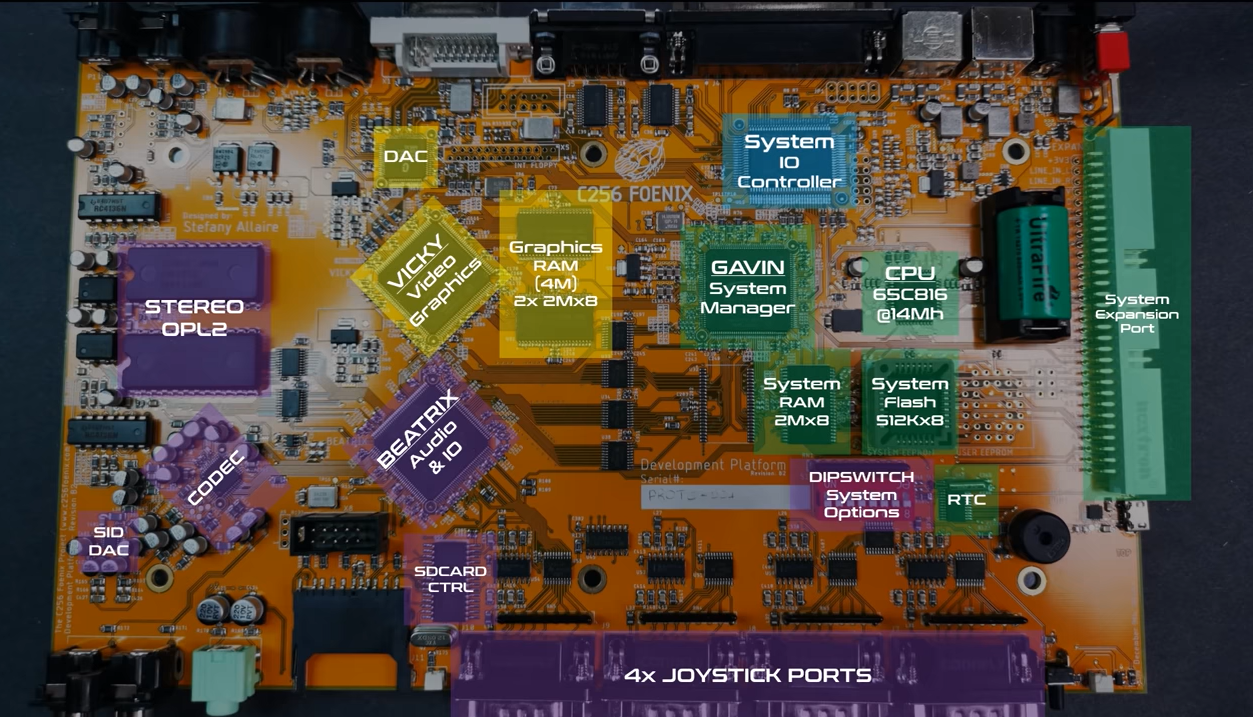
C256 – Developer Introduction Notes

# Introduction to the C256 System

1. C256 System Board PCB layout
2. CPU Sub-System
3. Video Sub-system
4. Audio Sub-system
5. Front Panel Inputs and Outputs
6. Back Panel Inputs and Outputs
7. System Inputs and Outputs
8. Final (almost) Specifications
9. WDC 65C816 Memory
10. Global memory Model
11. Gavin
12. Vicky Beatrix
13. Gavin
14. Vicky
15. Beatrix
16. Flash Boot Process
17. Software Tools and GIT Repository Information

# 1 - C256 System Board PCB layout

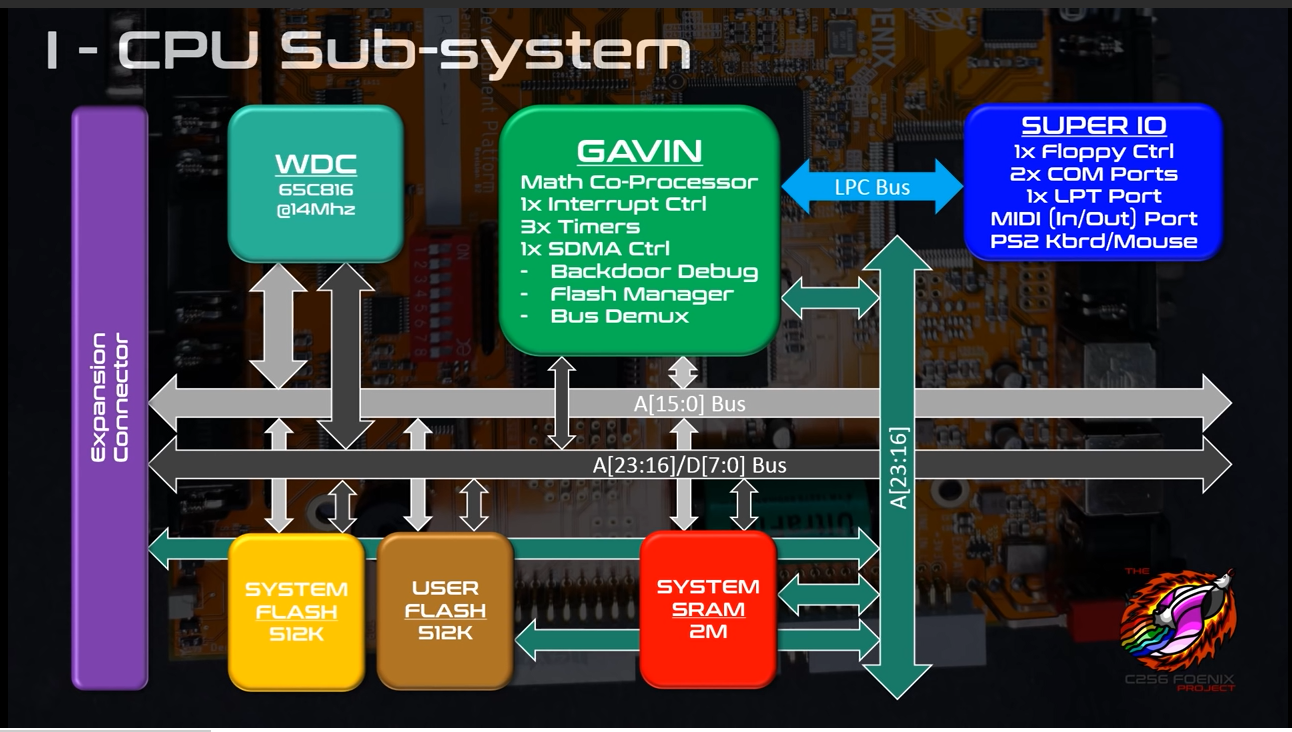


## CPU - WD-65C816 @ 14mHz

* 1. System flash, 512kx8, kernel etc.
     1. Optional user flash footprint to the right
  2. System RAM, static 2Mx8
  3. RTC – real time clock
  4. Expansion bus, external IO expansion possibilities

1. GAVIN – Custom FPGA, System Manager, connects to System IO controller on separate bus
2. System IO Controller
3. VICKY – Custom FPGA, Video Graphics
   1. Graphics RAM, 2x 2Mx8 (4M total)
   2. DAC
4. BEATRIX – Custom FPGA, Audio and IO (joysticks, dipswitches)
   1. Joystick ports – 4x
   2. SD Card Controller –
   3. Stereo OPL2 –
   4. CODEC – 24bit ADC/capture
   5. SID DAC – for FPGA SID

# 2 - CPU Sub-system



## WDC 65C816 @14Mhz

## GAVIN – Custom FPGA

## Math Co-Processor

* 1. Interrupt control
  2. SDMA control
     1. Backdoor debug
     2. (boot) Flash Manager
     3. Bus Demux
     4. System DMA controller

