# Technical Note TN2023

## Open Firmware Ethernet Debugging II: Telnet downloading

**CONTENTS**


One machine Open Firmware debugging vs. two machine debugging

The tools needed for two machine Ethernet debugging

Determining Ethernet readiness

Using the correct Ethernet connection

Setting up your machines

Setting up tftpd under MacOS X

Setting up tftpd under MacOS 9

Transferring a code file to the target machine

Summary

References

Downloadables


This Technote describes a method of debugging your device's firmware using Ethernet to connect a host machine and target machine. Ethernet and Telnet protocols are used in this method to edit, debug and run Forth code files.

This note is directed at any engineer or scientist who understands how to communicate at a local bus level with a PCI peripheral. This may include PCI device designers, driver writers, diagnostic writers, or other developers that may find this technical note interesting.

Note that the Ethernet method of debugging, described below, uses shareware tools. These tools are not provided or supported by Apple Computer and may require a user fee by the author.

**Updated: [October 22 2001]**

---

## Why use two machines instead of one?

**One machine Mode:**

As of the writing of this document, the majority of Apple systems that use Open Firmware enter the Open Firmware mode in "one machine mode". "One machine mode" refers to an Open Firmware environment where editing and debugging are accomplished on the same machine that is using Open Firmware.

"One machine mode" limits the user to a command-line interface where text commands (words) are entered from the system's keyboard and viewed on the system's display. Information on the screen is usually lost once it has scrolled off the screen.

Open Firmware is easy to use with one machine mode: use the Open Firmware user interface. That's great if you just want to look at the device tree, add a property to the node you may be debugging, or are going to be typing in small bits of OF code.

But that one machine mode only has a small I/O buffer. More than a page full of text scrolls off the top of the screen, and what you typed in, as well as the resulting output, is lost forever. If you are doing a great deal of interactive work in the Open Firmware UI, this will quickly become annoying.

In addition to scrolling issues, testing and debugging code in one machine mode environment can become troublesome and tedious since most work is lost between restarts. Developing code in one machine mode usually requires the user to enter all development code from scratch or use a nontrivial method to save code on the machine's mass storage device. More complex work, such as debugging your device's expansion ROM Forth code, routinely requires you to save and review your work.

**Two machine Mode:**

"Two machine mode" refers to an Open Firmware environment where editing and debugging of a computer are accomplished on a second machine that acts as a terminal and large buffer. Two machine mode is almost always preferred to one machine mode because it allows you to easily save your work.

With only one machine, your text is lost between restarts, which can occur frequently during debugging. Having the ability to quickly save your work on a second (less volatile machine) ensures that two machines will reduce your debug time.

If you are doing more complex work on just one computer, such as debugging your device's expansion ROM Forth code, large amounts of work can be lost if the machine needs to be restarted.

Two machine mode originally involved connecting the second machine to the Open Firmware target machine via a serial cable connected to both machines' serial ports. Since the iMac, traditional serial ports have been removed as I/O ports, making it difficult to use two machine mode to download your saved Forth code files.

This document will describe a method of connecting two machines, through the Ethernet ports, to regain the benefits of transferring your Forth code files between the machines for development and debugging.

Back to top

## The tools needed for two machine Ethernet debugging

The tools needed for two machine Ethernet debugging are as follows:

1. An Open Firmware, **target** machine that will also be the **client** machine for TFTP transfers in this technote.

   This is the machine that will be used as the target for development and where the Open Firmware code will be tested. This machine will be the machine in Open Firmware. Note that this machine must have "Ethernet or Telnet" readiness built into Open Firmware. This will be discussed in the next section, Determining Ethernet readiness.

2. A (server) **host** machine with an Ethernet port.

   This machine can be any machine with an Ethernet port capable running a host TFTP server application. This machine will be used to develop and edit the Open Firmware code and communicate with the machine in Open Firmware to download the edited code file.

   Use a normal Ethernet cable when connecting via a hub and an Ethernet crossover cable when connecting the machines directly.

3. An Ethernet cable (or cables) to connect the two machines.

   This is either a "straight through" or "crossover" Ethernet cable with RJ-45 connectors at each end. The type of cable depends on the kind of Ethernet network you are using.

4. A host TFTP server application.

   This is the application that will run on the host machine and will act as a "file transfer" server for the Open Firmware (target) machine. The TFTP server application used in the examples in this Technote is "TFTP Daemon" for the Macintosh for MacOS 9.

   Mac OS X has a built in UNIX based TFTP application and is explained in the Setting up tftpd under MacOS X section below.

   - TFTP Daemon for Mac OS 9, can be found at: TFTP Daemon
   - TFTP Daemon is a "shareware" application and the author requires a registration fee.

3. An application or process to "quit" the TFTP server.

   This is the application that will list and "kill" processes (i.e.. TFTP Daemon) running on the host machine. In our case, it will allow you to quit or "kill" the TFTP application.

