Freescale Semiconductor
Application Note

Document Number: AN3371
Rev. 0, 05/2007

Power Management in Linux® for the PowerQUICC™ MPC8313E

by *P V Suresh*
Network Computing Systems Group
Freescale Semiconductor, Inc.

This application note describes the implementation of the Linux® power management on the PowerQUICC™ MPC8313E processor. The MPC8313E has features to minimize power consumption at several levels. Dynamic power management locally minimizes power consumption when a block is idle. Many blocks in the MPC8313E can dynamically turn off clocks when sections of the block are idle. This feature is always enabled and occurs automatically. Clocks to the individual blocks can be shut down when they are not needed through a register system clock control register (SCCR). Additionally, the PowerPC™ core can be put into doze, nap, or sleep power-down states. Through the power management controller (PMC), on-chip devices can be programmed to enter low-power state when the PowerPC™ core enters nap or sleep states. The MPC8313E supports a low-power mode in which power is removed from a portion of the die for significant additional power saving. The PMC features work in concert with the PCI power management (PM) block (PME context) to provide support for PCI power management capabilities such as asserting or responding to power management events (PMEs).

Contents

freescale™ semiconductor

1 Power Management Basics

The basic features of the MPC8313E processor hardware unit for power management, called the power management controller (PMC), are as follows:

- Power management in both host and agent modes
- Supports PCI Power Management 1.2 specification.
- PME generation in PCI agent mode and PME detection in PCI host mode
- Wake-up from Ethernet (magic packet), USB, GPIO, and PCI (PME input as host)

The PCI power management specification defines five PCI device power (Dx) states (D0, D1, D2, D3hot, and D3cold). Also, it defines four bus power (Bx) states (B0, B1, B2, and B3). The Dx states are mapped into the ACPI specification, and the ACPI specification allows PCI device control at the system level. In PCI naming conventions, hot and cold refer to the availability of VCC. D3 hot and D3 cold are sub-states for D3. The OS can put a PCI device into the D3 hot state, which should require less power than D0 although there is no specific requirement for this. However, VCC and an always-on auxiliary PCI power pin (3.3 Vaux) are available as power sources. For D3 cold, VCC is removed. According to the PCI specification, the D0 and D3 hot states are mandatory states.

The MPC8313E processor can be used as a PCI host or agent. The MPC8313E supports the PCI power states D0, D1, D2, D3 hot as defined in Rev.1.2 of the PCI power management interface specification. As host, the MPC8313E responds to PME signaling as a wake-up event. As an agent, the MPC8313E can generate PME signaling by asserting an external PCI_PME signal. When the MPC8313E functions as a PCI device, the Gx, Cx, and Sx states do not apply to the MPC8313E itself but to the OS, which is running on another board controlling the MPC8313E as a device. Only the Dx and Bx states apply in this case.When the MPC8313E is used as a PCI host, the OS should comply with these global state definitions

A new state defined for the MPC8313E is called D3 warm state. The difference between D3 hot and D3 warm is that in D3 hot the entire MPC8313E core region is supplied with the nominal 1 V Vdd supply. In D3warm, a portion of the core region can be powered off. This partial power-down mode allows the MPC8313E to achieve significant reduction in power dissipation while still maintaining the capability to respond to wake-up events.

2 Changes in Kernel Source for Power Management

MPC8313E power management occurs in three different operating modes:

- *PCI host mode*. The MPC8313E acts as PCI Host and PCI is enabled in the Linux.
- *PCI agent mode*. The MPC8313E acts as an I/O processor and is a PCI device. The host processor can be another MPC8313E or any x86 machine.
- *Standalone mode*. The MPC8313E works without any PCI in standalone mode.

