Freescale Semiconductor
Application Note

Document Number: AN3928
Rev. 0, 08/2009

Web Server Using the MCF51CN Family and FreeRTOS

by: Paolo Alcantara
Applications Engineering
RTAC Americas

1 Introduction

This document describes a web server using the MCF51CN128, the open source RTOS FreeRTOS ® V5.3.0, and the TCP/IP stack lwIP V1.3.0.

This document discusses the following implementations on the MCF51CN128:

Web server with:

- Dynamic content
- AJAX
- DHCP
- File system (FAT16)

This document is intended to be used by all software development engineers, test engineers, and anyone else who needs to use an embedded web server.

Contents

2 Introduction to Web Server

This web server software allows you to post information to the world wide web (WWW) that is easily viewable by a standard web browser. It does not require a deep knowledge of Ethernet or TCP/IP to use or change it. Figure 1 shows web server features:

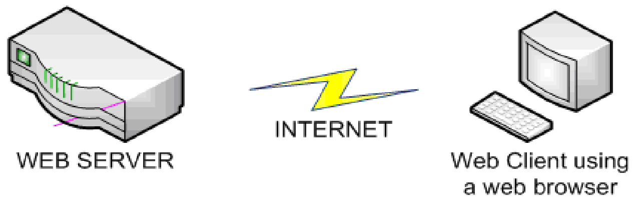


Figure 1. Web server implemented in MCF51CN family providing HTTP services

2.1 Hardware Implementation

This application note works with the 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.

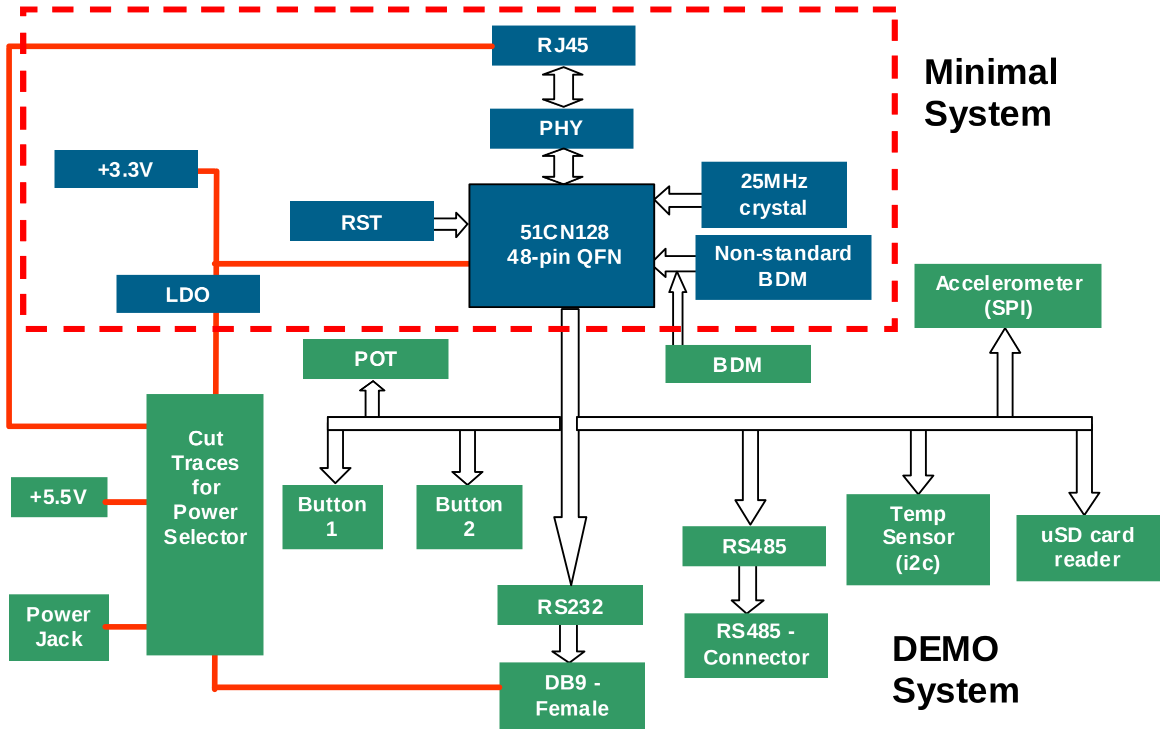


Figure 2. Hardware block diagram of MCF51CN128 reference design board

For the MCF51CN128 reference design hardware, jumpers must remain in the same 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 web server, like removing additional components besides Ethernet.

For the TWR-MCF51CN Tower Rev C, the default jumper configuration must be used.

2.2 Principle of Operation

The web server works with common web browsers like Internet Explorer or Mozilla Firefox. One way to know the IP address assigned if using dynamic IP assignation is to use a PC network protocol analyzer like Wireshark. Type the following in the expression filter option of the tool:

```
eth.src == 00:CF:52:35:00:07 || eth.dst == 00:CF:52:35:00:07
```

All the LAN packets with this Ethernet address are then shown. You must connect the web server hardware in the same network hub or network switch of the PC using Wireshark.

As soon as the web server address is known, this can be typed as follows in the web browser:

http://192.168.1.82/

If the web page does not appear, check Proxy settings. When using a direct connection from a PC to the web server, use a crossover cable or if using through a hub, use a straight cable.

For details on how to change MAC parameters, refer to application note *Serial-to-Ethernet Bridge Using MCF51CN Family and FreeRTOS* (document AN3906).

The software accompanying this application note can be obtained by the IP address using the dynamic host configuration protocol (DHCP). The DHCP can be enabled by changing a MAC parameter. Figure 3 shows mac.shtml DHCP selector stored in the web server.

