Freescale Semiconductor
Application Note

Document Number: AN3437
Rev. 0, 08/2007

Booting the MSC8144 Through the PCI Interface

by *Barbara Johnson*
NCSD DSP Applications
Freescale Semiconductor, Inc.

In a typical DSP system, a host processor downloads program and data to the slave DSP device after a reset sequence. When the download completes, the DSP jumps to the start of the user program and begins execution. The StarCore®-based MSC8144 DSP supports not only this typical scenario but also booting from an external host through the PCI interface.

This document describes how to implement a software driver to boot the MSC8144 DSP through the PCI interface from an external host device. In this example, the user program and data are downloaded from the host PowerQUICC™ MPC8560 to the target MSC8144 using the MSC8144ADS board. The software driver code is available in a file named `AN3437SW.zip` that you can download at the Freescale web site listed on the back cover of this document. Also available on this web site is the *MSC8144 Reference Manual*, which provides details on the MSC8144 boot program and the PCI controller.

Contents

1 PCI Boot Basics

After a reset sequence, all SC3400 cores jump to the ROM starting address at 0xFEF00000 and execute the boot code to initialize the MSC8144. The boot code in the ROM performs both private and shared configurations. Private configuration includes core-dependent initialization, and shared configuration initializes the entire system-on-a-chip (SoC).

After initialization, the boot code determines the selected boot port from the BPRT field of the reset configuration word high register (RCWHR). Table 1 shows the valid PCI boot modes based on various DDR memory sizes.

Table 1. MSC8144 PCI Boot Modes

RCWHR[BPRT]	PCI Boot Mode
0b000000	PCI with no DDR
0b000001	PCI using single DDR 32 Mbyte
0b000010	PCI 256 Mbyte (Default)
0b000011	PCI using single DDR 64 Mbyte
0b000100	PCI using single DDR 128 Mbyte
0b000101	PCI using DDR 512 Mbyte

When the boot code determines that the boot port is the PCI interface, it configures three PCI inbound address windows as shown in Table 2 so that the external host can access the MSC8144 internal memory space. Note that although PCI inbound window 2 is configured to allow access to the DDR memory space, the boot code does not initialize the DDR controller because it has no knowledge of the type of DDR device that is connected to the MSC8144. Therefore, do not download program or data to the DDR memory without first configuring the DDR controller memory-mapped registers.

Table 2. PCI Inbound Windows

PCI Window	Memory	Size	Start Address
Inbound 0	M2	512 Kbyte	0xC0000000
Inbound 1	M3	16 Mbyte	0xD0000000
Inbound 2	DDR	Depends on RCWHR[BPRT]	0x40000000

After the boot code finishes PCI initialization, the MSC8144 is ready to receive the user program and data from the host. The MSC8144 boot code initiates a handshake protocol by writing the predefined value of 0x17171717 to the designated address in M2 memory at 0xC007B000. The host device polls this address until the predefined value is read. Then the host accesses the MSC8144 memory space through PCI and begins to download the user program and data. When the host is finished downloading, it writes the predefined value of 0xA5A5A5A5 to the same designated address. The MSC8144 polls this address until it reads the predefined value.

Before jumping to the user code, the boot code performs cleanup tasks such as disabling all interrupts except NMIs, disabling GPIO signals, closing the program MMU windows, and invalidating the ICache. When cleanup completes, the MSC8144 writes the value 0x900D900D to M2 memory address

0xC007B00C and then jumps to the user code and begins executing it. The handshaking protocol is summarized in Table 3.

Table 3. MSC8144 PCI Boot Handshaking Values

Address	Predefined Value	Description
0xC007B000	0x17171717	The MSC8144 boot code writes 0x17171717 to address 0xC007B000 to indicate it is ready to receive the user program and data from the external host.
	0xA5A5A5A5	The MSC8144 boot code polls address 0xC007B000 until it reads the value 0xA5A5A5A5 to determine whether the host has completed the user program/data download.
0xC007B00C	0x900D900D	Before it jumps to the start of the user program, the MSC8144 boot code writes 0x900D900D to address 0xC007B00C to indicate the end of boot.

