

Rev. 0.1, 03/2004

*Adding New PowerPC™  
Processors and Bridge Chips  
to Linux*

*Maurie Ommerman  
risc10@email.sps.  
mot.com*

This application note describes the steps and code changes needed to support the Linux kernel on a new PowerPC™ processor with a newly assigned processor version register (PVR). Similarly, code changes to support a new bridge chip are provided.

The following topics are addressed:

| <b>Topic</b>  | <b>Page</b> |
|---|-------------|
| Section 1, "Introduction"   | 2           |
| Section 2, "Terminology"  | 2           |
| Section 3, "Building the Kernel"                                  | 2           |
| Section 4, "Changing the Source Code Files for a New Processor"   | 3           |
| Section 5, "Changing the Source Code Files for a New Bridge Chip" | 5           |
| Section 6, "Building the New Kernel"                              | 6           |
| Section 7, "Loading and Booting the New Kernel"                   | 6           |
| Section 8, "References"   | 6           |
| Section 9, "Documentation Revision History"                       | 6           |

# 1 Introduction

The Linux kernel must recognize a PowerPC processor and its associated bridge chip before it can boot itself. This is done by searching the `cpu_spec` table in `cputable.h` for the processor's PVR. In addition, two files, `cpu_setup_6xx.S` and `cputable.c`, must be changed to recognize the new PowerPC processor. The files, `include/asm-ppc/mpc10x.h` and `arch/ppc/kernel/mpc10x_common.c`, must be changed to accommodate a new bridge chip. If the Linux kernel does not recognize the new PowerPC processor PVR, the boot sequence will hang after the message "Now booting the kernel".

This application note outlines the changes and steps to perform when adding a new PowerPC processor and its bridge chip to Linux. A PowerPC MPC7447A processor is added to the Linux kernel, kernel 2.4.21-rc1, as an illustrative example. This kernel is available from the Freescale DINK32 web site.

## 2 Terminology

The following terms are used in this document:

PVR—Processor version register, a hardware register that is hardwired to identify each processor and bridge chip

## 3 Building the Kernel

It may seem strange to build the kernel before making any changes, but it is important to ensure that the current source tree builds without errors first. This is also a good time to ensure that the configuration is correct. Only the options that may not be set by default are described here; most other options are standard. Ensure that IDE hard drive support and Ethernet support are available.

1. Configure the kernel.
  - `make menuconfig`
  - Choose the following configuration items:
    - Platform support --->
      - (6xx/7xx/74xx/8260) Processor type
      - Sandpoint X3
      - AltiVec support (if the processor supports it, the MPC7447A supports it)
    - General setup ---->
      - Default bootloader kernel arguments
      - Initial kernel command string: 'root=/dev/hda3'
    - Network device support --->
      - Ethernet (10 or 100 Mbit) --->
      - Choose a driver for the Ethernet card.
2. Build the dependency files.
  - `make dep`
3. Make the original kernel image.
  - `make zImage`
4. Ensure that the build was successful. Ensure that the `zImage.sandpoint` executable was built in the `arch/ppc/boot/images` directory.
5. If there are any fatal errors or `zImage.sandpoint` was not built, fix the errors and build again.

## 4 Changing the Source Code Files for a New Processor

All the source file changes are in the directory, arch/ppc/kernel.

The structure, `cpu_spec`, is defined in the header file, `cputable.h`, in the directory, `include/asm-ppc`. No changes are required in `cputable.h`.

The file, `cputable.c`, contains the instantiation of the structure, `cpu_spec`.

