PowerPC Embedded Processors
Application Note

PowerPC 405GP Endianness and Bit Naming Conventions

IBM Microelectronics
Research Triangle Park, NC
ppcsupp@us.ibm.com
http://www.chips.ibm.com

January 5, 2000 Version: 1.1

Abstract *- This application note will help users of the PowerPC 405GP embedded controller chip understand various aspects of big and little endian support in the 405GP as well as how the bit naming conventions used for the chip apply to the PowerPC processor core and the peripherals and interfaces of the chip.*

1. Definitions

Endianness refers to the byte data significance versus byte address assignment of a multi-byte scalar quantity. *Little endian* assignment specifies that the least significant byte of a scalar item is assigned to the lowest byte address value. *Big endian* assignment specifies that the most significant byte of a scalar item is assigned to the lowest byte address value. Please note that endianness does not affect single byte items.

Bit naming conventions refer to the assignment of sequence numerals in a bus or register name versus the bit significance of a scalar quantity. Byte naming conventions refer to the assignment of sequence numerals for the byte lanes of a bus or register versus the byte significance of a multi-byte entity.

Please note well that bit and byte naming conventions are **not** related to the concept of endianness; the terms little endian and big endian **should not** be used to refer to naming conventions.

2. PowerPC and industry common bit naming conventions

The PowerPC architecture specifies a bus and register bit naming convention in which the most significant bit (MSB) name incorporates the numeral 0. As the significance of the bits decrease across the bit vector (register or bus), the numeral in the name increases linearly such that a 32-bit vector will have a least significant bit (LSB) name with the numeral 31. Most common interfaces at this time use thew opposite convention where a name with the numeral 0 represents the least significant bit vector position. Table 1 shows the correspondence of bit names between the two conventions.

PowerPC Architecture Bit Naming Convention											
MSB bit 0	bit 1 : bit 6	bit 7	bit 8	bit 9 : bit 14	bit 15	bit 16	bit 17 : bit 22	bit 23	bit 24	bit 25 : bit 30	LSB bit 31
Most Significant Byte			Next Byte			Next Byte			Least Significant Byte		
bit 31 MSB	bit 30 : bit 25	bit 24	bit 23	bit 22 : bit 17	bit 16	bit 15	bit 14 : bit 9	bit 8	bit 7	bit 6 : bit 1	Bit 0 LSB
Industry Common Bit Naming Convention											

Table 1: Bit Naming Conventions

The 405 CPU registers, 405GP internal data paths and address buses, and some of the external interfaces use the PowerPC bit naming convention. Certain external interfaces use the industry common convention.

A key point to note is that for a 32-bit address reference A0:A31, A0 is the most significant address bit, and A31 is the least significant address bit. The least two significant address bits represent the byte offset address from a 32-bit word address.

3. Data types and endianness in the PowerPC 405GP

Hardware supported data types of the PowerPC 405GP are byte, half-word, word and byte-string. Data types resident in a register are positioned in the least significant byte positions of the register. Table 2 shows a register view of word, half-word and byte operands.

PowerPC 405GP Data Types	Register R0:R7	Register R8:R15	Register R16:R23	Register R24:R31
Word (32 bits)	Byte 0	Byte 1	Byte 2	Byte 3
Half-word (16 bits)			Byte 0	Byte 1
Byte (8 bits)				Byte 0

Table 2: PowerPC 405GP data types

Items of the supported data types are stored in memory at a byte address representing the most significant byte of the item for big endian storage, or least significant byte for little endian storage. Subsequent bytes of the item are stored at linearly increasing byte address locations. The 405GP supports both big endian and little endian views of memory storage for the data types supported. Table 3 shows how the basic data types are stored in memory in both big and little endian modes.

