Ethernet Boot Loader for the Intel® SA-111X Development Platform, Version 4.5, for Microsoft Windows* CE

User's Guide

November 2000

Order Number: 278355-001

User's Guide

Contents

1.0 Overview

The Ethernet Boot Loader (hereafter called Eboot) included in this kit is designed exclusively for use with the Intel® StrongARM* SA-1110 Development Board (SA-1110 Development Board), the Intel® StrongARM* SA-1111 Development Module (SA-1111 Development Module), and the combined platform for these two modules which is the Intel® StrongARM* SA-1110 Development Platform (SA-1110 Development Platform). Eboot has one primary purpose, that is to provide a means by which the software developer can quickly and efficiently transfer an image of the Windows* CE operating system, drivers and associated applications to the target over the Ethernet through the use of either the Microsoft Windows CE Platform Builder Integrated Development Environment (IDE) or the Microsoft CEShell utility.

This kit contains the full source code, binaries and build documentation for two different Ethernet boot loaders. The project contained in "ebootNE2K" supports the NE2000 chipset through the PCMCIA/CF interface. The project contained in "ebootSMC" supports the SMC 91C96 Ethernet controller, the native device on the SA-1111 development module.

1.1 Related Documentation

Other documentation that may be helpful while reading this document are:

- *The Intel® StrongARM* SA-1110 Microprocessor Development Board User's Guide*, order number 278278, and the documents listed in the Related Documentation section.
- The *Intel® StrongARM* SA-1110/ SA-1111 Development Kit Quick Start Procedures User's Guide*, order number 278339.

1.2 Requirements:

The following lists the software requirements for the procedures in this document:

- You must install Microsoft Windows* CE Platform Builder version 2.12 or 3.00 in order to successfully build either project.

Note: Minor modifications to the source code are required for building under the Microsoft Windows* CE Platform Builder 2.12 tool set. For more information on Microsoft Windows* CE Platform Builder 2.12 modifications, see Section 1.4.2.

- Intel's JFLASH Version 1.2 utility is required to transfer the boot loader binary to the target device. You can find JFLASH at the following URL: **http://developer.intel.com/design/strong/swsup/**
- You must choose at least the following options when you install Platform Builder:
  - — Select the SA1100 CPU target type (the selection SA-1100 supports SA-1100 and SA-1110 microprocessors.)
  - — Under the options heading, choose "Platform Builder IDE", and "Developing and Debugging." Additionally, choose to install the Windows CE Operating System components.
- Make sure to "Register Environment Variables" when prompted.

1.3 Kit Installation Instructions:

Use the following instructions to install Eboot on your host system.

1. Unzip the archive into your temp directory. An EBOOT directory will be created which contains all of the files referenced here.
2. Copy the ebootNE2k.release and ebootSMC.release directories into the root of your Platform Builder installation, typically C:\WinCE300
3. Copy the SA11X0BD directory into your PLATFORM directory, typically C:\WinCE300\PLATFORM. Please note that this SA11X0BD source tree may not be compatible with your current source tree. If you already have an SA11X0BD source tree installed, please be sure to make a backup copy of your original tree before proceeding.
4. Copy both ne2k.bat and smc.bat into your Platform Builder root directory.
5. Create the following directories under your PUBLIC directory, typically C:\WinCE300\PUBLIC
   - EBOOT
   - EBOOT\cesysgen
   - EBOOT\cesysgen\sdk
   - EBOOT\cesysgen\sdk\inc
   - EBOOT\cesysgen\sdk\lib\arm\sa1100\ce\debug
   - EBOOT\cesysgen\oak
   - EBOOT\cesysgen\oak\inc
   - EBOOT\cesysgen\oak\lib\arm\sa1100\ce\debug
6. Copy the contents of:
   a. PUBLIC\COMMON\SDK\INC into PUBLIC\EBOOT\CESYSGEN\SDK\INC
   b. PUBLIC\COMMON\SDK\LIB\ARM\SA1100\CE\DEBUG into PUBLIC\EBOOT\CESYSGEN\SDK\LIB\ARM\SA1100\CE\DEBUG
   c. PUBLIC\COMMON\OAK\INC into PUBLIC\EBOOT\CESYSGEN\OAK\INC
   d. PUBLIC\COMMON\OAK\LIB\ARM\SA1100\CE\DEBUG into PUBLIC\EBOOT\CESYSGEN\OAK\LIB\ARM\SA1100\CE\DEBUG
7. Create an EBOOT shortcut with the following command line (target):

```
C:\WINNT\system32\CMD.EXE /k c:\WINCE300\public\common\oak\misc\WinCE.bat
ARM SA1100 CE EBOOT SA11X0BD
```

Note: Make sure you use the appropriate path for CMD.EXE and your Platform Builder root. WinCE.bat uses variables that are case sensitive. Be sure to maintain the capitalization when creating this shortcut. Place this shortcut into any convenient location.

1.4 Build Instructions

Use the following instructions to build Eboot on your host system.