1. SuperIO
   1. Floppy Ctrl
   2. 2x COM Ports
   3. 1x LPT Port
   4. MIDI In/Out
   5. PS2 Mouse/Keyboard

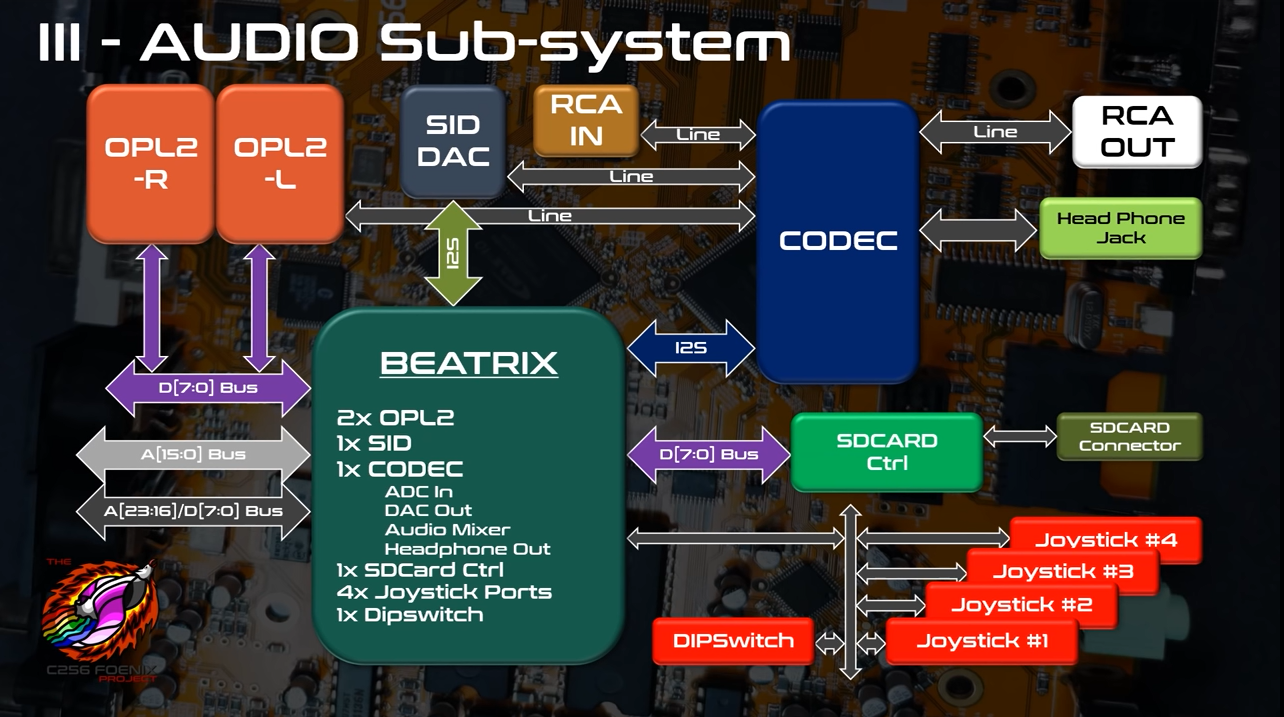
# 3 - Video Sub-system

## 

## VICKY – Custom FPGA

* 1. 200MHz some parts, 100MHz video memory
  2. Text mode
  3. Graphics modes
  4. Video DMA built in
  5. DVI and Analog outputs

# 4 - Video Sub-system



## BEATRIX – Custom FPGA

* 1. 2x OPL2
  2. 1x SID
  3. 1x CODEC – mixer/sampler

1. RCA IN/OUT
2. Provides interface for
   1. 4x Joysticks
   2. SD Card Control
   3. System Configuration DIP Switches

# 5 - Front Panel Inputs and Outputs

## 

## Audio – RCA Line-in

1. Audio – Headphone out
2. SD Card Port and Activity LED
3. FPGA JTAG
4. 4x Joystick Ports
5. Reset button
6. Power LED
7. Buzzer
8. RTC Battery

# 6 - Back Panel Inputs and Outputs

# 

## Power switch

1. Power input jack, 12V 1A, 2.1mm
2. PS2 Mouse/Keyboard inputs
3. LPT Port
4. COM 1 Serial Port
5. Video Output
   1. DVI-D and Analog
6. MIDI Input/output
   1. RCA Line Out (Stereo)

# 7 - Inputs and Outputs

## Expansion Connector

1. Backdoor debugger (eventually)
   1. Serial interface to GAVIN to control system when debugging
   2. Can download/upload, stop processor
   3. Debugger, breakpoint CPU
2. PS2 Mouse/Keyboard inputs

