Freescale Semiconductor
Application Note

Document Number: AN2661
Rev. 1, 01/2007

Software Migration from the IBM (AMCC) 440GP to the MPC8540

by *Paul Wilson*
NCSD Applications
Freescale Semiconductor, Inc.
Austin, TX

As embedded systems become more complex, software complexity is the deciding factor when a new system is designed and built. The cost of porting code can sometimes offset the benefits that come with the additional features of a new processor platform. Fortunately, software migration can be easy, especially when staying within a processor core architecture.

This document details device driver and lower-level software migration from the IBM 440GP Embedded PowerPC™ processor by Applied Micro Circuits Corporation (AMCC) to the Freescale PowerQUICC™ III product family. Both processors are PowerPC based, so software migration has few caveats, leaving architects with decisions based more on hardware functionality and processor feature sets.

Contents

1 Feature Comparison

The IBM 440GP embedded PowerPC processor includes a PowerPC core (based on a 32-bit implementation of the Book E architecture) as well as the following integrated peripherals:

- DDR SDRAM memory controller
- Fast Ethernet controllers

freescale™ semiconductor

- 32-/64-bit PCI-X bus
- UART
- I^2C
- DMA capability
- Peripheral bus

The Freescale PowerQUICC III architecture is based on the latest embedded PowerPC architecture Book E definition and uses a new high performance e500 core with 256-Kbytes of Level 2 cache. PowerQUICC III also offers two integrated 10/100/1000 three-speed Ethernet controllers (TSEC), a DDR SDRAM memory controller, a 64-bit PCI-X/PCI controller, and a RapidIO interconnect.

The MPC8540 and MPC8560 are examples of PowerQUICC integrated communication processors:

- MPC8540 integrated host processor Ethernet only options (two 10/100/1000 Mbps interfaces and one 10/100 Mbps interface) with dual UART interfaces to provide the highest level of performance and integration at lower system cost.
- MPC8560 integrated communication processor. This device differs from the MPC8540 in that it includes a dedicated RISC communications controller (referred to as the CPM) for handling communication protocols. The MPC8560 CPM supports three fast serial communications channels (FCCs) for 155 Mbps ATM, 10/100 Mps Ethernet, and HDLC up to DS3 rates. Support for 256 full-duplex, time-division-multiplexed (TDM) channels using 2 multi-channel controllers (MCCs) is provided. In addition, the MPC8560 CPM supports four serial communications controllers (SCCs), one serial peripheral interface (SPI), and one I^2C interface.

This application note discusses software migration from the IBM 440GP embedded PowerPC processor to the Freescale MPC8540 PowerQUICC III processor. Table 1 compares these devices, highlighting features common to both processors. The features of the IBM 440GP are a subset of the features on the MPC8540, enabling an easy hardware migration to the PowerQUICC III family. Also, the features common to both processors are similar from a software point of view, enabling reasonably easy migration from the IBM 440GP to the MPC8540 from both a hardware and software prospective.

Table 1. Feature Comparison of IBM 440GP versus MPC8540

Feature	IBM 440GP	MPC8540
Core frequency	400 to 500 MHz	667 to 833 MHz
Core architecture	32-bit Book E PowerPC	32-bit Book E PowerPC
L1 cache (I/D)	32-/32-Kbytes	32-/32-Kbytes
SRAM/L2 cache	8-Kbytes SRAM/No L2	256-Kbytes L2 or 256-Kbytes SRAM or 128-/128-Kbytes mix
DDR controller	32-/64-bit, DDR 266 MHz up to 2 Gbytes	32-/64-bit, DDR 333 MHz up to 4 Gbytes
User programmable memory controller	No	3
Interrupt controller	13 external/45 internal	12 external/22 internal
DMA channels	4	4

Table 1. Feature Comparison of IBM 440GP versus MPC8540 (continued)

Feature	IBM 440GP	MPC8540
RapidIO	No	8-bit parallel RapidIO
PCI-X (32-/64-bit)	PCI v2.2 33/66 MHz PCI-X 133 MHz	PCI v2.2 33/66 MHz PCI-X 133 MHz
Ethernet	10/100 (×2)	10/100 and 10/100/1 Gbit (×2)
UART	2	2
I^2C	2	1
Technology	0.25 µm	0.13 µm
Power	<4 W @ 400 MHz	~7 W @ 833 MHz

2 Software Migration Overview

High-level software such as a real-time operating system (RTOS) and any high-level applications running on top of the RTOS (for example, a web server) should not be affected by processor selection. An RTOS typically uses a board support package (BSP) to communicate with the underlying hardware, and this BSP is subject to change with new hardware. Changes to a BSP are outside of the scope of this document because these changes are not only RTOS dependent, but hardware dependent.

During a migration to a new hardware platform, device drivers are the most affected software element. A device driver communicates with anything external to the processor through mechanisms internal to the processor. This includes, but is not limited to: interrupt handling, Ethernet physical layer devices (PHYs) through an internal MAC, memory devices through the DDR-SDRAM memory controller, RS-232 transceivers through the UART interface, and the internal I^2C controller. In porting code from the IBM 440GP to the MPC8540, differences in features common to both processors are discussed:

- DDR SDRAM controller
- Interrupt controller
- General purpose timers
- DMA transfers
- Ethernet
- UART
- I^2C
- PCI-X

The byte ordering for IBM 440GP and MPC8540 is big-endian, enabling an easy software migration.

3 DDR SDRAM

The similarities between the integrated DDR-SDRAM controllers of the MPC8540 and the IBM 440GP are as follows:

- The IBM 440GP has a programmable DDR SDRAM controller that supports x8 DDR SDRAMs and can provide a 32- or 64-bit interface to SDRAM memory with optional error checking and correction (ECC). The controller supports page mode operation with bank interleaving always active and can maintain up to eight open pages. The controller supports up to four 512-Mbytes logical banks in limited configurations, providing memory up to 2 Gbytes. Global memory timings, address and bank sizes, and memory addressing modes are programmable. System power can be reduced by placing the DDR SDRAM controller in sleep and/or self-refresh mode.
- PowerQUICC III has a fully programmable DDR SDRAM controller which supports JEDEC complaint DDR-I SDRAM standard x8 or x16 memories up to 166 MHz (333-MHz data rate) with optional ECC. PowerQUICC III supports page mode operation to retain the currently active SDRAM page for pipelined burst accesses for up to 16 simultaneously open pages (4 for each memory bank). Fourteen multiplexed address signals and 2 logical bank signals provide support for device densities of 64 Mbits to 1 Gbits. Four chip select signals are provided to support up to 4 physical banks of memory each from 64 Mbytes to 1 Gbytes in size or up to 2 DIMM modules. As these 4 banks provide a theoretical maximum of 4-Gbytes of DDR main memory, the 4 Gbytes could span the entire 32-bit address space. However, since space must be reserved for boot ROM, configuration registers and other important addressable locations, the maximum DDR memory is limited to 3.5 Gbytes.

However, there are differences in the way the DDR-SDRAM controllers of the MPC8540 and the IBM 440GP are configured, as explained in the following sections.

3.1 IBM 440GP DDR-SDRAM Controller

The IBM 440GP uses 23 indirectly accessed registers to configure the SDRAM controller. These registers are accessed through the SDRAM0_CFGADDR and SDRAM0_CFGDATA registers. At system reset of the processor, software must configure and then enable the DDR SDRAM controller. The registers to configure the IBM 440GP DDR SDRAM controller are shown in the following tables. Table 2 shows the SDRAM0_CFG0 register, which enables specific features of the DDR SDRAM controller.

Table 2. IBM 440GP SDRAM0_CFG0

Bits [1]	Name	Description
0	DCEN	DDR SDRAM controller enabled/disabled
2–3	MCHK	ECC enabled/disabled
4	RDEN	Registered DIMM enabled/disabled
5	PMU	Page management unit enabled/disabled
6	DMWD	DDR SDRAM width 32- or 64-bit
8–9	UIOS	Unused I/O state
10	PDP	Page deallocation policy

1 Undocumented bits are reserved.

Table 3 shows the SDRAM0_CFG1 register, which enables the DDR SDRAM controller power management and self-refresh features.

Table 3. IBM 440GP SDRAM0_CFG1

Bits [1]	Name	Description
0	SRE	Self-refresh entry
1	PMEN	Power management enabled/disabled

1 Undocumented bits are reserved.

Table 4 shows the SDRAM0_DEVOPT register, which allows configuration of the defined DDR SDRAM device-specific options.

Table 4. IBM 440GP SDRAM0_DEVOPT

Bits [1]	Name	Description
0	DLL	DDR SDRAM device DLL enabled/disabled
1	DS	DDR SDRAM device I/O drive strength

1 Undocumented bits are reserved.

The IBM 440GP DDR SRAM controller supports four logical banks. Table 5 shows the SDRAM0_BnCR registers to configure and enable memory in each logical bank.

Table 5. IBM 440GP SDRAM0_BnCR

Bits [1]	Name	Description
0–8	SDBA	Base address
12–14	SDSZ	DDR-SDRAM size
16–18	SDAM	DDR SDRAM address mode/memory organization
31	SDBE	Memory bank enabled

1 Undocumented bits are reserved.

Table 6 shows the SDRAM0_TR0 register, which is used to configure DDR-SDRAM device-specific timing parameters.

Table 6. IBM 440GP SDRAM0_TR0

Bits [1]	Name	Description
0	SDWR	DDR SDRAM write recovery
1	SDWD	DDR SDRAM write-to-write delay
7–8	SDCL	DDR SDRAM $\overline{\text{CAS}}$ latency
12–13	SDPA	DDR SDRAM CBR precharge command to next activate command minimum
14–15	SDCP	DDR SDRAM read/write precharge command to command
16–17	SDLD	DDR SDRAM command to leadoff

Table 6. IBM 440GP SDRAM0_TR0 (continued)

Bits [1]	Name	Description
27–29	SDRA	DDR SDRAM CBR refresh command to next activate
30–31	SDRD	DDR SDRAM $\overline{RAS}$ to $\overline{CAS}$ delay

1 Undocumented bits are reserved.

The SDRAM0_TR1 register is used to control various aspects of the DDR SDRAM read data path.

3.2 MPC8540 DDR-SDRAM Controller

The MPC8540 DDR-SDRAM controller registers are directly accessed by the CPU and are shown in the following tables. Table 7 shows the chip select memory bound registers (CS*n*_BNDS), which define the address range for memory banks controlled by that chip select.

Table 7. MPC8540 CS*n*_BNDS

Bits [1]	Name	Description
0–7	SA*n*	Starting address for chip select (bank) *n*
16–23	EA*n*	End address for chip select (bank) *n*

1 Undocumented bits are reserved.

Table 8 shows the chip select configuration registers (CS*n*_CONFIG), which define the memory organization (number of row/columns).

Table 8. MPC8540 CS*n*_CONFIG

Bits [1]	Name	Description
0	CS_*n*_EN	Chip select *n* enable
8	AP_*n*_EN	Chip select *n* auto precharge enable
21–23	ROW	No. of row bits for SDRAM on chip select *n*
29–31	COL	No. of column bits for SDRAM on chip select *n*

1 Undocumented bits are reserved.

Table 9 shows the DDR SDRAM timing configuration register (TIMING_CFG1), which defines timing intervals between various SDRAM control commands.

Table 9. MPC8540 TIMING_CFG1

Bits [1]	Name	Description
1–3	PRETOACT	Precharge to activate interval (t_{rp})
5–7	ACTTOPRE	Activate to precharge interval (t_{ras})
9–11	ACTTORW	Activate to read/write interval for SDRAM (t_{rcd})
13–15	CASLAT	CAS latency from read command

Table 9. MPC8540 TIMING_CFG1 (continued)

Bits [1]	Name	Description
16–19	REFREC	Refresh recovery time (t_{rfc})
22–23	WRREC	Last data to precharge interval (t_{wr})
25–27	ACTTOACT	Activate to activate interval (t_{rrd})
30–31	WRTORD	Last write data pair to read command issue internal (t_{wtr})

1 Undocumented bits are reserved.

Table 10 shows the DDR SDRAM configuration register (DDR_SDRAM_CFG), which enables/disables features of the memory controller including: self-refresh, ECC, and dynamic power management.