Figure 1 illustrates the flow of the PCI boot sequence.

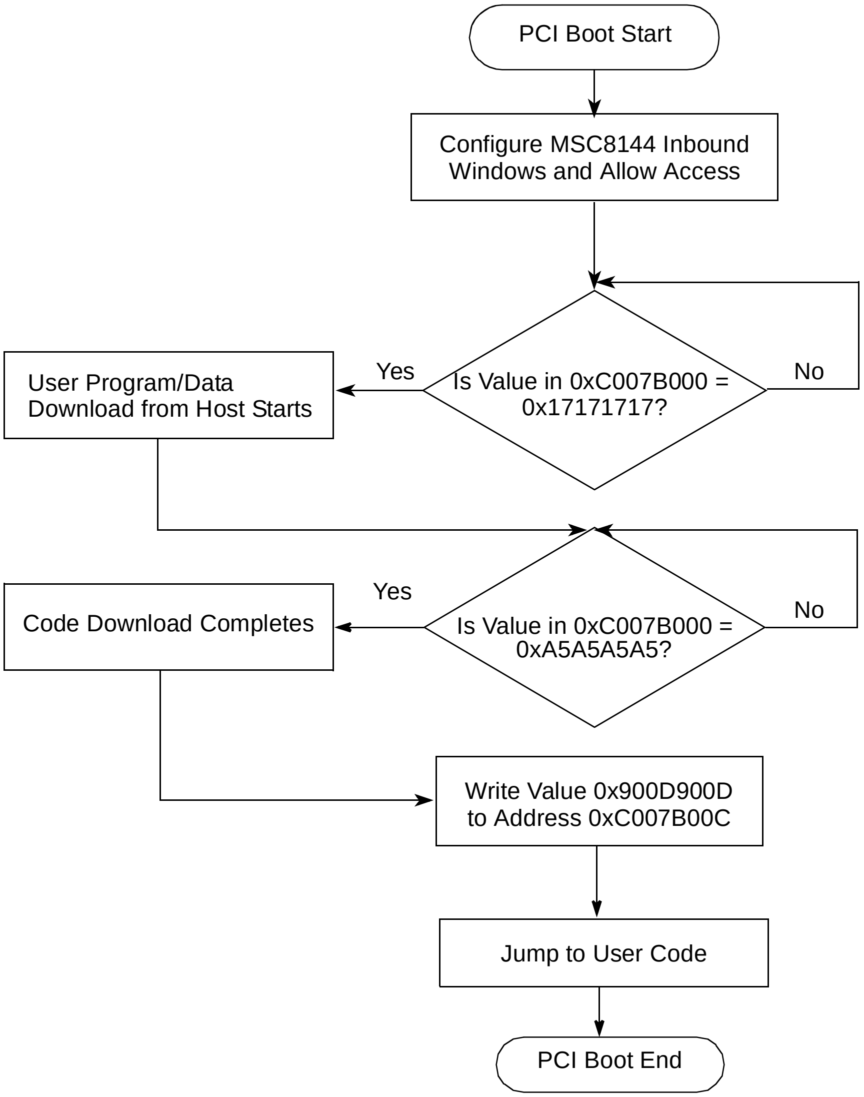
PCI Boot Start
Configure MSC8144 Inbound Windows and Allow Access
Is Value in 0xC007B000 = 0x17171717?
Yes
No
User Program/Data Download from Host Starts
Is Value in 0xC007B000 = 0xA5A5A5A5?
Yes
No
Code Download Completes
Write Value 0x900D900D to Address 0xC007B00C
Jump to User Code
PCI Boot End

Figure 1. MSC8144 PCI Boot Flow

2 Requirements and Setup

All tests described in this document were performed on the MSC8144 Application Development System (MSC8144ADS). The MSC8144ADS board contains an MPC8560 host processor that connects to the MSC8144 DSP through the PCI bus.

