User Manual

Rev 0.0.2– August 26th, 2022

FOENIX RETRO SYSTEMS

DOCUMENT HISTORY

By	Date	Modification	New Revision
Stefany Allaire	June 2nd 2022	Document Creation	0.0.0
Stefany Allaire	August 25th	Document Edition	0.0.1
Stefany Allaire	August 26th	First Release	0.0.2

TABLE OF CONTENTS

1 Contents

2 C256 FOENIX JR's CONNECTORS DESCRIPTION 6

3 C256 FOENIX JR's MAIN COMPONENTS & FEATURES 7

4 C256 FOENIX JR's BLOCK DIAGRAM 8

5 C256 FOENIX JR's MEMORY ORGANISATION 9

6 C256 FOENIX JR's IO MEMORY MAP 10

7 C256 FOENIX JR's MEMORY MAP IN MORE DETAILS 12

8 C256 FOENIX JR's MEMORY MANAGEMENT UNIT 13

9 C256 FOENIX JR's MMU CONTROL - $0000 14

10 C256 FOENIX JR's IO PAGE CTRL - $0001 15

11 C256 FOENIX JR's – IO PAGE 0 – I/O REGISTERS DEFINITION 16

11.1 Tiny VICKY - Master Control Registers - $D000 - $D0FF 16

11.2 Tiny VICKY – Bitmap Control Registers - $D100 - $D1FF 18

11.3 Tiny VICKY – Tile Map Control Registers - $D200 - $D2FF 18

11.4 SID Left Channel - $D400 - $D4FF 22

11.5 SID Right Channel - $D500 - $D5FF 23

11.6 PSG Left Channel - $D600 - $D60F 23

11.7 PSG Right Channel - $D610 - $D61F 23

11.8 CODEC - $D620 - $D62F 24

11.9 UART - $D630 - $D63F 24

11.10 PS2 - $D640 - $D64F 24

11.11 TIMER CONTROLLER - $D650 - $D65F 25

11.12 INTERRUPT CONTROLLER - $D660 - $D66F 26

11.13 DIP SWITCH - $D670 - $D67F 26

11.14 IEC CONTROLLER - $D680 - $D68F 27

11.15 RTC - $D690 - $D69F 28

11.16 System Control Registers (Leds, Buzzer, LFSR, Etc…) - $D6A0 - $D6AF 30

11.17 Tiny VICKY – SPRITES REGISTERS - $D900 - $DAFF 32

11.18 VIA - $DC00 - $DCFF 33

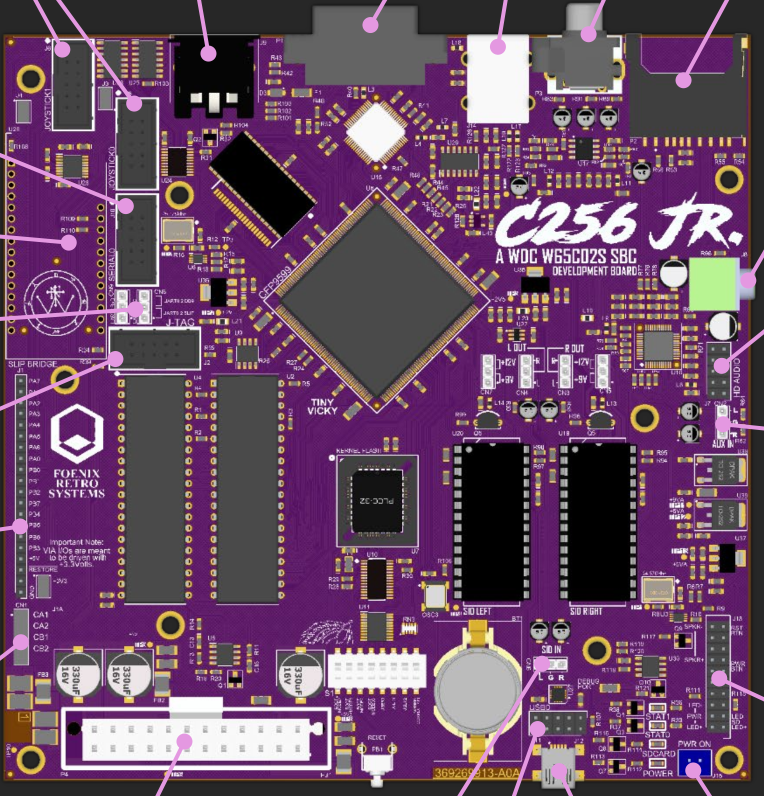
11.19 SD CARD - $DD00 - $DDFF 34

11.20 MATH - $DE00 - $DEFF 36

11.21 DMA CONTROLLER - $DF00 - $DFFF 37

2 C256 FOENIX JR's CONNECTORS DESCRIPTION

Details of all the connectors on the board and certain selectors.



