Freescale Semiconductor
Application Note

Document Number: AN3931
Rev. 0, 09/2009

FTP Server Using MCF51CN Family and FreeRTOS

by: Paolo Alcantara,
Jose Ruiz
Applications Engineering
RTAC Americas

1 Introduction

This document describes an FTP server using the MCF51CN128, the open source RTOS FreeRTOS ® v5.3.0 and the TCP/IP stack lwIP v1.3.0.

This document shows how an embedded device sends and receives files over the Internet using the file transfer protocol (FTP). It gives a brief explanation about the FTP and how a file system is mounted over an external portable storage device as an SD card using the MCF51CN128.

This document is intended to be used by all software development engineers, test engineers, and anyone else who needs details about FTP and file system implementation on the MCF51CN128.

Contents

1 Introduction 1
2 File Transfer Protocol (FTP) 2
2.1 Hardware Implementation 2
2.2 Principle of Operation 4
3 Introduction to the FTP Server Software 7
3.1 SD Interface 9
3.2 Limitations 9
3.3 Principle of Operation 10
4 FTP Software 10
4.1 Software Architecture 10
4.2 Software Hierarchy 11
5 FTP Server API 14
6 Customization 14
7 Conclusion 14
8 Considerations and References 14

freescale™ semiconductor

2 File Transfer Protocol (FTP)

An FTP server is one of many TCP/IP applications. The FTP's purpose is to exchange files on the Internet. It works in the client-server architecture. The following implementation acts as an FTP server allowing a single FTP client to connect to this software. The FTP server uses a file system implementing the FAT16 protocol allowing communications to a secure digital (SD) card. The use of a portable non-volatile memory card such as an SD card makes it compatible with other devices like a PC.

Here is a list of possible uses for the FTP server in an embedded device like the MCF51CN128:

- Store information using FAT and then have it accessible using FTP.
- Store large files using FTP and then view them on a separate SD card reader like a computer for further analysis.
- Use the software like an Ethernet-SD card bridge.

Figure 1 shows an FTP standard scenario.

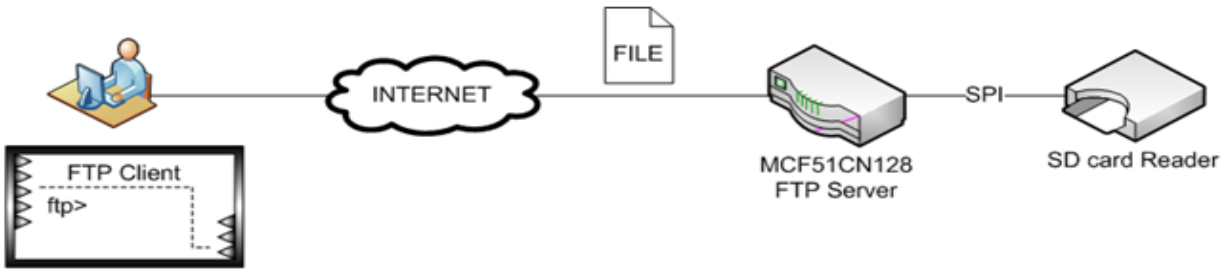


Figure 1. FTP client connecting to an FTP server

2.1 Hardware Implementation

This application note works with MCF51CN128 Reference Design and the Tower System. For more information about the MCF51CN128 reference design, go to the MCF51CN128 Product Summary Page. For Tower System information, visit www.Freescale.com/tower.

A hardware block diagram of the MCF51CN128 reference design is presented for clarity.

