White Paper

Sam Fleming

Technical Marketing Engineer

Intel Corporation

Accessing PCI Express* Configuration Registers Using Intel® Chipsets

December 2008

321090

Executive Summary

There are two different methods that can be used to access PCI Express* (PCIe*) configuration registers of devices in systems based on Intel silicon. The first method involves setting the desired Bus, Device, Function, and Register values in I/O register CF8h and then reading or writing the desired value via an access to I/O register CFCh. These registers are located in the memory controller component of most Intel chipsets (typically the "North Bridge" or MCH/GMCH component). The act of reading/writing to I/O register CFCh causes this component to generate downstream PCIe configuration bus cycles. This is the original "legacy" PCI method of accessing the configuration registers and is limited to accessing only the first 255 configuration registers of any PCIe device.

The second method that can be used to access PCIe* configuration registers involves standard memory read/write accesses to a reserved block of memory space. Intel® silicon is designed to convert these memory cycles into downstream PCIe configuration read and write cycles. This permits access to the full 4096 configuration register range in each PCIe device.

The second method can be a bit tricky for software programmers to implement.

This second method can be a bit a tricky for software programmers to implement, especially considering that some of the details vary from chipset to chipset. This paper will examine the exact formulas and processes that software developers need to implement in order to successfully read and write to these PCIe* registers.§

Contents

Business Challenge 4

PCIe* Configuration Register Access – Method 1: I/O CF8h and CFCh 4

PCIe* Configuration Register Access – Method 2: Memory Mapped 6

Conversion Formulas 8

Chipset Variance 9

PCIe* Memory Range and How It Impacts Available DRAM in 4GB Systems Running 32-bit Operating Systems 9

Conversion Formulas – Code 10

Conclusion 11

Business Challenge

The methodology used in accessing the PCIe* configuration registers goes back to the original PCI specification. Intel's first implementation of this specification used I/O locations CF8h and CFCh in an index/data fashion in order to create the PCI configuration cycles. This method can successfully access any of the 255 PCI configuration registers in any PCI device. Intel chipsets still support this configuration method.

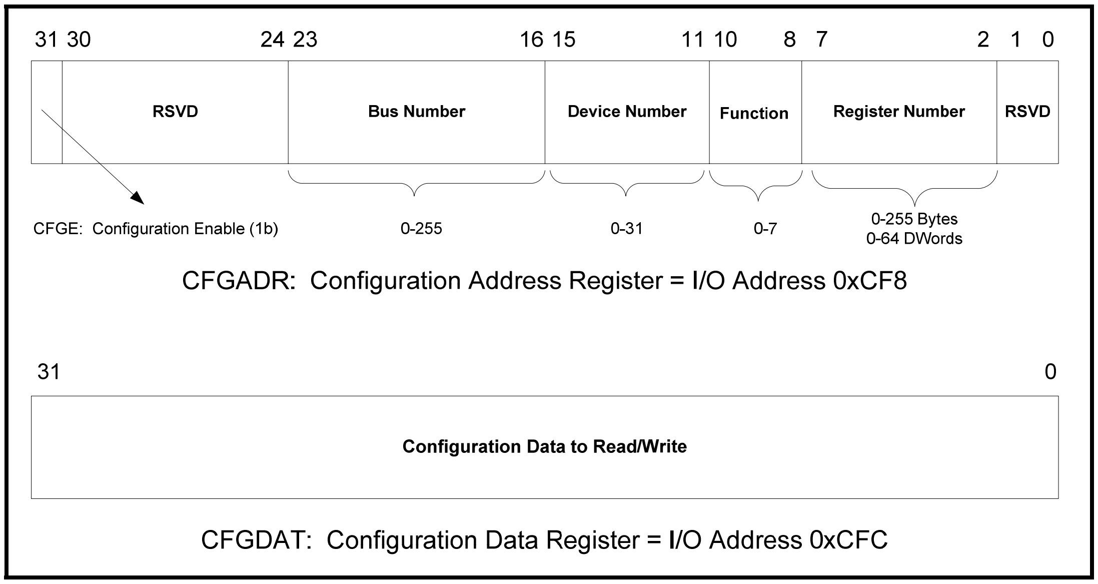
The PCIe* specification built on the PCI foundation by allowing each PCIe device/function to have up to 4096 configuration registers. Although the old CF8h/CFCh method can still be used to access the first 255 registers in each PCIe device, a different method had to be introduced to allow accesses to the full 4KB range of configuration registers. Intel chipsets solve this problem by moving the PCIe configuration registers into the memory address space. PCIe configuration registers can now be accessed by performing memory reads and writes to a very specific range in memory.