Table 10. MPC8540 DDR_SDRAM_CFG

Bits [1]	Name	Description
0	MEM_EN	DDR SDRAM controller enabled/disabled
1	SREN	Self-refresh enable
2	ECC_EN	ECC enable
3	RDEN	Registered DIMM enabled/disabled
6–7	SDRAM_TYPE	Type of SDRAM device used
10	DYN_PWR	Dynamic power management enabled/disabled

1 Undocumented bits are reserved.

Table 11 shows the DDR SDRAM mode configuration (DDR_SDRAM_Mode), which defines the mode registers of the SDRAM device.

Table 11. MPC8540 DDR_SDRAM_MODE

Bits	Name	Description
0–15	ESDMODE	Extended SDRAM mode
16–31	SDMODE	SDRAM mode

Table 12 shows the DDR SDRAM interval configuration (DDR_SDRAM_INTERVAL), which configures the precharge and refresh intervals.

Table 12. MPC8540 DDR_SDRAM_INTERVAL

Bits [1]	Name	Description
6–7	REFINT	Refresh interval
18–31	BSTOPRE	Precharge interval

1 Undocumented bits are reserved.

3.3 Porting DDR SDRAM to the MPC8540

There are similarities between the DDR SDRAM controller in the IBM 440GP and MPC8540, although the exact register formats are different. The MPC8540 DDR-SDRAM controller registers are directly accessed by the CPU, but the IBM 440GP DDR SDRAM controller registers are indirectly accessed by the CPU. Key features of the DDR SDRAM controller and how registers from the IBM 440GP map to the MPC8540 are shown in Table 13.

Table 13. DDR Register Mapping IBM 440GP to MPC8540

Feature	IBM 440GP Register [1]	MPC8540 Register
Refresh enable/timer	SDRAM0_CFG1[SRE]	DDR_SDRAM_CFG[SREN] DDR_SDRAM_INTERVAL[REFINT]
CAS latency (t_{aa})	SDRAM0_TR0[SDCL]	TIMING_CFG_1[CASLAT]
Precharge to activate timing (t_{rp})	SDRAM0_TR0[SDPA]	TIMING_CFG_1[PRETOACT]
Activate to precharge (t_{ras})	SDRAM0_TR0[SDCP]	TIMING_CFG_1[ACTTOPRE]
Refresh recovery timing (t_{rfc})	SDRAM0_TR0[SDRA]	TIMING_CFG_1[REFREC]
Activate to read/write timing (t_{rcd})	SDRAM0_TR0[SDCP]	TIMING_CFG_1[ACTTORW]
Write recovery (t_{wr})	SDRAM0_TR0[SDWR]	TIMING_CFG_1[WRREC]
Power management enable	SDRAM0_CFG1[PMEN]	DDR_SDRAM_CFG[DYN_PWR]
DDR SDRAM DLL disabled	SDRAM0_DEVOPT[DLL]	LCRR[DBYP]
DDR SDRAM base address	SDRAM0_B*n*CR[SDBA]	CSn_BNDs[SA*n*]
DDR SDRAM size	SDRAM0_B*n*CR[SDSZ]	CSn_BNDs[SA*n*] and [EA*n*]
Memory organization	SDRAM0_B*n*CR[SDBA]	CS*n*_CONFIG[ROW_BIT_CS_*n*] CS*n*_CONFIG[COL_BIT_CS_*n*]
Memory bank enabled	SDRAM0_B*n*CR[SDBE]	CS*n*_CONFIG[CS_*n*_EN]

[1] *n* = 0 to 3.

The IBM 440GP uses predefined modes (SDRAM0_B*n*CR[SDRAM]) for each logical bank, which defines the supported DDR-SDRAM configuration. In combination with the size (SDRAM0_B*n*CR[SDSZ]) of the memory, this mode initializes the DDR-SDRAM memory controller address muliplexing and A10 (precharge pin) settings.

The MPC8540 uses the CS*n*_CONFIG registers for each logical bank to define the DDR-SDRAM configuration in terms of number of row and columns. These settings combined with the size (CS*n*_BNDs) of the memory initialize the DDR-SDRAM memory controller address muliplexing and A10 (precharge pin) settings.

4 Interrupt Controller

Although there are many similarities between the interrupt controllers of the IBM 440GP and the MPC8540, they are configured somewhat differently, as explained in this section.

4.1 IBM 440GP Interrupt Controller

The IBM 440GP contains two universal interrupt controllers (UIC0 and UIC1) that provide all necessary control, status, and communication between 45 internal interrupts, 13 external interrupts, and the processor core. The UICs are cascaded as shown in Figure 1.

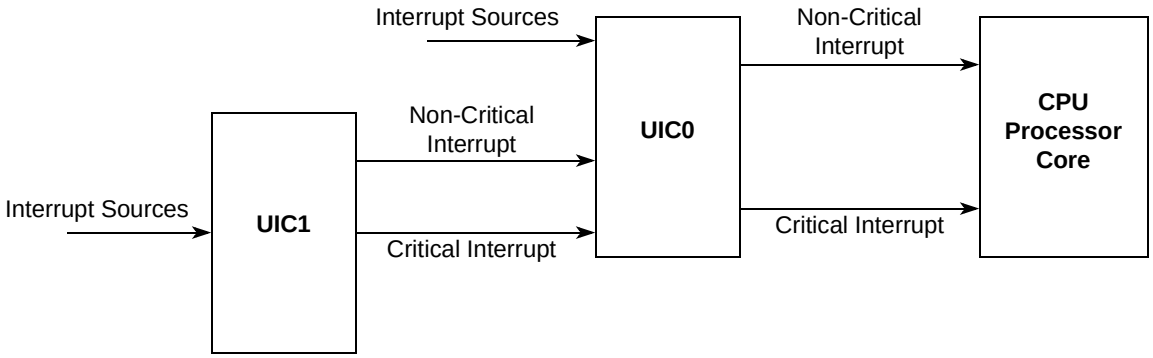

Figure 1. IBM 440GP UIC Overview

UIC0 collects interrupts from internal and external sources, including the critical and non-critical interrupt outputs of the secondary interrupt controller, UIC1. Table 14 shows the registers used to configure the IBM 440GP universal interrupt controller.

Table 14. IBM 440GP Interrupt Controller Registers

Offset	Name	Description
0x0C0	UIC0_SR	UIC status register 0
0x0D0	UIC1_SR	UIC status register 1
0x0C2	UIC0_ER	UIC enable register 0
0x0D2	UIC1_ER	UIC enable register 1
0x0C3	UIC0_CR	UIC critical register 0
0x0D3	UIC1_CR	UIC critical register 1
0x0C4	UIC0_PR	UIC polarity register 0
0x0D4	UIC1_PR	UIC polarity register 1
0x0C5	UIC0_TR	UIC trigger register 0
0x0D5	UIC1_TR	UIC trigger register 1
0x0C6	UIC0_MSR	UIC masked status register 0
0x0D6	UIC1_MSR	UIC masked status register 1
0x0C7	UIC0_VCR	UIC vector configuration register 0
0x0D7	UIC1_VCR	UIC vector configuration register 1
0x0C8	UIC0_VR	UIC vector register 0
0x0D8	UIC1_VR	UIC vector register 1

The UIC status registers (UIC0_SR and UIC1_SR) capture and hold internal and external interrupts until software resets them by writing to the interrupt (bit) location in these registers. The fields of the UIC enable registers (UIC0_E and UIC1_ER) match the UIC status registers and enable or disable interrupt reporting in the UIC status registers. Similarly, the fields of the UIC critical registers (UIC0_CR and UIC1_CR) determine whether an interrupt captured in the UIC status registers generates a non-critical or critical interrupt.

The fields of the UIC priority registers (UIC0_PR and UIC1_PR) determine whether an interrupt captured in the UIC status registers has a positive or negative priority. The fields of the UIC trigger registers (UIC0_TR and UIC1_TR) determine whether an interrupt captured in the UIC status registers are edge-sensitive or level-sensitive.

The UIC masked status registers (UIC0_MSR and UIC1_MSR) are read-only registers which contain the result of the masking of the UIC status registers (UIC0_SR and UIC1_SR) and the UIC enable registers (UIC0_E and UIC1_ER). This allows software to check which enabled interrupts are active.

The UIC vector configuration registers (UIC0_VCR and UIC1_VCR) are write-only registers which enable software control of interrupt vector generation for critical interrupts only. These registers contain an address, used as an interrupt vector base address (VBA), and a field specifying the interrupt ordering priority (PRO). The VBA field contains either the base address for an interrupt handler vector table or the base address for the interrupt handler associated with each interrupt. The actual interrupt vector (the address of the interrupt handler that services the interrupt) is generated in the UIC vector registers (UIC0_VR and UIC1_VR) using VBA. The PRO field controls whether the interrupt associated with UIC*n*_SR[0] or UIC*n*_SR[31] has the highest priority. If UIC*n*_VCR[PRO] = 0, the interrupt associated with UIC*n*_SR[31] has the highest priority; if UIC*n*_VCR[PRO] = 1, the interrupt associated with UIC*n*_SR[0] has the highest priority. The bit closest to the highest priority field that is programmed in the UIC*n*_CR as a interrupt has the second highest priority. Priority decreases across the UIC*n*_SR to the end opposite the highest priority field.

4.2 MPC8540 Interrupt Controller

The MPC8540 programmable interrupt controller (PIC) is compliant with the OpenPIC architecture and supports 12 external and 22 internal interrupt sources, 4 inter-processor interrupts, 4 global timers, and 4 message registers. The PIC can be reset through the global configuration register (GCR). In addition, this register can be used to program the PIC to operate in one of two modes:

- Pass through mode. One external active high interrupt request on $\overline{\text{IRQ0}}$ is directly passed to the e500 core. All other external requests are ignored and all internal interrupts are passed to the $\overline{\text{IRQ\_OUT}}$ signal.
- Mixed mode. External and internal interrupts are delivered according to the programmed options ($\overline{\text{IRQ\_OUT}}$ or e500 in the interrupt destination registers).

Each interrupt source (except for inter-processor interrupts) has an interrupt destination register, which determines the destination of that interrupt:

- For external sources—EIDR0–EIDR11
- For internal sources—IIDR0–IIDR31
- For messaging sources—MIDR0–MIDR3

- For timer sources—GTDR0–GTDR3

There are three interrupt destinations on a per interrupt basis:

- Core_int (e500 interrupt)
- Core_cint (e500 critical interrupt)
- $\overline{\text{IRQ\_OUT}}$ (external pin)

Each interrupt source has an interrupt vector/priority register to configure its priority (there are 16 programmable interrupt priority levels per interrupt), vector, masking, and polarity/sensing (for external interrupt sources only).

- For external sources—EIVPR0–EIVPR11
- For internal sources—IIVPR0–IIVPR31
- For messaging sources—MIVPR0–MIVPR3
- For timer sources—GTVPR0–GTVPR3
- For inter-processor sources—IPIVR0–IPIVR3

There are two critical interrupt summary registers (CISR0 and CISR1), which determine if an active interrupt is directed to the processor critical interrupt signal, Core_cint.

Software uses one processor current task priority register (CTPR) to set the current task priority executed by the core. The current priority of an interrupt should be higher than the value in the CTPR for it to be serviced. In response to an interrupt, software should read the processor interrupts acknowledge register (IACK) to obtain the vector of the interrupt being serviced. Reading IACK also has the following side effects:

- Associated field in the interrupt pending register is cleared for edge-sensitive interrupts.
- In-service register (ISR) is updated.
- Interrupt signal (int or cint) to the processor is negated.

