Contents

[1. PCI Configuration Space: - 2](#_Toc122904085)

[1.1. Configuration header: - 3](#_Toc122904086)

[1.2. 192 bytes (40h – FFh):- 5](#_Toc122904087)

[2. Permits Plug-And-Play: - 5](#_Toc122904088)

[3. Types of configuration space: - 5](#_Toc122904089)

[4. Access to Configuration Space: - 6](#_Toc122904090)

[4.1. Configuration of PCI compatible configuration space: - 6](#_Toc122904091)

[4.2. I/O Port CONFIG\_ADDRESS (CF8h): - 6](#_Toc122904092)

[4.3. I/O Port CONFIG\_DATA (CFCh): - 7](#_Toc122904093)

[4.4. PCI Device Identification: - 7](#_Toc122904094)

[5. Configuration Space register details: - 8](#_Toc122904095)

[5.1. Command register: - 9](#_Toc122904096)

[5.2. Status Register: - 9](#_Toc122904097)

[5.3. Class Code register: - 10](#_Toc122904098)

[5.4. Header Type register: - 10](#_Toc122904099)

[5.4.1 Header Type 0x0 11](#_Toc122904100)

[5.4.2 Header Type 0x1 (PCI-to-PCI bridge) 12](#_Toc122904101)

[5.4.3 Header Type 0x2 (PCI-to-CardBus bridge) 13](#_Toc122904102)

[5.5. BIST register: - 14](#_Toc122904103)

[5.6. Interrupt Line register: - 14](#_Toc122904104)

[5.7. Capability Pointer Register: - 15](#_Toc122904105)

# PCI Configuration Space: -

PCI devices have a set of registers referred to as configuration space. These registers are then mapped to memory locations such as the I/O Address Space of the CPU. On PCI Express buses, this configuration space may be referred to as the the Extended Configuration Space. Every PCI device in the system, including the PCI-PCI bridges has a configuration space.

Size of configuration space in PCI – 256 bytes

Size of configuration space in PCIe – 4096 bytes

In PCIe, out of 4k bytes, the first 256 bytes of configuration space will have general information of the device, remaining bytes are used for class specific information. This initial 256 bytes is same as that of PCI for backward compatibility. Every PCIe device has its configuration space mapped to memory.

In PCI, configuration space of 256 bytes is addressed by using 8-bit PCI bus, 5-bit device and 3-bit function numbers for the device, it is commonly known as bus/device/funtion or BDF.

Again these 256 bytes have two sub-categories: -

1. First 64 bytes (00h – 3Fh) is called Standard Configuration Header. I had PCI ID i.e., Vendor ID and Product ID registers, to identify device.
2. Remaining 192 byes ( 40h – FFh) represent user-definable configuration space, such as information specific to a PC card. I also have capabilities supported by device.

Note - PCI Express (PCIe) extends PCI compatible configuration space of size 256 bytes to 4K.

Diagram

Description automatically generated

Figure – 1

## Configuration header: -

Out of 256 bytes configuration space, first 64 bytes (00h – 3Fh) of configuration space are standardized, also called as “Configuration Header”.

In figure – 1, in 64 bytes configuration header, the black ones are read only and white ones are writable registers where the BSP code writes down the address details, configure this device and map it to a particular address space. Address ranges are assigned at boot time.

We can in 64 bytes configuration header, there resides PCI ID i.e., Vendor ID and Device ID registers, to identify the device.

If an Endpoint has 7 functionalities for example, then there will be seven 256 bytes configuration space regions.

Table

Description automatically generated

Figure – 2: PCI Configuration Header

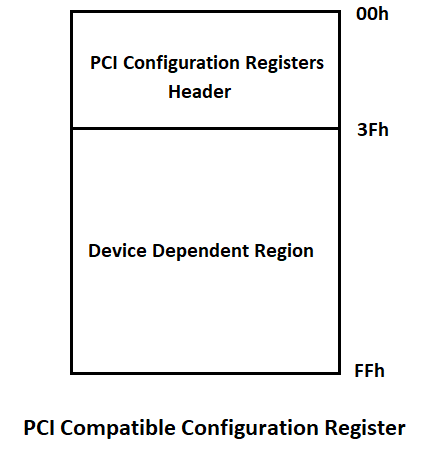


Figure - 3

## 192 bytes (40h – FFh):-

The remaining 192 bytes (40h – FFh) out of 256 bytes represent user-definable configuration space, such as the information specific to PC card for use by its accompanying software. These 192 bytes defines capabilities using various registers.

It consists of device specific information as per PCI-SIG documentation.

# Permits Plug-And-Play: -

1. Base address registers allow an agent to be mapped dynamically into memory or I/O space.
2. A programmable interrupt-line setting allows a software driver to program a PC card with an IRQ upon power up.

