Freescale Semiconductor
Application Note

Document Number: AN3845
Rev. 0, 05/2009

NAND Flash Boot for the Freescale MPC5121e

by: Gene Fortanely
Applications Engineering
Microcontroller Solutions Group

1 Introduction

This document describes the procedures needed to perform NAND flash boot with a Freescale MPC5121e. The hardware platform used in this example is a Silicon Turnkey Express ADS512101 rev 4 board.

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

1. Freescale document MPC5121ERM, *MPC5121e Microcontroller Reference Manual*, Rev. 3, October 2008.
2. Silicon Turnkey Express document ADS512101UM, *ADS512101 Advanced Development System User's Manual*, Rev. 1.1, September 4, 2008.
3. Freescale document AN3765, "Porting Linux for the MPC5121e," Rev. 0, 12/2008.

Contents

freescale™ semiconductor

2 ADS512101 Board Preparation

Verify your board is working. The board ships with a version of u-boot software in the NOR flash. U-boot should boot from NOR flash and give you a command prompt on the UART0 serial port. The UART0 serial port is configured to 115,200 baud, 8 data bits, 1 stop bit, and no hardware handshaking.

The factory default for all eight of the SW3 positions is to be on. To make the board boot from NAND flash you must put SW3 position 2 to off. See the *ADS512101 Advanced Development System User's Manual* for further information including a board layout diagram that shows the location of SW3. The board layout diagram is included in this document as Appendix A, "ADS512101 Top Board Layout."

The ADS512101 uses a Hynix HY27UG088G(5/D)M 1 GB NAND flash which has a ×8 bus width, and is arranged into (2k + 64) pages.

3 MPC5121e Microcontroller Configuration

3.1 Reset

Chapter 4, "Reset," of the *MPC5121e Microcontroller Reference Manual* describes the reset process. Four fields in the Reset Configuration Word High Register (RCWHR) affect the NAND flash boot and they must be set accordingly. See section 4.7.2, "Reset Configuration Word High Register (RCWHR)," for further information regarding these four fields:

- ROMLOC[1:0]—selects the boot device. A value of 01 or 11 will select a NAND boot.
- NFC_PS—selects the NAND flash page size and is dependent on the value selected for ROMLOC[1:0].
- NFC_DBW—selects the NAND flash data port size.
- BMS—selects where the e300 will fetch the first instruction, either at address 0x0000_0000 or at address 0xfff0_0000.

Further information regarding these four RCWHR fields is also included in section 26.6.1, "Modes of Operation," of the NFC chapter of the *MPC5121e Microcontroller Reference Manual*.

Section 4.6.7, "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. Here is a summary of the functionality that must be implemented in the NAND flash boot:

1. Configure the IMMR reset vector.
2. Configure the DRAM & 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 22, "IO Control," of the *MPC5121e Microcontroller Reference Manual* describes the muxing and configuring of the pads. See the notes at the end of table 22-10, "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 may need to be modified to meet the needs of your system.

3.3 NAND Flash Boot

Chapter 26, "NAND Flash Controller (NFC)," of the *MPC5121e Microcontroller Reference Manual* describes the NAND flash boot process.

Section 26.6.1, "Modes of Operation," describes the Reset Configuration register (RCWHR) settings that must be set to reflect the hardware environment, specifically the NAND flash page size (NFC_PS), NAND flash bus width (NFC_DBW), and selecting the NAND flash as the boot ROM (ROMLOC[1:0]).

Section 26.6.2, "Booting From a NAND Flash Device," describes the boot process. This information describes the behavior of the MPC5121e silicon when the NAND flash is the boot ROM.

The NFC ECC engine can correct four or eight symbols in an NFC page. If an unrecoverable number of symbol errors occur in the first NFC page, then software cannot do anything. If an unrecoverable number of symbol errors occur in any of the following NFC pages, then software can issue an error status, ignore the unrecoverable page, or read a duplicate copy of the page. Duplicate pages require that the NAND flash be programmed accordingly.

4 U-Boot NAND Flash Boot Software

U-boot is a bootloader software available from www.denx.de. A u-boot that supports NAND flash boot for the MPC5121e was developed in early 2009 using u-boot version "u-boot-2008.10."

Four source code files are available with this document and can be downloaded in a zip file that should appear with this application note on freescale.com. The first two files are patch files used to create a complete u-boot source code tree with NAND flash boot support. The other two files are a subset of the u-boot source code tree, but provide easy access to these two key files. The files are:

- u-boot_2008.10_ADS5121_NFC_NAND_Flash_Driver_20090130.patch
- u-boot_2008.10_ADS5121_NAND_Flash_Boot_20090130.patch
- nandstart.S
- nandload.c

Obtain u-boot version "u-boot-2008.10," as this is the basis of the software for this document. This version of software will only provide support for a NOR flash u-boot image for the ADS512101 board. Verify you are able to successfully build and run a NOR flash u-boot image on your board. If you are unfamiliar with

u-boot, refer to Freescale application note AN3765, “Porting Linux for the MPC5121e,” section 2.1, “u-boot Source Code from DENX,” for further instructions on how to obtain, configure, and build u-boot.