2.1 PMC Driver

The PMC driver handles all power management events. It registers platform-specific callback routines with the kernel PM subsystem. When the system enters deep sleep state, the PM driver routine

`mpc83xx_pm_enter` is called. This routine puts the RAM into self refresh mode, configures the wake-up event sources, and calls `mpc83xx_enter_deep_sleep` to put MPC8313E into D3 warm state.

```
static struct pm_ops mpc83xx_pm_ops = {
        ...
        .prepare = mpc83xx_pm_prepare,
        .enter = mpc83xx_pm_enter,
};
pm_set_ops(&mpc83xx_pm_ops);
```

2.2 MPC8313E IPIC Device Driver

The IPIC code has been changed to handle the suspend and resume functions. The `ipic_suspend` routine saves the contents of the ipcs registers to a data structure `ipic_saved_state` and in the resume routine restores the IPIC contents.

```
static struct {
        u32 sicfr;
        u32 siprr[2];
        u32 simsr[2];
        u32 sicnr;
        u32 smprr[2];
        u32 semsr;
        u32 secnr;
        u32 sermr;
        u32 sercr;
} ipic_saved_state;
static int ipic_suspend(struct sys_device *sdev, pm_message_t state)
{
        struct ipic *ipic = primary_ipic;

        ipic_saved_state.sicfr = ipic_read(ipic->regs, IPIC_SICFR);
        ipic_saved_state.siprr[0] = ipic_read(ipic->regs, IPIC_SIPRR_A);
         ...........
}

static int ipic_resume(struct sys_device *sdev)
{
        struct ipic *ipic = primary_ipic;
```

```
                ipic_write(ipic->regs, IPIC_SICFR, ipic_saved_state.sicfr);
                ipic_write(ipic->regs, IPIC_SIPRR_A, ipic_saved_state.siprr[0]);
                 .......
        }
        static struct sysdev_class ipic_sysclass = {
                 .....
                 .suspend = ipic_suspend,
                .resume = ipic_resume,
        };
```

2.3 MPC8313E On-Chip Device Driver Modifications

Modification of the MPC8313E on-chip device drivers is as follows:

- The MPC8313E on-chip device drivers, such as the USB and Ethernet controller are modified to configure all the necessary registers to wake up the MPC8313E from a power-down state.
- The required MPC8313E on-chip device driver suspend/resume functions are added or modified if already available. These functions save/restore the device context during the power-down and power-up states. The following code shows the code changes in the eTSEC driver:

```
/* Structure for a device driver for TSEC*/
        static struct platform_driver gfar_driver = {
                ...
                ....
        #ifdef CONFIG_PM
                .suspend = gfar_suspend,
                .resume = gfar_resume,
        #endif
                .....
};
```

2.4 MPC8313E Software Flow

The MPC8313E on-chip device drivers such as the USB driver, Ethernet driver, and MPC8313E PCI device drivers have suspend and resume capabilities. These device drivers call corresponding suspend/resume callback functions based on the event.