Writing to the end of interrupt (EOI) register signals the end of processing for the highest-priority interrupt currently in service by the processor. The write to the EOI updates the ISR by retiring the highest-priority interrupt.

For nested interrupts, if the processor is servicing an interrupt, it can only be interrupted if the PIC receives an interrupt request from a source with higher priority than the one being serviced. This is true even if software, as part of its interrupt service routine, writes a new and lower value into the CTPR. Thus, although several interrupts may be in-service simultaneously, the code currently executing always handles the highest-priority interrupts that are in service. When the processor performs an EOI cycle, this highest-priority interrupt is taken out-of-service. The next EOI cycle takes the next-highest priority interrupt out-of-service, and so on. An interrupt with lower priority than those currently in-service is not started until all higher priority interrupts complete even if its priority is greater than the CTPR value.

In addition, on the MPC8560 device, there is a separate CPM interrupt controller to manage, prioritize, and route interrupts from CPM serial peripheral and parallel IO ports to the PIC.

4.3 Porting Interrupts to the MPC8540

There are similarities between the interrupt controller in the IBM 440GP and the MPC8540, although the exact register formats are different. Table 15 lists key features of the interrupt controller and how registers from the IBM 440GP map to the MPC8540.

Table 15. Mapping Features of the IBM 440GP Interrupt Controller to PowerQUICC III

Feature	IBM 440GP Register	MPC8540 Register
Enabling/masking interrupts	UIC0_ER–UIC1_ER UIC0_MSR–UIC1_MSR	Mask field of the following registers: • EIVPR0–EIVPR11 • IIVPR0–IIVPR31 • IVPR0–MIVPR3 • GTVPR0–GTVPR3 • IPIVR0–IPIVR3
Interrupt status	UIC0_SR–UIC1_SR	Activity field of the following registers: • EIVPR0–EIVPR11 • IIVPR0–IIVPR31 • IVPR0–MIVPR3 • GTVPR0–GTVPR3 • IPIVR0–IPIVR3
Critical interrupts	UIC0_CR–UIC1_CR	CISR0 and CISR1
Interrupt polarity	UIC0_PR–UIC1_PR	Polarity field of the following registers: • EIVPR0–EIVPR11 • IIVPR0–IIVPR31 • IVPR0–MIVPR3 • GTVPR0–GTVPR3 • IPIVR0–IPIVR3
Edge or level sensitive	UIC0_TR–UIC1_TR	Sense field of the following registers for external interrupt sources only: • EIVPR0–EIVPR11
Vector/prioritization	UIC0_VR–UIC1_VR UIC0_VCR–UIC1_VCR	Priority and vector fields of the following registers: • EIVPR0–EIVPR11 • IIVPR0–IIVPR31 • IVPR0–MIVPR3 • GTVPR0–GTVPR3 • IPIVR0–IPIVR3

The IBM 440GP defines separate registers for different features of the interrupt controller. The MPC8540 defines registers for each interrupt source, which determines polarity, vector/prioritization, sensing, interrupt status, and if the interrupt source is enabled. The IBM 440GP, for critical interrupts, only has fixed priority levels defined by the priority field in the UIC vector configuration registers. The MPC8540 has a more flexible interrupt controller that defines an interrupt vector/priority register for each interrupt source. This controller can be used to configure up to 16 levels of priority.

5 General-Purpose Timers

Timer facilities (including time base, decrementer, fixed internal timer (FIT), and watchdog) between the IBM 440GP and MPC8540 CPU Book E implementation are identical and are not discussed in this section. However, the configuration of the general-purpose timers differs.

5.1 IBM 440GP General Purpose Timers

The IBM 440GP supports a general-purpose timer (GPT) with five maskable compare registers and a 32-bit time base counter. Each compare register has a corresponding GPT interrupt to the universal interrupt controller (UIC). GPT interrupts can be generated for a specific count by a match between the contents of a compare register and the time base counter. GPT interrupts can also be generated on a specific interval by masking individual bits of a compare register.

5.2 MPC8540 General Purpose Timers

The MPC8540 supports four global general purpose timers that are programmed in the programmable interrupt controller. There are appropriate clock prescalers and synchronizers to provide a time base for the four internal timers of the PIC unit. The timers can be individually programmed to generate a processor interrupt when they count down to zero and can be used to generate regular periodic interrupts. Each timer has four configuration and control registers:

- Global timer current count register (GTCCR*n*)
- Global timer base count register (GTBCR*n*)
- Global timer vector-priority register (GTVPR*n*)
- Global timer destination register (GTDR*n*)

The timer frequency for all four timers is determined by the timer frequency reporting register (TFRR), timer interrupts are all edge-triggered interrupts. If a timer period expires while a previous interrupt from the same source is pending or in-service, the subsequent interrupt is lost. Using the timer control register (TCR), you can create timers larger than the 31-bit global timers. Two 16-bit timers can be internally cascaded to form a 32-bit counter. Timer 1 can be internally cascaded to timer 2, and timer 3 can be internally cascaded to timer 4. The timer global configuration registers (TGCRs) are used to put the timers into cascaded mode. You can change the timer frequency by setting the appropriate fields of the TCR.

5.3 Porting General-Purpose Timers to the MPC8540

There are similarities in the general-purpose timers of the IBM 440GP and the MPC8540, although the exact register formats are different. Table 16 illustrates how general-purpose timer registers from the IBM 440GP map to the MPC8540.

Table 16. Mapping Features of the IBM 440GP to the MPC8540

Feature	IBM 440GP Register	MPC8540 Register
Number of timers	5	4
Time base counter	GP0_TBC	GTBCR0–3

Table 16. Mapping Features of the IBM 440GP to the MPC8540 (continued)

Feature	IBM 440GP Register	MPC8540 Register
Compare timers	GP0_COMP0–4 GP0_MASK0–4	GTCCR0–3
Interrupt enable/status information	GPT0_ISS, GPT0_ISC, and GPT0_IE	GTVPR0–3
Polarity (for external I/O trigger)	DMA0_POL	Not programmable.
Status information	DMA0_SR	DGSR

6 DMA Transfers

This section discusses how the DMA controller transfers data on the IBM 440GP and the MPC8540.

6.1 IBM 440GP DMA Controller

The IBM 440GP DMA controller provides four DMA channels, each with its own control, count and control, source address, destination address, and scatter/gather address registers (see Table 17).

Table 17. IBM 440GP DMA Controller Registers

Offset	Mnemonic	Register	Description
0x100	DMA0_CR0	DMA channel control register 0	DMA0_CR0–DMA0_CR3 configure and enable their respective DMA channels. Configuration settings include transfer modes, width, and channel priority. Before a DMA channel can transfer data, the channel control, count and control, source address, and destination address registers must be programmed.
0x101	DMA0_CTC0	DMA count and control register 0	DMA0_CT0–DMA0_CT3 contain the number of transfers remaining in the DMA transaction for their respective channels when EOT*n*[TC*n*] is programmed as a terminal count output, along with additional channel control information. In addition, these registers contain fields used to enable interrupts for terminal count, end of transfer, and error conditions. Other fields enable parity checking during the peripheral portion of peripheral type.
0x102	DMA0_SAH0	DMA source address high register 0	DMA0_SAH*n* and DMA0_SAL*n* contain the source address for memory-to-memory and memory-to-peripheral transfers. DMA destination address registers (DMA0_DAH0–DMA0_DAH3 and DMA0_DAL0–DMA0_DAL3) contain the destination address for memory-to-memory and peripheral-to-memory transfers.
0x103	DMA0_SAL0	DMA source address low register 0	
0x104	DMA0_DAH0	DMA description address high register 0	
0x105	DMA0_DAL0	DMA description address low register 0	

Table 17. IBM 440GP DMA Controller Registers

Offset	Mnemonic	Register	Description
0x106	DMA0_SGH0	DMA scatter/gather address high register 0	Contain the memory address of the next scatter/gather descriptor table. Prior to starting a scatter/gather transfer, software must write the address of the channel's descriptor table to DMA0_SGH*n* and DMA0_SGL*n*. Once the scatter/gather transfer starts, DMA0_SGH*n* and DMA0_SGL*n* are automatically updated from the descriptor table.
0x107	DMA0_SGL0	DMA scatter/gather address low register 0	
0x108–0x11F	—	As above for DMA channels 1–3	
0x120	DMA0_SR	DMA status register	Provides status information for each of the DMA channels. The sleep mode register (DMA0_SLP) enables the DMA controller to enter sleep (low-power) mode after a programmed number of idle cycles.
0x123	DMA0_SGC	DMA scatter/gather command register	Sets up a DMA channel for scatter/gather transfers (DMA0_SGC[SSG*n*] = 1). Determines whether start scatter/gather transfers are enabled for DMA channels 0 to 3.
0x125	DMA0_SLP	DMA sleep mode register	
0x125	DMA0_POL	DMA polarity configuration register	Sets the polarity (active state) of the external DMAI/O signals: DMAReq*n*, DMAAck*n*, and EOT*n*[TC*n*].

When you program these DMA registers, the DMA controller performs the requested data transfer between memory and peripherals or memory-to-memory without the need for host intervention. For external peripheral transfers, three external signals are supported for each channel, as follows:

- DMAReq*n*—DMA request used to request data transfer
- DMAAck*n*—DMA acknowledge used to instruct peripheral to start DMA transfer
- EOT*n*[TC*n*]—End of transfer or terminal count signal used to stop channel.

With a normal DMA transfer, it is necessary to program a channel's control, count and control, source, and destination registers for each transfer. The scatter/gather capability of the DMA controller provides a more efficient solution for applications that require multiple transactions on a single DMA channel. Instead of individually programming a channel's registers, software creates a set (linked list) of descriptor tables in system memory. To configure a channel for a scatter/gather transfer, the DMA scatter/gather descriptor address registers (DMA0_SGH*n* and DMA0_SGL*n*) for the channel are set to the address of the first descriptor table, which must be quadword (16-byte) aligned. The format of these descriptor tables is shown in Figure 2.

	Byte 1	BYTE 2	Byte 3	Byte 4
Word 1	DMA Channel Control Word (DMA_CR*n*)			
Word 2	Configuration Bits [1]	Count (DMA_CTC*n*)		
Word 3	Source Address High (DMA_SAH*n*)			
Word 4	Source Address Low (DMA_SAL*n*)			
Word 5	Destination Address High (DMA_DAH*n*)			
Word 6	Destination Address Low (DMA_DAL*n*)			
Word 7	Linked Next Scatter/Gather Descriptor Address High (DMA_SGH*n*)			
Word 8	Linked Next Scatter/Gather Descriptor Address Low (DMA_SGL*n*)			

[1] The MSB of word 2 is the link (LK).

Figure 2. Descriptor Table Format

To begin a scatter/gather transfer, software writes DMA0_SGC register as follows:

- Enable mask—DMA0_SGC[EM*n*] = 1
- Start scatter/gather bit DMA0_SGC[SSG*n*] = 1
- Finally, indicate the bus location of the descriptor tables DMA0_SGC[SGL*n*] = X.

The DMA controller then reads the descriptor table at address (DMA0_SGH*n* || DMA0_SGL*n*) and updates the DMA controller registers. On receiving the data from the scatter/gather descriptor table, the channel's terminal count status bit (DMA0_SR[TC*n*]) and end of transfer status bit (DMA0_SR[EOT*n*]) are automatically cleared.

After the channel registers are loaded from the descriptor table, DMA transfer functions as a normal non-scatter/gather operation. If the LK (link) bit was not set, the scatter/gather process stops when the current transfer completes. Otherwise, the DMA controller reads the descriptor table at address (DMA0_SGH*n* || DMA0_SGL*n*) and the process repeats.