Joysticks Port
9 Pins – 3 Buttons
Atari Style
Controlled by the VIA Chip

IEC Port
Slow/Fast Mode
Compatible
1541/1571/1581/Etc…

DVI D/A
DVI Digital
(HDMI Compatible)
VGA Output
640x480@60Hz
640x400@70Hz

PS2 Ports
PS2 Keyboard
PS2 Mouse

RCA Line-Out
Stereo
Line-Out
Level

SDCard
Full Size
SDCard
Connector

Simple UART
Serial Port

ESP32 Feather
Board
WIFI SLIP Bridge

Serial Port Output
Selector

FPGA JTAG PORT

VIA
I/O
(shared with Joysticks)
C64 Compatible
Keyboard Connector

VIA
CA1/CA2/CB1/CB2
VIA Control Pins
Fanned Out

Headphone
Output

HD Audio Header
Bring headphone
output to front
bezel.

RCA Line-In
Accepts Stereo
Line-In Level

Case Front Bezel
Connector Block
Includes:
Reset Button
PW On Button
Speaker
PW LED
HDD LED (SDCard)

24Pins ATX Power Supply
Standard Power Supply Connector
Supports
20/24Pins

SID Analog In
Direct
Connection
To SID Analog
Input

Background
Debug
Port
Standard USB2.0
Header

Background
Debug
Port
Mini-USB
(Requires drivers to run @ 6Mbits)

Power On
Short to Power
On the board

3 C256 FOENIX JR's MAIN COMPONENTS & FEATURES

Details of each important components in the system.

TINYVICKY CFP9599

Text Mode:
80x60/40x30/80x50/40x25

Graphics Mode:
320x240@60Hz/320x200@70Hz

Text Mode Colors:
16 Colors Background/Foreground
(from 24bits Palette)

Graphics Mode Colors:
1 Byte Per Pixel
4x LUT of ARGB (24bits Palette)

Graphics Engine Features:
2x Layer Bitmap*
3x Layer Tilemap(**)/(*) (8x8, 16x16)
64 Sprites (32x32, 24x24, 16x16, 8x8)

Sound:
2x PSG (SN76489)
Configurable in Mono or Stereo
1x Channel 24Kz, 8Bits (Mono/Stereo)

(**) DMA Controller:
Certain Blitter Features**

MMU:
LUT Based Memory Management Unit
Can Switch between 4 Different Memory arrangement (8K Page)

Notes:

*: Bitmap and 1 Layer Tile will be mutually exclusive. 4 Tile Layers or 3 Tile Layer + Bitmap.

**: These are tentative Specs – To be confirmed after implementation – Could be limited by Frame Rate @60/@70Hz

System RAM

256K SRAM

Audio CODEC

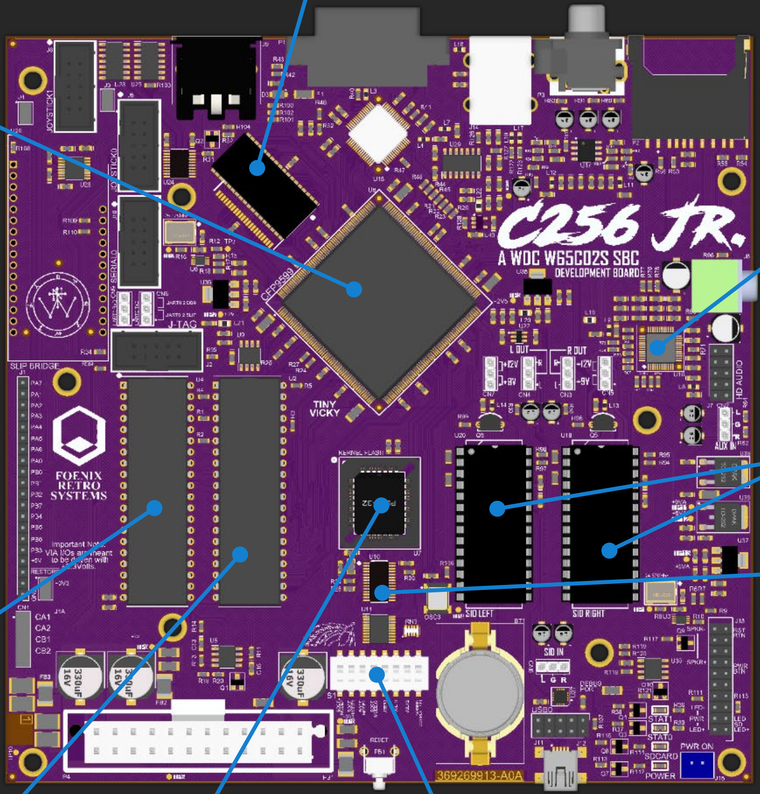
24Khz/8Bits (M/S) Output
Line-In
Line-Out
Headphone Output