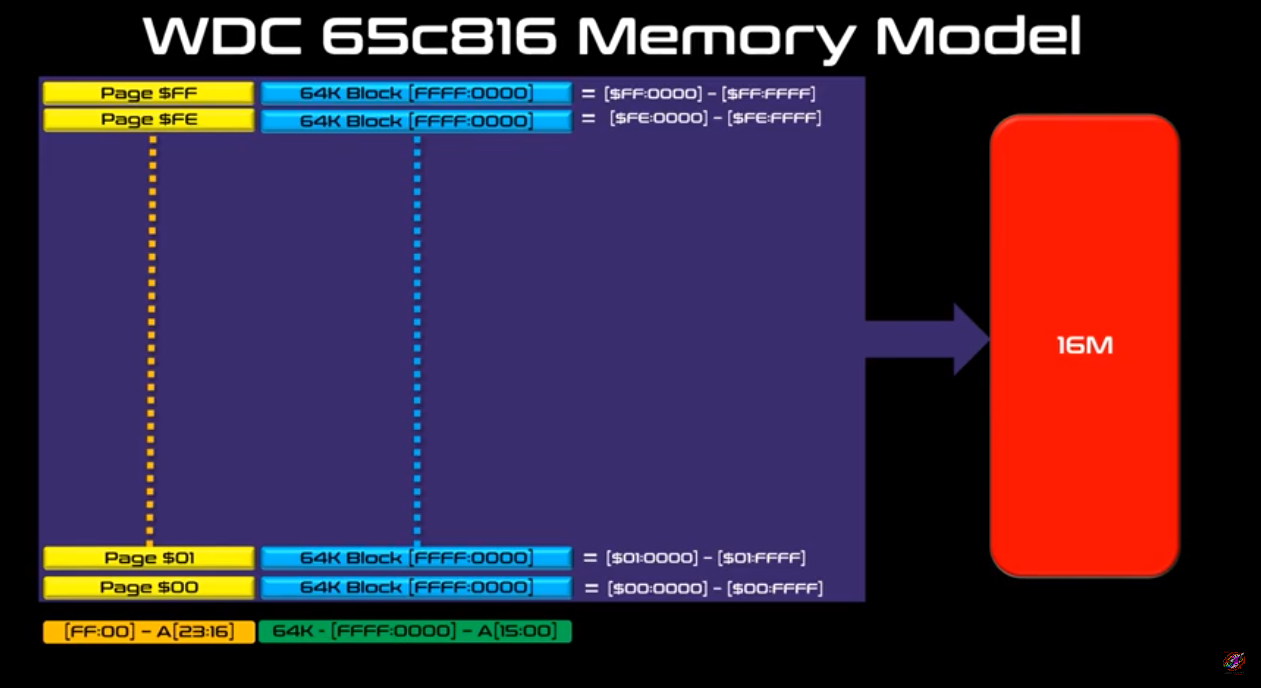
# 8 - Final (almost) Specifications



## Expansion Connector

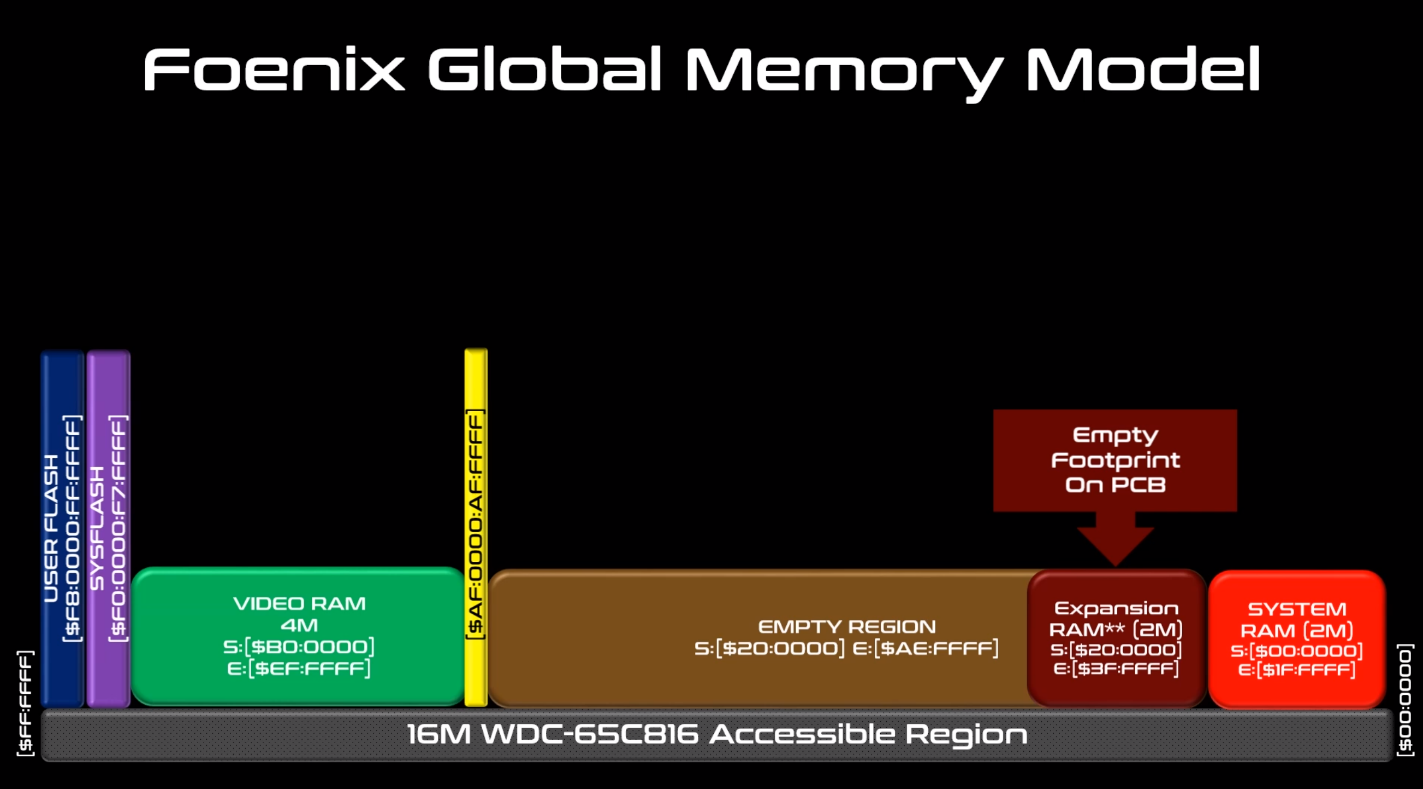
1. Backdoor debugger (eventually)
   1. Serial interface to GAVIN to control system when debugging
   2. Can download/upload, stop processor
   3. Debugger, breakpoint CPU
2. PS2 Mouse/Keyboard inputs

# 9 - WDC 65C816 Memory Model



1. Maximum memory is 16 MB
2. Memory pages in blocks of 64K
3. Processor runs at maximum specified speed of 14.318 MHz
4. Data bus multiplexed to provide page number on first half of cycle
   1. About 70ns clock period, first 35ns data buss multiplex, second access time
   2. Access time is more like 28MHz processor, need fast memory, i.e. 35ns or faster

# 10 - Global Memory Map



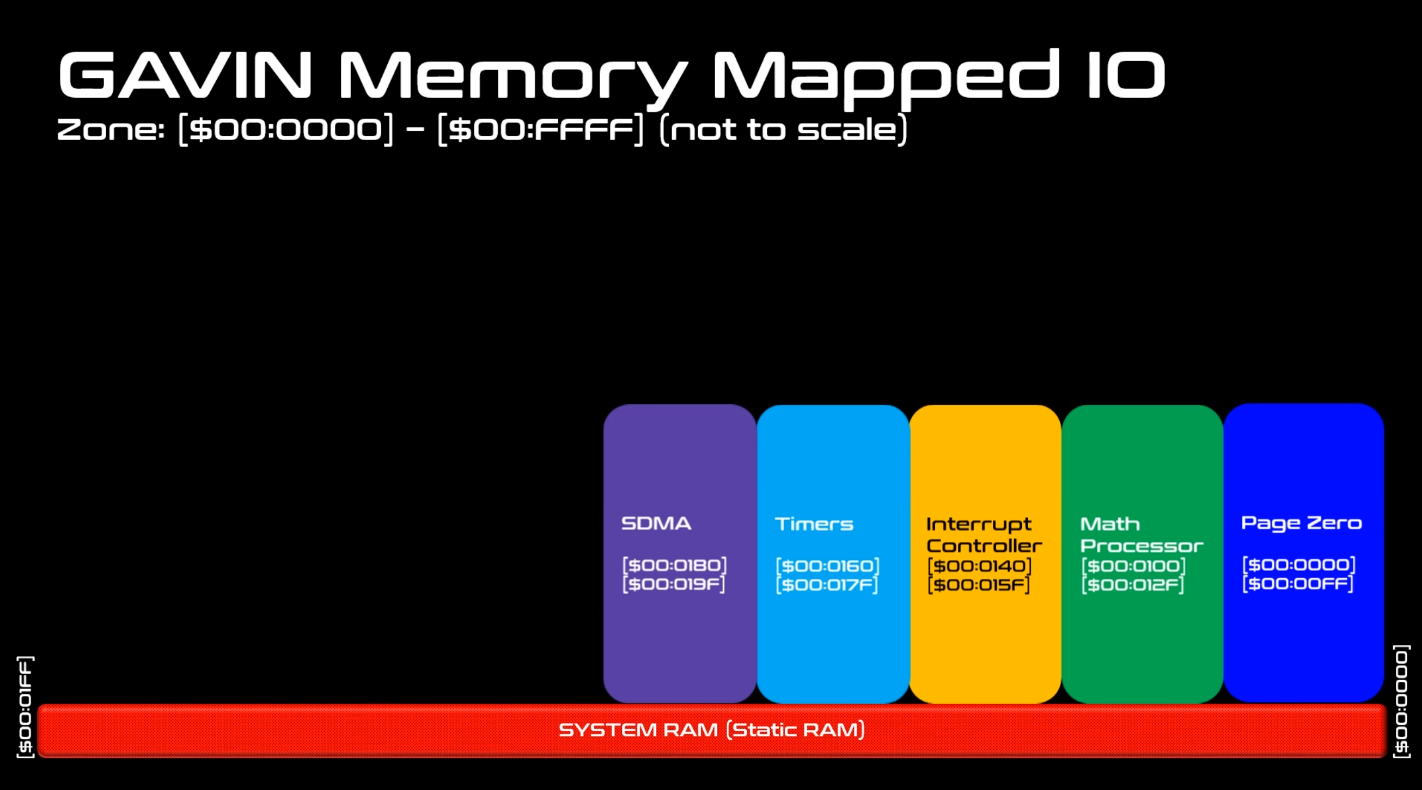
Global address space broken up into 64K pages.

|  |  |  |
| --- | --- | --- |
| $FF:0000 - $FF:FFFF | Bank $FF | **16 MB Address Space** |
| $FE:0000 - $FE:FFFF | Bank $FE |
|  |  |
|  |  |
|  |  |
|  |  |
| $00:0000 - $01:FFFF | Bank $01 |
| $00:0000 - $00:FFFF | Bank $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 - $B3:FFFF | 4 MB Video RAM |
| $AF:???? - $AF:???? | (BEATRIX) CODEC DAC FIFO - TBD |
| $AF:???? - $AF:???? | (BEATRIX) CODEC ADC FIFO - TBD |
| $AF:E820 - $AF:E823 | (BEATRIX) CODEC registers |
| $AF:E808 - $AF:E81F | (BEATRIX) SD Card Controller |
| $AF:E804 - $AF:E804 | (BEATRIX) I/O for DIP Switches |
| $AF:E800 - $AF:E803 | (BEATRIX) I/O for Joysticks |
| $AF:E700 - $AF:E7FF | (BEATRIX) OPL2 – Both, Write Only |
| $AF:E600 - $AF:E6FF | (BEATRIX) OPL2 - Left |
| $AF:E500 - $AF:E5FF | (BEATRIX) OPL2 - Right |
| $AF:E400 - $AF:E4FF | (BEATRIX) SID registers |
| $AF:C000 - $AF:DFFF | (VICKY) screen page 1, Additional RAM, page flipping etc. |
| $AF:A000 - $AF:BFFF | (VICKY) Text memory, screen page 0 |
| $AF:8800 - $AF:88FF | (VICKY) FONT1 Memory Bank |
| $AF:8000 - $AF:87FF | (VICKY) FONT0 Memory Bank |
| $AF:5000 - $AF:6FFF | (VICKY) Tile maps |
| $AF:4000 - $AF:43FF | (VICKY) Gamma LUT |
| $AF:2000 - $AF:3FFF | (VICKY) Graphics LUT |
| $AF:1F00 - $AF:1F7F | (VICKY) Text Color LUT |
| $AF:1000 - $AF:13FF | (VICKY) LPC (SuperIO) |
| $AF:0800 - $AF:080F | (VICKY) RTC |
| $AF:0400 - $AF:04FF | (VICKY) Video DMA |
| $AF:0300 - $AF:03FF | (VICKY) Sprites registers |
| $AF:0140 - $AF:014F | (VICKY) Bitmap registers |
| $AF:0100 - $AF:013F | (VICKY) Tiles registers |
| $AF:0000 - $AF:00FF | (VICKY) General registers |
| $40:0000 - $AE:FFFF | <empty> |
| $00:0180 - $00:019F | (GAVIN) SDMA, system DMA |
| $00:0160 - $00:017F | (GAVIN) Timers |
| $00:0140 - $00:015F | (GAVIN) Interrupt Controller |
| $00:0100 - $00:012F | (GAVIN) Math Coprocessor |
| $00:0000 - $00:00FF | Zero page |
| $20:0000 - $3F:FFFF | 2 MB RAM (optional) |
| $00:0000 - $1F:FFFF | 2 MB RAM |

## 11 - Gavin – Location $00:0000 to $00:1FFF



### Page Zero $00:000 to $00:00FF

The GAVIN includes the 256 bytes of zero page memory.

