KwikNet®

FTP Client / Server

User's Guide

Version 3

First Printing: **July 27, 1998**
Last Printing: **September 15, 2005**

Manual Order Number: PN303-9F

KADAK Products Ltd.
206 - 1847 West Broadway Avenue
Vancouver, BC, Canada, V6J 1Y5
Phone: (604) 734-2796
Fax: (604) 734-8114

TECHNICAL SUPPORT

KADAK Products Ltd. is committed to technical support for its software products. Our programs are designed to be easily incorporated in your systems and every effort has been made to eliminate errors.

Engineering Change Notices (ECNs) are provided periodically to repair faults or to improve performance. You will automatically receive these updates during the product's initial support period. For technical support beyond the initial period, you must purchase a Technical Support Subscription. Contact KADAK for details. Please keep us informed of the primary user in your company to whom update notices and other pertinent information should be directed.

Should you require direct technical assistance in your use of this KADAK software product, engineering support is available by telephone, fax or e-mail. KADAK reserves the right to charge for technical support services which it deems to be beyond the normal scope of technical support.

We would be pleased to receive your comments and suggestions concerning this product and its documentation. Your feedback helps in the continuing product evolution.

KADAK Products Ltd.
206 - 1847 West Broadway Avenue
Vancouver, BC, Canada, V6J 1Y5

Phone: (604) 734-2796
Fax: (604) 734-8114
e-mail: amxtech@kadak.com

DISCLAIMER

KADAK Products Ltd. makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability and fitness for any particular purpose. Further, KADAK Products Ltd. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of KADAK Products Ltd. to notify any person of such revision or changes.

TRADEMARKS

AMX in the stylized form and KwikNet are registered trademarks of KADAK Products Ltd.
AMX, AMX/FS, InSight, *KwikLook* and KwikPeg are trademarks of KADAK Products Ltd.
UNIX is a registered trademark of AT&T Bell Laboratories.
Microsoft, MS-DOS and Windows are registered trademarks of Microsoft Corporation.
All other trademarked names are the property of their respective owners.

KwikNet FTP Client / Server User's Guide

Table of Contents

Table of Figures

This page left blank intentionally.

1. KwikNet FTP Overview

1.1 Introduction

The File Transfer Protocol (FTP) is a standard protocol used for transferring files between machines over TCP/IP based networks. The KwikNet FTP option implements this protocol on top of the KwikNet™ TCP/IP Stack, a compact, reliable, high performance TCP/IP stack, well suited for use in embedded networking applications.

The KwikNet FTP option is best used with a real-time operating system (RTOS) such as KADAK's AMX™ Real-Time Multitasking Kernel. However, the KwikNet FTP option can also be used in a single threaded environment without an RTOS. The KwikNet Porting Kit User's Guide describes the use of KwikNet with your choice of RT/OS. Note that throughout this manual, the term RT/OS is used to refer to any operating system, be it a multitasking RTOS or a single threaded OS.

You can readily tailor the KwikNet stack to accommodate your FTP needs by using the KwikNet Configuration Builder, a Windows® utility which makes configuring KwikNet a snap. Your KwikNet stack will only include the FTP features required by your application.

This manual makes no attempt to describe the File Transfer Protocol (FTP), what it is or how it operates. It is assumed that you have a working knowledge of the FTP protocol as it applies to your needs. Reference materials are provided in Appendix A of the KwikNet TCP/IP Stack User's Guide.

The purpose of this manual is to provide the system designer and applications programmer with the information required to properly configure and implement a networking system using the KwikNet TCP/IP Stack and FTP. It is assumed that you are familiar with the architecture of the target processor.

KwikNet and its options are available in C source format to ensure that regardless of your development environment, your ability to use and support KwikNet is uninhibited. The source program may also include code fragments programmed in the assembly language of the target processor to improve execution speed.

The C programming language, commonly used in real-time systems, is used throughout this manual to illustrate the features of KwikNet and its FTP option.

> Note
>
> The KwikNet FTP option is founded upon the FTP client and server from Treck Inc. Hence you must become familiar with the FTP application programming interface (API) described in Chapter 6 of the Treck TCP/IP User Manual.

1.2 General Operation

The File Transfer Protocol (FTP) is a standard protocol used for transferring files between machines over TCP/IP based networks such as the Internet. FTP is formally defined by the IETF document RFC-959. The KwikNet FTP option is compliant with that specification. The RFC should be consulted for any detailed questions concerning the FTP protocol. The KwikNet FTP option implements the subset of FTP features typically required for use in embedded applications.

FTP is a client-server protocol. One machine, the client, initiates a file transfer by contacting another machine, the server. The client issues a sequence of requests to the server in order to transfer the files. The server must be operating before the client initiates its requests. Generally, a client communicates with one server at a time while most servers are designed to work concurrently with multiple clients.

The KwikNet FTP option provides all of the services necessary to implement one or more FTP clients and a single FTP server. Although multiple clients and the server can coexist and operate concurrently, most applications will require only a single FTP client or a single FTP server.

FTP Commands

When an FTP client contacts an FTP server, a TCP connection is established between the two machines. The server does a passive open by listening on a TCP socket for requests from potential clients. The client can then connect its own TCP socket to the server. This connection is referred to as the FTP command connection. The command connection persists for as long as the client maintains a session with the server. As its name implies, the command connection is used to convey commands from the client to the server and to return replies from the server back to the client.

An FTP command is an ASCII string which always includes a command directive followed by zero or more command parameters. All commands are terminated by a two character, end of line sequence consisting of a carriage return (ASCII 13, `0x0D`, `'\r'`) followed by a linefeed (ASCII 10, `0x0A`, `'\n'`). Figure 1.2-1 lists the FTP command strings which the KwikNet FTP client can generate and which the KwikNet FTP server can interpret.

The FTP server reply is an ASCII string consisting of a 3 digit decimal response code followed by some explanatory text. Generally, codes in the 200 range indicate success and codes in the 500 range indicate failures. See the RFC for a complete guide to reply codes.