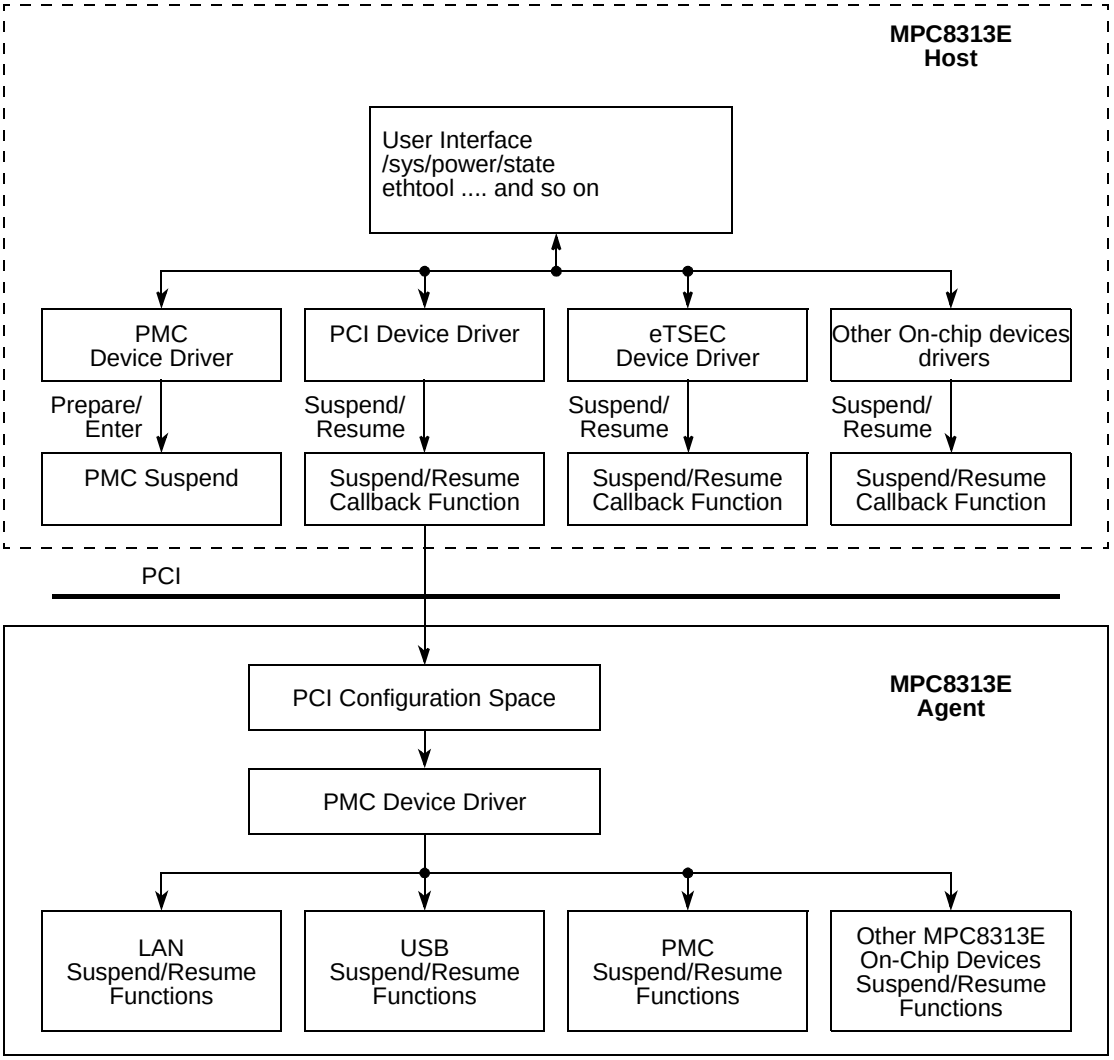

Figure 1. MPC8313E Software

2.5 Power-Down Sequence on MPC8313E as a PCI Host

Initialization of the power-down sequence proceeds as follows:

1. The host determines the power management capabilities of all agents and initializes their PME context. When triggered to power down mode, the host first brings the agent to a low-power state.
2. The PCI device driver saves the PCI function context.
3. The PCI device driver enables external PCI agents to generate PCI_PME. The driver clears PME_Status in the agent and programs the D3hot state into the PCI agent PowerState field.
4. In response to this PowerState field change, the external PCI agent transitions itself to the new power state and then updates its own PowerState field in its PCI configuration registers.
5. By looking into the configuration registers the host detects the external agents PowerState. After all the agents are in low-power state, the host can enter low-power state.

6. PCIPMR1[Power_state] is set to D3Warm.
7. The IPIC driver should set the SIMSR_L [PMC] bit to enable PMC interrupts.
8. DDR is set to self refresh state.
9. The PMC driver enables any desired wake-up sources by setting the appropriate bit in the PMCMR.
10. The PMC driver puts the e300 into deep sleep. This action causes the qseq signal to be asserted, which causes PMC to sequence in to D3warm. If the PMCCR1[POWER_OFF] bit is set, the EXT_PWR_CTRL signal transitions low, causing power to be removed to a portion of the die.