   ProcessWatcher is a "freeware" application that accomplishes this task and can be found at: ProcessWatcher

4. A functional Ethernet network

   A functional Ethernet network can be anything from a Local Area Network, DSL, T1, etc. connection to a single crossover cable connection between two machines.

5. A valid IP address for the target (client) machine

   This is required for the target machine in Open Firmware.

6. A valid IP address for the host (server) machine.

   This is required for the host machine where the FORTH files are being developed.

7. A valid router IP address.

   This is required to communicate with the target machine.

Back to top

## Determining Ethernet readiness

If you have already been using the serial interface to debug your code, then using Ethernet is almost the same set of procedures. Ethernet requires an Ethernet port instead of a serial port and of course has a different initialization sequence.

The target machine needs to have an Ethernet / telnet client implemented in Open Firmware. You must first determine if your target machine is Ethernet or telnet ready. The first Macintoshes that are Ethernet capable are some, not all, G4 machines. G4 machines that are not Ethernet or telnet ready can and this functionality via a firmware update for that machine.

You must first determine if your G4 has a Ethernet or telnet Open firmware **package**. Here are the steps to accomplish that.

At the Open Firmware user interface, go to the packages node in the device tree. The packages node is a support node that can be used by PCI device nodes. In earlier G4 machines the `/packages/`**`Ethernet`** node is used to support the connection between the two machines. If your machine has the /packages/Ethernet node then you can debug over Ethernet. As soon as you enter Open Firmware, at the prompt, type " `dev /packages ls` ". You should then be presented with a list that looks similar to the example below.

```
0 > dev /packages Ls
       ff83ebf8: /deblocker
       ff83f518: /disk-label
       ff83ff18: /obp-tftp
       ff847048: /Ethernet        <-----look for this node----
       ff8478c8: /mac-parts
       ff8489a0: /mac-files
       ff84b7b0: /hfs-plus-files
       ff850520: /fat-files
       ff852250: /iso-9660-files
       ff852e58: /bootinfo-loader
       ff854af8: /xcoff-loader
       ff855510: /pe-loader
       ff855ee8: /elf-loader
       ff857518: /usb-hid-class
       ff859858: /usb-ms-class
       ff85bbf8: /sbp2-disk
       ff85e1c0: /ata-disk
       ff85f420: /atapi-disk
       ff860af8: /bootpath-search
       ff8673e8: /terminal-emulator
        ok
       0 >
```

In later G4 machines, the `/packages/`**`telnet`** node is used to support the connection between the two machines. If your machine has the `/packages/telnet` node then you can debug over Ethernet.

```
0 > dev /packages Ls
       ff83ebf8: /deblocker
       ff83f518: /disk-label
       ff83ff18: /obp-tftp
       ff847048: /telnet        <-----look for this node----
       ff8478c8: /Mac-parts
       ff8489a0: /mac-files
       ff84b7b0: /hfs-plus-files
       ff850520: /fat-files
       ff852250: /iso-9660-files
       ff852e58: /bootinfo-loader
       ff854af8: /xcoff-loader
       ff855510: /PE-loader
       ff855ee8: /elf-loader
       ff857518: /usb-hid-class
       ff859858: /usb-ms-class
       ff85bbf8: /sbp2-disk
       ff85e1c0: /ata-disk
       ff85f420: /atapi-disk
       ff860af8: /bootpath-search
       ff8673e8: /terminal-emulator
        OK
       0 >
```

Back to top

## Using the correct Ethernet connection

Occasionally, sending Open Firmware commands over an Ethernet network can be slowed down by Internet traffic and unwanted broadcast messages. To avoid Internet traffic and allow a point to point connection between two 10BaseT Ethernet devices without using a hub, you can use a crossover cable. Crossover cables can be either constructed or purchased.

Constructing a crossover cable is basically rerouting the transmit lines from one RJ45 Ethernet connector to the receiving lines of another RJ45 Ethernet connect. Information about constructing and connecting a crossover cable between two Macintosh computers is summarized in the Tech Info Library Article:
Macintosh: Transferring Information Between Computers Using Ethernet. Article ID: 43015

Other Web sites describing the construction of crossover cables can be found at the following URLs.

1. Ethernet Crossover Cable Pin Out
2. Ethernet Crossover Cable How To Guide
3. RJ45 Connector

Crossover cables can be purchased at almost any outlet that sells or supplies computers or computer peripherals. The following web site is one of those sites.

We Sell Cables

Back to top

## Setting up your machines

Before you establish communications between the host and target machines, you will need to create an Open Firmware code file on the host machine to be downloaded to the target machine. In both MacOS 9 and MacOS X this file must be in a specific location on your host machine for the TFTP application to transfer it.

Using a text editor on the Host machine, create, edit and save your Forth code file.

Below is a sample FORTH code file created on the host machine that will be transferred to the target machine.

The sample file below (**myfile**) contains some colon defined, formatted, commonly used, basic Open Firmware functions.

