Donkey Kong Country Sprite Headers

For this document I will try to explain in detail the sprite header format Used by Many Rare games

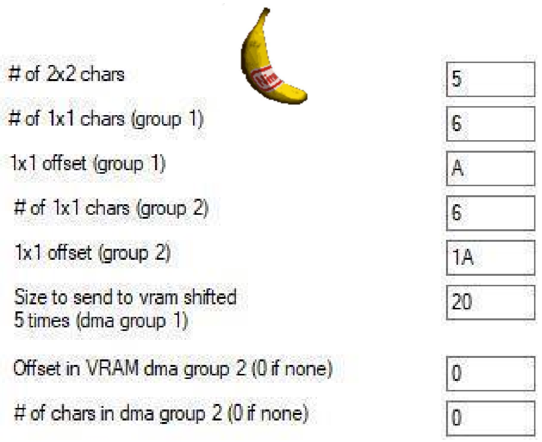
I w ill refer to rainbow sprinklez's DKC1 GFX Editor throughout this document. **Thanks rainbowsprinklez!**

What is a Header?

A sprite header in Donkey Kong Country contains variables and parameters for each sprite and is **0x8 bytes** large followed by the coordinates of each group of tiles. The header format explained here is used by many RARE games.

Large Banana Header

Header as seen in DKC1 GFX Editor...



# of 2x2 chars	5
# of 1x1 chars (group 1)	6
1x1 offset (group 1)	A
# of 1x1 chars (group 2)	6
1x1 offset (group 2)	1A
Size to send to vram shifted 5 times (dma group 1)	20
Offset in VRAM dma group 2 (0 if none)	0
# of chars in dma group 2 (0 if none)	0

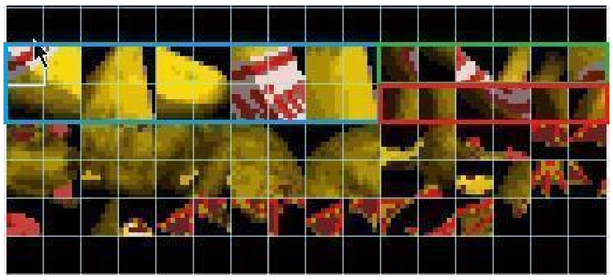
Sprite Address F3C0C6

# of 2x2 chars --> This is the number of 16x16 group of tiles (4 8x8 tiles joined together).

# of 1x1 chars --> The number of 8x8 tiles.

1x1 offset (group 1) --> Where in VRAM to store the single 8x8 tiles. They come after the 16x16 group of tiles or the end of a VRAM row. Each row is 16 wide(0x10 in HEX).

View of VRAM



The cells in blue are the 16x16 tile groups. There are 5 of them. In green are the 8x8 tiles in **1x1 chars (group 1)**. There are 6.

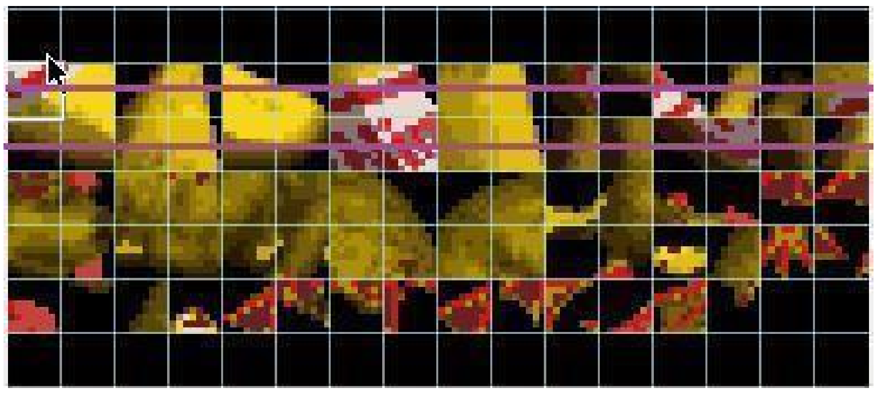
The **1x1 offset (group 1)** is 0xA because the 16x16 groups of tiles take up 10 (0xA) cells in the first VRAM row. In green and red are **1x1 chars group 1 and 1x1 chars group 2.**

The **1x1 offset (group 2)** is 0x1A because the first row that makes up the 16x16s is 10 cells wide + 6 cells wide of the green 8x8s + 10 cells wide of second row of the 16x16 = 26 (1A in hex) Which is the cell offset for **1x1 (offset group 2).**

* Note 16x16 groups of tiles are alw ays displayed like that in VRAM w ith each group taking up 4 cells of 2 VRAM row s (Top Left,Top Right, Bottom Left and Bottom Right). The top halves (in Blue) of the 16x16's w ill take up a portion the first VRAM row and the bottom halves the second row . The next place the group 2's can go is after the 16x16x groups and after group 1 and is 0x1A w hich is the value for the **1x1 (offset group 2)**.

