Freescale Semiconductor
Application Note

Document Number: AN4677

Output Redirection of the printf() Function in MCU CodeWarrior 10.x product

1. Introduction

The default MSL configuration for Kinetis family processors provides the stdout, stderr, and stdin file streams through serial I/O communication circuit on most evaluation boards. The I/O routines that refer to the standard streams implicitly, such as printf() function are available in the CodeWarrior 10.x product. To use the MSL console I/O functions, you have to include a special serial I/O library into the project and ensure that the hardware is properly initialized.

Contents

2. Use Output of the printf() function in CodeWarrior 10.x product

To use the output of the printf() function in the CodeWarrior 10.x product, you need to do:

- Create a New Project
- Import Resources into the Project
- Define New Macro in the Project
- Initialize System Peripherals

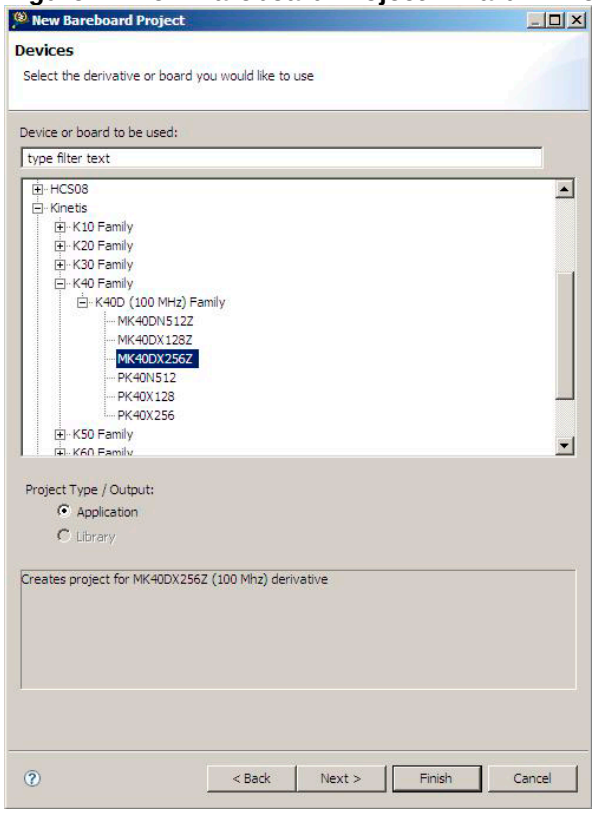
2.1. Create a New Project

Note: The official release of CodeWarrior 10.x product has to be installed on the computer.

To create a new project in CodeWarrior 10.x development environment perform the following steps:

1. Start the CodeWarrior 10.x development environment.
2. Select File > New > Bareboard project from the IDE main menu bar.
   The Create an MCU Bareboard Project page of the New Bareboard Project wizard appears.

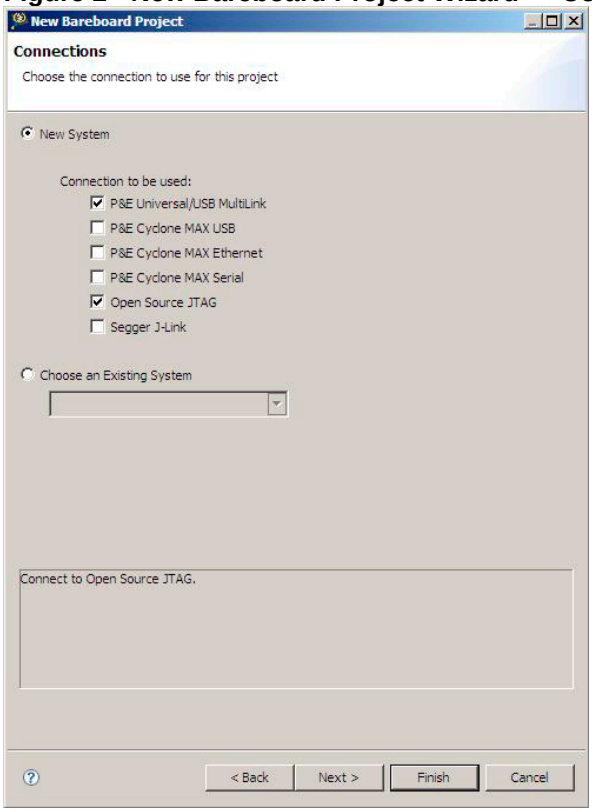
Figure 1 - New Bareboard Project Wizard — Devices Page



3. Specify a name for the new project. For example, enter the project name as "*printf_test*".
4. Click **Next.**
   The **Devices** page appears.
5. Expand the desired tree control and select the derivative or board you would like to use.
   For example, select **Kinetis > K40 Family > K40D (100 MHz) Family > MK40DX256Z** as shown in Figure 1.
6. Click **Next.**
   The **Connections** page appears.

7. Select the connections **P&E Universal/USB multilink** and **Open Source JTAG** as shown in Figure 2.

Figure 2 - New Bareboard Project Wizard — Connections Page



8. Click **Finish.**
   The wizard creates a new simulator project for the S08 architecture according to your specifications. You can access the project from the **CodeWarrior Projects** view in the **Workbench** window.

2.2. Import Resources into the Project

Import the system resources and source code files from the **Common**, **CPU**, **Drivers** and **Platforms** folders into the project as defined in Table 1.

Table 1. The table of required resources and source code

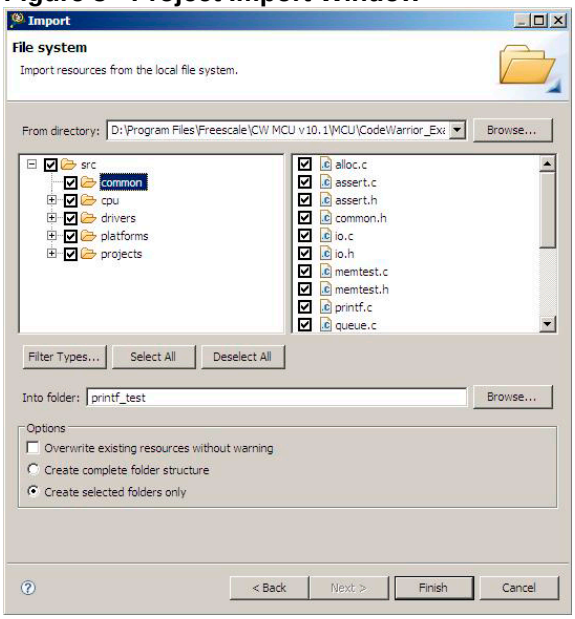
Common	CPU	Drivers	Platforms
common.h	arm_cm4.c	mcg.c	k40_tower.h
io.c	arm_cm4.h	mcg.h	k60_tower.h
io.h	dma_channels.h	uart.c	
printf.c	sysinit.c	uart.h	
uif.c	sysinit.h	wdog.c	
uif.h		wdog.h	

To import the sources into the created project, perform the following steps:

1. Set the path of local sources and files you want to import.
   a) Select File > Import > General > File System.
   b) Click **Next**.
   c) Click **Browse** and select the following file path.
   `current_install_directory_of_CodeWarrior_MCU_v10.3_tool}\MCU\CodeWarrior_Examples\Kinetis_Examples\src\`

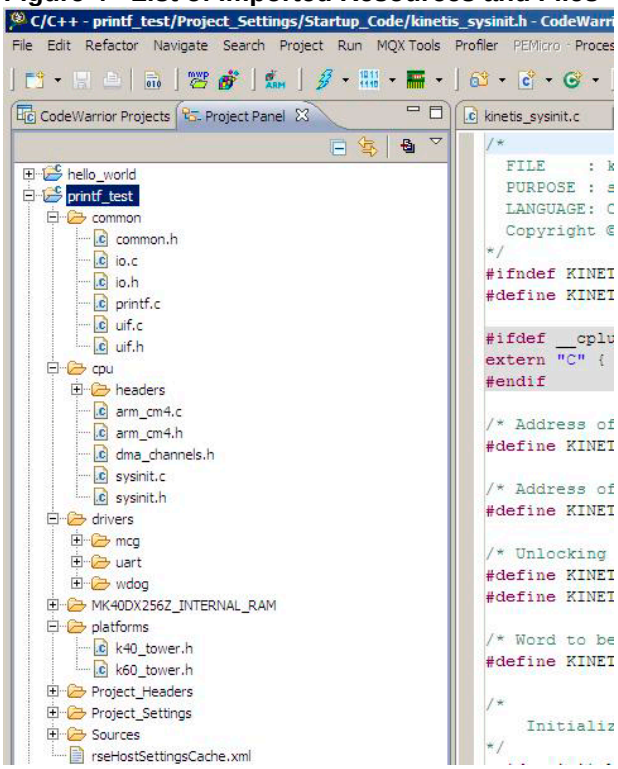
   The required resources and source files will appear in the **Import** window of the project wizard, as shown in Figure 3.

Figure 3 - Project Import Window



2. Click **Finish** to finalize the import process of required resources and source files into the created project. There are the required resources and source files imported into the currently used project, as shown in Figure 4.

Figure 4 - List of Imported Resources and Files



3. Set the path to the system resources in the **Properties** window of the new project.
   a) Select the project you want to add the system resources to.
   b) Select C/C++ Build > Settings > Tool settings > ARM_Compiler > Input.
   The Properties <for project> dialog box appears.
   c) Add the following resource paths in the include search paths list:
      - `"${PROJECT_LOC}/drivers/wdog"`
      - `"${PROJECT_LOC}/common"`
      - `"${PROJECT_LOC}/cpu"`
      - `"${PROJECT_LOC}/drivers/adc16"`
      - `"${PROJECT_LOC}/drivers/uart"`
      - `"${PROJECT_LOC}/cpu/headers"`
      - `"${PROJECT_LOC}/drivers/mcg"`
      - `"${PROJECT_LOC}/platforms"`
      - `"${ProjDirPath}/Project_Headers"`
      - `"${MCUToolsBaseDir}/ARM_EABI_Support/ewl/EWL_C/include"`

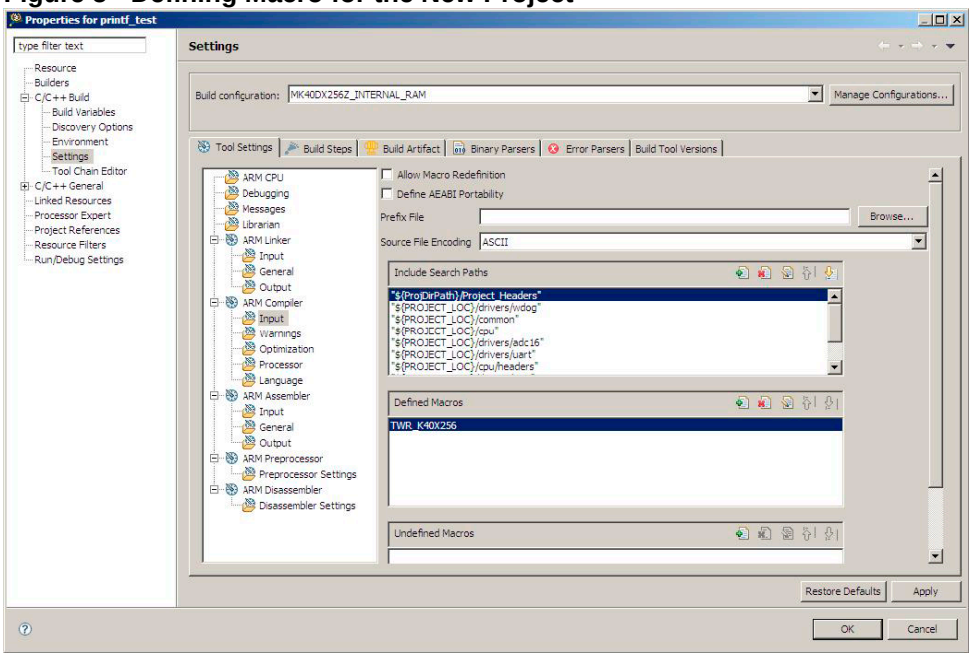
4. Check the current paths settings of source files in the **Include Search Paths** list in the **Properties** window of the project.

2.3. Define New Macro in the Project

For the new project to function correctly, you need to define the **TWR_K40X256** macro in the project:

1. Select the project you want to add the resources to.
2. Select C/C++ Build > Settings > Tool settings > ARM_Compiler > Input.
   The Properties <for project> dialog box appears.
3. Insert the new macro named *"TWR_K40X256"* into the list of **Defined Macros** as shown in Figure 5.

Figure 5 - Defining Macro for the New Project



4. Click **OK** to confirm the changes and create the new macro in the project.
   The *"TWR_K40X256"* named macro in defined in the project.

2.4. Initialize System Peripherals

To initialize the system peripherals into the source code of the created project, perform the following steps.

1. Open kinetis_sysinit.c source file from Project settings > Startup code folder in the CodeWarrior Projects view.
2. Update the source code of the application like below.
   a) Add **#include "sysinit.h"** command into the **kinetis_sysinit.c** source file, as shown in Listing 1.

Listing 1. Update of kinetis_sysinit.c source file - add include command

```
#include "kinetis_sysinit.h"
#include "derivative.h"
#include "sysinit.h"

typedef void (*const tIsrFunc)(void);
typedef struct {
uint32_t * __ptr;
tIsrFunc __fun[119];
} tVectorTable;
```

b) Add the **sysinit()** function call into the source code of **__init_hardware()** system function in **kinetis_sysinit.c** source file. Modify the source code of **kinetis_sysinit.c** file, as shown in Listing 2.

Listing 2. Update of kinetis_sysinit.c source file - call system initialization function

```
void __init_hardware()
{
SCB_VTOR = (uint32_t)__vector_table;
/*Set the interrupt vector tableposition */
/* Disable the Watchdog because it may reset the core before entering main(). There are 2
unlock words which shall be provided in sequence before accessing the control register.*/
*(volatile unsigned short *)KINETIS_WDOG_UNLOCK_ADDR KINETIS_WDOG_UNLOCK_SEQ_1;
*(volatile unsigned short *)KINETIS_WDOG_UNLOCK_ADDR = KINETIS_WDOG_UNLOCK_SEQ_2;
*(volatile unsigned short *)KINETIS_WDOG_STCTRLH_ADDR = KINETIS_WDOG_DISABLED_CTRL;

sysinit();
}
```

3. Build the new project and run the application.

3. Application Functionality Tests

The following tests are a practical example using and redirecting the output of the **printf()** function in the CodeWarrior development environment. You can use the console or the terminal for the text output of the function. The communication channel is serial link or OS-JTAG. You can activate one of the following options to select a serial circuit, as shown in Listing 3.

Listing 3. kinetis_sysinit.c Source File

```
/*
 * Select the serial port that is being used below. Only one of the
 * options should be uncommented at any time.
 */
#define SERIAL_CARD  // this is the default option, so it is uncommented
#define OSJTAG    // use this option for serial port over the OS-JTAG circuit
```

To use OS-JTAG as standard output it is necessary to change definition in the **k40_tower.h** or **k60_tower.h** include file. To use the **printf()** function to print the output text using the OS-JTAG circuit you have to add the following definition into the **k60_tower.h** or the **k60_tower.h** header file respectively, as shown in Listing 4.

Listing 4. Update kinetis_sysinit.c source file - Using OS-JTAG as standard output

```
/*
 * Select the serial port that is being used below. Only one of the
 * options should be uncommented at any time.
 */
//#define SERIAL_CARD  // this is the default option, so it is uncommented
#define OSJTAG     // use this option for serial port over the OS-JTAG circuit
```

To use serial circuit as standard output is it necessary to change definition in the **k40_tower.h** or the **k60_tower.h** include file. To use the printf() function to print output using the serial circuit you have to add the following definition in the **k60_tower.h** or the **k60_tower.h** header file respectively, as shown in Listing 5.

Listing 5. Update kinetis_sysinit.c source file - Using serial circuit as standard output

```
/*
 * Select the serial port that is being used below. Only one of the
 * options should be uncommented at any time.
 */
#define SERIAL_CARD   // this is the default option, so it is uncommented
//#define OSJTAG    // use this option for serial port over the OS-JTAG circuit
```

If the printf() function is declared in the **io.h** or **common.h** header files you can print the output text on a terminal only. To get the function output on eclipse console you need to use the printf() function declared in the **stdio.h** header file instead of the function included in **io.h** or **common.h,**

NOTE: Do not compile **printf.c** source code file.

The source code to use the printf() routine to print the output text on a console is shown in Listing 6.

