C256 – Developer Introduction Notes

# Git Repositories

Nu256 Emulator: <https://github.com/tomxp411/Nu256.git>

C256 Kernel: <https://github.com/Trinity-11/Kernel>

# Tools

To modify the Nu256 Emulator, you will need Visual Studio 2017 Community edition.

The emulator is written in C#.

# Memory Map

The CPU can access 24-bit worth of addresses. This doesn’t mean that there is writable memory here.

|  |  |  |
| --- | --- | --- |
| $FF:0000 - $FF:FFFF | Page $FF | **16 MB Address Space** |
| $FE:0000 - $FE:FFFF | Page $FE |
|  |  |
|  |  |
|  |  |
|  |  |
| $00:0000 - $01:FFFF | Page $01 |
| $00:0000 - $00:FFFF | Page $00 |

The address space is mapped as follows:

|  |  |
| --- | --- |
| $F8:0000 - $FF:FFFF | 512 KB User Flash (if populated) |
| $F0:0000 - $F7:FFFF | 512 KB System Flash |
| $B0:0000 - $EF:FFFF | 4 MB Video RAM |
| $AF:0000 - $AF:FFFF | IO Space |
| $40:0000 - $AE:FFFF | <empty> |
| $20:0000 - $3F:FFFF | 2 MB RAM (optional) |
| $00:0000 - $1F:FFFF | 2 MB RAM |

On boot, Gavin copies the content of System Flash (or User Flash, if present) to address range $18:0000 to $1F:FFFF.

SCREEN\_PAGE0 = $AF:A000 <- Text

SCREEN\_PAGE1 = $AF:C000 <-Color

FONT\_MEMORY\_BANK = $AF:8000 - $AF:BFFF

The 15 colours are looked up in the 24-bit LUT.

FG\_CHAR\_LUT\_PTR = $AF1F40

BG\_CHAR\_LUT\_PTR = $AF1F80

You assign an 8 Bit - R, 8 Bit - G, 8 - B for each of those colors in Text Mode

This is vicky\_def.asm by the way... I know there is no manual yet... One day, there will be and it is all going to be in there...

So, let me explain how the text buffer works

The overall buffer size is 128 Characters x 64 Characters

But what is visible is only a fraction of that

We’re only displaying a fraction of the total 128 x 64 character array.

72 x 56.

The display resolution is always 640 x 480

but with the border, it goes down to 576 x 448

but we can change the size of the border or turn it off