DUAL SID

Voltage Jumper Selector for 6581/8581
Audio Output Selector (M/S)

RTC

Full RTC with Battery Backup for time keeping and Alarm.
(No Storage EEPROM)



W65C22S

Works @ 3.3V

W65C02S @ 6,293Mhz

((Dot Clock)25.175Mhz/4)
Works @ 3.3V

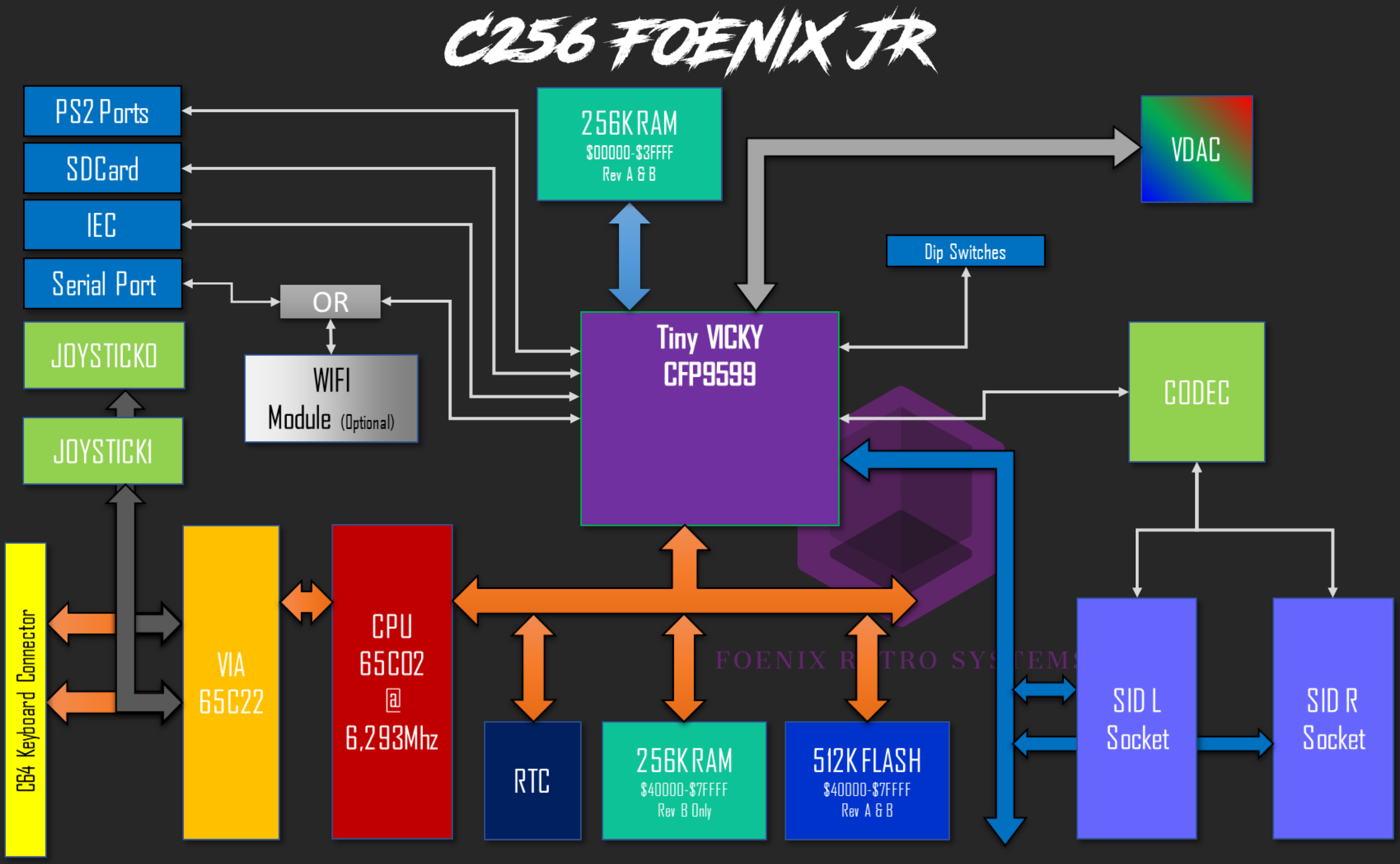
NOR FLASH

512K FLASH
For KERNEL, USER APPS, Etc…

DIP SWITCH

GAMMA CORRECTION
LOW/HI RES BOOT
BOOT SECTOR SELECTOR
USER DEFINABLE

4 C256 FOENIX JR's BLOCK DIAGRAM

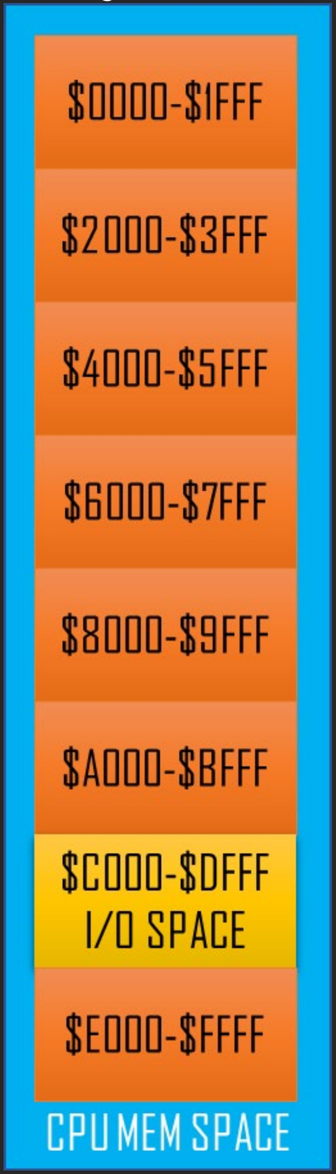


5 C256 FOENIX JR's MEMORY ORGANISATION

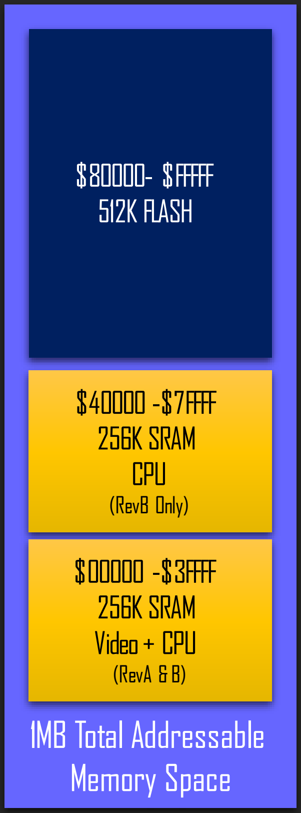
The C256 Foenix Jr. is designed to access a total of 1 MB of memory. In the original Version A of the board, 512K of flash are present and 256K of