Listing 6. Update kinetis_sysinit.c source file - Print the output text on a console

```
/*
 * Select the serial port that is being used below. Only one of the
 * options should be uncommented at any time.
 */
//#define SERIAL_CARD   // this is the default option, so it is uncommented
#define OSJTAG
//#include "common.h"
//#include "io.h"
#include <stdio.h>
#include "derivative.h" /* include peripheral declarations */
int main(void)
{
   int counter = 0;

   printf("Hello (Kinetis) World in 'C' from MK40DX256Z derivative! \n\r");

   for(;;) {
     counter++;
   }

   return 0;
```

```
}
```

The source code of the printf() function that allows you to print the output text on terminal is shown in Listing 7.

Listing 7. Update of kinetis_sysinit.c source file - Print the output text on a terminal

```
/*
 * Select the serial port that is being used below. Only one of the
 * options should be uncommented at any time.
 */
//#define SERIAL_CARD   // this is the default option, so it is uncommented
#define OSJTAG
//#include "common.h"
#include "io.h"
//#include <stdio.h>
#include "derivative.h" /* include peripheral declarations */

int main(void)
{
   int counter = 0;

   printf("Hello (Kinetis) World in 'C' from MK40DX256Z derivative! \n\r");

   for(;;) {
          counter++;
   }

   return 0;
}
```

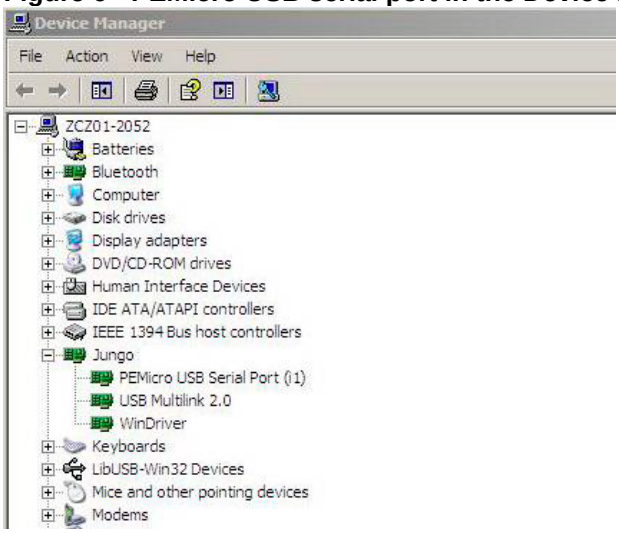
To use the OS-JTAG circuit for communication and to redirect the output of the **printf()** function to PEmicro terminal, you will need to install the PEmicro terminal application

To install the terminal application, perform the following instructions:

1. Install the driver kit for OS-JTAG by using **PE_Micro_Kinetis_Tower_Toolkit.exe** file from the installation source.

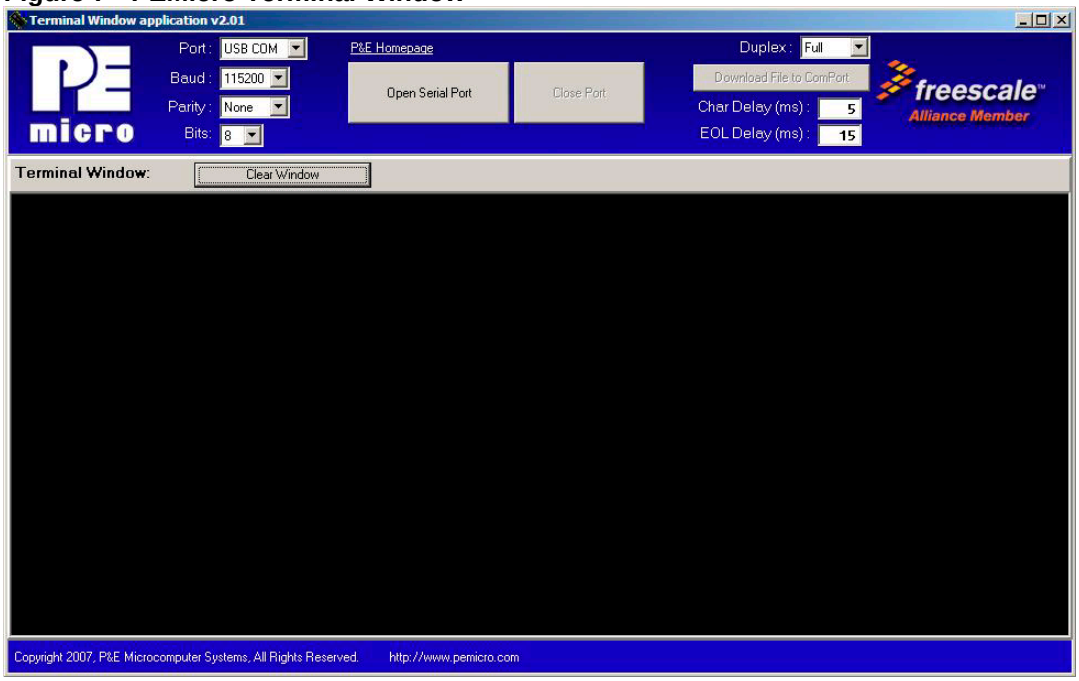
2. To activate drivers in MS Windows operating system, you have to check the **PEMicro USB Serial Port 1** item in **Jungo** folder of Device Manager of the operating system, as shown in Figure 6.

