displayed:

```
                    D E S T I N A T I O N   F I L E   T A B L E                FM001
                          FILE-ENTRY MAINTENANCE

    FILE-NAME       TYPE      FORMAT      REC-LEN      BLOCK-      DELAY TIME
                                                       FACTOR      IN SECONDS

    X--------X      X         X           XXXX         XX          XX
    X--------X      X         X           XXXX         XX          XX
    X--------X      X         X           XXXX         XX          XX
    X--------X      X         X           XXXX         XX          XX
    X--------X      X         X           XXXX         XX          XX
    X--------X      X         X           XXXX         XX          XX
    X--------X      X         X           XXXX         XX          XX
    X--------X      X         X           XXXX         XX          XX
    X--------X      X         X           XXXX         XX          XX
    X--------X      X         X           XXXX         XX          XX
  ENTER ACTION: X  AND KEY IN FILE-ENTRY DATA (EXCEPT FOR DELETE)
     FILE-NAME    TYPE    FORMAT    REC-LEN    BLOCKING    TIMING
     X--------X     X         X        XXXX         XX          XX

  PF1 = PREV. PAGE
  PF2 = NEXT  PAGE          PF9 = HELP-PANEL          PF12 = RETURN TO MAIN-MENU
```

The user can depress either the ENTER key to achieve the next help-panel, FM101, or the PF12 key to redisplay the maintenance panel (FM001).

If the ENTER key was depressed, the following panel will be displayed:

```
                    D E S T I N A T I O N   F I L E   T A B L E                FM101
                          FILE-ENTRY MAINTENANCE

CONTINUED                          HELP - INFORMATION
ALLOWABLE VALUES FOR MAINTENANCE OF THE DFT-ENTRY:

ACTION:              1       CHARACTER INDICATING THE DESIRED ACTION
                             A = ALTER  ENTRY
                             D = DELETE ENTRY
                             I = INSERT ENTRY

FILE-NAME:           1 -  8  CHARACTERS DEFINING THE HOST FILE-NAME AS DEFINED IN
                             THE CICS/VS FILE CONTROL TABLE (FCT)

REC-LEN:             4       NUMERIC CHARACTERS DEFINING THE LOGICAL RECORD-LENGTH
                             FOR THE ABOVE MENTIONED FILE

TYPE:                1       CHARACTER DEFINING THE TRANSFER DIRECTION OF THE FILE
                             T = TRANSFER FROM 3290 TO HOST
                             R = TRANSFER FROM HOST TO 3290
                             B = TRANSFER IN BOTH DIRECTIONS
HIT ENTER TO CONTINUE HELP-INFO              PF12 FOR RETURN TO MAINTENANCE-PANEL
```

Again, the user can depress the ENTER key to get the last help-panel, FM102, or the PF12 key to redisplay the maintenance panel (FM001).

If the ENTER key was depressed, the following panel will be displayed:

```
                    D E S T I N A T I O N   F I L E   T A B L E                FM102
                          FILE-ENTRY MAINTENANCE

 CONTINUED                          HELP - INFORMATION

 ALLOWABLE VALUES FOR MAINTENANCE OF THE DFT-ENTRY:

 FORMAT:              1       CHARACTER DEFINING THE FORM IN WHICH THE INDIVIDUAL
                              BYTES WILL BE TRANSMITTED BETWEEN THE 3290 AND THE
                              HOST, DEPENDING ON THE CHARACTER SET USED
                              A = NORMAL ASCII-EBCDIC FORMAT
                              S = TRANSPARENT SPLIT-CODE FORMAT

 BLK-FACTOR:          2       NUMERIC CHARACTERS DEFINING THE SIZE (IN NUMBER OF
                              LINES) TO BE TRANSMITTED
                              MAXIMUM IS 23 (I.E. 1840 CHARACTERS)

 DELAY-TIME:          2       NUMERIC CHARACTERS DEFINING THE DELAY TO OCCUR BEFORE
                              EACH TRANSMISSION TAKES PLACE.
                              THE VALUE IS USED BY AS WELL THE HOST FWA AS THE 3290

 HIT ENTER TO STOP HELP-INFO AND RETURN TO MAINTENANCE PANEL
```

To stop the help-display, the user must depress the ENTER key. When this has been done, the maintenance panel (FM001) is redisplayed.

When executing the maintenance procedure several messages could be displayed on the panel as the result of an invalid operator action. The error, which caused the message, must be corrected before the procedure can be continued.

These messages and their meaning are described in the following.

ERROR - ONLY THE ABOVE MENTIONED KEYS MUST BE DEPRESSED

- the operator has depressed a PF-key, which must not be used for the appropriate panel. Depress the correct ENTER/PF-key to continue processing.

THE FIELD IS NOT NUMERIC

- a field is keyed in, which contains non-numeric characters (i.e. each byte has a value fron 0-9). The cursor is positioned at the first location of the appropriate field. The erroneous field must be corrected, before the process can continue.

MISSING OR INVALID FIELD

- a field is either missing or contains a value, which is not valid. The cursor is positioned at the first location of the appropriate field. The erroneous field must be correted, before the process can continue.

ACTION-CODE NOT A, D OR I

- an invalid action code is keyed in. The cursor is positioned at the erroneous action code which must be corrected before the process can be continued.

