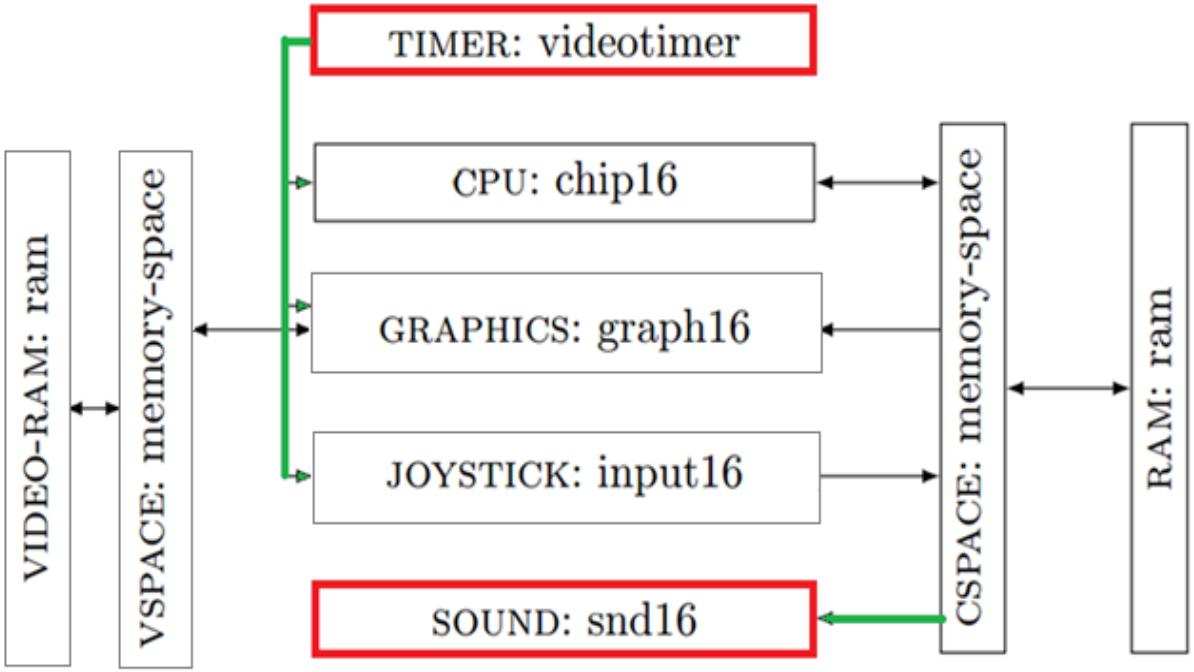
Sound and Timer

Шимченко Марина
shimchenko.sk@gmail.ru

17 октября 2014 г.

Лаборатория МФТИ-Интел

Chip16


TIMER: videotimer
CPU: chip16
GRAPHICS: graph16
JOYSTICK: input16
SOUND: snd16
VIDEO-RAM: ram
VSPACE: memory-space
CSPACE: memory-space
RAM: ram

1. Sound

Режимы

Режим,при котором не происходит генерации звука

Режим, когда мы можем генерировать звук, определенной частоты и длительности

Режим, в котором можно управлять формой сигнала и огибающей

Выключенный режим и генерации простейшего сигнала

Регистр tone :

Определяет частоту генерируемого сигнала

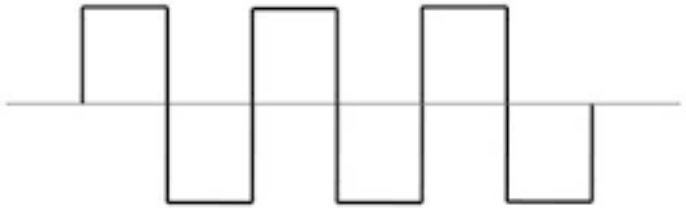
- 0 – звук выкл.
- 500 Hz
- 1000 Hz
- 1500 Hz

Регистр limit :