6.2 MPC8540 DMA Controller

The MPC8540 supports four high-speed DMA channels for transferring blocks of data between the RapidIO controller, PCI, and the local bus address space, independent of the e500 core or external hosts. Data transfers can be triggered and controlled by software or triggered and controlled by an external device using handshake signals $\overline{\text{DMA\_DREQ}}$, $\overline{\text{DMA\_DACK}}$, and $\overline{\text{DMA\_DDONE}}$. In addition, misaligned transfers and DMA sub-block transfers up to 256 bytes are supported on a per channel basis, to minimize the number of discrete memory transactions and to maximize performance over interfaces, such as RapidIO. A summary of the DMA controller registers is shown in Table 18.

Table 18. MPC8540 DMA Register Summary

Name [1]	Description
MR*n*	DMA *n* mode register
SR*n*	DMA *n* Status register
CLNDAR*n*	DMA *n* current link descriptor address register
SATR*n*	DMA *n* source attributes register
SAR*n*	DMA *n* source address register

Table 18. MPC8540 DMA Register Summary (continued)

Name [1]	Description
DATR*n*	DMA *n* destination attributes register
DAR*n*	DMA *n* destination address register
BCR*n*	DMA *n* byte count register
NLNDAR*n*	DMA *n* next link descriptor address register
CLSDAR*n*	DMA *n* current list alternate base descriptor address register
NLSDAR*n*	DMA *n* next list alternate base descriptor address register
SSR*n*	DMA *n* source stride register
DSR*n*	DMA *n* destination stride register
DGSR	DMA general status register

1 *n* = 0 to 3 for all four DMA channels.

There are two main modes of operation, direct mode and chaining mode. In direct mode, software initializes all parameters of the transfer in the appropriate register as follows:

- Mode register (MR*n*), including MR*n*[CTM] = 1
  - If stride mode is desired, set MR*n*[XFE]
  - If external I/O starts the transfer, set MR*n*[EMS_EN] = 1
  - If software (single-register write) starts the transfer, set MR*n*[SRW] = 1
  - Program other DMA options in MR*n*
- Program the stride registers (SSR*n* and DSR*n*), if enabled. Stride capability allows a DMA channel to transfer data from selected memory regions of a memory segment by skippng over certain regions of that segment.
- Program the number of bytes to transfer in the byte count register (BCR*n*), up to 64 Mbytes.
- Program DMA source attributes in the 32-bit source attributes register (SATR*n*).
- Program DMA source address in the 32-bit source address register (SAR*n*).
- Program DMA destination attributes in the 32-bit destination attributes register (DATR*n*).
- Program DMA destination address in the 32-bit destination address register (DAR*n*).

In chaining mode, software does not program source/destination parameters via registers. Instead, software provides the parameters of the transfer via link descriptor tables in memory (one chain).

- Program mode register (MR*n*)
- Current link descriptor address register (CLNDAR*n*) is used to point to the first descriptor.
- When DMA is started, the DMA reads parameters of the transfer including next descriptor address loaded in to the next link descriptor address register (NLNDAR*n*).
- When the current DMA segment is transferred, if NLNDAR*n*[EOLND] is set, the DMA is complete. If not set, the next link descriptor address from NLNDAR*n* is read in to the current link descriptor address register (CLNDAR*n*) and the process is repeated.

The link descriptor format for basic chaining mode is shown in Figure 3.

The extended chaining mode allows the user to setup a series buffer descriptors all referenced from a linked list. Through this linked list, the DMA controller can then walk through multiple BDs allowing complex DMA transactions to be performed.

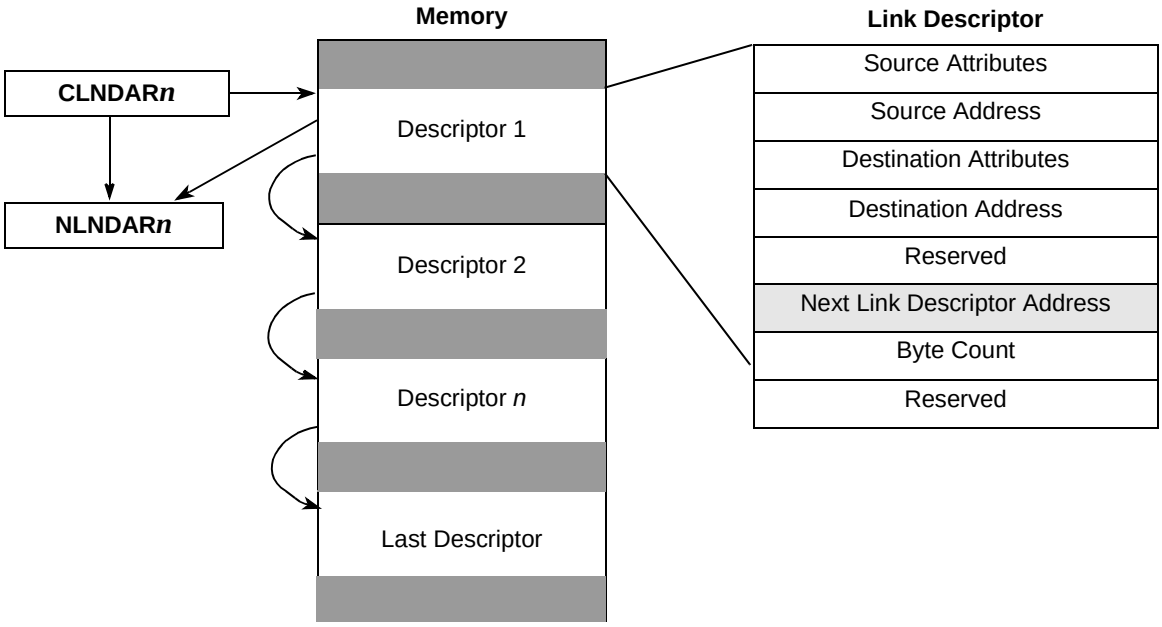

Figure 3. MPC8540 Basic Chaining Mode—Link Descriptor Format

6.3 Porting DMA Transfers to the MPC8540

Similarities exist between the DMA controller in the IBM 440GP and MPC8540, although the exact register formats are different. Key features of the DMA controllers and how registers from the IBM 440GP map to MPC8540 are shown in Table 19.

Table 19. Mapping Features of the IBM 440GP DMA Controller to the MPC8540

Feature	IBM 440GP Register	MPC8540 Register
Enabling DMA channels and mode of operation	DMA0_CR*n*	MR*n*
Transfer size	DMA0_CR*n*[PW] × DMA0_CT*n*[TC]	BCR*n*
Source address/attributes	DMA0_SAH*n* and DMA0_SAL*n*	SAR*n* and SATR*n*
Destination address/attributes	DMA0_DAH*n* and DMA0_DAL*n*	DAR*n* and DATR*n*
DMA chaining	DMA0_SCH*n*, DMA0_SGL*n*, DMA_SCR*n*, and link descriptor tables in external memory	MR*n*[CTM] = 0 CLNDAR*n*, NLNDAR*n*, and link descriptor tables in external memory
Sleep mode	DMA0_SLP	Supported globally on PQ3 but not DMA specific
Polarity (for external I/O trigger)	DMA0_POL	Not programmable
Status information	DMA0_SR	DGSR

The MPC8540 features are identical to those of the IBM 440GP for DMA transfers. In addition, the DMA controller on the MPC8540 is more flexible and offers support for extended chaining mode and DMA stride transfers.

7 Buffer Descriptors

Buffer descriptors (BDs) are the primary data structures used on the IBM 440GP and MPC8540 for passing data between higher level software and on-chip serial communication peripherals. Organization of BDs differs between the IBM 440GP and the MPC8540, but the basic structure is similar, easing migration to the MPC8540. For example, the differences in BDs for Ethernet operation on the IBM 440GP and MPC8540 will be examined.

7.1 IBM 440GP Buffer Descriptors

On the IBM 440GP, a hardware block referred to as the memory access layer (MAL) manages data transfers between two Ethernet controllers (EMACs). To communicate with software device drivers, MAL utilizes a buffer descriptor ring structure in external memory. A software device driver uses the buffer descriptor structure to inform MAL about buffer locations and packet or buffer status. The descriptors are pointers to the actual buffers organized as circular queues. Each EMAC is associated with two tables (transmit and receive) of up to 256 buffer descriptors per table. The format of the buffer descriptor (BD) shown in Figure 4, is the same for both transmit and receive.

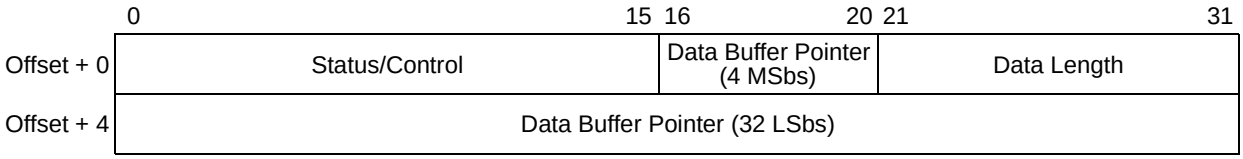

Figure 4. IBM 440GP Buffer Descriptor Format

The most significant half word in each buffer descriptor contains the status/control bits. The second half word determines the 4 MSbs of the data buffer pointer and the data length (in bytes) referenced in this buffer descriptor. The second word in the buffer descriptor contains the 32 LSbs of the data buffer pointer that points to the actual data buffer in memory. An example of the BD and data buffer structure in external memory is shown in Figure 5.

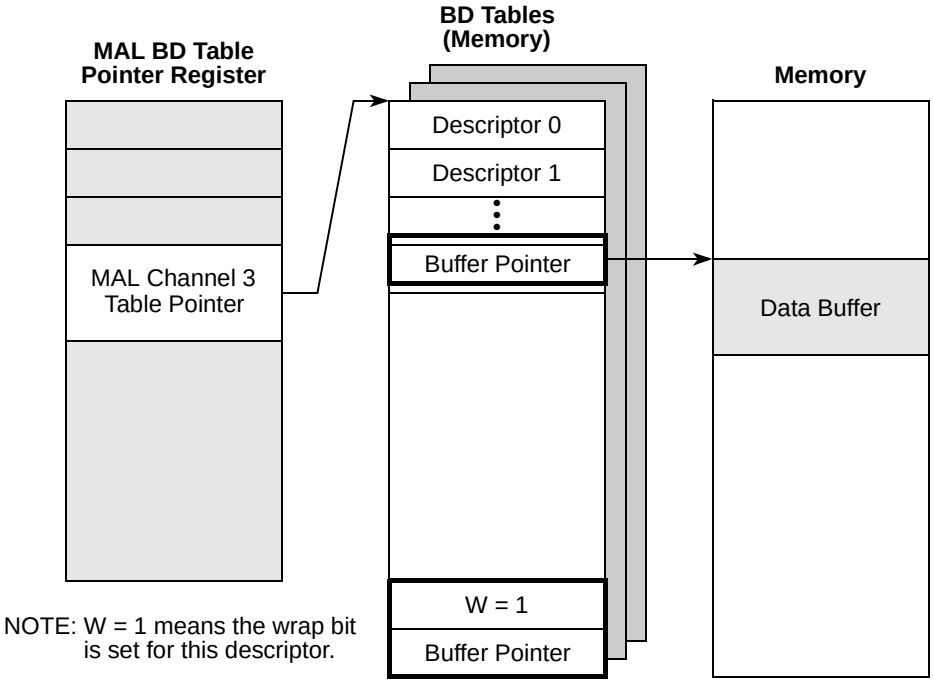

Figure 5. IBM 440GP BDs and Data Buffers in External Memory

Table 20 shows the format of the status/control bits for transmit buffer descriptors. Bits 6–15 are used for EMAC specific control information during a write access and status information during a read access.