2.1 Hardware Requirements

The following items are required to run the boot example presented in this document:

- MSC8144ADS board
- PC with CodeWarrior™ for PowerPC Processors, version 8.7 or later
- USB Tap for COP. Connects the MPC8560 on the MSC8144ADS to the CodeWarrior for PowerPC tools through JTAG. Figure 2 shows the hardware setup.

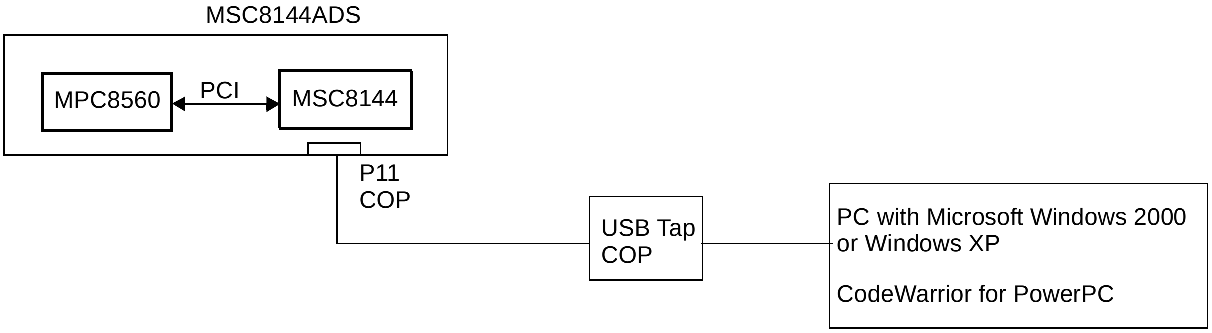

Figure 2. Hardware Setup

2.2 Switch Settings

Table 4 shows the switch settings to connect the MPC8560 to the debugger tools and allow the MSC8144 DSP to boot from PCI. For details on switch settings, refer to the *MSC8144ADS Processor Board Reference Manual* (MSC8144ADSRM).

Table 4. MSC8144ADS Switch Settings

Switch	Settings 1–8	Description
SW1	00000110	Default setting.
SW2	01111111	Host MPC8560 operates normally, disable JTAG chain, MSC8144 does not enter debug after reset.
SW3	10010111	Default setting.
SW4	01000010	Reset configuration word from I^{2}C EEPROM.
Note: 0 = ON, 1 = OFF		

2.3 Reset Configuration Word Setting

To boot from PCI, the MSC8144 reset configuration word high register (RCWHR) must be programmed so that the PCI boot port is selected as shown in Table 5. The I^{2}C EEPROM may need to be reprogrammed to ensure the value shown below get loaded to the RCWHR after reset.

Table 5. MSC8144 RCWHR Settings

RCWHR Setting	Description
RCWHR[BPRT] = 000010	Boot from PCI. DDR is configured to be 256 Mbyte in size.

3 Running the Example MPC8560 Host Driver

When you run the MPC8560 host driver, the MPC8560 processor downloads the user program and data to the MSC8144 DSP through the PCI interface. The user program can be opened using CodeWarrior for StarCore and and can be found in `\8144usercode\8144 PCI Agent.mcp` included in the `AN3437SW.zip` file.The user program performs the simple task of flashing two LEDs on the MSC8144ADS:

- Red LED (LD4)
- Green LED (LD5)

To run the example MPC8560 host driver, perform the following steps:

1. Open the CodeWarrior for PowerPC debugger tools.
2. Load the project `8560hostdriver.mcp`.
3. Run the project.
4. Verify that LD4 and LD5 are flashing to indicate that the boot is successful.

4 MPC8560 Host Driver Phases

The MPC8560 software driver included with this document operates in two phases: PCI initialization and user program/data download as shown in Figure 3. The host software driver can be found in `\8560hostdriver\8560hostdriver.mcp` included in the `AN3437SW.zip` file.

4.1 PCI Initialization

When the MSC8144 boots from the PCI interface, the boot code in ROM configures three PCI translation inbound windows so that these memory spaces are accessible to the host. The boot code sets up inbound windows for M2, M3, and DDR memory spaces. The size of the DDR space depends on the value programmed in the RCWHR[BPRT] field.