SRAM is present and shared between the CPU and the Video System. In the upcoming Revision B, an addition 256K of RAM will be available for the CPU to access.

Obviously, the CPU, the 65C02 can only address 64K at the time and thus this why the overall 1Mbyte of memory is divided in 8Kbytes chunk that could be mapped in the 65C02 memory space with the help of the MMU.

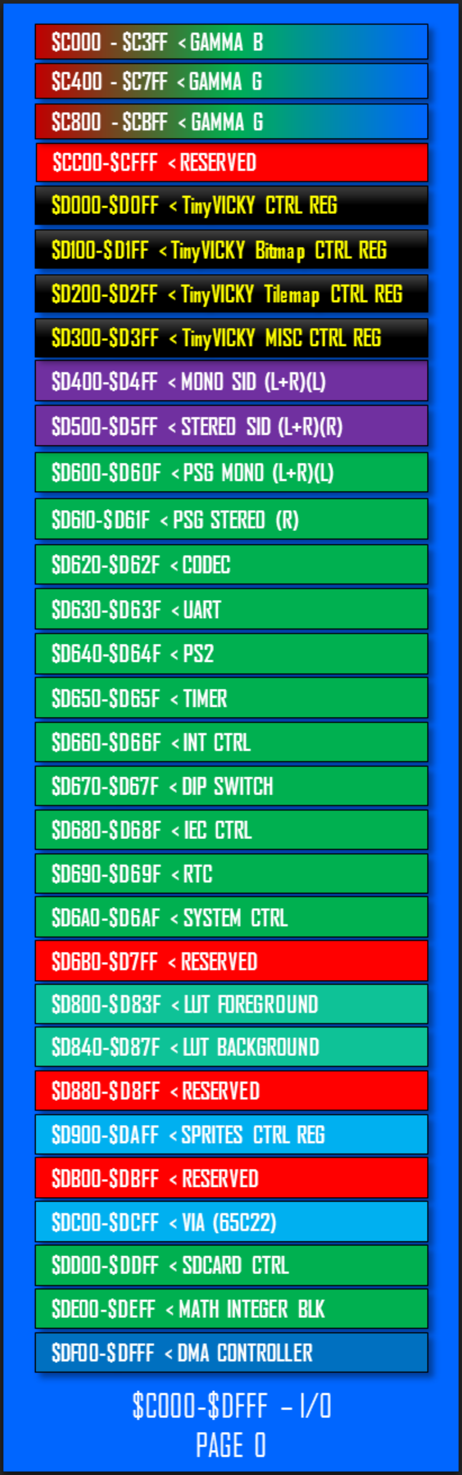
In the Next Chapter the MMU Functions will be explained, but simply put you can apply for every 8Kbyte chunk in the CPU addressable space, any 8K from the overall 1Mbyte space either being a block of RAM or a block of FLASH. The MMU includes many pages so you can switch from a MMU configuration to another.