Определяет длительность звучания сигнала в миллисекундах

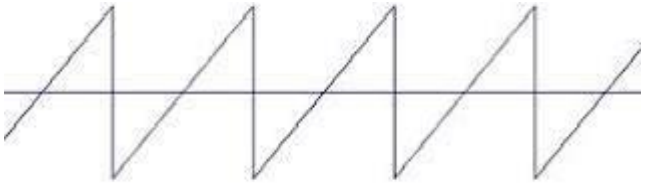
Режим управления формой сигнала

PULSE



TRIANGLE

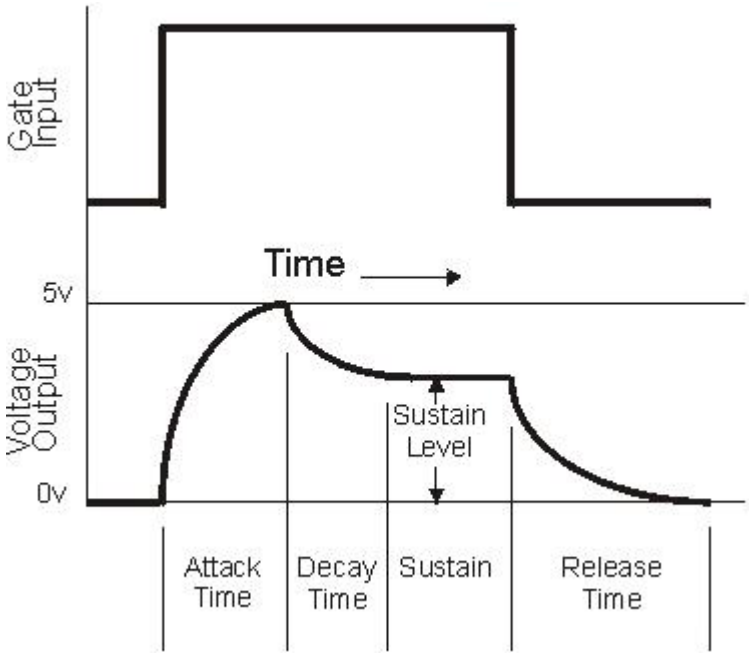
SAWTOOTH



NOISE

http://music.tutsplus.com/tutorials/an-introduction-to-adsr--audio-11150

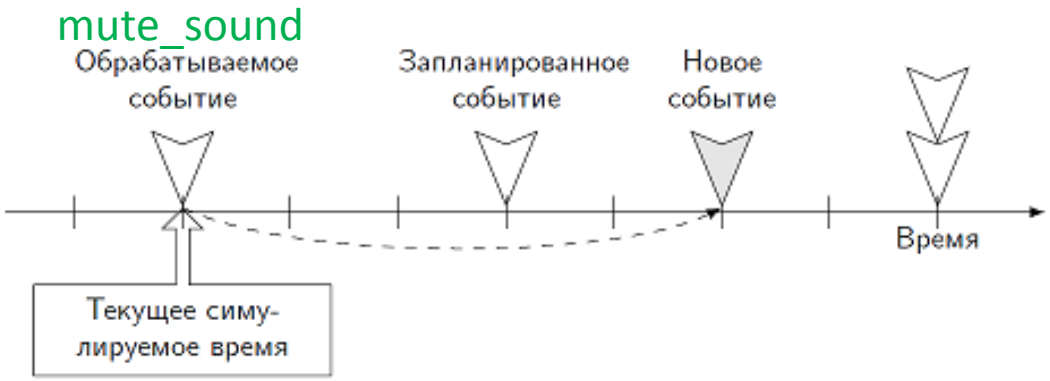
Управление огибающей сигнала



Операнды инструкции SNG AD VTSR:

- A - attack, длительность
- D - decay, длительность.
- V - volume, макс. значение при атаке.
- T – type,форма модулируемого сигнала: TRIANGLE = 0, SAWTOOTH = 1, PULSE = 2, NOISE = 3.
- S - sustain, громкость.
- R - release, длительность.

Как моделировать?