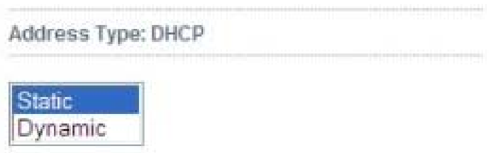

Address Type: DHCP

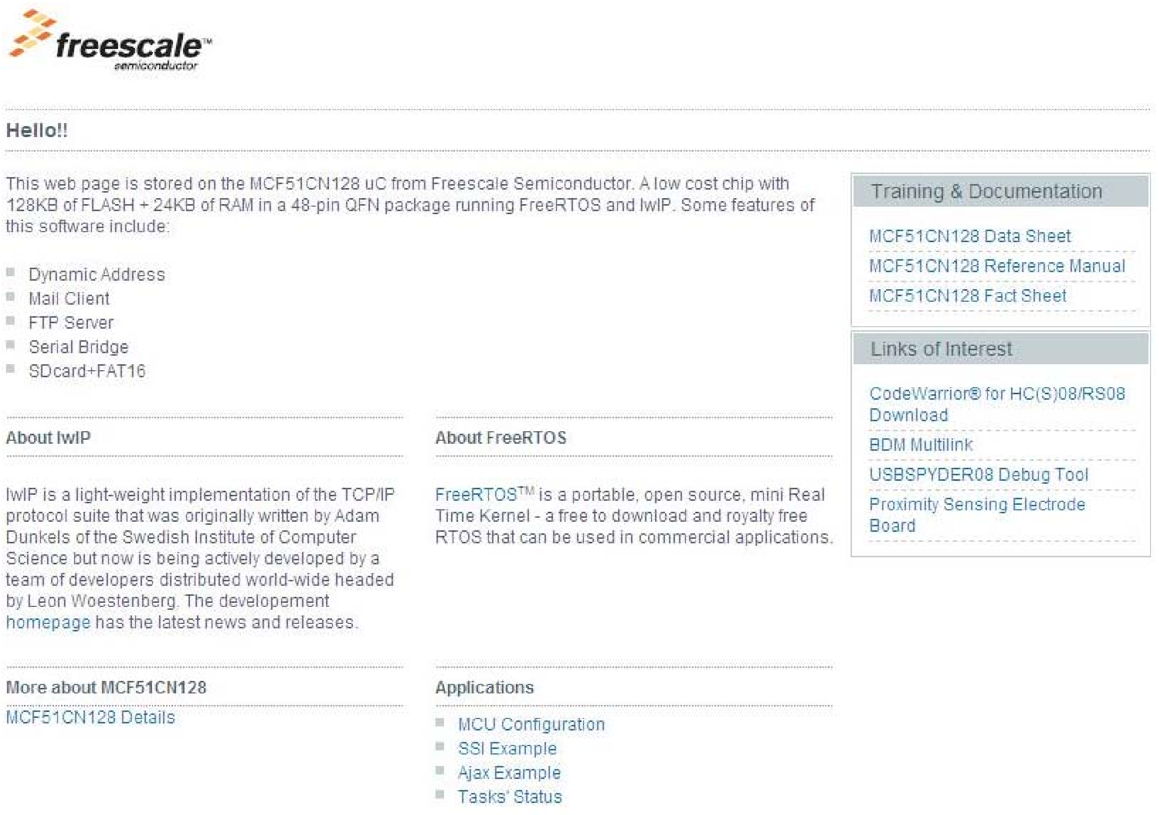
Figure 3. Static or dynamic IP address selector

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

Table 1. 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

For example, the following web page is the home page displayed when the assigned IP address is typed in the web browser. It also allows navigation to all the web pages stored in the MCF51CN128 internal ROM memory.



Hello!!

This web page is stored on the MCF51CN128 uC from Freescale Semiconductor. A low cost chip with 128KB of FLASH + 24KB of RAM in a 48-pin QFN package running FreeRTOS and lwIP. Some features of this software include:

- Dynamic Address
- Mail Client
- FTP Server
- Serial Bridge
- SDcard+FAT16

About lwIP

lwIP is a light-weight implementation of the TCP/IP protocol suite that was originally written by Adam Dunkels of the Swedish Institute of Computer Science but now is being actively developed by a team of developers distributed world-wide headed by Leon Woestenberg. The developement homepage has the latest news and releases.

About FreeRTOS

FreeRTOS™ is a portable, open source, mini Real Time Kernel - a free to download and royalty free RTOS that can be used in commercial applications.

More about MCF51CN128

MCF51CN128 Details

Applications

- MCU Configuration
- SSI Example
- Ajax Example
- Tasks' Status

Training & Documentation

MCF51CN128 Data Sheet
MCF51CN128 Reference Manual
MCF51CN128 Fact Sheet

Links of Interest

CodeWarrior® for HC(S)08/RS08 Download
BDM Multilink
USBSPYDER08 Debug Tool
Proximity Sensing Electrode Board

Figure 4. Home webpage

The web server is able to manage the following web services. For this particular application note, a list of web services is presented with their respective applications:

- HTTP 2.0 — Persistent connections or keep-alive sessions are useful to avoid opening new TCP connections for each file, thereby increasing HTTP response performance. This feature is needed to use the AJAX method.
- SSI — Server side include (SSI) directives that give dynamic content to a web page. For example, a string previously defined as "IP_ADDRESS" in HTML code can be replaced with "10.81.64.38" when it is requested by a client. In this application, the SSI shows MCU settings stored on the ROM at runtime.
- AJAX — Displays web page changes without refreshing all the contents on the web page. From an AJAX perspective, the web server needs to support only SSI directives. The web client must support JavaScript to request information in background mode and does not refresh the entire page. The web client is responsible for requesting this information at regular times. This application uses AJAX to update counters every millisecond.
- FORMS (POST request) — A web-form is a friendly scheme in which you can send information to the web server. MCU settings that can be changed are present in a web-form.

- CGI (POST request) — For this application, a common gateway interface (CGI) is used to run a predefined routine inside the web server (MCU) and generate a new web page in response. In this application, a CGI is run when a web-form is sent for processing.

NOTE

The web server also presents a webpage showing what RTOS tasks are running and what their stack consumption is for a better customization.