The host driver must configure its own outbound windows and it must also configure the PCI memory space to map to the MSC8144 inbound windows. After these windows are configured, MPC8560 accesses to these outbound windows are forwarded to the MSC8144 inbound windows.

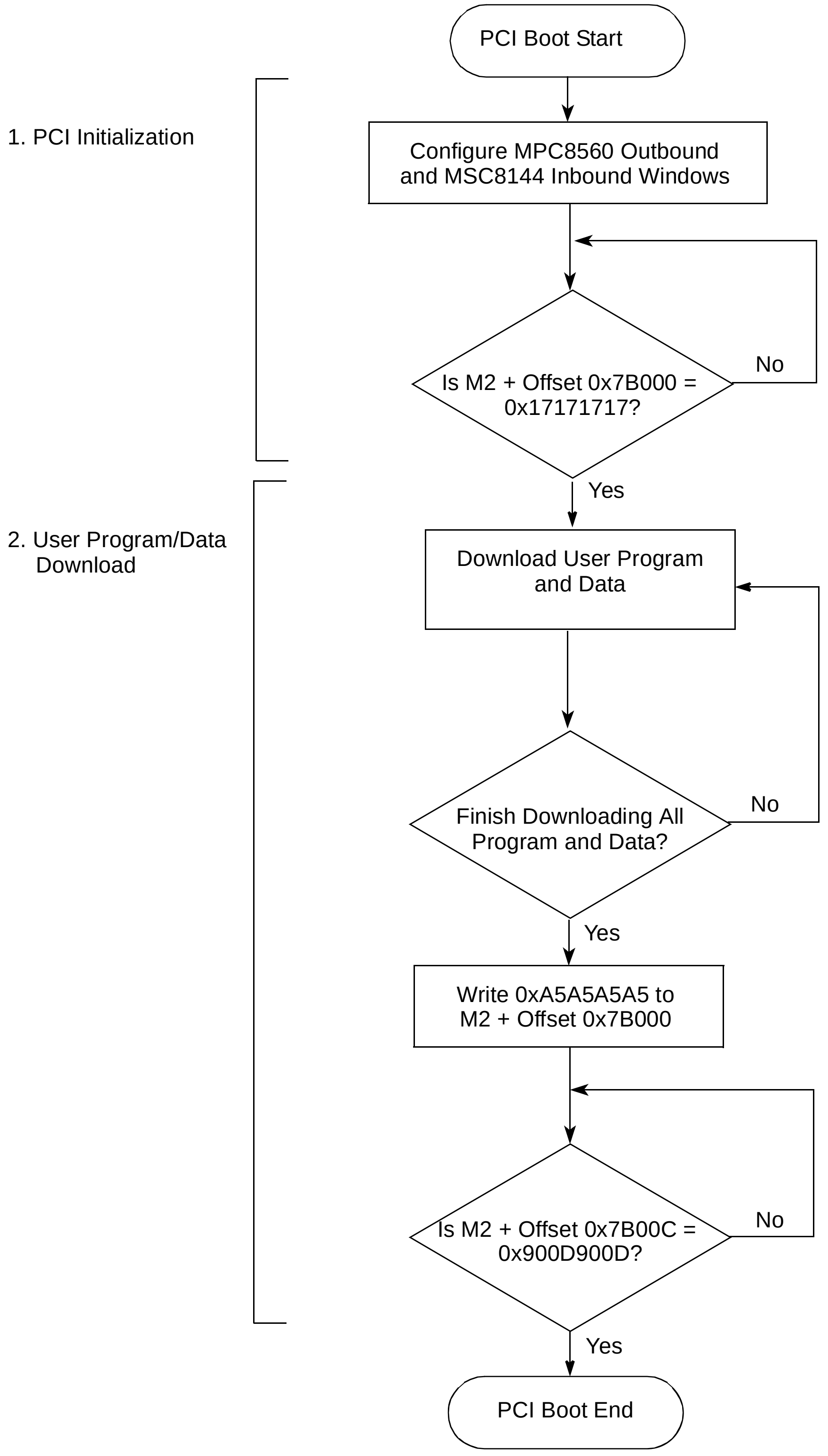
PCI Boot Start
1. PCI Initialization
Configure MPC8560 Outbound and MSC8144 Inbound Windows
Is M2 + Offset 0x7B000 = 0x17171717?
No
Yes
2. User Program/Data Download
Download User Program and Data
Finish Downloading All Program and Data?
No
Yes
Write 0xA5A5A5A5 to M2 + Offset 0x7B000
Is M2 + Offset 0x7B00C = 0x900D900D?
No
Yes
PCI Boot End