Command		Purpose
USER username		User name provided to start session
PASS password		Password needed to establish session
ACCT accountid	!	Account identifier needed to establish session
ABOR		Abort the previous command
QUIT		Terminate FTP session
CWD dirname		Specify current working directory
CDUP		Set current working directory up one level
PWD		Print (show) current working directory
RMD dirname		Remove (delete) a directory at the FTP server
MKD dirname		Make (create) a directory at the FTP server
LIST dirname		List files and attributes for specified directory
NLST dirname		List files (names only) for specified directory
LIST		List files and attributes for current working directory
NLST		List files (names only) for current working directory
STRU code	*	Specify file structure (File and Record only)
MODE code	*	Specify transfer mode (Stream only)
TYPE code		Specify text or binary data (*ASCII* or *IMAGE*)
RETR filepath		Retrieve a file from the FTP server
STOR filepath		Store a file at the FTP server
STOU	*	Store a file at the FTP server; give it a unique filename
APPE filepath		Append to a file at the FTP server
DELE filepath		Delete a file at the FTP server
RNFR oldfilename		Identify a file at the FTP server which is to be renamed
RNTO newfilename		Rename a file identified by a previous *RNFR* command
PORT h1,h2,h3,h4,p1,p2		Specify data connection IP address and port number
PASV		Request FTP server to use a passive data connection
HELP	!	Provide help for command use
REIN		Re-initialize FTP session
REST	!*	Restart
SITE parameters	!*	Site specific commands
SYST		Show operating system information
NOOP		No operation

Note ! FTP server does not support this command
* FTP client cannot send this command

Figure 1.2-1 KwikNet FTP Commands

FTP Data Transfers

If the FTP command requires the server to move a quantity of data, such as a file or a directory listing, across the network, the FTP client and server establish a second TCP connection. This connection is referred to as the FTP data connection. The data connection is usually made as follows. The FTP client creates a listening socket and sends its IP address and data port number to the server. The client then requests the FTP server to send or receive data. At that point, the FTP server actively connects to the client's data port and the transfer begins.

Alternatively, the FTP server can play a passive role when establishing the data connection. In this case, the FTP client requests the FTP server to enter passive mode. The server creates a listening socket and sends its IP address and data port number to the client. The client then requests the server to send or receive data. The server then passively waits for the connection. At that point, the FTP client actively connects to the server's data port and the transfer begins.

The KwikNet FTP server supports both the active and passive FTP data connection methods.

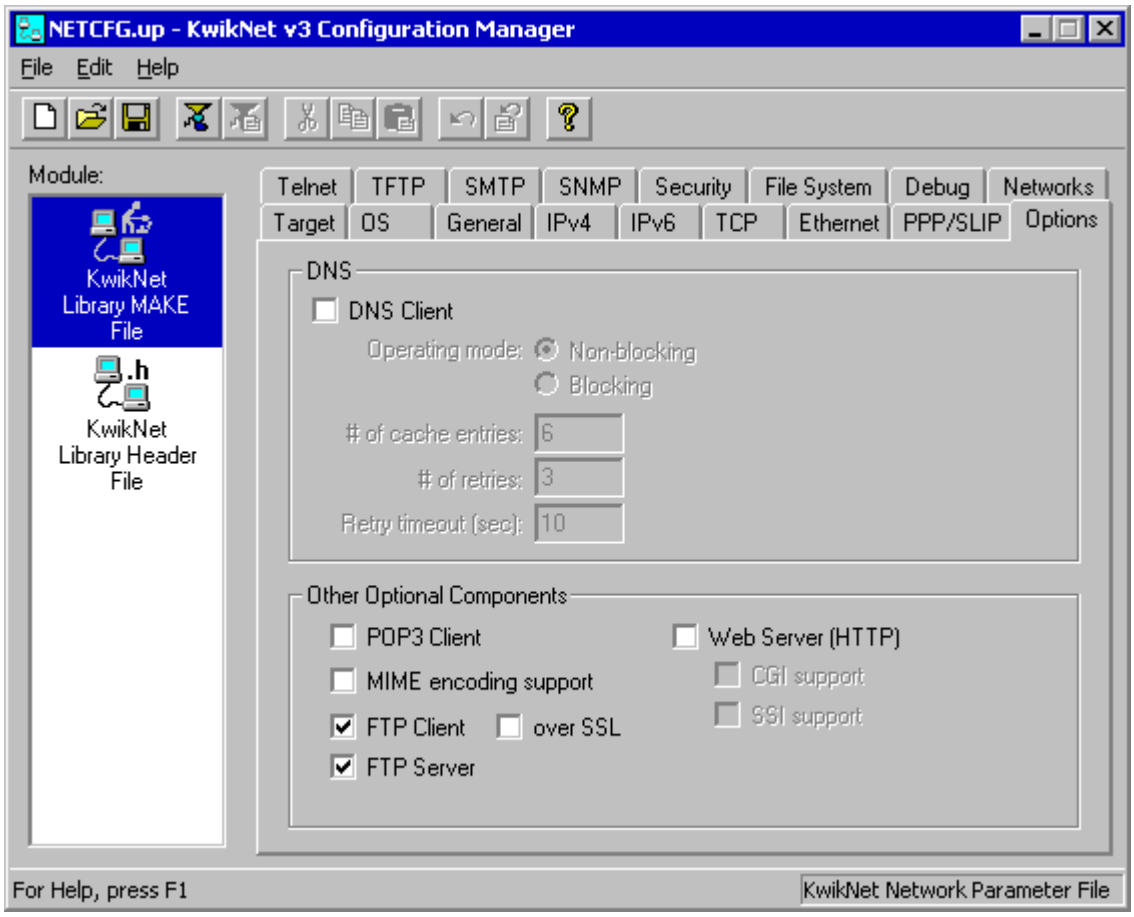
The KwikNet FTP client also supports both the active and passive FTP data connection methods. When operating in passive mode, the KwikNet FTP client sends a *PASV* command to the FTP server prior to establishing a data connection. The client then initiates the data connection to the passively listening FTP server.

