8259A Interrupt Controller on the PC

Cornelis Frank

November 10, 1999

This text was found on the internet. I added just some little things to it. This article won't save you from having to look at the data sheet as well... it's just an explanation of what those guys at Intel are trying to say in the specs, and how the 8259 is used in a PC. :)

1 Introduction & Scope

The 8259A is a wonderful device. It has a number of modes of operation, and some magic priority rotation features. On the PC, however, only the very simplest mode is used.
This discussion is limited to the way in which the 8259A is used on the PC.
The function of the 8259A is to take up to eight interrupt sources, and feed any active interrupts, one at a time, to the CPU. The processor can individually mask off interrupt sources. The 8259A has a priority mechanism, so that lower priority interrupt sources do not interrupt the CPU while it is servicing a higher priority interrupt – but a higher priority interrupt will be passed on to the processor.

2 The Jargon

An interrupt source, fed into an 8259A, is known as an IRQ – Interrupt Request. An 8259A has 8 IRQ inputs.
The 8 IRQ inputs are fed into an 8 bit Interrupt Request Register (IRR), via some ”rising edge” detection logic (8 bit Edge Sense Register – ESR).
The 8259A can be told to mask off any of the IRQ. The 8259A has an 8 bit Interrupt Mask Register (IMR). A one bit in the IMR masks off the corresponding IRQ.
To perform its priority arbitration the 8259A has an 8 bit In Service Register (ISR). In the register a bit is set to 1 when the corresponding interrupt has been passed to the CPU, and the CPU has not yet signalled End of Interrupt (EOI)
The CPU interrupt input is known as INTR. The 8259A interrupt output is known as INT, which is connected to the CPU (wait for it) INTR, or to another 8259A's IRQ.
The 8 IRR bits are ANDed with the NOT of the IMR, giving the interrupt request input to the priority arbitration logic. Reading between the lines, there is an INT latch, which is set by the OR of the bits of (IRR AND NOT IMR) higher than the highest priority bit in the ISR.
On an original PC there are 8 possible interrupt sources IRQ0 to IRQ7, fed into one 8259A (I/O address 0x020..0x03F).
On AT's and beyond, there are 16 possible interrupt sources IRQ0 to IRQ15, fed into two 8259A's. One 8259A (known as 0x1, I/O address 0x020..0x03F) is the ”Master” and the other is a ”Slave”

(known as 0x2, I/O address 0x0A0..0x0BF). Only the Master's INT is connected to the CPU's INTR. The Slave's INT is connected to the Master's IRQ2.

3 The Mechanisms

The PC sets the 8259A into:

- Edge Triggered Interrupts
- Cascaded (on AT and later) ; Single (on earlier machines)
- Not Special Fully Nested (to do with Slave 8259A, see below)
- Not Buffered Normal EOI (Not Automatic EOI on INTA)

With this in mind, we will start with the simple cases, and work up.

3.1 One 8259A, All IRQ Unmasked, No Interrupts In Service and None Active.

So we start from the simplest possible quiescent state. The sequence of actions is as follows:

1. The ESR, ISR, IRR and IMR are all zero.
2. IRQ3 becomes active (goes to 1)
3. B3 of the ESR is set to 1
4. B3 of the IRR is set to 1
5. B3 of the IMR is 0, so the IRR B3 is passed to the priority arbitration logic.
6. All bits of the ISR are 0 (no interrupts are in service), so the priority arbitration logic sets the INT latch – so the INT output is set active.
7. Eventually the CPU issues the first of two INTA cycles. The contents of the IRR are frozen. The 8279A selects the highest priority IRR (B3) and sets the corresponding ISR (B3).
8. Setting B3 of the ISR clears B3 of the ESR.
9. The CPU issues the second of two INTA cycles. The 8279A issues the interrupt vector associated with the highest priority ISR (B3). The contents of the IRR are unfrozen.
10. The INT latch is cleared – so the INT output is set inactive.
11. B3 of the IRR is set to 0 (IRR is unfrozen and B3 of ESR is zero).
12. At some time in the future, the CPU issues an EOI command, which clears B3 of the ISR.