3 Introduction to the Web Server Software

The web server is implemented using lwIP TCP/IP stack. To access the socket interface, the lwIP uses three levels. These levels are sorted based on ease of use but with more ROM and RAM usage.

- BSD Socket Interface — Is a friendly API that accesses network sockets with the well-known socket interface popularized by several TCP/IP stacks.
- Netconn Interface — Is a low-level implementation of the socket interface targeted for low-memory MCUs. It requires more cooperation from the application than BSD sockets, but it remains easy to use and thread-safe. This interface was used for the web server as a balance between low-memory code and user-friendly API.
- Raw API Interface — Is near the TCP layer and contains a set of callbacks in which the TCP/IP stack needs application cooperation to read/write network sockets.

The HTTP or web services work in two communication sides, server and client. The server executes client requests. These HTTP requests can be as long as 800 bytes. A standard web server application receives this information by copying it from the TCP/IP stack to the RAM area. Because RAM is a valuable element in the MCF51CN128, the web server takes the received request by reference or in a non-copy fashion. The received request is managed as a linked list of the packet’s segments and each can go up to a configurable number. In this application note, the TCP/IP packet's segments are limited to 256 bytes. The lwIP has a memory management that handles packets better than the application layer. The web server then reads the client request from a linked list that requires more bytes of ROM but avoids RAM segmentation at the application layer.

The web server is able to show static content. Content can be of any length and the TCP/IP is responsible of splitting it on necessary HTTP (TCP packets) responses from the server to the client.

3.1 Server Side Include (SSI) Support

The web server is also able to send internally-generated web pages based on some runtime-replaced strings using SSI directives.

For example:

```
<!--#echo var="GREETING"-->
```

This text is replaced by a standard string if the following is true:

- GREETING is a valid string that executes a function to replace the complete SSI string with a new one.
- Web page extension ends with .FSL or .SHTML

In this example, GREETING is replaced by a standard greeting that depends on the web browser language requesting the web page.

```
Accept-Language: en
```

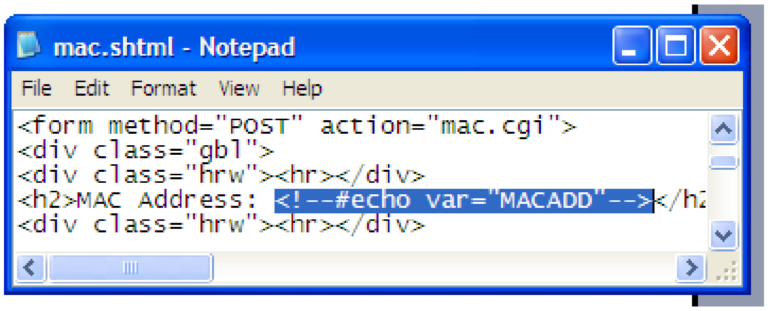
This request is sent during GET request and as soon as the GREETING request is received, it is replaced by "Hello" (English). By finding:

```
Accept-Language: fr
```

The GREETING string is replaced by "Salut" (French) and so on.

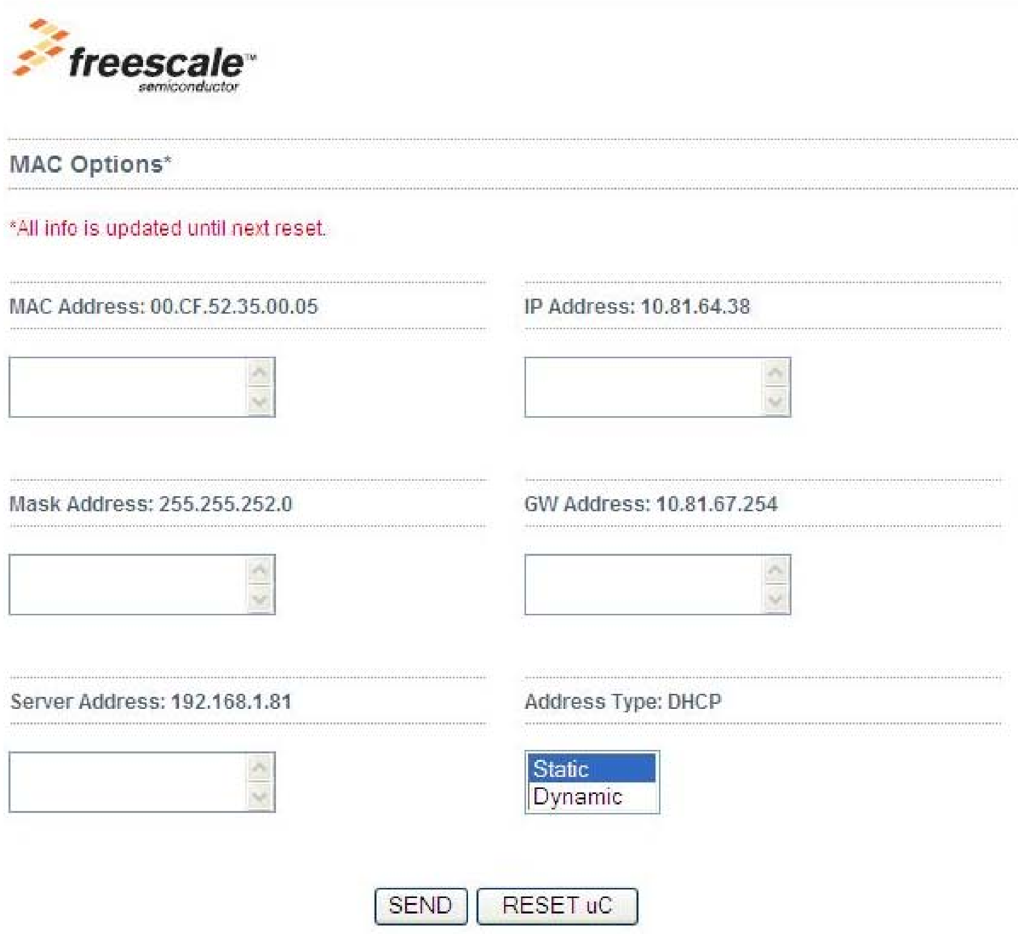
A web file stored in the internal memory is not limited to a specific number of SSI directives. A web page sent from the server to the client needs to include a content-length header showing the exact number of bytes of the web page. The web page length cannot be known when SSI directives are included. Due to the limited RAM memory on this MCU, the whole web page cannot be replaced and stored to find out the exact number of bytes to be sent to the client. This is resolved by using the transfer-encoding HTTP header option which is configured as "chunked". The content can be divided into chunks. Each chunk is prefixed by its size in bytes. A zero size chunk indicates the end of the response message.