The specific range of memory is controlled by the piece of Intel silicon that contains the memory controller. Typically, this is the "North Bridge" component. However, details of the implementation vary from chipset to chipset. Understanding the exact process in creating software to access the PCIe* configuration registers using this range of memory is the purpose of this paper.

PCIe* Configuration Register Access – Method 1: I/O CF8h and CFCh

To access PCI configuration registers, Intel® chipsets use I/O locations CF8h and CFCh to access the configuration registers in a PCI device. This method can also be used to access the first 255 configuration registers in PCIe* devices.

Figure 1. I/O Registers CF8h and CFCh Details


31 30
24 23
16 15
11 10
8 7
2 1 0
RSVD
Bus Number
Device Number
Function
Register Number
RSVD
CFGE: Configuration Enable (1b)
0-255
0-31
0-7
0-255 Bytes
0-64 DWords
CFGADR: Configuration Address Register = I/O Address 0xCF8
31
0
Configuration Data to Read/Write
CFGDAT: Configuration Data Register = I/O Address 0xCFC

In order to create a PCI Configuration Cycle to a given device, software must perform the following steps:

1. The device's bus number must be written into bits [23:16] of I/O location CF8h
2. The device's "PCI device number" must be written into bits [15:11] of I/O location CF8h
3. The device's function must be written into bits [10:8] of I/O location CF8h

   Note: The Dword address of the desired PCI configuration must be written into bits [7:2] of I/O location CF8h

   It is important to realize that all PCI configuration cycles are 4 byte (1 Dword) read/write routines. Since bits [1:0] are reserved, this simplifies things a bit since the actual register address can be written into the full 8 bits of [7:0] (data written to bits [1:0] has no affect on the created PCI configuration cycle).

4. The configuration bit in I/O location CF8h[31] must be set to 1b.
5. For writes, the specific data to be written out during the configuration cycle is written into I/O location CFCh. For reads, software must perform a read to I/O location CFCh.

   When I/O location CFCh is accessed, the corresponding read or write

PCI configuration cycle is created by the Intel® silicon and propagated throughout the system as necessary.

PCIe* Configuration Register Access – Method 2: Memory Mapped

The PCIe* specification added more configuration registers possible for each PCIe device. Each PCIe device can now have up to 4096 bytes of configuration registers. Although the method described above using I/O locations CF8h and CFCh to create configuration cycles still works for accessing the lower 255 registers, a new method must be used to access configuration registers to the rest of the 4096 (4K) range.

The solution involves reserving a 256MB block of memory address space. Any accesses to this region of memory address space will cause the Intel silicon to create a PCIe configuration cycle to be propagated throughout the system.