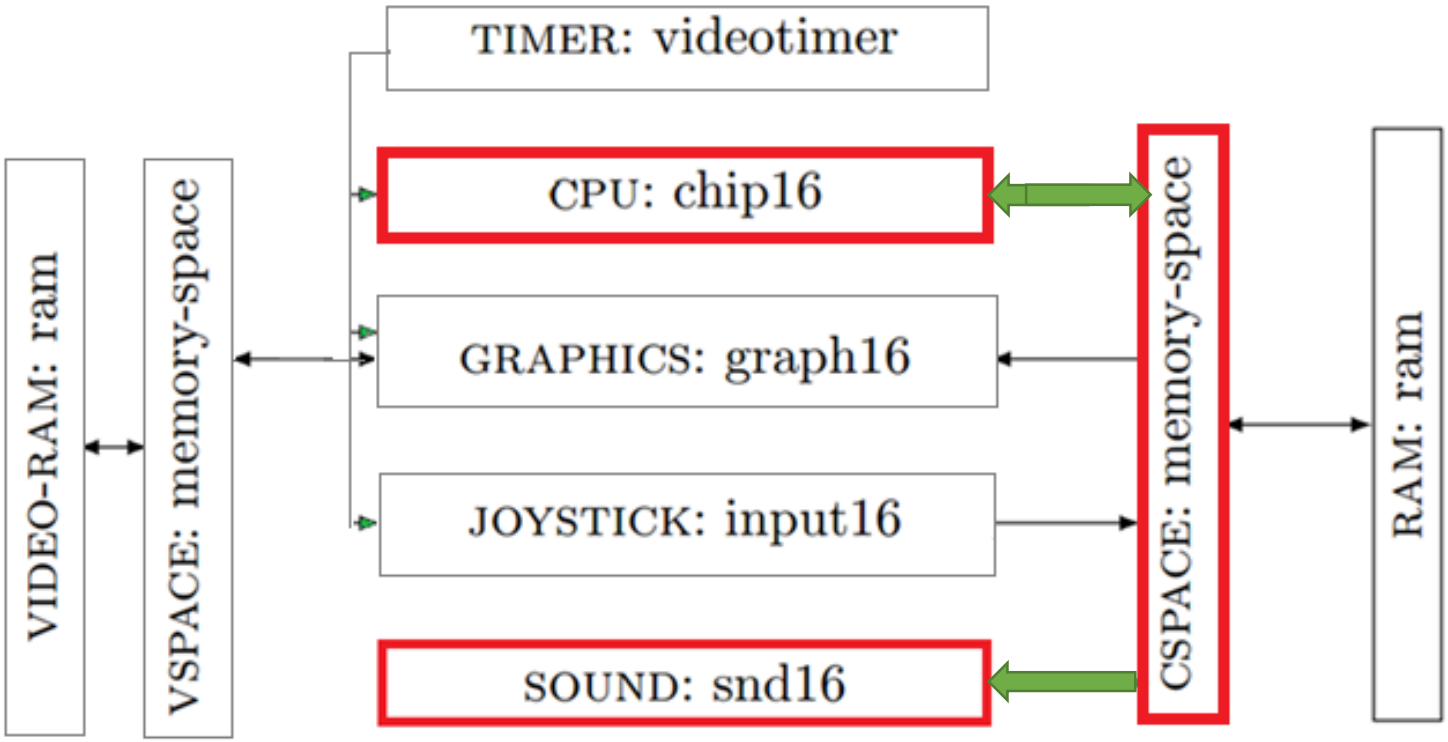


mute_sound – событие, которое генерирует выключение звука

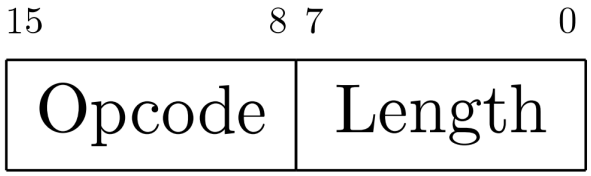
If (tone != 0) and (limit != 0)
{
    вкл генерация звука
    mute_sound порождается (или переставляется)
}

end_attack, end_decay, end_sustain, end_release – используются для модулирования разных фаз генерации