On or near line 19 of `cputable.c` is a set of extern lines defining `__setup_cpu_<type>`. Find one that is similar to the new processor being added. In the example case, `__setup_cpu_745x` is the most similar to the MPC7447A. Duplicate the following line and change it to the new processor name:

```
extern void __setup_cpu_7447A(unsigned long offset, int cpu_nr, struct cpu_spec*
spec);
```

On or near line 54 are a series of definitions for the structure, `cpu_spec` `cpu_specs[]`. Again choose a definition that is close to the processor being added (the 7457 in the example). Duplicate this entry and modify it to describe the new processor. In the example, the only changes are the name, the PVR, and the removal of the L3 cache, which does not exist in the MPC7447A. Note that the PVR for MPC7447A is 0x80030000, which was not previously defined.

```
{ /* 7447A */
    0xffff0000, 0x80030000, "7447A",
    CPU_FTR_SPLIT_ID_CACHE | CPU_FTR_USE_TB | CPU_FTR_CAN_NAP |
    CPU_FTR_L2CR | CPU_FTR_ALTIVEC_COMP |
    CPU_FTR_HPTE_TABLE | CPU_FTR_SPEC7450 | CPU_FTR_NAP_DISABLE_L2_PR |
    CPU_FTR_HAS_HIGH_BATS,
    COMMON_PPC | PPC_FEATURE_HAS_ALTIVEC,
    32, 32,
    __setup_cpu_7447A
},
```

`__setup_cpu_7447A` is defined in the next file, `cpu_setup_6xx.S`, which contains the functions, `__setup_cpu_<processor>` and `setup_<processor>_specifics`.

On or near line 21 of `cpu_setup_6xx.S` are the global definitions for all the `cpu_setup` entry points. Again, choose one that is similar to the processor being added. The 745x was chosen for this example.

```
_GLOBAL(__setup_cpu_7447A)
    mflr    r4
    bl     setup_common_caches
    bl     setup_7447A_specifics
    mtlr   r4
    blr
```



On or near line 199 is the function for setting up the specifics of the various processors, in this case again, the 745x. Since the MPC7447A does not have an L3 cache, the code for the errata in that processor was removed and all the references from 745x to 7447A were modified.

```
/* MPC 7447A
 * Enable Store Gathering (SGE), Branch Folding (FOLD)
 * Branch History Table (BHTE), Branch Target ICache (BTIC)
 * Dynamic Power Management (DPM), Speculative (SPD)
 * Ensure our data cache instructions really operate.
 * Timebase has to be running or we wouldn't have made it here,
 * just ensure we don't disable it.
 * Clear Instruction cache throttling (ICTC)
 * Enable L2 HW prefetch
 */
setup_7447A_specifics:
    mfspr    r11,HID0

    /* All of the bits we have to set.....
    */
    ori     r11,r11,HID0_SGE | HID0_FOLD | HID0_BHTE | HID0_BTIC | HID0_LRSTK
BEGIN_FTR_SECTION
    oris    r11,r11,HID0_DPM@h    /* enable dynamic power mgmt */
END_FTR_SECTION_IFCLR(CPU_FTR_NO_DPM)

    /* All of the bits we have to clear....
    */
    li     r3,HID0_SPD | HID0_NOPDST | HID0_NOPTI
    andc   r11,r11,r3            /* clear SPD: enable speculative */
    li     r3,0

    mtspr  ICTC,r3              /* Instruction Cache Throttling off */
    isync

    mtspr  HID0,r11
    sync
```

```

isync

/* Enable L2 HW prefetch
*/

mfspr    r3,SPRN_MSSCR0
ori      r3,r3,3
sync

mtspr    SPRN_MSSCR0,r3
sync

isync

blr

// end setup_7447A_specifics
    
```

## 5 Changing the Source Code Files for a New Bridge Chip

Two files, include/asm-ppc/mpc10x.h and arch/ppc/kernel/mpc10x\_common.c, must be changed to support any new bridge chip.