4.1 Adding NAND Flash Boot Support to U-boot

To add NAND flash boot support to the u-boot version “u-boot-2008.10” source code tree, you must apply two patches. Copy the two patch files available with this document:

- u-boot_2008.10_ADS5121_NFC_NAND_Flash_Driver_20090130.patch
- u-boot_2008.10_ADS5121_NAND_Flash_Boot_20090130.patch

to the base directory of your “u-boot-2008.10” source code tree. Then apply the two patch files in this order:

1. NAND flash driver patch
2. NAND flash boot patch

Apply the patches like this:

```
# patch file containing ADS5121 NAND flash driver support
        patch -p1 < u-boot_2008.10_ADS5121_NFC_NAND_Flash_Driver_20090130.patch
#patch file containing ADS5121 NAND flash boot support
        patch -p1 < u-boot_2008.10_ADS5121_NAND_Flash_Boot_20090130.patch
```

Once you have applied the patches, build a NAND flash boot u-boot image by entering the commands:

```
#remove derived files
        make clean
#config u-boot to build an ADS5121 NAND flash boot u-boot image
        make ADS5121_nand_config
#make an ADS5121 NAND flash boot u-boot image
        make
```

Once the “make” command has successfully completed executing, the u-boot images will exist. They are located in the base directory of your “u-boot-2008.10” source code tree and in the “nand_spl” sub-directory:

nand_spl/u-boot-spl-2k.bin—the 2 KB image that is loaded as the first page from the NAND flash into the NFC internal RAM buffer. This image should be programmed into offset 0x0000_0000 of the NAND flash.

u-boot.bin—the remaining portion of u-boot which is copied into DRAM by nand_spl/u-boot-spl-2k.bin. This image should be programmed into offset 0x0000_0800 of the NAND flash.

u-boot-nand.bin—a concatenation of the two files listed above, nand_spl/u-boot-spl-2k.bin and u-boot.bin. If you wish to flash program the entire u-boot software in one operation, then this image should be programmed into offset 0x0000_0000 of the NAND flash.

To make your board boot from NAND flash, you need to program the u-boot-nand.bin image using a JTAG NAND flash programmer or a u-boot that has NAND flash programming capability. You must also configure your board to boot from NAND flash as described in Section 2, "ADS512101 Board Preparation."

To program u-boot-nand.bin using a u-boot that has NAND flash programming capability, the network settings must be configured appropriately. Refer to Freescale application note AN3765, "Porting Linux for the MPC5121e," section 2.2, "u-boot Source Code from Freescale," for further instructions on configuring the network settings. At the u-boot prompt enter these commands:

```
# use the u-boot NAND flash driver to erase part of the NAND Flash
    nand erase 0 44000
# use tftp to do a network transfer of the u-boot-nand.bin image from the host server to the board
    tftp 300000 u-boot-nand.bin
# use the u-boot NAND Flash driver to program the u-boot-nand.bin image
    nand write 300000 0
```

To initialize the bad block table in a NAND flash, use this command before attempting to erase or program the NAND flash from u-boot:

```
# initialize NAND Flash bad block table
    nand bad
```

To program u-boot-nand.bin using a JTAG NAND flash programmer, set the NAND flash programmer to program the binary image at an offset of 0x0000_0000. The NAND flash programmer must support the NAND flash part and configuration, which is described in Section 2, "ADS512101 Board Preparation."

4.2 Files Modified in U-boot version u-boot-2008.10

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

1. board/ads5121/ads5121.c
2. drivers/mtd/nand/Makefile
3. drivers/mtd/nand/fsl_nfc_nand.c
4. include/configs/ads5121.h

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

1. Makefile
2. board/ads5121/ads5121.c
3. board/ads5121/config.mk
4. cpu/mpc512x/start.S
5. include/configs/ads5121.h
6. include/mpc512x.h
7. nand_spl/board/ads5121/Makefile

8. nand_spl/board/ads5121/config.mk
9. nand_spl/board/ads5121/dram.h
10. nand_spl/board/ads5121/nandload.c
11. nand_spl/board/ads5121/nandstart.S
12. nand_spl/board/ads5121/nfc.h
13. nand_spl/board/ads5121/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 page of the NAND flash. This first page is automatically loaded into the MPC5121e NFC memory during reset. The u-boot-2008.10/nand_spl/board/ads5121 directory contains the two source code files used to build the nand_spl image. These two source files are also included below in Section 5, "NAND Flash Boot Software." The files are:

1. nandstart.S—First code to execute. 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 MPC5121e should focus on following the algorithm described in the MPC5121e Microcontroller Reference Manual section 4.6.7, "NFC Initialization Sequence." This initial bootloader software must configure the hardware platform including the DRAM, so that the entire system software can be copied from NAND flash to DRAM. This initial bootloader software is limited in size to either 2 KB or 4 KB.

