**Freescale Semiconductor**
Application Note

Document Number: AN3517
Rev. 0, 09/2007

# Configuring the MCF5445*x* Family for PCI Host Operation

Microcontroller Division Applications Team

## 1 Introduction

The ColdFire® MCF5445*x* family of processors contain a PCI controller and arbiter. PCI is supported only on the MCF54452, MCF54453, MCF54454, and MCF54455 devices.

This application note discusses the method for setting up the MCF5445*x* PCI controller and arbiter for basic host configuration and operation. It does not discuss use of the MCF5445*x* in agent configuration, though much of this setup also does apply to that configuration. It also does not discuss development of application drivers for specific PCI peripherals.

For more information on the MCF5445*x* PCI controller and arbiter, please refer to the *MCF54455 Reference Manual*.

PCI host configuration is typically used when the MCF5445*x* is the master responsible for configuring all other PCI devices in the system. The host often is both an initiator and target of PCI transactions. When acting as a

**Contents**

host, the MCF5445*x* usually does not need to generate PCI interrupts and is not the target of PCI configuration transactions. These factors affect how you programs PCI functionality and how the boot modes of the MCF5445*x* configure the device out of reset. Most of the configuration done at boot can be reprogrammed at run time if the configuration needs to be changed.

Specifically, this application note discusses the following topics:

- Boot mode effects on the PCI setup
- Pin-multiplexing use for the PCI arbiter
- Configuring PCI pads for 66- or 33-MHz operation
- Setting up the PCI controller
- Setting up the PCI arbiter

## 2 Boot Mode Effects on Configuration

The MCF5445*x* has three options for reset configuration, depending on the setting of the BOOTMOD[1:0] pins at boot time. Each boot mode can have different effects on the PCI setup. You can change many of these options at run-time by reprogramming the appropriate registers. However, two items cannot be reconfigured at run-time.

- Use of the PCI_AD bus by PCI or FlexBus must be configured at boot.
- Some of the registers which are only configurable when using serial boot mode may not be reprogrammed at run-time.

The basic configuration effects are shown in Table 1. The following sections provide more detail.

**Table 1. Boot Mode Effects on PCI**

| Setting | Boot Mode (configuration mode) | | | |
|---|---|---|---|---|
| | **Default 00** | **Parallel 10 (FB_AD3 = 0)** | **Parallel 10 (FB_AD3 = 1)** | **Serial 11** |
| PCI Enable | Enabled | FB_AD[7:5] = 111 or 011 Enabled<br>FB_AD[7:5] ≠ 111 or 011 Disabled | | SBF_RCON[125] |
| PCI REQ/GNTs | Use for PCI | FB_AD[7:5] = 111 or 011 Use for PCI<br>FB_AD[7:5] ≠ 111 or 011 Use for GPIO | | SBF_RCON[125] |
| PCI pad slew rate | 66 MHz | FB_AD[2] = 1 66 MHz<br>FB_AD[2] = 0 33 MHz | | SBF_RCON[104] |
| PCI interrupt | Disabled | Enabled | Disabled | SBF_RCON[103] |
| PCI Cfg.retry | Disabled | Enabled | Disabled | SBF_RCON[102] |
| PCI BAR[5:0] enable | All enabled (0x3F) | All disabled (0x00) | All enabled (0x3F) | SBF_RCON[101:96] |
| PCI device ID | | | | SBF_RCON[95:80] |
| PCI vendor ID | | | | SBF_RCON[79:64] |

Table 1. Boot Mode Effects on PCI (continued)

| Setting | Boot Mode (configuration mode) | | | |
|---|---|---|---|---|
| | Default 00 | Parallel 10 (FB_AD3 = 0) | Parallel 10 (FB_AD3 = 1) | Serial 11 |
| PCI class code | | | | SBF_RCON[63:40] |
| PCI revision ID | | | | SBF_RCON[39:32] |
| PCI subsystem ID | | | | SBF_RCON[31:16] |
| PCI subsystem vendor ID | | | | SBF_RCON[15:0] |

## 2.1 Default Configuration (BOOTMOD = 00)

This boot mode uses the internal settings of the chip configuration module's (CCM) reset configuration register (RCON). The MCF5445*x* is automatically configured into a default PCI host configuration. The resulting configuration is shown in Table 2.