Figure 6 - PEMicro USB serial port in the Device Manager window



3. Run the **Terminal** utility application from main menu, select **Start > Programs > P&E kinetis tower toolkit > Utilities**, as shown Figure 7.

Figure 7 - PEMicro Terminal Window



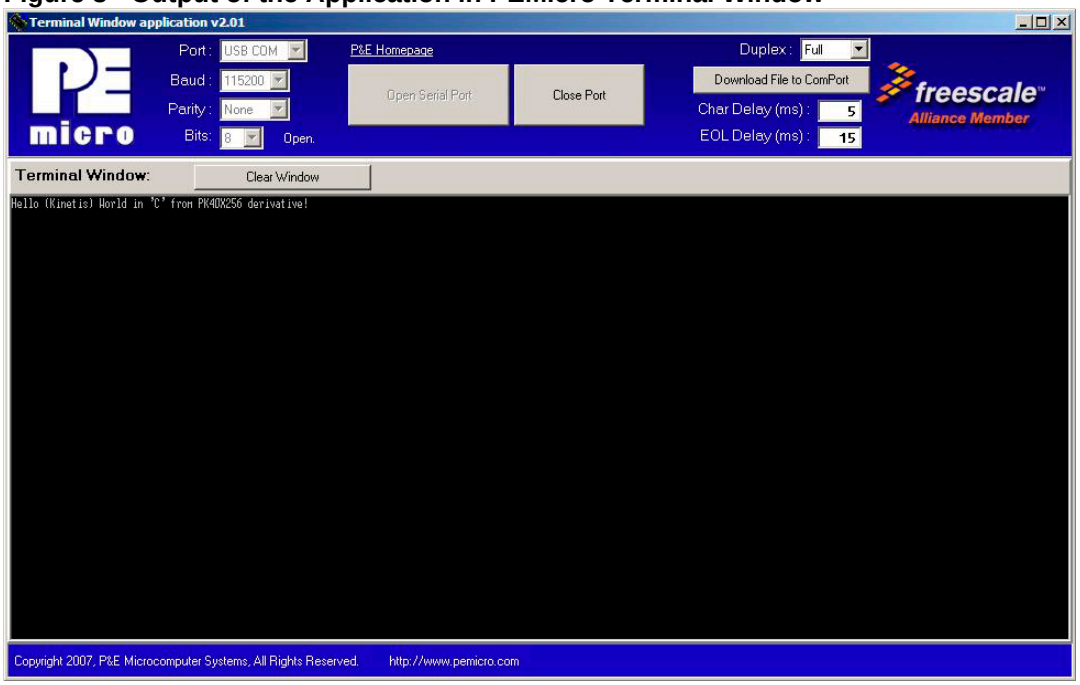
4. Select the specific USB COM port and set parameters of the communication as follows:
   Baudrate: 115200, Parity: None, Bits: 8

5. Click the **Open Serial Port** button to open the selected serial port for the communication.

6. Open the **k40_topwer.h** or the **k60_topwer.h** header file respectively from platforms directory of the project in project window and activate **#define OSJTAG** selection to use OS-JTAG circuit for the serial communication.

7. Build the project and run the application.

Figure 8 - Output of the Application in PEMicro Terminal Window



How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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)
support@freescale.com

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 Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

Document Number: AN4677

28 February 2013