As an example, Figure 5 shows values stored in the FLASH memory and requested by the HTML code using SSI directives. The information is replaced at runtime with the correct string stated in the http_ssi.h file.



```
<form method="POST" action="mac.cgi">
<div class="gbl">
<div class="hrw"><hr></div>
<h2>MAC Address: <!--#echo var="MACADD"--></h2
<div class="hrw"><hr></div>
```

Figure 5. SSI directives on original HTML source code



MAC Options*

*All info is updated until next reset.

MAC Address: 00.CF.52.35.00.05

IP Address: 10.81.64.38

Mask Address: 255.255.252.0

GW Address: 10.81.67.254

Server Address: 192.168.1.81

Address Type: DHCP

Static
Dynamic

Figure 6. Web page with replaced string using SSI method

3.2 Asynchronous JavaScript and XML (AJAX)

The web server implementation supports Asynchronous JavaScript and XML (AJAX). It is a web method that allows web page content to be updated in the background, or without updating the whole page. The web browser is responsible for regularly requesting a small file with the information that is required to be updated and shows it on specific parts of the web page. The web contentappears to be dynamic and changing at runtime. For this web server implementation, these small files must have an FSL extension and are written in SSI directive format, so they can be updated at runtime.

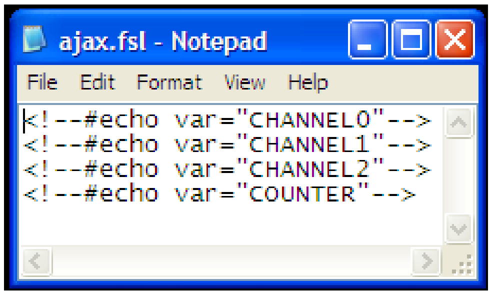
For this implementation, the web browser (client) must process all the transactions. The client requests the background file at regular times (1 second on this implementation) and the server must be able to supply this file.

The following steps must be followed to use AJAX:

- The HTML source code stored in the web server must have AJAX sentences. Go to ajax.htm for details on JavaScript language specific implementation.

- The requested file needs to be written in SSI directives, therefore the response to the client is correctly replaced inside the web server, hiding the details of replacement to the client.
- The web server must support the HTTP persistent connection feature present in the lwIP TCP/IP stack.

Figure 7 shows the SSI directives that are replaced for this specific AJAX implementation.



```
<!--#echo var="CHANNEL0"-->
<!--#echo var="CHANNEL1"-->
<!--#echo var="CHANNEL2"-->
<!--#echo var="COUNTER"-->
```

Figure 7. ajax.fsl file requested in background using AJAX method

Figure 8 shows how the information is dynamically updated on a web browser using AJAX.

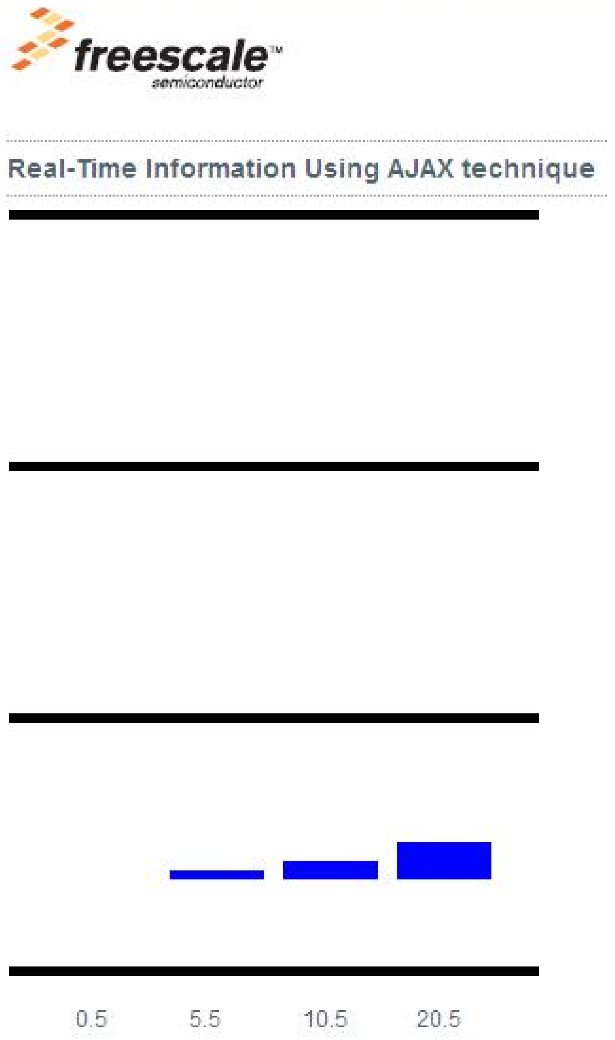
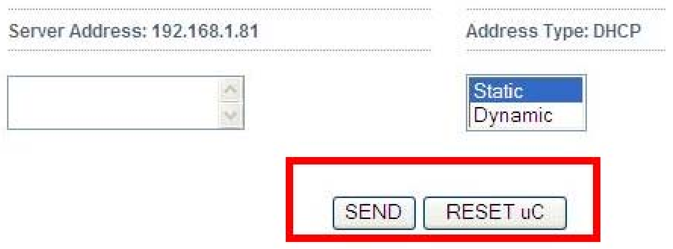


Figure 8. AJAX web page dynamically updated in background

3.3 Common Gateway Interface (CGI)

The web server supports CGI to run client requests by the server. The CGI functionality appears while executing a web-form action. See Figure 9 for details. The web-form is sent after pressing the SEND button.