Figure 3. MPC8560 PCI Download Flow

In this example, the outbound and inbound windows are mapped as shown in Figure 4. When the MPC8560 processor accesses its outbound window 1 starting at address 0x80000000, it is actually accessing the MSC8144 M2 memory space starting at 0xC0000000. Similarly, when the MPC8560 accesses the outbound window 2 starting at 0x81000000, it is actually accessing the MSC8144 M3 memory starting at 0xD0000000.

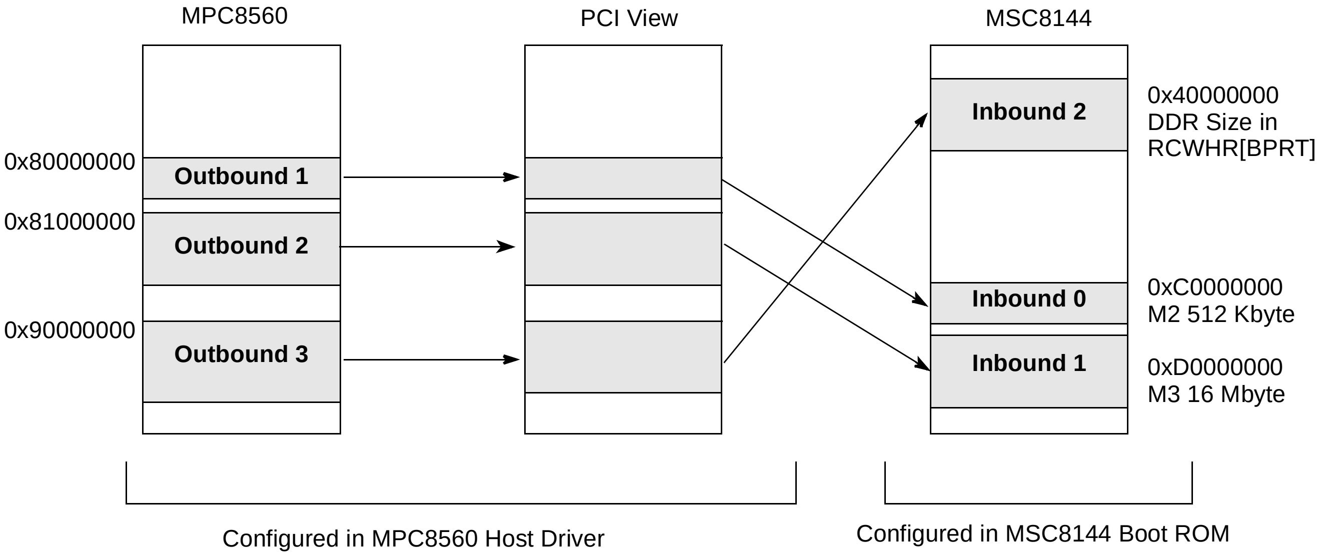

Figure 4. PCI Inbound and Outbound Window Mapping

Example 1 shows the PCI initialization performed by the host.