An alternative initial bootloader software algorithm that can be used to bypass the limit of a single NAND flash page of 512 bytes or 2 KB involves loading intermediate initial software, or even the entire system software, into the MPC5121e 128 KB on-chip SRAM.

The following code provides example code that performs the algorithm described in the *MPC5121e Microcontroller Reference Manual* section 4.6.7, "NFC Initialization Sequence." This code is from the u-boot bootloader software. See Section 4, "U-Boot NAND Flash Boot Software," for further information regarding u-boot.

5.1 nandstart.S

u-boot-2008.10/nand_spl/board/ads5121/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

        /* 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 the full U-Boot into DDR  */
        bl       nandload
        /* and jump to it */
jump_uboot:
        SET_MEM_BASE(r10, CFG_NAND_U_BOOT_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, 0x0000001c, 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-2008.10/nand_spl/board/ads5121/nandload.c

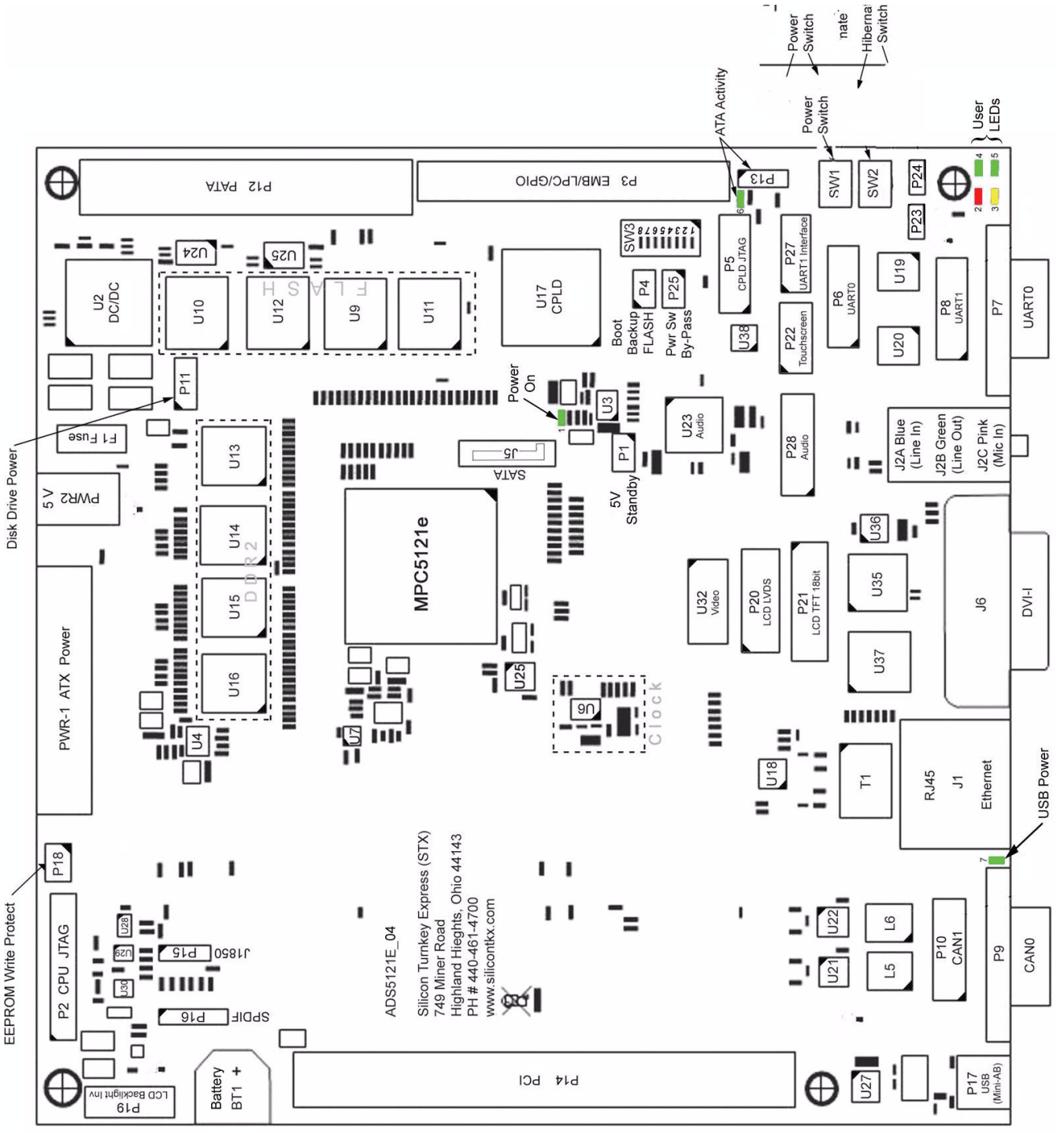
```
/*
 * (C) Copyright 2009
```

```
 * 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"

/*!
 * This function polls the NFC to wait for the basic operation to complete by
 * checking the INT bit of config2 register.
 *
 * @max_retries number of retry attempts
 */
static void wait_op_mjmdone(void)
{
        int i;
        int max_retries = 10000;
        u16 output;
        u16 temp;

        while (1) {
                max_retries--;
                output = in_be16((u16 *)(CFG_NAND_BASE + NFC_NF_CFG2));
                temp = output;
                if (output & NFC_INT) {
                        out_be16((u16 *)(CFG_NAND_BASE + NFC_NF_CFG2), 0x0);
                        break;
                } else
                        for (i = 1000; i > 0; i--) {
```

```
                                if (temp >= 0x8ffa)
                                        break;
                        }
                if (max_retries <= 0)
                        max_retries += 10000;
        }
        return;
}

void nandload(void)
{
        /* nand init */
        out_be16((u16 *)(CFG_NAND_BASE + NFC_NFC_CFG), 0x0002);
        out_be16((u16 *)(CFG_NAND_BASE + NFC_SPAS), 0x0020);
        out_be16((u16 *)(CFG_NAND_BASE + NFC_NF_CFG1), 0x0cb2);

