

# **iChip FTP Client**

## **Theory of Operation**

**Version 1.32**

**November 2003**

# Introduction

The FTP protocol is described in RFC 959.

## General

FTP (File Transfer Protocol) is defined as a protocol for file transfer between hosts over a network. The primary function of FTP is transferring files efficiently and reliably among hosts and allowing the convenient use of remote file storage capabilities. FTP, though usable directly by a user at a terminal, was designed mainly for use by programs.

What is FTP and what can it do for me?

A primary function of the Internet is to enable users to move files from one computer to another computer across the Internet. Using File Transfer Protocol (FTP), it is possible to move large files of any sort (i.e., video clips, large documents, audio or multi-media clips) between computers. This functionality is provided using the client/server model. An FTP server is mounted on one computer. This server acts as the repository of files.

Users access the FTP server using an FTP client. The FTP client allows users to place files on the server and to grab copies of these files from the server.

With the ability to add files to an FTP server, it is possible to make large files available for downloading by specific hosts (by password protecting the FTP site), or any host using an FTP client (via "anonymous" FTP - no password required).

Some FTP servers also allow performing file management activities, like creating new directories, deleting or changing the names of existing files, and so on.

Finally, nearly all Web servers provide FTP functionality as one component of the software package. This added functionality allows creating Web accessible documents, images, or other file types and placing them "on the Web " by FTPing the files from a computer other than the one running the Web server. In addition, entire Web sites containing tens or hundreds of documents can be maintained from a distance using this FTP functionality.

## FTP Login

The FTP server protocol incorporates authentication level security. That is, it requires a user name and password to allow access to its file system. The protocol may also further restrict access by requesting an account name. However, FTP servers may be configured for anonymous FTP, allowing guest access, usually enforcing access restrictions to guests. This means that a remote host can access a machine without having to have an account, user name and password on that machine (i.e. the host does not be an official user of the system). Typically user name "anonymous" or "ftp" and an email address as the password will have to be sent to the server for anonymous access. These anonymous FTP servers contain software, documents of various sorts, files for configuring networks, graphic images, and all sorts of other information.

## The FTP Client Component in iChip Family

The FTP client component in iChip Family extends the iChip's general-purpose sockets to incorporate an additional, dedicated socket for FTP activities. iChip implements the client side only.

iChip FTP capabilities support transferring data to and from files on a remote FTP server. iChip supports AT+i commands to log into an FTP server, navigate within the existing file system on the server, create new directories, retrieve lists of the directories and files in an existing directory, create and transfer data to files, retrieve data from files and delete files.

As in all other iChip protocol implementations, host involvement in the specifics of FTP is kept minimal. iChip needs to deal with non-standard FTP issues, such as possible differences between FTP server responses, on its own. Multi-stage FTP protocol sequences are atomized under iChip control to minimize complexity and need for host processor intervention.

The FTP client component in iChip Family can be used by the device connected to iChip for exchanging information between the device and remote computers or devices. Using the FTP client component for storing information on a remote file system managed by an FTP server will allow sharing the information with other computers or devices on the Internet that also have access to an FTP client. It is also possible to use the FTP client in order to retrieve data or instructions that were stored on the remote system by other computers or devices.

iChip does not include the FTPserver protocol, because it makes little sense for a remote device to act as an FTP server for the following reasons:

- a. An FTP server manages a local file system, providing clients with remote access to the file system – devices usually do not have local file systems.
- b. A server is a passive device, which lingers on the Internet, waiting for clients to approach it with service requests. Therefore, it should be always online and have a fixed IP address. Usually it will also have a registered domain name. It makes little sense for every device to have a fixed IP, domain name and always be online.

The iChip device, specifically the iChip-S, which connects to the Internet over a dialup or wireless modem connection, is normally offline. Moreover, once it goes online, by dialing a local ISP, it acquires a dynamic IP address, which is different in each session. An iChip-L operating in a LAN environment is always online, but it might still be assigned with a different IP address every time it boots by a DHCP server. Moreover, an IP address in a LAN environment might not be reachable from outside of the local network due to mechanism like NAT (Network Address Translation) or firewalls used inside the LAN. Therefore, a central management system cannot know how to approach the specific device over the Internet, even if the device is online. From this brief analysis, it stands to reason to have the device act as a client, while the central system acts as a server, has a fixed IP, domain name and is always online. In the case of FTP, it also makes for sense to leave the local file system management in the hands of the central system, rather than in each individual device.

## **FTP Data and Command Sockets**

The FTP command socket is normally on port 21 (decimal) of an FTP server. However, other ports may be specified to support special cases. The command socket is used for transferring commands and replies between the FTP server and client.

