InterNiche TFTP Technical Note

The InterNiche implementation of TFTP is fully compliant with RFC1350 - The TFTP Protocol (Revision 2).

TFTP Client is built into the InterNiche stack with the option **TFTP_CLIENT** and TFTP Server with the option **TFTP_SERVER** in the **ipport.h_h** file. TFTP uses the UDP protocol to send and receive files to remote systems. In the **makefile**, the **LIB** directive needs to include **TFTPLIB**.

Important Note!
All Filenames in TFTP Must Be Fully Qualified Path Names.

TFTP Menu Commands

TFTP Client

help tftpc	TFTP client menu
tfget	TFTP **get** a file
tfput	TFTP **put** a file

TFTP Server

tfsrv	toggle TFTP server on/off

NET_STATS defined

tfstate	displays TFTP server statistics

To **get** a file the user issues the **tfget** command:

tfget *<host ip addr> <dest filename> <source filename>*

To **put** a file the user issues the **tfput** command:

tfput *<host ip addr> <source filename> <dest filename>*

Remember all filenames in TFTP have to be fully qualified path names.

If an error occurs, the error message string is displayed.

TFTP client error: *<error message string>*

On completion, the following message is displayed:

tftp from|to *<host ip addr>* done, msg: None*(or error message string)*,
status: ok or *error number (decimal value of error)*

To toggle the TFTP server On or Off, the user issues the **tfsrv** command:

```
tfsrv
```

The message displayed depends on the current state of the server:

```
tftp server ON
tftp server OFF
```

To display TFTP server statistics, the user issues the **tfstate** command:

```
tfstate
```

The statistics message is as follows:

```
connection: <put to|get from> <host id_addr>, state: <dec value of state>
bytes moved: <total # of bytes moved>
<number of open connections> connections open
```

State Values

```
/* TFTP states */
#define DATAWAIT      1   /* GET: sent req; waiting for data */
#define ACKWAIT       2   /* PUT: sent data; waiting for ack */
#define DEAD          3   /* a killed connection, waiting for cleanup */
#define TIMEOUT       4   /* connection timed out */
#define RCVERR        5   /* connection got error from other side */
#define RCVACK        6   /* PUT: got ack, sending data */
#define RCVDATA       7   /* GET: sent ack; waiting for next block */
#define RCVLASTDATA   8   /* GET: received last block of file (A block > 512 bytes) */
#define SENTLAST      9   /* PUT: sent last packet, awaiting ack */
#define TERMINATED    10  /* PUT: sent last packet & and got ack for it */
```

Server Packet Types

The TFTP Server handles the following TFTP packet types:

```
/* TFTP opcodes */
#define  TF_RRQ       1   /* read  request */
#define  TF_WRQ       2   /* write request */
#define  TF_DATA      3   /* data packet */
#define  TF_ACK       4   /* acknowledgement packet */
#define  TF_ERROR     5   /* error packet */
```

Error Codes

ERRTXT	0	See text
FNOTFOUND	1	File not found
ACCESS	2	Access violation
DISKFULL	3	Disk full
ILLTFTP	4	Illegal TFTP operation
BADTID	5	Unknown transfer ID
FEXISTS	6	File already exists
NOUSER	7	No such user

Error Text Values Specified by ERRTXT

Bad mode - when an invalid mode command other than ASCII, binary, or image is specified

Transfers currently disabled - when the Server is not set to ON with the **tfsrv** command. The default value for this is OFF

Transfer refused - when the user has specified a security call-back function and the call-back function has refused to allow this file transfer. The default value for this call-back function is NULL.

Bad len (too short) - length less than the TFTP packet header size

Rcvd unexpected data block - received a data block when the TFTP connection state is not waiting for data

Retry limit exceeded, giving up - failed in attempts to send an UDP packet