Unlike many FTP servers, the KwikNet FTP server will accept an *ABOR* abort command during a data transfer, allowing an errant transfer to be terminated upon request from the client.

1.3 KwikNet FTP Configuration

You can readily tailor the KwikNet stack to support an FTP client or server by using the KwikNet Configuration Builder to edit your KwikNet Network Parameter File. The KwikNet Library parameters are edited on the Options property page. The layout of the window is shown below.

Note that the TCP protocol and a file system are prerequisites for the FTP protocol. You must include these components in your KwikNet Network Parameter File in order to use the FTP protocol.



FTP Parameters (continued)

FTP Client

Check this box if your application will include an FTP client which connects to an FTP server for file transfers. Otherwise, leave this box unchecked.

Check the box labeled "over SSL" if any FTP client will use the Secure Sockets Layer (SSL) for secure file transfers. Otherwise, leave the box unchecked.

Note: To use SSL with the FTP client, you require the optional version of the FTP client which includes SSL support.

Note: If you choose the FTP client over SSL option, any FTP client can operate with or without security, according to the requirements of each particular FTP session.

FTP Server

Check this box if your application will include an FTP server to provide network access to files present in your target system. Otherwise, leave this box unchecked.

1.4 File Services

As its name implies, the File Transfer Protocol is used for file transfers. Consequently, every FTP client and server requires access to a file system for the actual retrieval and storage of files. KwikNet includes a file system interface which provides any KwikNet client or server with access to one of the supported file systems: the AMX/FS File System, standard C file I/O, a user defined file system or the Treck RAM file system. This file system interface is described in Appendix C of the KwikNet TCP/IP Stack User's Guide.

KADAK's AMX/FS File System is ready for use with the AMX Real-Time Multitasking Kernel. Since AMX/FS includes a RAM disk driver, it is well suited for use with FTP in embedded applications.

The standard C file I/O library from the C compiler vendor can be used if it is available for the target processor. Unfortunately, few C libraries provide file services for embedded targets.

Special considerations apply when using AMX 86 which can coexist with MS-DOS®. AMX 86 includes a component called the PC Supervisor which permits tasks running under AMX 86 to concurrently access the MS-DOS file system using standard C file I/O calls. Hence, when using AMX 86, a KwikNet FTP client or server can connect to the standard C file I/O library.

> Note
>
> The file system must be ready for use before any KwikNet client or server starts to use the services in the KwikNet Library. See Appendix C of the KwikNet TCP/IP Stack User's Guide for details.

End of Line Indication

The use of CR (`'\r'`, ASCII `0x0D`), LF (`'\n'`, ASCII `0x0A`) or CRLF (CR followed by LF) as an end of line indicator presents a significant challenge when transferring files between different operating systems. The FTP protocol demands that CRLF be used as the end of line indicator for files sent in text mode.

The Treck file system API supports both binary and record oriented file access. For text transfers, files are read or written one record at a time through a streaming buffer. The underlying file system's end of line indicator (be it CR, LF or CRLF) is stripped or added as the text record is transferred to or from the FTP client or server.

For binary transfers, the FTP client and server read and write files in binary mode, transferring file data unaltered from or to the underlying file system.

1.5 FTP Client Task

The KwikNet FTP option includes a set of services for use by one or more FTP clients. Each FTP client is an application program which makes use of these services to communicate with an FTP server. The collection of application procedures which makes up such a program is called an FTP client task.

Before an FTP client task can use any KwikNet FTP client services, the underlying file system must be initialized and ready for use. All disk drives which the FTP client will be permitted to access must be mounted and ready for use.

Your FTP client task must call *tfFtpNewSession()* to create an FTP session. The client can then call *tfFtpConnect()* to establish a connection with a particular FTP server. The IP address of the FTP server must be provided by your client task. The client can then call *tfFtpLogin()* to identify itself. A valid user name must be provided to open a session with an FTP server. Most FTP servers will also require a password. Both user name and password must be provided by your client task in its call to *tfFtpLogin()*.

Once the session with an FTP server has been established, the client task can use the KwikNet FTP client services described in Chapter 6 of the Treck TCP/IP User Manual to converse with the server to transfer files or manipulate directories.

Using KwikNet client services is much like using a file system. Operations go to completion or until an error condition is encountered. Hence, when an application makes a call to a client service procedure, that operation must be allowed to complete before another operation can be initiated by the application.

When the connection with a particular FTP server is no longer required, the FTP client task terminates the connection with a call to procedure *tfFtpQuit()*. If this call fails to successfully terminate the connection, you should call procedure *tfFtpClose()* to forcefully break the connection.

When the FTP client is finished its final FTP connection, the FTP session can be terminated with a call to *tfFtpFreeSession()*. No further FTP transactions can be initiated by the client without first calling *tfFtpNewSession()* to establish a new session.

The FTP Sample Program provided with the KwikNet FTP option illustrates an interactive FTP client task. The client task uses the KwikNet console driver (see Chapter 1.8 of the KwikNet TCP/IP Stack User's Guide) to implement a command line interface with a user at a simple, interactive console device.

Multitasking Operation

When used with a real-time operating system (RTOS) such as KADAK's AMX Real-Time Multitasking Kernel, each FTP client must be an application task. Although one task can be written to service multiple FTP connections, it is usual, and conceptually simpler, to consider each FTP client to be a unique task. Such a task is referred to as an FTP client task.

An FTP client task is created and started just like any other application task. Although an FTP client session can be created to operate in non-blocking mode, it is more usual to operate in blocking mode. In that case, every FTP operation initiated by the client will go to completion before control is returned to the client task.

> Note
>
> In multitasking systems, each KwikNet client task MUST execute at a priority below that of the KwikNet Task.