2.6 Power-Up Sequence on MPC8313E as PCI Host

The power-up sequence on the MPC8313E as a PCI host proceeds as follows:

1. A wake-up event sets the corresponding wake-up event bit in the PMCER[x] register. This event triggers an interrupt from the PMC to the IPIC controller if it is not masked. The e300 core does not service this interrupt until it wakes up from D3warm.
2. The wake-up event triggers the PMC to begin the wake-up process. It asserts EXT_PWR_CTRL to the external power supply switch, thus applying power to the VDD power rail.
3. The PMC asserts a reset to the logic blocks in the powered-off region and starts a timer. When a wake-up event occurs the reset signal continues to be asserted for the duration of the reset timer count, allowing the e300 PLL to lock. During wake-up, the reset configuration word (RCW) is not reloaded. The e300 PLL locks with previous RCW settings.
4. When the PMC reset timer expires, the pmc_reset signal is negated and the conditioning logic is removed, allowing the powered-on and powered-off regions of the die to operate normally.
5. The e300 core begins fetching instructions as it did when initially reset (POR), starting at the memory address specified by MSRP[IP]. This address is decoded by the local address window logic (eLBC) and directed to a particular boot device.
6. The e300 core initialization code checks the PMCCR1[POWER_OFF] bit to see that this is a reset from the D3warm state, not a full POR. The e300 core initializes the DDR controller so that it does not initialize the DDR when coming out of self-refresh (DDR_SDRAM_CFG[BI] = 1).
7. The initialization code restores the IPIC controller and the MSR[EE] so that the e300 core can detect pending interrupts, including those from the PMC.
8. The PMCCR1[CURR_STATE] register field is updated to reflect the new active state.

2.7 Power-Down Sequence on MPC8313E as PCI Agent

A power-down sequence on the MPC8313E as a PCI agent proceeds as follows:

1. The PMC driver programs the PMC to allow wake-up on one of the PMC wake-up events by writing a 1 in the appropriate PMCMR[] mask register bit.
2. The host PCI driver executes code to save any MPC8313E context and enables the PCI function to generate PCI_PME. It then programs the D3hot state into the PCI function PCIPMR1[Power_State] field.

3. This PCI Configuration PowerState register setting is detected and also reflected into the PMCCR1[NEXT_STATE] register. This change generates an interrupt to the e300 core through the IPIC.
4. An e300 interrupt routine detects the PMCCR1[NEXT_STATE] notification and begins the process of power down. It stops all the masters on the CSB bus,
5. The PMC driver enables any desired wake-up sources by setting the appropriate bit in the PMCMR.
6. The PMC driver sets existing PMC registers that allow power-down when the e300 core enters nap or sleep mode (PMCCR[DLPEN] and PMCCR[SLPEN]). Settings are based on the PMCCR1[NEXT_STATE] field.
7. The context of the content in the DDR is stored, and the DDR is put into self refresh mode.
8. The e300 core writes the PMCCR1[NEXT_STATE] value into PMCCR1[CURR_STATE], which is reported in the PCIPMR1[Power_State].
9. The e300 core sets the PMCCR1[POWER_OFF] bit, indicating that the external EXT_PWR_CTRL signal is to be toggled at the appropriate time to switch off external power.
10. The e300 core asserts core_qreq_b, which is detected by the PMC.
11. The PMC asserts STOP to the CSB arbiter and the DDR memory controller.
12. The PMC asserts the core_qack_b signal to the e300 core, indicating that it should enter sleep mode. The external QUIESCE signal asserts. If PMCCR1[POWER_OFF] is set, the EXT_PWR_CTRL signal is negated to disable an external power switch from shutting off VDD to a portion of the die.

2.8 Power-Up Sequence on MPC8313E as PCI Agent

The PCI host or any external wake-up event on the PCI agent can wake up the PCI agent. For the agent to assert PCI_PME, both the PCIPMR1[PME_EN] bit and the PMCCR1[PME_EN] bit must be set. The power-up sequence on the MPC8313E as a PCI agent proceeds as follows:

1. .A wake-up event sets one of the PMCER[x] bits.

