RT–11 Ethernet File Transfer Program (RTEFTP)

User's Guide

and Protocol Specification

Revision/Update information: MG-RTEFTP-TM-002

Operating System and version: RT–11 V5.5 or higher

Software version: Y01.36

Preliminary

Contents

PREFACE vii

DOCUMENT STRUCTURE ix

INTENDED AUDIENCE xi

CHAPTER 1 INTRODUCTION 1–1

1.1 SUPPORTED DEVICES 1–1

1.2 FILE FORMATS 1–1

1.3 RTEFTP INSTALLATION 1–1

CHAPTER 2 RTEFTP COMMAND OVERVIEW 2–1

2.1 CALLING AND TERMINATING RTEFTP 2–1

2.2 COMMAND PROCESSING 2–1

BROADCAST 2–2
CLEAR 2–3
COPY 2–4
DELAY 2–5
DIRECTORY 2–6
EXIT 2–7
HELP 2–8
SET 2–9
SHOW 2–10

2.3 USING RTEFTP 2–11

Preliminary

Contents

CHAPTER 3 RTEFTP INTERNAL DATA STRUCTURES 3–1

3.1 TIMER ID'S 3–1
3.1.1 Node Identification Timer 3–1
3.1.2 Local Node Table Processing Timer 3–2
3.1.3 Packet Retransmit Timer 3–2
3.1.4 Connection Timer 3–2

3.2 FILE AND NETWORK ACCESS CHANNELS 3–2
3.2.1 File Access Channels 3–2
3.2.2 Network Access Channels 3–3
3.2.3 Utility Channel 3–3
3.2.4 RT–11 Overlay Channel 3–3

3.3 RECEIVE BUFFERS 3–3

3.4 VIRTUAL CIRCUITS 3–4
3.4.1 Virtual Circuit Definition Block 3–4

3.5 FINITE STATE MACHINE 3–6
3.5.1 State Definition Block 3–6
3.5.2 Event Definition Block 3–6
3.5.3 State Diagram 3–7

3.6 ACCESS CONTROL LIST 3–7

3.7 NODE DEFINITION BLOCK 3–8

CHAPTER 4 RTEFTP PROTOCOL SPECIFICATION 4–1

4.1 RTEFTP BUFFER FORMAT 4–1
4.1.1 Standard buffer header 4–1
4.1.2 Standard Ethernet header 4–2
4.1.3 RTEFTP Packet header format 4–2

4.2 PACKET FORMATS 4–3

Preliminary

4.2.1 Control packets 4–4
4.2.1.1 Node_Identification packet format • 4–4
4.2.1.2 Broadcast packet format • 4–4
4.2.2 File transfer 4–5
4.2.2.1 Transfer_Connect packets • 4–5
4.2.2.1.1 Transfer_Connect SEND packet format • 4–5
4.2.2.1.2 Transfer_Connect RECEIVE packet format • 4–5
4.2.2.1.3 Transfer_Connect SEND_START packet format • 4–6
4.2.2.2 Transfer_Data packet format • 4–6
4.2.2.2.1 Transfer_Data WRITE packet format • 4–6
4.2.2.2.2 Transfer_Data READ packet format • 4–7
4.2.2.3 Transfer_Disconnect packets • 4–7
4.2.2.3.1 Transfer_Disconnect END packet format • 4–7
4.2.2.3.2 Transfer_Disconnect ABORT packet format • 4–7
4.2.3 Directory 4–8
4.2.3.1 Directory_Connect packet • 4–8
4.2.3.2 Directory_Start packet format • 4–8
4.2.3.3 Directory_Data packet format • 4–8
4.2.3.4 Directory_End packet format • 4–9

4.3 TRANSFER TRANSACTION EXAMPLES 4–9
4.3.1 Sample SEND transaction 4–9
4.3.2 Sample RECEIVE transaction 4–11

CHAPTER 5 INTERNAL OPERATION 5–1

5.1 ENVIRONMENT CHECKS 5–1
5.1.1 Monitor 5–1
5.1.2 Ethernet 5–1
5.1.3 Operating Characteristics 5–1

5.2 ENVIRONMENT PREPARATION 5–1
5.2.1 Queue elements 5–1
5.2.2 Ethernet 5–2
5.2.3 Memory 5–2
5.2.4 Access Control List 5–2
5.2.5 Remote Node Directory List 5–2
5.2.6 Virtual circuit initialization 5–3

5.3 ENABLING OPERATION 5–3
5.3.1 Ethernet 5–3
5.3.2 Timers 5–3

Preliminary

5.3.3 Command processing 5–3

5.4 COMPLETION ROUTINE OPERATION 5–3
5.4.1 Timer Completion routines 5–4
5.4.1.1 Remote Node Directory timer • 5–4
5.4.1.2 Node Identification timer • 5–4

5.5 RECEIVE COMPLETION 5–4
5.5.1 Node Identification packets 5–5

APPENDIX A ERROR MESSAGES A–1

APPENDIX B PACKET CODES B–1

APPENDIX C STATUS CODES C–1

APPENDIX D FILE SYSTEM CODES D–1

APPENDIX E CUSTOMIZATION PATCHES E–1

E.1 AUTOMATIC VBGEXE E–1

E.2 CHANGING NUMBER OF RECEIVE BUFFERS E–1

E.3 CHANGING NUMBER OF VIRTUAL CIRCUITS E–2

E.4 CHANGING MULTICAST ADDRESS E–2

E.5 CHANGING PROTOCOL TYPE E–3

Preliminary

INDEX

TABLES

2–1	Access Rights Options	2–9
4–1	Sample SEND Transaction, Node B to A, Node A initiator	4–10
4–2	Sample RECEIVE Transaction, Node A to B, Node A initiator	4–11
B–1	Packet type codes	B–1
B–2	Control packet modifiers	B–1
B–3	Transfer_Connect packet modifiers	B–1
B–4	Transfer_Data packet modifiers	B–1
B–5	Transfer_Disconnect packet modifiers	B–2
B–6	Directory packet modifiers	B–2
B–7	Packet modifiers	B–2
B–8	Packet codes	B–3
C–1	Status Codes	C–1
D–1	File System Codes	D–1

Preliminary

Preface

Preliminary

Document structure

This manual describes the use of the RT–11 Ethernet File Transfer Program, RTEFTP. In addition, it documents the protocol and the internal structures employed by the program.

- Chapter 1 provides a general description of the RT–11 Ethernet File Transfer Program (RTEFTP)
- Chapter 2 provides a list of the recognized RTEFTP commands and their function
- Chapter 4 documents the RTEFTP protocol
- Chapter 3 documents the program internal data structures
- Chapter 5 documents the internal operation of the program
- Appendix A lists error messages
- Appendix B lists packet codes
- Appendix C lists packet status codes
- Appendix D lists file system codes
- Appendix E describes customization patches

Related documents:

- RT–11 Documentation Set

Preliminary

Intended audience

This manual is intended for RT–11 system users who plan on installing and using RTEFTP on their system and for system programmers who wish to understand the operation of the program and the protocol so that they might implement a version for their favorite operating system.

Preliminary

1 Introduction

The RT–11 Ethernet File Transfer Program (RTEFTP) is a general purpose file-access utility for use with Ethernet. You can use RTEFTP to transfer files between RT–11 systems using Ethernet.

RTEFTP is designed for use under an XM monitor with timer support. It can be run as a background, system, or foreground job. It may also be run under VBGEXE.

RTEFTP employs a transaction based protocol with error checking and timeout recovery. It may be in operation over the Ethernet concurrent with other Ethernet applications.

1.1 Supported devices

All devices which are defined as RT–11 file-structured, but not special directory, are supported. Magtapes are not supported.

1.2 File Formats

RTEFTP provides no conversion of file formats. All files are transferred in image mode, block for block.

1.3 RTEFTP Installation

The RT–11 Ethernet File Transfer Program (RTEFTP) is installed by simply copying RTEFTP.SAV to the system volume.

Apply any desired customization patches listed in Appendix E.

Preliminary

2 RTEFTP Command Overview

2.1 Calling and terminating RTEFTP

To run RTEFTP in the background, issue one of the following commands:

```
.RTEFTP !Program is executed via CCL from SY:
.R RTEFTP !Program is executed via KMON from SY:
```

In order to run RTEFTP as a system or foreground job, it is necessary to have it execute in the VBGEXE context. First ensure that VBGEXE.SAV and RTEFTP.SAV reside on the system volume, then you may issue the following command:

```
.{F|S}RUN SY:VBGEXE.SAV/NAME:RTEFTP/BUFFER:250.
```

To terminate RTEFTP, enter the command EXIT or type <CTRL/C> twice.

2.2 Command processing

RTEFTP employs a least-unambiguous command scanner. Its syntax is modelled after that of RT-11 DCL and DECnet. Commands to RTEFTP consist of a single word followed by one or more optional or required arguments. If a command is incomplete, RTEFTP will prompt for the missing elements.