### Math Co-Processor $00:0100 to $00:012F

The C256 provides a math co-processor to perform long addition, multiplications and divisions. To perform an operation, you write the little-endian values in the appropriate address locations and the results are automatically returned in the result addresses.

#### Multiplications

There are two multiplier locations: $00:0100 and $00:0108. Multiplier 0 is unsigned and Multiplier 1 is signed. Each operand must be 16-bits, and the result is 32-bits.

|  |  |  |
| --- | --- | --- |
| **Address** | **Name** | **Description** |
| $00:0100 | M0\_OPERAND\_A | 16-bit unsigned value |
| $00:0102 | M0\_OPERAND\_B | 16-bit unsigned value |
| $00:0104 | M0\_RESULT | 32-bit unsigned result of the multiplication of A and B |

|  |  |  |
| --- | --- | --- |
| **Address** | **Name** | **Description** |
| $00:0108 | M1\_OPERAND\_A | 16-bit signed value |
| $00:010A | M1\_OPERAND\_B | 16-bit signed value |
| $00:010C | M1\_RESULT | 32-bit signed result of the multiplication of A and B |

#### Divisions

There are two divider locations: $00:0110 and $00:0118. Divider 0 is unsigned and Divider 1 is signed. Each operand must be 16-bits. The result and remainder are 16-bits also.

|  |  |  |
| --- | --- | --- |
| **Address** | **Name** | **Description** |
| $00:0110 | D0\_OPERAND\_A | 16-bit unsigned value for the dividend |
| $00:0112 | D0\_OPERAND\_B | 16-bit unsigned value for the divisor |
| $00:0114 | D0\_RESULT | 16-bit unsigned result for the quotient |
| $00:0116 | D0\_REMAINDER | 16-bit unsigned result for the remainder |

|  |  |  |
| --- | --- | --- |
| **Address** | **Name** | **Description** |
| $00:0118 | D1\_OPERAND\_A | 16-bit signed value for the dividend |
| $00:011A | D1\_OPERAND\_B | 16-bit signed value for the divisor |
| $00:011C | D1\_RESULT | 16-bit signed result for the quotient |
| $00:011E | D1\_ REMAINDER | 16-bit signed result for the remainder |

#### Long Signed Additions

There is one long signed adder located at $00:0120. Both operands must be 32-bit signed integers. The result is also 32-bit signed.

|  |  |  |
| --- | --- | --- |
| **Address** | **Name** | **Description** |
| $00:0120 | ADDER32\_A | 32-bit signed value |
| $00:0124 | ADDER32\_B | 32-bit signed value |
| $00:0128 | ADDER32\_R | 32-bit signed result of the addition of A and B |

### Interrupt Controller $00:0140 to $00:014E

The 65C816 has four interrupt input pins  
RESET -> a reset signal, level-triggered  
NMI -> a non-maskable interrupt, edge-triggered  
IRQ -> a maskable interrupt, level-triggered (wired up to GAVIN)  
ABORT -> a special-purpose, non-maskable interrupt level-triggered

GAVIN acts as an interrupt controller. Interrupt signals from the various chips are fed to GAVIN’s interrupt input pins. When an interrupt input signal is received the corresponding Interrupt Pending Register bit will be set. GAVIN will send an IRQ signal to the CPU unless the corresponding input is masked in the Interrupt Masked Register; if the bit is set for the input it is masked.

At this time, the Polarity (INT\_POL\_REG#) and Edge (INT\_EDGE\_REG#) are not currently used. An IRQ is generated by setting the corresponding INT\_REG# bit unless the corresponding mask bit is set in INT\_MASK\_REG#.

There are two ways you can use the Interrupts:

1. Either you let the IRQ being triggered by your Device and you process some code in the IRQ Handler
2. Or you can Poll the Interrupt from the Pending Register INT\_PENDING\_REG0/INT\_PENDING\_REG1/INT\_PENDING\_REG2 without triggering an actual interrupt

You would set the masked bit in the corresponding interrupt register to prevent the interrupt controller from generating an IRQ. Then you would poll the Pending Register and act on the fact that an Interrupt has been received. Note that you need to clear the interrupt pending register by writing back to it a bitmask of the bit you want cleared. You can do that at the beginning or at the end of the code you are trying to process.

### Interrupt Controller Registers

|  |  |  |
| --- | --- | --- |
| **Address** | **Name** | **Description** |
| $00:0140 | INT\_PENDING\_REG0 | Interrupt Pending Resister 0, bit 1 = interrupt pending |
| $00:0141 | INT\_PENDING\_REG1 | Interrupt Pending Resister 1, bit 1 = interrupt pending |
| $00:0142 | INT\_PENDING\_REG2 | Interrupt Pending Resister 2, bit 1 = interrupt pending |
| $00:0144 | INT\_POL\_REG0 | Interrupt Polarity Reg 0 (not used), bit = 1 indicates active high input |
| $00:0145 | INT\_POL\_REG1 | Interrupt Polarity Reg 1 (not used), bit = 1 indicates active high input |
| $00:0146 | INT\_POL\_REG2 | Interrupt Polarity Reg 2 (not used), bit = 1 indicates active high input |
| $00:0148 | INT\_EDGE\_REG0 | Interrupt Edge Triggered Reg 0 (not used), bit=1 indicates edge triggered |
| $00:0149 | INT\_EDGE\_REG1 | Interrupt Edge Triggered Reg 1 (not used), bit=1 indicates edge triggered |
| $00:014A | INT\_EDGE\_REG2 | Interrupt Edge Triggered Reg 2 (not used), bit=1 indicates edge triggered |
| $00:014C | INT\_MASK\_REG0 | Interrupt Mask Resister 0, bit 1 = interrupt masked |
| $00:014D | INT\_MASK\_REG1 | Interrupt Mask Resister 1, bit 1 = interrupt masked |
| $00:014E | INT\_MASK\_REG2 | Interrupt Mask Resister 2, bit 1 = interrupt masked |

### Interrupt Pending Register 0

|  |  |  |
| --- | --- | --- |
| **Bit** | **Name** | **Description** |
| 0 | ALWAYS1 | Not used Always 1 |
| 1 | VICKY\_INT0 | Start of Frame (for now) |
| 2 | VICKY\_INT1 | Line Interrupt (For Split Screen) |
| 3 | Timer0 | Timer0 |
| 4 | Timer1 | Timer1 |
| 5 | Timer2 | Timer2 |
| 6 | RTC Interrupt | Clock Alarm, etc..., See the chip spec for detail for the registers: BQ4802 |
| 7 | LPC\_INT[6] | Floppy Disk Controller |

### Interrupt Pending Register 1

|  |  |  |
| --- | --- | --- |
| **Bit** | **Name** | **Description** |
| 0 | LPC\_INT[1] | Keyboard Interrupt (8042) |
| 1 | VICKY\_INT2 | Sprite Collision |
| 2 | VICKY\_INT3 | Tile Collision (TBD) |
| 3 | LPC\_INT[3] | Line Interrupt (For Split Screen) |
| 4 | LPC\_INT[4] | Timer0 |
| 5 | LPC\_INT[5] | Timer1 |
| 6 | LPC\_INT[7] | Timer2 |
| 7 | SDCARD | SD Card |