It first defines some "greeting strings" that are immediately presented to the user in Open Firmware that describe the newly defined words.

The new words include functions to display the current working device path, display the machine's device tree (complete with start and stop commands), select the root device, select the PCI bus for the PCI slots in the machine, display the property list for the current device node and a function to automatically update the Open Firmware code from the current version of "myfile" on the host machine.

**Example 1:** myfile (sample Open Firmware code file)

```
\ This is a file that contains some colon defined, formatted,
\ commonly used, basic Open Firmware functions.
\ Below, grtng1 through grtng7 are individual lines for the
\ "greeting" message documenting the words that can be used
\ in this file.

: grtng1 ( --) ." cmds? = display these words (commands) " cr ;

: grtng2 ( --) ." update = get new file update " cr ;

: grtng3 ( --) ." sl-root = select root (dev /) " cr ;

: grtng4 ( --) ." sl-pci = select PCI node (dev pci) & pwd " cr ;

: grtng5 ( --) ." my-pwd = formatted directory (pwd) " cr ;

: grtng6 ( --) ." dv-tree = key controlled dev tree list (Ls) " cr ;

: grtng7 ( --) ." prop-lst = formatted property list (.properties) " cr ;


\ hello is a simple and safe "hello" message for testing.

: hello ( -- ) cr ." Hello Open Firmware" ;

\ The next 6 words are the "feedback" lines (strings) for the actual
\ words that do work.


\ slctroot (select root) message.

: slctroot ( -- ) ." ... selecting 'root' " cr ;


\ slctpci (select pci) message

: slctpci ( -- ) ." ... selecting 'pci' " cr ;


\ slct-start (select start) scrolling message

: scrl-start ( -- ) cr ." press Control-Q to continue scrolling" ;


\ slct-stop (select stop) scrolling message.

: scrl-stop ( -- ) cr ." press Control-S to stop scrolling" ;


\ wt-4-key (wait for a key) message
```

```
: wt-4-key ( -- ) cr ." press space bar to continue ... " key clear ;


\ cmds? (commands?) formats and displays the entire greeting message.
\ cmds? can be called to display the words that the myfile provides.

: cmds? ( --) cr grtng1 grtng2 grtng3 grtng4 grtng5 grtng6 grtng7 ;



\ The first actual action after the file is loaded is to display
\ the words that can be used. This is also the first indication
\ that the file loaded.

cmds?   cr cr hello cr cr


\ update
\ Recreates, in a single word, the "boot" command that
\ downloaded myfile. It assumes that we know (in order)
\ the sever ip address, the file name (myfile), the client ip address,
\ the subnet mask and the router ip address.
\ Once the file is loaded, all that is needed to download the file
\ again, is to use the word update.

: update " boot enet:10.1.2.2,myfile,10.1.2.3;255.255.0.0,;10.1.2.2" evaluate ;


\ SL-root (select root) selects the root node of the device tree

: SL-root ( -- ) slctroot clear " /" find-device ;


\ my-pwd simply displays and puts a format around the
\ current working directory

: my-pwd ( -- ) cr cr ." " pwd ."  = current working directory" ;


\ sl-pci (select pci) assumes that the alias "pci" is defined.
\ if so, it selects the pci bus node.

: sl-pci ( -- ) slctpci clear " pci" find-device my-pwd cr cr ;


\ dv-tree (device tree) displays the device tree from the current
\ node. It will display a message and wait for a keypress before
\ displaying the device tree.

: dv-tree ( -- ) cr scrl-stop scrl-start cr cr wt-4-key cr cr Ls ;


\ prop-lst (property list) displays the properties of the current node.

: prop-lst ( -- ) cr cr ." *** Properties *** " cr Properties ;


\ end of file comment ... so I can locate the end!
```

**Example 1.** Text file example (myfile).

You will need to connect the host and target machine to a functional Ethernet network using the appropriate cables. As mentioned above, Internet traffic can complicate communications between the host and target machines and it is therefore

recommended to use a crossover cable between the two machines. The rest of this discussion assumes the use of the crossover cable.

Having obtained the 9 items in the "The tools needed for two machine Ethernet debugging" section above, take the following steps to establish communications between the host and target machines. These steps are somewhat different for MacOS X and MacOS 9. Step up for MacOS X is described in the section below and set for MacOS 9 is described in the section following the one below.

Back to top

### Setting up tftpd under MacOS X on the Host machine

Connect the host machine to the target machine using the crossover Ethernet cable.

The host machine will be running the Mac OS while the target machine will be running the Open Firmware user interface.

MacOS X is shipped with a UNIX TFTP Daemon (tftpd), however the tftpd application is disabled in OS X.
In order to enable and use the tftp application, the tftpd application must be found and the configuration file must be modified and saved.