Table 2. Default Configuration (BOOTMOD = 00)

| Functionality | Register(s) Affected | Setting | Description |
|---|---|---|---|
| PCI enable | PAR_PCI[15:0] | 0xD5D5 | PCI_AD bus used for PCI instead of FlexBus, All PCI requests/grants configured for PCI use |
| PCI pad slew rate | MSCR_PCI[0] | 1 | PCI pads configured for high slew rate (66MHz) |
| PCI interrupt | PAR_IRQ[1] | 0 | IRQ1 not configured for PCI interrupt output |
| PCI configuration retry | PCITCR2[0] | 0 | Inbound accesses allowed |
| PCI BAR enables | PCITCR2[13:8] | 0x3F | All BARs enabled |
| PCI host/agent mode indication | CCR[3] | 1 | Host mode |

## 2.2 Parallel Configuration (BOOTMOD = 10)

For parallel boot mode, the values on the FlexBus address/data signals at deassertion of system reset determine various system settings. Three of these settings affect the PCI controller.

- The 3-bit FBCONFIG (FB_AD[7:5]) field must be set to either 011 or 111 to use the external address/data bus for PCI instead of FlexBus.
- The PCI host/agent mode (FB_AD[3]) field must be set to 1 for the controller to operate in host mode.
- The PCI pad slew rate (FB_AD[2]) can be set depending on the desired frequency of operation (1 for 66MHz and 0 for 33MHz). You can change this setting at run-time by programming the PCI mode select control register (MSCR_PCI) as detailed in the “Pin Multiplexing and Control” chapter of the *MCF54455 Reference Manual*.

Table 3. Parallel Configuration (BOOTMOD = 10)

| FlexBus | Functionality | Register(s) Affected | Setting | Description |
|---|---|---|---|---|
| FB_AD[7:5] = 111 or 011 | PCI enable | PAR_PCI[15:0] | 0xD5D5 | PCI_AD bus used for PCI instead of FlexBus, All PCI request/grants configured for PCI use |
| FB_AD[7:5] ≠ 111 or 011 | | | 0x0000 | PCI_AD bus used for FlexBus, PCI request/grants not configured for PCI use |
| FB_AD[2] = 1 | PCI pad slew rate | MSCR_PCI[0] | 1 | PCI pads configured for high slew rate (66MHz) |
| FB_AD[2] = 0 | | | 0 | PCI pads configured for low slew rate (33MHz) |
| FB_AD[3] = 1 | PCI interrupt | PAR_IRQ[1] | 0 | IRQ1 not configured for PCI interrupt output |
| | PCI configuration retry | PCITCR2[0] | 0 | Inbound accesses allowed |
| | PCI BAR enables | PCITCR2[13:8] | 0x3F | All BARs enabled |
| | PCI host/agent mode indication | CCR[3] | 1 | Host mode |
| FB_AD[3] = 0 | PCI interrupt | PAR_IRQ[1] | 1 | IRQ1 configured for PCI interrupt output |
| | PCI configuration retry | PCITCR2[0] | 1 | Inbound accesses retried |
| | PCI BAR enables | PCITCR2[13:8] | 0x00 | All BARs disabled |
| | PCI host/agent mode indication | CCR[3] | 0 | Agent mode |

## 2.3 Serial Configuration (BOOTMOD = 11)

In serial boot mode, the values supplied from a serial memory determine the system settings. With this boot mode, configure the parameters individually, unlike in parallel configuration mode. Also, you can configure the reset values of more PCI parameters. The following parameters can be programmed individually instead of being assigned settings based on host/agent mode:

- PCI interrupt (SBF_RCON[103]) — Normally disabled in host mode
- PCI configuration retry (SBF_RCON[102]) — Normally disabled in host mode
- PCI BAR enables (SBF_RCON[101:96]) — Normally all enabled in host mode

Additionally, since there is no specific host/agent mode setting in serial configuration mode, the host/agent mode indicator in the CCR is determined by SBF_RCON[103].

The reset values for the following PCI registers can also be configured in serial boot mode:

- PCI device ID
- PCI vendor ID
- PCI class code
- PCI revision ID
- PCI subsystem ID
- PCI subystem vendor ID

The PCI pad slew rate (SBF_RCON[104]) and PCI enable (SBF_RCON[125]) can also be set in this mode.