Example 1. MPC8560 Code Outbound/Inbound Window Mapping

```
// ************************************************************

// 8560 Outbound / 8144 Inbound
// 8560 0x8000_0000 --> PCI 0x8000_0000 --> 8144 0xC000_0000

// ************************************************************

// Determine window sizes for 8144 GPLx
size = getWindowSize(BusNum, MSC8144_PCIDEVICENUM, REG_GPLBAR0); //M2

// Assign 8144 inbound window 0 start addr in PCI memory space
setPCIConfigReg32(BusNum, MSC8144_PCIDEVICENUM, REG_GPLBAR0, OutboundPCItoM2);

// Set outbound window local address, PCI address
setOutbound0(OutboundLocaltoM2, OutboundPCItoM2, calcWindowSize(size));

// ************************************************************
// 8560 Outbound / 8144 Inbound
// 8560 0x8100_0000 --> PCI 0x8100_0000 --> 8144 0xD000_0000

// ************************************************************

// Determine window sizes for 8144 GPLx
size = getWindowSize(BusNum, MSC8144_PCIDEVICENUM, REG_GPLBAR1); //M3

// Assign 8144 inbound window 1 start addr in PCI memory space
setPCIConfigReg32(BusNum, MSC8144_PCIDEVICENUM, REG_GPLBAR1, OutboundPCItoM3);

// Set outbound window local address, PCI address
setOutbound1(OutboundLocaltoM3, OutboundPCItoM3, calcWindowSize(size));

// ************************************************************
// 8560 Outbound / 8144 Inbound
// 8560 0x9000_0000 --> PCI 0x9000_0000 --> 8144 0x4000_0000
// *************************************************************

// Determine window sizes for 8144 GPLx
size = getWindowSize(BusNum, MSC8144_PCIDEVICENUM, REG_GPLBAR2); //DDR

// Assign 8144 inbound window 2 start addr in PCI memory space
setPCIConfigReg32(BusNum, MSC8144_PCIDEVICENUM, REG_GPLBAR2, OutboundPCItoDDR);

// Set outbound window local address, PCI address
setOutbound2(OutboundLocaltoDDR, OutboundPCItoDDR, calcWindowSize(size));
```

4.2 User Program/Data Download

After the MSC8144 boot code configures its inbound windows in the boot ROM, it writes the value 0x17171717 to M2 memory location 0xC007B000. In this example, this address is mapped to 0x8007B000 on the host side, so the MPC8560 host reads address 0x8007B000 to determine whether the MSC8144 completed the PCI initialization and is ready for download. If the MSC8144 is not ready, the MPC8560 waits until the predefined value is read from that memory location.

Next, the download is ready to begin. In this example, the user program/data image is stored in the `boot[]` array which contains a 32-bit address followed by a 32-bit data. The 32-bit data represents the opcode or data, and the 32-bit address specifies where the opcode or data is to reside. The host performs the download by writing the data to the corresponding address in the array. Example 2 shows `boot[]` array values. The host writes the value 0x9F792118 to address 0x80041000. In the next access, the host writes 0x80003860 to address 0x80041004.

Example 2. `boot[]` Array Example

```
unsigned int boot [] =
{
       ...
       0x80041000, 0x9F792118,
       0x80041004, 0x80003860,
       ...
}
```

When all the data in the array is downloaded, the host writes the value 0xA5A5A5A5 to address 0x8007B000 to indicate the end of the download to the MSC8144. The MSC8144 polls this address until the predefined value is read. Then it performs cleanup tasks and writes the value 0x900D900D to M2 memory address 0xC007B00C to indicate that the boot sequence is complete. Finally, it jumps to the user code. Example 3 shows the MPC8560 host PCI boot software.

Example 3. MPC8560 PCI Host Driver

```
// *************************************************************
// Start downloading code to MSC8144
// *************************************************************

do

{
         value = *(uint32_t *)(OutboundLocaltoM2 + 0x7B000);
         delay = 0xFFFF;
         while (delay--);
} while (value != SwapLong(0x17171717));

// Write boot code to 8144
for (i=0 ; i<boot_code_size ; i+=2)

{
        addr = boot[i];
        data = boot[i+1];
        if (addr == 0x8007B010)
                 asm(" nop");
        *(uint32_t *)addr = SwapLong(data);
}


//Finalize handshake with 8144
*(uint32_t *)(OutboundLocaltoM2 + 0x7B000) = SwapLong(0xA5A5A5A5);

// Poll 8144 until boot is done
do

{

        value = *(uint32_t *)(OutboundLocaltoM2 + 0x7B00C);
        delay = 0xFFFF;
        while (delay--);
} while (value != SwapLong(0x900D900D));
while(1);
```