The tftpd application is contained in the inetd.conf configuration file. In order to find the inetd.conf file you need to use the terminal application in OS X. The inetd.conf file contains set up services for the internet.
After opening terminal app., to locate the tftp application, at the UNIX cursor of the terminal application, type Ls -l / etc. This will give you the current directory. Note, Example 3 shows that the directory points us to the private directory for all of our applications.

```
[localhost:~] smartin% Ls -l /etc
lrwxrwxr-t  1 root  admin  11 Jun 25 14:33 /etc -> private/etc
```

**Example 3.** Determining the proper directory

To list the files in the current directory, type Ls -l <current directory>.
Example 4 shows a partial directory listing for the system in Example 3.

```
[localhost:~] smartin% Ls -l /private/etc
total 824
-rw-r--r--   1 root  wheel     515 Feb 25 00:05 afpovertcp.cfg
-rw-r--r--   1 root  wheel       6 Jun 25 14:33 appletalk.cfg
-rw-r--r--   1 root  wheel     256 Jun 25 14:34 appletalk.nvram.en0
-rw-r--r--   1 root  wheel     346 Feb 25 00:05 bootstrap.conf
-rw-r--r--   1 root  wheel     725 Feb 25 00:05 crontab
-rw-r--r--   1 root  wheel      31 Feb 16 19:48 csh.cshrc
-rw-r--r--   1 root  wheel      34 Feb 16 19:48 csh.login
-rw-r--r--   1 root  wheel      35 Feb 16 19:48 csh.logout
-rw-r--r--   1 root  wheel    3658 Feb 25 00:05 daily
-rw-r--r--   1 root  wheel       0 Mar 10 03:15 dumpdates
-rw-r--r--   1 root  wheel       0 Feb 25 00:05 find.codes
-r--r--r--   1 root  wheel     142 Feb 25 00:05 fstab.hd
-r--r--r--   1 root  wheel     142 Feb 25 00:05 fstab.rd
-r--r--r--   1 root  wheel     142 Feb 25 00:05 fstab.sd
-rw-r--r--   1 root  wheel     100 Feb 25 00:05 ftpusers
-rw-r--r--   1 root  wheel     576 Feb 17 17:44 gdb.conf
-rw-r--r--   1 root  wheel    5454 Feb 25 00:05 gettytab
-rw-r--r--   1 root  wheel     555 Feb 25 00:05 group
-rw-r--r--   1 root  wheel     495 Jun 25 14:33 hostconfig
-rw-r--r--   1 root  wheel     500 Dec 31  1969 hostconfig.old
-rw-r--r--   1 root  wheel     502 May 22 07:49 hostconfig~
-rw-r--r--   1 root  wheel     471 Feb 25 00:05 hosts
-rw-r--r--   1 root  wheel       0 Feb 25 00:05 hosts.equiv
```

```
-rw-r--r--   1 root  wheel       0 Feb 25 00:05 hosts.lpd
drwxr-xr-x  10 root  wheel     296 Feb 24 00:51 httpd
-rw-r--r--   1 root  wheel     973 Feb 25 00:05 iftab
-rw-r--r--   1 root  wheel    2543 Apr 10 14:17 inetd.conf
<------
-rw-------   1 root  wheel      12 Mar  6 18:39 kcpassword
-rw-r--r--   1 root  wheel       0 Feb 25 00:05 kern_loader.conf
-r--r--r--   1 root  wheel    1537 Feb 16 18:16 krb.conf
-r--r--r--   1 root  wheel     586 Feb 16 18:16 krb.realms
-r--r--r--   1 root  wheel    1947 Feb 16 18:16 krb5.conf
```

**Example 4.** Directory containing "inetd.conf"

Once the directory is listed locate the inetd.conf file. Note that this file is "owned" by root. In order to modify a root file the "sudo" command is used in Example 5. The next command "pico", invokes a built-in text editor so that the file can be modified. The last section describes the directory and file that is to be edited.

```
[localhost:~] smartin% sudo pico /etc/inetd.conf
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these two things:
        #1) Respect the privacy of others.
        #2) Think before you type.
Password:
```

**Example 5.** Using sudo to become"root"

Enter the root password for your machine. The terminal will run the PICO text editor and open the inetd.conf file as in Example 6.

```
  UW PICO(tm) 2.3               File: /etc/inetd.conf

#
# Internet server configuration database
#
#       @(#)inetd.conf  5.4 (Berkeley) 6/30/90
#
# Items with double hashes in front (##) are not yet implemented in the OS.
#
#finger stream  tcp     nowait  nobody  /usr/libexec/tcpd              fingerd$
ftp     stream  tcp     nowait  root    /usr/libexec/tcpd              ftpd -l
#login  stream  tcp     nowait  root    /usr/libexec/tcpd              rlogind
#nntp   stream  tcp     nowait  usenet  /usr/libexec/tcpd              nntpd
#ntalk  dgram   udp     wait    root    /usr/libexec/tcpd              ntalkd
#shell  stream  tcp     nowait  root    /usr/libexec/tcpd              rshd
#telnet stream  tcp     nowait  root    /usr/libexec/tcpd              telnetd
#uucpd  stream  tcp     nowait  root    /usr/libexec/tcpd              uucpd
#comsat dgram   udp     wait    root    /usr/libexec/tcpd              comsat
#tftp   dgram   udp     wait    nobody  /usr/libexec/tcpd              tftpd /$
#bootp  dgram   udp     wait    root    /usr/libexec/tcpd              bootpd
##pop3  stream  tcp     nowait  root    /usr/libexec/tcpd              /usr/lo$
                              [ Read 57 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Pg   ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where is  ^V Next Pg   ^U UnCut Text^D Del Char
```

