Прерывания от внешних устройств в системе x86. Часть 2. Опции загрузки ядра Linux

* [Настройка Linux](https://habr.com/ru/hub/linux/),
* [Системное программирование](https://habr.com/ru/hub/system_programming/)

В [предыдущей части](https://habr.com/post/430548/) мы рассмотрели эволюцию доставки прерываний от устройств в x86 системах (PIC → APIC → MSI), общую теорию и все необходимые термины.   
  
В этой практической части мы рассмотрим как откатиться к использованию устаревших методов доставки прерываний в Linux, а именно рассмотрим опции загрузки ядра:

* pci=nomsi
* noapic
* nolapic

Также мы посмотрим на порядок, в котором ОС смотрит таблицы роутинга прерываний (ACPI/MPtable/$PIR) и какое влияние на него окажет добавление опций загрузки:

* pci=noacpi
* acpi=noirq
* acpi=off

Возможно вы пробовали комбинации из всех этих опций, когда какое-либо устройство не работало из-за проблемы с прерываниями. Разберём, что именно они делают и как они меняют вывод /proc/interrupts.

Загрузка без дополнительных опций

Смотреть прерывания в данной статье мы будем на кастомной плате с Intel Haswell i7 с чипсетом lynxPoint-LP на которой запущен [coreboot](https://www.coreboot.org/).  
  
Информацию о прерываниях мы будем выводить через команду 

cat /proc/interrupts

Вывод при загрузке без дополнительных опций:

