Freescale Semiconductor
Application Note

Document Number: AN4022
Rev. 1, 12/2009

IP Camera and USB Snapshot with MQX

by: Juan Cazares,
Luis Garabito,
Applications Engineering
Freescale Technical Support

1 Introduction

This application note is intended to demonstrate web server, USB Host Stack for mass storage, and UART driver with interrupts of Freescale MQX RTOS capabilities. This application note is based on the M52259DEMOKIT board.

The Freescale MQX RTOS has support for many MCF5225x peripherals, but this application note is focused on the UART, Freescale MQX RTCS, and USB modules.

You can find the source code for this document in the AN4022SW.zip file. This document does not include a full description of Freescale MQX. For more information, please visit www.freescale.com/mqx.

2 Overview

Everyone has the necessity to be informed at any time. With the help of Internet, it is possible. For this reason,

Contents

freescale semiconductor

many applications like security, video conference, health care systems, and so on, are including internet connections. One of the most common applications is IP cameras. These devices allow the user to watch a remote location. This can be used as security, conference, vigilance, and more applications like those.

Figure 1 shows a block chart of the whole system. As mentioned before, the application highlights the USB, UART, and Ethernet modules.

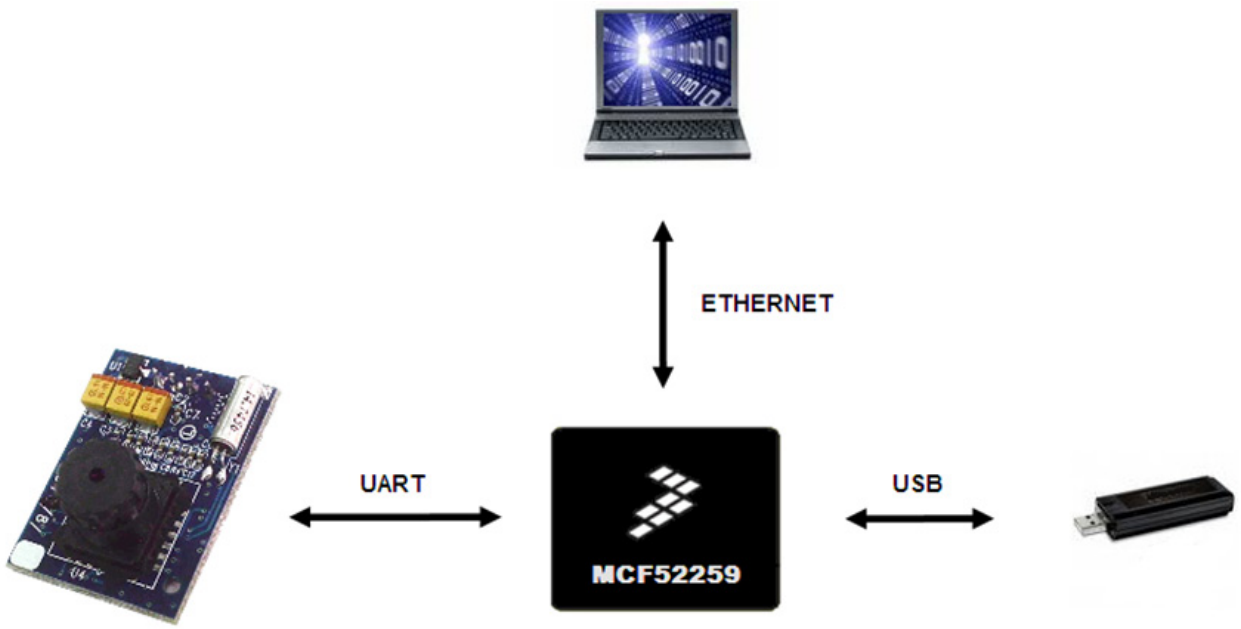

Figure 1. Block Chart

3 Hardware Settings

A demonstration project was developed to show the Freescale MQX capabilities. This project runs on the Freescale M52259DEMOKIT board. The demonstration project uses a serial VGA camera (PN: C328R) to take video and the USB mass storage key to store snapshots taken with the serial camera.

The C328R is VGA camera module that performs as a JPEG compressed still camera. It can be used to send out a snapshot command from the host in order to capture a full resolution single-frame still picture. The picture is then compressed by the JPEG engine and transferred to the host through serial port.