**Example 6.** "inetd.conf" opened with the PICO text editor

Note that the tftp file is third from the bottom and is disabled by the '#' comment character.

The next step is to enable the tftp application.

Enabling the tftp application and tftpd, within PICO becomes a simple matter of using the arrow keys on the keyboard to move the prompt to the "#" character (in front of tftp), deleting it, exiting PICO and restarting your machine. However, careful attention should be paid to the path for the tftpd file. While in PICO this path is defined at the far right in the line that contains the tftp listing.

In Example 6 the path to your file appears as tftpd /$. The "$" indicates that there is more text beyond and can be viewed by using the arrow keys to move to the right of the "$". In this example the path was changed from the default to " tftpd /Users/Shared". This makes it a simple task to drop the Open firmware text file into the "Shared" folder on the OS X volume. Example 7. shows the modified form of the file in Example 6.

```
   UW PICO(TM) 2.3                  File: /etc/inetd.conf
Modified
#
# Internet server configuration database
#
#       @(#)inetd.conf  5.4 (Berkeley) 6/30/90
#
# Items with double hashes in front (##) are not yet implemented in the OS.
#
#Finger stream  tcp  nowait  nobody /usr/libexec/tcpd  fingerd -s
ftp     stream  tcp  nowait  root   /usr/libexec/tcpd  ftpd -l
#login  stream  tcp  nowait  root   /usr/libexec/tcpd  rlogind
#nntp   stream  tcp  nowait  usenet /usr/libexec/tcpd  nntpd
#ntalk  dgram   udp  wait    root   /usr/libexec/tcpd  ntalkd
#shell  stream  tcp  nowait  root   /usr/libexec/tcpd  rshd
#telnet stream  tcp  nowait  root   /usr/libexec/tcpd  telnetd
#uucpd  stream  tcp  nowait  root   /usr/libexec/tcpd  uucpd
#comsat dgram   udp  wait    root   /usr/libexec/tcpd  comsat
tftp    dgram   udp  wait    nobody /usr/libexec/tcpd  tftpd /Users/Shared
#bootp  dgram   udp  wait    root   /usr/libexec/tcpd  bootpd
##pop3  stream  tcp  nowait  root   /usr/libexec/tcpd  /usr/local/libexec/popper
##imap4 stream  tcp  nowait  root   /usr/libexec/tcpd  /usr/local/libexec/imapd
#
# "Small servers" -- used to be standard on, but we're more conservative
# about things due to Internet security concerns.  Only turn on what you
# need.
                                                [ Wrote 57 lines ]
[localhost:~] smartin%
```

**Example 7.** "inetd.conf" modified

Use the "control" and "X" keys (^X) to exit PICO text editor. PICO will prompt you to save the file (y/n, typing "y" saves the modifications) followed by a prompt that asks you where you would like to save the file. Hitting the return key on the keyboard saves the modified file in the default location.

Quit the terminal application and copy your Open Firmware code, text file into the directory you just defined. Restart your system to use the tftpd daemon.

Back to top

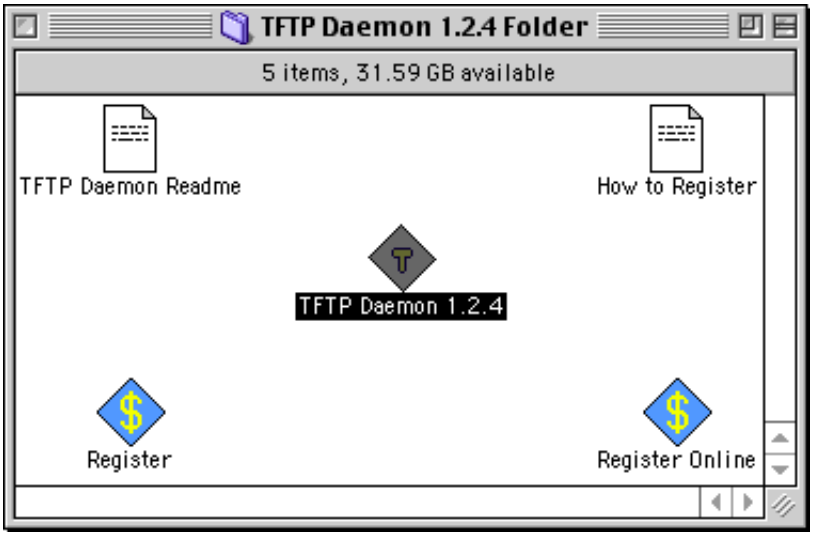
### Setting up tftpd under MacOS 9 on the Host machine

1. Connect the host machine to the target machine using the crossover Ethernet cable.

   The host machine will be running the Mac OS while the target machine will be running the Open Firmware user interface.