By default, the CPU will boot with the memory from $0000 to $BFFF mapped in the first blocks of SRAM, the shared memory from CPU Video since it is common to both board revisions. Then there will be the IO page located @ $C000 to $DFFF so all the system registers and screen memory can be accessed, finally, the last block will be assigned to Flash and that will be assigned to the last 8K of the 512K of Flash itself. It should boot in "Open Kernel" and if there is no other BASIC block supplied it should give access to a simple CLI that will allow to load and save programs.



6 C256 FOENIX JR's IO MEMORY MAP



$C000 - $C3FF < GAMMA B
$C400 - $C7FF < GAMMA G
$C800 - $CBFF < GAMMA G
$CC00-$CFFF < RESERVED
$D000-$D0FF < TinyVICKY CTRL REG
$D100-$D1FF < TinyVICKY Bitmap CTRL REG
$D200-$D2FF < TinyVICKY Tilemap CTRL REG
$D300-$D3FF < TinyVICKY MISC CTRL REG
$D400-$D4FF < MONO SID (L+R)(L)
$D500-$D5FF < STEREO SID (L+R)(R)
$D600-$D60F < PSG MONO (L+R)(L)
$D610-$D61F < PSG STEREO (R)
$D620-$D62F < CODEC
$D630-$D63F < UART
$D640-$D64F < PS2
$D650-$D65F < TIMER
$D660-$D66F < INT CTRL
$D670-$D67F < DIP SWITCH
$D680-$D68F < IEC CTRL
$D690-$D69F < RTC
$D6A0-$D6AF < SYSTEM CTRL
$D6B0-$D7FF < RESERVED
$D800-$D83F < LUT FOREGROUND
$D840-$D87F < LUT BACKGROUND
$D880-$D8FF < RESERVED
$D900-$DAFF < SPRITES CTRL REG
$DB00-$DBFF < RESERVED
$DC00-$DCFF < VIA (65C22)
$DD00-$DDFF < SDCARD CTRL
$DE00-$DEFF < MATH INTEGER BLK
$DF00-$DFFF < DMA CONTROLLER