Following is the hardware description to interface the C328R camera to the M52259DEMOKIT board. The C328R camera requires only +3.3 V, GND, UART-Tx, and UART-Rx. These signals are available on the J4 connector called MCU PORT. See the connections in Table 1 and Figure 2.

Table 1. Hardware settings

MCU PORT pin	MCU PORT signal	C328R camera
1	+3.3 V	VCC
3	GND	GND
9	UTXD2/CANTX	RX
11	URXD2/CANRX	TX

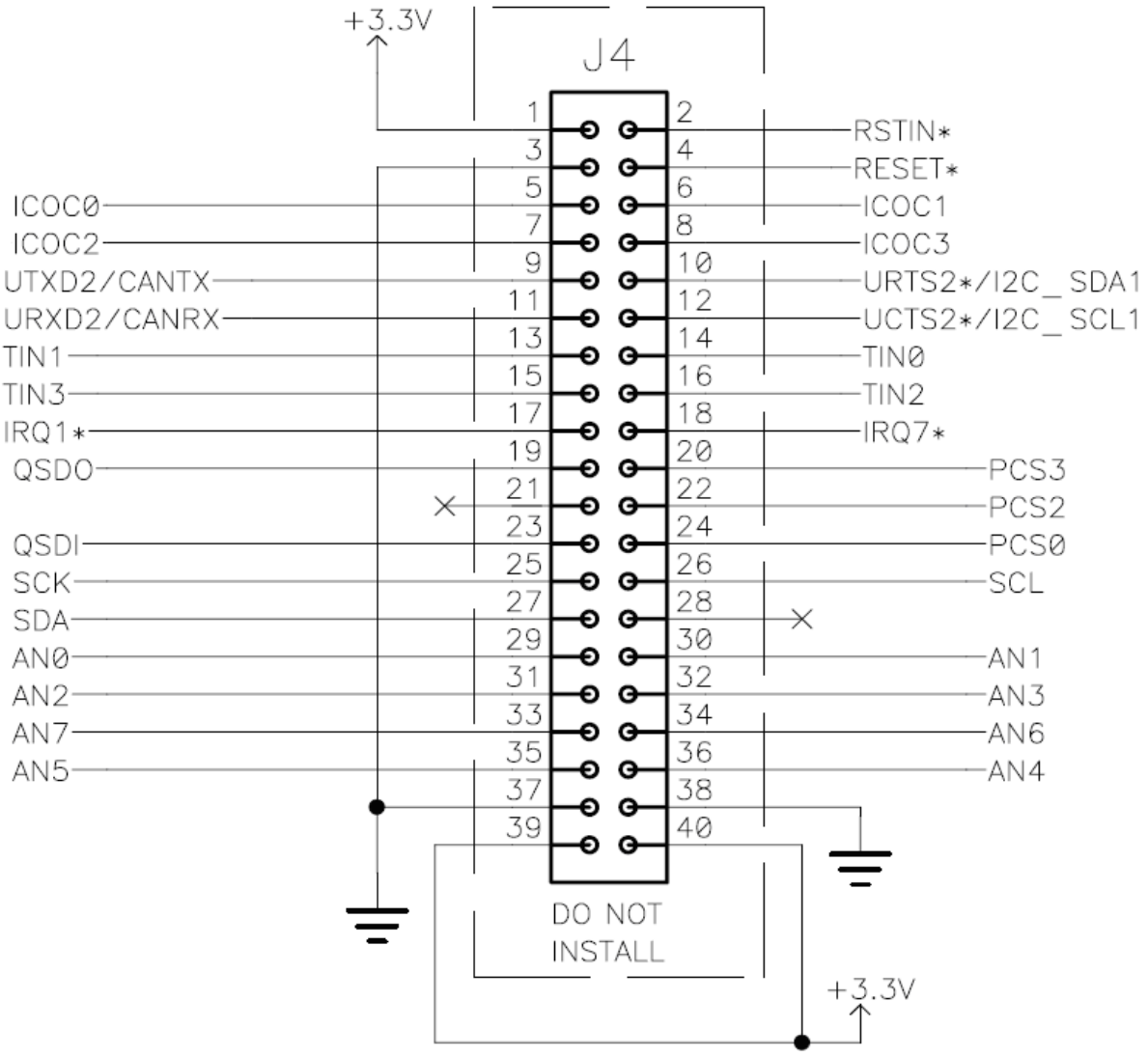

Figure 2. MCU PORT

4 Application

The application has three tasks.