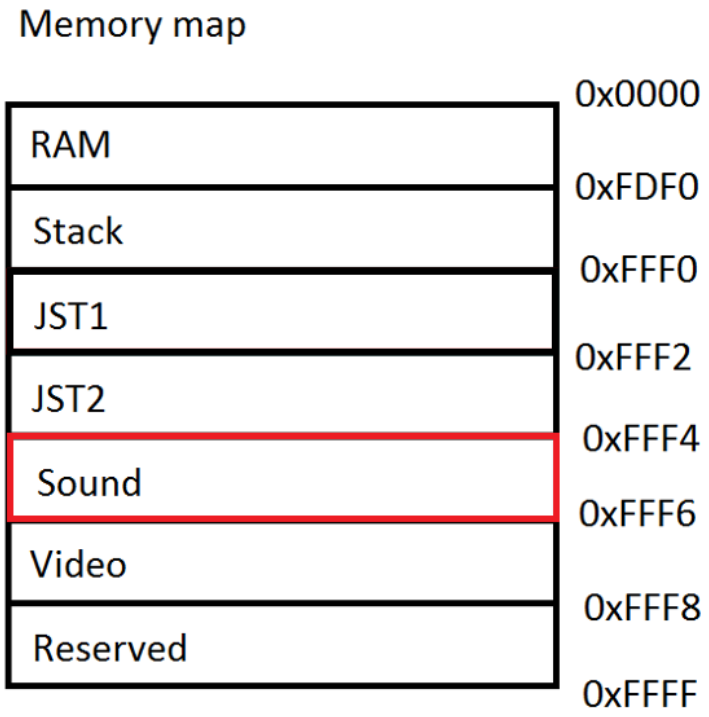
Общение ЦПУ с звуковой картой

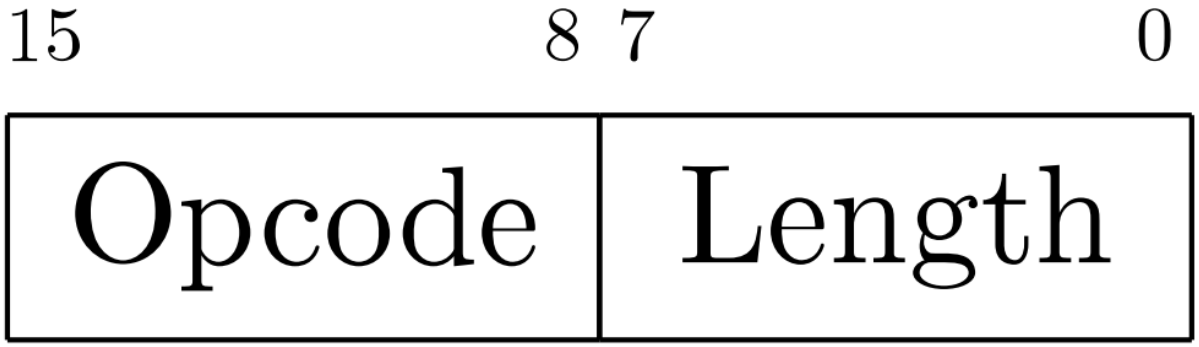


Memory mapping



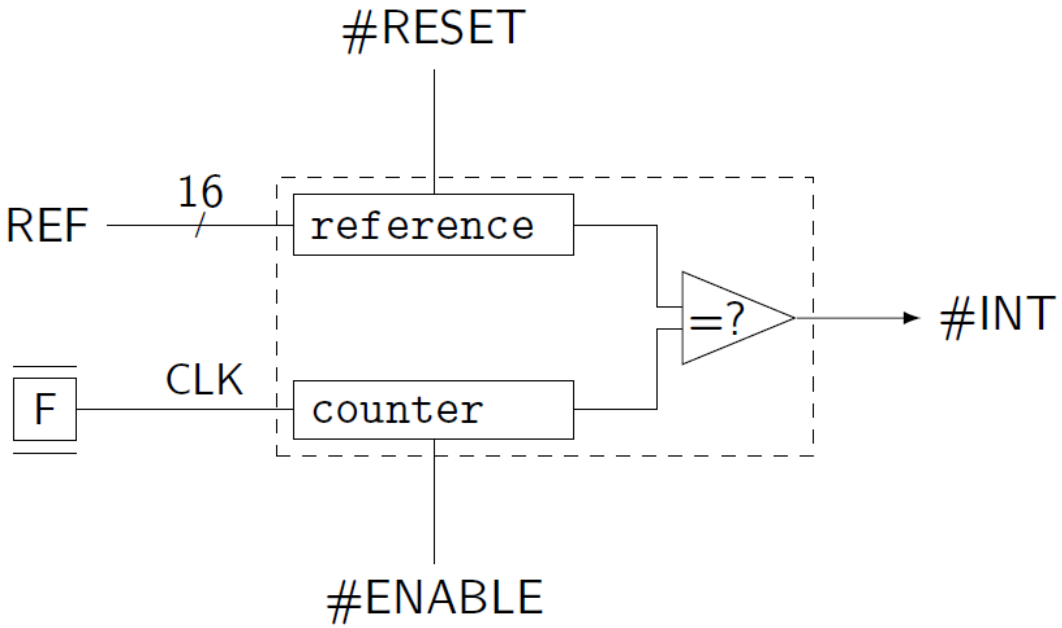
Opcode — код операции,