An additional socket is used when and if data needs to be transferred between the server and client. The client always initiates the creation of the additional socket. The client can choose to work in active mode, in which case it will open the additional socket to the server, or passive mode, in which case it will ask the server to open the additional socket to a passive port opened on the client.

By default, iChip FTP client chooses the active mode, but it will use the passive mode as backup if the attempt to create the data socket in active mode will fail. This procedure is transparent to the host and cannot be changed by the host, but it will automatically assure that the FTP client will work even if only one mode is possible with the specific server or specific network configuration. The active mode was chosen as default because it can work in most network configurations and will usually only fail if the FTP server was specifically set to not allow this mode. The passive mode might fail in some network configurations that use local IP addresses and a NAT server or implement security methods like firewalls, though even in those cases it is usually possible to configure those tools to allow an FTP connection.

The FTP commands supported in iChip are:

- Open FTP link to FTP Server (AT+IFOPN)
- Retrieve File List from Server (AT+IFDL / AT+IFDNL)
- Change Directory on Server (AT+IFCWD)
- Retrieve File Contents from Server (AT+IFRCV)
- Create a New Directory on Server (AT+IFMKD)
- Open a New File on Server (AT+IFSTO)
- Open an existing File on Server for Append (AT+IFAPN)
- Send Binary Data to an open File on Server (AT+IFSND)
- Close a File on Server After Binary Data Send (AT+IFCLF)
- Delete File on Server (AT+IFDEL)
- Close FTP Session (AT+IFCLS)

## **iChip FTP Client Operation Mode**

FTP specifies several operational modes. The RFC calls for a minimum implementation, which should be observed by all FTP servers. The iChip Family restricts its operation mode to the minimum implementation to assure best intersystem compatibility.

Character Types:	ASCII Non-print
Structure:	File
Mode:	Stream

## FTP Error Management

505	<i>Cannot open additional FTP session – all FTP handles in use</i>
506	<i>Not an FTP session handle</i>
507	<i>FTP server not found</i>
508	<i>Timeout when connecting to FTP server</i>
509	<i>Failed to login to FTP server (bad username or password or account)</i>
510	<i>FTP command could not be completed</i>
511	<i>FTP data socket could not be opened</i>
512	<i>Failed to send data on FTP data socket</i>
513	<i>FTP shutdown by remote server</i>

Possible error codes to be returned in response to:

### 1. AT+iFOPN:

505: Cannot open additional FTP session. An FTP session is already open on iChip. The open FTP session must be closed before another session can be opened.

507: FTP server not found. Possible causes: The FTP server name is a symbolic name but the iChip DNS settings are bad or the defined DNS servers are down and the FTP server IP address cannot be resolved.

508: Timeout when connecting to FTP server. iChip attempted to but was not able to open a socket connection to the FTP server. Possible causes: The FTP server name/IP address is incorrect, such a server does not exist, iChip requires the use of a gateway to connect to the FTP server, but the gateway settings are incorrect or the gateway server is currently down.

509: Failed to login to FTP server. The supplied user name and password combination do not consist of a valid account on this FTP server. This FTP server may require an account name, but it was not given in the command. Alternatively, an account was given, but it does not match the supplied user name and password. In an anonymous FTP, with user name “anonymous”, some FTP servers expect the password to be a reachable Email address.

### 2. AT+iFRCV / FDL / FDNL / FSTO / FAPN

511: FTP data socket could not be opened. The FTP command requires opening an additional socket for sending or receiving data, but such a socket could not be opened. Possible reasons: The FTP server is down, iChip requires the use of a gateway to connect to the FTP server, but the gateway server is currently down, or the FTP server is busy at the moment.

### 3. AT+iFSND

512: Failed to send data on FTP data socket. The server closed the data socket or connection between iChip and the FTP server was lost.

### 4. AT+iFOPN / FDL / FDNL / FCWD / FRCV / FMKD / FSTO / FAPN / FSND / FCLF / FDEL / FCLS (Any FTP command)

513: FTP session shut down by remote server. Possible causes: FTP server inactivity timeout, or internal error on the FTP server.

5. AT+IFDL/ FDNL / FCWD / FRCV / FMKD / FSTO / FAPN / FSND / FCLF / FDEL / FCLS (Any FTP command but FOPN)

506: Not an FTP session handle. Possible causes: Command included handle “0”, but no FTP session is open, or FTP session is open with handle “0”, but the command included a different handle number. Currently iChip allows only one FTP session at a time and so “0” is the only valid handle.