Preliminary

BROADCAST

The BROADCAST command is used to send a message to one or more participating nodes.

FORMAT

BROADCAST *destination text*

DESTINATION

ALL

Sends the message specified by 'text' to all RTEFTP nodes whether they are enabled for remote access or not.

node_name

Sends the message specified by 'text' to the specified node. The node must be enabled for remote access and show up in the remote node directory listing (see SHOW NODES).

TEXT

Specifies the text to be sent. Quotes are required when the command is specified in one line. If quotes are not included, only the first word of the text will be sent.

Preliminary

CLEAR

The CLEAR command is used to reset or disable certain program information or operating characteristics.

FORMAT

CLEAR *option*

OPTIONS

ACCESS [dev:filnam.typ]

Clears the contents of the access control list, allowing unrestricted access to local devices and files if remote access is enabled.

Optionally, you may remove one or more access control list entries by specifying the appropriate file specification.

BROADCAST

Disables reception of broadcast messages.

LOG

Disables local event logging.

NAME

Clears the local node name, disabling remote access and timebase service.

NODES [node_name]

Clears the local list of nodes known to be accepting remote access. This table is rebuilt as messages are received from nodes which have enabled remote access.

Optionally, you may remove the remote node directory entry for a specific node by specifying its name.

TIME

Disables timebase service. All participating nodes will resume deriving their date and time locally.

Preliminary

COPY

The COPY command initiates a file transfer from the local node to a remote node, or from a remote node to the local node. The COPY command has the form:

FORMAT

COPY[/qualifiers...] *[node_name::]in_spec*
[node_name::]out_spec

QUALIFIERS

/QUEUE
Initiates a transfer and returns to the command prompt, allowing further commands to be issued.

/DEVICE
Indicates a non-file-structured access of a local device. Required if output specification is local.

DESCRIPTION

A node_name must be specified for either the input or output file, but not both. Only nodes which have enabled remote access may be specified. Wildcards are not supported for either input or output.

Since the portion of the command string following the '::' on input or output is passed to the remote node for parsing, it must adhere to any device or file naming rules applicable for that operating system.

When RTEFTP is first started, it has no capability of performing file transfers until it has knowledge of at least one other node on the Ethernet running RTEFTP. As each node is enabled for remote access (see SET NAME), it begins sending out a message which informs all RTEFTP participating nodes of its existence.

File transfers may only be performed with nodes which have had remote access enabled, but it is not required that the local node be enabled for remote access to initiate a file transfer.

The user has full access rights to local devices and filenames while access rights to devices and filenames on a remote node are verified on that node.

Preliminary

DELAY

The DELAY command causes RTEFTP to wait for the specified node to become available for remote access before returning to the command prompt. The DELAY command has the form:

FORMAT

DELAY *node_name*

DESCRIPTION

This command is particularly useful when executed from within an indirect command file, since further file processing (generally COPY commands) will not proceed until the required node is available.

If the specified node does not become available in one minute, the program returns to the command prompt.

Preliminary

DIRECTORY

The DIRECTORY command obtains and displays the directory of a volume mounted on a remote node. The DIRECTORY command has the form:

FORMAT

DIRECTORY *node_name::file_specification[,...]*

DESCRIPTION

Since the portion of the command string following the '::' is passed to the remote node and parsed there, it must adhere to any device and file naming rules applicable on that machine.

For RT–11 nodes, the *file_specification* may consist of up to one device specification and up to 8_{10} file names separated by commas, with full wildcarding allowed. If no device is specified, the default device **DK:** is used.

Preliminary

EXIT

The EXIT command is used to terminate program execution. The EXIT command has the form:

FORMAT

EXIT

DESCRIPTION

If there are any currently active circuits, further remote access is disabled while the user is queried whether the active circuits should be aborted. If not, remote access is re-enabled and the program returns to the command prompt to await further commands.

If remote access has been enabled, a message is sent to inform all participating nodes that the local node is becoming unavailable.

Preliminary

HELP

The HELP command is used to obtain information on the use of RTEFTP commands. The HELP command has the form:

FORMAT

HELP *[topic]*

DESCRIPTION

where *topic* is the name of the command for which help is desired. If not specified, the program will list all commands for which help is available. If the topic specified is *, the program will list help for all commands for which help is available.

Preliminary

SET

The SET command is used to set various operating characteristics of the program. The SET command has the form:

FORMAT

SET *option*

OPTIONS

ACCESS dev:filnam.typ[/qualifier...]

Adds the specified device and file to the access control list, setting access as specified in the options. 'dev' may be any known device or logical on the local node. 'filnam.typ' may contain wildcard characters. 'option' is of one of the following:

Table 2–1 Access Rights Options

Qualifier	Meaning
D	Allows non-file structured access
E	Access is excluded
L	Access by explicit device name, no logical to physical translation performed
R	Read access is allowed, directory access implied
W	Write access is allowed, directory access implied

BROADCAST

Enables reception of broadcast messages.

LOG

Enables event logging. Event logging is useful to track node available /unavailable events as well as remote accesses to devices and files on the local machine. It also reports access violations if an access control list has been established.

NAME node_name

Establishes the name by which the local node is known to participating nodes, enabling remote access.

TIME

Establishes the local node as the timebase for all participating nodes. This service is useful in coordinating the dates and times on a series of machines. A node may be enabled as the timebase only if it is also enabled for remote access, and only one participating node may be enabled as the timebase at a time.

Ensure that your local node has the correct date and time before issuing this command, otherwise all participating nodes will have incorrect dates and times.

Preliminary

SHOW

The SHOW command is used to displays the settings of certain program information. The SHOW command has the form:

FORMAT

SHOW *option*

OPTIONS

ACCESS

Displays the current access rights list.

CHARACTERISTICS

Lists the operating characteristics, including protocol version number, number of receive buffers, number of virtual circuits, and number of active circuits.

CIRCUITS

Lists the status of all virtual circuits, specifying type of connection and including information relative to the connection.

For circuits performing file transfers, lists direction of transfer, file name, file size and current block.

For circuits performing directories, lists device and file specifications from DIRECTORY command.

NODES

Lists information on all nodes known to be enabled for remote access, including node name, Ethernet address and settings of various operating flags.

Preliminary

2.3 Using RTEFTP

The following section illustrates a typical setup and usage of the RTEFTP Ethernet file transfer program. In the example, RTEFTP.SAV resides on the system volume and will run in the background. User entered text is italicized.

Once RTEFTP has been installed on the system volume, it may be loaded and execution begun by issuing the following command at the KMON prompt:

.RTEFTP

RTEFTP responds with 'RTEFTP>' at the left margin to indicate its readiness to accept commands.

RTEFTP> *SET LOG* RET

The user will now be informed as nodes become available for remote access or become unavailable.

RTEFTP> *SET ACCESS SY:/R/L* RET

An access control list is established, restricting remote access to this machine to read access of all files on logical device SY:. Access to this device must be explicit, no device translation will occur.

RTEFTP> *SET NAME BOSKON* RET

The local node name is established as BOSKON, enabling remote access. All participating nodes will note the fact that BOSKON is now available for remote access.

RTEFTP> *COPY A.B HELMTH::SY:A.B* RET

Causes the local file DK:A.B to be sent to node HELMTH and placed on SY:A.B.

RTEFTP> *COPY/QUEUE BREADB::DU1:C.D C.D* RET

RTEFTP>

Causes the remote file DU1:C.D on node BREADB to be transferred to the local file DK:C.D, but the program returns immediately to the command prompt signifying its readiness to accept further commands. The user will be informed when the transfer completes.

Preliminary

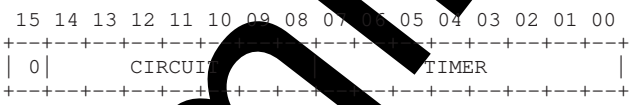
3 RTEFTP Internal Data Structures

The following chapter describes the internal data structures used in the RTEFTP file transfer program. A description of the protocol can be found in Chapter 4.

3.1 Timer ID's

The timer ID's were designed to be a unique number which encoded the virtual circuit number as well as the timer type. Since issuing a 'cancel mark-time' request with an ID of 0 cancels all outstanding timer requests, the lowest timer number defined is 1. This ensures that we never attempt to cancel timers using an ID of 0 (this could occur only in processing circuit 0 timers).

The format of timer ID's is as follows:



<15> must be zero
CIRCUIT circuit number
TIMER timer number

The architectural limit for the circuit number is 177_8 (127_{10}), the practical limit under RT-11 is 7. This limit is used to avoid excessive use of low memory through use of the .CDFN programmed request.

The architectural limit for the timer number is 377_8 (255_{10}).

3.1.1 Node Identification Timer

Timer 1 is the five-second node identification timer. When remote access has been enabled (via SET NAME) and this timer expires, a *Node_ Identification* packet is sent to the RTEFTP multicast address, notifying all participating nodes. The timer is then reposted.

Timer 1 is processed only by virtual circuit 0.