CPU0 CPU1 CPU2 CPU3

0: 15 0 0 0 IO-APIC-edge timer

1: 0 1 0 1 IO-APIC-edge i8042

8: 0 0 0 1 IO-APIC-edge rtc0

9: 0 0 0 0 IO-APIC-fasteoi acpi

12: 0 0 0 1 IO-APIC-edge

23: 16 247 7 10 IO-APIC-fasteoi ehci\_hcd:usb1

56: 0 0 0 0 PCI-MSI-edge aerdrv,PCIe PME

57: 0 0 0 0 PCI-MSI-edge aerdrv,PCIe PME

58: 0 0 0 0 PCI-MSI-edge aerdrv,PCIe PME

59: 0 0 0 0 PCI-MSI-edge aerdrv,PCIe PME

60: 0 0 0 0 PCI-MSI-edge aerdrv,PCIe PME

61: 0 0 0 0 PCI-MSI-edge aerdrv,PCIe PME

62: 3118 1984 972 3454 PCI-MSI-edge ahci

63: 1 0 0 0 PCI-MSI-edge eth59

64: 2095 57 4 832 PCI-MSI-edge eth59-rx-0

65: 6 18 1 1309 PCI-MSI-edge eth59-rx-1

66: 13 512 2 1 PCI-MSI-edge eth59-rx-2

67: 10 61 232 2 PCI-MSI-edge eth59-rx-3

68: 169 0 0 0 PCI-MSI-edge eth59-tx-0

69: 14 14 4 205 PCI-MSI-edge eth59-tx-1

70: 11 491 3 0 PCI-MSI-edge eth59-tx-2

71: 20 19 134 50 PCI-MSI-edge eth59-tx-3

72: 0 0 0 0 PCI-MSI-edge eth58

73: 2 1 0 152 PCI-MSI-edge eth58-rx-0

74: 3 150 2 0 PCI-MSI-edge eth58-rx-1

75: 2 34 117 2 PCI-MSI-edge eth58-rx-2

76: 153 0 2 0 PCI-MSI-edge eth58-rx-3

77: 4 0 2 149 PCI-MSI-edge eth58-tx-0

78: 4 149 2 0 PCI-MSI-edge eth58-tx-1

79: 4 0 117 34 PCI-MSI-edge eth58-tx-2

80: 153 0 2 0 PCI-MSI-edge eth58-tx-3

81: 66 106 2 101 PCI-MSI-edge snd\_hda\_intel

82: 928 5657 262 224 PCI-MSI-edge i915

83: 545 56 32 15 PCI-MSI-edge snd\_hda\_intel

NMI: 0 0 0 0 Non-maskable interrupts

LOC: 4193 3644 3326 3499 Local timer interrupts

SPU: 0 0 0 0 Spurious interrupts

PMI: 0 0 0 0 Performance monitoring interrupts

IWI: 290 233 590 111 IRQ work interrupts

RTR: 3 0 0 0 APIC ICR read retries

RES: 1339 2163 2404 1946 Rescheduling interrupts

CAL: 607 537 475 559 Function call interrupts

TLB: 163 202 164 251 TLB shootdowns

TRM: 48 48 48 48 Thermal event interrupts

THR: 0 0 0 0 Threshold APIC interrupts

MCE: 0 0 0 0 Machine check exceptions

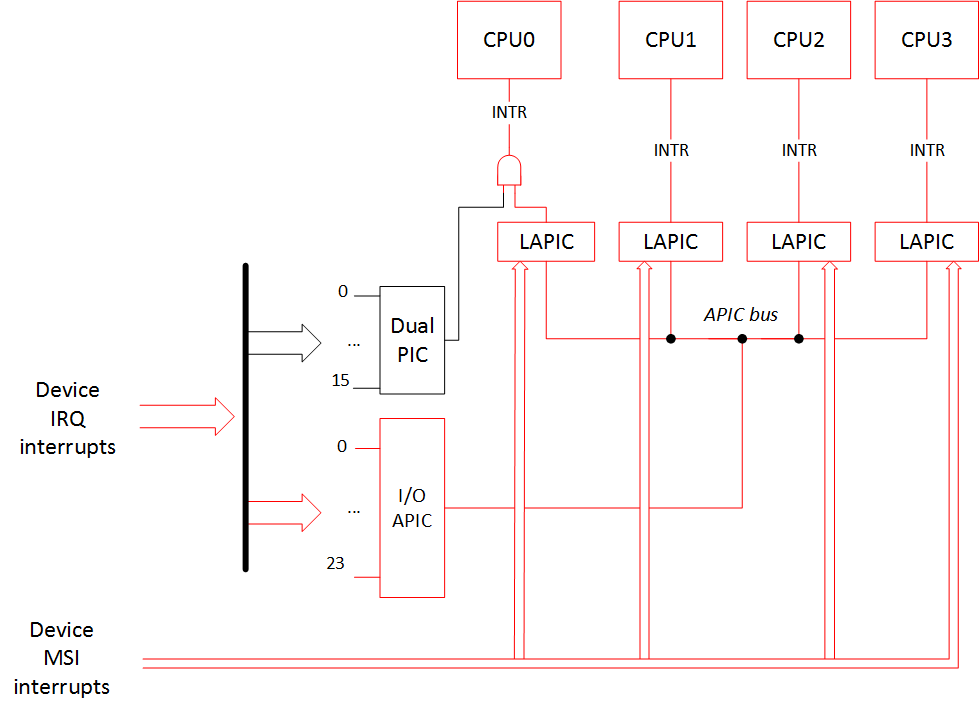
MCP: 3 3 3 3 Machine check polls

ERR: 0

MIS: 0

Файл /proc/interrupts предоставляет таблицу о количестве прерываний на каждом из процессоров в следующем виде:

* Первая колонка: номер прерывания
* Колонки CPUx: счётчики прерываний на каждом из процессоров
* Следующая колонка: вид прерывания:
  + IO-APIC-edge — прерывание по фронту на контроллер I/O APIC
  + IO-APIC-fasteoi — прерывание по уровню на контроллер I/O APIC
  + PCI-MSI-edge — MSI прерывание
  + XT-PIC-XT-PIC — прерывание на PIC контроллер (увидим позже)
* Последняя колонка: устройство, ассоциированное с данным прерыванием

Так вот, как и положено в современной системе, для устройств и драйверов, поддерживающих прерывания MSI/MSI-X, используются именно они. Остальные прерывания роутятся через I/O APIC.  
  
Упрощённо схему роутинга прерываний можно нарисовать так (красным помечены активные пути, чёрным неиспользуемые).  
  
  
  
Поддержка MSI/MSI-X устройством должна быть обозначена как соответствующая Capability в его конфигурационном пространстве PCI.  
  
В подтверждении приведём небольшой фрагмент вывода lspci для устройств, для которых обозначено, что они используют MSI/MSI-X. В нашем случае это SATA контроллер (прерывание ahci), 2 ethernet контроллера (прерывания eth58\* и eth59\*), графический контроллер (i915) и 2 контроллера HD Audio (snd\_hda\_intel).

lspci -v

00:02.0 VGA compatible controller: Intel Corporation Haswell-ULT Integrated Graphics Controller (rev 09) (prog-if 00 [VGA controller])

...

Capabilities: [90] MSI: Enable+ Count=1/1 Maskable- 64bit-

Capabilities: [d0] Power Management version 2

Capabilities: [a4] PCI Advanced Features

Kernel driver in use: i915

00:03.0 Audio device: Intel Corporation Haswell-ULT HD Audio Controller (rev 09

...

Capabilities: [60] MSI: Enable+ Count=1/1 Maskable- 64bit-

Capabilities: [70] Express Root Complex Integrated Endpoint, MSI 00

Kernel driver in use: snd\_hda\_intel

00:1b.0 Audio device: Intel Corporation 8 Series HD Audio Controller (rev 04)

...

Capabilities: [60] MSI: Enable+ Count=1/1 Maskable- 64bit+

Capabilities: [70] Express Root Complex Integrated Endpoint, MSI 00

Capabilities: [100] Virtual Channel

Kernel driver in use: snd\_hda\_intel

00:1f.2 SATA controller: Intel Corporation 8 Series SATA Controller 1 [AHCI mode] (rev 04) (prog-if 01 [AHCI 1.0])

...

Capabilities: [80] MSI: Enable+ Count=1/1 Maskable- 64bit-

Capabilities: [70] Power Management version 3

Capabilities: [a8] SATA HBA v1.0

Kernel driver in use: ahci

05:00.0 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)

...

Capabilities: [50] MSI: Enable- Count=1/1 Maskable+ 64bit+

Capabilities: [70] MSI-X: Enable+ Count=10 Masked-

Capabilities: [a0] Express Endpoint, MSI 00

Kernel driver in use: igb

05:00.1 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)

...

Capabilities: [50] MSI: Enable- Count=1/1 Maskable+ 64bit+

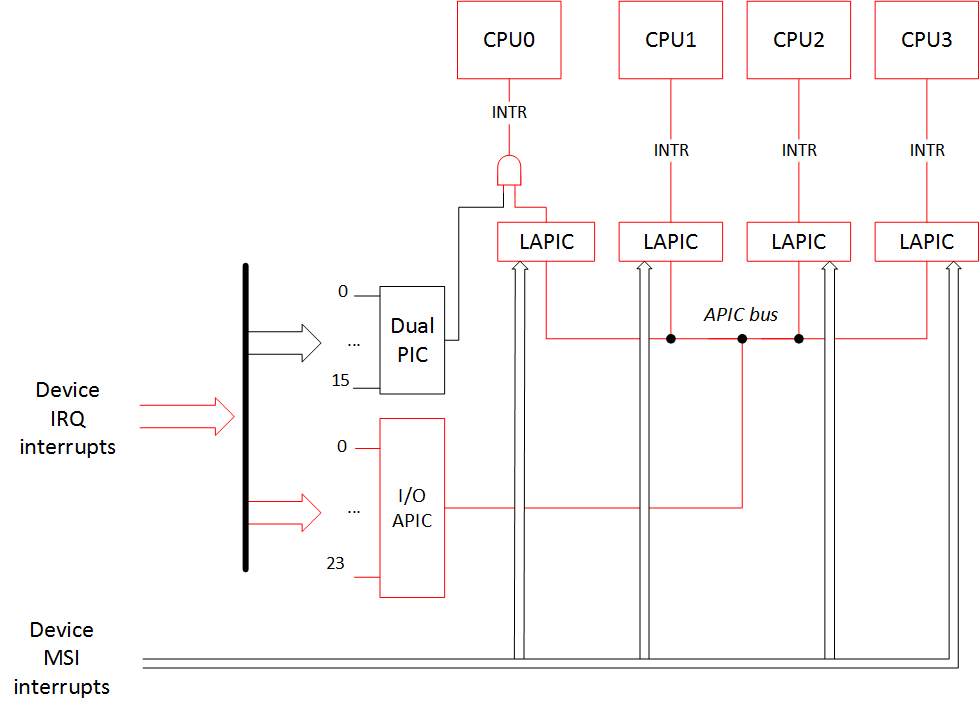
Capabilities: [70] MSI-X: Enable+ Count=10 Masked-