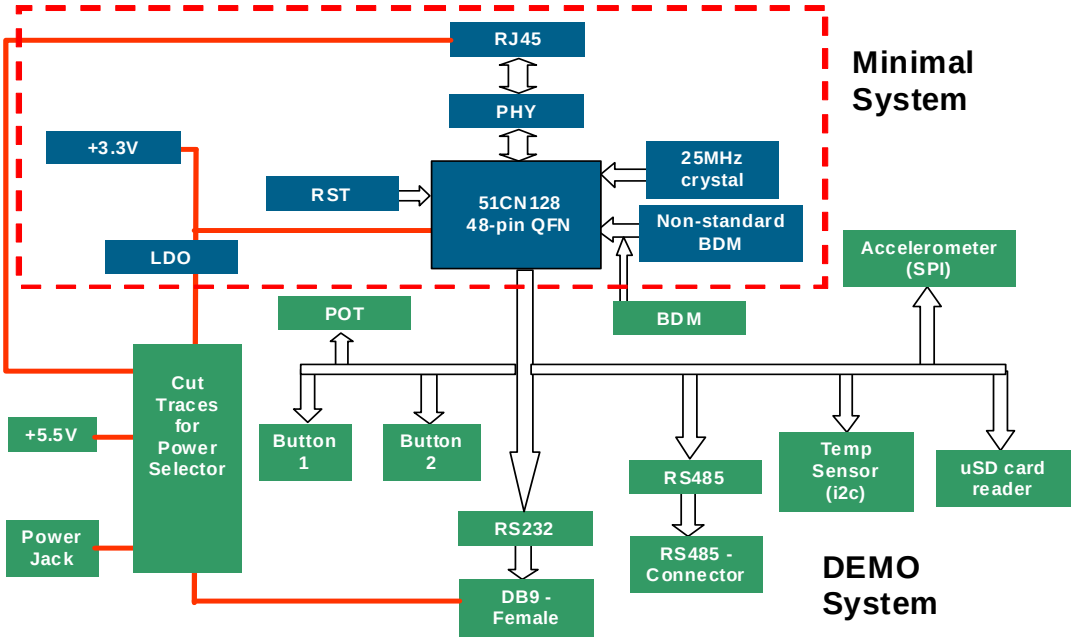


Figure 2. Hardware block diagram of MCF51CN128 reference design board

For the MCF51CN128 reference design hardware, jumpers must remain in the default position. The board is then ready to use as it is. Board schematics, layout, and Gerber files are provided in case a customization is required in the hardware for specific use of the email client, like removing additional components besides Ethernet or sensors.

For the TWR-MCF51CN and TWR-MEM Towers Rev C, the default jumper configuration must be used, except for the following jumpers:

Table 1. TWR-MCF51CN jumper settings

Jumper	Position	State
J2	1–2	Set
J3	1–2	Set
	3–4	Set
	5–6	Set
J5	9–10	Remove
	11–12	Remove
	13–14	Remove

NOTE

The dip switch SW1 (both contacts) must be in the position in which both point to the center of the board.

Table 2. TWR-MEM jumper settings

Jumper	Position	State
SD_CS	1–2	Set
SD_SEL1	1–2	Set

Differences between both hardware or target boards are:

- Reference design uses micro SD cards
- Tower uses SD cards

To test the application in the Tower System, the memory card (TWR-MEM) must be placed as close as possible to the MCU card for maximum SPI baud rates.

In the reference design, SPI baud rate goes up to 12.5 Mbps because its SPI trace lengths are very close to the MCU and hence very short.

2.2 Principle of Operation

The software presented in this application note is not a full RFC959-compliant FTP server. It will be demonstrated with the Microsoft Windows® command-line FTP client. Attempts to use other FTP client tools are not considered in this application note. A common FTP client like a web browser tries to use more than one data socket for performance, and then the FTP server implementation described here will fail.

An SD card needs to be connected to the hardware to start the FTP server. If the FTP task is running, it will be displayed in the list of running tasks typing [ip address]/tasks.htm in a web browser. For more details, refer to application note *Serial-to-Ethernet Bridge Using MCF51CN Family and FreeRTOS* (document AN3906).

To connect the FTP client to the FTP server, enter the following text in the Microsoft Windows command line.

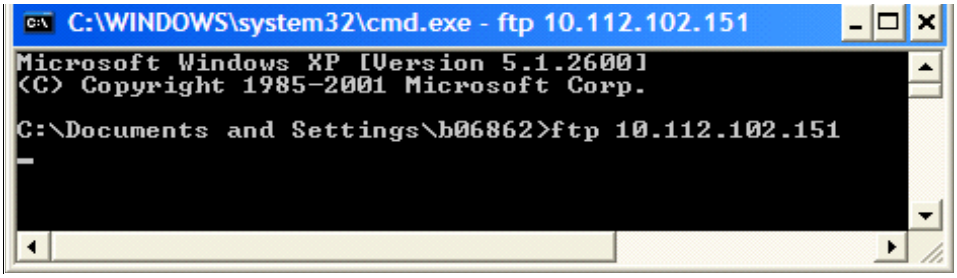


Figure 3. FTP client connecting to the FTP server

If the FTP client cannot connect to the FTP server, then the following is a list of possible causes.

- LAN dynamic host configuration protocol (DHCP) server cannot provide a valid IP address if working with dynamic addresses.
- LAN has FTP port 21 blocked
- FTP server cannot validate username or password provided
- FTP task does not start because no SD card is detected in the system.

Table 3 lists the interpreted commands implemented when using Microsoft Windows command-line FTP client. The commands are interpreted because one or more FTP command is required to run these commands. For example, LS requires PORT and NLST to work with the FTP server. For more details about FTP commands, go to Table 5.

Table 3. Microsoft Windows command-line FTP client implemented commands

FTP client	Description
LS	Displays files
GET [filename]	Copy a file from FTP server to FTP client
PUT [filename]	Copy a file from FTP client to FTP server
DELETE [filename]	Delete a file from FTP server
QUIT	Ends FTP session
HELP	Show description of FTP commands

Figure 4 shows an established FTP client session to the FTP server application. A username and password are required to view the files. Files can be:

- Listed
- Uploaded
- Downloaded
- Deleted

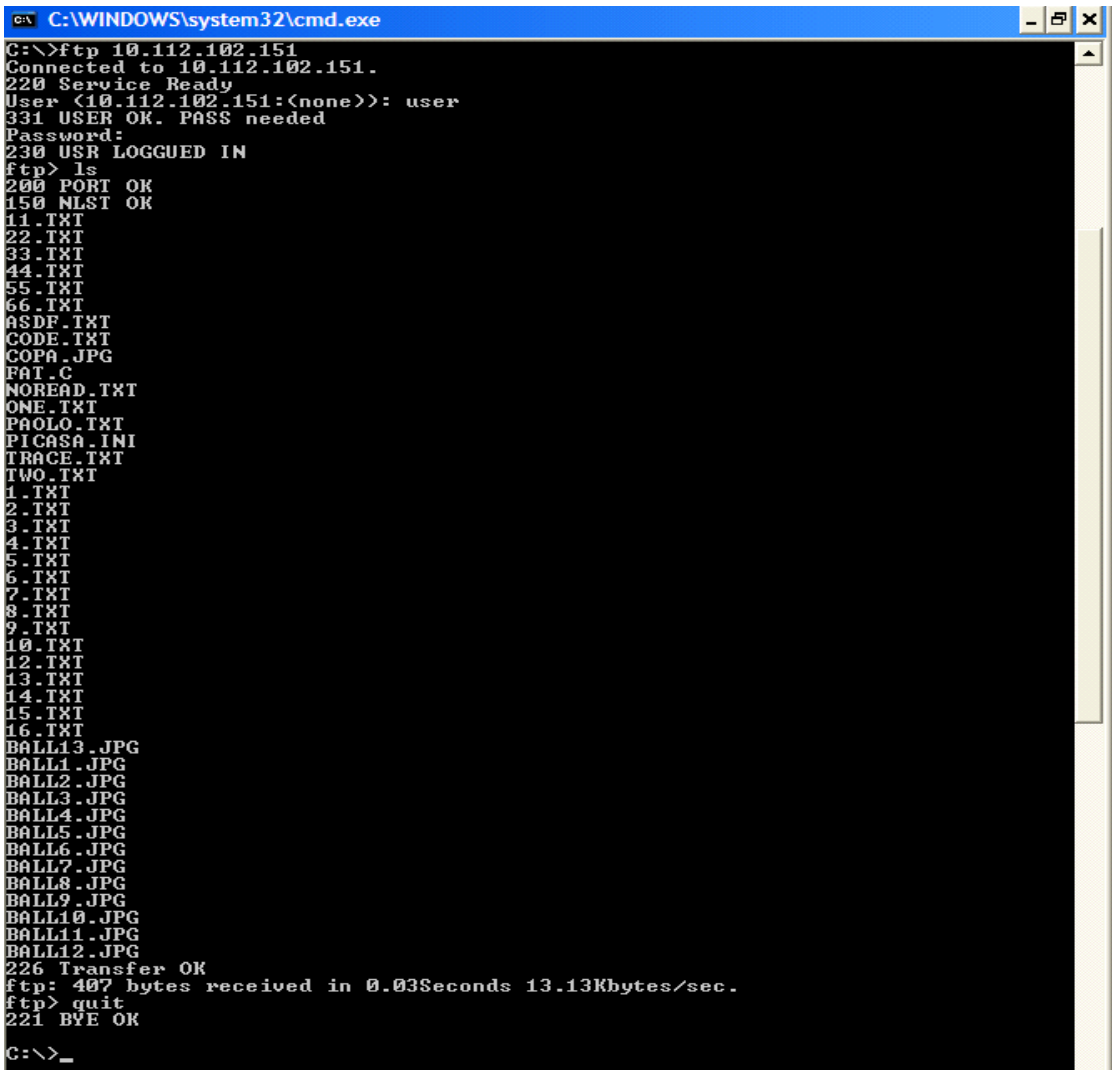


Figure 4. Microsoft Windows command-line FTP client with MCF51CN128

The FTP server application starts with the following configuration but can be changed at runtime using configuration web page viewable through a web browser.