6. AT+IFDL/ FDNL / FCWD / FRCV / FMKD / FSTO / FAPN / FSND / FCLF / FDEL (Any FTP command but FOPN and FCLS)

510: FTP command could not be completed. Possible causes:

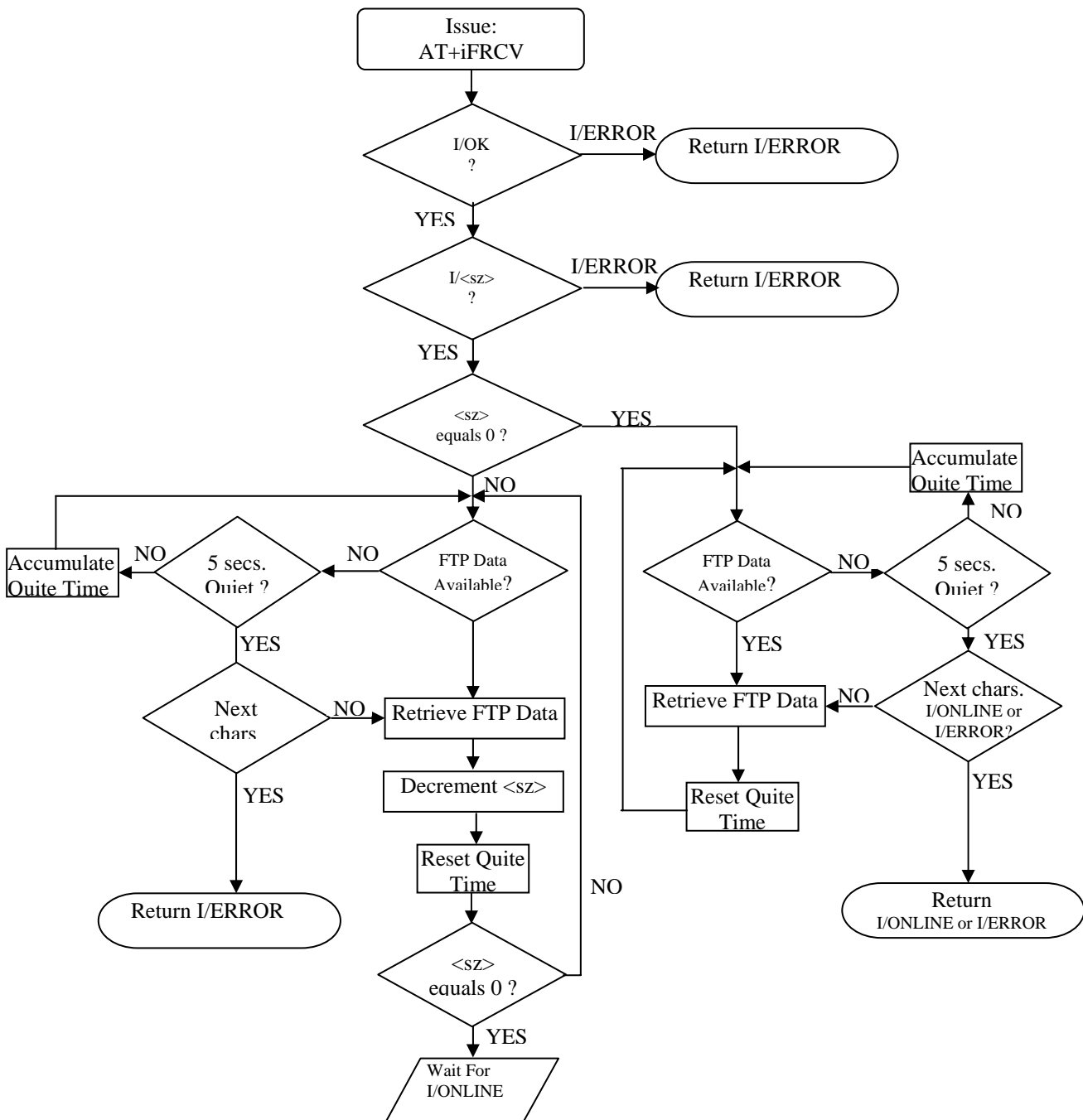
FDL/FDNL/FRCV/FDEL command included a file or directory name that does not exist, FMKD/FSTO/FAPN included a file or directory name that could not be created (for example name not valid on the FTP server’s file system or no available memory), or user logged in is not authorized to do the requested command on the FTP server.