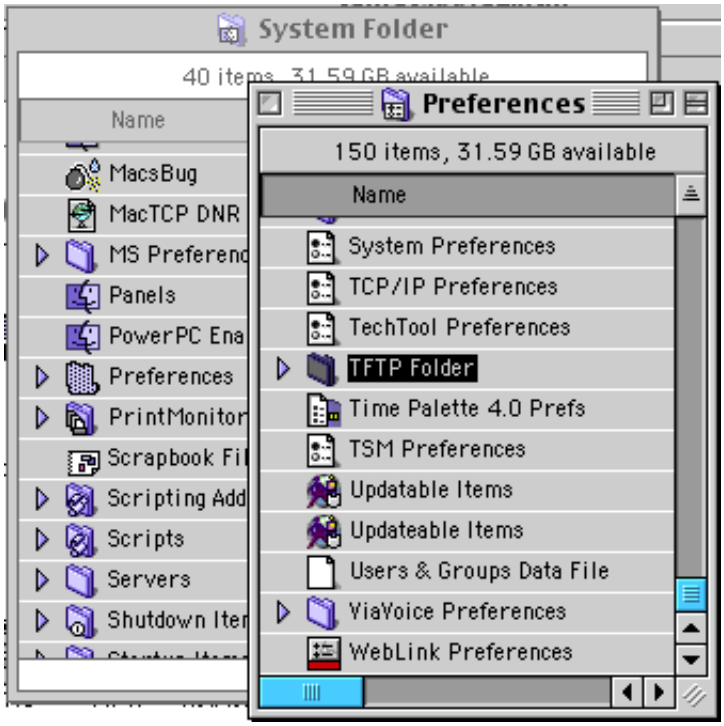
2. Launch the TFTP server application on the host machine.

3. Launch the TFTP Daemon application to start the TFTP server process.

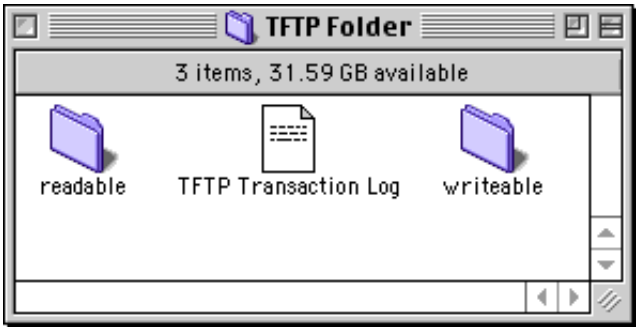



Example 2. TFTP Daemon server application example.

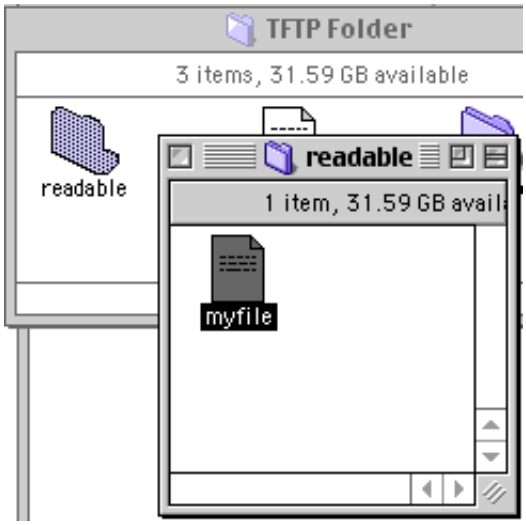
4. The application installs a "**TFTP Folder**" in the preference folder in the Host Macintosh's System folder.




The TFTP Folder contains 2 folders. These folders are the "readable" and "writeable" folders.




Save the Forth development code file as a text file in the "**readable**" folder.




The TFTP Daemon application (used in example 1), upon launch, installs a folder named "**TFTP Folder**" in the host machine's Preference folder inside the Mac OS' System folder. The server application continually runs in the background with no other outward indications. In this example "myfile" from example 1 is placed in the TFTP "readable" folder to be transferred.