### Interrupt Pending Register 2

|  |  |  |
| --- | --- | --- |
| **Bit** | **Name** | **Description** |
| 0 | OPL2\_Right | OPL2\_Right\_Channel |
| 1 | OPL2\_Left | OPL2\_Left\_Channel |
| 2 | Beatrix Interrupt | Beatrix Interrupt (TBD) |
| 3 | Gavin DMA | Gavin DMA Controller |
| 4 | Always1 | Not used Always 1 |
| 5 | DAC - Hot-Plug | DAC - Hot-Plug Interrupt |
| 6 | Expansion Port | Expansion Port Connector Interrupt |
| 7 | Always1 | Not used Always 1 |

## SuperIO Registers Location $AF:1060 to $AF:13FF

VICKY has an LPC47M10x ‘SuperIO’ chip mapped into its address space. This chip provides a number of peripherals including PS2 keyboard/mouse, game port, serial ports, parallel port, floppy controller and MIDI input/output. Even though the SuperIO is now interfaced through GAVIN the registers remain mapped in VICKY’s address space as this is where it resided in V1. This may change in V3. Please see the section on GAVIN for more information.

|  |  |
| --- | --- |
| **Address** | **Description** |
| $AF:1060 - $AF:1064 | LOGIC DEVICE 7 – KEYBOARD/MOUSE (SuperIO) |
| $AF:1100 - $AF:117F | LOGIC DEVICE A - PME (Runtime Registers) |
| $AF:1200 - $AF:1200 | LOGIC DEVICE 9 - GAME PORT |
| $AF:12F8 - $AF:12FF | LOGIC DEVICE 5 - SERIAL 2 |
| $AF:1330 - $AF:1331 | LOGIC DEVICE B - MPU-401 (MIDI) |
| $AF:1378 - $AF:137F | LOGIC DEVICE 3 - PARALLEL PORT |
| $AF:13F0 - $AF:13F7 | LOGIC DEVICE 0 - FLOPPY CONTROLLER |
| $AF:13F8 - $AF:13FF | LOGIC DEVICE 4 - SERIAL 1 |

## Keyboard Registers Location $AF:1060 to $AF:1064

|  |  |  |  |
| --- | --- | --- | --- |
| **Address** | **Command** | **Block** | **Function, 3 separate 8-bit registers, Status, Data/Command, Data Read** |
| $AF:1060 | Write | KDATA | Keyboard Data Write (C/D=0) |
|  | Read | KDATA | Keyboard Data Read |
| $AF:1064 | Write | KDCTL | Keyboard Command Write (C/D=1) |
|  | Read | KDCTL | Keyboard Status Read |

Host I/F Data Register  
The Input Data register and Output Data register ($AF:1060) are each 8-bits wide. A write into this 8-bit register will load the keyboard Data Read Buffer, set the OBF flag and set the KIRQ output if enabled. A read of this register will read the data form the Keyboard Data or Command Write Buffer and clear the IBF flag.

## Host I/F Status Register - This register is cleared on reset. This register is read only by host.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| UD | UD | UD | UD | C/D | UD | IBF | OBF |

UD Writeable by LPC47M10x. These bits are user definable.

C/D (Command Data) This bit specifies whether the inp0ut data register contains data or a command (0=data, 1=command). During a host data/command write operation this bit is set to “1” if SA2=1 or reset to “0” if SA2=0.

IBF (Input Buffer Full) This flag is set to 1 whenever the host system writes data into the input data register. Setting this flag activates the LPC47M10x CPU’s nIBF (MIRQ) interrupt if enabled. When the LPC47M10x CPU reads the input data register (DBB), this bit is automatically reset and the interrupt is cleared. There is no output pin associated with this internal signal.

OBF (Output Buffer Full) This flag is set to whenever the LPCM10c CPU write to the output data register (DBB). When the host system reads the output data register, this bit is automatically reset.

<https://wiki.osdev.org/%228042%22_PS/2_Controller>

## Floppy Drive Registers Location $??:???? to $??:????

|  |  |  |  |
| --- | --- | --- | --- |
| **Primary Address** | **Secondary Address** | **R/W** | **Register** |
| $03F0 | $0370 | R | Status Register A (SRA) |
| $03F1 | $0371 | R | Status Register B (SRB) |
| $03F2 | $0372 | R/W | Digital Output Register (DOR) |
| $03F3 | $0373 | R/W | Tape Drive Register (TSR) |
| $03F4 | $0374 | R | Main Status Register (MSR) |
| $03F4 | $0374 | W | Data Rate Select Register (DSR) |
| $03F5 | $0375 | R/W | Data (FIFO) |
| $03F6 | $0376 |  | Reserved |
| $03F7 | $0377 | R | Digital Input Register (DIR) |
| $03F7 | $0377 | W | Configuration Control Register (CCR) |

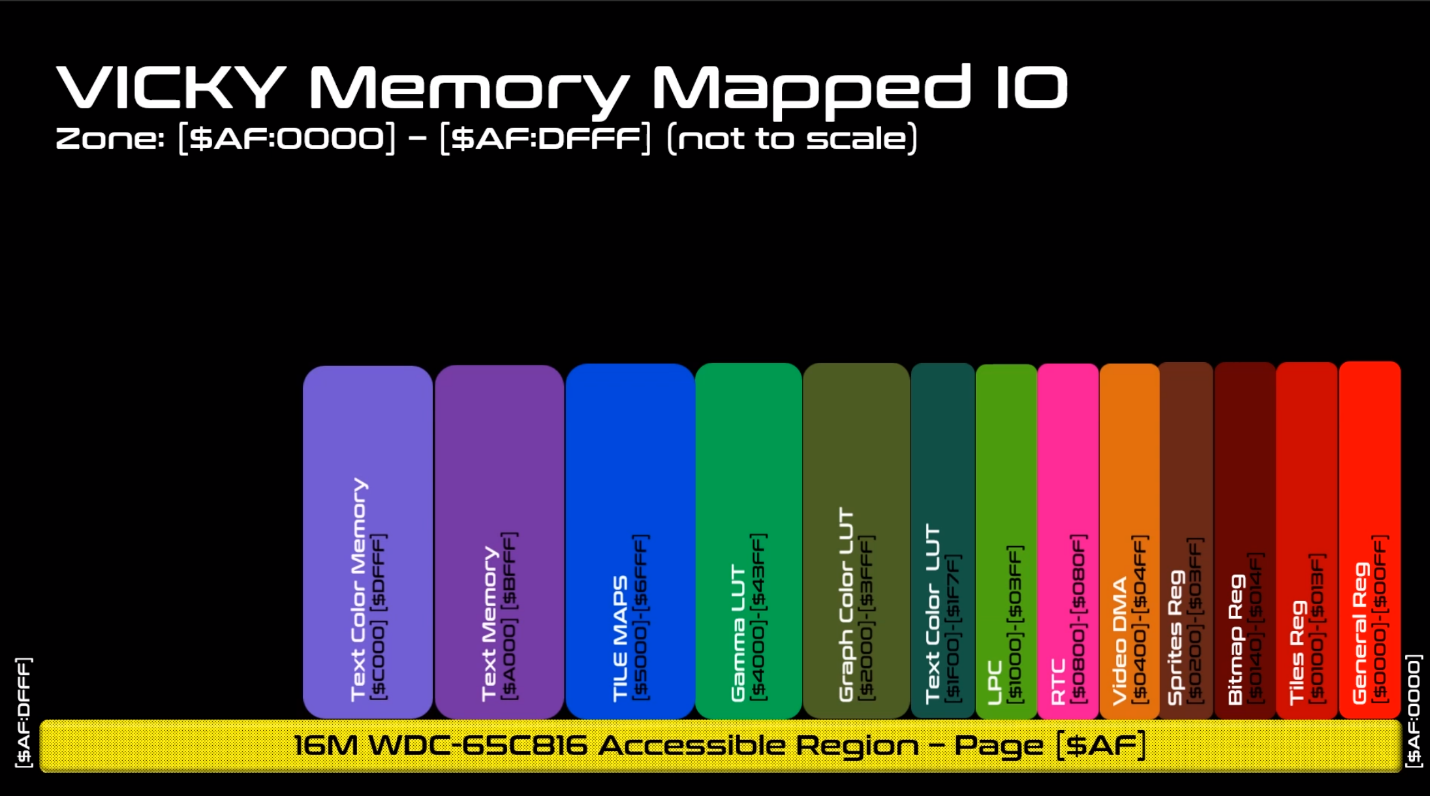
## Serial Port Registers Location $??:???? to $??:????