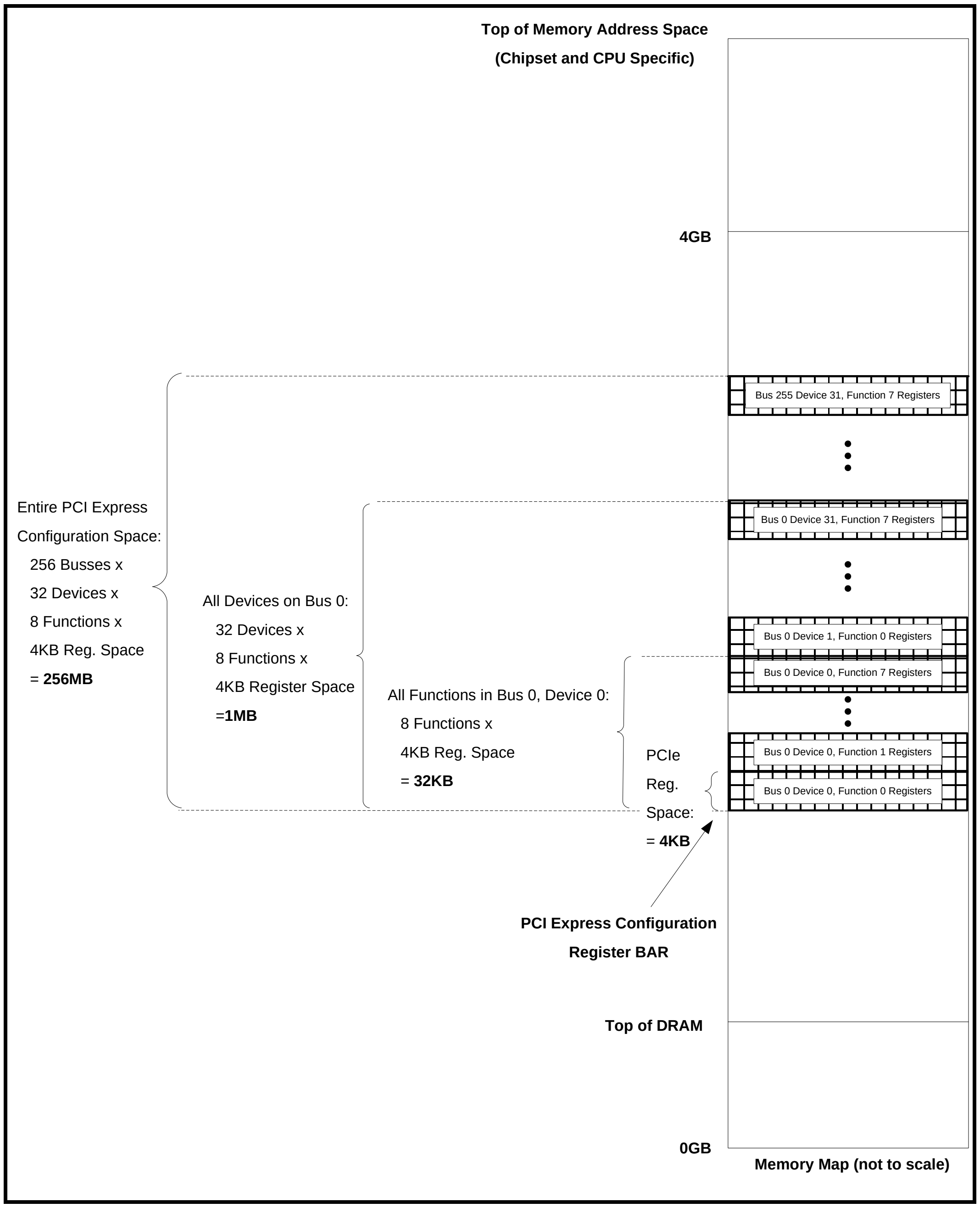
Since the registers are mapped directly into memory, the memory address space that needs to be reserved must include all 4096 registers for every possible function of every possible device on every possible bus that could be present in the system. This calculates out to 256MB of memory address space being required to “fit” all of the possible configuration registers:

```
  256    Possible Busses
   32    Possible Devices per Bus
    8    Possible Functions Per Device
 4096    Registers per Device/Function

=========================
 256 x 32 x 8 x 4096 = 256MBytes
```

Depending on the specific Intel® chipset, this block of memory can be moved anywhere within the system memory map. In the Intel® Q45 and Intel® PM965 Express Chipsets, for example, a register in the GMCH (or “Northbridge”) called the PCI Express* Configuration Register Base Address Register (BAR) located in PCI configuration space at B0:D0:F0-60h performs this function. This is shown in Figure 2 below.