# Types of configuration space: -

There are two types of configurations spaces: -

1. Type 0 Configuration Space – for Endpoints
2. Type 1 Configuration Space – for Root Complexes and Switches

# Access to Configuration Space: -

There are two ways to access PCI comaptible 256 bytes configuration space registers:

* Port I/O or Memory-mapped I/O

But bytes in configuration space other than first 256 bytes i.e., 257th byte to 4096tH byte can only be accessed using “Memory-mapped I/O”.

Access to Configuration Space will be done via Port I/O in below situations: -

1. When in Real Mode access to 32-bit memory space is limited.
2. If PCIEXBAR register is not configured, then access happens in Port I/O, as PCIEXBAR register enables MMIO.

## Configuration of PCI compatible configuration space: -

PCI configuration transaction is generated by CPU/BIOS for 256 bytes PCI compatible registers. CPU/BIOS program configures the registers to set device and system parameters.

Compatible PCI is configured using PORT I/O Address/Data pair (CONFIG\_ADDRESS, CONFIG\_DATA)

Two 32 bit I/O locations are used to generate configuration transactions: -

* CF8h (CONFIG\_ADDRESS)
* CFCh (CONFIG\_DATA)

They are mentioned as CF8/CFC in Intel chipset datasheets.

## I/O Port CONFIG\_ADDRESS (CF8h): -

Table

Description automatically generated

Figure - 4

* If Bit 31 is set to 1, then all read and writes to CONFIG\_DATA are PCI Configuration transaction.
* Bits 30:24 are read only and must return 0.
* Bits 23:16 select specific bus (upto 256 buses)
* Bits 15:11 select a device in the specific bus (upto 32 devices)
* Bits 10:8 select specific function of a device (upto 8 devices)
* Bits 7:0 select an offset within the configuration space (256 bytes max), DWORD aligned as bits 1:0 are hard coded to 0.
* Addresses are given in B/D/F format, Offset. It is also written as B:D:F, Offset
* We should visualize PCI configuration header , when we talk about access to specific register number / offset in CONFIG\_ADDRESS

## I/O Port CONFIG\_DATA (CFCh): -

* CONFIG\_DATA can be accessed in DWORD(32 bits), WORD, or BYTE configurations.
* Reads and Writes to CONFIG\_DATA with Bit 31 in CONFIG\_ADDRESS set/enabled results in PCI transaction to the device specified in CONFIG\_ADDRESS.
* PCI spec says that if Bit 31 is not enabled, then the transaction is forwarded out as Port I/O.

## PCI Device Identification: -

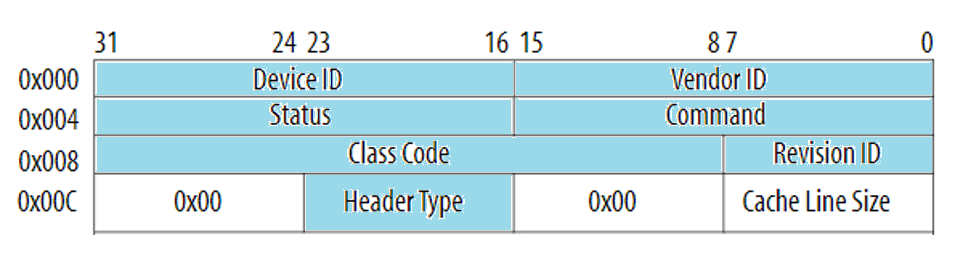


Figure – 5

Below five fields shown in figure – 5 highlighted in sky colour can be used to identify the device and its basic fucntionalities.

1. Vendor ID : - It is used to identify manufacturer of device. Vendor ID is allocated by PCI-SIG.
2. Device ID: - Device ID identifies the device, set by the vendor.
3. Revision ID: - Revision ID is set by the vendor, it is viewed as extension to Device ID.

Ex. – AMD controller is 1022h.

1. Class code: - Class code is used to identify the generic functionality of device.
2. Header type: - Header type identifies what type of header to expect( ex – general, PCI bridge, Cardbus bridge). If Bit – 7 is set (0x80), then it indicates device is a multi-function device.

Header type: 0x0 means Standard header

0x1 means PCI-to-PCI bridge

0x2 means CardBus bridge

# Configuration Space register details: -

Both type 0 and type 1 configurations have a set of common registers in the PCI compatible region (0 to 3Fh). The diagram below shows these common registers and their relative position in the configuration space.

A picture containing chart

Description automatically generated

Figure – 6

The *device ID* and *vendor ID* are read-only registers that uniquely identify the function. The *vendor ID* is assigned by the [PCI-SIG](https://pcisig.com/) and is different for each vendor, but the device ID is set by the vendor to identify the function. The revision ID field is also set by the vendor to identify hardware and/or firmware versions.

## Command register: -

By writing to “Command Register”, the system controls the device, for example allowing the device to access PCI I/O memory.

Command register has Master Enable (bit 2), if this bit is set then an endpoint may issue memory and I/O read and write requests. When a 0 is written to this register, the device is disconnected from the PCI bus for all accesses except Configuration Space access.