A new definition line for each new bridge chip must be added in the first file, include/asm-ppc/mpc10x.h. As an example, the MPC8245 was added in January 2002. The code is in this 2.4.21--rc1 kernel, so there is no need to change this kernel code; it is only being used as an example. Copy the line for the MPC8245 and substitute the appropriate line for the new bridge chip and its PVR. The following is a list of the current bridge chips:

```

#define MPC10X_BRIDGE_106      ((PCI_DEVICE_ID_FREESCALE_MPC106 << 16) | \
                                PCI_VENDOR_ID_FREESCALE)

#define MPC10X_BRIDGE_8240    ((0x0003 << 16) | PCI_VENDOR_ID_FREESCALE)

#define MPC10X_BRIDGE_107     ((0x0004 << 16) | PCI_VENDOR_ID_FREESCALE)

#define MPC10X_BRIDGE_8245    ((0x0006 << 16) | PCI_VENDOR_ID_FREESCALE)
    
```

Similarly, add a case statement to the second file, arch/ppc/kernel/mpc10x\_common.c, to detect the new bridge chips. The case statements for the existing bridge chips are as follows:

```

switch (host_bridge) {

    case MPC10X_BRIDGE_106:

    case MPC10X_BRIDGE_8240:

    case MPC10X_BRIDGE_107:

    case MPC10X_BRIDGE_8245:
    
```

```

        break;
    default:
}

```

## 6 Building the New Kernel

At this point, the kernel has already been configured and built once, so the next step is to simply build the new kernel.

```
make zImage
```

Assuming there are no errors, the new kernel should be in arch/ppc/boot/images/zImage.sandpoint.

## 7 Loading and Booting the New Kernel

Using any of the techniques described in AN2578, referenced in Section 8, “References,” download this kernel image to the Sandpoint with the new processor in it and start the kernel. The three techniques are as follows:

1. Copy the zImage.sandpoint to partition 1.  

```
dd if=zImage.sandpoint of=/dev/hda1
```
2. Use the Ethernet port through DINK32 to load the zImage from the host machine to the target.  

```
d1 -nw
```
3. Use the serial port through DINK32 to load the zImage from the host machine to the target.  

```
d1
```

If it does not boot, use debugging techniques, such as printk statements or a cop interface tool.

## 8 References

*Creating a Linux ‘Out of the Box’ Experience on a Sandpoint Platform (AN2578)*

*Porting Linux to the MPC8245 (AN2222)*

## 9 Documentation Revision History

Table 9-1 provides a revision history for this application note.

**Table 9-1. Document Revision History**

| Revision Number | Substantive Change(s)   |
|-----------------|---|
| 0               | Initial release to confidential web   |
| 0.1             | Removed ‘Confidential Proprietary’ and ‘Preliminary’ footers. Released to public web. |



THIS PAGE INTENTIONALLY LEFT BLANK

## How to Reach Us:

### Home Page:

[www.freescale.com](http://www.freescale.com)

### E-mail:

[support@freescale.com](mailto:support@freescale.com)

### USA/Europe or Locations Not Listed:

Freescale Semiconductor  
 Technical Information Center, CH370  
 1300 N. Alma School Road  
 Chandler, Arizona 85224  
 +1-800-521-6274 or +1-480-768-2130  
[support@freescale.com](mailto:support@freescale.com)

### Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH  
 Technical Information Center  
 Schatzbogen 7  
 81829 Muenchen, Germany  
 +44 1296 380 456 (English)  
 +46 8 52200080 (English)  
 +49 89 92103 559 (German)  
 +33 1 69 35 48 48 (French)  
[support@freescale.com](mailto:support@freescale.com)

### Japan:

Freescale Semiconductor Japan Ltd.  
 Headquarters  
 ARCO Tower 15F  
 1-8-1, Shimo-Meguro, Meguro-ku,  
 Tokyo 153-0064  
 Japan  
 0120 191014 or +81 3 5437 9125  
[support.japan@freescale.com](mailto: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](mailto: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 303-675-2140  
 Fax: 303-675-2150  
[LDCForFreescaleSemiconductor@hibbertgroup.com](mailto: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.