Capabilities: [a0] Express Endpoint, MSI 00

Kernel driver in use: igb

Как мы видим, у этих устройств присутствует строка либо «MSI: Enable+», либо «MSI-X: Enable+»  
  
Начнём деградировать систему. Для начала загрузимся с опцией pci=nomsi.

pci=nomsi

Благодаря этой опции MSI прерывания станут IO-APIC/XT-PIC в зависимости от используемого контроллера прерываний  
  
В данном случае у нас всё ещё приоритетный контроллер прерываний APIC, так что картина будет такая:  
  
  
  
Вывод /proc/interrupts:

CPU0 CPU1 CPU2 CPU3

0: 15 0 0 0 IO-APIC-edge timer

1: 0 1 0 1 IO-APIC-edge i8042

8: 0 0 1 0 IO-APIC-edge rtc0

9: 0 0 0 0 IO-APIC-fasteoi acpi

12: 0 0 0 1 IO-APIC-edge

16: 1314 5625 342 555 IO-APIC-fasteoi i915, snd\_hda\_intel, eth59

17: 5 0 1 34 IO-APIC-fasteoi eth58

21: 2882 2558 963 2088 IO-APIC-fasteoi ahci

22: 26 81 2 170 IO-APIC-fasteoi snd\_hda\_intel

23: 23 369 8 8 IO-APIC-fasteoi ehci\_hcd:usb1

NMI: 0 0 0 0 Non-maskable interrupts

LOC: 3011 3331 2435 2617 Local timer interrupts

SPU: 0 0 0 0 Spurious interrupts

PMI: 0 0 0 0 Performance monitoring interrupts

IWI: 197 228 544 85 IRQ work interrupts

RTR: 3 0 0 0 APIC ICR read retries

RES: 1708 2349 1821 1569 Rescheduling interrupts

CAL: 520 554 509 555 Function call interrupts

TLB: 187 181 205 179 TLB shootdowns

TRM: 102 102 102 102 Thermal event interrupts

THR: 0 0 0 0 Threshold APIC interrupts

MCE: 0 0 0 0 Machine check exceptions

MCP: 2 2 2 2 Machine check polls

ERR: 0

MIS: 0

Все прерывания MSI/MSI-X ожидаемо исчезли. Вместо них устройства теперь используют прерывания вида IO-APIC-fasteoi.  
  
Обратим внимание на то, что раньше до включения этой опции у eth58 и eth59 было по 9 прерываний! А сейчас только по одному. Ведь как мы помним, без MSI одной функции PCI доступно только одно прерывание!  
  
Немного информации из dmesg по инициализации ethernet контроллеров:  
  
— загрузка без опции pci=nomsi:

igb: Intel(R) Gigabit Ethernet Network Driver - version 5.0.5-k

igb: Copyright (c) 2007-2013 Intel Corporation.

acpi:acpi\_pci\_irq\_enable: igb 0000:05:00.0: PCI INT A -> GSI 16 (level, low) -> IRQ 16

igb 0000:05:00.0: irq 63 for MSI/MSI-X

igb 0000:05:00.0: irq 64 for MSI/MSI-X

igb 0000:05:00.0: irq 65 for MSI/MSI-X

igb 0000:05:00.0: irq 66 for MSI/MSI-X

igb 0000:05:00.0: irq 67 for MSI/MSI-X

igb 0000:05:00.0: irq 68 for MSI/MSI-X

igb 0000:05:00.0: irq 69 for MSI/MSI-X

igb 0000:05:00.0: irq 70 for MSI/MSI-X

igb 0000:05:00.0: irq 71 for MSI/MSI-X

igb 0000:05:00.0: irq 63 for MSI/MSI-X

igb 0000:05:00.0: irq 64 for MSI/MSI-X

igb 0000:05:00.0: irq 65 for MSI/MSI-X

igb 0000:05:00.0: irq 66 for MSI/MSI-X

igb 0000:05:00.0: irq 67 for MSI/MSI-X

igb 0000:05:00.0: irq 68 for MSI/MSI-X

igb 0000:05:00.0: irq 69 for MSI/MSI-X

igb 0000:05:00.0: irq 70 for MSI/MSI-X

igb 0000:05:00.0: irq 71 for MSI/MSI-X

igb 0000:05:00.0: added PHC on eth0

igb 0000:05:00.0: Intel(R) Gigabit Ethernet Network Connection

igb 0000:05:00.0: eth0: (PCIe:5.0Gb/s:Width x1) 00:15:d5:03:00:2a

igb 0000:05:00.0: eth0: PBA No: 106300-000

igb 0000:05:00.0: Using MSI-X interrupts. 4 rx queue(s), 4 tx queue(s)

acpi:acpi\_pci\_irq\_enable: igb 0000:05:00.1: PCI INT B -> GSI 17 (level, low) -> IRQ 17

