Foreword
Acknowledgements
History

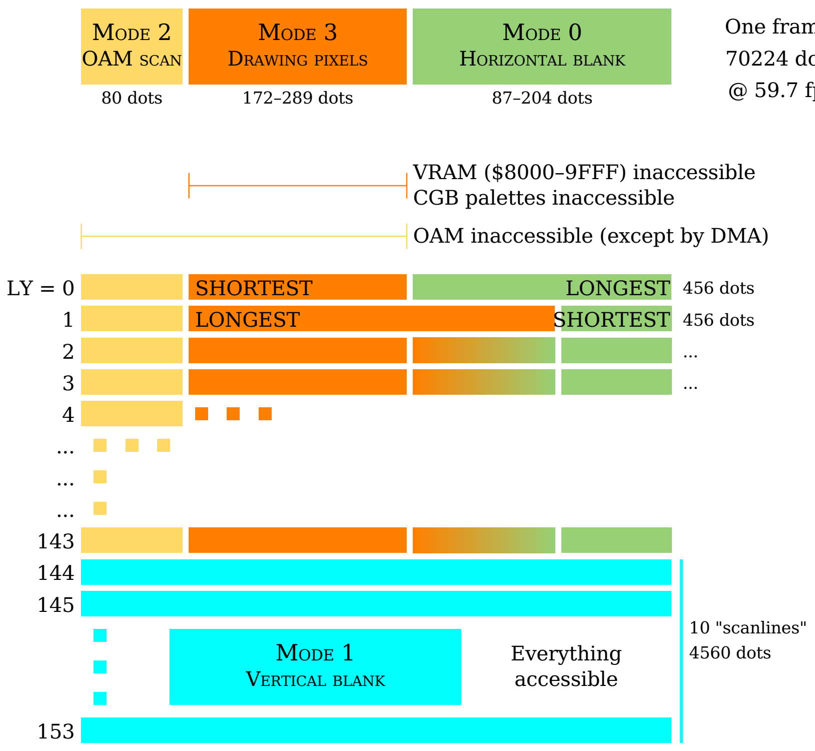
Overview
1. Specifications
2. Memory Map
I/O Ports
3. Rendering
3.1. Tile Data
3.2. Tile Maps
3.3. OAM
3.4. LCD Control
3.5. LCD Status
3.6. Scrolling
3.7. Palettes
3.8. Pixel FIFO
4. Sound Controller
5. Joypad Input
6. Serial Data Transfer
7. Timer and Divider Registers
8. Interrupts
9. CGB Registers
10. Infrared Communication
11. SGB Functions
CPU Specifications
12. CPU Registers and Flags
13. CPU Instruction Set
14. CPU Comparison with Z80
Cartridges
15. The Cartridge Header
16. MBCs
Accessories
17. Game Boy Printer
18. Game Boy Camera
19. 4-Player Adapter
20. Game Genie/Shark Cheats
Other
21. Power-Up Sequence
22. Reducing Power Consumption
23. Accessing VRAM and OAM
24. OAM Corruption Bug
25. External Connectors
26. GBC Approval Process

References

Pixel FIFO

TERMINOLOGY

All references to a dot are meant as dots (4.19 MHz). Dots remain the same regardless of CGB double speed. When it is stated that a certain action *lengthens mode 3* it means that mode 0 (HBlank) is shortened to make up for the additional time in mode 3, as shown in the following diagram.



Introduction

FIFO stands for *First In, First Out*. The first pixel to be pushed to the FIFO is the first pixe to be popped off. In theory that sounds great, in practice there are a lot of intricacies.

There are two pixel FIFOs. One for background pixels and one for OAM (sprite) pixels. These two FIFOs are not shared. They are independent of each other. The two FIFOs are mixed only when popping items. Sprites take priority unless they're transparent (color 0) which will be explained in detail later. Each FIFO can hold up to 16 pixels. The FIFO and Pixel Fetcher work together to ensure that the FIFO always contains at least 8 pixels at a given time, as 8 pixels are required for the Pixel Rendering operation to take place. Each FIFO is manipulated only during mode 3 (pixel transfer).