DMA Groups

Where to store groups of tiles in **VRAM.** They start at **0** and go up to the first garbage cell and are stored in this order in the ROM.

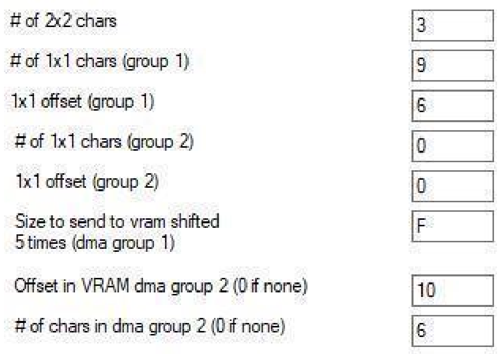


Size to send to vram shifted 5 times (dma group 1) --> In this example **2 VRAM rows** are used with no '**garbage cells**'. Meaning the rows are filled with no space between tile groups and no left over garbage cells. This also means there is only **1 DMA group**. The size of DMA group 1 is 0x20 (which is the size of all char groups). A sprite will always have **at least 1 DMA Group. There is never a DMA Group 3.**

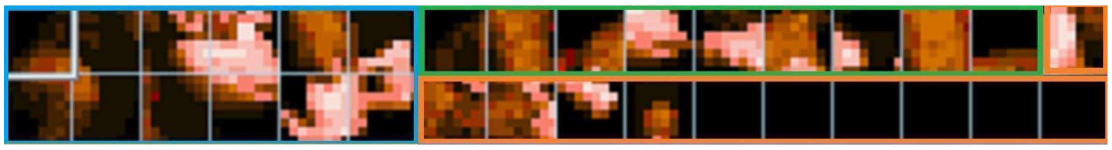
Example # 2

DMA groups in more detail.

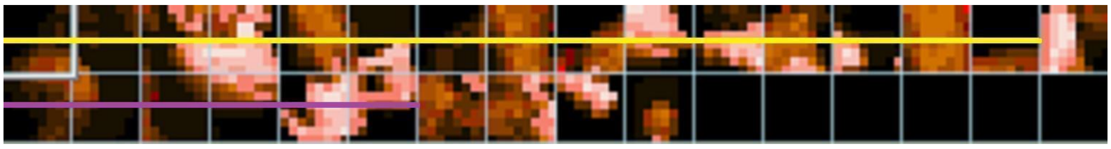
Donkey Kong Sprite Address D8E5D0. Image dc



# of 2x2 chars	3
# of 1x1 chars (group 1)	9
1x1 offset (group 1)	6
# of 1x1 chars (group 2)	0
1x1 offset (group 2)	0
Size to send to vram shifted 5 times (dma group 1)	F
Offset in VRAM dma group 2 (0 if none)	10
# of chars in dma group 2 (0 if none)	6

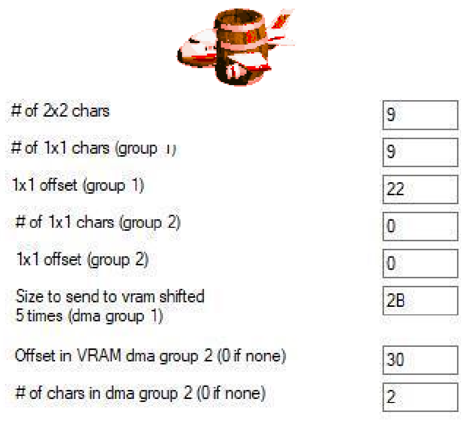


In blue are the 2x2 chars. Green the 1x1 chars group 2. Orange is garbage. Garbage cells are just left over sprites in memory. They are not cleared every frame as that would be a waste of cpu processing and not efficient.