igb 0000:05:00.1: irq 72 for MSI/MSI-X

igb 0000:05:00.1: irq 73 for MSI/MSI-X

igb 0000:05:00.1: irq 74 for MSI/MSI-X

igb 0000:05:00.1: irq 75 for MSI/MSI-X

igb 0000:05:00.1: irq 76 for MSI/MSI-X

igb 0000:05:00.1: irq 77 for MSI/MSI-X

igb 0000:05:00.1: irq 78 for MSI/MSI-X

igb 0000:05:00.1: irq 79 for MSI/MSI-X

igb 0000:05:00.1: irq 80 for MSI/MSI-X

igb 0000:05:00.1: irq 72 for MSI/MSI-X

igb 0000:05:00.1: irq 73 for MSI/MSI-X

igb 0000:05:00.1: irq 74 for MSI/MSI-X

igb 0000:05:00.1: irq 75 for MSI/MSI-X

igb 0000:05:00.1: irq 76 for MSI/MSI-X

igb 0000:05:00.1: irq 77 for MSI/MSI-X

igb 0000:05:00.1: irq 78 for MSI/MSI-X

igb 0000:05:00.1: irq 79 for MSI/MSI-X

igb 0000:05:00.1: irq 80 for MSI/MSI-X

igb 0000:05:00.1: added PHC on eth1

igb 0000:05:00.1: Intel(R) Gigabit Ethernet Network Connection

igb 0000:05:00.1: eth1: (PCIe:5.0Gb/s:Width x1) 00:15:d5:03:00:2b

igb 0000:05:00.1: eth1: PBA No: 106300-000

igb 0000:05:00.1: Using MSI-X interrupts. 4 rx queue(s), 4 tx queue(s)

— загрузка с опцией pci=nomsi

igb: Intel(R) Gigabit Ethernet Network Driver - version 5.0.5-k

igb: Copyright (c) 2007-2013 Intel Corporation.

acpi:acpi\_pci\_irq\_enable: igb 0000:05:00.0: PCI INT A -> GSI 16 (level, low) -> IRQ 16

igb 0000:05:00.0: added PHC on eth0

igb 0000:05:00.0: Intel(R) Gigabit Ethernet Network Connection

igb 0000:05:00.0: eth0: (PCIe:5.0Gb/s:Width x1) 00:15:d5:03:00:2a

igb 0000:05:00.0: eth0: PBA No: 106300-000

igb 0000:05:00.0: Using legacy interrupts. 1 rx queue(s), 1 tx queue(s)

acpi:acpi\_pci\_irq\_enable: igb 0000:05:00.1: PCI INT B -> GSI 17 (level, low) -> IRQ 17

igb 0000:05:00.1: added PHC on eth1

igb 0000:05:00.1: Intel(R) Gigabit Ethernet Network Connection

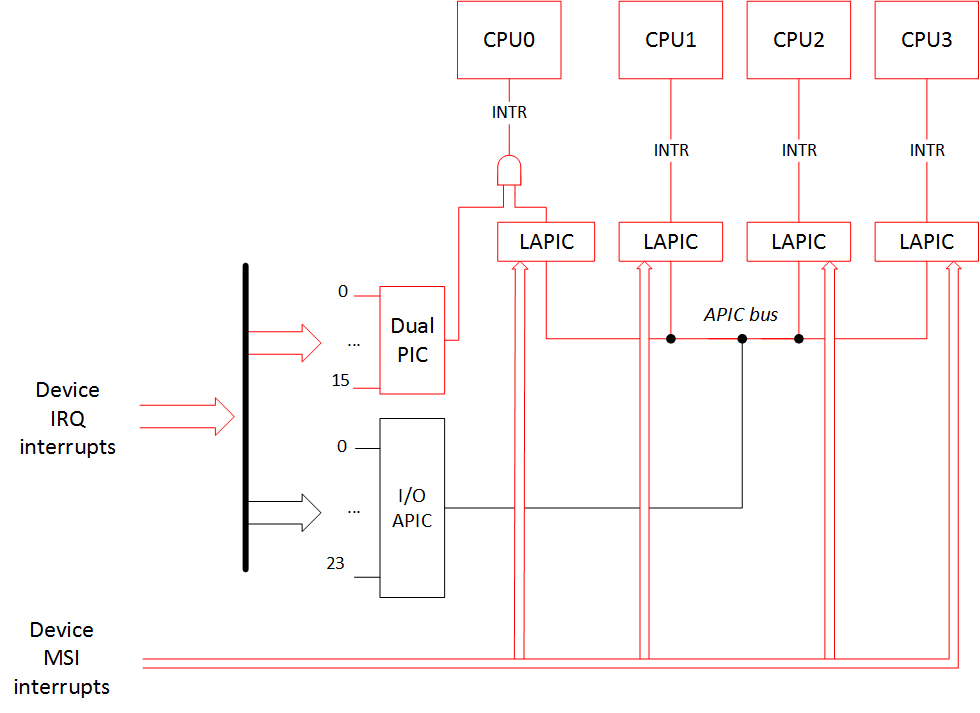
igb 0000:05:00.1: eth1: (PCIe:5.0Gb/s:Width x1) 00:15:d5:03:00:2b

igb 0000:05:00.1: eth1: PBA No: 106300-000

igb 0000:05:00.1: Using legacy interrupts. 1 rx queue(s), 1 tx queue(s)

Из-за уменьшения количества прерываний на устройство, включение данной опции может приводить к существенному ограничению производительности работы драйвера (это без учёта того, что согласно исследованию Intel [Reducing Interrupt Latency Through the Use of Message Signaled Interrupts](https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/msg-signaled-interrupts-paper.pdf)прерывания через MSI в 3 раза быстрее чем через IO-APIC и в 5 раз быстрее чем через PIC).

noapic

Данная опция отключает I/O APIC. MSI прерывания всё ещё могут идти на все CPU, но прерывания от устройств смогут идти только на CPU0, так как PIC связан только с CPU0. Но LAPIC работает и другие CPU могут работать и обрабатывать прерывания.  
  