        out_be16((u16 *)(CFG_NAND_BASE + NFC_FLASH_CMD), NAND_CMD_READ0);
        out_be16((u16 *)(CFG_NAND_BASE + NFC_NF_CFG2), NFC_CMD);
        wait_op_mjmdone();
        out_be16((u16 *)(CFG_NAND_BASE + NFC_FLASH_ADDR), 0x0);
        out_be16((u16 *)(CFG_NAND_BASE + NFC_NF_CFG2), NFC_ADDR);
        wait_op_mjmdone();
        out_be16((u16 *)(CFG_NAND_BASE + NFC_NF_CFG2), NFC_ADDR);
        wait_op_mjmdone();
        out_be16((u16 *)(CFG_NAND_BASE + NFC_NF_CFG2), NFC_ADDR);
        wait_op_mjmdone();
        out_be16((u16 *)(CFG_NAND_BASE + NFC_NF_CFG2), NFC_ADDR);
        wait_op_mjmdone();
        out_be16((u16 *)(CFG_NAND_BASE + NFC_NF_CFG2), NFC_ADDR);
        wait_op_mjmdone();
        out_be16((u16 *)(CFG_NAND_BASE + NFC_FLASH_CMD), NAND_CMD_READCACHE);
        out_be16((u16 *)(CFG_NAND_BASE + NFC_NF_CFG2), NFC_CMD);
        wait_op_mjmdone();
        unsigned long *i, *j;
        int k;
        int num_pages = NUMPAGES;
        unsigned long *mem_idx = (unsigned long *)0x100;

        i = (unsigned long *)CFG_NAND_U_BOOT_DST;
        do {
                /* readout a page and copy to mem --
                   cache mode means we can skip the address cycle
                 */
                out_be16((u16 *)(CFG_NAND_BASE + NFC_RAM_BUF_ADDR),
                        RAM_BUFFER_ADDRESS_RBA_4);
                out_be16((u16 *)(CFG_NAND_BASE + NFC_NF_CFG2), NFC_OUTPUT);
                /* Wait for operation to complete */
                wait_op_mjmdone();

                for (j = (u32 *)(CFG_NAND_BASE + TWO_K), k = 0;
                                k < TWO_K/4; i++, j++, k++) {
                        *i = *j;
                /* put exception vectors at 0x0 - vector size is 0x13ff */
                        if (i >= (u32 *)(CFG_NAND_U_BOOT_DST + TWO_K + 0x100)
                                        && mem_idx < (u32 *)0x1400)
                                (*(mem_idx++)) = *j;
                }
```

```
                num_pages--;
        } while (num_pages > 0);

        out_be16((u16 *)(CFG_NAND_BASE + NFC_FLASH_CMD), NAND_CMD_READCACHEND);
        out_be16((u16 *)(CFG_NAND_BASE + NFC_NF_CFG2), NFC_CMD);
        wait_op_mjmdone();
        return;
}
```

Appendix A ADS512101 Top Board Layout



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 +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3845
Rev. 0
05/2009

freescale™
semiconductor