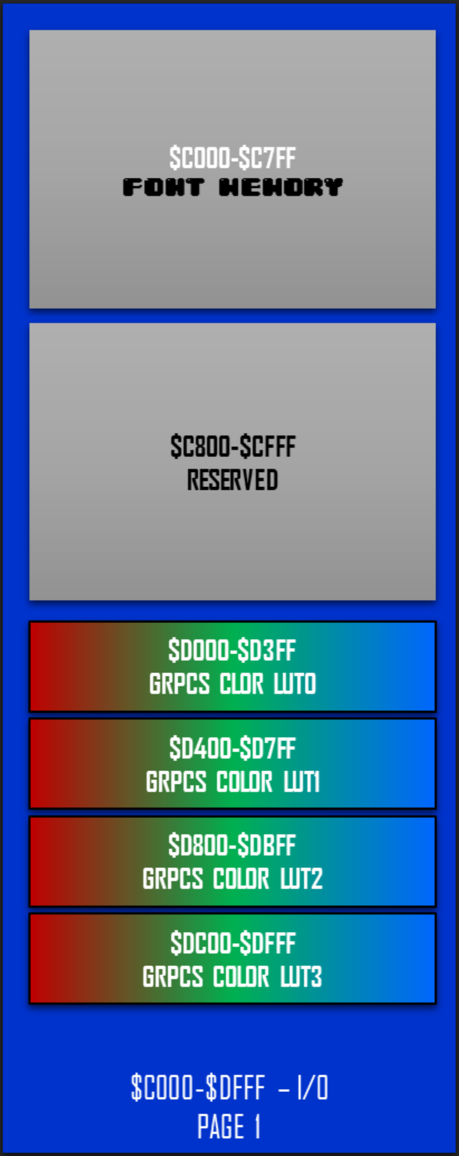
$C000-$DFFF – I/O
PAGE 0

There are 4 pages of Input/Output in the C256 Foenix Junior, but for the most part, all the necessary registers to control the system, the music and/or the different graphical aspects resides in PAGE 0.

We are talking about all the control registers for Tiny Vicky, the bitmaps, the Tile maps and all the sprites. There are the PSG and SID controls and finally all the other system related control registers like the SDCard, the timers, the interrupt controllers, etc…

Later in the manual, each section will be detailed with all the registers and their different bits meanings.

In PAGE 1, it mostly contains the memory space for the FONT set. Then in the second portion of the 8K byte block you will find the LUT Tables for the Graphics Engine. Those are used for the bitmap, Tiles, and Sprites. They each contains 256 Entry of 4 Bytes, (LSB) Blue, Green, Red, Alpha (not used for now).