Back to top

## Transferring code files to the target machine

If you are not using a loopback cable as the only ethernet connection between 2 machines, security may be an issues as tftp is not a secure protocol. The following article on tftp security may be useful for MacOS X users.

http://www.netsys.com/bsdi-users/1998-03/0206.html

On the target (client) machine, in the Open Firmware user interface of the system to be debugged, type:

```
boot enet:<server IP>,<file name>,<client IP>;<subnet>,;<router IP>
```

where

"*<server IP>*" is the IP address of the host / TFTP server system (the one where you are developing and editing your files),

"*<file name>*" is the name of the text file to be transferred (the file in the TFTP "readable" folder,

"*<client IP>*" IP address of the target / client system (the machine in Open Firmware where you transferring your files),

"*<subnet>*" is the server subnet IP mask and the

"*<router IP>*" is the IP address of the address of the router for the immediate network.

Fortunately, most of these IP addresses can be obtained easily on a Macintosh from the TCP/IP node under "Network overview" heading in the Apple System Profiler (ASP) application.

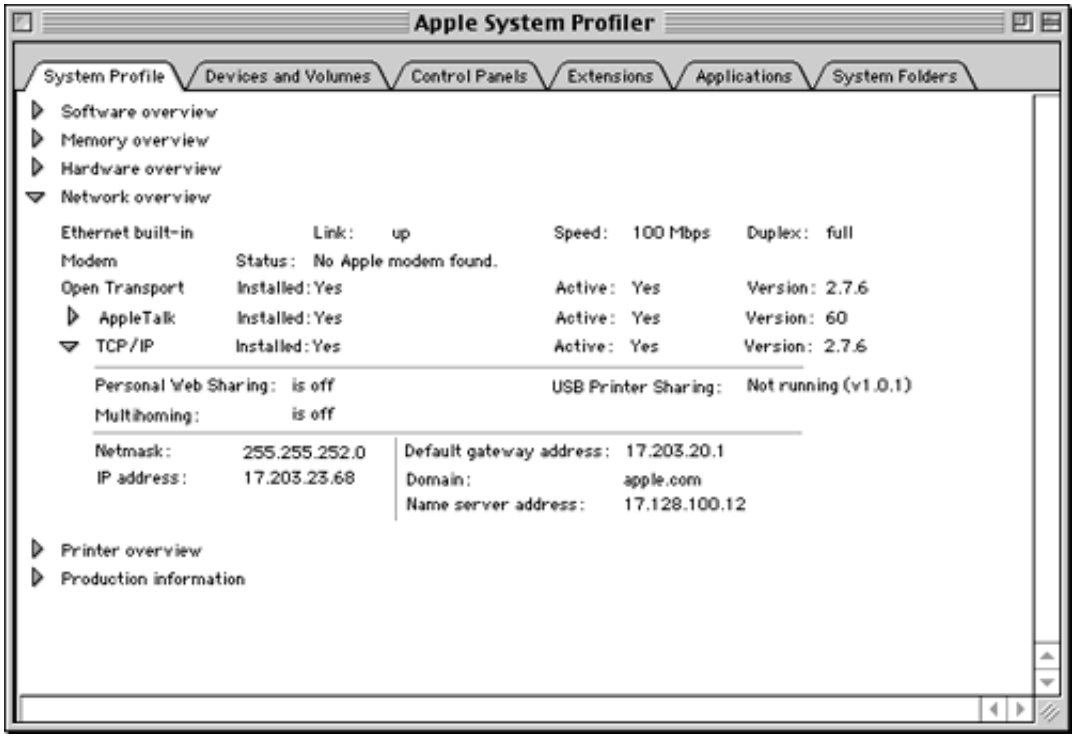
**Example 8.** Apple System Profiler Network Overview

Under "Network overview" heading in the Apple System Profiler

*<server IP>* = IP Address **(e.g.17.203.23.68)**

*<subnet>* = Netmask **(e.g. 255.255.252.0)**

*<router IP>* = Default gateway address **(e.g. 17.203.20.1)**




As mentioned in the "Using the correct Ethernet connection" section, using a crossover cable eliminates problems introduced by Internet traffic and unwanted broadcast messages. In a network where a crossover cable is used, the Host's (or server's) IP address becomes both the server IP and router IP address and must be entered twice.

This can be seen in Example 1 above, where the server IP and router IP address are both `10.1.2.2` and the target's IP address is `10.1.2.3`.

The IP address for the server and router in a network where a crossover cable is used, must be the same as that in the TCP/IP control panel, but the last numeral to the right, must be different than the IP address for the target IP address.

For example, on the system to be debugged, type:

```
boot enet:10.1.2.2,myfile,10.1.2.3;255.255.0.0,;10.1.2.2
```

Pressing the carriage return or enter at the keyboard of the target machine (in Open Firmware)will execute the instruction. The contents of designated file (e.g. myfile) will be transferred from the host (server) machine to the target machine where the code will be executed.

Editing to the FORTH code file can be done on the host machine and each update can be "tested" by reentering the "boot enet" words.

Note that Example 1 includes the word "update" that becomes a short cut for the entire "boot enet" words and IP addresses.

Once the systems are set up the editing and debugging session becomes a simple flow as documented in Figure 2 below.