Each pixel in the FIFO has four properties:

- Color: a value between 0 and 3
- Palette: on CGB a value between 0 and 7 and on DMG this only applies to sprites
- Sprite Priority: on CGB this is the OAM index for the sprite and on DMG this doesn't exist
- Background Priority: holds the value of the OBJ-to-BG Priority bit

FIFO Pixel Fetcher

The fetcher fetches a row of 8 background or window pixels and queues them up to be mixed with sprite pixels. The pixel fetcher has 5 steps. The first four steps take 2 dots eac and the fifth step is attempted every dot until it succeeds. The order of the steps are as follows:

- Get tile
- Get tile data low
- Get tile data high
- Sleep
- Push

Get Tile

This step determines which background/window tile to fetch pixels from. By default the tilemap used is the one at $9800 but certain conditions can change that.

When LCDC.3 is enabled and the X coordinate of the current scanline is not inside the window then tilemap $9C00 is used.

When LCDC.6 is enabled and the X coordinate of the current scanline is inside the windov then tilemap $9C00 is used.

The fetcher keeps track of which X and Y coordinate of the tile it's on:

If the current tile is a window tile, the X coordinate for the window tile is used, otherwise the following formula is used to calculate the X coordinate: ((SCX / 8) + fetcher's X coordinate) & $1F. Because of this formula, fetcherX can be between 0 and 31.

If the current tile is a window tile, the Y coordinate for the window tile is used, otherwise the following formula is used to calculate the Y coordinate: (currentScanline + SCY) & 25 Because of this formula, fetcherY can be between 0 and 255.

The fetcher's X and Y coordinate can then be used to get the tile from VRAM. However, if the PPU's access to VRAM is blocked then the value for the tile is read as $FF.

CGB can access both tile index and the attributes in the same clock dot.

Get Tile Data Low

Check LCDC.4 for which tilemap to use. At this step CGB also needs to check which VRAM bank to use and check if the tile is flipped vertically. Once the tilemap, VRAM and vertical flip is calculated the tile data is retrieved from VRAM. However, if the PPU's access to VRAM is blocked then the tile data is read as $FF.

The tile data retrieved in this step will be used in the push steps.

Get Tile Data High

Same as Get Tile Data Low except the tile address is incremented by 1.

The tile data retrieved in this step will be used in the push steps.

This also pushes a row of background/window pixels to the FIFO. This extra push is not pa of the 8 steps, meaning there's 3 total chances to push pixels to the background FIFO eve time the complete fetcher steps are performed.

Push

Pushes a row of background/window pixels to the FIFO. Since tiles are 8 pixels wide, a "row" of pixels is 8 pixels from the tile to be rendered based on the X and Y coordinates calculated in the previous steps.

Pixels are only pushed to the background FIFO if it's empty.

This is where the tile data retrieved in the two Tile Data steps will come in handy. Depending on if the tile is flipped horizontally the pixels will be pushed to the background FIFO differently. If the tile is flipped horizontally the pixels will be pushed LSB first. Otherwise they will be pushed MSB first.

Sleep

Do nothing.

VRAM Access

At various times during PPU operation read access to VRAM is blocked and the value reac $FF:

- LCD turning off
- At scanline 0 on CGB when not in double speed mode
- When switching from mode 3 to mode 0
- On CGB when searching OAM and index 37 is reached

At various times during PPU operation read access to VRAM is restored:

- At scanline 0 on DMG and CGB when in double speed mode
- On DMG when searching OAM and index 37 is reached
- After switching from mode 2 (oam search) to mode 3 (pixel transfer)

NOTE: These conditions are checked only when entering STOP mode and the PPU's acces to VRAM is always restored upon leaving STOP mode.

Mode 3 Operation

As stated before the pixel FIFO only operates during mode 3 (pixel transfer). At the beginning of mode 3 both the background and OAM FIFOs are cleared.

The Window

When rendering the window the background FIFO is cleared and the fetcher is reset to st
1. When WX is 0 and the SCX & 7 > 0 mode 3 is shortened by 1 dot.

When the window has already started rendering there is a bug that occurs when WX is changed mid-scanline. When the value of WX changes after the window has started rendering and the new value of WX is reached again, a pixel with color value of 0 and the lowest priority is pushed onto the background FIFO.

Sprites

The following is performed for each sprite on the current scanline if LCDC.1 is enabled (th
condition is ignored on CGB) and the X coordinate of the current scanline has a sprite on
If those conditions are not met then sprite fetching is aborted.

At this point the fetcher is advanced one step until it's at step 5 or until the background FIFO is not empty. Advancing the fetcher one step here lengthens mode 3 by 1 dot. This process may be aborted after the fetcher has advanced a step.