Storage Mode	**Data Type**	Memory Address – byte offset A30:A31			
		b00	b01	b10	b11
Big Endian	Word	Byte 0	Byte 1	Byte 2	Byte 3
	Half-word	Byte 0	Byte 1		
	Byte	Byte 0			
Little Endian	Word	Byte 3	Byte 2	Byte 1	Byte 0
	Half-word	Byte 1	Byte 0		
	Byte	Byte 0			

Table 3: Endianness vs. memory storage for the data types

Multi-byte data items in memory are not required to be naturally aligned, e.g. a 32-bit word can be stored at an address with a byte offset of 01. PowerPC instructions are always single 32-bit words and **must** be naturally aligned (located at an address with a byte offset of 00). More detailed examples of the various

data types mapped into big-endian and little-endian memory spaces are in the PowerPC 405GP User's Manual, section 3.4, "Byte Ordering".

The 405 CPU treats all operands in the ALU and GPRs as big endian. The CPU expects all instructions fed into the pipeline to be in big endian format. Byte swapping in the data path to or from memory provides for support of little endian data storage. The instruction interface also has byte steering logic to support instruction fetch from little endian memory. All memory is viewed as big endian by default. A memory region may be programmed as little endian via a memory attribute. The endian storage attribute for a memory region is sourced either from the SLER register when operating with the memory management unit (MMU) disabled or from the endian attribute in the MMU's TLB entries. It is important to note that when the 405 CPU boots, ALL memory is defined big endian. The PowerPC 405GP User's Manual, section 3.4.3, "Endian Storage Attribute" provides a more detailed explanation.

4. Instruction fetch considerations

Table 5 shows the endianness relationship of instructions in the CPU pipeline, instruction cache and memory for both big endian and little endian organizations. Instructions in memory can be in big or little endian format; in the I-cache, they are always stored in big endian format. The 405 CPU's internal instruction bus interface performs the byte swapping necessary when instructions are stored in little endian memory. Even though it is possible for PowerPC instructions to be stored in little endian memory space, there are no apparent advantages to doing so. It is highly recommended that PowerPC code be located in big endian storage. Because the 405 CPU defines all storage as big endian at boot, the boot code **must** be ordered as big endian. When using the 405GP's on-chip memory (OCM) for instruction storage, this memory space **must** be defined as big endian because the byte steering logic is not in the instruction-side OCM interface's path to the instruction pipeline of the 405 CPU.

5. Load / store data access considerations

Table 6 shows an example of the endianness relationship of four-byte (word) data in the registers, data cache and memory for both big endian and little endian organizations. Byte steering logic that provides for little endian support is between the 405 CPU and the data cache. This means that data resident in both the cache and system memory is either big endian or little endian depending on the corresponding endian attribute. The diagram shows the byte steering for a 32-bit word, but the logic also provides for the correct byte steering for half-word loads and stores as described in Table 3. When using the 405GP's on-chip memory (OCM) for data storage, this memory space may be defined for either big endian or little endian mode.

The PowerPC architecture also provides byte-reverse load and store instructions as a mechanism for obtaining little endian storage and as a means to change the endianness of a data word; these instructions are documented in Table 4. It should be noted that when using the byte reverse load and store instructions to change the endianness of data, it does not automatically change the storage attribute definition. It is possible to have little endian (byte reversed big endian) data stored in a big endian memory region. Other code that operates on the data may assume that the data is big endian because it is in a big endian region.

Mnemonic	Description of PowerPC Byte-reverse Instruction
lhbrx	Load half-word to register with byte reverse
lwbrx	Load word to register with byte reverse
sthbrx	Store half-word to memory with byte reverse
stwbrx	Store word to memory with byte reverse