CPU0 CPU1 CPU2 CPU3

0: 5 0 0 0 XT-PIC-XT-PIC timer

1: 2 0 0 0 XT-PIC-XT-PIC i8042

2: 0 0 0 0 XT-PIC-XT-PIC cascade

8: 1 0 0 0 XT-PIC-XT-PIC rtc0

9: 0 0 0 0 XT-PIC-XT-PIC acpi

12: 172 0 0 0 XT-PIC-XT-PIC ehci\_hcd:usb1

56: 0 0 0 0 PCI-MSI-edge aerdrv, PCIe PME

57: 0 0 0 0 PCI-MSI-edge aerdrv, PCIe PME

58: 0 0 0 0 PCI-MSI-edge aerdrv, PCIe PME

59: 0 0 0 0 PCI-MSI-edge aerdrv, PCIe PME

60: 0 0 0 0 PCI-MSI-edge aerdrv, PCIe PME

61: 0 0 0 0 PCI-MSI-edge aerdrv, PCIe PME

62: 2833 2989 1021 811 PCI-MSI-edge ahci

63: 0 1 0 0 PCI-MSI-edge eth59

64: 301 52 9 3 PCI-MSI-edge eth59-rx-0

65: 12 24 3 178 PCI-MSI-edge eth59-rx-1

66: 14 85 6 2 PCI-MSI-edge eth59-rx-2

67: 17 24 307 1 PCI-MSI-edge eth59-rx-3

68: 70 18 8 10 PCI-MSI-edge eth59-tx-0

69: 7 0 0 23 PCI-MSI-edge eth59-tx-1

70: 15 227 2 2 PCI-MSI-edge eth59-tx-2

71: 18 6 27 2 PCI-MSI-edge eth59-tx-3

72: 0 0 0 0 PCI-MSI-edge eth58

73: 1 0 0 27 PCI-MSI-edge eth58-rx-0

74: 1 22 0 5 PCI-MSI-edge eth58-rx-1

75: 1 0 22 5 PCI-MSI-edge eth58-rx-2

76: 23 0 0 5 PCI-MSI-edge eth58-rx-3

77: 1 0 0 27 PCI-MSI-edge eth58-tx-0

78: 1 22 0 5 PCI-MSI-edge eth58-tx-1

79: 1 0 22 5 PCI-MSI-edge eth58-tx-2

80: 23 0 0 5 PCI-MSI-edge eth58-tx-3

81: 187 17 70 7 PCI-MSI-edge snd\_hda\_intel

82: 698 1647 247 129 PCI-MSI-edge i915

83: 438 135 16 59 PCI-MSI-edge snd\_hda\_intel

NMI: 0 0 0 0 Non-maskable interrupts

LOC: 1975 2499 2245 1474 Local timer interrupts

SPU: 0 0 0 0 Spurious interrupts

PMI: 0 0 0 0 Performance monitoring interrupts

IWI: 132 67 429 91 IRQ work interrupts

RTR: 3 0 0 0 APIC ICR read retries

RES: 1697 2178 1903 1541 Rescheduling interrupts

CAL: 561 496 534 567 Function call interrupts

TLB: 229 254 170 137 TLB shootdowns

TRM: 78 78 78 78 Thermal event interrupts

THR: 0 0 0 0 Threshold APIC interrupts

MCE: 0 0 0 0 Machine check exceptions

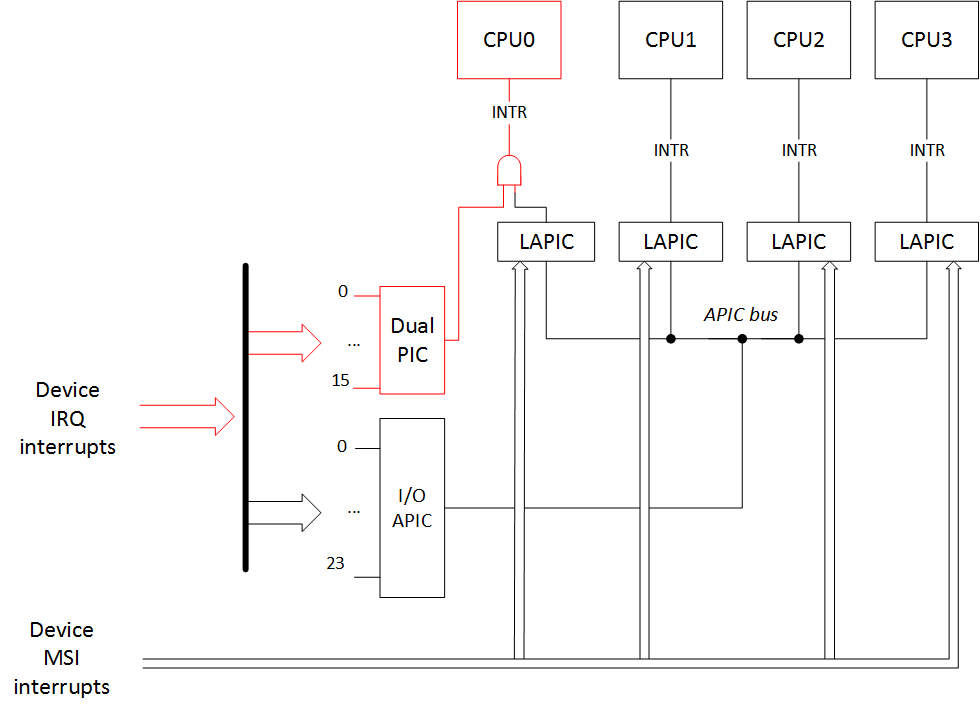
MCP: 2 2 2 2 Machine check polls

ERR: 0

MIS: 0

Как видим, все прерывания IO-APIC-\* превратились в XT-PIC-XT-PIC, причём эти прерывания роутятся только на CPU0. Прерывания MSI остались без изменений и идут на все CPU0-3.

nolapic

Отключает LAPIC. MSI прерывания не могут работать без LAPIC, I/O APIC не может работать без LAPIC. Поэтому все прерывания от устройств будут идти на PIC, а он работает только с CPU0. И без LAPIC остальные CPU даже работать в системе не будут.  
  