- **task_init**—This is an auto-start task. It initializes the networking and start the camera_task. The task_init also handles the snapshot-to-USB function.
- **camera_task**—This task handles the communication with the camera.
- **USB_task**—This task initializes the USB, and monitors the plug and unplug operation of USB memory keys.

The figure 3 shows the flow charts of the tasks.

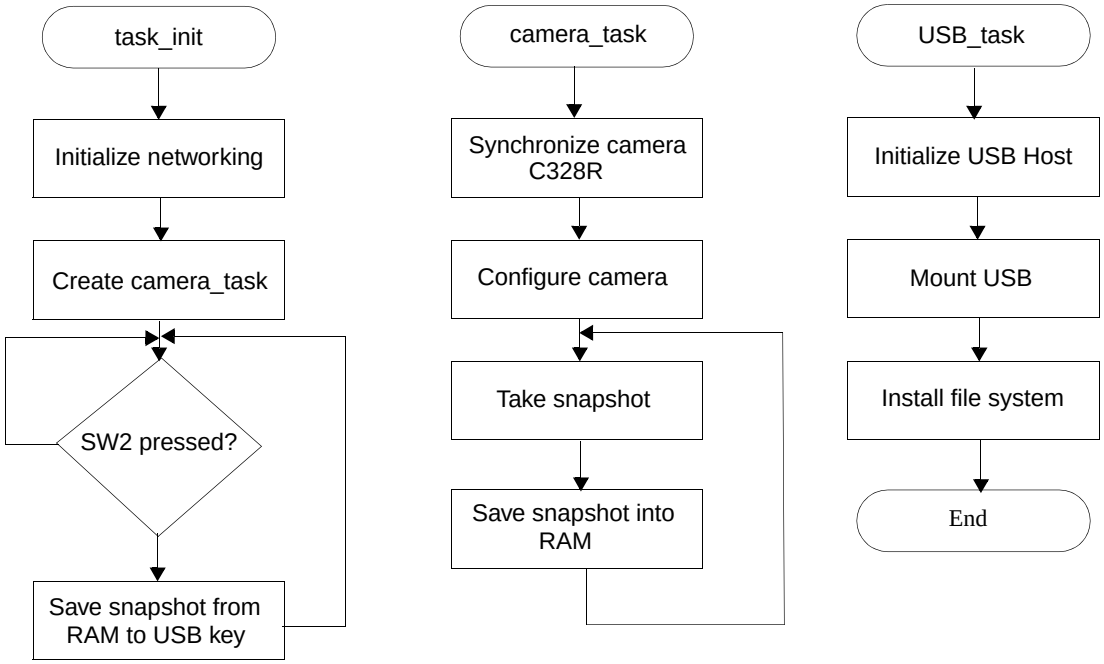

Figure 3. Task running in demo application

4.1 UART configuration

The application uses two UART ports. The UART0 is used as console support where the application status and some other messages are displayed. The UART2 is used for the application to communicate with the C328R camera. While the UART0 works by poling, the UART2 must be configured to send and receive data using interrupts. The reason of the interrupts usage is the C328R camera timing is very critical.

Because the default UART2 setting is poling, some macros into user_config.h file must be changed to use the UART with interrupts. The user_config.h file is allocated at the path where the MQX OS is installed. Commonly the path is C:/program files/Freescale/Freescale MQX 3.2/config/M52259DEMO/config.h

The following code enables the UART2 port to work with interrupts. This code has to be placed in the user_config.h file. In Freescale MQX, the names TTYA, TTYB and TTYC are the respective alias for the UART0, UART1, and UART2 ports.

```
#define BSPCFG_ENABLE_TTYC       0          //interrupt
#define BSPCFG_ENABLE_ITTYC      1
#define BSPCFG_UART2_BAUD_RATE   19200
```

It is necessary to recompile all the MQX libraries to allow the modifications in the user_config.h file take effect the next time the application is compiled. The MQX libraries are compiled in a separated CodeWarrior project, so to recompile the MQX libraries, the project build_libs.mcp has to be open and compiled again. After this process is done, the changes in the libraries will be reflected in the application.

The build_libs.mcp project is allocated at the path where the MQX OS is installed. Commonly the path is C:/program files/Freescale/Freescale MQX 3.2/config/M52259DEMO/ build_libs.mcp.

4.2 RTCS Configuration

The TCPIP stack is configured to start the DHCP client. This helps to obtain an IP address from the DHCP server in the LAN where the board is connected. An IP address assigned by a DHCP server is used to access the web server from anywhere in the LAN.

4.3 Web Server

This demo application uses the web server included in Freescale MQX RTCS. This web server is used to get access to the image captured by the C328R camera. The web server helps to post this image to a HTML web page that can be accessed by user through internet or LAN access. For this demo, the web server was modified to add a mutual exclusion to the memory area that stores the JPG image. Just like was mentioned before, the camera task is saving the image in a buffer in RAM. This buffer is also accessed by the web server to send image to the web browser client that is connected. For that reason, the Mutual Exclusion (MUTEX) is needed. This prevents any strange behavior. The mutual exclusion synchronizes the camera task and the web server task to have only one task at the time of accessing the memory storing the JPG image.

The MUTEX is used to synchronize writing and reading operations over the image buffer to protect the memory from any corruption. While the Camera task writes to the memory, the web server cannot reads it and vice versa.

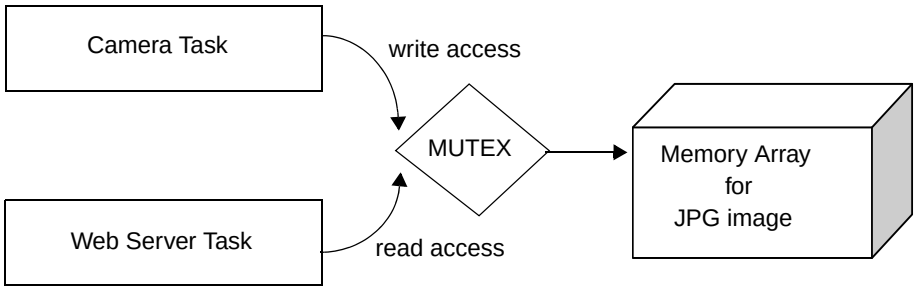

Figure 4. MUTEX synchronization for Camera, and Web Server tasks

To ensure the mutual exclusion, a validation was added in the web server. If the file requested by the web browser is MCF5225X.jpg, then the web server locks the MUTEX to ensure that it can be sent easily. After the file was sent, the web server unlocks the MUTEX to let the camera task continue with any update to the file in RAM. The files modified for this implementation are the httpd.c and httpd_supp.c. These two files can be found in the path C:\Program Files\Freescale\Freescale MQX 3.2\rtcs\source\httpd. The file cam_mutex.h is a new module added to define the MUTEX. This file must be saved in the path C:\Program Files\Freescale\Freescale MQX 3.2\rtcs\source\httpd. The task_init is responsible for the initialization and creation of the MUTEX. If there is any problem creating the MUTEX, then none of the task will start its execution and the MQX will exit.

After all the modifications were done for the web server, it is necessary to recompile all the MQX libraries. This step was explained in previous steps where the UART with interrupts was enabled.

4.4 HTML

The web server displays HTML files. These files were converted from HTML files to hexadecimal arrays. If other elements are shown in the HTML web page, then these are also converted into hexadecimal arrays. The tool **mktfs.exe** is provided with Freescale MQX to convert these files into hexadecimal arrays. These arrays are then stored in flash as a trivial file system that can be used by the web server to display HTML contents.

In this demo, the web server is displaying two HTML files; the mqx.html file is set as the web page that will be loaded when a web browser reaches our IP address. The mqx.html file includes the mqx_cam.html. The mqx_cam.html is refreshing its contents, the image from the camera; every 500 milliseconds. The camera takes from 500 milliseconds to 800 milliseconds to take a snapshot, send it to the MCU, and store it in RAM. The time fluctuates depending on the size of the image. The image size depends on the color saturation and resolution of the snapshot.

The mqx.html uses the following html code to show the mqx_cam.html file:

```
<center><iframe src="mqx_cam.html" width="30%" height="300"></iframe></center>
```

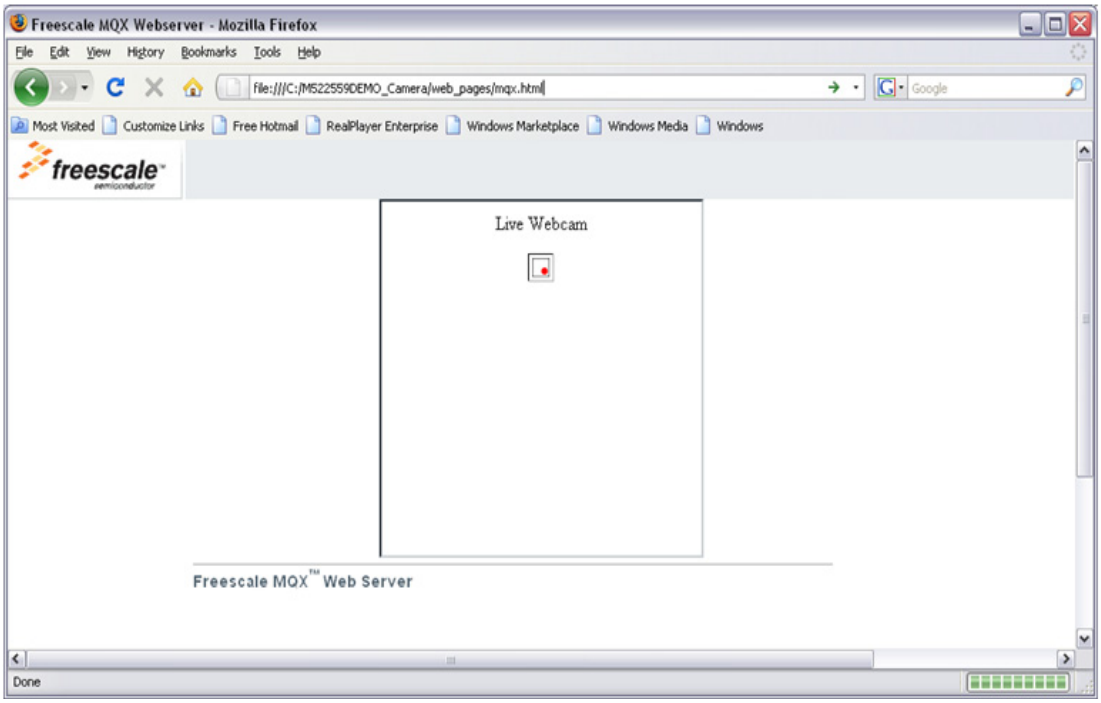

Figure 5. View of the mqx.html

The mqx_cam.html uses a code to refresh the image every 500 milliseconds as follows:

```
<meta http-equiv="refresh" content="0.5">
```

Finally, the mqx_cam.html shows the image with the folowing code:

```
<p align="center"><img src="MCF5225X.jpg" name="image" id="webcamimage"></p>
```

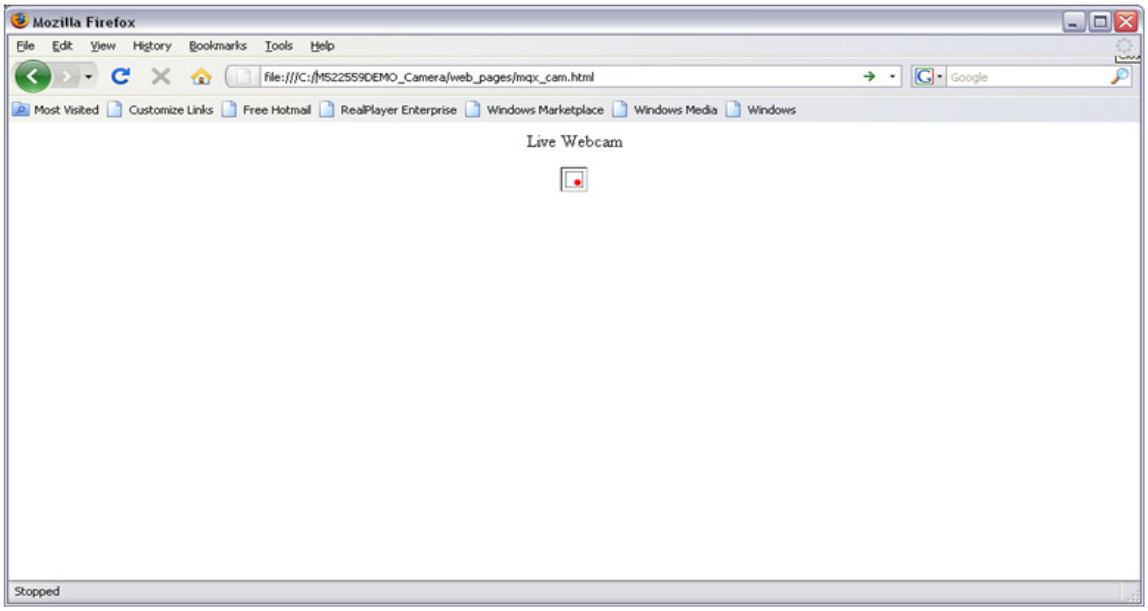

Figure 6. View of the mqx_cam.html

The file mcf5225x.jpg is the file updated in RAM. The mcf5225x.jpg was assigned to RAM instead of flash memory when the trivial file system was defined to make an update easier and faster.

The intention of two HTML files is to have a frame where all the presentation contents like logos, titles, CGI, and other HTML components can e added for better presentation of the web page.

4.5 USB Mass Storage Feature

This demo also includes the functionality to store a snapshot in a flash memory every time the SW2 button is pressed.

A task is created to configure the USB host for mass storage. The code works just like the web_hvac demo (C:\Program Files\Freescale\Freescale MQX 3.2\demo\web_hvac). When the USB memory key is plugged in the USB port, MQX USB host stack sets up the device, and install the partition and file system from the USB memory key. At this moment, it is possible to read and write files into the USB memory key.

After the task_init initialize the MUTEX and the networking, it opens the GPIO driver to configure the SW2 button in the M52259DEMO board as input. The task_init enters into a loop to poll the SW2 button. If the SW2 button is pressed, the task_init locks the MUTEX to have exclusion access to the RAM memory where the JPG image is stored. The camera can not write this way to the memory and discard any memory corruption. After the MUTEX is locked, the task_init verifies that a filesystem is available in the USB memory key and creates a file there. It copies the contents of the memory into this new file and closes the file. It releases the MUTEX to let the camera task and web server to continue with their activities.

The MUTEX is used to synchronize the access of the three tasks to the memory resource and this is called mutual exclusion.

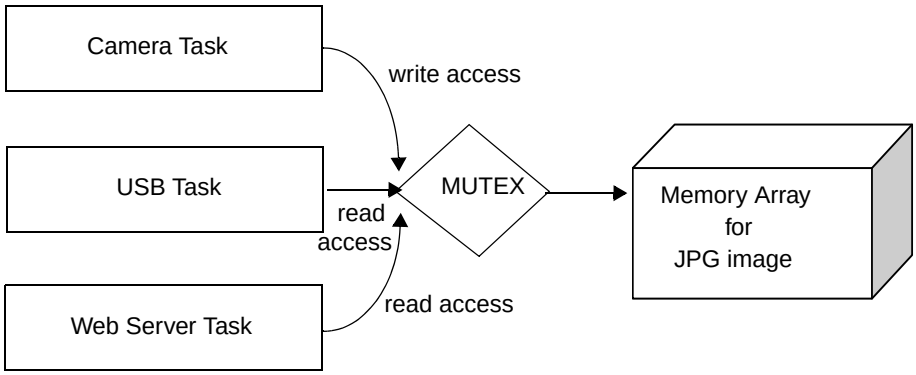

Figure 7. MUTEX synchronization for Camera, USB, and Web Server tasks

5 References

For more details, refer to the following documents at www.freescale.com.

- Application note AN3902, “How to Develop I/O Drivers for MQX”
- Application note AN3907, “Using MQX Libraries”

THIS PAGE IS INTENTIONALLY BLANK

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: AN4022
Rev. 1
12/2009

freescale™
semiconductor