Table 4: Byte reverse instructions

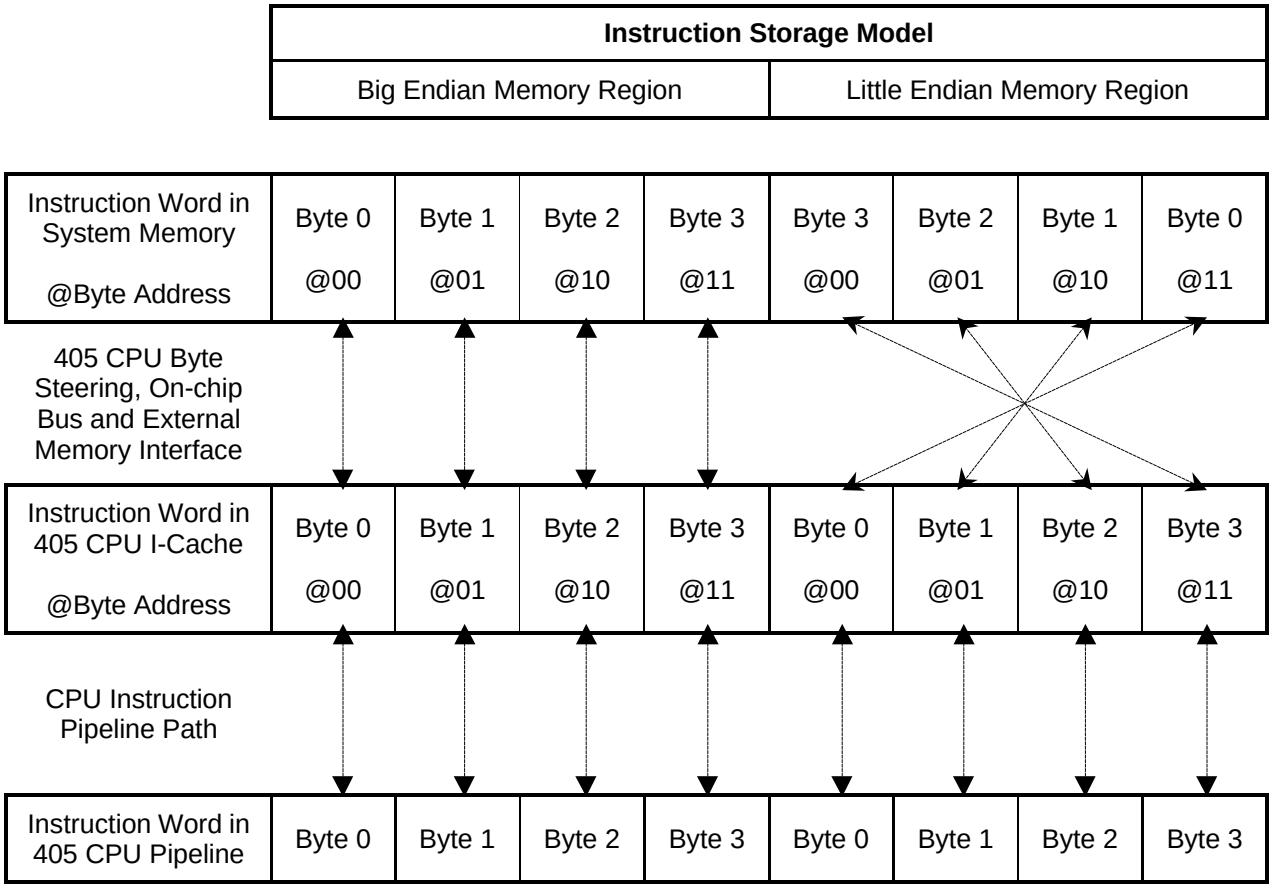

Table 5: Instruction storage for big and little endian modes

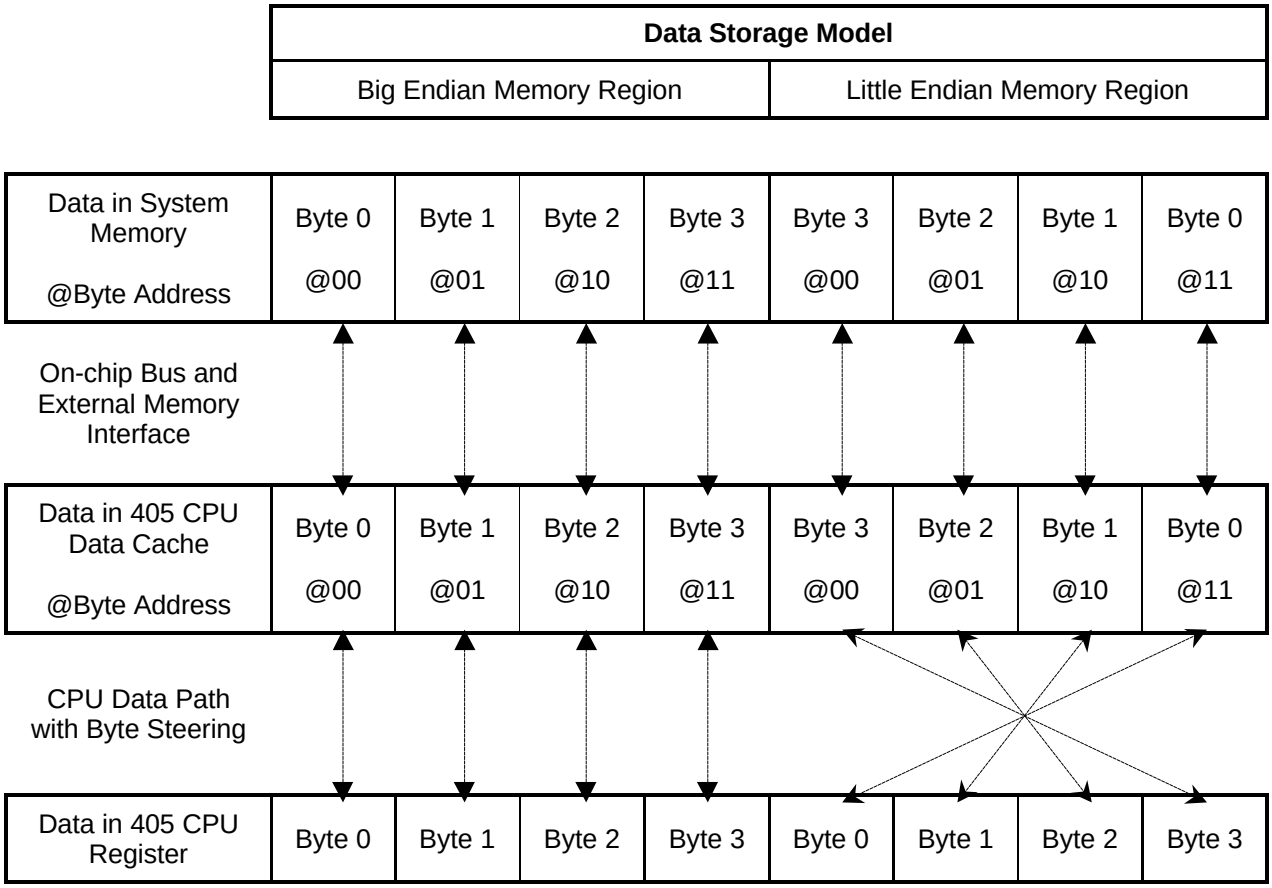

Table 6: Data storage for big and little endian modes

6. Naming conventions for interfaces of the PowerPC 405GP chip

Table 7 shows the address bus naming conventions used for the major functional blocks and interfaces on the 405GP chip. The 405 CPU, on-chip buses and external bus interface use the PowerPC bit naming conventions. Since the PCI bus is an industry standard interface, it uses the PCI specified conventions for its multiplexed address / data bus. The SDRAM interface uses industry common naming conventions for the address bus. Please consult the PowerPC 405GP User's Manual, section 13.4, "PLB Physical Address to Memory Address Mapping" for details on how the SDRAM address pins are used in a row / column manner configuration for various SDRAM configurations.

Table 8 shows the data bus naming conventions used for the major functional blocks and interfaces on the 405GP chip. As with the address bus, the 405 CPU, on-chip buses and external bus interface use the PowerPC bit naming conventions. Again, the PCI bus uses the PCI specified conventions for its multiplexed bus. For the SDRAM interface, data bit 0 corresponds to the 405 CPU and on-chip bus data bit 0.