Figure 2. Memory Mapped PCIe* Configuration Registers


Top of Memory Address Space
(Chipset and CPU Specific)
4GB
Bus 255 Device 31, Function 7 Registers
Entire PCI Express
Configuration Space:
256 Busses x
32 Devices x
8 Functions x
4KB Reg. Space
= 256MB
Bus 0 Device 31, Function 7 Registers
All Devices on Bus 0:
32 Devices x
8 Functions x
4KB Register Space
=1MB
Bus 0 Device 1, Function 0 Registers
Bus 0 Device 0, Function 7 Registers
All Functions in Bus 0, Device 0:
8 Functions x
4KB Reg. Space
= 32KB
Bus 0 Device 0, Function 1 Registers
Bus 0 Device 0, Function 0 Registers
PCIe
Reg.
Space:
= 4KB
PCI Express Configuration
Register BAR
Top of DRAM
0GB
Memory Map (not to scale)

When software wants to access a specific configuration register in a given device, it must calculate exactly where this register resides in the PCIe* configuration memory map and perform a simple memory read/write to this location.

Conversion Formulas

Figure 3 provides the formula used to calculate the desired memory location needed to access a PCIe* configuration register given the device's bus number, device number, function number, and register offset.

Figure 3. Formula to Calculate Memory Address Given the Bus, Device, Function, and Register Offset

```
Memory Address =
        PCI Express* Configuration Space Base Address  +
        (Bus Number x 100000h)                         +
        (Device Number x 8000h)                        +
        (Function Number x 1000h)                      +
        (Register Offset)
```

For example, to access the following configuration register:

- PCI Express Configuration Register F0000000h
- Bus Number 15h
- Device Number 00h
- Function Number 05h
- Register Offset 84h

Software would access this register by performing a memory read/write to address F1505084h.

Calculating the Bus, Device, Function, and Register offset given an actual address is a little bit more complex. The formula shown in Figure 4 can be used when given just an address.

Figure 4. Formula to Calculate Bus, Device, Function, and Register Offset Given a Memory Address

```
Bus Number =        (Memory Address - PCI Express* Configuration Space Base Address) /
                    (100000h)

Device Number =     (Memory Address - PCI Express Configuration Space Base Address -
                    (Bus Number x 100000h))  /
                    (8000h)

Function Number =   (Memory Address - PCI Express Configuration Space Base Address -
                    (Bus Number x 100000h) - (Device Number x 8000h))  /
                    (1000h)
```

```
Register Offset =    (Memory Address) & (00000FFFh)
```

Chipset Variance

The memory address range containing the PCIe* configuration registers is implemented differently between different Intel chipsets. In Intel's older chipsets, the PCI Express* Configuration Base Address Register is contained inside the memory controller portion of the chipset (MCH and GMCH). The location of this register varies between chipsets. For the Intel® Q45 and PM965 Express chipsets, for example, this register is located in PCI space at B0:D0:F0-60h. In the Intel® EP80579 Integrated Processor ("Silicon on Chip") however, this register is called the HECBASE register and is located in at B0:D0:F0-CEh. Programmers must be aware that the memory range and its control registers change between chipsets.

Additionally, since the memory address range occupied by the PCIe configuration registers will mask any DRAM located at the same address range (see the section below for more details), most Intel® chipsets permit the memory address range occupied by the PCIe configuration registers to consume less than the full 256MB range. These chipsets permit BIOS to program the PCIe* configuration register memory range to occupy 256MB, 128MB, or 64MB of memory address space. This is done by simply limiting the number of PCIe busses allowable in a given system.

For example, in the Intel® PM965 chipset, if configuration software detects less than 64 PCI/PCIe busses within the system (very common), then the software can program the chipset to only allow 64MB of PCIe Configuration memory space. Accesses above 64MB will be sent to DRAM.