Table 20. IBM 440GP Status/Control Bits of TxBDs

Bit	Name	Description
0	R	Ready—indicates Tx buffer is ready for transmission
1	W	Wrap—last buffer in the circular BD table
2	CM	Continuous mode—indicates continuous transmission of buffer regardless of the R bit
3	L	Indicates last buffer in frame
4	—	Reserved
5	I	Specifies an Interrupt to be generated on processing of BD
6	FCS	Generate FCS (control)
		Indicates BAD FCS (status)
7	PAD	Generate padding (control)
	BPACK	Indicates BAD packet (status)
8	ISA	Insert source address (control)
	LOCS	Loss of carrier sense (status)
9	RSA	Replace source address (control)
	ED	Excessive deferral (status)

Table 20. IBM 440GP Status/Control Bits of TxBDs (continued)

Bit	Name	Description
10	IVTg	Insert VLAN tag (control)
	EC	Excessive collisions (status)
11	RVTg	Replace VLAN tag (control)
	LC	Late collisions (status)
12	MC	Multiple collisions (status)
13	SC	Single collision (status)
14	UR	Underrun (status)
15	SQE	Signal quality error (status)

Table 21 shows the format of the status/control bits for receive buffer descriptors. Bits 6–15 are used for EMAC-specific status information.

Table 21. IBM 440GP Status/Control Bits of RxBDs

Bit	Name	Description
0	E	Empty—indicates the receive data buffer is ready to receive data
1	W	Wrap—last buffer in the circular BD table
2	CM	Continuous mode—indicates continuous reception of data regardless of the E bit
3	L	Indicates last buffer in frame
4	F	Indicates first buffer in frame
5	I	Specifies an interrupt to be generated on processing of BD
6	OR	Indicates overrun error indication
7	PP	Pause packet indication
8	BP	Bad packet—indicates early termination caused by packet error
9	RP	Runt packet error indication
10	SE	Short event error indication
11	AE	Alignment error indication
12	BF	Bad FCS error indication
13	PTL	Packet too long error indication
14	ORE	Out of range error indication
15	IRE	In range error indication

7.2 MPC8540 Buffer Descriptors

The three-speed ethernet controllers (TSECs) on the MPC8540 also use buffer descriptors and data buffers in the transmission and reception of Ethernet frames.

The format of the buffer descriptor (BD) shown in Figure 6 is the same for both transmit and receive:

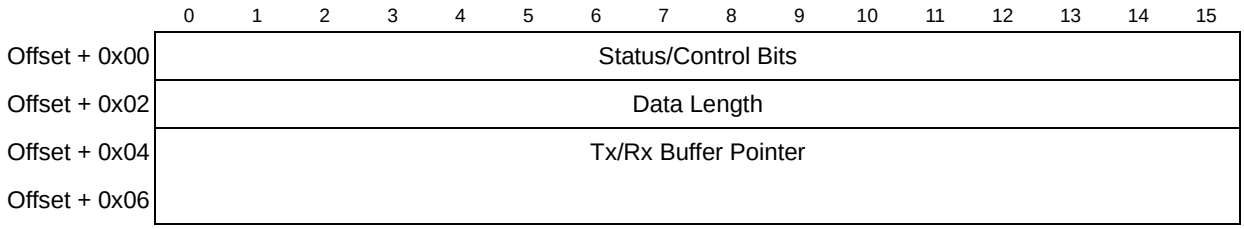

Figure 6. MPC8540 Buffer Descriptor Format

The first word contains the status/control bits and the buffer data length (in bytes) to be transmitted or received. The second word in the buffer descriptor contains the data buffer pointer that points to the actual data buffer in memory. An example of the BD and data buffer structure in external memory is shown in Figure 7.

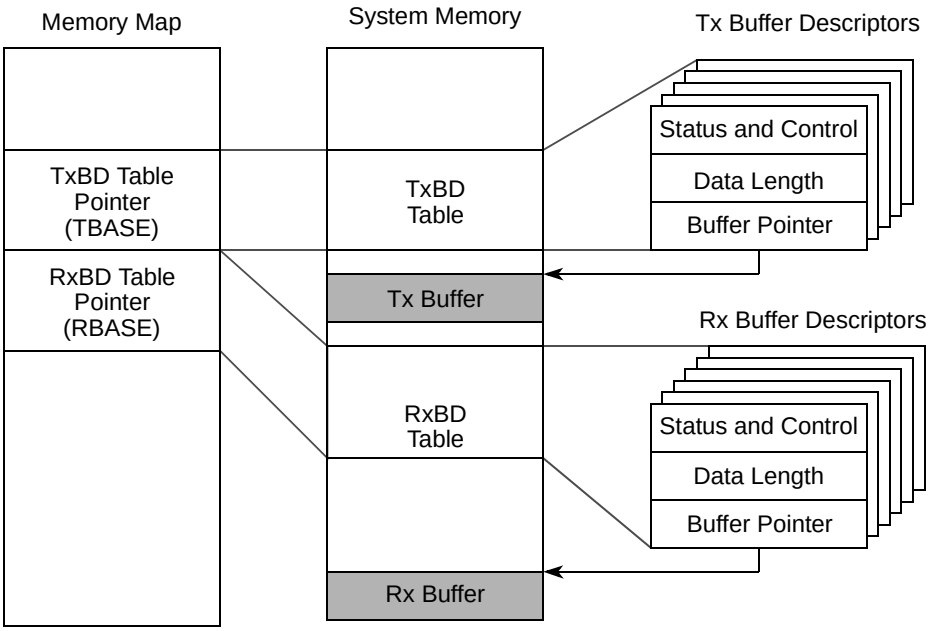

Figure 7. Example of MPC8540 TSEC Memory Structures for BDs

Table 22 shows the format of the status/control bits for the MPC8540 TSEC transmit buffer descriptors.

Table 22. MPC8540 TSEC Status/Control Bits of Tx BDs

Bit	Name	Description
0	R	Ready—indicates TX buffer is ready for transmission
1	PAD	Padding for short <64-byte Ethernet frames
2	W	Wrap—last buffer in the circular BD table
3	I	Specifies an interrupt to be generated on processing of BD
4	L	Indicates last buffer in frame
5	TC	Transmit CRC

Table 22. MPC8540 TSEC Status/Control Bits of Tx BDs (continued)

Bit	Name	Description
6	DEF	Defer
7	—	Reserved
8	HFE	Hugh frame enable (control)
	LC	Late collision indication
9	RL	Retry limit reached indication
10–13	RC	Retry count indicates number of retries
14	UN	Underrun indication
15	—	Reserved

Table 23 shows the format of the status/control fields for the MPC8540 TSEC receive buffer descriptors.

Table 23. MPC8540 TSEC Status/Control Fields of RxBDs

Bit	Name	Description
0	E	Empty—indicates the receive data buffer is ready to receive data
1	—	Reserved
2	W	Wrap—last buffer in the circular BD table
3	I	Specifies an interrupt to be generated on processing of BD
4	L	Indicates last buffer in frame
5	F	Indicates first buffer in frame
6	—	Reserved
7	M	Miss—frame accepted because of promiscuous but no match address
8	BC	Broadcast—frame had a broadcast address
9	MC	Multicast—frame had a multicast address
10	LG	Large—indicates frame bigger than maximum frame length received
11	NO	Non-octet—frame not divisible by 8 received
12	SH	Short—indicates short frame received if enabled
13	CRC	CRC—frame had a CRC error
14	OV	Overrun indication
15	TR	Truncation—indicates frame was truncated

7.3 Porting Buffer Descriptors to MPC8540

Since both the MPC8540 and the IBM 440GP rely on circular queues of buffer descriptors, porting software applications from the IBM 440GP to the MPC8540 should be relatively straightforward. Buffer descriptors themselves have a similar format, although the exact contents do differ.

8 Ethernet

Table 24 compares the key Ethernet features of IBM 440GP and MPC8540. Many similarities exist between the Ethernet access controllers (EMACs) on the IBM 440GP and the three-speed ethernet controllers (TSECs) of the MPC8540. In the following sections we discuss only Ethernet features common to both processors and the configuration differences.

Table 24. Ethernet Comparison of IBM 440GP versus MPC8540

Ethernet Feature	IBM 440GP	MPC8540
Standards	IEEE 802.3, 802.3u, and 802.3x,	IEEE 802.3, 802.3u, 802.3x, 802.3z, and 802.3ab
Speeds	10/100 Mbps	10/100/1000 Mbps
Interfaces	MII, RMII, and SMII	GMII, RGMII, MII, RMII, TBI, and RTBI
Half-duplex support	10/100 Mbps	10/100 Mbps
Full-duplex support	10/100 Mbps	10/100/1000 Mbps
Automatic retransmission of packets on collision	Yes	Yes
FCS (CRC) checking and generation	Yes	Yes
Programmable inter-packet gap	Yes	Yes
Flow control (including pause packet checking and generation)	Yes	Yes
Address recognition	Unicast, multicast, broadcast, and promiscuous mode	Unicast, multicast, broadcast, and promiscuous mode
Jumbo frame support	No	Programmable option
Automatic source address insertion or replacement	Programmable option	No
VLAN support	VLAN ID tag insertion/replacement in transmit packets	None
Wake on LAN (WOL) handling	Yes	No

8.1 IBM 440GP Ethernet Controllers

The IBM 440GP supports two Ethernet access controllers (EMACs) for half-/full-duplex operation at 10/100 Mbps. Each EMAC uses a media independent interface (MII) controlled by a ZMII bridge to connect to standard Ethernet physical devices (PHYs). The ZMII bridge supports one MII interface, or two RMII/SMII interfaces selected by enabling bits 28–29 of the power-on configuration register 0 (CPC0_STRP0). In addition, there are three registers which control and maintain status information on the ZMII bridge. A register summary for each EMAC is shown in Table 25.

Table 25. IBM 440GP Ethernet MAC Register Summary

Mnemonic	Register	Description
EMAC*n*_MR0[1]	Mode register 0	Define the operating mode including: Tx/Rx enable, full-duplex operation, flow control, 10/100 Mbps operation, FIFO size, loopback mode, and VLAN enable.
EMAC*n*_MR1	Mode register 1	

Table 25. IBM 440GP Ethernet MAC Register Summary

Mnemonic	Register	Description
EMAC*n*_TR0	Transmit mode register 0	
EMAC*n*_TR1	Transmit mode register 1	
EMAC*n*_RMR	Receive mode register	Defines the EMAC mode during receive operation including: address recognition modes, strip padding/FCS enable/disable, propagate pause packets, and reception of oversized packets.
EMAC*n*_ISR	Interrupt status register	The EMAC generates interrupt events using the contents of the interrupt status register (EMAC*n*_ISR) and the corresponding mask bits in the interrupt status enable register (EMAC*n*_ISER).
EMAC*n*_ISER	Interrupt status enable register	
EMAC*n*_IAHR	Individual address high	EMAC*n*_IAHR contains the high-order half word of the station unique individual address. During packet reception, if EMAC is programmed in individual address match mode (EMAC*n*_RMR[IAE] = 1), the contents of EMAC*n*_IAHR are concatenated with the contents of individual address low register (EMAC*n*_IALR) to form a composite address that is compared with the destination address of the received packet. The individual address hash tables 1–4 (EMAC*n*_IAHT1–EMAC*n*_IAHT4) are used in the hash table function for multiple individual addressing mode. The group address hash tables 1–4 (EMAC*n*_GAHT1–EMAC*n*_GAHT4) are used in the hash table function for multiple group addressing mode.
EMAC*n*_IALR	Individual address low	
EMAC*n*_VTPID	VLAN TPID (tag protocol identifier) register	
EMAC*n*_VTCI	VLAN TCI (tag control information) register	
EMAC*n*_PTR	Pause timer register	Defines the time period for which a pause function is enabled. The inter-packet gap value register (EMAC*n*_IPGVR) defines the value of one-third of the inter-packet gap for the next packet to be transmitted.
EMAC*n*_IAHT1–4	Individual address hash tables 1–4	
EMAC*n*_GAHT1–4	Group address hash tables 1–4	
EMAC*n*_LSAH	Last source address high	
EMAC*n*_LSAL	Last source address low	
EMAC*n*_IPGVR	Inter-packet gap value register	
EMAC*n*_STACR	STA control register	
EMAC*n*_TRTR	Transmit request threshold register	
EMAC*n*_RWMR	Receive low/high water mark register	Defines the conditions that cause the EMAC to activate a low or urgent priority MAL request, and that manage flow control
EMAC*n*_OCTX	Transmitted octets	Read-only registers which contain the number of octets transmitted or received. These registers can be used by software for RMON statistics.
EMAC*n*_OCRX	Received octets	
Note: *n* = 1 or 0		

8.2 MPC8540 Ethernet Controllers

The MPC8540 supports two three-speed Ethernet controllers (TSECs). Each TSEC supports 10/100/1000 Mbps Ethernet/802.3 networks using standard MAC-PHY interfaces as follows to connect to an external Ethernet transceiver:

- MII interface running at 10/100 Mbps
- GMII interface running at 1 Gbps
- TBI interface that can be connected to a SerDes device for fibre channel applications
- Reduced signal count versions of the GMII (RGMII) and ten-bit (RTBI) interfaces

The TSEC interface (GMII or TBI) and width (reduced or standard) are initialized by sampling certain signals during the power-on reset sequence. The value of all these signals are sampled into the PORDEVSR (POR device status register) while $\overline{\text{HRESET}}$ is asserted. In addition, the Ethernet control register (ECNTRL) determines the interface type and width, and enabled RMON statistics. A block diagram of the TSEC controller is shown in Figure 8.