$C000-$C7FF
FONT MEMORY

$C800-$CFFF
RESERVED

$D000-$D3FF
GRPCS CLOR LUT0

$D400-$D7FF
GRPCS COLOR LUT1

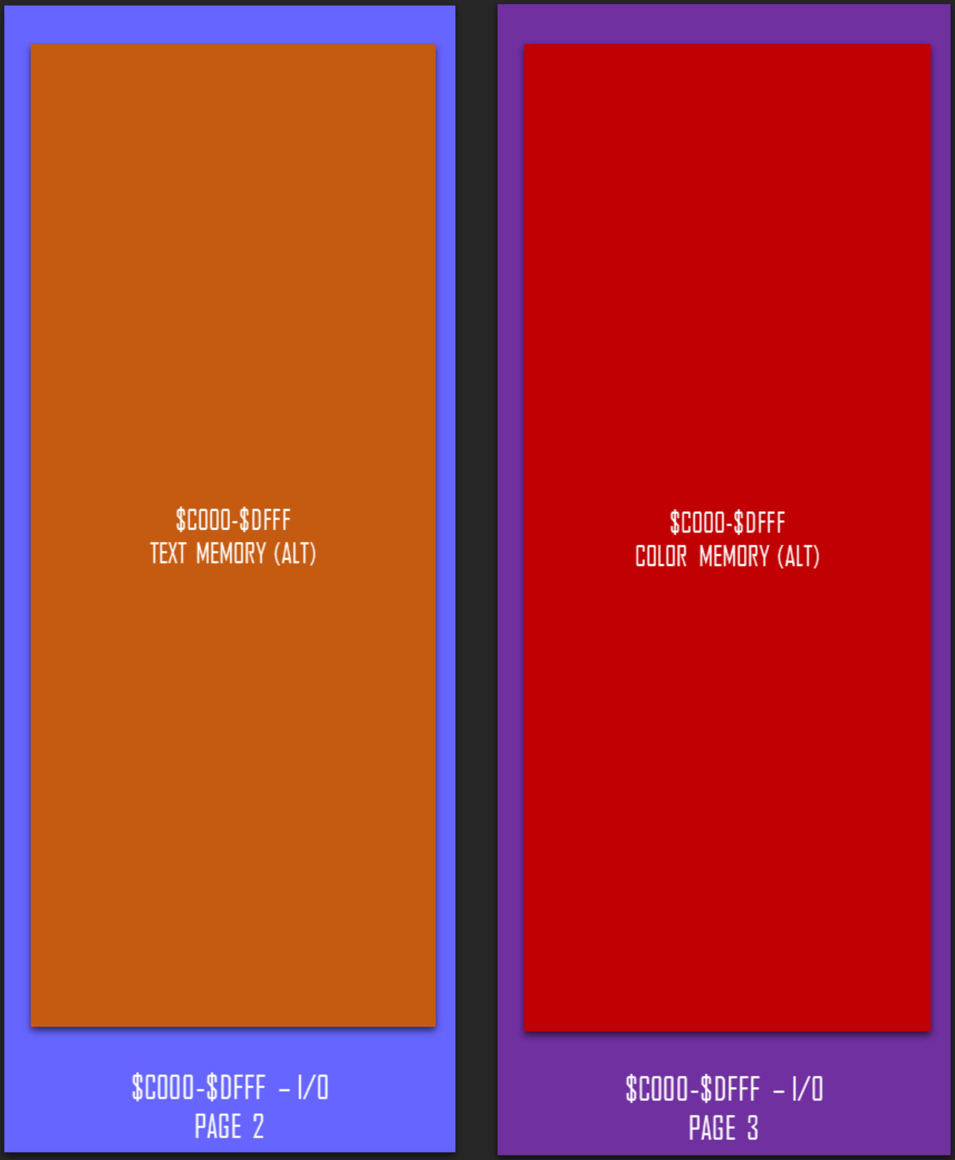
$D800-$DBFF
GRPCS COLOR LUT2

$DC00-$DFFF
GRPCS COLOR LUT3

$C000-$DFFF – I/O
PAGE 1

In PAGE 2 & 3, it is about the text screen memory. Essentially, when in Text or Overlay mode, the text that is displayed on screen will be residing in the 8K Allocated in Page 2 for the characters themselves and the Color byte that comes along in Page 3.

It is important to note that is also possible to disable the IO page for that specific page in the CPU address space to be RAM or FLASH. With the powerful MMU in place, pretty much anything is possible.



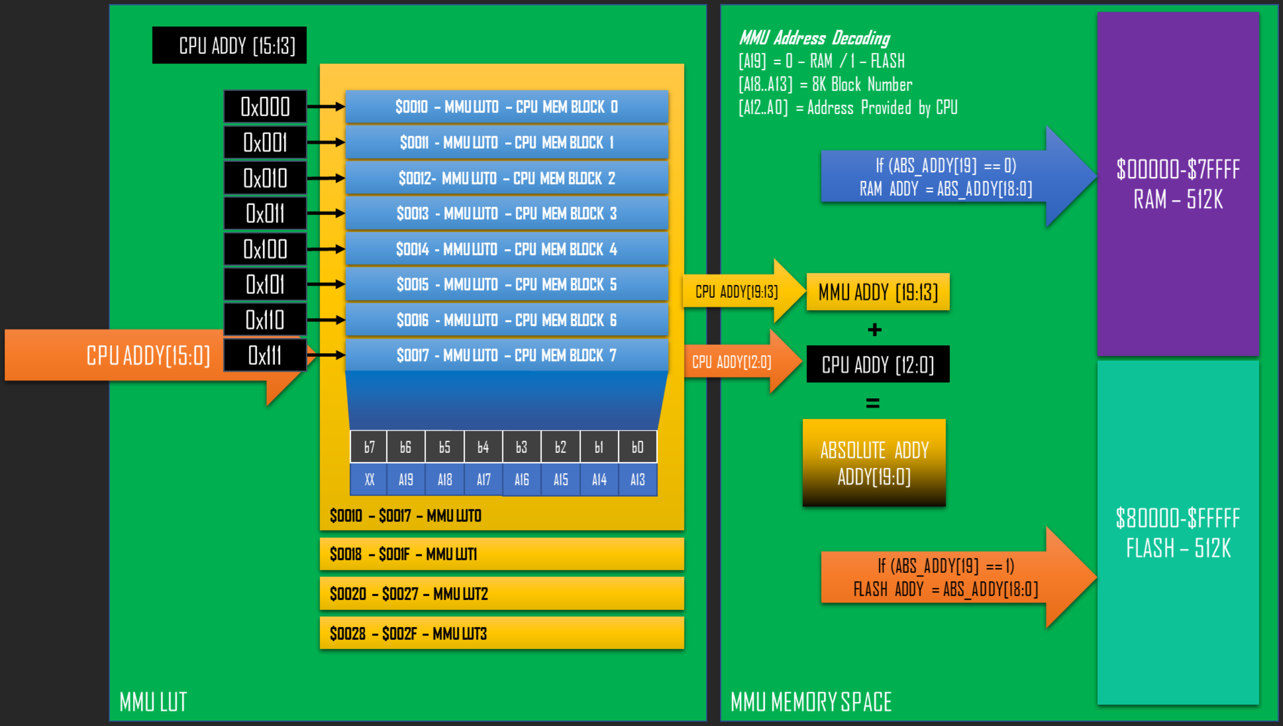
7 C256 FOENIX JR's MEMORY MAP IN MORE DETAILS