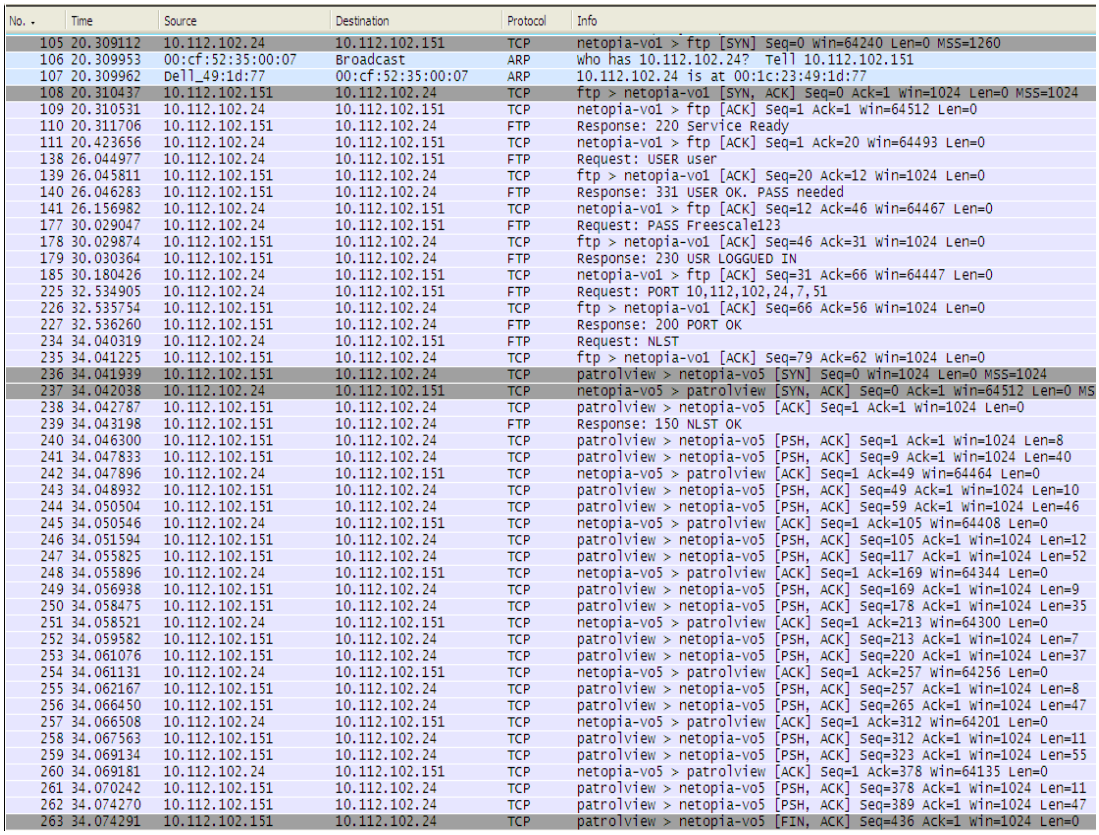
Table 4. Default MAC parameters

MAC Parameters	
MAC Address	00:CF:52:35:00:07
IP Address	192.168.1.3 for static implementation
Mask Address	255.255.255.0
Gateway Address	192.168.1.1
Server Address to Connect to an Address	192.168.1.3
Static or Dynamic Address	Static

3 Introduction to the FTP Server Software

lwIP provides three levels of APIs for sockets. The email client uses the netconn level. For more details about socket's level with lwIP, refer to application note *Serial-to-Ethernet Bridge Using MCF51CN Family and FreeRTOS* (document AN3906).

Figure 5 shows a log file from Wireshark network analyzer. It displays all the TCP transactions needed to conduct an FTP transfer — USER, PASS, PORT, and NLST commands.



No.	Time	Source	Destination	Protocol	Info
105	20.309112	10.112.102.24	10.112.102.151	TCP	netopia-vo1 > ftp [SYN] Seq=0 Win=64240 Len=0 MSS=1260
106	20.309953	00:cf:52:35:00:07	Broadcast	ARP	Who has 10.112.102.24? Tell 10.112.102.151
107	20.309962	Dell_49:1d:77	00:cf:52:35:00:07	ARP	10.112.102.24 is at 00:1c:23:49:1d:77
108	20.310437	10.112.102.151	10.112.102.24	TCP	ftp > netopia-vo1 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1024
109	20.310531	10.112.102.24	10.112.102.151	TCP	netopia-vo1 > ftp [ACK] Seq=1 Ack=1 Win=64512 Len=0
110	20.311706	10.112.102.151	10.112.102.24	FTP	Response: 220 Service Ready
111	20.423656	10.112.102.24	10.112.102.151	TCP	netopia-vo1 > ftp [ACK] Seq=1 Ack=20 Win=64493 Len=0
138	26.044977	10.112.102.24	10.112.102.151	FTP	Request: USER user
139	26.045811	10.112.102.151	10.112.102.24	TCP	ftp > netopia-vo1 [ACK] Seq=20 Ack=12 Win=1024 Len=0
140	26.046283	10.112.102.151	10.112.102.24	FTP	Response: 331 USER OK. PASS needed
141	26.156982	10.112.102.24	10.112.102.151	TCP	netopia-vo1 > ftp [ACK] Seq=12 Ack=46 Win=64467 Len=0
177	30.029047	10.112.102.24	10.112.102.151	FTP	Request: PASS Freescale123
178	30.029874	10.112.102.151	10.112.102.24	TCP	ftp > netopia-vo1 [ACK] Seq=46 Ack=31 Win=1024 Len=0
179	30.030364	10.112.102.151	10.112.102.24	FTP	Response: 230 USR LOGGUED IN
185	30.180426	10.112.102.24	10.112.102.151	TCP	netopia-vo1 > ftp [ACK] Seq=31 Ack=66 Win=64447 Len=0
225	32.534905	10.112.102.24	10.112.102.151	FTP	Request: PORT 10,112,102,24,7,51
226	32.535754	10.112.102.151	10.112.102.24	TCP	ftp > netopia-vo1 [ACK] Seq=66 Ack=56 Win=1024 Len=0
227	32.536260	10.112.102.151	10.112.102.24	FTP	Response: 200 PORT OK
234	34.040319	10.112.102.24	10.112.102.151	FTP	Request: NLST
235	34.041225	10.112.102.151	10.112.102.24	TCP	ftp > netopia-vo1 [ACK] Seq=79 Ack=62 Win=1024 Len=0
236	34.041939	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [SYN] Seq=0 Win=1024 Len=0 MSS=1024
237	34.042038	10.112.102.24	10.112.102.151	TCP	netopia-vo5 > patrolview [SYN, ACK] Seq=0 Ack=1 Win=64512 Len=0 MS
238	34.042787	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [ACK] Seq=1 Ack=1 Win=1024 Len=0
239	34.043198	10.112.102.151	10.112.102.24	FTP	Response: 150 NLST OK
240	34.046300	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=1 Ack=1 Win=1024 Len=8
241	34.047833	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=9 Ack=1 Win=1024 Len=40
242	34.047896	10.112.102.24	10.112.102.151	TCP	netopia-vo5 > patrolview [ACK] Seq=1 Ack=49 Win=64464 Len=0
243	34.048932	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=49 Ack=1 Win=1024 Len=10
244	34.050504	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=59 Ack=1 Win=1024 Len=46
245	34.050546	10.112.102.24	10.112.102.151	TCP	netopia-vo5 > patrolview [ACK] Seq=1 Ack=105 Win=64408 Len=0
246	34.051594	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=105 Ack=1 Win=1024 Len=12
247	34.055825	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=117 Ack=1 Win=1024 Len=52
248	34.055896	10.112.102.24	10.112.102.151	TCP	netopia-vo5 > patrolview [ACK] Seq=1 Ack=169 Win=64344 Len=0
249	34.056938	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=169 Ack=1 Win=1024 Len=9
250	34.058475	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=178 Ack=1 Win=1024 Len=35
251	34.058521	10.112.102.24	10.112.102.151	TCP	netopia-vo5 > patrolview [ACK] Seq=1 Ack=213 Win=64300 Len=0
252	34.059582	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=213 Ack=1 Win=1024 Len=7
253	34.061076	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=220 Ack=1 Win=1024 Len=37
254	34.061131	10.112.102.24	10.112.102.151	TCP	netopia-vo5 > patrolview [ACK] Seq=1 Ack=257 Win=64256 Len=0
255	34.062167	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=257 Ack=1 Win=1024 Len=8
256	34.066450	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=265 Ack=1 Win=1024 Len=47
257	34.066508	10.112.102.24	10.112.102.151	TCP	netopia-vo5 > patrolview [ACK] Seq=1 Ack=312 Win=64201 Len=0
258	34.067563	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=312 Ack=1 Win=1024 Len=11
259	34.069134	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=323 Ack=1 Win=1024 Len=55
260	34.069181	10.112.102.24	10.112.102.151	TCP	netopia-vo5 > patrolview [ACK] Seq=1 Ack=378 Win=64135 Len=0
261	34.070242	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=378 Ack=1 Win=1024 Len=11
262	34.074270	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [PSH, ACK] Seq=389 Ack=1 Win=1024 Len=47
263	34.074291	10.112.102.151	10.112.102.24	TCP	patrolview > netopia-vo5 [FIN, ACK] Seq=436 Ack=1 Win=1024 Len=0