Preliminary

3.1.2 Local Node Table Processing Timer

Timer 2 is the one-second remote node directory timer. When it expires, the remote node directory is scanned and any entry which has timed-out is removed and the node it defined is reported as unavailable. The timer is then reposted.

Timer 2 is processed only by virtual circuit 0.

3.1.3 Packet Retransmit Timer

Timer 3 is the two-second packet retransmit timer. If a response to a packet is not received by the time this timer expires, the last packet transmitted for the indicated virtual circuit is retransmitted and the timer is reposted.

Timer 3 is processed by all virtual circuits.

3.1.4 Connection Timer

Timer 4 is the ten-second connection timer. If a response to a packet is not received by the time this timer expires, the remote node involved in a transfer on the indicated virtual circuit is assumed be have become unavailable and the connection is terminated.

Timer 4 is processed by all virtual circuits.

3.2 File and Network access channels

The 16_{10} channels provided by RT–11 are divided by use into file access, network access and temporary-use or utility channels.

3.2.1 File Access Channels

Eight channels ($0\text{-}7_8$) are available for file access. The virtual circuit number maps directly to its corresponding file access channel.

The file access channel number can be extracted from a timer ID or from within a virtual circuit definition block (see Section 3.4).

Preliminary

3.2.2 Network Access Channels

Two channels (14_8 and 15_8) are used for network access, one for synchronous traffic and one for asynchronous traffic. Two channels are used in order to separate synchronous and asynchronous requests.

Under RT–11, synchronous requests night never complete on a channel for which asynchronous requests were outstanding.

3.2.3 Utility Channel

One channel (16_8) is reserved for use by support routines so long as it is available for use when the routine completes.

An example of this use is the routine CVLFOP which converts logical device names to physical device names. This routine uses .CSTAT, which requires a channel to be opened to the specified device.

3.2.4 RT–11 Overlay Channel

One channel (17_8) is reserved for use by RT–11 and unused by RTEFTP at the present time.

3.3 Receive buffers

There is a table of pointers to the allocated receive buffers, a byte which indicates how many buffers are available and a byte which is used as an index into the table for retrieving the pointer to the next receive buffer to be processed, updated as each receive buffer is processed.

The program is built to handle a maximum of eight asynchronous receive buffers, but only two are built into the image. Additional buffers are dynamically allocated from available memory at run-time. Receive buffers are of the form described in Section 4.1.

A customization patch (see Section E.2) allows you to specify the maximum number of receive buffers for which RTEFTP will attempt to reserved space.

Receive buffer sequencing is performed via completion.

Preliminary

3.4 Virtual circuits

There is a table of pointers to the allocated virtual circuit definition blocks and a byte which indicates the number of available virtual circuits. The contents of a byte in a receive buffer is used as an index into the table for retrieving the pointer to the virtual circuit for which the buffer is to be procesed.

The program is built to handle a maximum of eight virtual circuits, but only two are built into the image. Additional circuits are dynamically allocated from available memory at run-time. One transmit buffer, of the form described in Section 4.1, is defined within each virtual circuit.

A customization patch (see Section E.3) allows you to specify the maximum number of virtual circuits RTEFTP will attempt to make available.

3.4.1 Virtual Circuit Definition Block

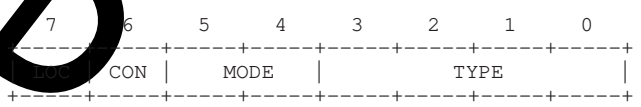
The structure of a virtual circuit definition block is as follows:

AFG (1) : B

Virtual circuit allocation flag. -1 if circuit is free. Allocation of a circuit should be done using an atomic operation (such as an INCB) in order to avoid a race between circuit allocation from mainline code and circuit allocation from completion code.

ORG (1) : BM

the set of flags indicating the type of connection.



LOC	indicates a locally-initiated transfer
CON	indicates a two-way connection has been established
MODE	indicates connection modifiers, from initial connect packet (refer to Appendix B)
TYPE	indicates connection type, from initial connect packet (refer to Appendix B)

STA (2) : B

a pointer to the current state definition block in the finite state machine.

Preliminary

CFG (2) : B|BM

the circuit flag word. Contents are routine-specific.

LCN (1) : B

the local circuit number, maps directly to the file access channel number.

RCN (1) : B

the remote circuit number, used to verify unique circuit allocation when establishing a connection for an incoming request.

PAD (6) : B

the Ethernet address of the remote node. Used for packet validation.

DYN (2) : B

a word pointing to a region of dynamically-allocated memory used to store operation-specific data for the circuit. Zero indicates none.

HAN (2) : B

a word pointing to a region of dynamically-allocated memory used to fetch a handler for use by a circuit. Zero indicates none.

DBK (8) : B

the four words containing the file specification in Radix–50.

FSZ (2) : B

the size of the file being accessed. Zero until known.

BLK (2) : B

the number of the block being accessed during a file transfer. The record number during a directory operation.

XSZ (2) : B

the size of the packet contained in the transmit buffer, used for packet retransmit.

XBF (BF.SIZ) : B

the asynchronous transmit buffer. Same size and format as a receive buffer as described in Section 4.1.

DBS (2) : B

the status reported following the posting of the last I/O request for the file access channel associated with this circuit. The MSB indicates an error occurred, the low byte reflects the contents of byte 52 following the error.

Preliminary

DBF (512) : B

the one-block read-ahead or write-behind buffer used in double-buffering operations.

3.5 Finite State Machine

The finite state machine operates through a series of states. Each state consists of a series of event definitions, each specifying an event that might occur in that state, the name of the routine to execute when the event occurs, and a specification for the next state to be entered.

3.5.1 State Definition Block

The structure of a state definition block is as follows:

3.5.2 Event Definition Block

The structure of an event definition block is as follows:

EVENT (2) : B

a word containing either the complete timer ID as described in section 3.1 (for timer completion routines), or the expected packet code as described in Appendix B (for receiver completion routines) in the low-order byte. Zero indicates the end of the current state definition block.

PROCESS (2) : B

a word containing the address of the routine to be used to process the event.

NEW STATE (2) : B

a word containing the address of the state definition block to which the finite state machine will transition following processing of this event. Contains zero if no transition is to occur, the state machine remains in the same state.

Preliminary

3.5.3 State Diagram

tbs

3.6 Access Control List

The access control list specifies the devices, files and types of remote access which are allowed or disallowed by the local node. It consists of a series of entries, with each entry defining a device, files on that device and type of access allowed or disallowed for the specified files.

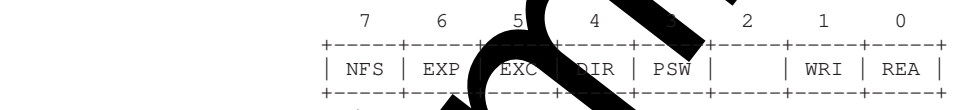
An access control list entry has the following format:

LNK (2) : B

a pointer to the next access control list entry (zero if none)

FLG (1) : BM

the set of flags indicating the access type for the device and file specified. Zero for a free entry.



NFS	indicates non-file-structured access is allowed
EXP	indicates explicit device access allowed (reference to device must be by use of specified name)
EXC	indicates access is not allowed
DIR	indicates directory may be obtained for this device. Enabled if read access is enabled.
PSW	indicates password is required for access (future)
WRI	indicates write access is allowed
REA	indicates read access is allowed

res (1) : B

undefined and reserved.

DEV (2) : B

a word containing the Radix–50 representation of a three-character device name.

FIL (4) : B

two words containing the Radix–50 representation of a six-character file name.

Preliminary

TYP (2) : B

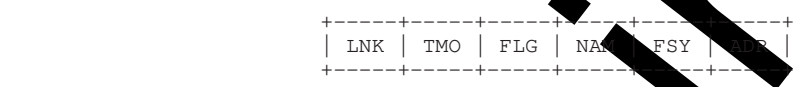
a word containing the Radix–50 representation of a three-character file type.

PSW (4) : B (future)

two words containing the Radix–50 representation of a six-character password, possibly encoded, used for access verification.

3.7 Node Definition Block

The Node Definition Block is used to store information relating to a single node which has enabled remote access. A linked list of Node Definition Blocks provides the remote node directory information reported via the SHOW NODES command. It is also used when the local node has enabled remote access (see Section 4.2.1.1).



LNK (2) : B

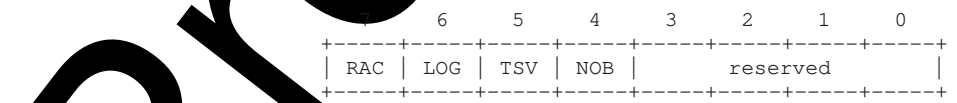
a word containing the address of the next node definition block. In *Node_Identification* packets, this word is unused and reserved. Zero indicates end of list.

TMO (1) : B

the timeout value in seconds. If another *Node_Identification* packet is not received from the node defined by this structure in the specified number of seconds, the node is to be assumed to be unavailable and its entry should be removed from the remote node directory list.