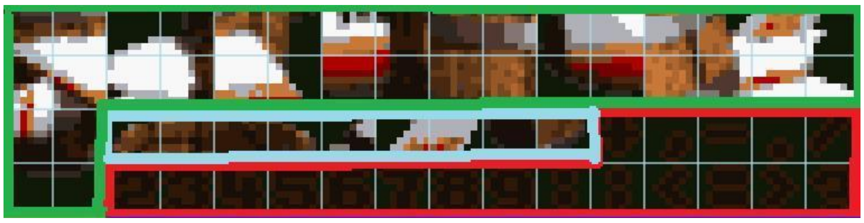


This example has 2 DMA groups. 2 DMA groups are only needed if there are any garbage cells after the first DMA group. DMA Group 1 is in yellow and DMA group 2 is in purple. The first row is **full** however the header specifies **15 8x8 tiles (3 * 4 / 2 + 9 = 0xF or 15 in Dec.** It was divided by 2 because we are only counting the first row in **VRAM**. That means there is 1 garbage cell in the first **VRAM** Row and a second DMA Group is needed. Remember a single char is 8x8 pixels and takes up 1 cell in vram. The example with the Large Banana did not have a 2nd DMA group because all the rows were full with no "garbage tiles".

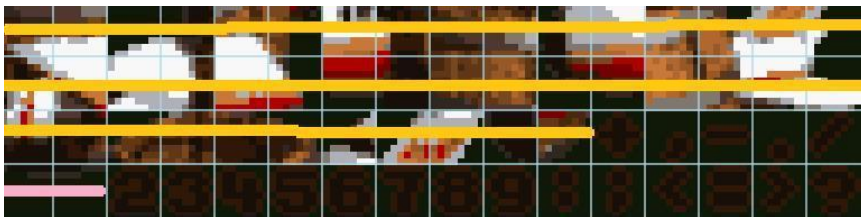
Example # 3



# of 2x2 chars	9
# of 1x1 chars (group 1)	9
1x1 offset (group 1)	22
# of 1x1 chars (group 2)	0
1x1 offset (group 2)	0
Size to send to vram shifted 5 times (dma group 1)	2B
Offset in VRAM dma group 2 (0 if none)	30
# of chars in dma group 2 (0 if none)	2



Green cells are the 16x16 tile group chars. Blue cells are the 8x8 tile chars. Red Cells are garbage.
1x1 offset (group 1) is 0x22 (16 * 2 rows + 2 cells of the top row of the last 16x16 tile group).
The **last** 2 cells (the bottom chars) of the last 16x16 group are stored right after the last cell in blue in the ROM.



A **second DMA group** is needed for this example because the bottom half of the 9th 16x16 group would come after the last 8x8 and would be then be "garbage". The second DMA group places the last 2 bottom 8x8 chars of the 9th group on row 4 in VRAM and are stored in this order in **ROM**.

dma group 1 is in yellow and is 0x2b in size.
dma offset for DMA group 2 is 0x30 which specifies which row to start the DMA Group 2.
# of chars in DMA group 2 is in pink and is 0x2 in size.

How The Header and Sprites Are Stored in the ROM * Note this section is incomplete and needs work.

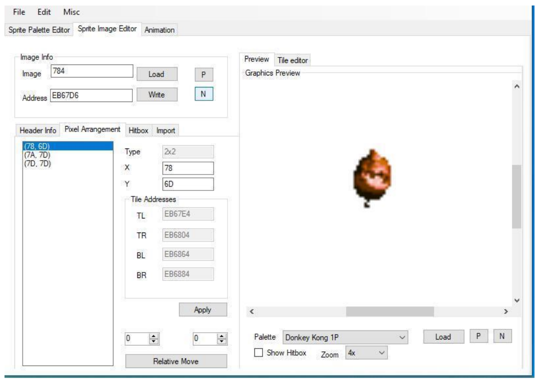

File Edit Misc
Sprite Palette Editor | Sprite Image Editor | Animation

Image Info
Image: 784
Address: EB67D6

Header Info | Pixel Arrangement | Hitbox | Import

(78, 6D)
(7A, 7D)
(7D, 7D)