Figure 5. Wireshark window showing FTP requests and responses

FTP server uses two sockets to connect to an FTP client:

- FTP Command Port — Used to transmit and receive FTP commands and results. By using this command port and FTP PORT command, the FTP data port is selected. The FTP client always connects to an FTP server command port.
- FTP Data Port — Used to transmit and receive FTP data during read, write, and list file operations. The FTP client or FTP server can connect to each other depending on whether the FTP communication is active or passive. There can be more than one data port.

The following FTP server implementation uses the designated TCP port 21 to receive commands from the FTP client. Some firewalls block TCP port 21 as a security policy.

The FTP server allows only active port mode, this means the FTP server opens the FTP data port socket to the FTP client. A PORT command sent from the FTP client instructs the FTP server port to attempt

connection. The FTP server receives information by using sockets. With this implementation the FTP server returns the control to the CPU while waiting for FTP client requests. This avoids blocking the execution of critical actions by FreeRTOS.

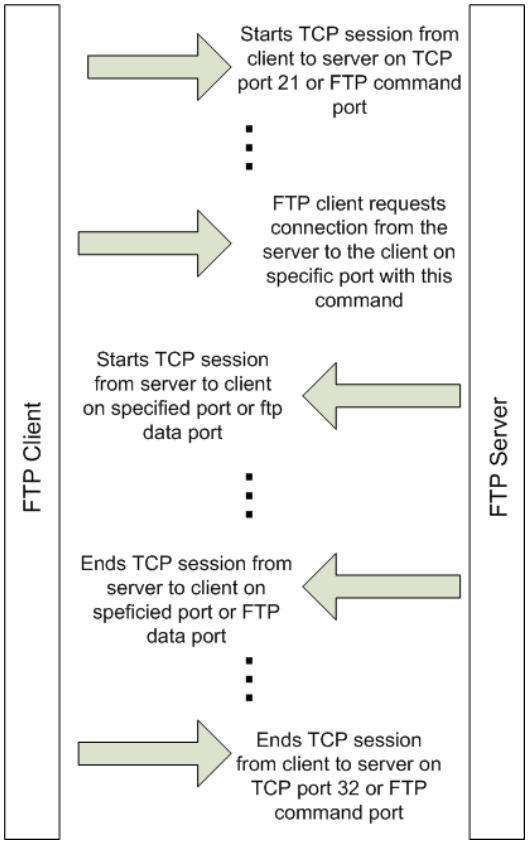


Figure 6. FTP active port mode

In this implementation, the FTP requires a simple authentication method. The FTP server asks only for username and password. This information is sent with no security scheme, which means username and password are transmitted as they are. Data is also transmitted without any security scheme.