FLG (1) : BM

the node service flags, as follows:



RAC	indicates node accepts remote access
LOG	indicates node is logging transactions
TSV	indicates node provides timebase service
NOB	indicates node does not want to receive broadcast messages

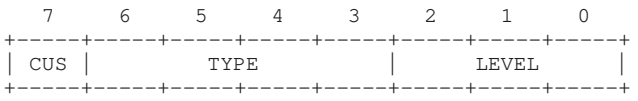
NAM (7) : B

six byte field containing the node name in ASCII, padded with spaces to fill 6 bytes, and a byte containing <200>.

Preliminary

FSY (1) : BM

indicates the file system in use on the node, defined as follows:



CUS	indicates customer/CSS file system
TYPE	indicates file system type (refer to Appendix D)
LEVEL	indicates file system level

ADR (6) : B

the six bytes containing the Ethernet address of the node.

Preliminary

4 RTEFTP Protocol specification

4.1 RTEFTP Buffer format

An RTEFTP buffer consists of four parts: The header required by RT-11 for Ethernet access, the header required by Ethernet for all frames, the RTEFTP packet header and the RTEFTP packet.

The four parts are concatenated into one contiguous section in memory, with the format as follows:

```
+-----------------------+
|  RT-11 buffer header  |
+-----------------------+
| Ethernet frame header |
+-----------------------+
| RTEFTP packet header  |
+-----------------------+
|     RTEFTP packet     |
+-----------------------+
```

4.1.1 Standard buffer header

The standard RT-11 Ethernet buffer header is as follows:

```
+--------------+--------------+--------+
| MAJOR_STATUS | MINOR_STATUS | LENGTH |
+--------------+--------------+--------+
```

MAJOR_STATUS (1) : B

a byte which contains the major status returned from an Ethernet request. Refer to the RT–11 documentation.

MINOR_STATUS (1) : B

a byte which contains the minor status returned from an Ethernet request. Refer to the RT–11 documentation.

LENGTH (2) : B

a word which indicates the actual size, in bytes, of a received frame. Undefined for transmit requests.

Preliminary

4.1.2 Standard Ethernet header

The format of the standard Ethernet header is as follows:

```
+-----+-----+----+
| DST | SRC | PT |
+-----+-----+----+
```

DST (6) : B

the 6 bytes which specify the destination Ethernet address. For *Node_Identification* and *Broadcast* control packets the destination is the multicast address, Hex AB-00-04-00-60-06. All other packets use Ethernet physical addresses.

A customization patch (see Section E.4) allows you to specify the multicast address to be used.

SRC (6) : B

the 6 bytes which specify the source Ethernet address. For transmit, normally specified as Hex 00-00-00-00-00-00, which causes the local Ethernet physical address to be used. On receive, indicates the physical address of the node which sent the packet.

PT (2) : B

the 2 bytes which specify the Ethernet protocol type. Hex 60-06 for RTEFTP protocol.

A customization patch (see Section E.5) allows you to specify the protocol type to be used.

4.1.3 RTEFTP Packet header format

The format of an RTEFTP packet header is as follows:

```
+-------------+----------+-----+-----+
| CODE STATUS | PROTOCOL | ICN | OCN |
+-------------+----------+-----+-----+
```

CODE (1) : BM

the set of flags to indicate the type of RTEFTP packet.

```
  7     6     5     4     3     2     1     0
+-----+-----+-----+-----+-----+-----+-----+-----+
| NAK | RES | MODIFIERS |         TYPE          |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

NAK	indicates a negative acknowledgement (only valid for a response packet)
RES	indicates a response packet
MODIFIERS	indicates the packet type modifiers (refer to Appendix B)
TYPE	indicates the packet type (refer to Appendix B)

All combinations of NAK and RES bits are valid with the exception of NAK set and RES reset. This pattern is invalid and reserved.

Preliminary

STATUS (1) : B

indicates status returned with NAK responses. (refer to Appendix C)

PROTOCOL (2) : B

indicates the RTEFTP protocol version information in Radix–50. It is a three character string of the form 'x.y' where 'x' is the major protocol version number and 'y' is the minor protocol version number.

For the current release, this word contains Radix–50 '1.3'.

ICN (1) : B

indicates the incoming virtual circuit number (0-127$_{10}$). Used to indicate which virtual circuit is to process this packet. A negative value is used by the node initiating the transfer in order to request allocation of a virtual circuit.

It should be noted that in a packet to be transmitted, this byte contains the number of the virtual circuit on the partner's node which is to receive the packet. In a received packet, this byte contains the number of the virtual circuit on the local node to which this packet should be directed.

OCN (1) : B

indicates the outgoing virtual circuit number (0-127$_{10}$). Used to indicate the number of the virtual circuit to which responses are to be sent.

It should be noted that in a packet to be transmitted, this byte contains the number of the virtual circuit on the local machine which is sending the packet. In a received packet, this byte contains the number of the virtual circuit on the remote node from which this packet was sent.

4.2 Packet formats

In each of the following sections, only the contents of the packet beyond the packet header are described. The contents of the code byte in the packet header is the logical OR of the **Type** and **Modifiers** specified.

Packet turn-around is accomplished by moving the source address field in the received packet to the destination address field, zeroing the source address field (which causes the station physical address to be used), swapping the incoming and outgoing circuit number fields, and ACKing (setting the RES bit) or NAKing (setting NAK and RES) the packet code. The frame may then be transmitted. The minimum Ethernet frame size is always used for responses as they are always shorter than the minimum.

Preliminary

4.2.1 Control packets

Control packets are used to distribute information to or solicit information from participating nodes on the network. There is no response expected to control packets.

4.2.1.1 Node_Identification packet format

The *Node_Identification* control packet is used to identify a node which has enabled remote access to participating nodes on the network.

The format of the *Node_Identification* packet is as follows:

```
(Type = 0, Modifiers = 00)
+-----------+------+------+-----------+-----------+-----------+
| NODE_INFO | DATE | TIME | LIST_SIZE | RESERVED | NODE_LIST |
+-----------+------+------+-----------+-----------+-----------+
```

NODE_INFO (nnn) : B

contains the node definition block for the node sending the *Node_ Identification* packet, the format of which is defined in Section 3.7

DATE (2) : B

the current date, in RT–11 format. Refer to the RT–11 documentation.

TIME (4) : B

the current time, in RT–11 ticks-past-midnight format. Refer to the RT–11 documentation.

4.2.1.2 Broadcast packet format

The *Broadcast* packet is used to send messages to one or all nodes on the network.

The format of the *Broadcast* packet is as follows:

```
(Type = 0, Modifiers = 20₈)
+-----+-----+
| TLN | TBF |
+-----+-----+
```

TLN (1) : B

a byte which contains the length ($0\text{-}255_{10}$) of the broadcast message text in bytes not including the trailing null byte.

TBF (255) : B

a NUL-terminated ASCII string comprising the characters of the broadcast message.

Preliminary

4.2.2 File transfer

The following packets are used to accomplish a file transfer between nodes on the network. The *type* and *modifiers* specified are documented in Appendix B.

4.2.2.1 Transfer_Connect packets

The *Transfer_Connect* class of packets are used to establish and synchronize a connection between two nodes for the purpose of transferring a file.

4.2.2.1.1 Transfer_Connect SEND packet format

The *Transfer_Connect SEND* packet is used to establish a connection between the local node and a node allowing remote access for the purpose of transferring a file from the remote node to the local node. The packet indicates to the remote node that it should prepare to send a file, the name of which is specified in the packet.

The format of a *Transfer_Connect SEND* packet is as follows:

```
(Type = 1, Modifiers =  0)
+-----+-----+
| FSZ | SPC |
+-----+-----+
```

FSZ (4) : B

significant only in a response, these two words indicate the size, in 512-byte blocks, of the file to be transferred. Low-order size first, high-order second).

SPC (100) : B

a NUL-terminated ASCII string specifying the name of the file to be transferred.

4.2.2.1.2 Transfer_Connect RECEIVE packet format

The *Transfer_Connect RECEIVE* packet is used to establish a connection between the local node and a node allowing remote access for the purpose of transferring a file from the local node to the remote node. The packet indicates to the remote node that it should prepare to receive a file, the name and size of which are specified in the packet.

The format of a *Transfer_Connect RECEIVE* packet is as follows:

```
(Type = 1, Modifiers = 20₈)
+-----+-----+
| FSZ | SPC |
+-----+-----+
```

Preliminary

FSZ (4) : B

two words indicating the size, in 512-byte blocks, of the file to be transferred. Low-order size first, high-order size second.

SPC (100) : B

a NUL-terminated ASCII string specifying the name of the file to be transferred.

4.2.2.1.3 Transfer_Connect SEND_START packet format

The *Transfer_Connect SEND_START* packet is used only in a locally-initiated file transfer from the remote node to the local node. It signals to the remote node that the local node is prepared to receive the file. There is no specific response to this packet other than the first data block from the remote node.

The format of the *Transfer_Connect SEND_START* packet is as follows:

(Type = 1, Modifiers = 40_8)

4.2.2.2 Transfer_Data packet format

The *Transfer_Data* class of packets is used to transfer a 512-byte block of data between nodes participating in a file transfer.

4.2.2.2.1 Transfer_Data WRITE packet format

The *Transfer_Data WRITE* packet is used to transfer a 512-byte block of data to be written on the receiving node.

The format of the *Transfer_Data WRITE* packet is as follows:

BLK (4) : B

two words indicating the block number of the transfer. Low-order first, high-order second.

DATA (512) : B

data buffer containing 512 bytes of data to be transferred.

Preliminary

4.2.2.2.2 Transfer_Data READ packet format

The *Transfer_Data READ* packet is used to request a 512-byte block of data from a remote node. The response contains the requested block of data.

The format of the *Transfer_Data READ* packet is as follows:

```
(Type = 2, Modifiers = 20₈)
+-----+---~---+
| BLK | DATA |
+-----+---~---+
```

BLK (4) : B

two words indicating the desired block number. Low-order first, high-order second.

DATA (512) : B

significant only in response, contains one 512-byte block of data.

4.2.2.3 Transfer_Disconnect packets

The *Transfer_Disconnect* class of packets is used to signal the end of a file transfer, whether the transfer completed successfully or not.

4.2.2.3.1 Transfer_Disconnect END packet format

The *Transfer_Disconnect END* packet is used to indicate the successful completion of a file transfer.

The format of a *Transfer_Disconnect END* packet is as follows:

```
(Type = 3, Modifiers =  0)
```

A packet containing this code should only be sent by the node which is the sender in a file transfer or directory operation.

4.2.2.3.2 Transfer_Disconnect ABORT packet format

The *Transfer_Disconnect ABORT* packet is used to indicate that an error occurred during the file transfer.

The format of the *Transfer_Disconnect ABORT* packet is as follows:

```
(Type = 3, Modifiers = 20₈)
```

A packet containing this code can be sent by either node involved in a file transfer or directory operation.

Preliminary

4.2.3 Directory

The following sections define the packets used to accomplish a remote device directory.

4.2.3.1 Directory_Connect packet

The *Directory_Connect* packet is used to establish a connection between the local node and a node allowing remote access for the purposes of obtaining a directory.

The format of a *Directory_Connect* packet is as follows:

```
(Type = 4, Modifiers = 00)
+------+-----+
| RSVD | SPC |
+------+-----+
```

RSVD (4) : B

two reserved words

SPC (100) : B

a NUL-terminated ASCII string specifying the device and files for which a directory is desired.

4.2.3.2 Directory_Start packet format

The *Directory_Start* packet is used to signal the remote node that it may begin sending directory data packets.

The format of the *Directory_Start* packet is as follows:

```
(Code = 4, Modifiers = 20₈)
```

4.2.3.3 Directory_Data packet format

The *Directory_Data* packet is used to send a line of directory text from the remote node to the local node.

The format of the *Directory_Data* packet is as follows:

```
(Type = 4, Modifiers = 40₈)
+-------+-----+------+
| BLOCK | LEN | LINE |
+-------+-----+------+
```

BLOCK (4) : B

two words containing the size in 512 byte blocks, of the file described by this packet.

LEN (1) : B

the count of characters in LINE (not including trailing <200> byte)

Preliminary

LINE (255) : B

a line of directory information, terminated with a <200>

4.2.3.4 Directory_End packet format

The *Directory_End* packet is used by the remote node to signal the end of the directory.

The format of the *Directory_End* packet is as follows:

(Type = 4, Modifiers = 60_8)

4.3 Transfer transaction examples

The examples in this section illustrate the packet interchange required to accomplish some operation.

4.3.1 Sample SEND transaction

The following example illustrates the interchange of RTEFTP packets required to accomplish a file transfer from Node B to Node A, requested by Node A.

Preliminary

Table 4–1 Sample SEND Transaction, Node B to A, Node A initiator

Node A	Packet Code	Node B
❶		
Transfer_Connect SEND (file specification)	[001] ->	
	<- [301]	Transfer_Connect SEND NAK (reason)
	<- [101]	Transfer_Connect SEND ACK (file size)
Transfer_Disconnect ABORT	[023] ->	
	<- [123]	Transfer_Disconnect ABORT ACK
	<- [123]	Transfer_Disconnect ABORT ACK
	<- [123]	Transfer_Disconnect ABORT ACK
	<- [123]	Transfer_Disconnect ABORT ACK
	<- [123]	Transfer_Disconnect ABORT ACK
Transfer_Connect SEND_START	[041] ->	
❷		
	<- [002]	Transfer_Data (block number, block)
Transfer_Data ACK	[102] ->	
Transfer_Data NAK	[302] ->	
❸		
	<- [003]	Transfer_Disconnect END
Transfer_Disconnect END ACK	[103] ->	
Transfer_Disconnect END ACK	[103] ->	
Transfer_Disconnect END ACK	[103] ->	
Transfer_Disconnect END ACK	[103] ->	
Transfer_Disconnect END ACK	[103] ->	
	<- [023]	Transfer_Disconnect ABORT
Transfer_Disconnect ABORT ACK	[123] ->	
Transfer_Disconnect ABORT ACK	[123] ->	
Transfer_Disconnect ABORT ACK	[123] ->	
Transfer_Disconnect ABORT ACK	[123] ->	
Transfer_Disconnect ABORT ACK	[123] ->	

❶ A circuit is established between Node A (the local node) and Node B (the remote node). The file specification is passed, the file size is returned, local temporary file is created and a request is made for the remote node to begin sending the file.

❷ The blocks comprising the file are sent in sequential order.

❸ The transfer completes (through success or failure) and the circuit is taken down.

Preliminary

4.3.2 Sample RECEIVE transaction

The following example illustrates the interchange of RTEFTP packets required to accomplish a file transfer from Node A to Node B, requested by Node A.

Table 4–2 Sample RECEIVE Transaction, Node A to B, Node A initiator

Node A	Packet Code	Node B
❶		
Transfer_Connect RECEIVE (filespec, filesize)	[021] ->	
	<- [321]	Transfer_Connect RECEIVE NAK (reason)
	<- [121]	Transfer_Connect RECEIVE ACK
❷		
Transfer_Data (block number, block)	[002] ->	
	<- [302]	Transfer_Data NAK
	<- [102]	Transfer_Data ACK
❸		
Transfer_Disconnect END	[003] ->	
	<- [103]	Transfer_Disconnect END ACK
	<- [103]	Transfer_Disconnect END ACK
	<- [103]	Transfer_Disconnect END ACK
	<- [103]	Transfer_Disconnect END ACK
	<- [103]	Transfer_Disconnect END ACK
Transfer_Disconnect ABORT	[023] ->	
	<- [123]	Transfer_Disconnect ABORT ACK
	<- [123]	Transfer_Disconnect ABORT ACK
	<- [123]	Transfer_Disconnect ABORT ACK
	<- [123]	Transfer_Disconnect ABORT ACK
	<- [123]	Transfer_Disconnect ABORT ACK

❶ A circuit is established between Node A (the local node) and Node B (the remote node). The file specification and size are passed, the remote temporary file is created and readiness to accept the file is signalled.

❷ The blocks comprising the file are sent in sequential order

❸ The transfer completes (through success or failure) and the circuit is taken down.

Preliminary

5 Internal operation

This chapter describes the internal operation of the RT–11 version of the program.

5.1 Environment checks

5.1.1 Monitor

The program verifies that it is running under a monitor with the required support available. RTEFTP runs on a V5.5 or greater XM monitor which has timer support.

5.1.2 Ethernet

Verifies that the required Ethernet handler is both present and loaded. On CTI-based systems the NC handler is required. On Qbus-based systems the NQ handler is required, and on Unibus-based systems the NU handler is required.

5.1.3 Operating Characteristics

This portion of the program validates those operating characteristics which may be customized by the user such as multicast address, protocol type, number of receive buffers, number of virtual circuits, number of access control list and node directory entries.

5.2 Environment preparation

5.2.1 Queue elements

The program establishes a list of additional queue elements, ensuring that they reside beyond the program as loaded, and outside PAR1 space. If this fails, there is either insufficient memory for the program to be run in the background, or insufficient pool memory for the program to be run under VBGEXE.

Preliminary

5.2.2 Ethernet

The program establishes a channel to the Ethernet handler, allocates a unit of the Ethernet handler and enables the multicast address and protocol type used for RTEFTP message traffic. It then establishes a second channel to provide separation of asynchronous and synchronous requests. If any part of this step fails, it could be due to the operation of other programs which access the Ethernet.

The program then prepares all pre-built packets using the multicast address and protocol type.

5.2.3 Memory

The program determines the amount of memory available for use, then reserves space for the maximum number of access control list entries (a dynamically created and maintained linked list), the maximum number of remote node directory entries and the maximum amount of dynamically-allocated memory required for all operations on all circuits.

Remaining memory is allocated first for additional receive buffers for increased performance and then for additional virtual circuits for better remote accessability. For each additional virtual circuit, additional memory is reserved to ensure the availability of dynamic memory availability sufficient for all possible operations.

The data structure which defines free memory is then initialized and the reserved memory is placed on the free list for access by the memory allocation and deallocation routines.

5.2.4 Access Control List

The data structure which provides access to the access control list is initialized with an empty list.

5.2.5 Remote Node Directory List

The data structure which provides access to the remote node directory list is initialized with an empty list.

Preliminary

5.2.6 Virtual circuit initialization

Each virtual circuit is initialized by marking it as available, setting its finite state machine state (Host Wait), setting its circuit number and building pointers to its internal transmit buffer.

5.3 Enabling Operation

5.3.1 Ethernet

The program cycles through all available receive buffers, posting them to the Ethernet handler. Receives which complete will be processed through the centralized receiver completion routine.

5.3.2 Timers

The program establishes a five-second timer for use in sending the *Node_Identification* message when remote access has been enabled. This timer is processed through the centralized timer completion routine.

The program also establishes a one-second timer for use in timing-out entries in the remote node directory. This timer is processed through the centralized timer completion routine.

5.3.3 Command processing

The program then enters a command loop which gathers and processes commands from the user. With the exception of request initiation, all Ethernet operations occur through completion routines.

5.4 Completion routine operation

All Ethernet operations, with the exception of the initial send or receive command or directory command, are performed in completion routines.

Dispatching of both timer and receive completion routines is performed through the use of two separate routines which implement a finite state machine. One completion routine dispatches timer completions while the other dispatches receive completions.

Both routines use the same state machine tables.

Preliminary

5.4.1 Timer Completion routines

All timer requests specify the same routine as the completion routine. The dispatching is done based on the timer ID.

5.4.1.1 Remote Node Directory timer

When the one-second timer expires, this routine scans the remote node list, decrementing the timeout value for each entry. If the timeout value for an entry goes negative, the entry is removed from the list and the memory returned to the free list. If logging has been enabled, a message indicating that the node is no longer available is printed.

When the list has been processed, the routine re-posts a one second timer.

5.4.1.2 Node Identification timer

When the five-second timer expires, this routine sends a *Node_ Identification* message if a node name has been established for this node.

Whether a node identification message is sent or not, the routine re-posts a five second timer before returning.

5.5 Receive completion

The first step in processing a receiver completion is to determine the buffer which contains the message just received. This information is contained in a byte which is an index into the table of pointers to receive buffers. When the end of the table is reached, the index is reset to zero.

Once we know which buffer contains the packet, we perform further dispatching based on the contents of the packet code byte.

Once the program has identified the recieve buffer which contains the packet, the packet is validated by checking the protocol version. If it is not correct, the packet is ingored and the buffer is re-posted for receive.

Once the packet has passed the previous checks, further dispatching is performed based on the contents of the packet code byte. The table corresponding to the current state is scanned for an entry which specifies what to do with a packet of the received type. If there is no entry, the packet is ignored and the buffer is re-posted for receive. If there is an entry, the state machine for the virtual circuit is transitioned to the specified state and the routine to process the packet is called.

When the routine has completed, it returns to the centralized completion routine which re-posts the buffer for receive and returns.

Preliminary

5.5.1 Node Identification packets

When a node identification message is received, the current remote node list is scanned to see if it has already been entered. If not, space for a new entry is allocated from free memory, the information is copied into the new list entry and it is added to the remote node list. If logging is enabled, a message is printed to report that the node is available for remote access.

If the identification message is for a node for which there already seems to be an entry, the information is checked to see if it is still current. If the node name has changed and logging is enabled, a message is printed to report the name change.

If the Ethernet address for the node is different from the address stored in the entry, the packet is ignored. This is so that attempts by a node to usurp the name already in use by another node are not honored. If logging is enabled, a message is printed to report the attempt to use a name which is already in use.

Preliminary

A Error messages

?RTEFTP-I-Adjacent node available - *node_name*

Explanation: The reported node has become available for remote access. This message is reported only when event logging has been enabled.

?RTEFTP-I-Adjacent node unavailable - *node_name*

Explanation: The reported node is no longer available for remote access. This message is reported only when event logging has been enabled.

?RTEFTP-I-Beginning directory of *dev:filnam.typ* for *node_name*

Explanation: The local node is in the process of performing a directory for a remote node. This message is reported only when event logging has been enabled.

?RTEFTP-I-Beginning receive of *dev:filnam.typ* from *node_name*

Explanation: The local node is in the process of receiving a file from a remote node. This message is reported only when event logging has been enabled.

?RTEFTP-I-Beginning send of *dev:filnam.typ* to *node_name*

Explanation: The local node is in the process of sending a file to a remote node. This message is reported only when event logging has been enabled.

?RTEFTP-I-Directory completed, *dev:filnam.typ*

Explanation: A directory operation has completed successfully. This message is reported only when event logging has been enabled.

?RTEFTP-I-Node *old_name* changing name to *new_name*

Explanation: Indicates that a remote node has changed its name.

?RTEFTP-I-Insufficient resources to queue

Explanation: A COPY command was issued using the /QUEUE qualifier but no virtual circuits were available for use in the queueing operation. The command proceeds by using the primary virtual circuit, no further commands may be entered until the COPY completes.

?RTEFTP-I-Node name truncated to six characters

Explanation: A node_name longer than six characters was specified to the SET NAME command. The name has been truncated to six characters and remote access has been enabled under that name.

Preliminary

?RTEFTP-I-Transfer aborted - *dev:filname.typ*

Explanation: A file transfer has failed. This message is reported if the transfer was locally-initiated and queued or if it was a remote access and event logging has been enabled.

?RTEFTP-I-Transfer completed - *dev:filnam.typ*

Explanation: A file transfer has completed successfully. This message is reported if the transfer was locally-initiated and queued or if it was a remote access and event logging has been enabled.

?RTEFTP-W-Connection timed-out to node, transfer aborted - *dev:filname.typ*

Explanation: The connection to a remote node in a file transfer timed-out because the remote node is no longer responding. This message is reported only when event logging has been enabled.

?RTEFTP-W-Local node table is filled

Explanation: A node has become available for remote access but the local node table is filled.

?RTEFTP-W-Read access rejected to node *node_name* for *dev:filnam.typ*

Explanation: The local node has rejected a remote access because it violated the access rights list which has been estalished for this node. This message is reported only when event logging has been enabled.

?RTEFTP-W-Write access rejected to node *node_name* for *dev:filnam.typ*

Explanation: The local node has rejected a remote access because it violated the access rights list which has been established for this node. This message is reported only when event logging has been enabled.

?RTEFTP-W-Timed-out waiting for response from node *node_name*

Explanation: The remote access attempt has been terminated because of lack of response from the remote node.

?RTEFTP-F-Node name already in use

Explanation: The node_name specified in a SET NAME command was found to be already enabled for remote access.

User Action: Check for a typing error. Verify that the specified node_name is correct and unique and retry the operation.

?RTEFTP-F-Ambiguous command *xxxxxx*

Explanation: The command issued was found to be an abbreviation for more than one recognized RTEFTP command.

User Action: Check for a typing error. Verify that the format and syntax are correct and unambiguous and retry the operation.

Preliminary

?RTEFTP-F-Ambiguous option *xxxxxx*

Explanation: The command issued was found to contain an abbreviation which matched more than one option for the command issued.

User Action: Check for a typing error. Verify that the format and syntax are correct and unambiguous and retry the operation.

?RTEFTP-F-Ambiguous topic *xxxxxx*

Explanation: The command issued was found to contain an abbreviation which matched more than one topic for which HELP is available.

User Action: Check for a typing error. Verify that the topic is valid and unambiguous and retry the operation.

?RTEFTP-F-Command too complex

Explanation: The command issued to RTEFTP was too complex to be parsed because it contained more than 5 tokens.

User Action: Specify a command which uses fewer tokens.

?RTEFTP-F-Connection rejected, Access denied

Explanation: The remote node rejected the remote access because it violated the access rights list established for that node.

?RTEFTP-F-Connection rejected, Device unavailable

Explanation: The remote node rejected the remote access because the specified device was not available for use.

?RTEFTP-F-Connection rejected, File not found

Explanation: The remote node rejected the remote access because the specified file could not be found.

User Action: Check for a typing error. Verify that the file exists and retry the operation.