Single Threaded Operation

When used with a single threaded operating system, the FTP client operates in the user domain as part of your App-Task as described in Chapter 1.2 of the KwikNet TCP/IP Stack User's Guide. When your App-Task is executing your client code, the App-Task is referred to as an FTP client task.

The client FTP session must be created to operate in non-blocking mode. Upon return from each client operation, the client task must check the return error code. If the operation completes successfully, the client task can proceed to its next operation. If the request fails, the client task can immediately take steps, if possible, to correct the situation. If the error code returned to the client task is *`TM_EWOULDBLOCK`*, the client task must repeatedly call procedure *`tfFtpUserExecute()`* to service the FTP request until the operation completes with an error code other than *`TM_EWOULDBLOCK`*.

While executing as an FTP client, your App-Task must continue to regularly call KwikNet procedure *`kn_yield()`* to let the KwikNet TCP/IP Stack continue to operate. The FTP Sample Program implements procedure *`sam_wait()`* which alternately calls *`kn_yield()`* and *`tfFtpUserExecute()`* while waiting for the completion of an FTP client request.

Although KwikNet can support multiple, concurrent FTP connections, it is up to your FTP client task to manage the separate FTP transaction sequences for each of the connections.

1.6 FTP Server Task

The KwikNet FTP option includes a set of services which can be used to implement an FTP server. The FTP server is an application program which makes use of these services to communicate with one or more FTP clients.

Before the FTP server can use any KwikNet FTP services, the underlying file system must be initialized and ready for use. All disk drives which the FTP server will be permitted to access must be mounted and ready for use.

The FTP server will service all FTP requests directed to any of the IP addresses assigned to the server's network node. The server will therefore accept client requests received on any of the operational (open) network interfaces which KwikNet services.

The FTP server accepts requests directed to the well known FTP command port number 21. Data transfers will occur on the associated FTP data port number 20.

The FTP Sample Program provided with the KwikNet FTP option includes a fully functional FTP server.

Multitasking Operation

When used with a real-time operating system (RTOS) such as KADAK's AMX Real-Time Multitasking Kernel, the FTP server operates as an application task. Such a task is referred to as an FTP server task. Only one FTP server task is allowed.

The FTP server task is created and started just like any other application task. When ready to begin operation, the server task simply calls procedure *`tfFtpdUserStart()`* to establish an FTP server session and begin service. Although an FTP server session can be created to operate in non-blocking mode, it is more usual to operate in blocking mode. In that case, there is no return from procedure *`tfFtpdUserStart()`* until the server is forced to stop.

The FTP server task will operate until some unrecoverable error condition is detected or until some other application task calls procedure *`tfFtpdUserStop()`* requesting the server to stop. The FTP server task will abort all of its active FTP sessions and resume execution following the initial call to *`tfFtpdUserStart()`*.

> Note
>
> In multitasking systems, the KwikNet FTP server task MUST execute at a priority below that of the KwikNet Task.

Single Threaded Operation

When used with a single threaded operating system, the FTP server operates in the KwikNet domain in the context of the KwikNet Task as described in Chapter 1.2 of the KwikNet TCP/IP Stack User's Guide.

When your App-Task calls KwikNet procedure *tfFtpdUserStart()*, an FTP server session is established. The FTP server must be started to operate in non-blocking mode.

Once the FTP server has been started, you must periodically call procedure *tfFtpdUserExecute()* to allow the server to service its clients. The easiest way to do this is to use KwikNet procedure *kn_addserver()* to add a server function to the KwikNet server queue. Thereafter, the KwikNet Task will call your server function at the periodic interval specified by you in your call to *kn_addserver()*.

When an FTP server function is added to the KwikNet server queue, the function is referred to as an FTP server task. Only one FTP server task is allowed.

The FTP Sample Program implements server function *ftps_service()* which calls *tfFtpdUserExecute()* at the specified service interval until the server is stopped.

Once the FTP server is operational, your App-Task must regularly call KwikNet procedure *kn_yield()* to let KwikNet and all server tasks, including your FTP server task, operate.

The FTP server task will operate until some unrecoverable error condition is detected or until your App-Task calls procedure *tfFtpdUserStop()* requesting the server to stop. The FTP server task will abort all of its active FTP sessions and remove itself from the KwikNet server queue as illustrated by service function *ftps_service()* in the FTP Sample Program.

FTP User Administration

An FTP server is expected to guard the local operating system from access by unauthorized FTP clients. The FTP client user name and password form the first two levels of protection. A connection with a client must be prohibited unless the client's user name and password are acceptable locally.

Unfortunately, most operating systems used in embedded applications are not like UNIX; they do not require or support user names and passwords. The KwikNet Administration interface resolves this dilemma by allowing you to define the users which will be permitted access to your system. Follow the instructions presented in Appendix D of the KwikNet TCP/IP Stack User's Guide.

Directory Access by Users

Each of your defined users can be given a unique base directory which will be the default current working directory established when that user logs in as an FTP client. The definition of the user's base directory must be a full path, including drive if required by the file system.

The extent to which a user is permitted to traverse directories is determined by the visibility access right granted in the user's definition. Unless the user has been granted full visibility, the user will only be allowed to traverse directories forward from the user's default directory.

> Note
>
> User names, passwords and access rights are handled by the KwikNet Administration interface which is described in Appendix D of the KwikNet TCP/IP Stack User's Guide.

2. KwikNet FTP Applications

2.1 FTP Sample Program

An FTP Sample Program is provided with the KwikNet FTP option to illustrate the use of the KwikNet FTP client and server. The sample program is ready for use with the AMX Real-Time Multitasking Kernel and the Treck RAM File System. The sample program can also be used with any of the porting examples provided with the KwikNet Porting Kit.

With simple modifications to the configuration and link process, the KwikNet FTP Sample Program can also be adapted to use the AMX/FS File System or your own custom file system. The AMX 86 sample can be adapted to use the PC Supervisor to access MS-DOS® file services.