THE ENTRY BEING INSERTED ALREADY EXISTS

- action code I is specified for the file name pointed to by the cursor,

which already exists in the DFT. Review all the DFT-entries to look for redundant file names and correct the erroneous file name.

THE ENTRY BEING UPDATED OR DELETED DOES NOT EXIST

- action code A or D is specified for the file name pointed to by the cursor, but the entry for that file could not be found. Be sure that the appropriate entry is created before any attempt is made to update or delete it.

REQUEST EXCEEDS NUMBER OF FORWARD SCREENS

- a page forward has been requested and the current display is last screen of a multiscreen sequence.

REQUEST EXCEEDS NUMBER OF BACK SCREENS

- a page back has been requested but the current screen is the first of a multiscreen sequence.

C.7 Maintenance

In this section the program maintenance, file maintenance and Destination File Table Maintenance is described.

C.7.1 Programs

If a module is to changed for any reason, it must be compiled or assembled. When doing this, it should be noticed, that the option NOOPT must be specified.

However, if user standards require option OPT, the user must correct all modules to include the CICS command SERVICE RELOAD after each EXEC CICS command, in order to maintain proper addressability to the CICS/VS.

The modular structure of the FWA allows changes in all the modules to be

performed quite easily by skilled CICS/VS programmers. However care must taken not to damage addressability between modules or to violate the rules for the File Transfer Protocol.

If major changes are performed, which would influence the basic structure of the FWA, it is the user's own responsibility to secure not only proper addressability and program initiation, but also to make the necessary changes to the CICS/VS tables.

Before any changes are made, review the following sections, to be sure you quite understand the nature of the FWA and how it works:

Section C.2 - Programs and Modules
Section C.4 - Name Conventions
Section C.5 - System Requirements
Section C.6 - Installation

In addition to this, source listings would be a very helpfull guide, as they contain a rather large part of comments, which explain the individual steps in the process.

If changes are made to the BMS mapsets, the corresponding programs should be altered in accordance to this. The connection between programs and mapsets is the following:

Program	**Mapset**
FTPMAIN	FTPMS00 FTPMS01 FTPMS09

C.7.2 Files

File access methods could be changed to suit individual user requirements.

Changes could be performed for the following two types of files:

- the Destination File Table (DFT)
- the source and target files

Changes to the DFT are decribed in section C.7.3 - Destination File Table.

However, it should be noticed that any change to the DFT includes changes to the following other modules:

- FTPVAL - file validation rutine
- FTPCTR - transfer control, main procedure

If a new file is to be created for use in transfer session, an entry in the CICS/VS File Control Table (FCT) must be created first.

After this, an entry in the Destination File Table (DFT) must be added.

If a file is no longer needed for transfer, the appropriate entry must be deleted from the Destination File Table (DFT). If required, the corresponding entry in the CICS/VS FCT may be retained.

Refer to section C.6.3, Installation, Destination File Table, for a detailed description of how to create, update or delete file entries.

When changes are to be performed for the source and target files, the following modules must be changed:

FTPMAIN -	DFT maintenance, which looks up in the CICS/VS FCT
FTPVAL -	file validation , which checks the files for presence
FTPCTRL -	transfer control, which checks for presence of files just before transfer takes place.
FTPDSKI -	the file interface module, which performs the actual file access.

Additionally, it could be necessary to change the CICS/VS tables (i.e. CICS/VS FCT and/or CICS/VS DCT).

Before any change is performed, review the following sections:

Section C.2 - Programs and Modules
Section C.3 - File Support and Handling

Section C.5 - System Requirements
Section C.6 - Installation

C.7.3 Destination File Table (DFT)

The access method could be changed for the DFT in order to meet the specific user requirements. Additionally, normal maintenance could be performed on the DFT file. In the latter case, this should only be done by the CICS/VS system administrator at times, where no FTP sessions are running.

Refer to section C.6.3 for a detailed description of how to maintain the DFT.

If changes are performed to the access method for the DFT, care must also be taken to correct the modules in the the transfer part of the FWA. These modules are:

FTPVAL -	file validation, which reads the DFT-entries, whenfiles are validated
FTPCTRL -	transfer control, which reads the DFT-entry for a file to be transferred, just before a transfer is to take place

Before any change is performed, review the following sections:

Section C.2 - Programs and Modules
Section C.3 - File Support and Handling
Section C.5 - System Requirements
Section C.6 - Installation

C.8 Bibliography

SC28-6478-3 IBM DOS/VS COBOL Compiler and Library Programmers Guide

5746-XX3 CICS/DOS/VS
Customer Information Control System
Virtual Storage (CICS/VS)
Version 1, Release 5

Application Programmer's Reference Manual
(Command Level).

GC24-5158 DOS/VS Data Management Concepts

GC33-5382 DOS/VS access Method Services

RETURN LETTER

Title: ITT 3290 File Transfer Protocol and Application Programming Guide, Ref. Manual RCSL No.: 42-i2382

A/S Regnecentralen af 1979/RC Computer A/S maintains a continual effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback, your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability:

_____

_____

_____

_____

Do you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

How can this manual be improved?

_____

_____

_____

_____

Other comments?

_____

_____

_____

_____

_____

Name: _____ Title: _____

Company: _____

Address: _____

Date: _____

Thank you

42-i1288

. . . . . Fold here . . . . .

. . . . . Do not tear - Fold here and staple . . . . .

Affix
postage
here

REGNECENTRALEN
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark