Freescale Semiconductor
Application Note

Document Number: AN4033
Rev. 0, 01/2010

NAND Flash Boot for the Freescale MPC5125 MPU

by: Ju Yingyi,
32 bit Applications Engineering,
Microcontroller Solutions Group, China.

1 Introduction

This application note describes the procedures to perform NAND flash boot with the Freescale MPC5125 MPU. The hardware platform used in this example is a TWR-MPC5125 board.

Refer to the latest silicon and board documentation for updates to the information in this document. This application note is written using the information in:

- *MPC5125 Microcontroller Reference Manual* (document MPC5125RM)
- Schematics of TWR-MPC5125

2 TWR-MPC5125 Board Preparation

You must verify that TWR-MPC5125 board is working. The board ships with a version of U-boot software in the NAND flash. U-boot should boot from NAND flash and

Contents

1 Introduction 1
2 TWR-MPC5125 Board Preparation 1
3 MPC5125 Microcontroller Configuration 2
3.1 Reset 2
3.2 I/O Control 3
3.3 NAND Flash Boot 3
4 U-Boot NAND Flash Boot Software 4
4.1 Adding NAND Flash Boot Support to U-Boot 4
4.2 Files Modified in U-boot 5
4.3 Description of the nand_spl Directory Source Code 6
5 NAND Flash Boot Software 6
5.1 nandstart.S 6
5.2 nandload.c 12
Appendix ATWR-MPC5125 Board Layout 16

© Freescale Semiconductor, Inc., 2010. All rights reserved.

freescale™ semiconductor

give you a command prompt on the USB-to-serial port (J19). The port is configured to 115,200 baud, 8 data bits, 1 stop bit, and no hardware handshaking.

The factory default setting for SW1 is 0b010011 (RST_CONF_LPCMX = 0, RST_CONF_LPCWA = 1, RST_CONF_LPC_DBW1 = 0, RST_CONF_LPC_DBW0 = 0, RST_CONF_BMS = 1, RST_CONF_ROMLOC0 = 1). To make the board boot from NAND flash, you must turn SW1 position 1 and 2 to ON. See the schematics of MPC5125 for further information. For board layout diagrams, refer to Appendix A, “TWR-MPC5125 Board Layout.”

The TWR-MPC5125 board uses a Micron M29F64G08CFABA 8GB NAND flash that has 8-bit bus width, and is arranged into (4096 + 224 bytes) pages.

When U-boot is not already available in the memory and you want to re-program it, you will need CodeWarrior 9.2 and USB-TAP. CodeWarrior 9.2 can be downloaded from http://www.freescale.com. USB-TAP can also be bought from Freescale website. For more information of programming U-boot, refer to Section 4.

3 MPC5125 Microcontroller Configuration

3.1 Reset

Chapter 4, "Reset," of the *MPC5125 Microcontroller Reference Manual* (document MPC5125RM) describes the reset process. Two fields in the Reset Configuration Word High Register (RCWHR) affect the NAND flash boot and they must be set accordingly. See Section 4.6.2, "Reset Configuration Word High Register (RCWHR)," for further information regarding these fields:

- ROMLOC[1:0] — Selects the boot device. A value of 01 will select a NAND boot.
- BMS — Selects where the e300 will fetch the first instruction, either at address 0x0000_0000 or at address 0xfff0_0000.

Further information regarding RCWHR fields is also included in the *MPC5125 Microcontroller Reference Manual* (document MPC5125RM), Section 4.5, "Reset Configuration Word (RST_CONF).”

Section 4.5.7, "NFC Initialization Sequence," describes the steps that must be performed by the initial bootloader software when booting from NAND flash. This section also describes the functionality that can be deferred until the software is executing from DRAM. The following steps summarize the functionality that must be implemented in the NAND flash boot.

1. Configure the IMMR reset vector.
2. Configure the DRAM and NFC clock dividers. The default NFC clock values work fine, but increasing the clock speed will decrease boot time.
3. Configure the NFC parameters.
4. Initialize DRAM. Initialization should include DRAM access window as well as timings and initialization.
5. Copy the system software image to DRAM. The example code in this document uses a software loop to copy the image from NFC RAM to DRAM. However, DMA can also be used to copy the image with the added benefit of decreased boot time.

6. Perform absolute jump from NFC RAM to the DRAM system software image, where additional system initialization can be performed. A relative branch should not be used.

3.2 I/O Control

Chapter 20, "IO Control," of the *MPC5125 Microcontroller Reference Manual* (document MPC5125RM) describes the muxing and configuring of the pads. See the notes at the end of Table 20-6, "Pad IO Control Register Table," which point out that the default slew rates for the LPC and NFC signals depend on the designated boot source. The LPC/EMB pins, which are not used during NAND flash boot, are by default configured to the slowest slew rate. This behavior can be modified to meet the requirements of your system.

3.3 NAND Flash Boot

Chapter 23, "NAND Flash Controller (NFC)," of the *MPC5125 Microcontroller Reference Manual* (document MPC5125RM) describes the NAND flash boot process.

Section 23.8.3, "NAND Flash Boot," describes the boot process. The following information describes the behavior of the MPC5125 silicon when the NAND flash is the boot ROM.

- 4 boot blocks are identified in the flash. These are blocks at row addresses: 0, 256, 512, and 768.
- The flash controller does a burst read from the first boot block to memory. The burst is 4 pages long, therefore a total of 4 KB is read.
- ECC correction depth is 32 bits errors in an NFC page. If there is no ECC failure during the burst read, boot is successful.
- If there is an ECC failure during the burst read, the process is started again from the next boot block. If the fourth block still has ECC failure, boot is unsuccessful, which means it is not possible to read a reliable boot image from the flash.
- CPU access will be held until boot completion.
- After boot, boot image is visible on IPS memory map, address ranges from 0 to 3983. A special hash function is active on the NFC SRAM read to have the boot image in one continuous address range, and not in 4 address ranges, one for each page. The config bit, BOOT_MODE controls this hash function, and this bit should be turned off after the CPU has read/executed the boot image, and before it operates the NFC in the standard mode.

For details of the organization of NFC buffer memory, see section 23.5, "NFC Buffer Memory Space" and Figure 23-25, "Boot and BOOT_MODE" of the *MPC5125 Microcontroller Reference Manual* (document MPC5125RM).

The boot is identical for 8-wide and 16-wide flash. If a 16-wide flash is connected, the data on the upper lane is discarded. If a 8-wide flash is connected, the upper 1 KB of data is not read. For details, refer to Figure 23-26, "Boot data's location in a page of different size flash" of *MPC5125 Microcontroller Reference Manual* (document MPC5125RM).

To program the boot image onto the boot block in standard mode, complete the following steps.

1. Set the BOOT_MODE bit.

2. Fill the boot data into the NFC data buffer in sequence.
3. Clear the BOOT_MODE bit to allow NFC operate in standard mode. This operation will split the data in NFC buffer into 4 parts and these 4 parts of boot data are in hash function order.
4. Program the 4 pages of boot block at 8 bit data mode.

4 U-Boot NAND Flash Boot Software

Package of U-boot for TWR-MPC5125 board can be found on http://www.freescale.com.

4.1 Adding NAND Flash Boot Support to U-Boot

After you have extracted the source code and applied patches according to the readme file in the package, build a NAND flash boot U-boot image by entering the following commands to:

- Remove derived files:
  ```
  make clean
  ```
- Config U-boot to build an TWR-MPC5125 NAND flash boot U-boot image:
  ```
  make ads5125_nand_config
  ```
- Make an TWR-MPC5125 NAND flash boot U-boot image:
  ```
  make
  ```

After the "`make`" command has successfully completed executing, the U-boot images exist. These images are located in the base directory of "u-boot" source code tree and in the "nand_spl" sub-directory:

- u-boot-spl-2k.bin — The 2 KB image that is loaded as the first 4 pages from the NAND flash into the NFC internal RAM buffer.
- u-boot.bin — The remaining portion of U-boot that is copied into DRAM by u-boot-spl-2k.bin. This image should be programmed into offset 0x0080_0000 of the NAND flash.
- u-boot-nand.bin — This image is a concatenation of the two files listed above, u-boot-spl-2k.bin and u-boot.bin. It cannot be downloaded to NAND flash directly, because the 2 KB image (u-boot-spl-2k.bin) should be split into 4 pages and should obey the hash function organization rules.

To make your board boot from NAND flash, you need to program the u-boot-spl-2k.bin and u-boot.bin images by using U-boot that has NAND flash programming capability. This can also be done by using CCS of CodeWarrior for MobileGT V9.2 with CodeWarrior USB TAP.

To program these two binary files using a U-boot that has NAND flash programming capability, the network settings must be configured appropriately. At the U-boot prompt, enter the following commands.

- To update u-boot-spl-2k.bin:
  - — Use the U-boot NAND flash driver to erase the boot block.
    ```
    nand_e 0x00 0x01
    ```
  - — Use tftp to do a network transfer of the u-boot-spl-2k.bin image from the host server to the board.
    ```
    tftp 0x300000 u-boot-spl-2k.bin
    ```
  - — Use the U-boot NAND flash driver to program the u-boot-spl-2k.bin image.

```
nand_loader 0x300000 0x000 0x800
```

- To update u-boot.bin:
  - Use the u-boot NAND flash driver to erase the blocks that stores the U-boot.
    ```
    nand_e 0x100 0x101
    ```
  - Use tftp to do a network transfer of the u-boot.bin image from the host server to the board.
    ```
    tftp 0x300000 u-boot.bin
    ```
  - Use the U-boot NAND flash driver to program the U-boot.bin image.
    ```
    nand_loader 0x300000 0x100 0x40000
    ```

You can find ccs.exe at folder: {CodeWarrior for MobileGT V9.2 install directory}\ccs\bin.

To program these two images:

1. Prepare the script for CCS:
   - After building the U-boot successfully, two script files ({u-boot}\u-boot-second-scrip.txt and {u-boot}\nand_spl\loader-script-5125.txt) are generated automatically.
   - Copy these two script files to U-boot image folder where the other two script files (5125_init.txt and 5125_nand_block_erase.txt) are stored.
2. Connect CodeWarrior USB-TAB between PC and TWR-MPC5125 board, and then power-on the board.
3. Run CCS.exe on your PC. Load script file 5125_init.txt, 5125_nand_block_erase.txt, u-boot-second-scrip.txt, and loader-script-5125.txt in sequence.

After programming successfully, set your serial console to 115200, 8-N-1, attach the board to PC through USB-to-serial port, and reset the board. The U-boot will then start and will give you a command prompt on the USB-to-serial port.

4.2 Files Modified in U-boot

Files that are modified to support the NAND flash driver in U-boot are:

1. board/ads5125/ads5125.c
2. drivers/mtd/nand/Makefile
3. drivers/mtd/nand/fsl_nfc_nand.c
4. include/configs/ads5125.h

Files that are modified to support NAND flash boot in U-boot are:

1. Makefile
2. board/ads5125/ads5125.c
3. board/ads5125/config.mk
4. cpu/mpc512x/start.S
5. include/configs/ads5125.h
6. include/mpc512x.h
7. nand_spl/board/ads5125/Makefile
8. nand_spl/board/ads5125/config.mk

9. nand_spl/board/ads5125/dram.h
10. nand_spl/board/ads5125/nandload.c
11. nand_spl/board/ads5125/nandstart.S
12. nand_spl/board/ads5125/nfc.h
13. nand_spl/board/ads5125/u-boot.lds

4.3 Description of the nand_spl Directory Source Code

The nand_spl (secondary program loader) directory contains the source to build the image that is automatically loaded into the first four pages of the NAND flash. These first four pages are automatically loaded into the MPC5125 NFC memory during reset.

The {u-boot}/nand_spl/board/ads5125 directory contains the two source code files used to build the nand_spl image. These two source files are also included in Section 5, “NAND Flash Boot Software." The files are:

1. nandstart.S — This is the first code to execute. It performs early initialization, calls nandload() function, and jumps to the fully loaded U-boot image once it is in DRAM.
2. nandload.c — Contains code that reads the entire U-boot image (as NAND flash pages) and writes them to DRAM.

5 NAND Flash Boot Software

The initial bootloader software to execute in support of NAND flash boot on the MPC5125 should focus on following the algorithm described in the *MPC5125 Microcontroller Reference Manual* (document MPC5125RM), Section 4.5.7, “NFC Initialization Sequence.”

This initial bootloader software must configure the hardware platform including the DRAM, so the entire system software can be copied from NAND flash to DRAM. This initial bootloader software is limited in size to 3984 bytes. For more details, see *MPC5125 Microcontroller Reference Manual* (document MPC5125RM), Section 23.8.3, "Nand Flash Boot.”

The following sections provide example code that performs the algorithm described in the *MPC5125 Microcontroller Reference Manual* (document MPC5125RM), Section 4.5.7, "NFC Initialization Sequence." This code is from the U-boot bootloader software. See Section 4 for further information regarding U-boot.

5.1 nandstart.S

```
{u-boot}\nand_spl\board\ads5125\nandstart.S
/*
 * (C) Copyright 2009
 * Martha Marx, Silicon Turnkey Express, mmarx@silicontkx.com
 *
 * Based on original start.S done by
 *       Copyright (C) 1998  Dan Malek <dmalek@jlc.net>
 *       Copyright (C) 1999  Magnus Damm <kieraypc01.p.y.kie.era.ericsson.se>
 *       Copyright (C) 2000, 2001, 2002, 2007 Wolfgang Denk <wd@denx.de>
 * start.S for mpc512x was originally based on the MPC83xx code.
```

```
 *
 * See file CREDITS for list of people who contributed to this
 * project.
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston,
 * MA 02111-1307 USA
 */

/*
 *  U-Boot - NAND Boot Startup Code for MPC5121 Embedded Boards
 */
#define DEBUG

#include <config.h>
#include <mpc512x.h>
#include <version.h>
#include "dram.h"
//#include "nfc.h"

#define CONFIG_521X1/* needed for Linux kernel header files*/

#include <ppc_asm.tmpl>
#include <ppc_defs.h>

#include <asm/cache.h>
#include <asm/mmu.h>

#ifndef  CONFIG_IDENT_STRING
#define  CONFIG_IDENT_STRING "MPC512X"
#endif

/*
 * Floating Point enable, Machine Check and Recoverable Interr.
 */
#undef   MSR_KERNEL
#ifdef DEBUG
#define MSR_KERNEL (MSR_FP|MSR_RI)
#else
#define MSR_KERNEL (MSR_FP|MSR_ME|MSR_RI)
#endif

/* Macros for manipulating CSx_START/STOP */
#define START_REG(start)((start) >> 16)
#define STOP_REG(start, size)(((start) + (size) - 1) >> 16)
#define SET_MEM_BASE(r, b) \
```

```
        lis     r,(b)@h;\
        ori     r,r,(b)@l;\

#define SET_REG32(r, v, offset, mr)\
        lis     r, v@h; \
        ori     r, r, v@l;\
        stw     r, offset(mr);\

#define SET_REG16(r, v, offset, mr)\
        li      r, v;             \
        sth     r, offset(mr);\


        .text
        .globl  version_string
version_string:
        .ascii U_BOOT_VERSION
        .ascii " (", __DATE__, " - ", __TIME__, ")"
        .ascii " ", CONFIG_IDENT_STRING, " "
        .ascii "2K NAND BOOT ","\0"
        . = EXC_OFF_SYS_RESET

        .globl  _start
        /* Start from here after reset/power on */
_start:
boot_cold:
        /* Save msr contents */
        mfmsr   r5
        lis     r4, CONFIG_DEFAULT_IMMR@h

        /* Set IMMR area to our preferred location */
        mfspr   r6, MBAR
        lis     r3, CFG_IMMR@h
        ori     r3, r3, CFG_IMMR@l

        cmpw    r3, r6
        beq     1f /* it has already been set to what we want it to be */
                   /* -- nice to chk if coming out of the BDI          */


        stw     r3, IMMRBAR(r4)
        mtspr   MBAR, r3 /* IMMRBAR is mirrored into the MBAR SPR (311) */
        isync
1:      lis     r4, START_REG(CFG_FLASH_BASE)
        ori     r4, r4, STOP_REG(CFG_FLASH_BASE, CFG_FLASH_SIZE)
        stw     r4, LPBAW(r3)
        stw     r4, LPCS0AW(r3)
        isync
        /* Initialise the machine */
        bl      cpu_early_init
        isync

        /*
         * The SRAM window has a fixed size (256K),
         * so only the start addressis necessary
         */
        lis     r3, CFG_IMMR@h
```

```
        ori     r3, r3, CFG_IMMR@l
        lis      r4, START_REG(CFG_SRAM_BASE) & 0xff00
        stw      r4, SRAMBAR(r3)

        /*
         * According to MPC5121e RM, configuring local access windows should
         * be followed by a dummy read of the config register that was
         * modified last and an isync
         */
        lwz      r4, SRAMBAR(r3)
        isync

#if 0
#ifdef CONFIG_ADS5125 /* CS2 FUNC MUX must be done before CS is enabled */
        lis     r4, (CONFIG_SYS_IOCTRL_ADDR)@h
        ori     r4, r4, (CONFIG_SYS_IOCTRL_ADDR)@l
        li       r5, IOCTRL_MUX_CS2
        stb      r5, IO_CTRL_LPC_AX03(r4)
/* change the pin muxing on PSC9 here in case it is being used as console*/
        li       r5, IOCTRL_MUX_PSC9
        stb      r5, IO_CTRL_I2C1_SCL(r4)
        stb      r5, IO_CTRL_I2C1_SDA(r4)

#endif
#endif

        /* r3: BOOTFLAG */
        mr       r3, r21

        bl       dram_init


        /* r3: BOOTFLAG */
        mr       r3, r21
        lis      r1, (CFG_INIT_RAM_ADDR + CFG_GBL_DATA_OFFSET)@h
        ori      r1, r1, (CFG_INIT_RAM_ADDR + CFG_GBL_DATA_OFFSET)@l

        /*copy sram to ddr*/
        bl       sram_to_ddr
        /* copy the full U-Boot into DDR  */
        /* and jump to it */
jump_uboot:
        SET_MEM_BASE(r10, nandload-CONFIG_SYS_NAND_BASE+CFG_LOADER_DDR_START)
        mtlr     r10
        isync
        blr

        /* NOTREACHED - nand_boot() does not return */
/*
 * This code initialises the machine,
 * it expects original MSR contents to be in r5
 */
cpu_early_init:
        /* Initialize machine status; enable machine check interrupt */
        /*-----------------------------------------------------------*/

        li       r3, MSR_KERNEL    /* Set ME and RI flags */
```

```
        rlwimi   r3, r5, 0, 25, 25/* preserve IP bit */
#ifdef DEBUG
        rlwimi   r3, r5, 0, 21, 22/* debugger might set SE, BE bits */
#endif
        mtmsr    r3
        SYNC
        mtspr    SRR1, r3          /* Mirror current MSR state in SRR1 */

        lis      r3, CFG_IMMR@h

        /* Disable the watchdog */
        /*----------------------*/
        lwz r4, SWCRR(r3)
        /*
         * Check to see if it's enabled for disabling: once disabled by s/w
         * it's not possible to re-enable it
         */
        andi. r4, r4, 0x4
        beq 1f
        xor r4, r4, r4
        stw r4, SWCRR(r3)
1:

        /* Initialize the Hardware Implementation-dependent Registers */
        /* HID0 also contains cache control*/
        /*--------------------------------------------------------*/
        lis      r3, CFG_HID0_INIT@h
        ori      r3, r3, CFG_HID0_INIT@l
        SYNC
        mtspr    HID0, r3

        blr

dram_init:

        SET_MEM_BASE(r3, CFG_IMMR + IOCTL_BASE_ADDR)
        SET_REG32(r4, IOCTRL_MUX_DDR, IOCTL_MEM , r3)

        SET_MEM_BASE(r3, CFG_IMMR)
        SET_REG32(r4, CFG_DDR_BASE & 0xFFFFF000, DDR_LAW_BAR, r3)
        SET_REG32(r4, 0x0000001b, DDR_LAW_AR, r3)
        lwz     r0, DDR_LAW_AR(r3)
        isync

        SET_MEM_BASE(r3, CFG_IMMR + MDDRC_BASE_OFFSET)
        SET_REG32(r4, CFG_MDDRC_SYS_CFG_EN, DDR_SYS_CONFIG, r3)

        SET_REG32(r4, CFG_MDDRCGRP_PM_CFG1, DRAMPRIOM_PRIOMAN_CONFIG1, r3)
        SET_REG32(r4, CFG_MDDRCGRP_PM_CFG2, DRAMPRIOM_PRIOMAN_CONFIG2, r3)
        SET_REG32(r4, CFG_MDDRCGRP_HIPRIO_CFG, DRAMPRIOM_HIPRIO_CONFIG, r3)
        SET_REG32(r4, CFG_MDDRCGRP_LUT0_MU, DRAMPRIOM_LUT_TABLE0_MAIN_UP, r3)
        SET_REG32(r4, CFG_MDDRCGRP_LUT0_ML, DRAMPRIOM_LUT_TABLE0_MAIN_LOW, r3)
        SET_REG32(r4, CFG_MDDRCGRP_LUT1_MU, DRAMPRIOM_LUT_TABLE1_MAIN_UP, r3)
        SET_REG32(r4, CFG_MDDRCGRP_LUT1_ML, DRAMPRIOM_LUT_TABLE1_MAIN_LOW, r3)
        SET_REG32(r4, CFG_MDDRCGRP_LUT2_MU, DRAMPRIOM_LUT_TABLE2_MAIN_UP, r3)
        SET_REG32(r4, CFG_MDDRCGRP_LUT2_ML, DRAMPRIOM_LUT_TABLE2_MAIN_LOW, r3)
        SET_REG32(r4, CFG_MDDRCGRP_LUT3_MU, DRAMPRIOM_LUT_TABLE3_MAIN_UP, r3)
```

```
SET_REG32(r4, CFG_MDDRCGRP_LUT3_ML, DRAMPRIOM_LUT_TABLE3_MAIN_LOW, r3)
SET_REG32(r4, CFG_MDDRCGRP_LUT4_MU, DRAMPRIOM_LUT_TABLE4_MAIN_UP, r3)
SET_REG32(r4, CFG_MDDRCGRP_LUT4_ML, DRAMPRIOM_LUT_TABLE4_MAIN_LOW, r3)
SET_REG32(r4, CFG_MDDRCGRP_LUT0_AU, DRAMPRIOM_LUT_TABLE0_ALT_UP, r3)
SET_REG32(r4, CFG_MDDRCGRP_LUT0_AL, DRAMPRIOM_LUT_TABLE0_ALT_LOW, r3)
SET_REG32(r4, CFG_MDDRCGRP_LUT1_AU, DRAMPRIOM_LUT_TABLE1_ALT_UP, r3)
SET_REG32(r4, CFG_MDDRCGRP_LUT1_AL, DRAMPRIOM_LUT_TABLE1_ALT_LOW, r3)
SET_REG32(r4, CFG_MDDRCGRP_LUT2_AU, DRAMPRIOM_LUT_TABLE2_ALT_UP, r3)
SET_REG32(r4, CFG_MDDRCGRP_LUT2_AL, DRAMPRIOM_LUT_TABLE2_ALT_LOW, r3)
SET_REG32(r4, CFG_MDDRCGRP_LUT3_AU, DRAMPRIOM_LUT_TABLE3_ALT_UP, r3)
SET_REG32(r4, CFG_MDDRCGRP_LUT3_AL, DRAMPRIOM_LUT_TABLE3_ALT_LOW, r3)
SET_REG32(r4, CFG_MDDRCGRP_LUT4_AU, DRAMPRIOM_LUT_TABLE4_ALT_UP, r3)
SET_REG32(r4, CFG_MDDRCGRP_LUT4_AL, DRAMPRIOM_LUT_TABLE4_ALT_LOW, r3)

/* Initialize MDDRC */
SET_REG32(r4, CFG_MDDRC_SYS_CFG_EN, DDR_SYS_CONFIG, r3)
SET_REG32(r4, CFG_MDDRC_TIME_CFG0, DDR_TIME_CONFIG0, r3)
SET_REG32(r4, CFG_MDDRC_TIME_CFG1, DDR_TIME_CONFIG1, r3)
SET_REG32(r4, CFG_MDDRC_TIME_CFG2, DDR_TIME_CONFIG2, r3)


/* Initialize DDR */
SET_REG32(r4, CFG_MICRON_NOP, DDR_COMMAND, r3)
stw     r4, DDR_COMMAND(r3);
stw     r4, DDR_COMMAND(r3);
stw     r4, DDR_COMMAND(r3);
stw     r4, DDR_COMMAND(r3);
stw     r4, DDR_COMMAND(r3);
stw     r4, DDR_COMMAND(r3);
stw     r4, DDR_COMMAND(r3);
stw     r4, DDR_COMMAND(r3);
stw     r4, DDR_COMMAND(r3);
stw     r4, DDR_COMMAND(r3);

SET_REG32(r4, CFG_MICRON_PCHG_ALL, DDR_COMMAND, r3)
SET_REG32(r5, CFG_MICRON_NOP, DDR_COMMAND, r3)
SET_REG32(r4, CFG_MICRON_RFSH, DDR_COMMAND, r3)
stw     r5, DDR_COMMAND(r3);
SET_REG32(r4, CFG_MICRON_RFSH, DDR_COMMAND, r3)
stw     r5, DDR_COMMAND(r3);
SET_REG32(r4, CFG_MICRON_INIT_DEV_OP, DDR_COMMAND, r3)
stw     r5, DDR_COMMAND(r3);
SET_REG32(r4, CFG_MICRON_EM2, DDR_COMMAND, r3)
stw     r5, DDR_COMMAND(r3);
SET_REG32(r4, CFG_MICRON_PCHG_ALL, DDR_COMMAND, r3)
SET_REG32(r4, CFG_MICRON_EM2, DDR_COMMAND, r3)
SET_REG32(r4, CFG_MICRON_EM3, DDR_COMMAND, r3)
SET_REG32(r4, CFG_MICRON_EN_DLL, DDR_COMMAND, r3)
SET_REG32(r4, CFG_MICRON_INIT_DEV_OP, DDR_COMMAND, r3)
SET_REG32(r4, CFG_MICRON_PCHG_ALL, DDR_COMMAND, r3)
SET_REG32(r4, CFG_MICRON_RFSH, DDR_COMMAND, r3)
SET_REG32(r4, CFG_MICRON_INIT_DEV_OP, DDR_COMMAND, r3)
SET_REG32(r4, CFG_MICRON_OCD_DEFAULT, DDR_COMMAND, r3)
SET_REG32(r4, CFG_MICRON_PCHG_ALL, DDR_COMMAND, r3)
stw     r5, DDR_COMMAND(r3);

/* Start MDDRC */
```

```
        SET_REG32(r4, CFG_MDDRC_TIME_CFG0_RUN, DDR_TIME_CONFIG0, r3)
        SET_REG32(r4, CFG_MDDRC_SYS_CFG_RUN, DDR_SYS_CONFIG, r3)
        isync
        blr
```