Server Address: 192.168.1.81

Address Type: DHCP

Figure 9. CGI interface using web-form buttons

NOTE

To reset the board using the web page interface the following must be checked:

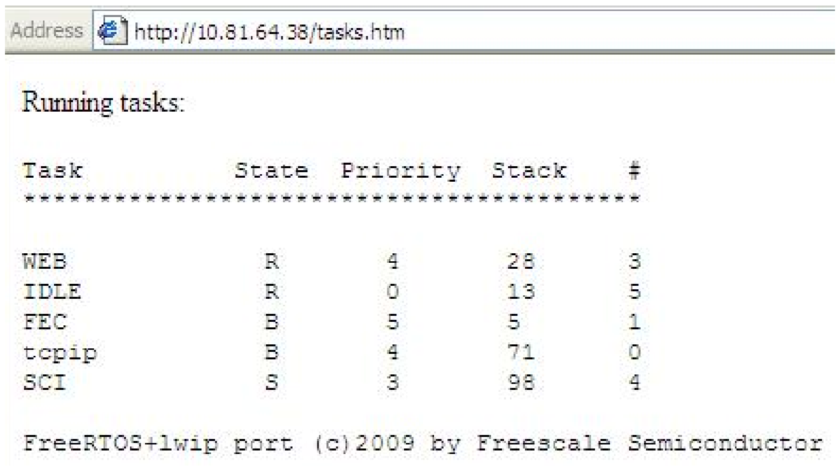
- Software is working in a standalone way (not using the debug interface).
- A single POR must be executed after each programming of the MCU to allow the system to support software resets.

Not following these steps before executing a web-form reset can lead to an unresponsive system that is allowed to resume operation only after an on-board reset.

3.4 Tasks Status

By typing [assigned IP address like http://10.81.64.38]/tasks.htm, the task status is viewable. This web page is useful to know the task stack consumption. The stack column shows the number of long words that are available for each task stack. If one of these numbers is near to zero, then the respective stack is near to overflow. Most of the time, an overflow in the stack means an overflowed area is overwriting other memory areas and the error can affect the whole system in unknown ways.

The following FreeRTOS port does not support a dedicated interrupt stack when switching to interrupts. Then the sum of all the interrupt stacks must be considered as delta and needs to be considered in each task, besides its own task stack space. A future version of FreeRTOS for ColdFire will provide user/supervisor support present in ColdFire architecture. This will allow switching to a dedicated interrupt stack during interrupt handling. The delta needs to be considered as part of the interrupt stack only, therefore reducing RAM consumption per task.



Running tasks:

```
Task           State  Priority  Stack   #
*******************************************

WEB              R       4       28     3
IDLE             R       0       13     5
FEC              B       5       5      1
tcpip            B       4       71     0
SCI              S       3       98     4

FreeRTOS+lwip port (c)2009 by Freescale Semiconductor
```

Figure 10. Tasks' status on the MCF51CN128

3.5 SD Card Support

The web page is implemented in a way that a file system can be added with an SD card. A file stored in the SD memory can then be displayed on a regular web browser window.

3.6 Limitations

Due to limited RAM memory, only one client must be connected to the web server at the same time. One way to manage multiple clients is to create a task for each new client. However, per each new task, a considerable amount of RAM memory is required for stacking purposes. If multiple clients are required, an upgrade to the MCF5223x or MCF5225x must be considered.

3.7 Principle of Operation

The software was 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 selection between either the 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 11. Code snippet for hardware change

4 Web Server Software

4.1 Software Architecture

Figure 12 shows how the web server is divided and what software blocks are used for this implementation.

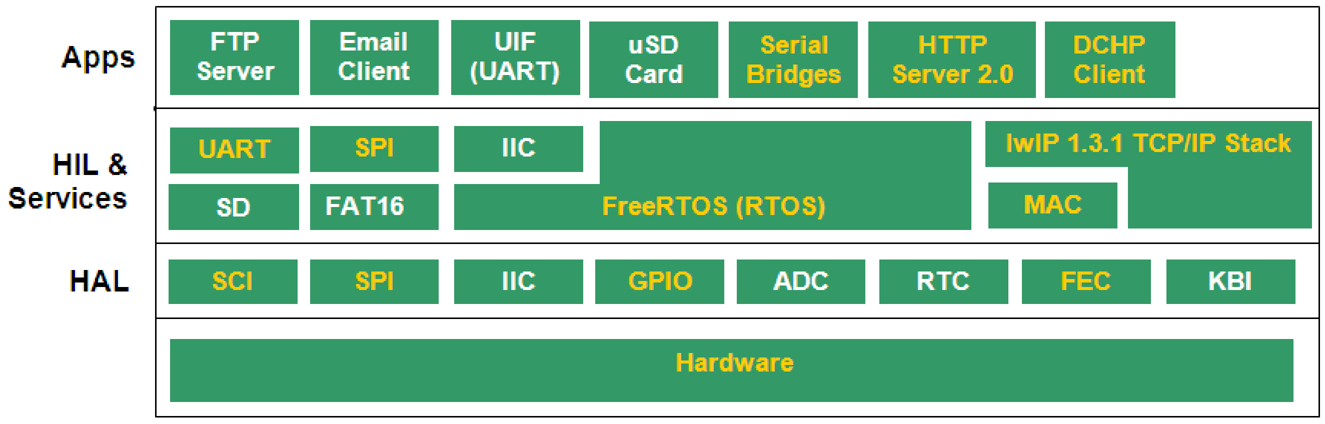


Figure 12. Software segmentation

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

4.2 Software Hierarchy

Figure 13 shows the software files hierarchy.

