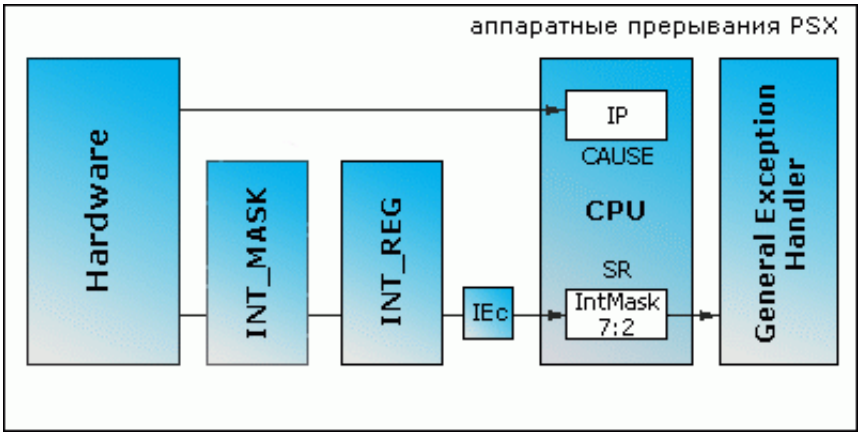
PSX TIMING

к таймингу относится обработка возникших прерываний (аппаратных или программных). следовательно, чтобы сэмулировать тайминг, нужно выяснить как возникают те или иные прерывания.

с программными прерываниями у PSX всё просто -- они вообще не используются. возникают они при записи в биты SW регистра CAUSE: битов две штуки, значит и прерываний тоже два. ОС может запретить программные прерывания, замаскировав биты 8 и/или 9 в поле IntMask, регистра SR.

следующий тип прерываний -- исключения процессора. все исключения шины и конвейера откидываются, как бесполезные. переполнение результата, деление на ноль и резервированные инструкции перехватываются ядром, или не используются вообще (в случае с OVF). про TLB можно и не знать, и не помнить... он не используется. единственная проблема -- адрес FFFE0130. это регистр сопроцессора (похоже на то). на что можно, и нужно обратить внимание -- инструкция syscall и её высокоуровневая эмуляция.

что касается аппаратных прерываний, то здесь наблюдается большая нехватка документации. реальным выходом была бы возможность получить PsyQ SDK, но его нет, и похоже уже не будет. однако, совершенно ясно, что прерываний столько, сколько устройств и аппаратных узлов имеет PSX. это GPU, SPU, CDROM, SIO/PIO, DMA, ROOT COUNTERS. вот так выглядит схема возникновения аппаратных прерываний:



прерывание проходит в процессор, только если оно разрешено соответствующим битом INT_MASK. каждый бит INT_MASK маскирует соотв. бит в регистре INT_REG, который является коллектором для всех аппаратных прерываний.
соответствие битов INT_REG и битов: IP в CAUSE , IntMask 7:2 в SR -- мне не известно, но оно точно есть.
необходимо учесть, что прерывание может возникнуть в любой момент времени, поэтому идеальным случаем проверки железа на прерывание, была бы проверка после каждой инструкции. реально можно выполнять проверку после DELAY-слотов или вместо NOP'ов.
после долгих поисков описания регистров INT_REG и INT_MASK, я нашел их в каком-то readme от исходников эмулятора FPSE:

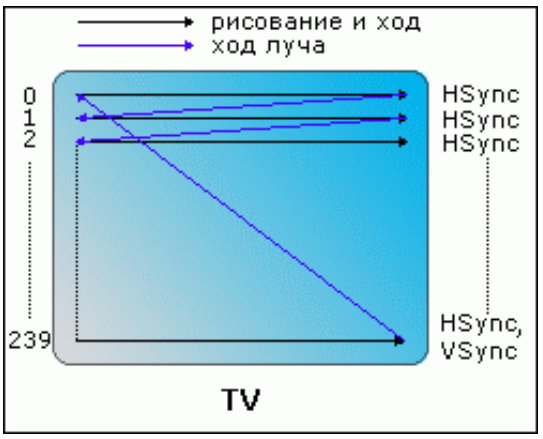
```
At "http://www.egroups.com/group/psxdev/"
I have found some info on the irq mask.

bit 0: VSync (Root counter 3)
bit 1: GPU
bit 2: CD-ROM
bit 3: DMA
bit 4: Root counter 0
bit 5: Root counter 1
bit 6: Root counter 2
bit 7: Controllers
bit 8: Serial port
bit 9: SPU
bit A: PIO
```

ROOT COUNTERS

RTC0 -- в режиме pixelclock не эмулируется, так как существующие видеоплагины не могут синхронизироваться с эмулятором на период, меньший чем VSync (suxx). у реальной PSX, RTC0 увеличивает свое значение с каждой выводимой GPU точкой.