5.2 nandload.c

```
{u-boot}\nand_spl\board\ads5125\nandload.c
/*
 * (C) Copyright 2009
 * Martha Marx, Silicon Turnkey Express, mmarx@silicontkx.com
 *
 * See file CREDITS for list of people who contributed to this
 * project.
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston,
 * MA 02111-1307 USA
 */

#include <common.h>
#include <mpc512x.h>
#include <version.h>

#define CONFIG_521X1/* needed for Linux kernel header files*/

#include <ppc_asm.tmpl>
#include <ppc_defs.h>
#include <asm/bitops.h>
#include <asm/io.h>
#include <asm/cache.h>
#include <asm/mmu.h>
//#include "nfc.h"
#include "mpc5125_nfc.h"

#define NUMPAGES 80 /* if u-boot grows .. extend this */

#define UBOOT_START_PAGE0x8

#define  NAND_PAGE_SIZE0x1000

#define UBOOT_IMAGIC_NUMBER0x27051956
#define  UBOOT_HEADER_FLAG0x4d544300
struct uboot_header{
unsigned int uboot_imagic;
unsigned int uboot_len;
```

```
unsigned int uboot_heaerflag;
unsigned int uboot_checksum;
};

#define  UBOOT_BLOCK1_PAGE00x100


void sram_to_ddr(void)
{
        unsigned int *dst=(unsigned int *)CFG_LOADER_DDR_START;
        unsigned int *src=(unsigned int *)CONFIG_SYS_NAND_BASE;
         unsigned int size;
        for(size=0;size<0x200;size++)
        {
                *dst++=*src++;
        }
        return ;
}
static  int nand_read_page(unsigned int row_addr,unsigned int *buf)
{
        unsigned int tmp,timeout,i;
        unsigned int *src;
        int ret=-1;

        out_be32(CONFIG_SYS_NAND_BASE+0x3f00,0x30000000);

        /*addr */
        out_be32(CONFIG_SYS_NAND_BASE+0x3f08,0x00000000);
        out_be32(CONFIG_SYS_NAND_BASE+0x3f0c,0x11000000+row_addr);

        /*sector size*/
        out_be32(CONFIG_SYS_NAND_BASE+0x3f2c,0x840);
        out_be32(CONFIG_SYS_NAND_BASE+0x3f10,0);

        out_be32(CONFIG_SYS_NAND_BASE+0x3f30,0x000ea632);
        out_be32(CONFIG_SYS_NAND_BASE+0x3f14,0);
        /*start*/
        out_be32(CONFIG_SYS_NAND_BASE+0x3f38,(3<<17));

        out_be32(CONFIG_SYS_NAND_BASE+0x3f04,0x007ee001);
        timeout=1000000;

        /*out_be32(CONFIG_SYS_NAND_BASE+0x3f38,(1<<16));*/
        while(1)
        {
                tmp=in_be32(CONFIG_SYS_NAND_BASE + 0x3f38);
                if((tmp&(3<<29))==(3<<29))
                {
                        out_be32(CONFIG_SYS_NAND_BASE+0x3f38,(3<<17));
                        break;
                }
        }
        src=CONFIG_SYS_NAND_BASE;
        for(i=0;i<0x200;i++)
        {
                *buf++=*src++;
        }
```

```
        src=CONFIG_SYS_NAND_BASE+0x1000;
        for(i=0;i<0x200;i++)
        {
                *buf++=*src++;
        }

        return ret;

}
void nandload(void)
{
        void (*uboot)(void);
        int i=0,pages;
        struct uboot_header uh,*puh;
        unsigned int *uboot_buf,checksum=0;
        unsigned char *s;
        /*first read block1 ,if uboot exist ,read it*/
        uboot_buf=CFG_NAND_U_BOOT_DST;
        puh=CFG_NAND_U_BOOT_DST;
        uboot=(void (*)(void) )CFG_NAND_U_BOOT_START;
        nand_read_page(UBOOT_BLOCK1_PAGE0,uboot_buf);

if((puh->uboot_imagic!=UBOOT_IMAGIC_NUMBER)||(puh->uboot_heaerflag!=UBOOT_HEADER_FLAG))
        {
                goto start_from_block0;

        }
        uh.uboot_checksum=puh->uboot_checksum;
        uh.uboot_len=puh->uboot_len;
        pages=(uh.uboot_len+0x10+NAND_PAGE_SIZE-1)/NAND_PAGE_SIZE;
        pages=(pages>0x100)?0x100:pages;

        uboot_buf=CFG_NAND_U_BOOT_DST-0x10;
        for(i=0;i<pages;i++)
        {
                nand_read_page(UBOOT_BLOCK1_PAGE0+i,uboot_buf);
                uboot_buf+=NAND_PAGE_SIZE>>2;
        }

        /**checksum**/

        s=CFG_NAND_U_BOOT_DST;
        checksum=0;
        for(i=0;i<uh.uboot_len;i++)
        {
                checksum+=*s++;
        }
#if 0
        if(checksum==uh.uboot_checksum)
        {

                (*uboot)();
                return;
        }
#else
        (*uboot)();
#endif
```

```
start_from_block0:
        uboot_buf=CFG_NAND_U_BOOT_DST;
        for(i=0;i<NUMPAGES;i++)
        {
                nand_read_page(UBOOT_START_PAGE+i,uboot_buf);
                uboot_buf+=NAND_PAGE_SIZE>>2;
        }
        uboot=(void (*)(void) )CFG_NAND_U_BOOT_START;
        (*uboot)();

        return;
}
```