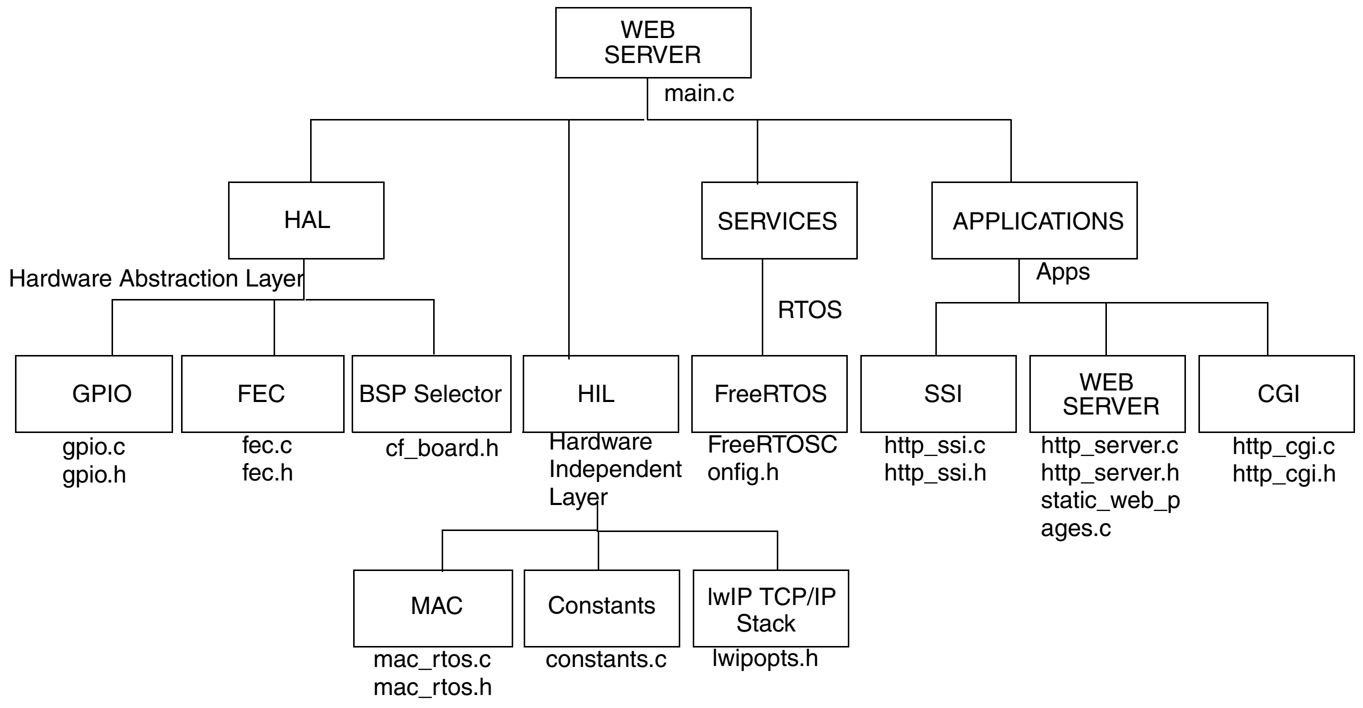


Figure 13. File implementation

Table 2. Software file descriptions

Layer	File Name	Description
Main	main.c	Enable and disable the tasks running on the MCF51CN128 MCU: in this case, the web server
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
	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
	constants.c	Structure containing all the default parameters after reset
	lwipopts.h	lwIP options to enable or disable services
	FreeRTOSConfig.h	FreeRTOS options to enable or disable services
Applications	http_cgi.c	Selector between different CGIs
	http_cgi.h	CGIs defined by the user and application
	http_ssi.c	Parser to replace SSI directives at runtime
	http_ssi.h	SSI defined by the user and application
	http_server.c	Manages GET and POST requests
	http_server.h	RTOS tasks' status and error messages
	static_web_pages.c	Contains web pages in an array form

4.2.1 Hardware Abstraction Layer (HAL) Implementation

The HAL is defined as the collection of software components that gives 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.

Figure 12 is a representation of the software blocks that are most important for the HAL. The HAL uses modules available in the MCF51CN128 MCU.

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

Figure 14 and Figure 15 are representations of the software blocks that are more important for the HIL. The HIL is the nearest software layer to applications using HAL software layers.

4.3 Socket Interface

Figure 14 shows the socket flow from the server and client perspective.