The sample configuration supports a single network interface. The network uses the KwikNet Ethernet Network Driver. Because the sample must operate on all supported target processors without any specific Ethernet device dependence, KwikNet's Ethernet Loopback Driver is used. Use of this driver allows the FTP client and server to be tested even if network hardware is not available. Once the FTP Sample Program has been tested in loopback fashion, you can replace the Ethernet Loopback Driver with your own network device driver. Then the KwikNet FTP client will be able to connect to other FTP servers and foreign clients will be able to access the KwikNet FTP server.

The KwikNet TCP/IP Stack requires a clock for proper network timing. The examples provided with the KwikNet Porting Kit all illustrate the clock interface. If you are using KwikNet with AMX, you must provide an AMX clock driver. If you have ported the AMX Sample Program to your hardware platform, you can use its AMX Clock Driver.

The sample includes an FTP server task and an FTP client task. The client uses the KwikNet console driver to provide a command line interface with a user. The console driver can be configured as described in Chapter 1.8 of the KwikNet TCP/IP Stack User's Guide to use any of several possible terminal devices as an interactive terminal. If you are using KwikNet with AMX and have ported the AMX Sample Program to your hardware platform, you can use its serial UART driver for console I/O.

The sample also uses the KwikNet data logging and message recording services to record messages generated by the KwikNet TCP/IP Stack as it operates. These services are described in Chapters 1.6 and 1.7 of the KwikNet TCP/IP Stack User's Guide. The messages are recorded into an array of strings in memory. The FTP client's interactive *dump* command can be used to list these messages on the console device and empty the recording array.

Startup

The manner in which the KwikNet FTP Sample Program starts and operates is completely dependent upon the underlying operating system with which KwikNet is being used. All sample programs provided with KwikNet and its optional components share a common implementation methodology which is described in Appendix E of the KwikNet TCP/IP Stack User's Guide. Both multitasking and single threaded operation are described.

When used with AMX, the sample program operates as follows. AMX is launched from the *main()* program. Restart Procedure *rrproc()* starts the print task, creates the FTP server task and then creates and starts the FTP client task. The FTP server task remains idle until started by the FTP client task as will be described.

Once the AMX initialization is complete, the high priority print task executes and waits for the arrival of AMX messages in its private mailbox. Each AMX message includes a pointer to a log buffer containing a KwikNet message to be recorded.

Once the print task is ready and waiting, the FTP client task finally begins to execute. It starts KwikNet at its entry point *kn_enter()*. KwikNet self starts and forces the KwikNet Task to execute. Because the KwikNet Task operates at a priority above all tasks which use its services, it temporarily preempts the FTP client task. The KwikNet Task initializes the network and its associated loopback driver and prepares the IP and TCP protocol stacks for use by the sample program.

Once the KwikNet initialization is complete, the FTP client task resumes execution.

FTP Client Operation

Once the KwikNet initialization is complete, the FTP client task resumes execution. It initializes the KwikNet console driver and generates a signon message on the console device. It then calls *tfFtpNewSession()* to create an FTP client session.

The FTP client generates a command line prompt *"KwikNet FTP>"* and waits for a user to enter a lower case directive and any parameters required by that directive. The directive is terminated by the Enter key (*'\r'* character).

The FTP client decodes the directive and makes the call to the appropriate Treck FTP client service procedure to perform the requested action. Since every client service procedure is exercised by the FTP client task, its code can serve as an excellent programming model for your own FTP client software.

The following complete list of directives will be presented if either *help* or *?* is entered as the command line directive.

```
Commands:
   help -  Display this text.
   exit -  Terminate this sample program.
   open  <server IPv4 address> <port>[ <username>]
   open6 <server IPv6 address>[%<scope id>] <port>[ <username>]
              Connect to an FTP server.  If port is 0,
              FTP server port 21 will be used.
              For "anonymous, guest", omit user name.
   close    - End a previously opened FTP session.
   pasv on  - FTP server to use passive data connects.
   pasv off - FTP server to use active data connects.
   ascii - File transfers are ASCII.
   binary- File transfers are binary.
   list  - List files and attributes.
   nlst  - List file names only.
   lfile <local file path> - Specify a local file path
           for use with 'get' and 'put' file transfers.
   get <remote file path> - Retrieve a file from server.
   put <remote file path> - Send a file to the server.
   app <remote file path> - Append to file on the server.
   del <remote file path> - Delete a file on the server.
   cwd <remote path> - Change current working directory.
   pwd - Show current working directory at the server.
   mk <remote path> - Create new directory on the server.
   rm <remote path> - Remove directory from the server.
   syst - Get system information from the server.
   last - Show the status of the last FTP command.
   ok   - See if connected to FTP server.

   serv - Start the KwikNet FTP server on this machine.
   stop - Stop  the KwikNet FTP server.
   dump [stat] - Dump KwikNet recorded log [and statistics].

glossary:
   <text> - String you must provide.
   [optional] - Parameter(s) within [] can be omitted.
            (omit the <,>,[ and ] characters).
```

FTP Server Operation

The FTP client cannot open a connection to an FTP server unless such a server exists on the network. Unless you have replaced the Ethernet Loopback Driver with a real device driver, the FTP Sample Program has no direct network connection. Hence no FTP server is accessible.

So why not use the KwikNet FTP server? If you give the FTP client the *serv* directive, it will start the KwikNet FTP server task which will immediately begin operation since it is of higher priority than the client task. After starting the server task, the client task pauses briefly and then fetches the FTP server task's IP address and port number.

The FTP client task displays the server's IP address and port number giving you, the user, the chance to see that a server now exists to whom you can connect. Use the *open* directive to connect to the server at that IP address and port number and log in as user *anonymous* with password *guest*. The KwikNet FTP client is now conversing with the KwikNet FTP server *across the network* even though both are executing on the same processor.