IRQ3 can remain active beyond step 10, without generating any further interrupts – because B3 of IRR has been cleared. To produce another interrupt requires IRQ3 to go inactive (0), and then active (1) again.

3.2 Meaning of ”Edge Triggered Interrupt Mode”

The behaviour of the ESR, IRR and ISR described above is what happens in the famous Edge Triggered Interrupt Mode.
The purpose is to allow for IRQ signals to be short down/up pulses. When the 8259A is reset the ESR is set to zero. An upward transition of the IRQ sets the corresponding ESR bit to 1, which allows the IRQ state to be copied to the IRR – provoking the interrupt. When the interrupt is acknowledged the ISR bit is set, which resets the ESR bit, which forces the IRR bit to zero – irrespective of the IRQ. So even if IRQ is still 1 when the ISR bit is cleared, at End of Interrupt, no further interrupts will be generated.

3.3 What Happens if IRQ Changes with the Interrupt is In Service

It is clear what happens if IRQ does not do any further down/up transitions until after EOI. It is OK for IRQ to go down before EOI, but going up again is not explicitly described in the manuals. If a down/up IRQ transition cannot be prevented before EOI, then it can be (reasonably safely) assumed that this will generate a further interrupt after EOI – provided the IRQ is still up (active, 1) at EOI. Multiple down/up transitions can be assumed to have the same effect.
What happens if there are one or more down/up IRQ transitions followed by a final down transition before EOI, is also undocumented. I guess that this has no effect. The corresponding IRR bit will follow the IRQ, but this may be expected to have no effect on the (supposed) INT latch, because the ISR bit prevents it.
Obviously, it would be safer to ensure that IRQ does not go down and then up again before EOI (just down is OK). If this is not possible, then I believe the given assumptions to be reasonable – perhaps MEJ’s boys could help us !

3.4 Master and Slave Handling

The PC does not use ”Special Fully Nested Mode”. What this means is that once one of the Slave’s interrupts is In Service it takes precedence over all other Slave interrupts.
Slave interrupts are generally indistinguishable from Master interrupts. The only tricky bit is that an EOI must be sent to both Master and Slave. EOI should be sent to the Slave first – to allow any lower priority interrupts on the slave to assert themselves. The EOI to the Master then allows any lower or equal priority interrupts to assert themselves.

3.5 Fiddling with Interrupt Masking

Clearing masking or unmasking an interrupt when all is quiet (no IRQ, IRR or ISR) is trivially OK, and produces no side effects.
If an interrupt was masked and is unmasked, then any ”pending” IRQ will immediately take effect. According to the 8259A diagram published by Intel, the IMR mask gates the IRR bits into the interrupt priority resolution logic. Masking and unmasking an interrupt while its IRR is active is equivalent, as far as the interrupt priority resolution logic is concerned, to the IRQ coming and going. The effect can be seen to be the same as in 3.3 above.
There is doubt and uncertainty about what happens if IRQ bounces up and down while ISR is set. To avoid difficulties it would be reasonable (unless the Intel diagram I am working from is entirely wrong) to mask off the interrupt until EOI, and then unmask it again – assuming that there is some other way of detecting and dealing with the reasons for IRQ changing.

4 PIC registers on a PC

A PC has two programmable interrupt controllers. One at address `0x0020` and the other at `0x00A0`. The addresses of the registers of the PIC 1 are given in the following table [1].

addr	read write	function
0020	-W	PIC initialization command word ICW1
0020	-W	PIC output control word OCW2
0020	-W	PIC output control word OCW3
0020	R-	PIC interrupt request/in-service registers after OCW3
0021	-W	PIC ICW2,ICW3,ICW4 immed after ICW1 to 0020
0021	RW	PIC master interrupt mask register OCW1

For the second PIC, just replace every `0020` and `0021` respectively with `00A0` and `00A1`. When you read from `0020` after the `OCW3` you get two registers:

request register:
bit 7-0 = 0 no active request for the corresponding int. line
= 1 active request for corresponding interrupt line
in-service register:
bit 7-0 = 0 corresponding line not currently being serviced
= 1 corresponding int. line currently being serviced

Next tables give more information on the setup of the registers.