PCIe* Memory Range and How It Impacts Available DRAM in 4GB Systems Running 32-bit Operating Systems

It is important to realize that since the PCIe* configuration register memory space occupies a chunk of the memory address map, any DRAM located in this space will essentially be lost to the system. This often happens in systems with 4GB of DRAM running 32 bit Operating Systems.

For example, Windows* XP uses only 32 bits when accessing memory space. This permits accesses to memory address region from 0-4GB. If the system is fully populated with 4GB of DRAM, Windows XP* could theoretically access all of the 4GB of DRAM. However, since the PCIe* configuration registers requires some of this memory address space, up to 256MB of DRAM will be "lost" to the operating system since the PCIe configuration register address space overlaps the same memory region used by the DRAM. This is why most Intel® chipsets contain a register to change the memory address space used by the PCIe configuration registers to something smaller than 256MB (typically 128MB or 64MB).

Note that 64 bit operating systems do not have this problem. Since they use 64 bits for addresses, they can access memory ranges above 4GB. Intel chipsets typically contain logic that can "remap" the DRAM overlapping the PCIe configuration register space to memory addresses above 4GB. Details of this are beyond the scope of this paper.

Conversion Formulas – Code

The code in Figure 5 written in C shows how to:

- Create the Memory Address needed to access a PCIe* configuration register given the Bus Number, Device Number, Function, and Register Offset
- Calculate the Bus Number, Device Number, Function, and Register offset when provided with a memory address.

Figure 5. Sample Code Demonstrating How to Access PCIe* Configuration Registers

```
//***********************************************************************
unsigned long  PCIeBase = 0xF0000000UL;
unsigned long  FinalAddress;
unsigned long  Bus = 0;
unsigned long  Device = 0;
unsigned long  Function = 0;
unsigned long  Register = 0;


//***********************************************************************
void Convert_to_Memory()
{
   FinalAddress = PCIeBase          +
                  (Bus*0x100000UL)  +
                  (Device*0x8000)   +
                  (Function*0x1000) +
                  Register;
}
//***********************************************************************
void Convert_to_Register()
{
   Bus =      (FinalAddress-PCIeBase) / (0x100000UL);
   Device =   (FinalAddress-PCIeBase - (Bus*0x100000UL)) / (0x8000);
   Function = (FinalAddress-PCIeBase - (Bus*0x100000UL) -
              (Device*0x8000)) / (0x1000);
   Register = (FinalAddress) & (0x00000FFFUL);
}
//***********************************************************************
```

Conclusion

Since most modern computers implement PCI Express* devices, it is imperative for programmers to understand how to access the PCIe* configuration registers contained within these devices. Although the original access method using I/O locations CF8h and CFCh will still work, that method will only access the first 255 configuration registers of a device. To access the remaining 4096 registers, memory accesses must be made to the PCI Express* Configuration Register memory block that is contained in memory address space. The Convert_to_Memory routine shown in Figure 5 can be used to calculate the final memory address to be used when given a device's bus number, device number, function number, and register offset. The Convert_to_Register routine can be used to calculate the bus, device, function and register offset when provided with a memory address within the PCI Express* Configuration Register memory range.

Authors

Sam Fleming is a Technical Marketing Engineer with the Applications Design-In Center at Intel in Folsom, California.

Acronyms

MCH: Memory Controller Hub. This is the generic name given to the specific integrated circuit that contains the memory interface in a given chipset.

GMCH: Graphics Memory Controller Hub. Same definition as above, but applied to MCH's that also contain an integrated graphics controller.

North Bridge: A generic term that refers to the MCH or GMCH.

PCI: Peripheral Component Interconnect. A standardized bus specification. See http://www.pcisig.com/specifications/ for more details.

PCIe*: PCI Express*. Another standardized bus. Although the hardware interface is completely different from PCI, the configuration register implementation is very similar.

321090