  
  
Вывод /proc/interrupts:

CPU0

0: 6416 XT-PIC-XT-PIC timer

1: 2 XT-PIC-XT-PIC i8042

2: 0 XT-PIC-XT-PIC cascade

3: 5067 XT-PIC-XT-PIC aerdrv, aerdrv, PCIe PME, PCIe PME, i915, snd\_hda\_intel, eth59

4: 32 XT-PIC-XT-PIC aerdrv, aerdrv, PCIe PME, PCIe PME, eth58

5: 0 XT-PIC-XT-PIC aerdrv, PCIe PME

6: 0 XT-PIC-XT-PIC aerdrv, PCIe PME

8: 1 XT-PIC-XT-PIC rtc0

9: 0 XT-PIC-XT-PIC acpi

11: 274 XT-PIC-XT-PIC snd\_hda\_intel

12: 202 XT-PIC-XT-PIC ehci\_hcd:usb1

15: 7903 XT-PIC-XT-PIC ahci

NMI: 0 Non-maskable interrupts

LOC: 0 Local timer interrupts

SPU: 0 Spurious interrupts

PMI: 0 Performance monitoring interrupts

IWI: 0 IRQ work interrupts

RTR: 0 APIC ICR read retries

RES: 0 Rescheduling interrupts

CAL: 0 Function call interrupts

TLB: 0 TLB shootdowns

TRM: 0 Thermal event interrupts

THR: 0 Threshold APIC interrupts

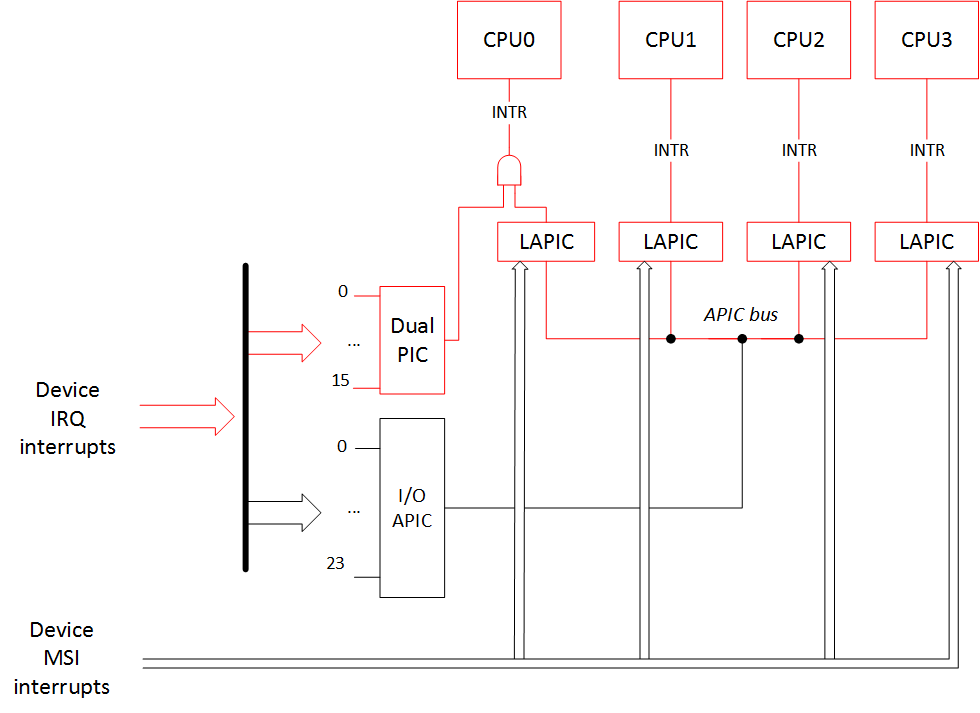
MCE: 0 Machine check exceptions

MCP: 1 Machine check polls

ERR: 0

MIS: 0

Комбинации:

На самом деле всего одна для нового варианта: «noapic pci=nomsi». Все прерывания от устройств смогут идти только на CPU0 через PIC. Но LAPIC работает и другие CPU могут работать и обрабатывать прерывания.  
  
Одна, потому что с «nolapic» можно ничего не комбинировать, т.к. эта опция и так сделает недоступным I/O APIC и MSI. Так что если вы когда-то прописывали опции загрузки «noapic nolapic» (или самый распространённый вариант «acpi=off noapic nolapic»), то судя по всему вы набирали лишние буквы.   
  
Итак, что будет от опций «noapic pci=nomsi»:   
  
  
  
Вывод /proc/interrupts:

CPU0 CPU1 CPU2 CPU3

0: 5 0 0 0 XT-PIC-XT-PIC timer

1: 2 0 0 0 XT-PIC-XT-PIC i8042

2: 0 0 0 0 XT-PIC-XT-PIC cascade

3: 5072 0 0 0 XT-PIC-XT-PIC i915, snd\_hda\_intel, eth59

4: 32 0 0 0 XT-PIC-XT-PIC eth58

8: 1 0 0 0 XT-PIC-XT-PIC rtc0

9: 0 0 0 0 XT-PIC-XT-PIC acpi

11: 281 0 0 0 XT-PIC-XT-PIC snd\_hda\_intel

12: 200 0 0 0 XT-PIC-XT-PIC ehci\_hcd:usb1

15: 7930 0 0 0 XT-PIC-XT-PIC ahci

NMI: 0 0 0 0 Non-maskable interrupts

LOC: 2595 2387 2129 1697 Local timer interrupts

SPU: 0 0 0 0 Spurious interrupts

PMI: 0 0 0 0 Performance monitoring interrupts

IWI: 159 90 482 135 IRQ work interrupts

RTR: 3 0 0 0 APIC ICR read retries

RES: 1568 1666 1810 1833 Rescheduling interrupts

CAL: 431 556 549 558 Function call interrupts

TLB: 124 184 156 274 TLB shootdowns

TRM: 116 116 116 116 Thermal event interrupts

THR: 0 0 0 0 Threshold APIC interrupts

MCE: 0 0 0 0 Machine check exceptions

MCP: 2 2 2 2 Machine check polls

ERR: 0

MIS: 0

Таблицы роутинга прерываний и опции «acpi=noirq», «pci=noacpi», «acpi=off»