Table 9 shows the data bus byte enable naming conventions used for the major functional blocks and interfaces on the 405GP chip. It shows the byte enable assignment to the chip bus byte lanes and the address correspondence to the byte enables for a byte address.

Functional Unit or Interface	Address Bus Naming				
	Word Address			Byte Address	
	MSB		Word LSB	Byte MSB	LSB
405 CPU	A0	A1 : A28	A29	A30	A31
External Bus Interface	PerAddr0	PerAddr1 : PerAddr28	PerAddr29	PerAddr30	PerAddr31
PCI Bus Interface	PCIAD31	PCIAD30 : PCIAD3	PCIAD2	PCIAD1	PCIAD0
SDRAM Interface	MemAddr12	MemAddr11 : MemAddr1	MemAddr0		

Table 7: Address Bus Naming

Functional Unit or Interface	Data Bus Naming and Byte Lane Assignment					
	MSB	BL0	BL1	BL2	BL3	LSB
405 CPU	D0 : D7		D8 : D15	D16 : D23	D24 : D31	
External Bus Interface	PerData0 : PerData7		PerData8 : PerData15	PerData16 : PerData23	PerData24 : PerData31	
PCI Bus Interface	PCIAD31 : PCIAD24		PCIAD23 : PCIAD16	PCIAD15 : PCIAD8	PCIAD7 : PCIAD0	
SDRAM Interface	MemData0 : MemData7		MemData8 : MemData15	MemData16 : MemData23	MemData24 : MemData31	

Table 8: Data Bus Naming

Functional Unit or Interface	Byte Enable Naming and Byte Address Correspondence			
405 CPU	BE0 ---------- A[30:31] = b00	BE1 ---------- A[30:31] = b01	BE2 ---------- A[30:31] = b10	BE3 ---------- A[30:31] = b11
External Bus Interface	PerWBE0* ---------- PerAddr[30:31] = b00	PerWBE1* ---------- PerAddr[30:31] = b01	PerWBE2* ---------- PerAddr[30:31] = b10	PerWBE3* ---------- PerAddr[30:31] = b11
PCI Bus Interface	PCIC0[BE0*] ---------- PCIAD[1:0] = b00	PCIC1[BE1*] ---------- PCIAD[1:0] = b01	PCIC2[BE2*] ---------- PCIAD[1:0] = b10	PCIC3[BE3*] ---------- PCIAD[1:0] = b11
SDRAM Interface	DQM0	DQM1	DQM2	DQM3

Table 9: Byte Enable Naming

7. Interface endianness issues

Little endian support is a common requirement for exchanging data with x86 architecture peripherals and for data exchange across a PCI interface as normal data definition in PCI space is little endian. Please note that there are no endian translation mechanisms on 405GP chip other than 405 CPU's load / store instructions in conjunction with memory region endianness attributes, and the byte-swapping load / store instructions. The 405 CPU can be used to perform endianness translation by the following method:

1. Use a load followed by byte-reverse store (or byte reverse load followed by a store) loop to modify data in place in a memory region.

2. Change the endian attribute for that memory region.

Table 10 provides an example of data mapping between the 405 CPU and PCI address space. This table assumes a little endian attribute value for the memory region mapped to the PCI memory space.

Data logical	Size bytes	405 CPU Address [30:31].	405 CPU Byte Lanes				PCI Byte Lanes				PCI Address [1:0]
			BL0	BL1	BL2	BL3	BL3	BL2	BL1	BL0	
11	1	b00	11							11	00
		b01		11					11		01
		b10			11			11			10
		b11				11	11				11
1122	2	b00	22	11					11	22	00
		b10			22	11	11	22			10
11223344	4	b00	44	33	22	11	11	22	33	44	00

Table 10: 405 CPU to PCI memory map for PowerPC data types