When SCX & 7 > 0 and there is a sprite at X coordinate 0 of the current scanline then mo
3 is lengthened. The amount of dots this lengthens mode 3 by is whatever the lower 3 bits
SCX are. After this penalty is applied object fetching may be aborted. Note that the timing
the penalty is not confirmed. It may happen before or after waiting for the fetcher. More research needs to be done.

After checking for sprites at X coordinate 0 the fetcher is advanced two steps. The first advancement lengthens mode 3 by 1 dot and the second advancement lengthens mode 3 b
3 dots. After each fetcher advancement there is a chance for a sprite fetch abortion to occ

The lower address for the row of pixels of the target object tile is now retrieved and lengthens mode 3 by 1 dot. Once the address is retrieved this is the last chance for sprite fetch abortion to occur. Exiting object fetch lengthens mode 3 by 1 dot. The upper addres
for the target object tile is now retrieved and does not shorten mode 3.

At this point VRAM Access is checked for the lower and upper addresses for the target object. Before any mixing is done, if the OAM FIFO doesn't have at least 8 pixels in it then
transparent pixels with the lowest priority are pushed onto the OAM FIFO. Once this is do
each pixel of the target object row is checked. On CGB, horizontal flip is checked here. If
target object pixel is not white and the pixel in the OAM FIFO *is* white, or if the pixel in th
OAM FIFO has higher priority than the target object's pixel, then the pixel in the OAM FI
is replaced with the target object's properties.

Now it's time to render a pixel! The same process described in Sprite Fetch Abortion is performed: a pixel is rendered and the fetcher is advanced one step. This advancement lengthens mode 3 by 1 dot if the X coordinate of the current scanline is not 160. If the X coordinate is 160 the PPU stops processing sprites (because they won't be visible).

Everything in this section is repeated for every sprite on the current scanline unless it wa
decided that fetching should be aborted or the X coordinate is 160.

Pixel Rendering

This is where the background FIFO and OAM FIFO are mixed. There are conditions where
either a background pixel or a sprite pixel will have display priority.

If there are pixels in the background and OAM FIFOs then a pixel is popped off each. If th
OAM pixel is not transparent and LCDC.1 is enabled then the OAM pixel's background priority property is used if it's the same or higher priority as the background pixel's background priority.

Pixels won't be pushed to the LCD if there is nothing in the background FIFO or the curre
pixel is pixel 160 or greater.

If LCDC.0 is disabled then the background is disabled on DMG and the background pixel won't have priority on CGB. When the background pixel is disabled the pixel color value w
be 0, otherwise the color value will be whatever color pixel was popped off the backgroun
FIFO. When the pixel popped off the background FIFO has a color value other than 0 and has priority then the sprite pixel will be discarded.

At this point, on DMG, the color of the pixel is retrieved from the BGP register and pushed
to the LCD. On CGB when palette access is blocked a black pixel is pushed to the LCD.

When a sprite pixel has priority the color value is retrieved from the popped pixel from th
OAM FIFO. On DMG the color for the pixel is retrieved from either the OBP1 or OBP0 register depending on the pixel's palette property. If the palette property is 1 then OBP1 i

used, otherwise OBP0 is used. The pixel is then pushed to the LCD. On CGB when palette access is blocked a black pixel is pushed to the LCD.

The pixel is then finally pushed to the LCD.

CGB Palette Access

At various times during PPU operation read access to the CGB palette is blocked and a bla pixel pushed to the LCD when rendering pixels:

- LCD turning off
- First HBlank of the frame
- When searching OAM and index 37 is reached
- After switching from mode 2 (oam search) to mode 3 (pixel transfer)
- When entering HBlank (mode 0) and not in double speed mode, blocked 2 dots later matter what

At various times during PPU operation read access to the CGB palette is restored and pix are pushed to the LCD normally when rendering pixels:

- At the end of mode 2 (oam search)
- For only 2 dots when entering HBlank (mode 0) and in double speed mode

> **NOTE**
>
> These conditions are checked only when entering STOP mode and the PPU's access to CGB palettes is always restored upon leaving STOP mode.

Sprite Fetch Abortion

Sprite fetching may be aborted if LCDC.1 is disabled while the PPU is fetching an object from OAM. This abortion lengthens mode 3 by the amount of dots the previous instructio took plus the residual dots left for the PPU to process. When OAM fetching is aborted a pi is rendered, the fetcher is advanced one step. This advancement lengthens mode 3 by 1 d if the current pixel is not 160. If the current pixel is 160 the PPU stops processing sprites because they won't be visible.