At any time, you can enter the *stop* directive which causes the FTP client to call KwikNet procedure *tfFtpdUserStop()* requesting the FTP server task to stop execution. If the client still has an open connection to the server, the connection will be broken by the server before the server ceases to operate. In this case, if you issue the *ok* directive, you will probably observe a message indicating that the client's connection to the server has been lost.

Shutdown

When the FTP client task decodes the *exit* directive, it closes any open connection which it may have had with an FTP server. It then terminates the FTP client session for which it has been responsible.

If the FTP server task is still operating, the FTP client task requests it to stop.

Next, the FTP client task generates a signoff message and relinquishes use of the KwikNet console driver. It then calls procedure *kn_exit()* to stop operation of the KwikNet TCP/IP Stack.

Finally, after pausing briefly, it initiates a shutdown of the underlying operating system (if possible) and a return to the *main()* procedure.

KwikNet and FTP Server Logging

The FTP Sample Program uses the simple KwikNet message recording service to log text messages. The recorder saves the recorded text strings in a 30,000 byte memory buffer until either 500 strings have been recorded or the memory buffer capacity is reached.

The FTP Sample Program directs messages to this recorder by calling the KwikNet log procedure *kn_dprintf()*. This procedure operates similarly to the C *printf()* function except that an extra integer parameter of value *0* must precede the format string. The FTP client task uses this feature to record a shutdown message.

KwikNet formats the message into a log buffer and passes the buffer to an application log function for printing. Log function *sam_record()* in the KwikNet Application OS Interface serves this purpose.

In a multitasking system the log buffer is delivered as part of an RTOS dependent message to a print task. The print task calls *kn_logmsg()* in the KwikNet message recording module to record the message and release the log buffer.

In a single threaded system, the log function *sam_record()* can usually call *kn_logmsg()* to record the message and release the log buffer. However, if the message is being generated while executing in the interrupt domain, the log buffer must be passed to the KwikNet Task to be logged. The sample programs provided with the KwikNet Porting Kit illustrate this process.

Since the recorded strings are just stored in memory, they are not readily visible. To overcome this restriction, you can use the FTP client's interactive *dump* command to list all of the recorded messages on the client's console device and empty the recording array.

Alternatively, if a debugger is used to control execution of the FTP Sample Program, the program can be stopped and the strings can be viewed in text form in a display window by viewing the array variable *kn_recordlist[]* in module *KNRECORD.C*.

Client Logging

The FTP Sample Program's client task provides its own logging function *ftp_logfn()* which is used to display the lines of text which it receives following a request to an FTP server for a directory listing.

The logging function directs its output to the same console terminal which is used by the client task for its command line user interface.

Running the Sample Program

The KwikNet FTP Sample Program requires that your target hardware include an interface to a console terminal. The FTP client task will use its command line interface to interact with you at that terminal. You are therefore the real user behind the FTP client task.

Each action which you initiate using a command line directive will generate a response on the terminal as the client task handles your request. The FTP server task will only run if you use the *serv* directive to start it. The FTP Sample Program runs until you issue the *exit* directive to shut it down.

KwikNet includes a number of debug features (see Chapter 1.9 of the KwikNet TCP/IP Stack User's Guide) which can assist you in using the FTP Sample Program. With KwikNet's debug features enabled, you can place a breakpoint on procedure *kn_bphit()* to trap all errors detected by KwikNet. Of course, if you are using AMX, it is always wise to execute with a breakpoint on the AMX fatal exit procedure *cjksfatal* (*ajfatl* for AMX 86).

KwikNet records selected debug and trace information if any of these features are enabled. Unless you have modified the KwikNet recording method, these messages are simply saved in memory and are therefore not visible. However, you can use the FTP client's interactive *dump* command to list all of the recorded messages on the client's console device and empty the recording array.

2.2 Making the FTP Sample Program

The sheer volume of detail necessary to understand and use FTP with TCP/IP may at first be daunting. However, constructing the KwikNet FTP Sample Program is actually a fairly simple process made even simpler by the KwikNet Configuration Manager, a Windows® utility provided with KwikNet.

The FTP Sample Program includes all of the components needed to build the sample application for a particular target processor and file system. You can take these components and, with minor modifications, adapt them for your particular target processor and development environment.

As delivered, the KwikNet FTP Sample Program uses the Treck RAM File System. However, the sample can also be used with AMX or AMX 86 and the AMX/FS File System. For PC targets with AMX 86, it can be used with MS-DOS® file services. Use the KwikNet Configuration Manager to edit the sample Network Parameter File to select the alternate file system and, if necessary, link the file system modules with the sample.

> Note
>
> The KwikNet FTP Sample Program for a particular target processor family is provided ready for use on one of the development boards used at KADAK for testing.

FTP Sample Program Directories

When KwikNet and its FTP Option are installed, the following subdirectories on which the sample program construction process depends are created within directory *KNTnnn*.

TCPIP	KwikNet header and source files, Ethernet Network Driver Ethernet and Serial Loopback Drivers
CFGBLDW	KwikNet Configuration Builder; template files
ERR	Construction error summary
MAKE	KwikNet Library make directory
TOOLXXX	Toolset specific files
TOOLXXX\DRIVERS	KwikNet device drivers and board driver
TOOLXXX\LIB	Toolset specific KwikNet Library will be built here
TOOLXXX\SAM_MAKE	Sample program make directory
TOOLXXX\SAM_FTP	KwikNet FTP Sample Program directory
TOOLXXX\SAM_COMN	Common sample program source files

One or more toolset specific directories *TOOLXXX* will be present. There will be one such directory for each of the software development toolsets which KADAK supports. Each toolset vendor is identified by a unique two or three character mnemonic, *XXX*. The mnemonic *UU* identifies the toolset vendor used with the KwikNet Porting Kit.

FTP Sample Program Files

To build the KwikNet FTP Sample Program using make file *KNFTPSAM.MAK*, each of the following source files must be present in the indicated destination directory.