Table 5 lists the FTP commands implemented for the following FTP server implementation.

Table 5. FTP server implemented commands

FTP command	Description
USER	Send username
PASS	Send password
QUIT	End the FTP session
PORT	Open a data port

Table 5. FTP server implemented commands (continued)

FTP command	Description
NLST	List names of the FTP file system
RETR	Retrieve a file from FTP server
STOR	Store a file to FTP server
DELE	Delete a file from FTP server

For this implementation, the following information is required to log in:

Username: user

Password: Freescale123

3.1 SD Interface

Regarding the SD interface, two transfer modes are available from SD manufacturers:

- Four-bit SD mode — Extra pins are used for communication
- SPI mode — Using serial peripheral interface (SPI) interface

The following SD application implements the SPI mode and in the MCF51CN128 can run up to 12.5 Mbps. The SD application reads or writes the SD card in 512-bytes blocks.

A file system is used because a file allocation table stores:

- Which areas belong to the files and where they are
- All unused areas

The used file system is FAT16. FAT16 implementation is mounted over the SD card software, but the software can point to another storage device.

Files are stored in several sectors called “clusters”. Each one is identified by 16-bit addresses and then by its name.

The FTP server implementation hides many of the FAT16 implementation details, presenting functions to read, write, or delete files. Because the file system is working on a multitasking system, the use of semaphores must be used to avoid concurrent accesses to SD shared resources. In this implementation, a web server and the FTP server have access to the SD card.

3.2 Limitations

The FTP server implementation has the following limitations:

- Allows only one client and one transfer per time.
- FTP server was tested using Microsoft Windows command-line FTP client.
- If an non-implemented command is received, the FTP client is notified and the FTP session is closed.

For the FAT16 implementation, there are two limitations:

- The name of the file must be up to eight characters and the extension must be three characters.
- Current implementation does not work with directories.

3.3 Principle of Operation

The software is developed for the MCF51CN128 reference design hardware to demonstrate low-cost and small board size. But, it can also be used in the Tower board.

Select between either M51CN128RD or V1TOWER C-macros inside the m51cn128evb.h file.

```
/*****************************************************************************/

/*Warning: only define one of them*/
#define M51CN128RD            /*pins moved to reference design hardware*/
//#define V1_TOWER              /*pins moved to reference design hardware*/
```

Figure 7. Code snippet for hardware change

4 FTP Software

4.1 Software Architecture

Figure 8 shows how the FTP server is divided and which software blocks are used for this implementation.

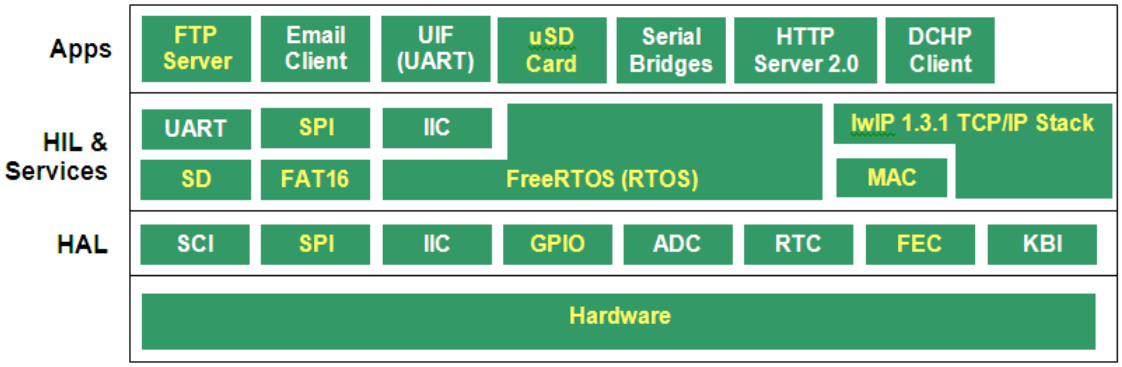


Figure 8. Software segmentation

For more information regarding memory footprint or details regarding FreeRTOS or lwIP, refer to the application note *Serial-to-Ethernet Bridge Using MCF51CN Family and FreeRTOS* (document AN3906).

The FTP server+FAT16+SD software takes 6.1 kB of ROM and 65 B of RAM. A more detailed memory footprint is presented as follows:

Table 6. FTP server memory footprint

Files	ROM	RAM
SD	1028	4
FAT16	2508	53
FTP_Server	2724	8
TOTAL	6260B	65B

NOTE

The RAM requested by the FTP server at runtime is not considered.

4.2 Software Hierarchy

Figure 9 shows the files hierarchy.