Appendix A TWR-MPC5125 Board Layout

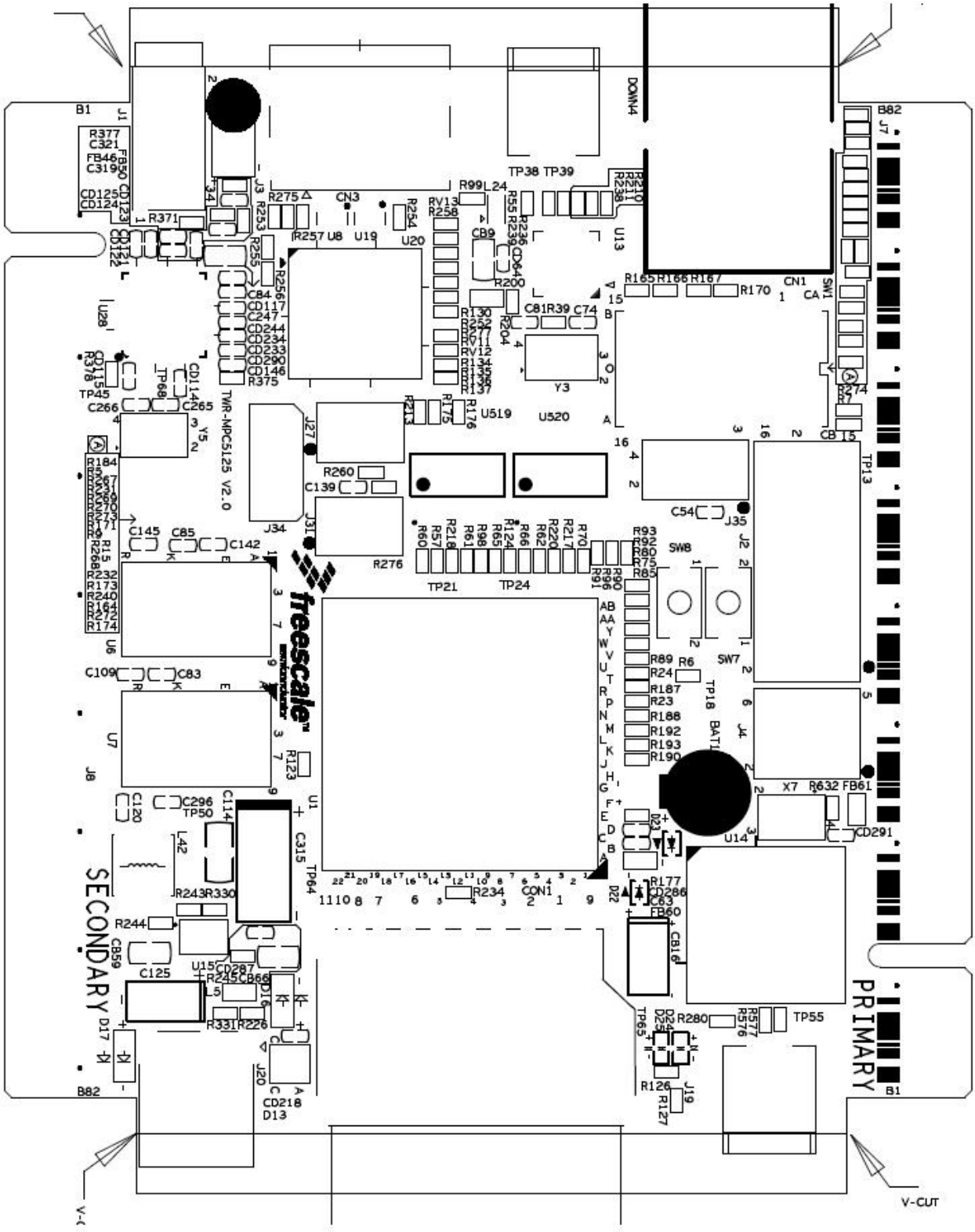


Figure 1. TWR-MPC5125 Board Top View

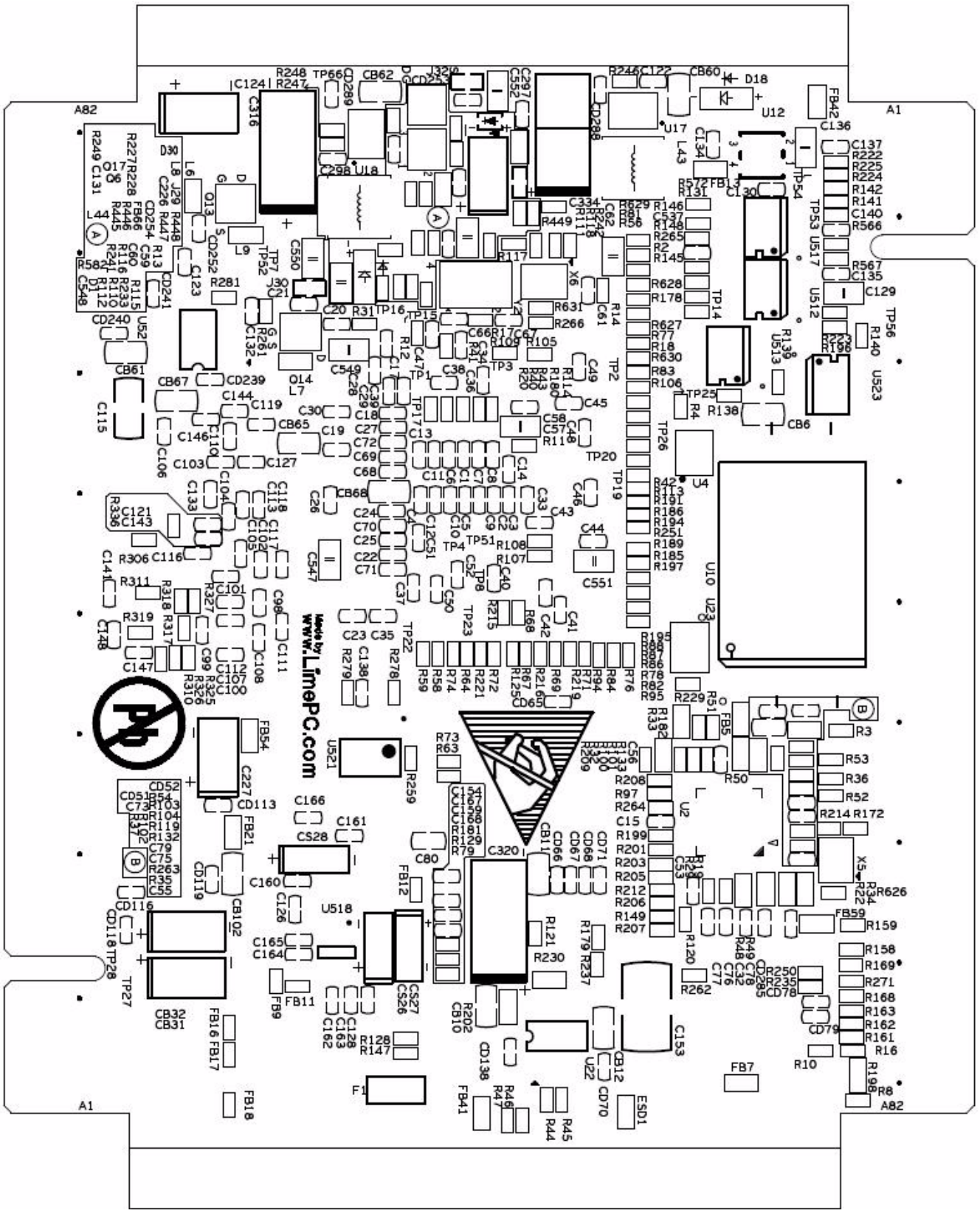


Figure 2. TWR-MPC5125 Board Bottom View

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

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.
© Freescale Semiconductor, Inc. 2010. All rights reserved.

Document Number: AN4033
Rev. 0
01/2010

freescale™
semiconductor