Length — полная длина последующей команды в словах

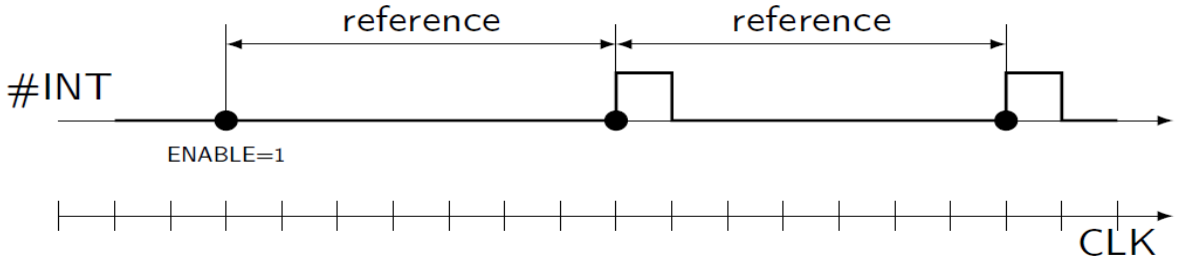




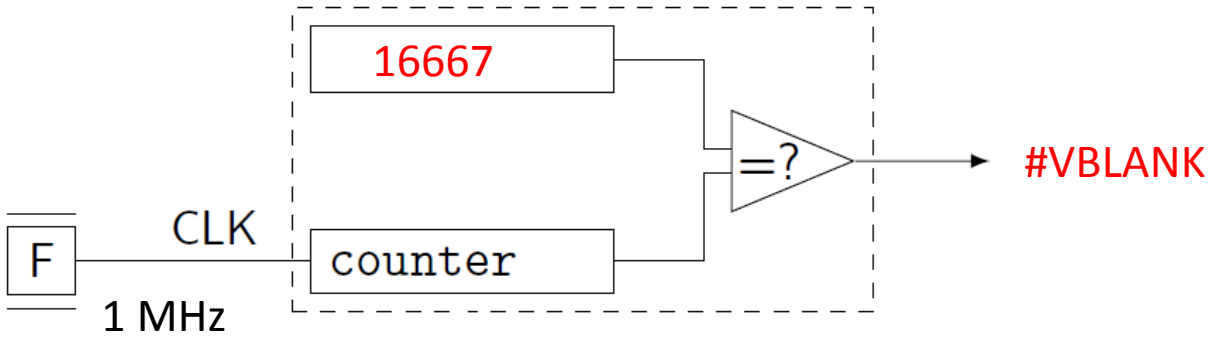
Упр. слово	Аргументы	Описание
0, 2	TONE HHLL	Инстр. SND0…SND3, SNP
1, 2	ADVT SR00	Инструкция SNG