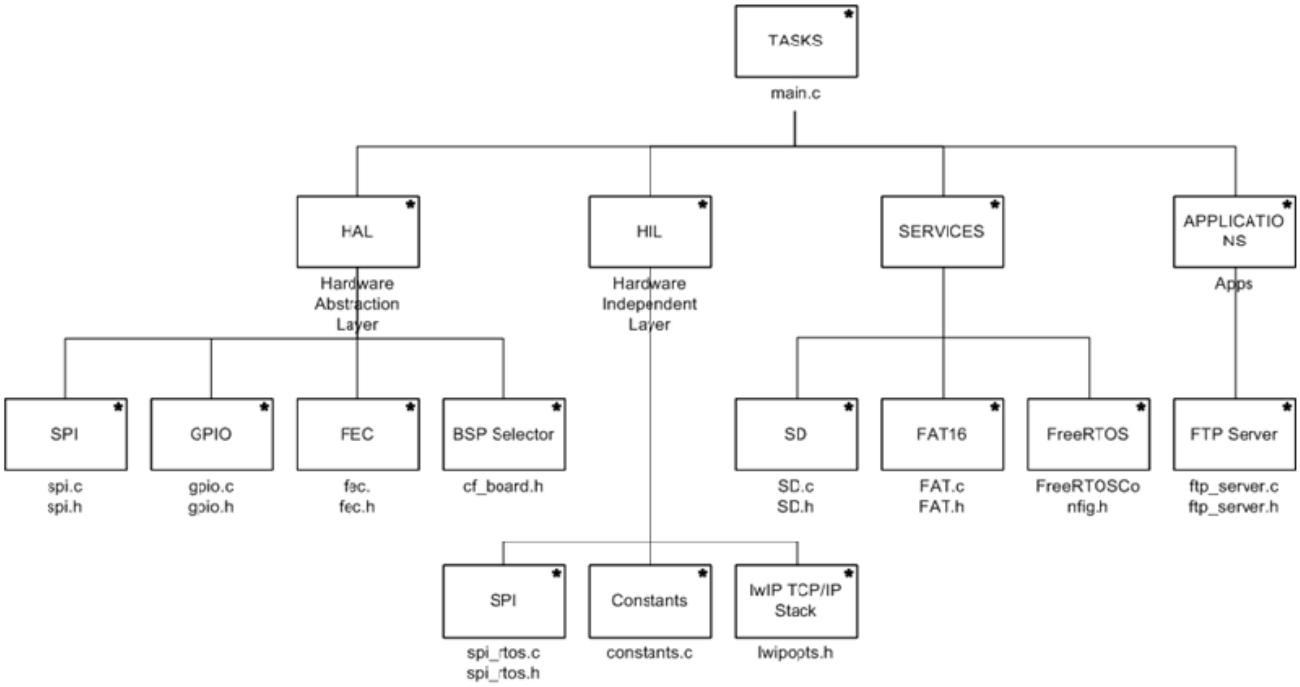


Figure 9. File implementation

Table 7 provides an explanation of each hierarchical file.

Table 7. Software file descriptions

Layer	File Name	Description
Main	main.c	File that calls the FTP task
HAL	gpio.c	Routines that use pins directly for the selected MCU
	gpio.h	Points all the modules to a specific pin for the selected MCU
	spi.c	Low-level Init for SPI bridge
	spi.h	Macros containing SPI low-level utilities
	fec.c	Low-level Init for FEC driver
	fec.h	Number and length of RX/TX buffers
	cf_board.h	HAL layer to use with this serial bridge
HIL	mac_rtos.c	MAC driver used by lwIP TCP/IP stack
	mac_rtos.h	MAC driver header
	spi_rtos.c	SPI interrupted or polling processing
	spi_rtos.h	SPI settings for high-level
	constants.c	Structure containing all the default parameters after reset
	lwipopts.h	lwIP options to enable and disable services
Services	SD.c	Uses SPI low level interface to access an SD card
	SD.h	SD card header
	FAT.c	FAT16 implementation over SD
	FAT.h	FAT16 header
	FreeRTOSConfig.h	FreeRTOS options to enable and disable services
Applications	ftp_server.c	Task running the FTP server
	ftp_server.h	Contains task priority and FTP request messages

4.2.1 Hardware Abstraction Layer (HAL) Implementation

The HAL is defined as the collection of software components that make direct access to the hardware resources such as peripherals, configuration registers, optimized assembler routines (with their appropriate prototypes), pre-compiled object code libraries, or any other hardware-dependent resource, through the HAL-HW interface.

4.2.2 Fast Ethernet Controller (FEC) Handling

Due to reduced memory footprint, a single Tx buffer is used to transmit data and two Rx buffers are used to receive information. For details on fast Ethernet controller (FEC) handling, refer to application note *Serial-to-Ethernet Bridge Using MCF51CN Family and FreeRTOS* (document AN3906).

4.2.3 Hardware Independent Layer (HIL) Implementation

To maintain hardware independence, software components that belong to this layer can only access the controller's resources by means of HIL components. Therefore, they refrain from directly accessing the resources of the controller on which they are running. This feature allows for components from this and the above sitting layers to run on different controllers without further change.

For more details about HIL blocks, refer to application note *Serial-to-Ethernet Bridge Using MCF51CN Family and FreeRTOS* (document AN3906).

4.2.4 FTP Server Application Implementation

Figure 10 represents the FTP server logic that handles all FTP requests.

