The Picture Processing Unit

**A. Overview:**

Memory Map:

|  |  |  |  |
| --- | --- | --- | --- |
| Section Name | Starting address | Physical size | Details |
| Stack | 0x7FFFFFFF | 128 kB | Stack grows down |
| Heap Segment | 0x10500000 | 128 kB |  |
| Colors Palette | 0x104FF000 | 1 kB |  |
| Sprite Ram | 0x10400000 | 512 Bytes |  |
| Tilemaps | 0x10100000 | 3 MB |  |
| Sprites’ Tileset | 0x10030000 | 64 kB |  |
| Background Tileset | 0x10020000 | 64 kB |  |
| Data Segment | 0x10010000 | 64 kB | Static Memory |
| Text segment | 0x04000000 | 64 kB |  |

The Picture Processing Unit is the coprocessor responsible for video generation using the data stored in the memory. It doesn’t support regular bitmap graphics display. Instead, it uses a tile-based engine for graphics display to save RAM and increase real-time rendering performance.

Each scene is divided into *tiles,* 16×16 pixels each. The unique tiles are stored in the *tileset*. To create the image from the tiles stored in the *tileset* we use a *tilemap*. A *tilemap* divides the scene into elements each containing a *tile index* which indicates which tile should be placed and rendered in this particular position.



*Figure 1 - The scene divided into tiles*

**B. Technical Details:**

**1- The Memory:**

1. The tilesets

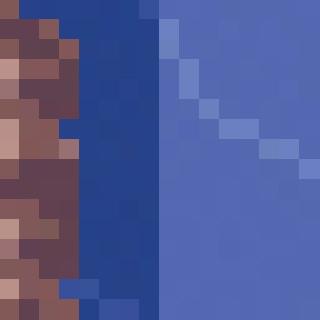
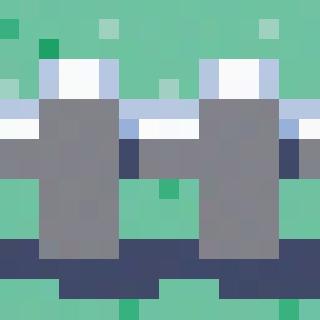
The tileset is the data structure that stores all the unique tiles. It has the capacity to store up to 256 different tiles. Each tile is a 2-dimensional array of pixels of dimensions 16×16. The color of each pixel in the matrix is represented using 1 Byte; the index of the color in the Palette table.

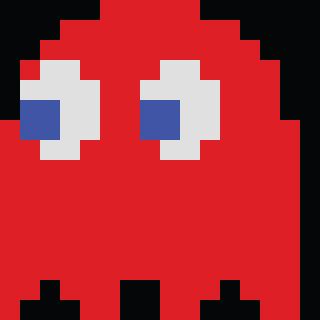
The size of each tile (in Bytes) = 16 pixels width × 16 pixels height × 1 Byte = 256 Bytes.

The total size of a tileset = 256 tiles × 256 Bytes per tile = 65536 Bytes = 64 kB.

The PPU has two different tilesets one for the background and one for the sprites.

Tiles are stored one after another in the memory starting from address 0x010020000 for the background tileset and 0x010030000 for the sprites’ tileset.





*Figure 2 - tiles examples*

1. The tilemaps

These are the 2-dimensional arrays that tell the engine which tile should be placed at which position. Each cell in the matrix contains 1 Byte; the index of the tile in the tileset.

The PPU holds place for 4 tilemaps. They are placed one after another in the memory starting from address 0x00210000. The size of each tilemap is 40×30 = 1200 Bytes.

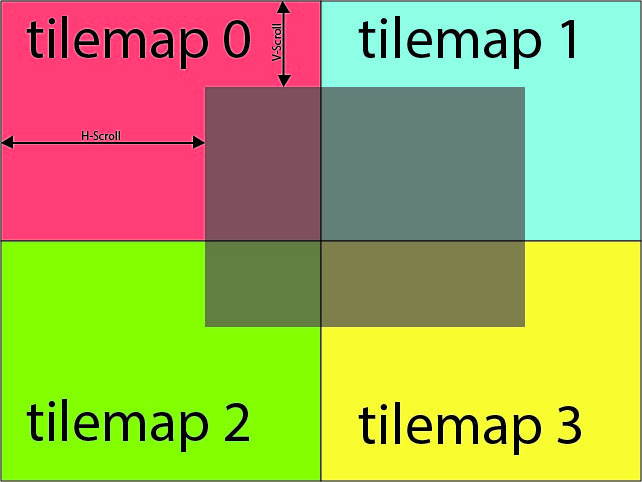
Tilemap 0: starts at address 0x00210000

Tilemap 1: starts at address 0x002104B0

Tilemap 2: starts at address 0x00210960

Tilemap 3: starts at address 0x00210E10

The screen can only display 1 tilemap at a time. it is allowed to move across the tilemaps using the 2 registers HScroll and VScroll.



*Figure 3 - tilemaps positioning*

1. The OAM:

The sprites are stored in a separate section in the memory known as the Object Attribute Memory (OAM in-short). A maximum of 64 Sprites can be stored. Each one of them consists of 8 Bytes divided as follows:



Figure 4 - Sprite consisting of 2 tiles

|  |  |
| --- | --- |
| Byte 0 | The index in the tileset of tile 0 for this sprite |
| Byte 1 | The index in the tileset of tile 1 for this sprite |
| Byte 2 and 3 | The X position of the Sprite from the left side of the screen |
| Byte 4 and 5 | The Y position of the Sprite from the top of the screen |
| Byte 6 | * Bit 1 and 0 orientation for tile 0   + 00 = No rotation   + 01 = rotate 90 degrees clockwise   + 10 = rotate 90 degrees counter-clockwise   + 11 = rotate 180 degrees * Bit 2 flip vertically for tile 0   + 0 = Don’t flip   + 1 = Flip * Bit 3 flip horizontally for tile 0   + 0 = Don’t flip   + 1 = Flip * Bit 5 and 4 orientation for tile 1   + 00 = No rotation   + 01 = rotate 90 degrees clockwise   + 10 = rotate 90 degrees counter-clockwise   + 11 = rotate 180 degrees * Bit 6 flip vertically for tile 1   + 0 = Don’t flip   + 1 = Flip * Bit 7 flip horizontally for tile 1   + 0 = Don’t flip   + 1 = Flip |
| Byte 7 | * Bit 0 Sprite size   + 0 = 1 Tile   + 1 = 2 Tiles * Bit 1 Attachment orientation   + 0 = Vertical   + 1 = Horizontal * Bit 2 Scale x2 Horizontally   + 0 = Don’t Scale   + 1 = Scale * Bit 3 Scale x2 Vertically   + 0 = Don’t Scale   + 1 = Scale * Bit 4, 5 Sprite Layer   + 00 = Bottom layer   + 11 = Top layer * Bit 6 Not Used * Bit 7 Render This Sprite   + 0 = Don’t Render   + 1 = Render |

**2- The Registers:**

|  |  |
| --- | --- |
| The status register |  |
| The H-Scroll |  |
| The V-Scroll |  |

**3- The Color Palette:**