2. Timer

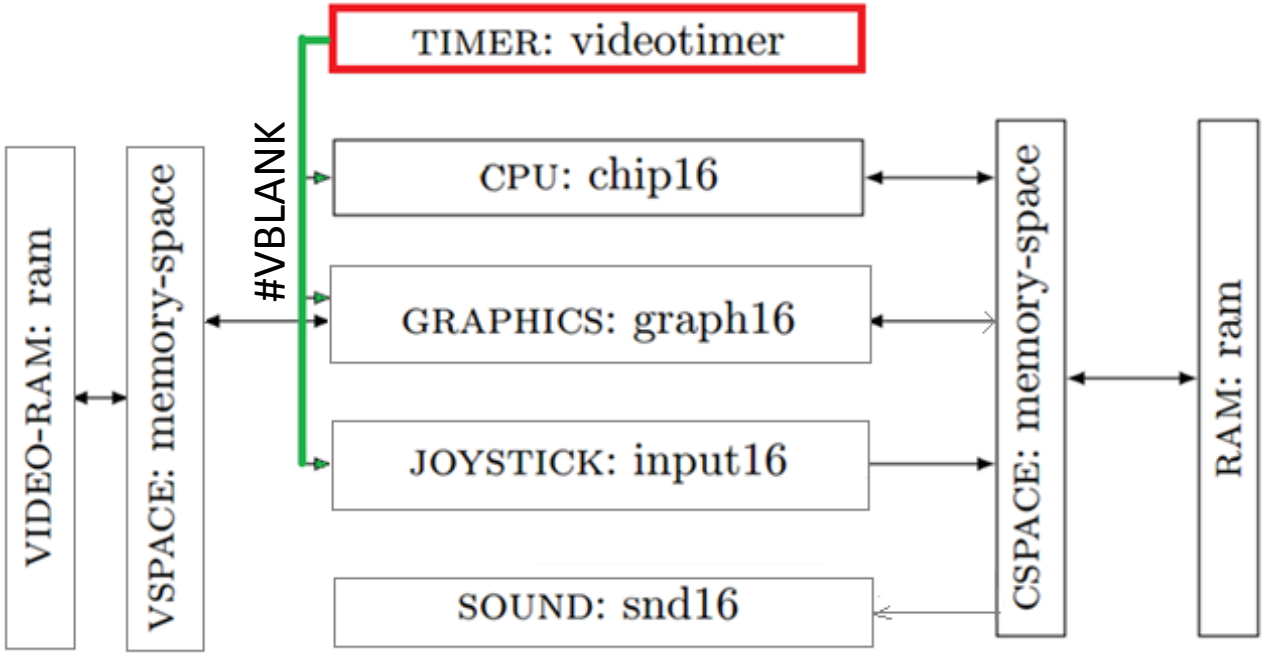

#RESET
16
REF
reference
=?
#INT
F
CLK
counter
#ENABLE


reference
reference
#INT
ENABLE=1
CLK

Наша модель таймера

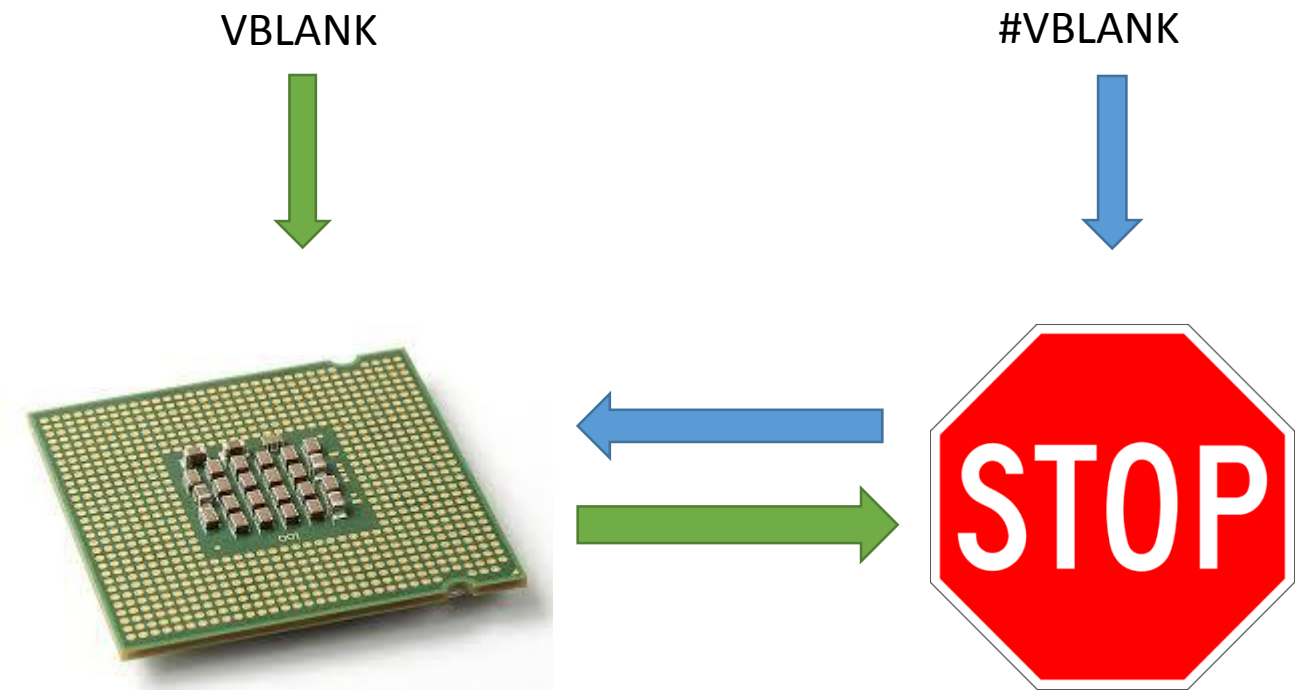


Зачем таймер??



Инструкция VBLANK –

отвечает за синхронизацию ЦПУ и выводов видеокарты



About DML: banks and registers

- A register bank (or simply bank) is an abstraction that is used to group registers in DML
- A register is an component of a register bank that contains an integer value, which can be either unsigned (the default) or signed.

```
register r1 {
parameter size = 4;
...
}
```

```
register r1 size 4 {
...
}
```

```
register r1 {
parameter offset = 0x0100;
...
}
```

```
register r1 @ 0x0100 {
...
}
```

About DML: bank and registers

register r1 size 4 @ 0x0100; // Как часто делают на самом деле

log "info": "the value of register r1 is %d", $r1;

if the bank contains several registers of the same size:

```
bank b1 {
        parameter register_size = 4;
        register r1 @ 0x0100;
        register r2 @ 0x0103;
        ...
}
```

About DML: registers and fields

```
bank b2 {
register r0 size 4 @ 0x0000 {
        field status [2:0];
        field flags [8:3];
        field reserved [15:9] {
                parameter allocate = false;
        } ;
}
...
}
```

```
log "info": "the value of the status field is %d", $r0.status;
```

About DML: attributes

Registers defined in the DML files are automatically registered as both object structure
variables and attributes.

log "info": "the value of attribute a is %d", $dev.a;

```
attribute counter {
        parameter documentation = "A
        sample counter attribute";
        parameter type = "i";
        parameter allocate_type =
"int64";
        parameter configuration =
"required";
}
```

About DML: connects and interfaces

- A connect is thus similar to a simple attribute object!!!!!

```
connect plugin {
        interface serial_device;
        interface rs232_device { parameter required = false; }
}
implement io_memory {
        method map(addr_space_t sp, map_info_t info)-> (int status)
        {       ...
        }
        method operate (generic_transaction_t *op, map_info_t info)-
>(exception_type_t exc)
        {
        ...
        }
}
```

About DML: events

```
event future {
        method event(void *data) {
                log "info", 1, 1 : "The future is here";
        }
}
```

- The **event()** method is called when the queue reaches a posted event.
- To post an event, use the **post()** method.

```
// post an event 0.1 s in the future
inline $future.post(0.1, NULL);
```

Спасибо за внимание!