**Table 4. Serial Configuration (BOOTMOD = 11)**

| SBF_RCON | Functionality | Register(s) Affected | Setting | Description |
|---|---|---|---|---|
| [125] = 1 | PCI enable | PAR_PCI[15:0] | 0xD5D5 | PCI_AD bus used for PCI instead of FlexBus and all PCI request/grants configured for PCI use |
| [125] = 0 | | | 0x0000 | PCI_AD bus used for FlexBus and PCI request/grants not configured for PCI use |
| [104] = 1 | PCI pad slew rate | MSCR_PCI[0] | 1 | PCI pads configured for high slew rate (66Mhz) |
| [104] = 0 | | | 0 | PCI pads configured for low slew rate (33Mhz) |
| [103] = 1 | PCI Interrupt | PAR_IRQ[1] | 1 | IRQ1 configured for PCI interrupt output |
| [103] = 0 | | | 0 | IRQ1 not configured for PCI interrupt output |
| [102] = 1 | PCI configuration retry | | 1 | Inbound accesses retried |
| [102] = 0 | | | 0 | Inbound accesses allowed |
| [101:96] | PCI BAR enables | PCITCR2[13:8] | 0xXX | Set each bit to enable the corresponding BAR |
| [103] = 1 | PCI host/agent mode indication | CCR[3] | 0 | Agent mode |
| [103] = 0 | | | 1 | Host mode |
| [95:80] | PCI device ID | PCIIDR[31:16] | | The affected PCI registers are assigned the corresponding SBF_RCON values. PCI software may use these registers for informational purposes. |
| [79:64] | PCI vendor ID | PCIIDR[15:0] | | |
| [63:40] | PCI class code | PCICCRIR[31:8] | | |
| [39:32] | PCI revision ID | PCICCRIR[7:0] | | |
| [31:16] | PCI subsystem ID | PCISID[31:16] | | |
| [15:0] | PCI subsystem vendor ID | PCISID[15:0] | | |

## 2.4 Pin Multiplexing

Two sets of PCI signals are multiplexed with other signals on the MCF5445*x*:

- The PCI address/data bus (PCI_AD)
- The PCI arbiter request/grant pairs ($\overline{\text{PCI\_REQ}}$[3:0], $\overline{\text{PCI\_GNT}}$[3:0])

When PCI is enabled at boot, the PCI_AD signals can only be used by the PCI controller. If PCI is not enabled at boot, then some of the PCI_AD signals may be used by the FlexBus, depending on the configuration chosen at boot. The usage of the PCI_AD signals cannot be reprogrammed at run time.

Also, when PCI is enabled at boot, the PCI request/grant pairs are automatically configured for PCI. However, usage of these signals can be changed at run time by reprogramming the PCI_PAR register. The PCI request/grant pairs can be enabled or disabled to fit the needs of the system.

Table 5 shows how these signals are configured based on whether PCI is enabled or not at boot.

**Table 5. PCI Pin Multiplexing Boot Configuration**

| Signal | PCI Enabled | PCI Disabled |
|---|---|---|
| PCI_AD | PCI_AD | FB_A[1] |
| $\overline{\text{PCI\_REQ}}$[3:0]/$\overline{\text{PCI\_GNT}}$[3:0] | PCI_REQ/PCI_GNT | GPIO |

[1] These signals are used as FlexBus address only.

## 3 Configuring the PCI Controller

You must configure a number of registers in the PCI controller for the controller to be able to generate (PCI initiator) and accept (PCI target) PCI transactions. Mostly, this involves setting up the various outgoing and incoming PCI windows and the behavior of accessing the windows. Most of the registers controlling this behavior are in the general control/status register portion of the PCI controller memory map. However, you must also configure some registers in the PCI type 0 configuration header.

The following sections describe how to configure the PCI controller for initiator and target operation. Some registers are set up in both modes and can be written in one step in program code.

### 3.1 Configuring the Controller for Initiator Operation

For the controller to generate any kind of PCI transactions (configuration, memory, or I/O), you must set up the registers in Table 6. In general, you can program these registers in any order. However, it is best to disable PCI reset last.

**Table 6. Registers to Program for Initiator Operation**

| Register Long Name | Register Short Name | Description |
|---|---|---|
| PCI initiator window *n* base/translation register | PCIIWxBTAR | Configures the internal addresses that create PCI traffic by setting up:<br>• Window base address (WBA)<br>• Size of the window via the window address mask (WAM)<br>• External addresses the accesses are translated to via the window translation address (WTA). |
| PCI initiator window configuration register | PCIIWCR | Configures the type of access created by transactions to each initiator window and enables each window. For each window you must program:<br>• Whether it is an I/O or memory window (IO/M#),<br>• What type of memory read command is generated by reads if it is a memory window (PRC)<br>• The enable (ENABLE). |
| PCI status/command register | PCISCR | Set the bus master enable bit (B) to allow the PCI controller to initiate transactions. |
| PCI configuration 1 register | PCICR1 | Set the latency timer (LTMR) to a non-zero value. |
| PCI initiator control register | PCIICR | (Optional) Set the maximum retries to 0 to use infinite retries. |
| PCI global status control register | PCIGSCR | Disable PCI reset by clearing the PR bit. |

## 3.2 Configuring the Controller for Target Operation

For the controller to accept PCI transactions (the controller only responds to memory and configuration transactions), you must set up the registers in Table 7.