?RTEFTP-F-Connection rejected, Insufficient resources

Explanation: The remote node rejected the remote access because it did not have any virtual circuits available. This can occur if the remote node is processing all the remote accesses it can.

User Action: Wait and retry the operation.

?RTEFTP-F-Connection rejected, Insufficient space available

Explanation: The remote node rejected the remote access because there was insufficient space to create the output file.

?RTEFTP-F-Connection rejected, Invalid file specification

Explanation: The remote node rejected the remote access because the file specified was invalid.

User Action: Check for a typing error. Verify that the file specification is in a proper syntax for the remote node and retry the operation.

Preliminary

?RTEFTP-F-Connection rejected, Node is not accepting connections

Explanation: The remote node specified is not accepting any connections for remote access. The node may be in the process of shutting down.

?RTEFTP-F-Connection rejected, Non file-structured access

Explanation: The remote RT–11 node rejected the remote access because the specified device is not valid for use.

?RTEFTP-F-Connection rejected, Protocol mismatch

Explanation: The remote node rejected the remote access because the message packet indicated an incompatible protocol level.

This message is obsolete.

?RTEFTP-F-Connection rejected, Special directory device

Explanation: The remote RT–11 node rejected the remote access because the specified device is not valid for use.

?RTEFTP-F-Connection rejected, status = *nnnnnn*

Explanation: The remote node rejected the remote access, but the reason code is out of range of those which can be reported by the local machine. The actual return code is displayed as an unsigned octal value)

?RTEFTP-F-Connection rejected, Transfer process unavailable

Explanation: The remote node rejected the remote access because it was already involved in a file transfer.

This message is obsolete.

?RTEFTP-F-Connection rejected, Unable to close file

Explanation: The remote node in a COPY command was unable to make the output file permanent. If the remote node is an RT–11 system, this can be caused by a permanent file of the same name being created, closed, and protected before the file being transferred can be closed.

?RTEFTP-F-Connection rejected, Unable to create file

Explanation: The remote node rejected the remote access because it was unable to create the output file of the required size.

?RTEFTP-F-Connection rejected, Unable to perform function

Explanation: The remote node in a DIRECTORY command was unable to access the directory.

?RTEFTP-F-Connection rejected, Unknown device

Explanation: The remote node rejected the remote access because the device specified was not valid for the remote node.

User Action: Check for a typing error. Verify that the device specified is valid for the remote node and retry the operation.

Preliminary

?RTEFTP-F-Date not set

Explanation: The system date and time have not been established on the local node.

User Action: Set the system date and time and retry the operation.

?RTEFTP-F-Error writing file on remote node *node_name*

Explanation: An error occurred while writing the file on the remote node.

?RTEFTP-F-Error writing file *dev:filnam.typ*

Explanation: An error occurred while writing the file on the local node.

?RTEFTP-F-File not found *dev:filnam.typ*

Explanation: A COPY command was issued which specified a local file which does not exist.

User Action: Check for a typing error. Verify the file exists and retry the operation.

?RTEFTP-F-Help not available for *xxxxxx*

Explanation: There is no help available for the topic specified in a HELP command.

User Action: Check for a typing error. Verify that the topic is valid and retry the operation.

?RTEFTP-F-I/O Error

Explanation: An input/output error occurred on the remote node. The connection has been terminated.

?RTEFTP-F-Insufficient resources

Explanation: There was not enough memory available to perform a locally-initiated DIRECTORY.

User Action: Verify that no virtual circuits are active or reduce the number of entries in the access rights list and retry the operation.

?RTEFTP-F-Insufficient space to expand access control list

Explanation: There was not enough memory available to add another entry to the access control list.

User Action: Verify that there are no active circuits and retry the operation.

?RTEFTP-F-Invalid access specification

Explanation: The file specification in a SET ACCESS command was not valid.

User Action: Check for a typing error. Verify that the format and syntax are correct and retry the operation.

Preliminary

?RTEFTP-F-Invalid command *xxxxxx*

Explanation: The command issued was not recognized by RTEFTP as a valid command.

User Action: Check for a typing error. Verify that the format and syntax are correct and retry the operation.

?RTEFTP-F-Invalid device *dev:*

Explanation: The local device specified in a command was not known to the system.

User Action: Check for a typing error. Verify that the device specified is valid and retry the operation.

?RTEFTP-F-Invalid input file *dev:filnam.typ*

Explanation: The input file specification was in error.

User Action: Check for a typing error. Verify that the format and syntax are correct and retry the operation.

?RTEFTP-F-Invalid multiple node copy

Explanation: An attempt was made to specify a copy between two remote nodes. This is not supported.

?RTEFTP-F-Invalid node *node_name*

Explanation: A node specification contained too many characters.

User Action: Check for a typing error. Verify that the format and syntax are correct and retry the operation.

?RTEFTP-F-Invalid option */x*

Explanation: The access rights option specified was not valid.

User Action: Check for a typing error. Verify that the format and syntax are correct and retry the operation.

?RTEFTP-F-Invalid option *xxxxxx*

Explanation: The option specified was invalid for the command.

User Action: Check for a typing error. Verify that the format and syntax are correct and retry the operation.

?RTEFTP-F-Invalid output file *dev:filnam.typ*

Explanation: The output file specification was in error.

User Action: Check for a typing error. Verify that the format and syntax are correct and retry the operation.

?RTEFTP-F-Invalid syntax

Explanation: The command issued was not valid syntax.

User Action: Check for a typing error. Verify that the format and syntax are correct and retry the operation.

Preliminary

?RTEFTP-F-Node specification required

Explanation: A command which requires specification of a remote node was issued without specifying a remote node. Likely errors include attempts to copy local files to local files or performing local directories.

User Action: Check for a typing error. Verify that the format and syntax are correct and retry the operation.

?RTEFTP-F-Only one timebase allowed

Explanation: A remote node is already providing the timebase.

?RTEFTP-F-Remote access must be enabled first

Explanation: A SET TIME command was issued before the node was enabled for remote access.

User Action: Enable the local node for remote access and retry the operation.

?RTEFTP-F-Transfer already in progress

Explanation: A COPY command was issued while a file transfer (a remote access) was in progress.

This message is obsolete.

?RTEFTP-F-Unbalanced quotes

Explanation: The command had a quoted string which was not properly terminated by a matching quote.

User Action: Check for a typing error. Verify that the format and syntax are correct and retry the operation.

?RTEFTP-F-Unable to create file *dev:filnam.typ*

Explanation: In a locally-initiated file transfer of a file from a remote node to the local node, RTEFTP was unable to create the local file. This can occur when the requested file is larger than the space available on the specified device.

User Action: Free some space on the local device and retry the operation.

?RTEFTP-F-Unrecognized node *node_name*

Explanation: The specified remote node is not known to the local node. This can occur if the remote node has not enabled remote access.

User Action: Wait for the remote node to become available for remote access and retry the operation.

?RTEFTP-U-Error opening channel to Ethernet

Explanation: An error occurred when RTEFTP attempted to open a channel to the Ethernet handler.

User Action: Remove the handler, reinstall it and attempt the operation again.

Preliminary

?RTEFTP-U-Ethernet handler must be loaded - *dev*

Explanation: The required Ethernet handler (NC, NQ or NU) is not loaded.

User Action: Load the appropriate handler by using the LOAD command and retry the operation.

?RTEFTP-U-Ethernet handler not available - *dev*

Explanation: The required Ethernet handler (NC, NQ or NU) was not found in the system tables.

User Action: Make sure NC, NQ or NU has been installed and retry the operation.

?RTEFTP-U-Insufficient memory

Explanation: There is not enough memory available to RTEFTP for the required number of additional queue elements.

User Action: Free some background space by unloading unused handlers and programs or try running RTEFTP under VBGEXE. (Refer to section 3 of the **RT–11 System Message Manual**)

?RTEFTP-U-Insufficient pool memory, use /BUFF:xxx

Explanation: The program has been run under VBGEXE, but there is not enough pool memory available to RTEFTP for the required number of additional queue elements.

User Action: Retry the operation specifying a larger pool using the /BUFF:xxx qualifier as reported.

?RTEFTP-U-Invalid multicast address

Explanation: The multicast address set by customization patch is not a valid multicast address.

User Action: Correct the previously applied customization.

?RTEFTP-U-Invalid protocol type

Explanation: The protocol type set by customization patch is not a valid protocol type for Ethernet. This is due to a protocol type which is in the range normally used for 802.3 packets.

User Action: Correct the previously applied customization.

?RTEFTP-U-Multicast enable error

Explanation: An error occurred when RTEFTP attempted to enable the multicast address used by RTEFTP for message traffic. This can occur if another job on the system has enabled enough multicast addresses to fill the table.

User Action: Remove any currently active programs which use the Ethernet and retry the operation.

Preliminary

?RTEFTP-U-Protocol enable error

Explanation: An error occurred when RTEFTP attempted to enable the protocol type used by RTEFTP for message traffic. This can occur if another copy of RTEFTP is already running on the machine.

User Action: Remove any currently active programs which use the Ethernet and retry the operation.

?RTEFTP-U-Timer support required

Explanation: An attempt was made to run RTEFTP under an XM monitor that was built without specifying timer support during the system generation process.

User Action: RTEFTP requires timer support. Perform a system generation and select timer support.

?RTEFTP-U-Unit allocation error

Explanation: An error occurred when RTEFTP attempted to allocate an Ethernet unit. This can occur if another job on the system is using the Ethernet handler and has allocated a unit other than that for its job.

User Action: Remove any currently active programs which use the Ethernet and retry the operation.

?RTEFTP-U-Unable to reopen asynchronous I/O channel

Explanation: An error occurred when RTEFTP attempted to reopen a channel to the Ethernet handler for use in asynchronous traffic.

?RTEFTP-U-Unable to reopen synchronous I/O channel

Explanation: An error occurred when RTEFTP attempted to reopen a channel to the Ethernet handler for use in synchronous traffic.

?RTEFTP-U-Unable to save channel status

Explanation: An error occurred when RTEFTP attempted to save the status of the channel it has open to the Ethernet handler.

?RTEFTP-U-Wrong version of RT–11

Explanation: An attempt was made to run RTEFTP on a version of RT–11 earlier than version 5.5.

User Action: Run the program under an up-to-date release of RT–11.

?RTEFTP-U-XM Monitor required

Explanation: An attempt was made to run RTEFTP under a monitor other than XM.

User Action: RTEFTP requires an XM monitor. Perform a system generation and select XM.

Preliminary

B Packet codes

Packet codes are found in the first byte of any RTEFTP packet and are used to indicate the contents of the packet beyond the packet header.

Packet type codes, bits <03:00> of the packet code byte, are used to indicate the specific type of packet.

In the following lists, all numbers are in unsigned octal.

Table B–1 Packet type codes

Code	Meaning
0	Control
1	Transfer_Connect
2	Transfer_Data
3	Transfer_Disconnect
4	Directory
5-17	undefined and reserved

Packet type modifiers, bits <05:04> of the packet code byte, are used to further define the contents of the packet.

Table B–2 Control packet modifiers

Code	Meaning
0	Node_Identification packet
20	Broadcast packet
40	undefined and reserved
60	undefined and reserved

Table B–3 Transfer_Connect packet modifiers

Code	Meaning
0	SEND
20	RECEIVE
40	SEND_START
60	undefined and reserved

Table B–4 Transfer_Data packet modifiers

Code	Meaning
0	WRITE
20	READ
40	undefined and reserved
60	undefined and reserved

Preliminary

Table B–5 Transfer_Disconnect packet modifiers

Code	Meaning
0	END
20	ABORT
40	undefined and reserved
60	undefined and reserved

Table B–6 Directory packet modifiers

Code	Meaning
0	Connect
20	Start
40	Data
60	End

The high order two bits of the packet code byte, bits <07:06>, are used to indicate the kind of packet.

Table B–7 Packet modifiers

Code	Meaning
0	Message
100	ACK response
200	undefined and reserved
300	NAK response

Combinations of the above-defined codes are ORed together, forming packet codes from the following list:

Preliminary

Table B–8 Packet codes

Code	Meaning
0	Node_Identification
1	Transfer_Connect SEND
2	Transfer_Data WRITE
3	Transfer_Disconnect END
4	Directory Connect
20	Broadcast
21	Transfer_Connect RECEIVE
22	Transfer_Data READ
23	Transfer_Disconnect ABORT
24	Directory Start
41	Transfer_Connect SEND_START
44	Directory Data
64	Directory End
101	Ack, Transfer_Connect SEND
102	Ack, Transfer_Data WRITE
103	Ack, Transfer_Disconnect END
104	Ack, Directory Connect
121	Ack, Transfer_Connect RECEIVE
122	Ack, Transfer_Data READ
123	Ack, Transfer_Disconnect ABORT
144	Ack, Directory Data
164	Ack, Directory End
301	Nak, Transfer_Connect SEND
302	Nak, Transfer_Data WRITE
303	Nak, Transfer_Disconnect END
304	Nak, Directory Connect
321	Nak, Transfer_Connect RECEIVE
322	Nak, Transfer_Data READ
323	Nak, Transfer_Disconnect ABORT
344	Nak, Directory Data

Preliminary

C Status codes

The following table lists the status codes which may be returned in the second byte of an RTEFTP NAK packet.

In the following table, all numbers are in unsigned octal.

Table C–1 Status Codes

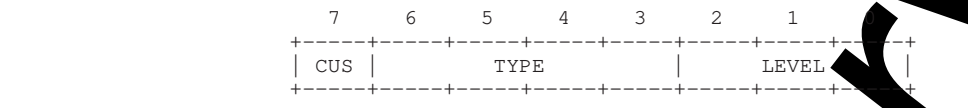
Code	Meaning
377	Transfer process unavailable (obsolete)
376	Invalid file specification
375	Unknown device
374	Device unavailable
373	File not found
372	Unable to create file
371	Unable to close file
370	Protocol mismatch (obsolete)
367	Insufficient space available
366	Non-file-structured access
365	Access denied
364	Special directory device
363	Insufficient resources
362	I/O error
361	Unable to perform function
360	Node is not accepting connections

In designing the file transfer protocol, care was taken to try to make the error codes as system-independent as possible. This list may expand as versions of RTEFTP appear for other operating systems.

Preliminary

D File system codes

The file system codes are defined in the boot-block standard and are included here for reference purposes.



CUS	indicates customer/CSS file system
TYPE	indicates file system type (see Table D–1)
LEVEL	indicates file system level

In the following table, all numbers are in unsigned octal.

Table D–1 File System Codes

Code	Meaning
0	Unknown
10	PDP-8
20	RT–11
30	RSTS
40	Files–11 (ODS-1 or ODS-2)
50	U**X
60	DSM
70	unknown
100	CP/M
110	UCSD P-system
120	MS-DOS
130-170	undefined and reserved
200-370	Customer/CSS use

Preliminary

E Customization patches

This section describes customization patches which may be applied to RTEFTP to change certain operational characteristics.

E.1 Automatic VBGEXE

This customization patch causes RTEFTP to run under VBGEXE context whenever it is run.

This patch applies to V5.6 and is not yet implemented.

E.2 Changing Number of Receive Buffers

This customization patch alters the number of receive buffers for which RTEFTP will attempt to reserve space. More receive buffers increases performance by decreasing the chance that there will not be a receive buffer available when a packet is received.