1. Click on the EBOOT shortcut that you made above. This will launch WinCE.bat and setup some environment variables for use with this build. A DOS* window will remain after this step. (Ignore any “Environment variable PLAT_DBGSER_UART1 not defined” messages.)
2. To build the boot loader that supports the NE2000 chipset through the PCMCIA/CF interface, from within this same DOS window, run the ne2k.bat file which is now located in your Platform Builder root. To build the boot loader that supports the SMC 91C96 Ethernet controller, run the smc.bat file instead.

Note: Each batch file sets up additional environment variables, changes to the appropriate source directory, and launches the build. Note that there are two components to the SMC boot loader build: the project that builds the smcb.lib and the boot loader project itself. Note that the ne2k build procedure does not build the ne2kdbg.lib file, as this library comes with the Platform Builder kit.

When the build completes, a binary image called EBOOT.EXE will be created. The final step in the build procedure uses ROMIMAGE.EXE (included with Platform Builder) to create a flashable image called EBOOT.NB0. This step is accomplished automatically at the end of the build when you run either of the build batch files included with this kit. The EBOOT.NB0 file will also be placed into the appropriate release directory, either %_winceroot%\ebootne2k.release or %_winceroot%\ebootsmc.release.

Once the EBOOT.NB0 file has been created, the build process is complete.

1.4.1 Eboot Image Loading Instructions

Use the following procedure to load the Eboot image into flash memory.

1. Copy the EBOOT.NB0 image into your JFLASH directory.

Note: If you are using the SA-1111 Development Module, set switch pack 2 (SW2) switch 8 into the appropriate position. In the ON position, the flash memory on the SA-1110 Development Board will be programmed and the device will boot from the SA-1110 Development Board. In the OFF position, the flash memory on the SA-1111 Development Module will be programmed and the device will boot from the SA-1111 Development Module.

2. Verify that the programming cable supplied with the SA-1110 Development kit is connected to both your host computer and the target device.
3. Apply power to your development target.
4. Run JFLASH with EBOOT.NB0 as the source image. (For example, run C:\JFLASH\JFLASH.EXE EBOOT.NB0)

After JFLASH finishes the verification step, you are ready to use the system to download your images.

For more information about JFLASH, see the JFLASH documentation and the section on changing the installed boot executable in the *Intel® StrongARM* SA-1110/SA-1111 Development Kit Quick Start Procedures.*

1.4.2 Notes of Interest

This section contains additional information and notes.

- Both the Platform Builder IDE and CEShell need to know the appropriate ID of the device you will communicate with in order to establish a communications link. Make sure to choose the appropriate ID of the device you will communicate with (i.e. SA11X0BD48454, SA11X0BD256). If you do not choose the appropriate device ID, neither the IDE nor the CEShell will be able to communicate with the target.
- Hold down button 2 on the SA-1110 Development Board while applying power. This selects the use of the boot loader on powerup. If you do not hold down button 2, the preloaded Diagnostic Manager program will be launched instead if it is installed in high FLASH memory. If you wish to reverse this behavior, edit fwsarm.s. Locate and change the instruction from:

```
bne    NoJumptoDiags
```

  to

```
beq    NoJumptoDiags
```

- The SMC 91C96 Ethernet Controller does not come programmed from the factory with an individual address (MAC address). At power up, this device reads its MAC address from an EEPROM located on the SA-1111 Development Module and stores this data into its native SRAM. When the SMC boot loader initializes the system, it reads the contents of the controller's SRAM and validates the value of the MAC address. Any value that is not (0xFFFF, 0xFFFF, 0xFFFF) is considered valid. If an invalid MAC address is read, the boot loader will program a predefined MAC address into the EEPROM and SRAM (currently 0x0800, 0x0F00, 0x0100). This scheme works well in a development and test environment with a private network but should be changed for use in a production environment in order to avoid network collisions. That is, every Ethernet controller should be programmed with a unique MAC address before being placed onto a public network. If you wish to examine the source code that writes this predefined MAC address, please refer to the SMCInit() function in the smc.c source file.
- The HyperTerminal should be configured to establish a connection at 38400 baud, 8 data bits, no parity, and 1 stop bit (8,N,1). Flow control should be disabled (i.e. none).
- If you build the software using the Platform Builder 2.12 toolset, two minor changes are necessary for a successful build.

  a. In main.c, change the current EbootSendBootme call from:

```
EbootSendBootme(&MyAddr, EBOOT_VERSION_MAJOR,
EBOOT_VERSION_MINOR, PLATFORM_STRING, NULL, EDBG_CPUID, 0);
```

  to:

```
EbootSendBootme(&MyAddr, EBOOT_VERSION_MAJOR,
EBOOT_VERSION_MINOR, PLATFORM_STRING, NULL, EDBG_CPUID);
```

  b. Additionally, change the function declaration for NKDbgPrintfW from:

```
void WINAPIV NKDbgPrintfW(LPCWSTR lpszFmt, ...)
```

  to:

```
void WINAPIV NKDbgPrintfW(LPWSTR lpszFmt, ...)
```