Bitfields for PIC initialization command word ICW1	
Bit(s)	**Description**
7-5	0 (only used in 8080/8085 mode)
4	ICW1 is being issued
3	(LTIM) =0 edge triggered mode =1 level triggered mode
2	interrupt vector size =0 successive interrupt vectors use 8 bytes (8080/8085) =1 successive interrupt vectors use 4 bytes (80x86)
1	(SNGL) =0 cascade mode =1 single mode, no ICW3 needed
0	ICW4 needed

Bitfields for PIC initialization command word ICW2	
Bit(s)	**Description**
7-3	address lines A0-A3 of base vector address for PIC
2-0	reserved

Bitfields for PIC initialization command word ICW3	
Bit(s)	**Description**
7-0	=0 slave controller not attached to corresponding interrupt pin =1 slave controller attached to corresponding interrupt pin

[1] Source: Ralf Brown's Interrupt List

Bitfields for PIC initialization command word ICW4	
Bit(s)	**Description**
7-5	reserved (0)
4	running in special fully-nested mode
3-2	mode 0x nonbuffered mode 10 buffered mode/slave 11 buffered mode/master
1	Auto EOI
0	=0 8085 mode =1 8086/8088 mode

Bitfields for PIC output control word OCW1	
Bit(s)	**Description**
7	disable IRQ7 (parallel printer interrupt)
6	disable IRQ6 (diskette interrupt)
5	disable IRQ5 (fixed disk interrupt)
4	disable IRQ4 (serial port 1 interrupt)
3	disable IRQ3 (serial port 2 interrupt)
2	disable IRQ2 (video interrupt)
1	disable IRQ1 (keyboard, mouse, RTC interrupt)
0	disable IRQ0 (timer interrupt)

Bitfields for PIC output control word OCW2	
Bit(s)	**Description**
7-5	operation 000 rotate in auto EOI mode (clear) 001 (WORD_A) nonspecific EOI 010 (WORD_H) no operation 011 (WORD_B) specific EOI 100 (WORD_F) rotate in auto EOI mode (set) 101 (WORD_C) rotate on nonspecific EOI command 110 (WORD_E) set priority command 111 (WORD_D) rotate on specific EOI command
4-3	reserved (00 - signals OCW2)
2-0	interrupt request to which the command applies (only used by WORD_B, WORD_D, and WORD_E)

Bitfields for PIC output control word OCW3	
Bit(s)	**Description**
7	reserved (0)
6-5	special mask 0x no operation 10 reset special mask 11 set special mask mode
4-3	reserved (01 - signals OCW3)
2	poll command
1-0	function 0x no operation 10 read interrupt request register on next read from 0020h 11 read interrupt in-service register on next read from 0020h

Note:

The special mask mode permits all other interrupts (even those with lower priority) to be processed while an interrupt is already in service, but will not re-issue an interrupt for a particular IRQ while it remains in service.

4.1 PIC 2

The registers of PIC 2 are the same as with PIC 1 except for OCW1 which is given in the following table.

Bitfields for PIC2 output control word OCW1	
Bit(s)	**Description**
7	disable IRQ15 (reserved)
6	disable IRQ14 (fixed disk interrupt)
5	disable IRQ13 (coprocessor exception interrupt)
4	disable IRQ12 (mouse interrupt)
3	disable IRQ11 (reserved)
2	disable IRQ10 (reserved)
1	disable IRQ9 (redirect cascade)
0	disable IRQ8 (real-time clock interrupt)