   The wake-up event remains set in the PMCER[x] register until the e300 core clears it. Alternatively, the PCI host can request a change of state in the device through the PCIPMR1[Power_State] field. For example, it can configure it to a value of 0b00 (D0 mode). This state change is reflected in the PMCCR1[NEXT_STATE] field.
2. In response to the wake-up event, PME signaling is generated by asserting the PCI_PME signal to the host.
3. The PCI host recognizes the PCI_PME signal.

   If the host decides to wake up the device, it configures PCIPMR1[Power_State] in the device PCI PME context block to 0b00 (D0 mode). This change is reflected in the PMCCR1[NEXT_STATE] register bits. The PCIPMR1[Power_State] register field maintains its D3hot coding until the device transitions to D0. After the device is awake, the e300 core writes the D0 state to the PMCCR1[CURR_STATE]) field, which is then reflected into the PCIPMR1[PowerState] field to indicate to the host that the device has completed the transition to D0.
4. The PMCCR1[NEXT_STATE] transition triggers the PMC to power up the device.

5. The PMC toggles the external EXT_PWR_CTRL signal to apply voltage to the VDD supply rail. The PMC also asserts an interrupt to the IPIC. This interrupt and the source of the wake-up event are stored in PMC registers until the e300 core clears them.
6. The PMC asserts a reset to the logic blocks in the powered-off region and starts a timer. When a wake-up event occurs, the reset signal continues to be asserted for the duration of the reset timer count, thus allowing the e300 PLL to lock.

   During wake-up the reset configuration word (RCW) is not reloaded. The e300 PLL locks with previous RCW settings.
7. When the PMC reset timer expires, the pmc_reset signal is negated and the conditioning logic is removed, allowing the powered-on and powered-off regions of the die to operate normally.
8. The e300 core begins fetching instructions as it did when initially reset (POR), starting at the memory address specified by MSRP[IP]. This address is decoded by the local address window logic (eLBC) and directed to a particular boot device.
9. The e300 core initialization code checks the PMCCR1[POWER_OFF] bit to verify that this is a reset from the D3warm state, not a full POR. The e300 core initializes the DDR controller so that it does not initialize the DDR when coming out of self-refresh (DDR_SDRAM_CFG[BI] = 1).
10. The initialization code restores the IPIC controller and the MSR[EE] so that the e300 can detect pending interrupts, including those from the PMC.
11. The PMCCR1[CURR_STATE] register field is updated to reflect the new active state.

3 Build Procedure

To enable power management, Linux must be configured with ACPI support, as follows:

1. Set the environment variables CROSS_COMPILE and ARCH:

```
export CROSS_COMPILE=powerpc-e300c3-linux-
export ARCH=powerpc make menuconfig
```

2. Configure the kernel:

```
make mpc8313emds_defconfig
make menuconfig
```

3. From the kernel configuration menu:

```
Select Kernel Options.
Select Power Management Support.
```

   To enable PCI

```
Select Bus Options
```

```
Select PCI Support
```

4. Save this menu configuration and recompile the kernel:

```
make uImage
```

The new image of the Linux kernel is built with ACPI support enabled.

4 Test Procedure

To test the sleep state, put the MPC8313E into to D3warm state by writing into the `/sys/power/state` file:

```
echo mem > /sys/power/state
```

To change the state to non deep-sleep state

```
echo standby > /sys/power/state
```

To verify that the MPC8313E has entered deep sleep state, ping the MPC8313E. It should not respond to the ping requests. To test the wake-up events, first put the MPC8313E into D3warm state by writing to the `/sys/power/state` file:

```
echo mem > /sys/power/state
echo standby > /sys/power/state
ethtool -s eth0 wol g
```

To wake up the MPC8313E from the sleep state, send an Ethernet magic packet to it.

5 References

- *MPC8313E PowerQUICC II Pro Integrated Host Processor Family Reference Manual.*
- *PCI Bus Power Management Interface Specification.*

6 Revision History

Table 1 provides a revision history for this application note.

Table 1. Document Revision History

Rev. Number	Date	Substantive Change(s)
0	05/2007	Initial draft.

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

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

Document Number: AN3371
Rev. 0
05/2007