5 Creating the Image for Download

The user program and data that are downloaded to the MSC8144 DSP from the MPC8560 processor must be in a correct PCI-bootable format. Creating the boot image involves two steps:

1. Convert the object files to an S-record.
2. Convert the S-record to a PCI-bootable format.

5.1 Generating the S-record

When the user program is compiled, an object file is created for each of the four cores. The object filename has a `.eld` extension. To create a combined S-record of the `.eld` files of all four cores, the `sc100-elf2xx` utility must be used. This program comes with the CodeWarrior for StarCore tools and can be found in the `\CodeWarrior\StarCore_Support\compiler\bin` directory. Note that even if the user program uses only one core, private data for the other cores must also be included in the S-record for proper boot operation. For example, suppose the user program generates the following output files for each of the cores:

- Core 0, `project.eld`
- Core 1, `c1_project.eld`
- Core 2, `c2_project.eld`
- Core 3, `c3_project.eld`

To create an S-record output called `led_output.s` from the combined object files, the `sc100-elf2xx` is called as shown in Example 4. The `-t srec` selects an S-record output format. The `-o led_output.s` specifies the name of the output file. The `-m msc8144` selects the device. Finally, all four input `.eld` files specify the object code to combine.

Example 4. Generating the S-record

```
C:\Program Files\Freescale\CodeWarrior\StarCore_Support\compiler\bin>

sc100-elf2xx -t srec -o led_output.s -m msc8144 project.eld c1_project.eld c2_project.eld
c3_project.eld
```

5.2 Generating the Download Image for the Host

When the S-record is generated, it must be translated in such a way that memory addresses in the S-record file are converted into the corresponding PCI outbound addresses that the MPC8560 accesses for download. One way to convert the S-record file is to create a script that reads the each line in the S-record file and parses information into addresses and data as shown in Figure 5.

Note that the original S-record file contains addresses as viewed by the MSC8144. For example, address 0xC0041000 is in MSC8144 M2 memory range. However, the output of the S-record parser must contain the corresponding address as viewed by the host. In this case, this address is accessed by the host at 0x80041000. The S-record parser must also perform the address translation for downloads to M3 memory.