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DLAB\*** | **A2** | **A1** | **A0** | **Register Name** |
| 0 | 0 | 0 | 0 | Receiver Buffer (read) |
| 0 | 0 | 0 | 0 | Transmit Buffer (write) |
| 0 | 0 | 0 | 1 | Interrupt Enable (read/write) |
| X | 0 | 1 | 0 | Interrupt Identification (read) |
| X | 0 | 1 | 0 | FIFO Control (write) |
| X | 0 | 1 | 1 | Line Control (read/write) |
| X | 1 | 0 | 0 | Modem Control (read/write) |
| X | 1 | 0 | 1 | Line Status (read/write) |
| X | 1 | 1 | 0 | Modem Status (read/write) |
| X | 1 | 1 | 1 | Scratchpad (read/write) |
| 1 | 0 | 0 | 0 | Divisor LSB (read/write) |
| 1 | 0 | 0 | 1 | Divisor MSB (read/write) |

## MIDI Registers Location $??:???? to $??:????

|  |  |  |  |
| --- | --- | --- | --- |
| **Address** | **Name** | **R/W** | **Description** |
| $???? | MIDI Data | R/W | Used for MIDI transmit/receive data and command ack. |
| $???? | Status | R | Used to indicate the send/receive status of MIDI Data port. |
| $???? | Command | W | Used for MPU-401 commands |

## 12 - Vicky – Location $AF:0000 to $AF:DFFF



### VICKY Address Space

|  |  |
| --- | --- |
| **Address** | **Description** |
| $AF:0000 - $AF:00FF | (Internal Memory) Vicky General Registers |
| $AF:0100 - $AF:013F | (Internal Memory) Vicky Tiles Registers |
| $AF:0140 - $AF:014F | (Internal Memory) Vicky Bitmap Registers |
| $AF:0200 - $AF:03FF | (Internal Memory) Vicky Sprites |
| $AF:0400 - $AF:04FF | (Internal Memory) Vicky VDMA |
| $AF:0800 - $AF:080F | Real-time clock (RTC) |
| $AF:1000 - $AF:13FF | SuperIO, actually wired to GAVIN |
| $AF:1F00 - $AF:1F3F | (Internal Memory) Vicky Text Mode 16 Color Look-up Table Foreground Color |
| $AF:1F40 - $AF:1F7F | (Internal Memory) Vicky Text Mode 16 Color Look-up Table Background Color |
| $AF:2000 - $AF:23FF | (Internal Memory) Graphic Mode LUT0 |
| $AF:2400 - $AF:27FF | (Internal Memory) Graphics Mode LUT1 |
| $AF:2800 - $AF:2BFF | (Internal Memory) Graphics Mode LUT2 |
| $AF:2C00 - $AF:2FFF | (Internal Memory) Graphics Mode LUT3 |
| $AF:4000 - $AF:40FF | (External Memory) 256 Bytes GAMMA LUT - RED |
| $AF:4100 - $AF:41FF | (External Memory) 256 Bytes GAMMA LUT - GREEN |
| $AF:8000 - $AF:8FFF |  |
| $AF:9000 - $AF:9FFF | FONT\_MEMORY\_BANK1 |

### Vicky Address Space

|  |
| --- |
| $AF:0000 - $AF:00FF (Internal Memory) Vicky General Registers |
| BORDER\_COLOR\_B = $AF:0005 - when in text mode, this is the border color shown.  BORDER\_COLOR\_G = $AF:0006  BORDER\_COLOR\_R = $AF:0007 |
| BACKGROUND\_COLOR\_B = $AF:000D - When in Graphic Mode, if a pixel is "0" then the Background pixel is chosen  BACKGROUND\_COLOR\_G = $AF:000E  BACKGROUND\_COLOR\_R = $AF:000F |

### Screen Page 0 – Location $AF:A000

Screen Page 0 memory is used to store text characters for display.

One page of text is 128 columns by 64 rows. This adds up to 8 KB of memory of text. C256 does not display the entire buffer on the screen. Typically, we render 72 characters per row, with 56 rows. This uses 576 x 448 of the available 640 x 480 resolution. The border size can be modified or turned off completely. The display process reads Screen Page 0 and for each character, displays it’s character set bitmap.

### Screen Page 1 – Location $AF:C000

An additional page of 128 x 64 is used to store the colors. Each byte is split into foreground (4bits) and background (4 bits). The high nibble (bits 7..4) are the foreground and the low nibble (bits 3..0) are the background.

The colors used (the 4 bits) are used to lookup RBG values in two lookup tables (LUT).

The foreground (FG) LUT is located at $AF:1F40 for 64 bytes – only 16 x 3 = 48 bytes are used. The extra byte may be used for alpha (transparency) later on.

The background (BG) LUT is located at $AF:1F80 for 64 bytes – only 16 x 3 = 48 bytes are used. The extra byte may be used for alpha (transparency) later on.

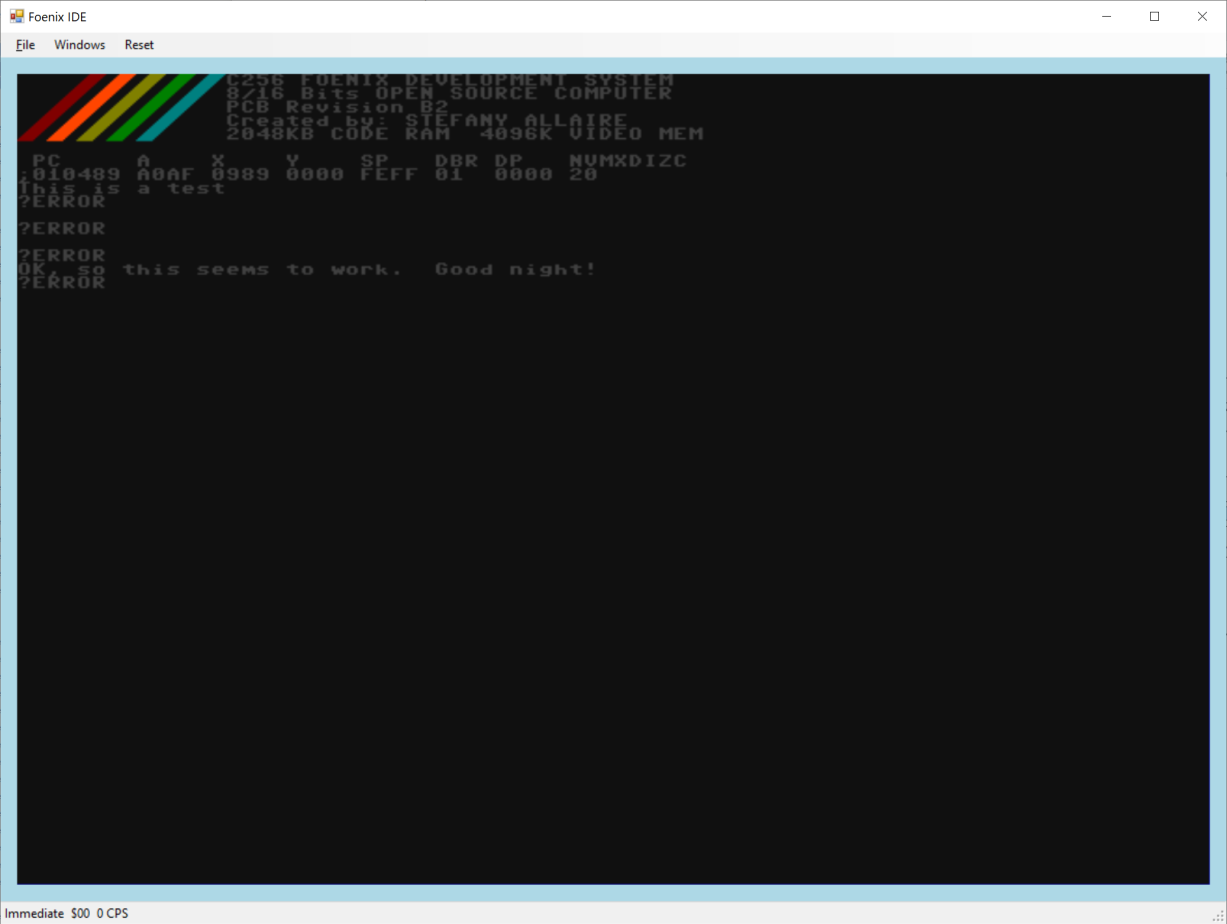
The colors are assigned 8-bit blue, 8-bit green, 8-bit red, 8-bit alpha (not used) for each of those colors in Text Mode.

