Accessing VRAM and OAM

> **WARNING**
>
> When the PPU is drawing the screen it is directly reading from Video Memory (VRAM) and from the Sprite Attribute Table (OAM). During these periods the Game Boy CPU may not access VRAM and OAM. That means that any attempts to write to VRAM or OAM are ignored (data remains unchanged). And any attempts to read from VRAM or OAM will return undefined data (typically $FF).
>
> For this reason the program should verify if VRAM/OAM is accessible before actually reading or writing to it. This is usually done by reading the Mode bits from the STAT Register (FF41). When doing this (as described in the examples below) you should take care that no interrupts occur between the wait loops and the following memory access - the memory is guaranteed to be accessible only for a few cycles just after the wait loops have completed.

VRAM (memory area at $8000-$9FFF) is accessible during Modes 0-2

```
Mode 0 - HBlank Period,
Mode 1 - VBlank Period, and
Mode 2 - Searching OAM Period
```

A typical procedure that waits for accessibility of VRAM would be:

```
    ld   hl, $FF41     ; STAT Register
.wait
    bit  1, [hl]       ; Wait until Mode is 0 or 1
    jr   nz, .wait
```

Even if the procedure gets executed at the *end* of Mode 0 or 1, it is still safe to assume tha VRAM can be accessed for a few more cycles because in either case the following period i Mode 2, which allows access to VRAM also. However, be careful about STAT interrupts or other interrupts that could cause the PPU to be back in Mode 3 by the time it returns. In CGB Mode an alternate method to write data to VRAM is to use the HDMA Function (FF5 FF55).

If you do not require any STAT interrupts, another way to synchronize to the start of Mod is to disable all the individual STAT interrupts except Mode 0 (STAT bit 3), enable STAT interrupts (IE bit 1), disable IME (by executing `di`), and use the `halt` instruction. This allow use of the entire Mode 0 on one line and Mode 2 on the following line, which sum to 165 t 288 dots. For comparison, at single speed (4 dots per machine cycle), a copy from stack th takes 9 cycles per 2 bytes can push 8 bytes (half a tile) in 144 dots, which fits within the worst case timing for mode 0+2.

OAM (memory area at $FE00-$FE9F) is accessible during Modes 0-1

```
Mode 0 - HBlank Period
Mode 1 - VBlank Period
```

During those modes, OAM can be accessed directly or by doing a DMA transfer (FF46). Outside those modes, DMA out-prioritizes the PPU in accessing OAM, and the PPU will re $FF from OAM during that time.

A typical procedure that waits for accessibility of OAM would be:

```
    ld   hl, $FF41    ; STAT Register
    ; Wait until Mode is -NOT- 0 or 1
.waitNotBlank
    bit  1, [hl]
    jr   z, .waitNotBlank
    ; Wait until Mode 0 or 1 -BEGINS- (but we know that Mode 0 is what will begin)
.waitBlank
    bit  1, [hl]
    jr   nz, .waitBlank
```

The two wait loops ensure that Mode 0 (and Mode 1 if we are at the end of a frame) will la for a few clock cycles after completion of the procedure. If we need to wait for the VBlank period, it would be better to skip the whole procedure, and use a STAT interrupt instead. any case, doing a DMA transfer is more efficient than writing to OAM directly.

> **NOTE**
>
> While the display is disabled, both VRAM and OAM are accessible. The downside is that the screen is blank (white) during this period, so disabling the display would be recommended only during initialization.