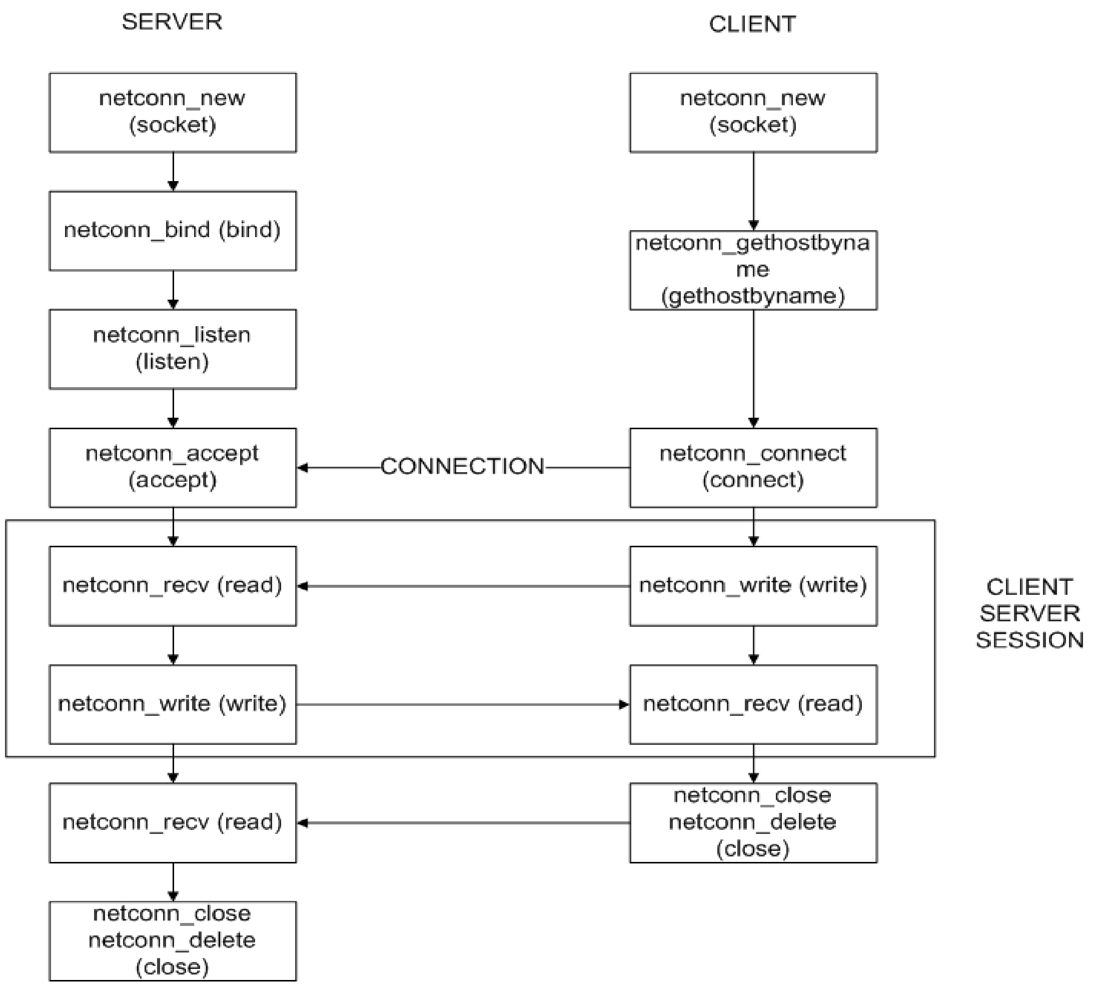


Figure 14. Netconn socket flow diagram

Figure 15 shows the web server flow diagram. It uses a non-blocking accept function to switch from one HTTP session to another. HTTP sessions are important because a single web browser can connect to the web server port 80 by using more than one connection or session at the same time. This improves performance and takes advantage of keeping connections open by using the keep-alive feature.

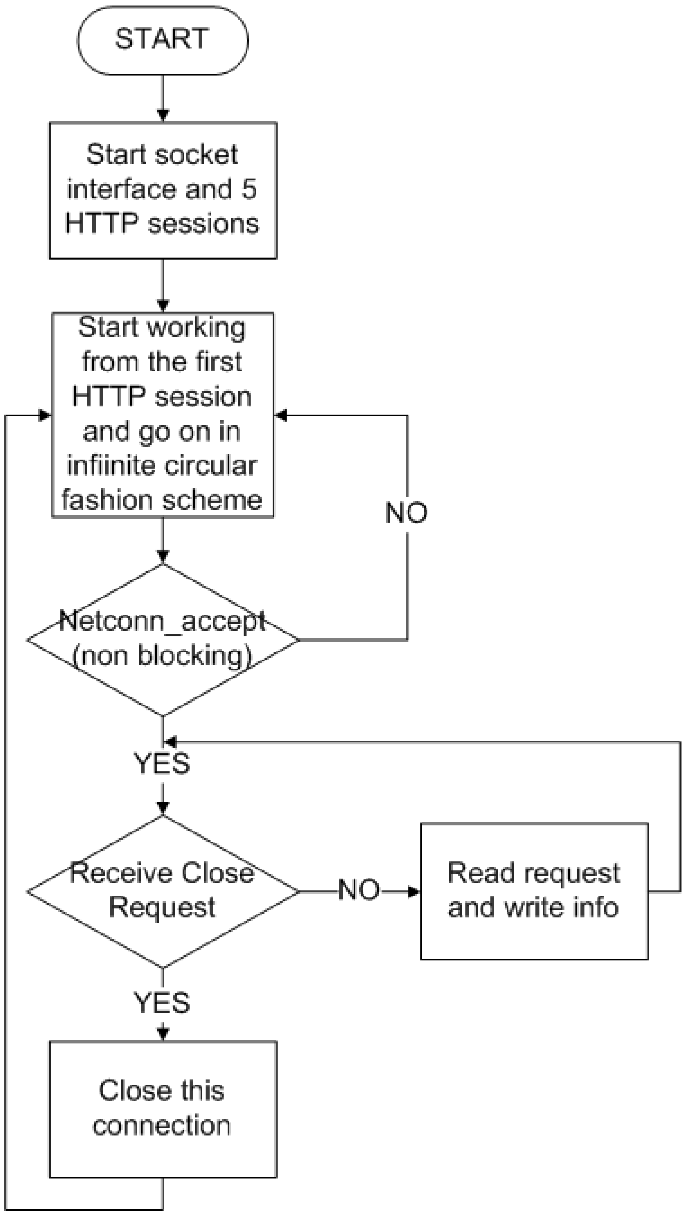

START
Start socket interface and 5 HTTP sessions
Start working from the first HTTP session and go on in infiinite circular fashion scheme
NO
Netconn_accept (non blocking)
YES
Receive Close Request
NO
Read request and write info
YES
Close this connection

Figure 15. Webserver flow diagram

5 Web Server API

This section shows the main functions used for the web server application. This task needs to be added using the thread support (sys_thread_new) of lwIP instead of using the task creation of FreeRTOS (xTaskCreate).

The way the web server task must be called is as follows:

```
( void )sys_thread_new("WEB", HTTP_Server_Task, NULL, WEBSERVER_STACK_SPACE,
HTTP_TASK_PRIORITY );
```

Details about the function are provided:

Syntax:

```
      void
      HTTP_Server_Task( void *pvParameters )
```

Description:

```
      /**
       * Start an embedded HTTP server Task: 1 client and multiple files per transfer
       *
       * @param none
       * @return none
       */
      Starts the Web server
```

The web server is capable of supporting GET and POST requests from a web client. The support given to these requests is enough and must not be changed.