Как операционная система получает информацию о роутинге прерываний от устройств? BIOS подготавливает информацию для ОС в виде:

* ACPI таблиц (методы \_PIC/\_PRT)
* \_MP\_ таблицы (MPtable)
* $PIR таблицы
* Регистров 0x3C/0x3D конфигурационного пространства PCI устройств

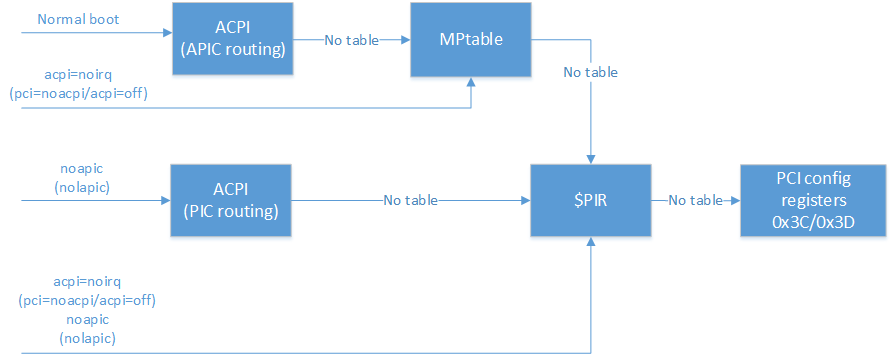
Следует отметить, что для обозначения прерываний MSI BIOSу не надо ничего дополнительно делать, вся вышеупомянутая информация нужна только для линий APIC/PIC прерываний.  
  
Таблицы в списке выше обозначены в порядке приоритета. Рассмотрим это подробней.  
  
Допустим BIOS предоставил все эти данные и мы грузимся без каких-либо дополнительных опций:

* ОС находит таблицы ACPI
* ОС выполняет метод ACPI "\_PIC", передаёт ему аргумент, что нужно грузиться в режиме APIC. Тут код метода обычно сохраняет выбранный режим в переменной (допустим PICM=1)
* Для получения данных о прерываниях ОС вызывает метод ACPI "\_PRT". Он внутри себя проверяет переменную PICM и возвращает роутинг для APIC случая  
  В случае если мы грузимся с опцией **noapic**:
* ОС находит таблицы ACPI
* ОС выполняет метод ACPI "\_PIC", передаёт ему аргумент, что нужно грузиться в режиме PIC. Тут код метода обычно сохраняет выбранный режим в переменной (допустим PICM=0)
* Для получения данных о прерываниях ОС вызывает метод ACPI "\_PRT". Он внутри себя проверяет переменную PICM и возвращает роутинг для PIC случая

Если таблица ACPI отсутствует или функционал роутинга прерываний через ACPI отключен с помощью опций **acpi=noirq** или **pci=noacpi** (или ACPI полностью выключен с помощью **acpi=off**), то ОС смотрит для роутинга прерываний таблицу MPtable (\_MP\_):

* ОС не находит/не смотрит таблицы ACPI
* ОС находит таблицу MPtable (\_MP\_)

Если таблица ACPI отсутствует или функционал роутинга прерываний через ACPI отключен с помощью опций **acpi=noirq** или **pci=noacpi** (или ACPI полностью выключен с помощью **acpi=off**) и если таблица MPtable (\_MP\_) отсутствует (или передана опция загрузки **noapic** или **nolapic**):

* ОС не находит/не смотрит таблицу ACPI
* ОС не находит/не смотрит таблицу MPtable (\_MP\_)
* ОС находит таблицу $PIR  
  Если и таблицы $PIR нет, или она не полна, то операционная система для угадывания прерываний будет смотреть значения регистров 0x3C/0x3D конфигурационного пространства PCI устройств.  
    
  Суммируем всё вышеизложенное следующей картинкой:  
    
    
    
  Следует помнить, что не каждый BIOS предоставляет все 3 таблицы (ACPI/MPtable/$PIR), так что если вы передали опцию загрузчику отказаться от использования ACPI или ACPI и MPtable для роутинга прерываний, далеко не факт, что ваша система загрузится.  
    
  **Замечание 1**: в случае если мы попытаемся загрузиться в режиме APIC с опцией acpi=noirq и без наличия MPtable, то картина прерываний будет как и в случае обычной загрузки с единственной опцией noapic. Операционная система сама перейдёт в режим PIC прерываний.  
  В случае если мы попытаемся загрузиться вообще без таблиц ACPI (acpi=off) и не предоставив MPtable, то картина будет такая:

CPU0

0: 6 XT-PIC-XT-PIC timer

1: 2 XT-PIC-XT-PIC i8042

2: 0 XT-PIC-XT-PIC cascade

8: 0 XT-PIC-XT-PIC rtc0

12: 373 XT-PIC-XT-PIC ehci\_hcd:usb1

16: 0 PCI-MSI-edge PCIe PME

17: 0 PCI-MSI-edge PCIe PME

18: 0 PCI-MSI-edge PCIe PME

19: 0 PCI-MSI-edge PCIe PME

20: 0 PCI-MSI-edge PCIe PME

21: 0 PCI-MSI-edge PCIe PME

22: 8728 PCI-MSI-edge ahci

23: 1 PCI-MSI-edge eth59

24: 1301 PCI-MSI-edge eth59-rx-0

25: 113 PCI-MSI-edge eth59-tx-0

26: 0 PCI-MSI-edge eth58

27: 45 PCI-MSI-edge eth58-rx-0

28: 45 PCI-MSI-edge eth58-tx-0

29: 1280 PCI-MSI-edge snd\_hda\_intel

NMI: 2 Non-maskable interrupts

LOC: 24076 Local timer interrupts

SPU: 0 Spurious interrupts

PMI: 2 Performance monitoring interrupts

IWI: 2856 IRQ work interrupts

RTR: 0 APIC ICR read retries

RES: 0 Rescheduling interrupts

CAL: 0 Function call interrupts

TLB: 0 TLB shootdowns

TRM: 34 Thermal event interrupts

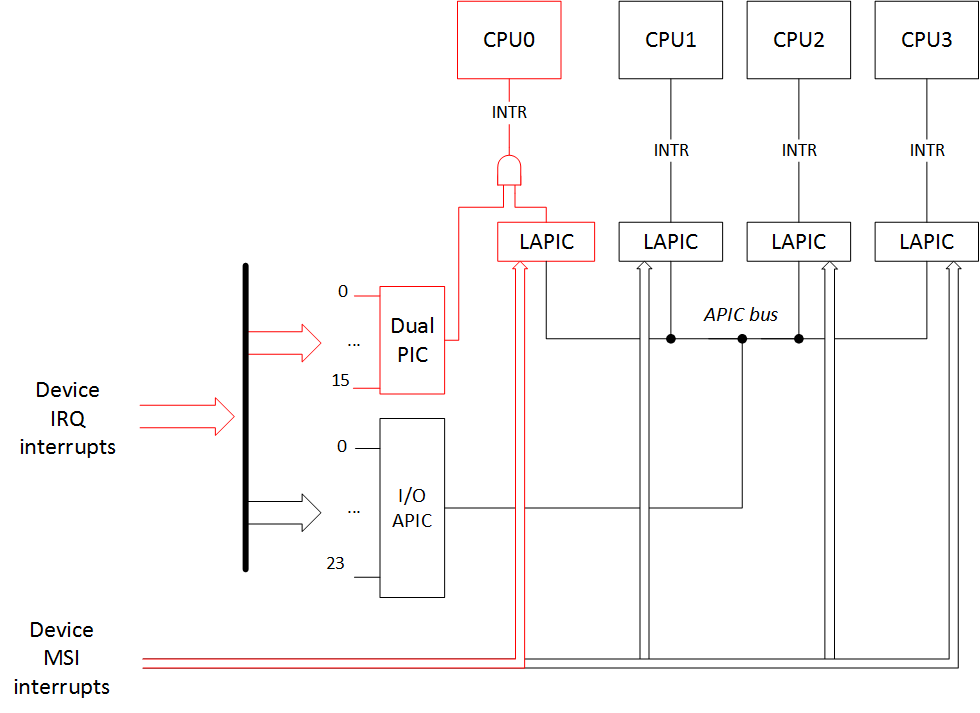
THR: 0 Threshold APIC interrupts

MCE: 0 Machine check exceptions

MCP: 2 Machine check polls

ERR: 0

MIS: 0

Это проиcходит из-за того, что без ACPI таблицы MADT ([Multiple APIC Description Table](https://wiki.osdev.org/MADT)) и необходимой информации из MPtable, операционная система не знает APIC идентификаторы (APIC ID) для других процессоров и не может с ними работать, но LAPIC основного процессора работает, так как мы это не запрещали, и MSI прерывания могут на него приходить. То есть будет так:  
  
  
  
**Замечание 2**: в целом роутинг прерываний при использовании ACPI в случае APIC совпадает с роутингом прерываний через MPtable. А роутинг прерываний через ACPI в случае использования PIC совпадает с роутингом прерываний через $PIR. Так что и выводы /proc/interrupts отличаться не должны. Однако в процессе исследований заметил одну странность. При роутинге через MPtable в выводе почему-то присутствует каскадное прерывание «XT-PIC-XT-PIC cascade».

CPU0 CPU1 CPU2 CPU3

0: 15 0 0 0 IO-APIC-edge timer

1: 2 0 0 0 IO-APIC-edge i8042

2: 0 0 0 0 XT-PIC-XT-PIC cascade

8: 0 1 0 0 IO-APIC-edge rtc0

9: 0 0 0 0 IO-APIC-edge acpi

...

Немного странно, что так происходит, но в [документации ядра](https://elixir.bootlin.com/linux/latest/source/Documentation/x86/i386/IO-APIC.txt) вроде говорится, что это нормально.

Заключение:

В заключении ещё раз обозначим разобранные опции.  
  
Опции выбора контроллера прерываний:

* **pci=nomsi** — MSI прерывания станут IO-APIC/XT-PIC в зависимости от используемого контроллера прерываний
* **noapic** — Отключает I/O APIC. MSI прерывания всё ещё могут идти на все CPU, остальные прерывания от устройств смогут идти только на PIC, а он работает только с CPU0. Но LAPIC работает и другие CPU могут работать и обрабатывать прерывания
* **noapic pci=nomsi** — Все прерывания от устройств могут идти только на PIC, а он работает только с CPU0. Но LAPIC работает и другие CPU могут работать и обрабатывать прерывания
* **nolapic** — Отключает LAPIC. MSI прерывания не могут работать без LAPIC, I/O APIC не может работать без LAPIC. Все прерывания от устройств будут идти на PIC, а он работает только с CPU0. И без LAPIC остальные CPU не будут работать.

Опции выбора приоритетной таблицы роутинга прерываний:

* **без опций** — роутинг через APIC с помощью таблиц ACPI
* **noapic** — роутинг через PIC с помощью таблиц ACPI
* **acpi=noirq** (**pci=noacpi**/**acpi=off**) — роутинг через APIC с помощью таблицы MPtable
* **acpi=noirq** (**pci=noacpi**/**acpi=off**) **noapic** (**nolapic**) — роутинг через PIC с помощью таблицы $PIR

В следующей части посмотрим как coreboot настраивает чипсет для роутинга прерываний.