```
sipp  rteftp.sav/a
Base?      0
Offset?    ..RCV
  Base          Offset        Old   New?
    000000       ..RCV     000006   \

  Base          Offset        Old   New?
    000000       ..RCV        006   <number>
    000000       ..RCV+1      000   ^Y
```

Regardless of the number selected through patching, the program will ensure that there are no fewer than two and no more than six receive buffers. This will be enforced at run-time.

For performance reasons, the space for additional receive buffers is allocated first, so this will have an affect on the number of virtual circuits which may be available.

Preliminary

E.3 Changing Number of Virtual Circuits

This customization patch alters the number of virtual circuits for which RTEFTP will attempt to reserve space. More virtual circuits makes the system available for more simultaneous accesses.

```
.sipp  rteftp.sav/a
Base?       0
Offset?    ..CIR
   Base            Offset         Old   New?
      000000        ..CIR      000010   \

   Base            Offset         Old   New?
      000000        ..CIR         010   <number>
      000000        ..CIR+1       000   ^Y
```

E.4 Changing Multicast Address

This customization patch alters the multicast address used by RTEFTP for control packets. It should not be altered unless it is determined that operation of RTEFTP is conflicting with another program using the same multicast address.

The multicast address built into the program, Hex AB-00-04-00-60-06, was selected because it resides in the range of multicast addresses available to Digital Customers. It should not conflict with any commercially-available products, but care should be taken to determine what Multicast Addresses are in use on your network and to ensure that there are no conflicts.

The Ethernet address to be used as the multicast address should be converted to the equivalent byte values for placement in the image.

```
.sipp  rteftp.sav/a
Base?       0
Offset?    ..MA
   Base            Offset         Old   New?
      000000        ..MA       000253   \

   Base            Offset         Old   New?
      000000        ..MA          253
      000000        ..MA+1        000
      000000        ..MA+2        004
      000000        ..MA+3        000
      000000        ..MA+4        140
      000000        ..MA+5        006
      000000        ..MA+6        000   ^Y
```

If the address specified is not a valid multicast address, the program will terminate with an error message to that affect.

Preliminary

E.5 Changing Protocol Type

This customization patch alters the protocol type used by RTEFTP for all packets. It should not be altered unless it has been determined that operation of RTEFTP is conflicting with another program using the same protocol type.

The protocol type built into the program, Hex 60-06, was selected because it is the protocol type available to Digital Customers. It should not conflict with any commercially-available products, but care should be taken to determine what Protocol Types are in use on your network.

The hex Protocol Type to be used by RTEFTP should be converted to the equivalent byte values for placement in the image.

```
.sipp  rteftp.sav/a
Base?      0
Offset?   ..PT
   Base          Offset        Old   New?
      000000       ..PT      003140   \

   Base          Offset        Old   New?
      000000       ..PT         140
      000000       ..PT+1       006
      000000       ..PT+2       000   ^Y
```

If the address specified is not a valid protocol type, the program will terminate with an error message.

Preliminary