6 Customization

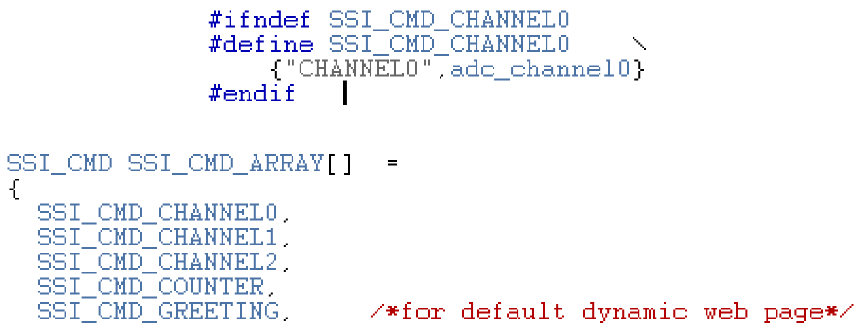
For customization, only the following files must be modified for a change in software or hardware:

Table 3. Customization

File Names	Description
cf_board.h	Used to point to a new BSP, new HAL software drivers
lwipopts.h	lwip configuration file. Enable/disable tcp/ip options
gpio.c/gpio.h	Change GPIO used for all modules in the MCU
FreeRTOSConfig.h	FreeRTOS user configuration file. Enable/Disable features
static_web_pages.c	Contains web pages
http_server.h	Presents error messages
http_ssi.h	The list of SSI directives and their string replacements
http_cgi.h	The list of CGI applications

For a standard web server, only the following files are required to be changed:

- Static_web_pages.c — This C file contains each file on an array-scheme. To add or remove new files, the array contains only the file's data. The HTTP header is generated at runtime, therefore there is no need to create it at the compile time.
- http_ssi.h — To add new SSI directives, SSI_CMD_CHANNEL0 must be followed as a guide. The first element states the string to be used as ID and the second function to be called when the string is found in the HTML source code. This function must return the string to be used to replace the ID string. Finally, add it to the array of SSI directives called SSI_CMD_ARRAY.

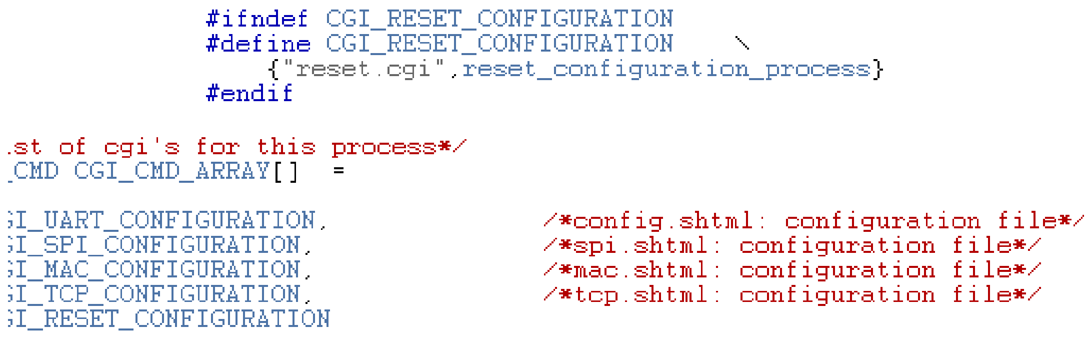


```
                #ifndef SSI_CMD_CHANNEL0
                #define SSI_CMD_CHANNEL0    \
                    {"CHANNEL0",adc_channel0}
                #endif

SSI_CMD SSI_CMD_ARRAY[]  =
{
  SSI_CMD_CHANNEL0,
  SSI_CMD_CHANNEL1,
  SSI_CMD_CHANNEL2,
  SSI_CMD_COUNTER,
  SSI_CMD_GREETING,      /*for default dynamic web page*/
```

Figure 16. Example Code 1 — SSI Example

- http_cgi.h — To add new CGI routines, CGI_RESET_CONFIGURATION must be followed as a guide. The first element states the name of the CGI to be used for the POST request and the second element states the function to be called for the found CGI. This function will receive as a parameter, the request received from the POST request. Finally, add it to the array of CGI routines called CGI_CMD_ARRAY.



```
             #ifndef CGI_RESET_CONFIGURATION
             #define CGI_RESET_CONFIGURATION    \
                 {"reset.cgi",reset_configuration_process}
             #endif

.st of cgi's for this process*/
_CMD CGI_CMD_ARRAY[]  =

;I_UART_CONFIGURATION,              /*config.shtml: configuration file*/
;I_SPI_CONFIGURATION,               /*spi.shtml: configuration file*/
;I_MAC_CONFIGURATION,               /*mac.shtml: configuration file*/
;I_TCP_CONFIGURATION,               /*tcp.shtml: configuration file*/
;I_RESET_CONFIGURATION
```

Figure 17. Example Code 2 — CGI Example

- cf_board.h — If a change of the MCU is required to upgrade to MCF5223x or MCF5225x, cf_board.h must point to the new low level drivers to be used by the application. Use MCF51CN128 for details on how MCU selection is used on the CodeWarrior project.

Additional customizations are not required because the web server has enough services with the provided RAM memory. The application is fine-tuned to provide good performance with limited resources.

7 Conclusion

This document described how a web server can be implemented in the MCF51CN128 using only internal memory in the smallest package (48-pin QFN). The document also described some HTTP services like: SSI, Forms, Ajax, and Dynamic replacement using lwIP as the TCP/IP stack and FreeRTOS as the RTOS. The software showed that all web services mentioned can be implemented in less than 128 KB of ROM and 24 KB of RAM.

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 the AN3928SW.
- 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 AN3928SW.zip from www.freescale.com.
- For more information regarding software or hardware refer to www.freescale.com/support.

THIS PAGE IS INTENTIONALLY BLANK

Document Number: AN3928
Rev. 0
08/2009