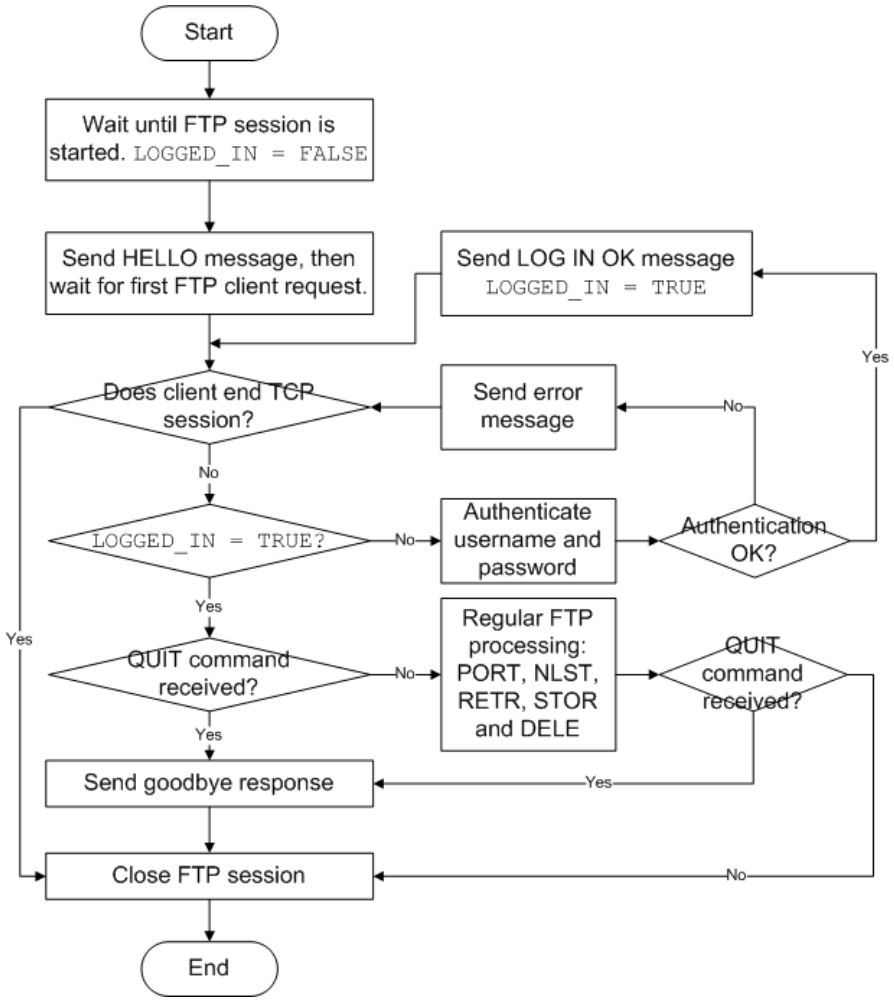


Figure 10. Basic FTP software implementation

5 FTP Server API

The main function used for the FTP server task is as follows:

Syntax

```
void
vBasicFTPServer( void *pvParameters )
```

Description — Starts FTP server task

```
/**
 * Start an embedded FTP server Task: 1 client and 1 file per transfer
 * * @param parameter pointer to use with task
 * @return none
 */
```

6 Customization

For customization, the following files must be modified for a change in software or hardware:

File Name	Description
cf_board.h	Used to point to a new BSP, new HAL software drivers
lwipopts.h	lwip configuration file. Enable and disable TCP/IP options
gpio.c/gpio.h	Change GPIO used for all modules in the MCU
FreeRTOSConfig.h	FreeRTOS user configuration file. Enable and disable features
ftp_server.c	Basic FTP server: Upgrades to FTP must be included to this file
ftp_server.h	New FTP request strings must be added to this file.

To add new commands to the FTP server, the following files must be changed:

- ftp_server.c — If there are more FTP commands like PASV, RMD, RNTO, or TYPE they must be implemented in this file. To change default username and password, change them at compile time. Username is stored in a C-macro named `FTP_USERNAME`. Password is stored in a C-macro named `FTP_PASSWORD`.
- ftp_server.h — FTP commands and responses are declared on this file. To add more or change responses string use this file.

7 Conclusion

The application note document and software demonstrate how to share files to the Internet using FTP and file systems mounted over an SD card. Details about the implementation were shown so the current software can be changed to a specific user application.

8 Considerations and References

Find the newest software updates and configuration files for the MCF51CN128 on the Freescale Semiconductor home page, www.freescale.com.

- MCF51CN128 Reference Design and Tower System were the hardware used to test AN3931SW.
- For more information on FEC, refer to *MCF51CN128 Reference Manual* (document MCF51CN128RM) at www.freescale.com.
- To learn more about the Tower System, refer to www.freescale.com/tower.
- To learn more about the MCF51CN128 Reference Design details, refer to MCF51CN128 Product Summary Page.
- The BridgeSoftwareDemo software was developed and tested with CodeWarrior for ColdFire V6.2.1.
- Download the source files for AN3931SW.zip from www.freescale.com.
- For more information regarding software or hardware, refer to www.freescale.com/support.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3931
Rev. 0
09/2009

freescale™
semiconductor