**Table 7. Registers to Program for Target Operation**

| Register Long Name | Register Short Name | Description |
|---|---|---|
| PCI base address register *n* | PCIBAR*n* | Configure each base address used in PCI space the MCF5445*x* responds to. The size for each window is hardcoded. Typically, a subset of those available fit the size of memory areas accessible to PCI. |
| PCI target base address translation register *n* | PCITBATRn | Sets up the internal address (BAT) to which accesses to the corresponding PCIBARs are routed or translated and enables them (EN). |
| PCI target control 2 register | PCITCR2 | Set up additional enables (B*n*E) for PCIBARs. |
| PCI status command register | PCISCR | Set up the memory access control bit (M) to allow the controller to accept memory transactions. |
| PCI configuration 1 register | PCICR1 | Set up the cache line size (CLS). |
| PCI target control register | PCITCR | (Optional) Adjust performance features such as read prefetching. |
| PCI global status control register | PCIGSCR | Disable PCI reset by clearing the PR bit. |

## 3.3 Enabling the Controller

After the PCI controller is configured for the desired operation, you must disable PCI reset by clearing the PCIGSCR[PR] bit.

**NOTE**

Setting the PCIGSCR[PR] bit resets some of the controller registers back to their default values. So, you must reconfigure the PCI controller if this occurs. See the device reference manual for more detail.

# 4 Configuring the PCI Arbiter

The MCF5445*x* contains an internal PCI arbiter that implements a two-level least-recently-used (LRU) arbitration scheme. The MCF5445*x* allows usage of an external arbiter, so you must configure a few registers before the internal arbiter is operational.

- Enable the arbiter request and grant signals ($\overline{\text{PCI\_REQ}}$[3:0], $\overline{\text{PCI\_GNT}}$[3:0]). By default, if PCI is enabled at boot, these signals are enabled. But, they can also be used as GPIOs if not needed for PCI or other functions. Each pair that is required must be selected in the PCI pin assignment register (PAR_PCI) in the pin multiplexing and control section of the device memory map.
- Enable the internal arbiter by clearing the disable bit (DS) in the PCI arbiter control register.
- Each of the request/grant pairs may be set to one of two different priority groups.

## 5 Clocking

When using PCI on the MCF5445*x*, the PCI bus clock should be used as the clock input to the chip (EXTAL/PCI_CLK). The MCF5445*x* does not supply a PCI clock to other devices; it must be generated by an external source.

When PCI is enabled during one of the boot configuration modes, the MCF5445*x* automatically bypasses the on-chip oscillator and uses the chip input clock as the reference clock for the system PLL. The PCI controller and arbiter on the MCF5445*x* use an output from the system PLL that runs through an output divider as their version of the PCI system clock. At boot time, the value of this output divider (PCR[OUTDIV4]) defaults to a value that causes the internal PCI clock to be the same frequency as the chip input clock.

The PCI controller only supports a limited number of frequency ratios between the MCF5445*x* internal bus clock and the PCI clock. The default and parallel configuration modes automatically ensure that these ratios are met by limiting the VCO multiplication factor to specific values. However, the serial boot mode allows a wider range of multipliers. If using this mode, you must not choose a multiplication factor that violates the supported ratios or the VCO range of the PLL. In any boot mode, you are also responsible for this if you reprogram the PLL control register (PCR) at run-time.

As with any PCI design, you should minimize clock and signal skew as much as possible.