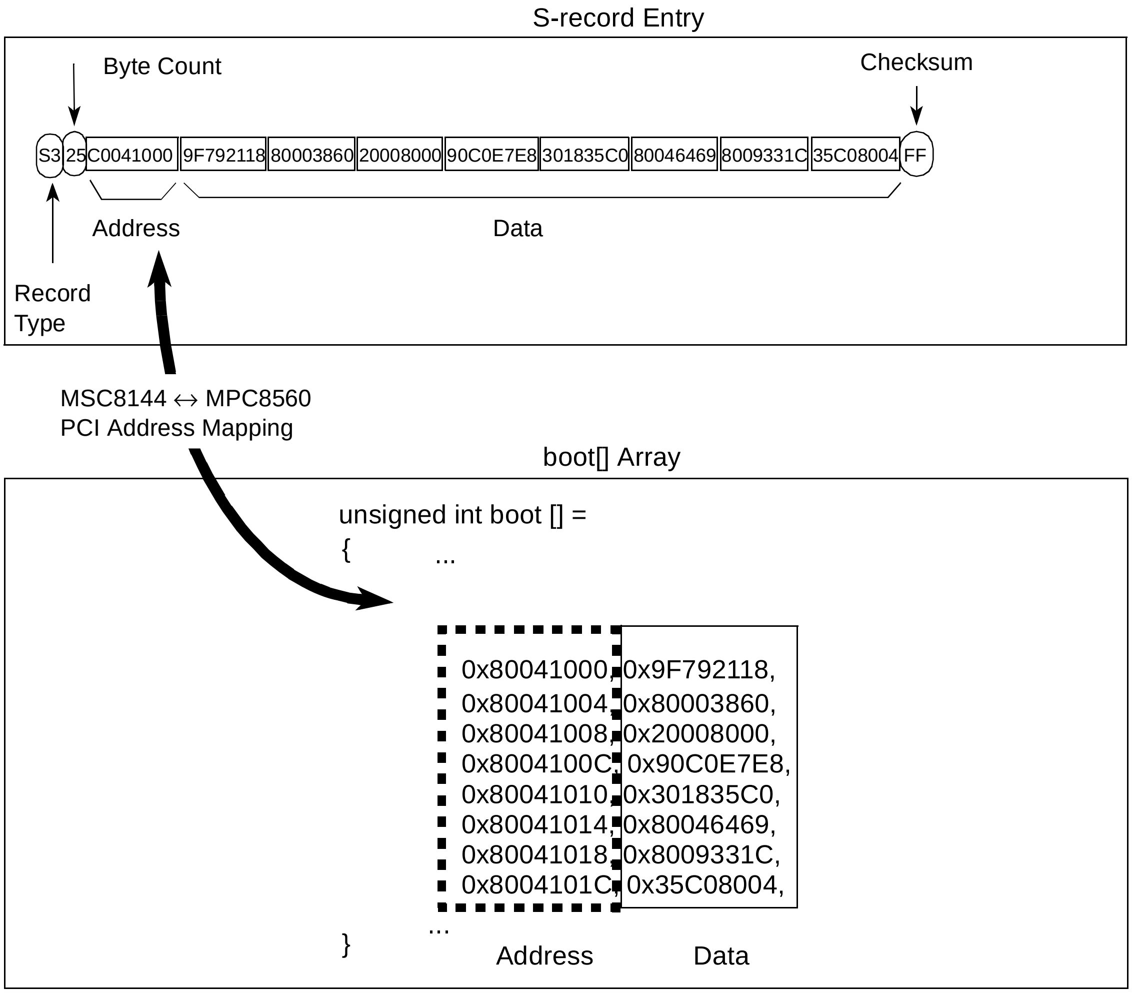

Figure 5. Generating PCI-Bootable File

This application note includes a Perl script, called `msc8144_translate.pl`, that creates a file with an array of addresses and data. The S-record parser generates an array called `boot[]` that contains a group of 32-bit address followed by 32-bit data. The MPC8560 downloads the user code by writing the data to the corresponding address.

Example 5 shows how to call the script to create a file called `led_output.h` from the original `led_output.s` S-record file.

Example 5. Generating Download Image for Host

```
C:\Program Files\Perl\bin>
perl 8144_translate.pl -host pci -in led_ouput.s -out led_output.h
```

After the `led_output.h` file is created, it must be included in the host driver software as shown here:

```
#include "led_output.h"
```

6 Summary

The following list summarizes the overall procedure for booting the MSC8144 through the PCI interface:

1. Set the appropriate MSC8144 boot port in RCWHR[BPRT] as shown in Table 1.
2. In the host driver, set up the address mapping between the host outbound windows and the MSC8144 inbound windows as discussed in Section 4.1. The PCI address mapping allows the host to access the MSC8144 internal memory space. This step must be completed before the program and data can be downloaded.
3. In the host driver, follow the handshaking protocol between the host and the MSC8144 as shown in Table 3. The host reads from and writes to designated memory locations in the MSC8144 M2 memory to determine when the MSC8144 is ready for the download, to terminate the download, and to determine when the MSC8144 completes the boot sequence.
4. Generate the S-record of the user program and data using the `sc100-elf2xx` tool as discussed in Section 5.1. Program and data should be in placed in M2 and/or M3 memory. Because the boot code does not initialize the DDR controller, the host should not download to the DDR memory space.
5. Use an S-record parser to create a PCI-bootable file as discussed in Section 5.2. The generated file should include the address and the program opcode or data. The addresses must be the addresses that the host sees as configured in the PCI address mapping.

7 Revision History

Table 6 provides a revision history for this application note.

Table 6. Document Revision History

Rev. Number	Date	Substantive Change(s)
0	08/2007	Initial release.

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 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

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
+1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
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 which 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.

Document Number: AN3437
Rev. 0
08/2007