Source File	Destination Directory	File Purpose
.	*CFGBLDW*	KwikNet Configuration Builder; template files
	KwikNet source directories containing:	
KN_API.H	*TCPIP*	KwikNet Application Interface definitions
KN_OSIF.H	*TCPIP*	KwikNet OS Interface definitions
KN_FILES.H	*TCPIP*	KwikNet Universal File System definitions
KNFSUSER.H	*TCPIP*	KwikNet User File System definitions
KN_SOCK.H	*TCPIP*	KwikNet Socket Interface definitions
	Toolset root directory containing:	
KN_OSIF.INC	*TOOLXXX*	OS Interface Make Specification
KNZZZCC.INC	*TOOLXXX*	Tailoring File (for use with make utility)
KNZZZCC.H	*TOOLXXX*	Compiler Configuration Header File
	KwikNet FTP Sample Program directory containing:	
KNFTPSAM.MAK	*TOOLXXX\SAM_FTP*	FTP Sample Program make file
KNFTPSAM.C	*TOOLXXX\SAM_FTP*	FTP Sample Program
KNZZZAPP.H	*TOOLXXX\SAM_FTP*	FTP Sample Program Application Header
KNFTPLIB.UP	*TOOLXXX\SAM_FTP*	Network Parameter File
KNFTPSAM.LKS	*TOOLXXX\SAM_FTP*	Link Specification File (toolset dependent) Other toolset dependent files may be present.
KNFTPSCF.UP	*TOOLXXX\SAM_FTP*	User Parameter File (for use with AMX)
KNFTPTCF.UP	*TOOLXXX\SAM_FTP*	Target Parameter File (for use with AMX)
	Common sample program source files:	
KNSAMOS.C	*TOOLXXX\SAM_COMN*	Application OS Interface
KNSAMOS.H	*TOOLXXX\SAM_COMN*	Application OS Interface header file
KNRECORD.C	*TOOLXXX\SAM_COMN*	Message recording services
KNCONSOL.C	*TOOLXXX\SAM_COMN*	Console driver
KNCONSOL.H	*TOOLXXX\SAM_COMN*	Console driver header
		Console driver serial I/O support:
KN8250S.C	*TOOLXXX\SAM_COMN*	INS8250 (NS16550) UART driver
KN_BOARD.C	*TOOLXXX\DRIVERS*	Board driver for target hardware

FTP Sample Program Parameter File

The Network Parameter File *KNFTPLIB.UP* describes the KwikNet and FTP options and features illustrated by the sample program. This file is used to construct the KwikNet Library for the FTP Sample Program.

The Network Parameter File *KNFTPLIB.UP* also describes the network interfaces and the associated device drivers that the sample program needs to operate.

FTP Sample Program KwikNet Library

Before you can construct the KwikNet FTP Sample Program, you must first build the associated KwikNet Library.

Use the KwikNet Configuration Builder to edit the sample program Network Parameter File *KNFTPLIB.UP*. Use the Configuration Builder to generate the Network Library Make File *KNFTPLIB.MAK*.

Look for any KwikNet Library Header File *KN_LIB.H* in your toolset library directory *TOOLXXX\LIB*. If the file exists, delete it to ensure that the KwikNet Library is rebuilt to match the needs of the FTP Sample Program.

Then copy files *KNFTPLIB.UP* and *KNFTPLIB.MAK* into the *MAKE* directory in the KwikNet installation directory *KNTnnn*. Use the Microsoft make utility and your C compiler and librarian to generate the KwikNet Library. Follow the guidelines presented in Chapter 3.2 of the KwikNet TCP/IP Stack User's Guide.

> Note
>
> The KwikNet Library must be built before the FTP Sample Program can be made. If file *KN_LIB.H* exists in your toolset library directory *TOOLXXX\LIB*, delete it to force the make process to rebuild the KwikNet Library.

The FTP Sample Program Make Process

Each KwikNet sample program must be constructed from within its own directory in the KwikNet toolset directory. Hence, the KwikNet FTP Sample Program must be built in directory *TOOLXXX\SAM_FTP*.

All of the compilers and librarians used at KADAK were tested on a Windows® workstation running Windows NT, 2000 and XP. However, you can build each KwikNet sample program using any recent version of Windows, provided that your software development tools operate on that platform.

To create the KwikNet FTP Sample Program, proceed as follows. From the Windows Start menu, choose the MS-DOS Command Prompt from the Programs folder. Make the KwikNet toolset *TOOLXXX\SAM_FTP* directory the current directory.

To use Microsoft's *NMAKE* utility, issue the following command.

```
NMAKE -fKNFTPSAM.MAK "TOOLSET=XXX" "TRKPATH=treckpath"
    "OSPATH=yourospath" "TPATH=toolpath" "FSPATH=afspath"
```

The make symbol *TOOLSET* is defined to be the toolset mnemonic *XXX* used by KADAK to identify the software tools which you are using.

The symbol *TRKPATH* is defined to be the string *treckpath*, the full path (or the path relative to directory *TOOLXXX\SAM_FTP*) to your Turbo Treck TCP/IP installation directory.

The symbol *OSPATH* is defined to be the string *yourospath*, the full path (or the path relative to directory *TOOLXXX\SAM_FTP*) to the directory containing your RT/OS components (header files, libraries and/or object modules). When using AMX, string *yourospath* is the path to your AMX installation directory.

The symbol *TPATH* is defined to be the string *toolpath*, the full path to the directory in which your software development tools have been installed. For some toolsets, *TPATH* is not required. The symbol is only required if it is referenced in file *KNZZZCC.INC*.

If you are using KwikNet with AMX and the AMX/FS File System, you must define the symbol *FSPATH* to be the string *afspath*, the full path (or the path relative to directory *TOOLXXX\SAM_FTP*) to the directory in which you installed AMX/FS. If you are not using the AMX/FS File System, omit the definition of symbol *FSPATH*.

The KwikNet FTP Sample Program load module *KNFTPSAM.xxx* is created in toolset directory *TOOLXXX\SAM_FTP*. The file extension of the load module will be dictated by the toolset you are using. The extension, such as *OMF*, *ABS*, *EXE*, *EXP* or *HEX*, will match the definition of macro *XEXT* in the tailoring file.