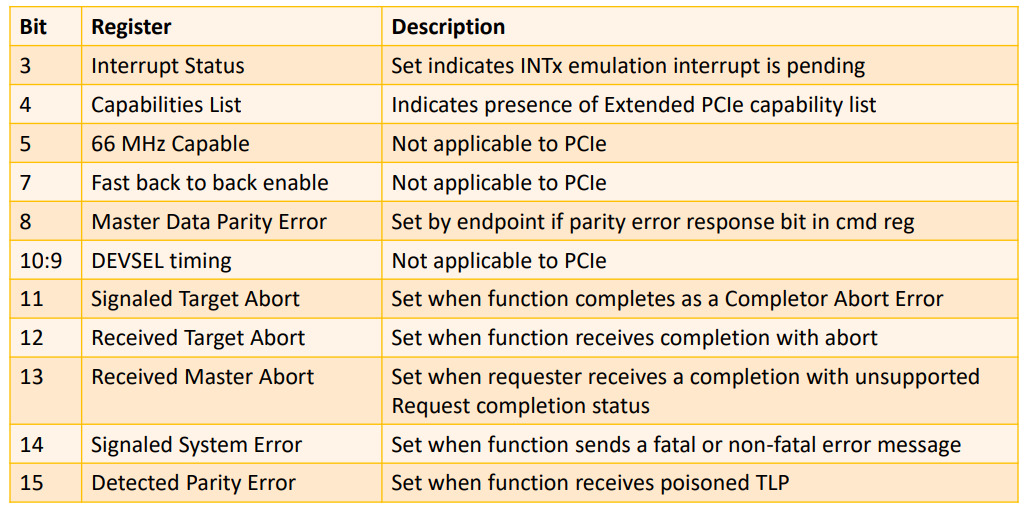
Table

Description automatically generated

*In progress: https://wiki.osdev.org/PCI*

## Status Register: -

This field gives the status of the device with the meaning of the bits of this field set by the standard.



## Class Code register: -

The Class Code register identifies the type of function, with different numbers representing different classes of functionality. For example, a class code of 02h is a network controller or 01h is a mass storage device. These are defined in the PCI Code and ID Assignment Specification*.*

## Header Type register: -

The header type register identifies whether the space is type 0 or type 1. Header Type field is a byte in a register at offset address – 0x0C. Header type field identifies the layout of rest of the header starting at offset position – 0x10.

If bit 7 of this register is set, the device has multiple functions; otherwise, it is a single function device.

|  |  |
| --- | --- |
| Value in “Header Type” field | Meaning |
| 0x0 | General device |
| 0x1 | PCI-to-PCI bridge |
| 0x2 | Card-bus bridge |

### 5.4.1 Header Type 0x0

Below table (Figure – xx) describes Header if the Header Type = 0x0

Graphical user interface, application

Description automatically generated with medium confidence

Figure – PCI configuration header if Header Type = 0x0

Table

Description automatically generated

Figure – PCI configuration header if Header Type = 0x0

### 5.4.2 Header Type 0x1 (PCI-to-PCI bridge)

Below table (Figure – yy)  is applicable if the Header Type is 0x1 (PCI-to-PCI bridge).

Graphical user interface, text, application, email

Description automatically generated

Figure – yy:

### 5.4.3 Header Type 0x2 (PCI-to-CardBus bridge)

Below table (Figure – zz) is applicable if the Header Type is 0x2 (PCI-to-CardBus bridge).

Graphical user interface, text, application, email

Description automatically generated

Figure – zz:

## BIST register: -

The BIST register allows control of any built-in-self-test of the function. Bit 7 indicates whether a BIST capability is available, and bit 6 is written to 1 to start the test if available. A result is returned in bits 0 to 3.

## Interrupt Line register: -

The Interrupt Line register is a read-write register that is programmed by the operating system if an interrupt pin is implemented for interrupt routing. The device doesn’t use this value but must provide the register if an interrupt pin is implemented. The Interrupt Pin register is read-only that indicates which legacy interrupts are used (if any). Valid values are 1, 2, 3, and 4 for each of the INTA to INTD legacy interrupt messages.

## Capability Pointer Register: -

The capabilities pointer register indicates an offset, beyond the header registers to further capability register structures. In other words, beyond the PCI registers, the location of other capabilities is not fixed within the configurations space. Instead, capabilities are arranged as a linked list of structures.

The capability pointer register gives the offset for the first capability structure. A popular value would be 40h, the first offset beyond the PCI compatible space—but it need not be.

***In progress:*** [***https://wiki.osdev.org/PCI***](https://wiki.osdev.org/PCI) ***{page No - 9}***

<https://www.linkedin.com/pulse/pci-express-primer-4-configuration-space-simon-southwell/>

**Very good use this: -** https://en.wikipedia.org/wiki/PCI\_configuration\_space