#### Example – Color Lookup

Consider the following Foreground and Background Lookup Tables

|  |  |
| --- | --- |
| Foreground Color Lookup Table, starting at address $AF:1F40 | Background Color Lookup Table, starting at address $AF:1F80 |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Index | Blue | Green | Red | Alpha | Color | | 0 | $00 | $00 | $00 | $FF | Black | | 1 | $00 | $00 | $80 | $FF | Maroon | | 2 | $00 | $80 | $00 | $FF | Green | | 3 | $80 | $00 | $00 | $FF | Navy | | 4 | $00 | $80 | $80 | $FF | Olive | | 5 | $80 | $80 | $00 | $FF | Teal | | 6 | $80 | $00 | $80 | $FF | Purple | | 7 | $80 | $80 | $80 | $FF | Gray | | 8 | $00 | $45 | $FF | $FF | Orange | | 9 | $13 | $45 | $8B | $FF | Brown | | A | $00 | $00 | $20 | $FF | Dark Red | | B | $00 | $20 | $00 | $FF | Dark Green | | C | $20 | $00 | $00 | $FF | Indigo | | D | $20 | $20 | $20 | $FF | Dark Gray | | E | $40 | $40 | $40 | $FF | Slate Gray | | F | $FF | $FF | $FF | $FF | White | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Index | Blue | Green | Red | Alpha | Color | | 0 | $00 | $00 | $00 | $FF | Black | | 1 | $00 | $00 | $80 | $FF | Maroon | | 2 | $00 | $80 | $00 | $FF | Green | | 3 | $80 | $00 | $00 | $FF | Navy | | 4 | $00 | $20 | $20 | $FF | ?? | | 5 | $20 | $20 | $00 | $FF | ?? | | 6 | $20 | $00 | $20 | $FF | ?? | | 7 | $20 | $20 | $20 | $FF | ?? | | 8 | $1E | $69 | $D2 | $FF |  | | 9 | $13 | $45 | $8B | $FF | Brown | | A | $00 | $00 | $20 | $FF | Dark Red | | B | $00 | $20 | $00 | $FF | Dark Green | | C | $40 | $00 | $00 | $FF | Blue | | D | $10 | $10 | $10 | $FF | Midnight Gray | | E | $40 | $40 | $40 | $FF | Slate Gray | | F | $FF | $FF | $FF | $FF | White | |

If a character in Screen Page 1 is $ED (the default text color combination), then the foreground color index is E and the background color index is D. Looking up the index for E will make the foreground “Slate Gray” and the background “Midnight Gray”. The image below shows this color combination in text.



### Text Gamma Lookup Table

The Gamma lookup table is used to adjust the color between different display devices (such as DVI versus VGA). Each of the red, green and blue can be corrected. Each table consists of 256 values.

GAMMA\_B\_LUT\_PTR = $AF:4000

GAMMA\_G\_LUT\_PTR = $AF:4100

GAMMA\_R\_LUT\_PTR = $AF:4200

Gamma can be enabled or disabled.

### Master Control Register

The Master Control Register (MCR) is used to enable/disable various video mode. The MCR is located at address $AF:0000.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Master Control Register ($AF:0000)** | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Disable Vid | Gamma | Sprite | Tilemap | Bitmap | Graph Mode | Text Overlay | Text Mode |

|  |  |  |
| --- | --- | --- |
| **MCR Bit** | **MCR Name** | **Description** |
| 0 | Mstr\_Ctrl\_Text\_Mode\_En | Enable the Text Mode |
| 1 | Mstr\_Ctrl\_Text\_Overlay | Enable the Overlay of the text mode on top of Graphic Mode (the Background Color is ignored) |
| 2 | Mstr\_Ctrl\_Graph\_Mode\_En | Enable the Graphic Mode |
| 3 | Mstr\_Ctrl\_Bitmap\_En | Enable the Bitmap Module in Vicky |
| 4 | Mstr\_Ctrl\_TileMap\_En | Enable the Tile Module in Vicky |
| 5 | Mstr\_Ctrl\_Sprite\_En | Enable the Sprite Module in Vicky |
| 6 | Mstr\_Ctrl\_GAMMA\_En | Enable the GAMMA correction - The Analog and DVI have different color value, the GAMMA is great to correct the difference.  NOTE: This could also be used for fade-in and out. |
| 7 | Mstr\_Ctrl\_Disable\_Vid | This bit disables the Scanning of the Video Memory, hence giving 100% bandwidth to the CPU to access graphic data.  NOTE: In this case the Border color or the background is displayed on the screen (I can't remember) to be advised |

### Displaying Graphics

C256 has 4 MB of Video RAM available, starting at $B0:0000 and ending at $EF:FFFF.

The order in which images are drawn are:

* L0 - Sprites (Closest to the Viewer)
* L1 - Tile Layer 0
* L2 - Sprite Layers
* L3 - Tile Layer 1
* L4 - Sprite Layers
* L5 - Tile Layer 2
* L6 - Sprite Layers
* L7 - Tile Layer 3
* L8 - Sprite Layers
* L9 - Background (bitmap)

#### Bitmaps

Once the Bitmap bit is set in the MCR, Vicky will retrieve the Bitmap Control Register at address $AF:0140.

The Bitmap Control Register is shown below.

|  |  |  |
| --- | --- | --- |
| **Bitmap Control Register ($AF:0140)** | | |
| 7 .. 4 | 3 .. 1 | 0 |
| Reserved | LUT | Enable |

|  |  |  |
| --- | --- | --- |
| **BCR Bit** | **BCR Name** | **Description** |
| 0 | Enabled | Enable the bitmap |
| 1 .. 3 | LUT | Lookup Table Index 0 to 7 |
| 4 .. 7 | Reserved | Reserved |

The address pointer of the bitmap in the Video RAM is located at addresses $AF:0141 to $AF:0143. The address stored must be offset by $B0:0000. As an example, if the bitmap data is stored in at address $B1:4000 in memory, the address pointer must be $01:4000. The bitmap width is saved in the word $AF:0144 to $AF:0145. The bitmap height is saved in the word $AF:0146 to $AF:0147.

#### Tiles

Once the Tile bit is set in the MCR, Vicky will retrieve the Tile Control Register at addresses $AF:0100, $AF:0108, $AF:0110 and $AF:0118 to determine if a tileset should be displayed. There can be four tilesets at any given time in the display.

The Tile Control Registers are shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tile Control Register ($AF:0100, $AF:0108, $AF:0110, $AF:0118)** | | | |
| 7 | 6 .. 4 | 3 .. 1 | 0 |
| Tile Striding | Reserved | LUT | Enable |

|  |  |  |
| --- | --- | --- |
| **TCR Bit** | **TCR Name** | **Description** |
| 0 | Enabled | Enable the bitmap |
| 1 .. 3 | LUT | Lookup Table Index 0 to 7 |
| 4 .. 6 | Reserved | Reserved |
| 7 | Tile Striding | 0: sequential, 1: 256 x 256 Tile sheet striding |

Each tile has its own control register, video address pointer, X and Y stride.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tile 0 | Tile 1 | Tile 2 | Tile 3 |
| Tile Control Register | $AF:0100 | $AF:0108 | $AF:0110 | $AF:0118 |
| Tile Start Address | $AF:0101 .. $AF:0103 | $AF:0109 .. $AF:010B | $AF:0111 .. $AF:0113 | $AF:0119 .. $AF:011B |
| Tile Map X Stride | $AF:0104 .. $AF:0105 | $AF:010C .. $AF:010E | $AF:0114 .. $AF:0115 | $AF:011C .. $AF:011E |
| Tile Map Y Stride | $AF:0106 .. $AF:0107 | $AF:010E .. $AF:010F | $AF:0116 .. $AF:0117 | $AF:011E .. $AF:011F |

Tile maps are stored at addresses $AF:5000, $AF:5800, $AF:6000 and $AF:6800.

#### Sprites

Once the Sprite bit is set in the MCR, Vicky will display sprites in the appropriate layer. There can be 32 sprites displayed for each screen refresh.

The Sprite Control Registers are shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprite Control Register ($AF:0200 to $AF:02F8, offset by 8 bytes)** | | | |
| 7 | 6 .. 4 | 3 .. 1 | 0 |
| Tile Striding | Layer | LUT | Enable |

Each sprite has a Control Register, a video memory address, and X and Y locations.