## FTP Session Flow - Examples

1. *Generate a file on a remote server and store application data:*
  - Step 1: Open FTP server and Login (FOPN)
  - Step 2: Select the destination directory (FCWD) or create a new one (FMKD)
  - Step 3: Open a new destination file (FSTO) or an existing file when appending (FAPN)
  - Step 4: Store application data in file (FSND)
  - Step 5: Repeat Step 4 until no more data to send
  - Step 6: Close destination File (FCLF)
  - Step 7: Go to step 3 if more files need to be stored in same directory or to step 2 if a new directory needs to be selected or created first
  - Step 8: When done, close the FTP session (FCLS)
  
2. *Retrieve a file from a remote server:*
  - Step 1: Open FTP server and Login (FOPN)
  - Step 2: Select the destination directory (FCWD)
  - Step 3: Retrieve a list of the files in the destination directory (FDNL or FDL for a list that contains more detailed about each file)
  - Step 4: Choose the file you need from the list and retrieve it (FRCV) – see FTP Receive Flow
  - Step 5: Delete the file if there is no more use for it (FDEL)
  - Step 6: Go to step 3 if more files need to be retrieved in same directory or to step 2 if a new directory needs to be selected first
  - Step 7: When done, close the FTP session (FCLS)

## FTP Receive Flow

An AT+iFRCV command will retrieve the contents of a file. If the FTP server reports the size of the file when the data transfer starts, it will be reported to the host via the I/<sz> response and the host can expect the file to end after exactly /<sz> bytes of data. However, some servers will not report the size of the file. In those cases, the data transfer will start with I/0, but the host can still scan the incoming data for the terminating I/ONLINE message to determine where the file ends. In order to make the process simpler and not force the host to scan all the data, the following mechanism was added: If the reported size is 0, then after the last byte of incoming data from the server, iChip will not send any data to the host for 5 seconds and only then the terminating I/ONLINE message will appear. This allows the host to not scan the data until 5 seconds of silence were counted.



## Example of an FTP Session

Below is an example of an FTP session. In this example, the file system on FTP server ("[ftp.devices.com](http://ftp.devices.com)") contains the following directories –

- \deviceDir – a directory dedicated to the specific device.
- \deviceDir\Upload – a sub directory used to store data generated by the device.
- \deviceDir\Download - a sub directory used to store data to be read by the device.

The server allows write access to the upload directory and read access to the download directory to the user name used by the device ("myuser"). The device uploads the data records it generates to text files. In this example "1.txt" is the name of the first data record, "2.txt" can be the name of the second and so on. Alternatively, the current date and time can be used to create unique names for files. Another option could be to append the data records into one file (using the AT+IFAPN command).

When new instructions or data need to be delivered to the device, a file called "input.txt" containing the instructions or data will be placed in the download directory (by a human or machine manager with direct or FTP write access to the download sub directory on the deviceDir directory on the FTP server).

As soon as the device is done reporting the data records, it changes the working directory to the download directory and scans it for the existence of the "input.txt" file. If such a file exists, it will read its contents and delete it from the directory (so it will not find the same data again). In this example, the FTP session is then closed. The whole procedure can be repeated on a periodic basis or as a result of a trigger or an event. Alternatively, the FTP session can be left open (skip step 5) and then the next time start from step 2 instead of from step 1.

AT+ifopn:ftp.devices.com:myuser,mypass <b>I/000</b>	<i>Step 1: Open an FTP session to the remote server and authenticate</i> open a session and authenticate
AT+ifcwd:0,deviceDir <b>I/OK</b>	<i>Step 2: Navigate to the device's directory on the server</i> navigate to the destination directory
AT+ifcwd:0,"upload" <b>I/OK</b>	<i>Step 3: Store data records generated by the device on the FTP server</i> navigate to the upload directory
AT+ifsto:0,"1.txt" <b>I/OK</b> AT+ifsnd:0,17:data record no 1 <b>I/OK</b> AT+ifclf:0 <b>I/OK</b>	open a file in the uploads directory store one data record in the file close the file
AT+ifcwd:0,"../download" <b>I/OK</b> AT+ifdnl:0 <b>input.txt</b> <b>I/ONLINE</b>	<i>Step 4: Read new values for operational parameters of the device from a file</i> navigate to the downloads directory look for an input file in the downloads directory
AT+ifrcv:0,input.txt <b>I/OK</b> <b>I/46</b> <b>Set Parameter1 0</b> <b>Set Parameter2 "testvalue"</b> <b>I/ONLINE</b> AT+ifdel:0,input.txt <b>I/OK</b>	read values from the input file (when exists) if an input file was read - delete the input file
AT+ifcls:0 <b>I/ONLINE</b>	<i>Step 5: Close the FTP session</i> close the session