The final step is to use your debugger to load and execute the KwikNet FTP Sample Program load module *KNFTPSAM.xxx*.

2.3 Adding FTP to Your Application

Before you can add the FTP protocol to your application, there are a number of prerequisites which your application must include. You must have a working KwikNet TCP/IP stack operating with your RT/OS and a file system. It is imperative that you start with a tested TCP/IP stack and file system with functioning device drivers before you add FTP. If these components are not operational, the KwikNet FTP option cannot operate correctly.

KwikNet Library

Begin by deciding whether you need an FTP client or server or both. Rarely are both required.

Read Appendix C of the KwikNet TCP/IP Stack User's Guide to see how the KwikNet file system interface will have to be configured to operate with the file system you have chosen.

If you are incorporating an FTP server, read Appendix D of the KwikNet TCP/IP Stack User's Guide to see how the KwikNet Administration interface must be adapted to define the users (clients) which your FTP server will be able to serve. Edit the administration file *KN_ADMIN.C* to include a user definition for each potential FTP client.

Use the KwikNet Configuration Manager to edit your application's KwikNet Network Parameter File to support a file system with administration services and to include an FTP client and/or server. Armed with your modified files (if any), rebuild your KwikNet Library. The library extension may be *.A* or *.LIB* or some other extension dictated by the toolset which you are using.

Memory Allocation

Memory is allocated for each client session initiated by your FTP client. Memory is also allocated for each client session managed by your FTP server. The FTP server requires one socket for listening for requests from potential clients. Each client session, whether for a client task or server task, will require two sockets: one for control (commands) and one for data transfer.

To meet these requirements, you may have to edit your KwikNet Network Parameter File to increase the memory available for allocation.

FTP Client and Server Tasks

You must provide one task for each FTP client and server which you wish to incorporate into your application. Usually one FTP client task or one FTP server task is required. Rarely are both needed. Even more rarely are two or more clients required.

In a multitasking system, you may have to increase the total number of tasks allowed by your RTOS in order to add the FTP tasks.

A stack size of 4K to 8K bytes is considered adequate for most FTP client or server tasks when used with most file systems and device drivers. The stack size can be trimmed after your FTP tasks have been tested and actual stack usage observed using your debugger.

In a multitasking system, all FTP tasks must be of lower execution priority than the KwikNet Task. If both FTP server and client tasks exist, it is usual to make the FTP server task of higher priority than FTP client tasks.

If you are incorporating an FTP client, then you have a significant coding responsibility. You must create an FTP client task procedure which performs the FTP operations required by your application. Only you can define such a procedure. All of the Treck FTP client services listed in Chapter 6 of the Treck TCP/IP User Guide are at your disposal. You can use the FTP client task in the FTP Sample Program as a guideline for proper form.

If you are incorporating an FTP server, then you need only create an FTP server task procedure which calls procedure *`tfFtpdUserStart()`* to begin operation as described in Chapter 1.6

The FTP client and server task C source modules must be compiled just like any other KwikNet application module. The compilation procedure is described in Chapter 3.4 of the KwikNet TCP/IP Stack User's Guide.

Reconstructing Your KwikNet Application

Since you are adding FTP to an existing KwikNet application, there is little to be done.

To meet the memory demands of your FTP client and server, you may have to edit your KwikNet Network Parameter File to increase the memory available for allocation. If you do so, you must then rebuild your KwikNet Library.

Your application link and/or locate specification file must include the KwikNet Library which you built with support for FTP. The object modules for your FTP client and server tasks and any support modules that they might require must also be included in your link specification together with your other application object modules.

With these changes in place, you can link and create an updated KwikNet application with FTP support included.

AMX Considerations

When reconstructing a KwikNet application which uses the AMX Real-Time Multitasking Kernel, adapt the procedure just described to include the following considerations.

You may have to edit your AMX User Parameter File to increase the maximum number of tasks allowed in order to add FTP client and server tasks.

FTP client and server tasks can be predefined in your AMX User Parameter File or they can be created dynamically at run-time as is done in the KwikNet FTP Sample Program. These are simple AMX trigger tasks without message queues.

A stack size of 4K to 8K bytes is considered adequate for use with most file systems and device drivers. The stack size can be trimmed after your FTP tasks have been tested and actual stack usage observed using your debugger.

The FTP task priorities must be lower than that of the KwikNet Task. If both FTP server and client tasks exist, it is usual to make the FTP server task of higher priority than FTP client tasks. If you are using AMX 86 to access MS-DOS® file services, the PC Supervisor Task should be below all FTP client and server tasks in priority.

If you edit your AMX User Parameter File, you must then rebuild and compile your AMX System Configuration Module. If you are using the AMX/FS File System, you should also rebuild and compile your AMX/FS File System Configuration Module.

No changes to your AMX Target Configuration Module are required to support FTP unless your FTP client task requires special device support which is not already part of your application.

Performance Considerations

A meaningful discussion of all of the issues which affect the performance of an FTP server or client are beyond the scope of this document. Factors affecting the performance of the KwikNet FTP client and server include the following:

processor speed
memory access speed and caching effects
file system performance and disk access times
competing disk accesses for different users
network type (Ethernet, SLIP, PPP)
network device driver implementation (buffering, polling, DMA support, etc.)
TCP protocol effects (window size adaptations)
IP packet fragmentation
network hops required for connection
operation of the remote (foreign) connected client or server
KwikNet TCP/IP Stack configuration (clock, memory availability, sockets, etc.)

Of all these factors, only the last one can be easily adjusted. Increasing the fundamental clock rate for the KwikNet TCP/IP Stack beyond 50Hz will have little effect and will adversely affect systems with slow processors or memory. Increasing the memory available for use by the TCP/IP stack will help if high speed Ethernet devices are in use and the processor is fast enough to keep up.