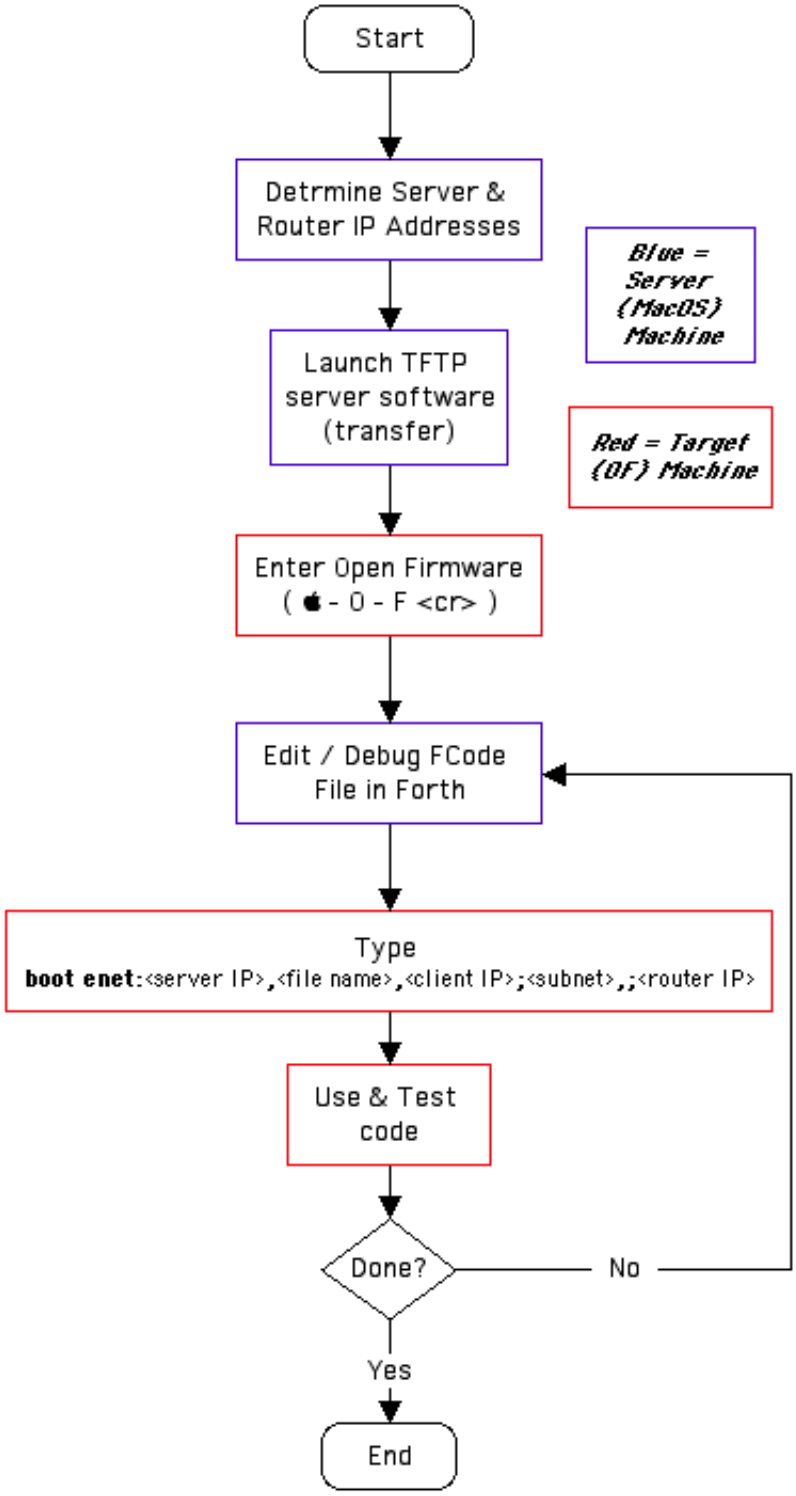



**Figure 2.** Ethernet / Telnet Debug Flow

Back to top

## Summary

Debugging any code during the boot sequence can be difficult. This is due to system resources not being available while the sequence is in progress. This is especially true for any Forth code embedded in a peripheral device. The Open Firmware ("one machine mode") interface can limit a developer's ability to save information and therefore, slow the debugging process.

Using Ethernet to provide a "two machine mode" interface is just one more tool that may be of help speed up the development cycle. It adds the ability to quickly save and edit FORTH code files on the host machine. Transferring the edited file to and executing this file on the target machine, provides quick and convenient testing in Open Firmware.

Back to top

## References

- Apple's PCI Development Web Page
- Apple's Hardware Technical Note Collection
- Apple's Hardware Technical Q&A Collection

Back to top

## Downloadables

 Acrobat version of this Note (2600K). Download

Back to top

---

Technical Notes by Date | Number | Technology | Title
Developer Documentation | Technical Q&As | Development Kits | Sample Code