#### SuperIO Registers Location $AF:1000 to $AF:13FF

VICKY has a LPC47M10x ‘SuperIO’ chip mapped into its address space. This chip provides a number of peripherals including PS2 keyboard/mouse, game port, serial ports, parallel port, floppy controller and MIDI input/output. Even though the SuperIO is now interfaced through GAVIN the registers remain mapped in VICKY’s address space as this is where it resided in V1. This may change in V3. Please see the section on GAVIN for more information.

assign Background\_Blue\_o = VICKY\_MASTER\_REG[13];  
assign Background\_Green\_o = VICKY\_MASTER\_REG[14];  
assign Background\_Red\_o = VICKY\_MASTER\_REG[15];  
assign Interrupt\_Enable\_o = VICKY\_MASTER\_REG[27][3:0];  
assign Vicky\_Interrupt\_LineCompare0\_o = {VICKY\_MASTER\_REG[29][3:0], VICKY\_MASTER\_REG[28]};  
assign Vicky\_Interrupt\_LineCompare1\_o = {VICKY\_MASTER\_REG[31][3:0], VICKY\_MASTER\_REG[30]};

First part is about you talking about $AF000D, that would the background color when in graphic mode

The second part is about the line Interrupt 27/28/29/30

If color index read from bitmap mem is 0, background bgr gets drawn.

lut[0] never used then? For all Sprtie/Tiles/Bitmap pixels = 0 = transparent, then background color prevail

The 2k buffer are for the tile maps 40 x 30. They work the same way as the text memory, but instead of displaying an 8x8 character, they display 16 x 16 tile. 64 x 32 = 2048. Forget about the stride for now.

The 16x16 tile is stored in vram, at the start address specified in TLx\_START\_ADDY\_L|M|H?

The TL... Start is the pointer to the tile set from Vicky's perspective

each tile set being a 256x256 bitmap? 8bpp, so 64k

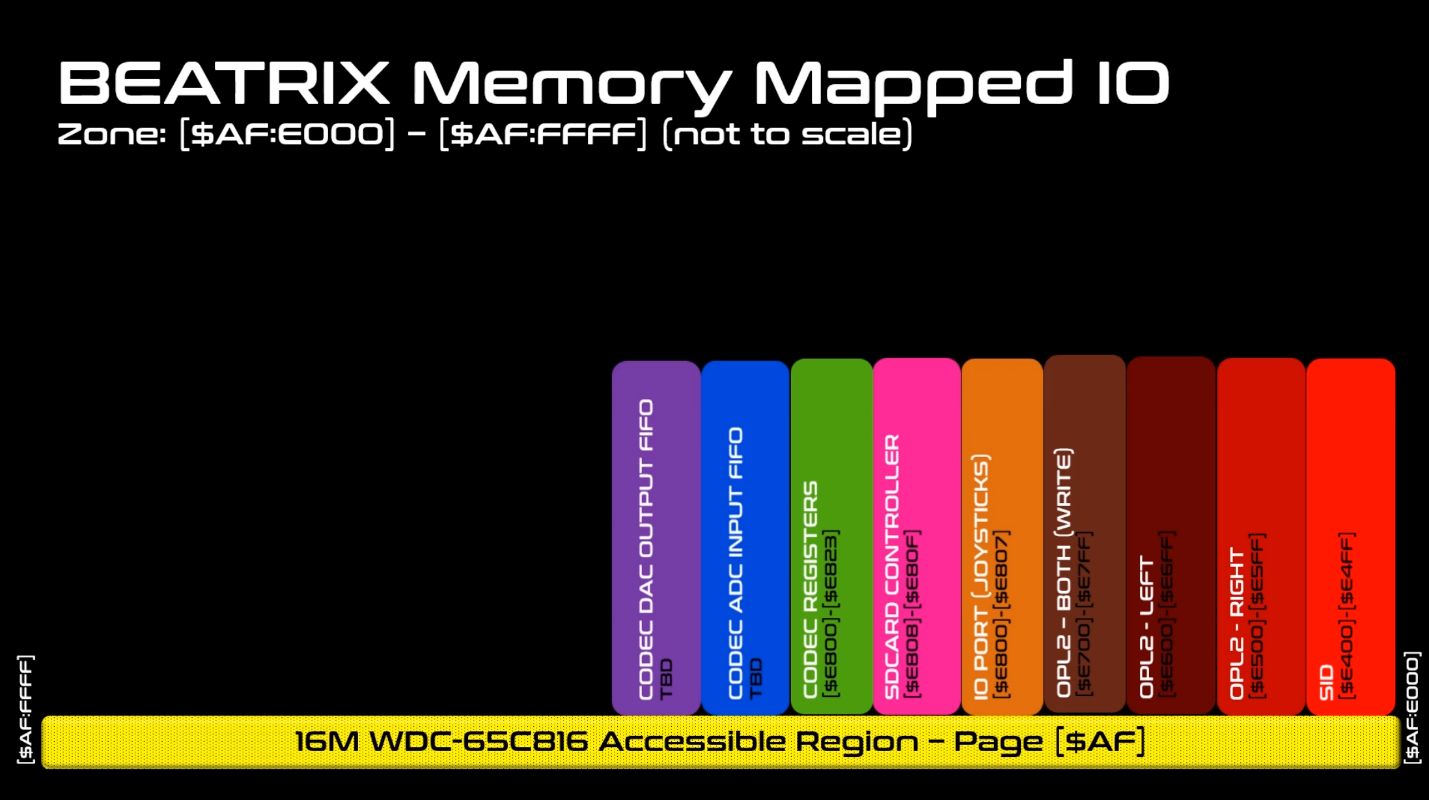
So, the address won't be $bx:xxxx but $0x:0000

And the Tileset can be either linear or 256 x 256..

what is the meaning of linear in this context?

The difference is that the tile in the file would be arrange like 16x16x256 that is linear as opposed to 256 x 256. The pointer won't move the same way when the line changes.

## 13 - Beatrix– Location $AF:E000 to $AF:FFFF



// $AF:E400..$AF:E4FF // SID (Still to be defined) Not Implemented yet  
// $AF:E500..$AF:E5FF // OPL2 - Left Side  
// $AF:E600..$AF:E6FF // OPL2 - Right Side  
// $AF:E700..$AF:E7FF // OPL2 - Both Side (Write Sequence Only)  
// $AF:E800..$AF:E807 // Joystick + AD Channel + SD Controller  
// $AF:E820..$AF:E823 // CODEC Register  
// $AF:E810..$AF:E81F //SD Card Stat

; Joystick Ports  
JOYSTICK0 = $AF:E800 ;(R) Joystick 0 - J7 (Next to Buzzer)  
JOYSTICK1 = $AF:E801 ;(R) Joystick 1 - J8  
JOYSTICK2 = $AF:E802 ;(R) Joystick 2 - J9  
JOYSTICK3 = $AF:E803 ;(R) Joystick 3 - J10 (next to SD Card)

; Dip switch Ports  
DIPSWITCH = $AF:E804 ;(R) $AFE804...$AFE807

; SD Card CH376S Port  
SDCARD\_DATA = $AF:E808 ;(R/W) SDCARD (CH376S) Data PORT\_A (A0 = 0)  
SDCARD\_CMD = $AF:E809 ;(R/W) SDCARD (CH376S) CMD/STATUS Port (A0 = 1)

; SD Card Card Presence / Write Protect Status Reg  
SDCARD\_STAT = $AF:E810 ;(R) SDCARD (Bit[0] = CD, Bit[1] = WP)

; Audio WM8776 CODEC Control Interface (Write Only)  
CODEC\_DATA\_LO = $AF:E820 ;(W) LSB of Add/Data Reg to Control CODEC See WM8776 Spec  
CODEC\_DATA\_HI = $AF:E821 ;(W) MSB od Add/Data Reg to Control CODEC See WM8776 Spec  
CODEC\_WR\_CTRL = $AF:E822 ;(W) Bit[0] = 1 -> Start Writing the CODEC Control Register

# 14 - Flash Boot Process

The access time of the FLASH RAM is very much slower than the effective 35ns access time of the CPU. To prevent having to use a lot of wait states to ‘pause’ the CPU and wait for the FLASH the contents of FLASH are copied into RAM which is much faster and will let the CPU run at full speed.

On boot, GAVIN copies the entire 512KB of system flash are copied from address range $F0:0000-$F7:FFFF to $18:0000 to $1F:FFFF.

Then GAVIN, copies the first 64KB of the flash RAM for the reset vectors from address range $F0:0000-$F0:FFFF to $00:0000 to $00:FFFF

# 15 - Software Tools and GIT Repository Information

Foenix IDE: <https://github.com/Trinity-11/FoenixIDE.git>   
C256 Kernel: <https://github.com/Trinity-11/Kernel>

# Tools

To modify the Foenix IDE, you will need Visual Studio 2017 Community edition for C#.

Link?