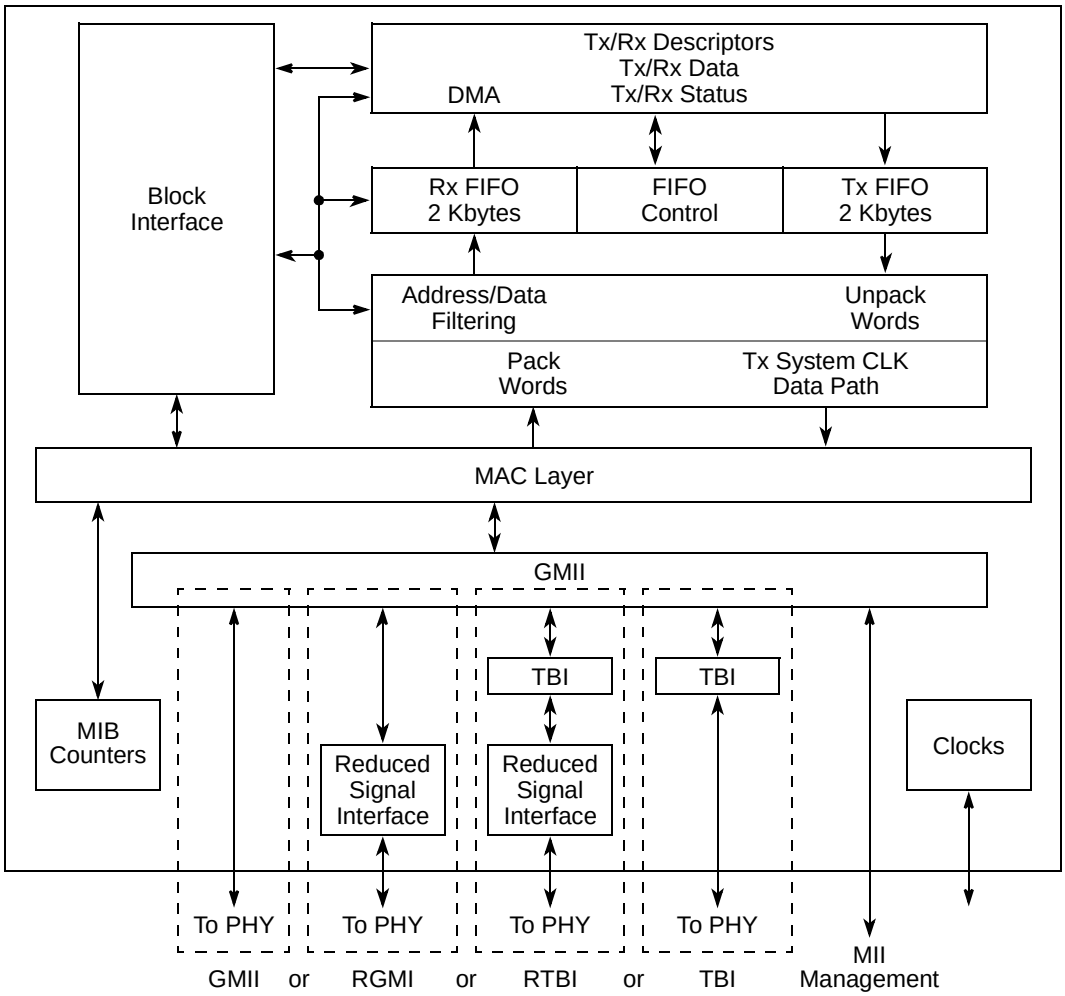

Figure 8. MPC8540 TSEC Block Diagram

The TSEC device is programmed by a combination of control/status registers and buffer descriptors. The CSRs are used for mode control, interrupts, and to access status information. The descriptors pass data buffers and related buffer status or frame information between the hardware and software. All TSEC control/status registers are 32-bits wide and are divided into the following sections depending on the functions they support:

- General control/status registers—The interrupt event (IEVENT) generates operational interrupts, transceiver/network error interrupts, if the corresponding bit in the interrupt enable register (IMASK) is also set. The Ethernet control register (ECNTR0) is used to reset, configure, and initialize the TSEC including: enabling/resetting counters for statistics, enabling TBI or MII mode, reduced or standard interfaces, and the interface speed. The pause timer value register (PTV) is written by the user to store the pause duration used when the TSEC initiates a pause.
- FIFO control/status registers—Allow the user to change some of the default settings in the FIFO that can be used to optimize operation for performance or for safety.
- Transmit specific control/status registers—The transmit control register (TCTRL) is used for half-duplex flow control operation and pause frame transmission requests. Additional registers are used to maintain TxBD parameters (including length, base address, and pointers).
- Receive specific control/status registers—The receive control register (RCTRL) enables broadcast reject, promiscuous mode, and receive short frame modes. Additional registers are used to maintain RxBD parameters (including length, base address, and pointers).
- MAC control/status registers—The MAC configuration register 1 (MACCFG1) is used to configure MAC functionality including: MAC transmit/receive enable, flow control transmit/receive enable, and loopback operation. The MAC configuration register 2 (MACCFG2) configures MAC functionality, including: preamble length, interface modes, padding/CRC enable, Hugh frame enable, and full- or half-duplex operation. The inter-packet gap/inter-frame gap register (IPGIFG) is used to program the inter-frame frame, and non-back-to-back and back-to-back inter-packet gaps. The half-duplex register (HAFDUP) is used to program features of the TSEC for half-duplex operation at 10/100 Mbps. The maximum frame length register (MAXFRM) sets the maximum frame size in both the transmit and receive directions. Additional registers are used for the MII management interface.
- MIB statistic registers—The MIB has 37 separate statistics or counter registers, which simply count or accumulate statistical events that occur as packets are transmitted and received.
- Hash function registers—The individual address registers (IADDR*n)* represent 256 entries of the individual (unicast) address hash table used in the address recognition process. While the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 bits of the CRC remainder is mapped to one of the 256 entries. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry. The group address registers (GADDR*n)* represent 256 entries of the group (multicast) address hash table used in the address recognition process. While the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 bits of the CRC remainder is mapped to one of the 256 entries. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry.

Finally, the attribute register (ATTR) defines attributes and transaction types to access buffer descriptors, to write receive data, and to read transmit data buffers including: snooping and L2 cache data extraction. Finally, a set of registers define operation for the ten-bit interface (TBI).

8.3 Porting Ethernet to the MPC8540

There are similarities between the IBM 440GP Ethernet MAC controllers and three-speed Ethernet controllers on the MPC8540, although the exact register formats are different. Key features of the Ethernet controllers and how registers from the IBM 440GP map to MPC8540 are shown in Table 26.

Table 26. Mapping Features of the IBM 440GP Ethernet Controller to the MPC8540

Feature	IBM 440GP Register	MPC8540 Register
Enable interrupts	EMAC*n*_ISR and EMAC*n*_ISER	IEVENT and IMASK
Interface modes/speed	ZMII0_FER and ZMII0_SSR	ECNTR0 and MACCFG2
Pause packet time	EMAC*n*_PTR	PTV
Address recognition (individual)	EMAC*n*_IAHR, EMAC*n*_IAHR EMAC*n*_IAHT1–4, EMAC*n*_GAHT1–4	IADDR0–7, GADDR0–8
Inter-packet (frame) gap	EMAC*n*_IPGVR	IPGFG
Enabled MAC/flow control functionality	EMAC*n*_MR0 and EMAC*n*_MR1	MACCFG1
Loopback operation	EMAC*n*_MR1	MACCFG1
RMON statistics	Limited support with octet counts in EMAC*n*_OCTRX and EMAC*n*_OCTTX registers	MIB has 37 separate statistics or counter registers

In addition, the MPC8540 has TSEC FIFO control and status registers to enable features including:

- FIFO thresholds
- Underrun trigger
- Flow control
- Interrupt coalescing

This allows the user to optimized Ethernet system performance and minimize underrun and overrun events.

9 UART

This section compares the IBM 440GP UART and the MPC8540 UART.

9.1 IBM 440GP UART

On the IBM 440GP there are two UART interfaces and two complete sets of UART registers (one for UART0 and one for UART1) almost identical to the MPC8540. There are 12 registers shown in Table 27 for each UART interface used for configuration, control, and status.

Table 27. IBM 440GP UART Register Summary

Name [1]	Description
UART*n*_RBR	UART receive buffer register
UART*n*_THR	UART transmitter holding register
UART*n*_IER	UART interrupt enable register
UART*n*_IIR	UART interrupt identification register
UART*n*_FCR	UART *n* FIFO control register
UART*n*_LCR	UART *n* line control register
UART*n*_MCR	UART *n* modem control register
UART*n*_LSR	UART *n* line status register
UART*n*_MSR	UART *n* modem status register
UART*n*_SCR	UART *n* scratch register
UART*n*_DLL	UART *n* divisor latch (LSB)
UART*n*_DLM	UART *n* divisor latch (MSB)

1 *n* = 0 or 1.

9.2 MPC8540 PowerQUICC III

The MPC8540 has two (dual) universal asynchronous receiver/transmitters (UARTs). There are two complete sets of DUART registers (one for UART0 and one for UART1). There are 14 registers shown in Table 28 for each UART interface used for configuration, control, and status.

Table 28. MPC8540 DUART Register Summary

Name [1]	Description
URBR*n*	UART receive buffer register
UTHR*n*	UART transmitter holding register
UDLB*n*	UART *n* divisor least significant byte register
UIER	UART interrupt enable register
UDMB*n*	UART *n* divisor most significant byte register
UIIR*n*	UART interrupt identification register
UFCR*n*	UART *n* FIFO control register
UAFR*n*	UART *n* alternate function register
ULCR*n*	UART *n* line control register
UMCR*n*	UART *n* modem control register
ULSRs	UART *n* line status register
UMSR*n*	UART *n* modem status register
USCR*n*	UART *n* scratch register
UDSR*n*	UART *n* DMA status register

1 *n* = 0 or 1.

9.3 Porting UART to the MPC8540

The UART programming mode between IBM 440GP and MPC8540 is almost identical. All the registers are 1 byte wide. Reads and writes to these registers must be byte-wide operations. However, there are two additional registers in the MPC8540 DUART programming model, as follows:

- UART alternate function registers (UAFRs), which enable software to write to both UART0 and UART1 registers simultaneously with the same write operation. The UAFRs also provide a means for the device performance monitor to track the baud clock.
- UART DMA status registers (UDSRs), which are read-only registers that return transmitter and receiver FIFO status.