Type: 2x2
X: 78
Y: 6D

Tile Addresses
TL: EB67E4
TR: EB6804
BL: EB6864
BR: EB6884

Palette: Donkey Kong 1P
Zoom: 4x

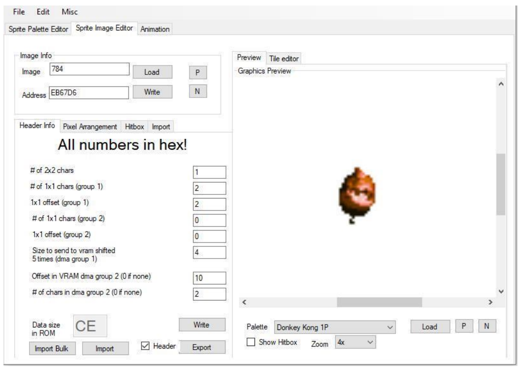

File Edit Misc
Sprite Palette Editor | Sprite Image Editor | Animation

Image Info
Image: 784
Address: EB67D6

Header Info | Pixel Arrangement | Hitbox | Import

All numbers in hex!

of 2x2 chars: 1
of 1x1 chars (group 1): 2
1x1 offset (group 1): 2
# of 1x1 chars (group 2): 0
1x1 offset (group 2): 0
Size to send to vram shifted 5 times (dma group 1): 4
Offset in VRAM dma group 2 (0 if none): 10
# of chars in dma group 2 (0 if none): 2

Data size in ROM: CE
[✓ Header]

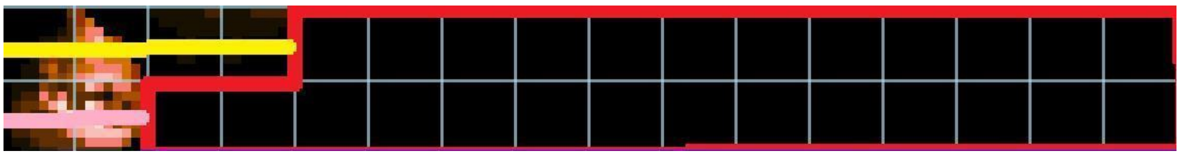
Palette: Donkey Kong 1P
Zoom: 4x

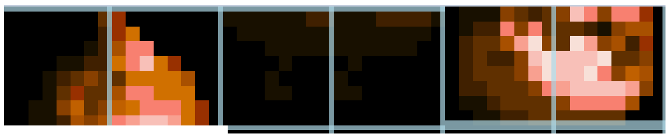
```
002B67B8  00 00 00 00 00 00 04 04 0E 1A 00 00 00 00 00 00 00 00 00 00 00 00 04 00 1E 00 00 00 00 00
002B67D6  01 02 02 00 00 04 10 02 78 6D 7A 7D 7D 7D 01 01 00 01 02 01 0C 03 15 0D 1B 0F 32 06 35 08
002B67F4  00 00 00 00 00 00 00 00 02 00 00 04 0D 00 07 00 00 80 40 C0 E0 60 F0 D8 FC F8 FC 7C 7C FE
```

Highlighted in yellow are the **8 byte** addresses of the header parameters.

```
002B67D6  01 02 02 00 00 04 10 02 78 6D 7A 7D 7D 7D 01 01 00 01 02 01 0C 03 15 0D 1B 0F 32 06 35 08
```

The above addresses are the **coordinates** of each group of tiles. Two bytes per group. **X & Y**. They are in order with the coordinates of the 2x2 chars first then followed by the 1x1 chars. The addresses of each tile varies according to the DMA Groups. A single 16x16 sprite tile set will be in **top left, top right, bottom left and bottom right order.** Below is a different example of how the tiles are stored. Remember Tiles are stored in ROM according to the DMA Groups and the first DMA group will be stored first then the second group after that. In red are garbage cells and are not stored in the ROM. So...





The order in ROM is...

Top Left, Top Right (16x16 group), then 1st and 2nd (8x8 groups)

and Bottom Left and Bottom Right of the 16x16 group.

Conclusion

I hope this wasn't too confusing for you. I did my best in making this as straight forward as possible however there may be mistakes!!! And is in no way complete.
If you have any comments you can email me at Cyclone.Chris@gmail.com
or visit the DKC-Atlas.com forums.

Credits

Cyclone (That's me who made this Doc ;))
rainbowsprinklez (DKC-Atlas Forums) <- Much credit goes to rainbow sprinklez for the help in understanding the material covered in this document and for letting me use his know ledge. **https://www.romhacking.net/utilities/1756/**
Kingizor (DKC-Atlas forums)
Mattrizzel (DKC-Atlas forums)

Last Updated July 20th 2023