## 6 Example Code

The following example code sets up the MCF5445*x* to use the internal PCI arbiter and all PCI request/grant pins. It also configures the PCI controller to allow initiation and reception of PCI transactions. Descriptive macros are used for the register names and parameters with the actual value assigned in comments on the right. Other configurations are possible.

```
/* Set up the PCI arbiter */
MCF_PCIARB_PACR = 0                          /* 0x001f001f */
         | MCF_PCIARB_PACR_INTMPRI
         | MCF_PCIARB_PACR_INTMINTEN
         | MCF_PCIARB_PACR_EXTMPRI(0x1F)
         | MCF_PCIARB_PACR_EXTMINTEN(0x1F);

/* Set up all PCI request and grant pins for PCI usage */
MCF_GPIO_PAR_PCI = 0                         /* 0xd5d5*/
         | MCF_GPIO_PAR_PCI_GNT3_GNT3
         | MCF_GPIO_PAR_PCI_GNT2_GNT2
         | MCF_GPIO_PAR_PCI_GNT1_GNT1
         | MCF_GPIO_PAR_PCI_GNT0_GNT0
         | MCF_GPIO_PAR_PCI_REQ3_REQ3
         | MCF_GPIO_PAR_PCI_REQ2_REQ2
         | MCF_GPIO_PAR_PCI_REQ1_REQ1
         | MCF_GPIO_PAR_PCI_REQ0_REQ0;

/* Set up the PCI pad slew rate for 66Mhz or 33Mhz. This is initially configured during
boot, but can be reconfigured if needed at run time. */
MCF_GPIO_MSCR_PCI = MCF_GPIO_MSCR_PCI_PCI_HI_66MHZ;
```

```
/* Set up initiator windows: base address, size, and translation address */
/* Window 0 will decode 64MB starting at 0xa0000000 and translate to the same area in
PCI memory address space. It will be used as the PCI memory window.*/
MCF_PCI_PCIIW0BTAR = 0                        /* 0xa003a000 */
         | MCF_PCI_PCIIWBTAR_WBA(PCI_MEM_ADDRESS)
         | MCF_PCI_PCIIWBTAR_WAM(PCI_MEM_SIZE)
         | MCF_PCI_PCIIWBTAR_WTA(PCI_MEM_T_ADDRESS);

/* Window 1 will decode 16MB starting at 0xa4000000 and translate to 0xa4000000 in PCI
IO address space. It will be used as the PCI IO window. */
MCF_PCI_PCIIW1BTAR = 0                        /* 0xa400a400 */
         | MCF_PCI_PCIIWBTAR_WBA(PCI_IO_ADDRESS)
         | MCF_PCI_PCIIWBTAR_WAM(PCI_IO_SIZE)
         | MCF_PCI_PCIIWBTAR_WTA(PCI_IO_T_ADDRESS);

/* Window 2 will decode 16MB starting at 0xa5000000 an can be used as the configuration
data window. Any IO window can also be used as a configuration window if the enable
bit in the PCICAR is set. When used as a configuration window, the address in the PCICAR
is used instead of the translation value. */
MCF_PCI_PCIIW2BTAR = 0                        /* 0xa500a500 */
         | MCF_PCI_PCIIWBTAR_WBA(PCI_CFG_ADDRESS)
         | MCF_PCI_PCIIWBTAR_WAM(PCI_CFG_SIZE)
         | MCF_PCI_PCIIWBTAR_WTA(PCI_CFG_T_ADDRESS);

MCF_PCI_PCIIWCR = 0                           /* 0x01090900 */
         | MCF_PCI_PCIIWCR_WINCTRL0_MEMREAD
         | MCF_PCI_PCIIWCR_WINCTRL1_IO
         | MCF_PCI_PCIIWCR_WINCTRL2_IO;

/* Enable Initiator accesses and Target memory accesses */
MCF_PCI_PCISCR |= 0                           /* 0x00000006 */
         | MCF_PCI_PCISCR_B
         | MCF_PCI_PCISCR_M;

/* Set Max Write retries to 0 for infinite retry and disable interrupts from the block*/
MCF_PCI_PCIICR = MCF_PCI_PCIICR_MAXRETRY(0);/* 0x00000000 */

/* Set up latency timer for Initiator transactions and cache line size */
MCF_PCI_PCICR1 = 0                            /* 0x0000f804 */
         | MCF_PCI_PCICR1_LATTIMER(255)
         | MCF_PCI_PCICR1_CACHELINESIZE(4);


/* Set up Target registers. Often only one BAR register is needed and it is pointed to
DRAM. However, BAR0 can be used for special purposes like pointing to the internal
register space or SRAM.*/
MCF_PCI_PCIBAR0 = PCI_TARG0_BAR0;             /* 0xfc000000 */
MCF_PCI_PCIBAR5 = PCI_TARG0_BAR1;             /* 0x00000001 */

MCF_PCI_PCITBATR0 = IPSBAR_ADDRESS | 1;       /* 0xfc000001 */
MCF_PCI_PCITBATR5 = SDRAM_ADDRESS | 1;        /* 0x40000001 */

MCF_PCI_PCITCR2 = 0                           /* 0x00002100 */
         | MCF_PCI_PCITCR2_B0E
         | MCF_PCI_PCITCR2_B5E;

/* Enable target performance features such as prefetching and write combining */
```

```
MCF_PCI_PCITCR = 0                           /* 0x00010008 */
         | MCF_PCI_PCITCR_P
         | MCF_PCI_PCITCR_WCT(8);

/* Deassert PCI reset bit*/
MCF_PCI_PCIGSCR = 0;

/* Insert wait here to allow PCI devices to come out of reset*/
```

THIS PAGE IS INTENTIONALLY 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 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3517
Rev. 0
09/2007