10 I^2C

The inter-IC (I^2C) bus is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. The synchronous, multi-master bus of the I^2C allows the IBM 440GP or the MPC8540 to exchange data with other I^2C devices such as microcontrollers, EEPROMs, real-time clock devices, and A/D converters. Table 29 compares key I^2C features of the IBM 440GP and the MPC8540.

Table 29. I^2C Comparison of IBM 440GP versus MPC8540

Ethernet Feature	IBM 440GP	MPC8540
Clocking	100 and 400 kHz	Software programmable clock frequency
Addressing	7- and 10-bit	7-bit only
Data transfers	8-bit	8-bit
Support master/slave operation	Yes	Yes
Multi-master operation	Yes	Yes
On-chip filtering	No	Yes

Many similarities exist between the I^2C controllers on the IBM 440GP and the PowerQUICC III.

10.1 IBM 440GP I^2C Controller

The IBM 440GP supports two I^2C bus interfaces. Table 30 shows the registers for configuring the I^2C interfaces on the IBM 440GP. There are 1-byte wide registers for each I^2C interface.

Table 30. IBM 440GP I^2C Register Summary

Mnemonic	Register	Description
I2C*n*_MDBUF	I^2C master data buffer	I2C*n*_MDBUF and I2C*n*_SDBUF are a 1-byte × 4-byte FIFO buffer.
I2C*n*_SDBUF	I^2C slave data buffer	
I2C*n*_LMADR	I^2C low master address	I2C*n*_LMADR and I2C*n*_LSADR form the addresses that the I^2C interface transmits on the bus. The I^2C master and slave high address (I2C*n*_HMADR and I2C*n*_HSADR) registers provide the upper address bits for 10-bit addressing mode.
I2C*n*_HMADR	I^2C high master address	

Table 30. IBM 440GP I²C Register Summary (continued)

Mnemonic	Register	Description
I2C*n*_CNTL	I²C control	Starts and stops I²C interface master transfers on the I²C bus. When a transfer begins, the I²C interface uses the values in I2C*n*_CNTL to determine the type and size of the transfer. The I2C*n* mode control register (I2C*n*_MDCNTL) sets the major modes of operation on the I²C bus. In addition, I2Cx_MDCNTL can force the data buffers into the empty state.
I2C*n*_MDCNTL	I²C mode control	
I2C*n*_STS	I²C status	Contains the state of the I²C interface and the status of any previously requested master transfers.
I2C*n*_EXTSTS	I²C extended status	Reports additional I²C status. During and after transfers, software can read the I2C*n*_STS and I2C*n*_EXTSTS registers to determine the state of the I²C interface and the I²C bus.
I2C*n*_LSADR	I²C low slave address	I2C*n*_LMADR and I2C*n*_LSADR form the addresses that the I²C interface transmits on the bus. The I²C master and slave high address (I2C*n*_HMADR and I2C*n*_HSADR) registers provide the upper address bits for 10-bit addressing mode.
I2C*n*_HSADR	I²C high slave address	
I2C*n*_CLKDIV	I²C clock divide	Contains a clock divider ratio to determine the I²C base clock from the IBM 440 GP OPB (on-chip peripheral bus) clock.
I2C*n*_INTRMSK	I²C interrupt mask	Specifies which conditions can generate an I²C interrupt when the I²C interrupt is enabled.
I2C*n*_XFRCNT	I²C transfer count	Reports the number of bytes transferred on the I²C bus during a master or a slave operation.
I2C_XTCNTLSS	I²C extended control and slave status	Provides additional control of I²C interface functions and reports the status of slave operations.
I2C*n*_DIRECTCNTL	I²C direct control	Controls and monitors the I²C serial clock (I2CSCL) and serial data (I2CSDA) signal, is used for error recovery when a malfunction is detected on the I²C interface.

10.2 MPC8540 I²C Controller

The MPC8540 supports one I²C interface. Table 31 shows the registers (6 × 1-byte wide) for configuration, control, and status information of the I²C interface on the MPC8540.

Table 31. MPC8540 I²C Register Summary

Name	Description
I2CADR	I²C address register
I2CFDR	I²C frequency divider register
I2CCR	I²C control register
I2CSR	I²C status register
I2CDR	I²C data register
I2CDFSRR	I²C digital filter sampling rate register

The I^2C address register defines the address to which the I^2C interface responds when addressed as a slave. This is not the address sent on the bus during the address-calling cycle when the I^2C module is in master mode. The I^2C frequency divider register contains a frequency divider ratio to determine the serial bit clock frequency SCL from the CCB clock. The I^2C control register determines the I^2C controller mode of operation. The I^2C status register is read-only and software can use it to determine status information on the I^2C controller. The I^2C data register initiates transmission and reception of data one byte at a time. The I^2C digital filter sampling rate register assists in filtering out signal noise.

10.3 Porting I^2C to the MPC8540

The I^2C programming mode between IBM 440GP and the MPC8540 is almost identical. However, the IBM 440GP I^2C controller supports 10-bit addressing, and the programming model differs in that separate address/data registers are supported for master and slave I^2C operation. Therefore, the programming model on the MPC8540 for I^2C operation is much simpler. Table 32 illustrates how registers from the IBM 440GP map to MPC8540 for I^2C operation.

Table 32. Mapping Features of the IBM 440GP I^2C Controller to the MPC8540

Feature	IBM 440GP Register	MPC8540 Register
7-bit addressing	I2C_LMADR and I2C_LMSDR	I2CADR
10-bit addressing	I2C_HMADR and I2C_HSADR	—
Master/slave data buffers	I2C_MBUF and I2C_SBUF	I2CDR
Clock divide	I2C_CLKDIV	I2CFDR
Status information	I2C_STS	I2CCSR
Control	I2C_CNTL, I2C_MDCNTL and I2C_XTCNTLSS	I2CCCR

11 PCI-X

The PCI interface on the IBM 440GP and the MPC8540 are version 2.2 compliant and can support transaction speeds up to 66 MHz. In addition to PCI support, both devices support the PCI-X standard (version 1.0A) up to 133 MHz. Designers of systems incorporating PCI/PICE-X devices should refer to the respective specification for a thorough description of the PCI/PCI-X buses.

11.1 IBM 440GP PCI-X Bridge

The PCI-X bridge on the IBM 440GP provides an interface between the PCI bus and the processor local bus (PLB). It enables PCI initiators to access PLB slaves and PLB masters to access PCI targets. PCI initiators and PCI targets may be in PCI conventional or PCI-X mode. The PCI-X bridge includes an optional PCI arbiter typically used only in host-bridge mode. This internal arbiter can be used with up to six external PCI masters (six $\overline{\text{REQ}}$ and $\overline{\text{GNT}}$ pairs) or it can be disabled. When disabled, the PLB PCI-X bridge has one $\overline{\text{REQ}}/\overline{\text{GNT}}$ pair that can attach to an external arbiter.

The PCI-X bridge register memory map consists of internal registers used for controlling the PLB PCI-X bridge. These registers can be accessed from both the PLB and the PCI (if enabled). Most are accessed

from the PCI by type 0 configuration cycles when the IDSEL input of PCI-X bridge is active. The PCI-X bridge hardware implements the registers in little-endian byte ordering. Thus, software that runs in big-endian mode must take this into account when accessing the registers.

All PCI-X bridge registers on the IBM 440GP must be must be accessed with single-beat, 1- to 4-byte transfers. These are divided into the following sections depending on the functions they support:

- PCI-X standard header registers—Located at offsets 0x00-0x3f are part of the PCI defined standard header, and include PCI configuration address and data registers used to generate external PCI configuration cycles.
- Bridge options registers—Determine miscellaneous control operation of the PCI-X bridge including frequency of operation, interrupts sources, PCI-X enabled, etc.
- Error handling registers— 32-bit read/write registers to enable detection and reporting of errors on the PCI bus.
- PCI outbound/inbound registers—The PCI outbound (POM) registers are used to map PLB address space to PCI memory space and *vice versa* for PCI inbound (PIM) registers.
- MSI capability block definition registers—Generate interrupts to the PCI (outbound interrupts) and receive interrupts from the PCI (inbound interrupts). These interrupts may be either standard interrupt signals or message signaled interrupts (MSI).
- Power management register block definition registers—Include capabilities and power management status and control registers to determine functions related to power management as defined by the *PCI Bus Power Management Interface Specification, Version 1.1*.
- PCI-X capability block definition registers—Support additional standard registers defined by the standard for PCI-X addendum to the *PCI Local Bus Specification*.
- Simple message passing and inbound MSI registers—For the simple message passing mechanism and for inbound MSI.

PLB masters generate configuration cycles to the PCI bus by accessing the CONFIG_ADDRESS and CONFIG_DATA registers that are located in PLB space. CONFIG_ADDRESS and CONFIG_DATA must be accessed with single-beat PLB transfers that don't cross a 32-bit boundary. The general mechanism for accessing PCI configuration space is to write a value into CONFIG_ADDRESS that specifies the PCI bus number, the device number on that bus, and the configuration register in that device being accessed. Then, a read or write to CONFIG_DATA causes the bridge to generate the PCI configuration cycle, with the address and type translated from the CONFIG_ADDRESS value. The format of the CONFIG_ADDRESS register is shown in Figure 9. The CONFIG_ADDRESS register controls what type of cycle is generated when CONFIG_DATA is accessed.

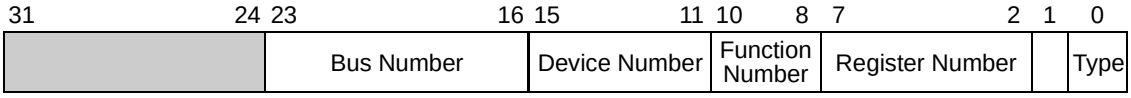

Figure 9. Format of IBM 440GP PCI CONFIG_ADDRESS Register

The PLB PCI-X bridge controller supports detection and reporting of several types of errors. The errors are reported to the PLB or the PCI and status information is saved in the configuration register set, so that error type determination can be done.

The PLB PCI-X bridge register set has several ranges of PLB address space and several ranges of PCI address space that it can claim. These ranges allow a PLB master to access the internal register set, and to cause the PLB PCI-X bridge to generate memory, I/O, configuration, and special cycles to the PCI bus.

If the destination is PCI memory space, there are up to three PLB address ranges that the PLB PCI-X bridge can claim. The number of ranges and their location in PLB address space is programmable and determined by the PCI outbound map 'POM' registers. Figure 10 shows how the POM registers map PLB address space to PCI address space.

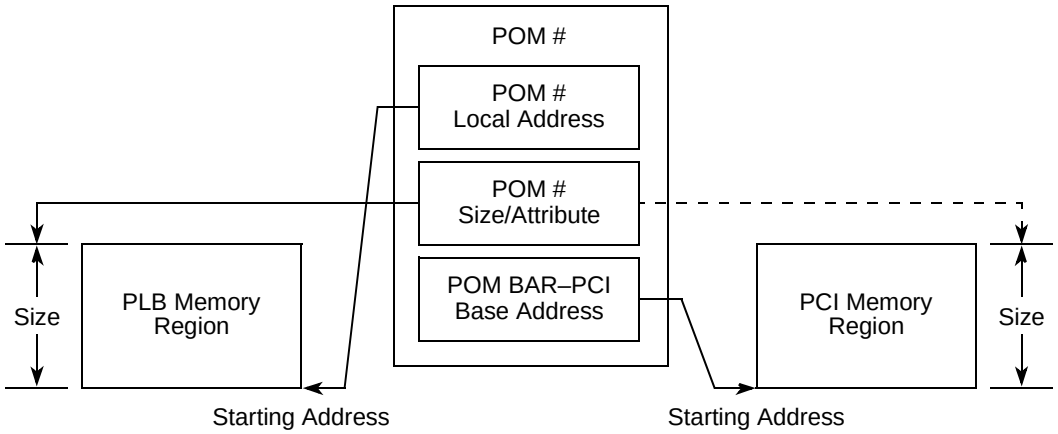

Figure 10. Registers Used in PCI Outbound Address Translation

If the destination is PLB space, there are up to two PCI memory address ranges and up to one PCI I/O address range that the PLB PCI-X bridge can claim. The number of ranges and their locations in PCI address space is programmable and determined by the PCI inbound map (PIM) and BAR registers. Figure 11 shows how the PIM and BAR registers map PCI address space to PLB memory space.