RTC1 (HSync) и RTC3(VSync) эмулируются по следующим соображениям:



то есть, каждую 1/50 секунды (используя функцию Win32 GetTickCount) мы выполняем процедуру эмулятора doVSync, которая делает следующие вещи:

```
работа doVSync:
- GPU_Update (всегда)
- опрос PAD и клавиатуры (всегда)
- инкремент RTC3 Count (всегда)
- прерывание от RTC3 в INT_REG (если установлены биты Iq1 и Iq2 RTC3)
```

и каждую 1/(50*240) секунды мы вызываем функцию эмулятора doHSync. время вызова уже нельзя посчитать с помощью GetTickCount, поэтому мы создаем отдельный программный счетчик, который увеличивает свое значение после каждой эмулируемой инструкции, и по достижении определенного значения, вызывается doHSync. смысл работы этого псевдо-счетчика в том, что за одну scanline процессор выполняет строго определенное количество инструкций. для PAL-режимов мы вызываем doHSync после каждой 3240-й инструкции, а для NTSC-режимов -- после каждой 2530-й. как считать?

```
работа doHSync:
- инкремент RTC1 Count (только, если счетчик включен)
- прерывание от RTC1 в INT_REG (если установлены биты Iq1 и Iq2 RTC1)
```

примечание к RTC1 и RTC3: если используется NTSC видеорежим, то время синхронизаций изменяется на 1/60 и 1/(60*256) для RTC3 и RTC1 соответственно.
примечание: RTC3 -- программно не доступный счетчик. используя его мы нарушаем работу видеосистемы, поэтому если вы будете писать программу на ассемблере, лучше не использовать RTC3.

короче говоря, RTC1 увеличивает свое значение после каждой отрисованной scan-line, а RTC3 после каждого кадра.

RTC2 -- наиболее точный счетчик. он "тикает" одновременно с генератором тактовой частоты CPU (системные часы, CLK), а именно с частотой 33868800 Гц. также он способен работать на частоте 4233600 Гц (1/8 от ГТЧ), при установленном бите Div. после каждой эмулируемой инструкции RTC2 увеличивает свое значение на 1 , или на 8 (если установлен бит Div).
примечание: счетчики RTC0 и RTC1 также можно превратить в RTC2, сбросив бит Clc.

таким образом, программист PSX может использовать 3 независимых, _ОЧЕНЬ_ точных таймера. сравните: 1 миллисекунда у Win32 и 0.03 микросекунды у PSX!

DMA

после завершения DMA-передачи, контроллер генерирует прерывание. в регистре INT_REG DMA имеет свой бит, только вот какой канал вызвал прерывание, по содержимому INT_REG сказать нельзя. для этой цели используется другой регистр -- DMA_ICR. опять-же во всех документациях про этот регистр ничего не написано, но я нашел его описание в документе, называемом "отрисовка полигонов GPU" !

```
The DMA asks for more data by sending an interrupt.
You can read the register DMA_ICR, to figure out which channel needs feeding.
The top 8 bits of the 32 bit register can be masked
and tested to find out which, as follows:

0x80000000 - indicates a DMA bus error has occured.
0x40000000 - Feed dma channel 6
0x20000000 - Feed DMA channel 5
0x10000000 - Feed DMA channel 4
0x08000000 - Feed DMA channel 3
0x04000000 - Feed DMA channel 2
0x02000000 - Feed DMA channel 1
0x01000000 - Feed DMA channel 0
```

```
структура DMA_ICR:
бит 31 - ошибка шины (неверный физический адрес)
бит 30 - окончание пересылки по 6 DMA-каналу (очистка GPU OT)
бит 29 - окончание пересылки по 5 DMA-каналу (PIO)
бит 28 - окончание пересылки по 4 DMA-каналу (SPU)
бит 27 - окончание пересылки по 3 DMA-каналу (CD-ROM)
бит 26 - окончание пересылки по 2 DMA-каналу (GPU данные, списки)
бит 25 - окончание пересылки по 1 DMA-каналу (MDECout)
бит 24 - окончание пересылки по 0 DMA-каналу (MDECin)
остальные биты не используются
```

что означает приблизительно следующее: после окончания пересылки по любому DMA-каналу, устанавливается бит 3 в регистре INT_REG (конечно, если это позволяет INT_MASK) и бит соответствующего канала в регистре DMA_ICR. если происходит ошибка шины, то устанавливается старший бит регистра DMA_ICR.

назад...