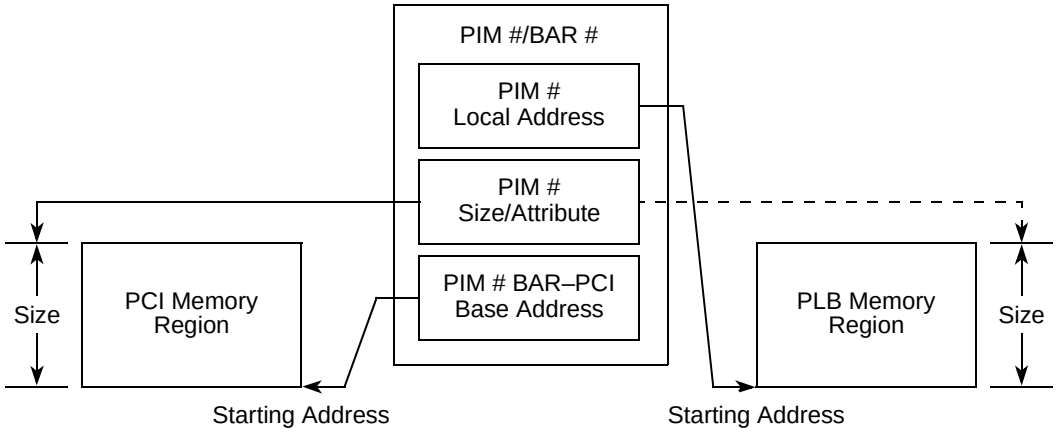

Figure 11. Registers Used in PCI Inbound Address Translation

The simple message passing mechanism enables messages to be exchanged between a PCI master device (the host) and a PLB master device (the local CPU). The interface consists of a 32-bit message in register and a 32-bit message out register. For 64-bit messages, the message in high and message out high registers may be used.

11.2 MPC8540 PCI-X Controller

The PCI bus architecture is a hierarchical, multi-master arbitration scheme that uses either 32- or 64-bit addressing to post transactions onto the PCI bus. Transactions can be either accepted, retried, or deferred. In the later two cases, the master repeats any transaction that needs to be retried and deferred transactions can be accepted and started by the target while the master retries the transaction.

The MPC8540 has five different pairs of request/grant pairs, and hence up to five external PCI masters can be supported. The performance of the PCI interface is enhanced by the two level round-robin arbitration algorithm used in the arbiter and through the ability to do mirror and pre-fetched PCI read accesses. While the PCI interface supports both inbound and outbound data streaming, the amount of data that can actually be streamed is limited by both the depth of pre-fetching and the target disconnect limit of the PCI specification. On the MPC8540, this disconnect will occur after two cache lines (that is, after 32 bytes). This helps PCI devices hogging the bus and prevents system bottlenecks/interface starvation, when operating high speed interfaces such as RapidIO or Gigabit Ethernet.

The MPC8540 PCI-X controller supports the following register types:

- Memory-mapped registers—these registers control PCI address translation (inbound/outbound), PCI error management, and PCI configuration register access on the MPC8540.
- PCI-X configuration registers contained within the PCI-X configuration header—these registers are specified by the PCI bus specification for every PCI device.

The PCI-X memory-mapped registers are accessed by reading and writing to an address comprised of the base address (specified in the CCSRBAR on the local side or the PCSRBAR on the PCI-X side) plus the offset of the specific register to be accessed. Note that all memory-mapped registers (except the PCI-X configuration data register, PCI CFG_DATA) must only be accessed as 32-bit words.

The outbound address translation and mapping unit controls the mapping of transactions from the internal 32-bit address space of the MPC8540 to the external PCI address space. The outbound ATMU consists of four translation windows plus a default translation for transactions that do not hit in one of the four windows. Each window contains a base address that points to the beginning of the window in the local address map, a translation address that specifies the high-order bits of the transaction in the external PCI address space, and a set of attributes including window size and external transaction type.

The inbound address translation and mapping unit controls the mapping of transactions from the external PCI address space to the local address space of the MPC8540. The inbound ATMU is comprised of four windows—a configuration window (highest priority) and three general translation windows. Each window contains a base address, which points to the beginning of the window in the external PCI address map, a translation address that specifies the upper order bits of the transaction in the local address space, and a set of attributes including window size and internal transaction attributes.

When a PCI-X error is detected, the appropriate error bit is set in the PCI-X error detect register. Subsequent errors set the appropriate error bits in the error detection registers, but relevant information (attributes, address, and data) is captured only for the first error.

The *PCI Local Bus Specification* defines the configuration registers contained within the PCI-X configuration header from 0x00 through 0x3F. The *PCI-X Addendum* to the *PCI Local Bus Specification* defines additional registers beyond 0x3F. The common PCI-X configuration header implemented by the

MPC8540 is shown in Figure 12, and is accessed via an indirect method using a pair of 32-bit memory-mapped access registers, CFG_ADDR and CFG_DATA.

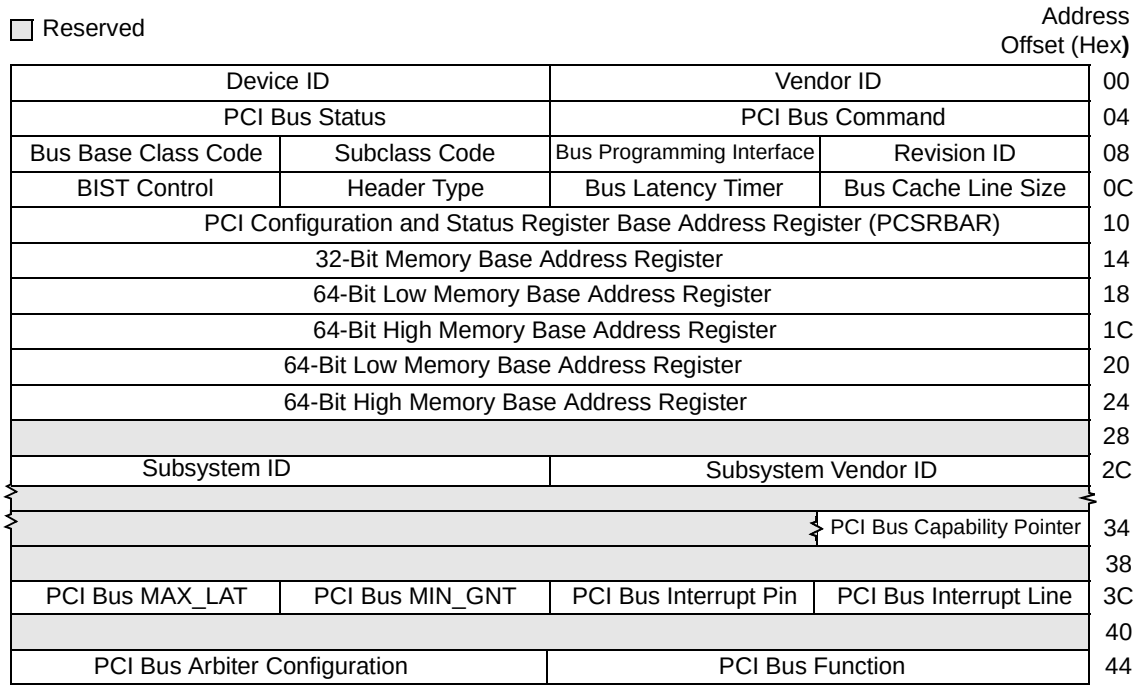

Figure 12. MPC8540 Common PCI-X Configuration Header

To access the configuration space, a 32-bit value must be written to the PCI CFG_ADDR register specifying the target PCI bus, the target device on that bus, and the configuration register to be accessed within that device. A read or write to the PCI CFG_DATA register causes the host bridge to translate the access into a PCI configuration cycle (provided the enable bit in CONFIG_ADDR is set and the device number is not 0b1_1111). The translated information will be according to type 0 (if the bus number is 0 and device number is not 31) or type 1 (if the bus number is not equal to 0).

11.3 Porting PCI to the MPC8540

The PCI interface on the IBM 440GP and the MPC8540 are both version 2.2 compliant and support the PCI-X standard (version 1.0A). Therefore, many programming similarities between the PCI programming mode of the IBM 440GP and the MPC8540 exist, although the exact register formats do differ. Table 33 illustrates how registers from the IBM 440GP map to the MPC8540 for general PCI operation.

Table 33. Mapping Features of the IBM 440GP PCI-X Controller to the MPC8540

Feature	IBM 440GP Register	MPC8540 Register
PCI configuration header	PCI standard header registers (offsets 0x00–0x3F)	Accessed indirectly using bus and device number in the CONFIG_ADDR register. PCI and PCI-X standard header registers (offsets 0x00–0x44)
PCI type transaction	CONFIG_ADDR register Bit 31—Reserved Bit 0—Determines the PCI transaction type	PCI type transaction determined from translated information in the CONFIG_ADDR register. Type 0, if bus number is 0 and device number is not 31. Type 1, if bus number is not equal to 0. Note on CONFIG_ADDR register, bit 31 is enabled and bit 0 is reserved.
Inbound transactions—window size and attributes	PICX0_PIM0SA PICX0_PIM1SA PICX0_PIM2SA	PIWAR 1–3
Inbound transactions—address mapping	PICX0_PIM0LAL PICX0_PIM0LAH PICX0_BAR0L PICX0_BAR0H PICX0_PIM1LAL PICX0_PIM1LAH PICX0_BAR1 PICX0_PIM2LAL PICX0_PIM2LAH PICX0_BAR2L PICX0_BAR2H	PITAR 1–3 PIWBAR 1–3 PIWBEAR 1–3
Outbound transactions—window size and attributes	PICX0_POM0SA PICX0_POM1SA PICX0_POM2SA	POWAR 0–4
Outbound transactions—address mapping	PICX0_P0M0LAL PICX0_P0M0LAH PICX0_POM0PCIAL PICX0_POM0PCIAH PICX0_P0M1LAL PICX0_P0M1LAH PICX0_POM1PCIAL PICX0_POM1PCIAH POM 2 fixed hardware coded address fixed	POTAR 0–4 POTEAR 0–4 POWBAR 0–4
Error handling—enable errors and status/detection	PCIX0_ERREN PCIX0_ERRSTS	ERR_EN ERR_DR
Error handling—attributes	PCIX0_PLBBESR (error attributes)	ERR_ATTRIB

Table 33. Mapping Features of the IBM 440GP PCI-X Controller to the MPC8540 (continued)

Feature	IBM 440GP Register	MPC8540 Register
Error handling—address capture	PCIX0_PLBEARL PCIX0_PLBEARH	ERR_ADDR ERR_EXT_ADDR
Error handling—data capture	N/A	ERR_DL ERR_DH

12 Conclusions

Migrating to a new hardware platform has inherent software implications and typically becomes a challenge many software engineers are unwilling to face. Fortunately, both the IBM 440GP and the MPC8540 are based on a 32-bit implementation of the Book E PowerPC architecture, enabling high-level software to port with minimal trouble. The low-level feature set of the IBM 440GP is a subset of the features available on the MPC8540, with many similarities between implementation in the two architectures. As shown in this document, porting low-level code is a relatively straightforward process. Software migration is no longer a deciding factor in processor selection. Performance, flexibility, and features become the deciding factors, and the PowerQUICC III family provides a high level of performance and integration at a low cost for today's embedded systems.

13 Document Revision History

Table 34 provides a revision history for this document.

Table 34. Document Revision History

Rev. No.	Date	Substantive Change(s)
0	5/26/2004	Initial release.
1	1/2007	Non-substantive formatting.

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: AN2661
Rev. 1
01/2007