Addy Start	Addy End	Type	R/W	Description
0x0_0000	0x0_0000	I/O	R/W	MMU Control Register
0x0_0001	0x0_0001	I/O	R/W	IO Page Control Register
0x0_0002	0x3_FFFF	MEM	R/W	256K SRAM – Shared in between Tiny VICKY & CPU
0x4_0000	0x7_FFFF	MEM	R/W	256K SRAM (Rev B Only)
0x8_0000	0xF_FFFF	MEM	R/W	512K FLASH (Run in Place)
				IO PAGE 0
0x0_C000	0x0_C3FF	MEM	R/W	GAMMA Table – Blue Color
0x0_C400	0x0_C7FF	MEM	R/W	GAMMA Table – Green Color
0x0_C800	0x0_CBFF	MEM	R/W	GAMMA Table – Red Color
0x0_CC00	0x0_CFFF			RESERVED
0x0_D000	0x0_D0FF	I/O	R/W	Tiny VICKY - Master Control Registers
0x0_D100	0x0_D1FF	I/O	R/W	Tiny VICKY - Bitmap – Control Registers
0x0_D200	0x0_D2FF	I/O	R/W	Tiny VICKY – Tile map – Control Registers
0x0_D300	0x0_D3FF			RESERVED
0x0_D400	0x0_D4FF	I/O	W	External SID Left
0x0_D500	0x0_D5FF	I/O	W	External SID Right
0x0_D600	0x0_D60F	I/O	W	PSG Left
0x0_D610	0x0_D61F	I/O	W	PSG Right
0x0_D620	0x0_D62F	I/O	W	CODEC
0x0_D630	0x0_D63F	I/O	R/W	UART (TL16C750 Compatible)
0x0_D640	0x0_D64F	I/O	R/W	PS2 Interface
0x0_D650	0x0_D65F	I/O	R/W	Timers
0x0_D660	0x0_D66F	I/O	R/W	Interrupt Controller
0x0_D670	0x0_D67F	I/O	R	DIP Switch
0x0_D680	0x0_D68F	I/O	R/W	IEC Controller
0x0_D690	0x0_D69F	I/O	R/W	RTC
0x0_D6A0	0x0_D6AF	I/O	R/W	System Control Registers (LEDs, LFSR, Etc..)
0x0_D6B0	0x0_D7FF			RESERVED
0x0_D800	0x0_D83F	MEM	W	Text Foreground LUT
0x0_D840	0x0_D87F	MEM	W	Text Background LUT
0x0_D6B0	0x0_D7FF			RESERVED
0x0_D900	0xDAFF	MEM	W	Sprites Control Registers
0x0_DB00	0x0_DBFF			RESERVED
0x0_DC00	0x0_DCFF	I/O	R/W	65C22 – VIA (C64 Keyboard + Joysticks)
0x0_DD00	0x0_DDFF	I/O	R/W	SD Card Controller
0x0_DE00	0x0DEFF	I/O	R/W	Integer Math Block (16bits Multiply & 16bits Divide)
0x0_DF00	0x0DFFF	I/O	R/W	DMA Controller
				IO PAGE 1
0x0_C000	0x0_CFFF	MEM	R/W	FONT Memory
0x0_D000	0x0_D3FF	MEM	R/W	GRAPHICS LUT0
0x0_D400	0x0_D7FF	MEM	R/W	GRAPHICS LUT1
0x0_D800	0x0_DBFF	MEM	R/W	GRAPHICS LUT2
0x0_DC00	0x0_DFFF	MEM	R/W	GRAPHICS LUT3
				IO PAGE 2
0x0_C000	0x0_DFFF	MEM	R/W	Text Memory - Max Usage (80x60 = 4.8K) out of 8K (3.2K Free)
				IO PAGE 3
0x0_C000	0x0_DFFF	MEM	R/W	Color Memory - Max Usage (80x60 = 4.8K) out of 8K (3.2K Free)

8 C256 FOENIX JR’s MEMORY MANAGEMENT UNIT

In this section, I will explain how the memory management unit works. Fundamentally, the idea is to segment the CPU accessible memory in chunk of 8Kbytes and configure the MMU to assign to each of those chunks of 8Kbytes a portion of the available 256K of RAM or one of the 8K block of the available 512K of FLASH.

To do that, we are going to use LUT (look-up tables) which means that the circuitry will collect the active Address of the CPU and will isolate the bit A15, A14 and A13 to determine the 8K block the CPU wants to access. Those 3 bits will choose one of the 8 registers that are setup as a table and within that LUT entry being pointed by the CPU will contain the higher part of the complete address we will need to complete the transaction.


CPU ADDY [15:13]
0x000
0x001
0x010
0x011
0x100
0x101
0x110
0x111
CPU ADDY[15:0]
$0010 – MMU LUT0 – CPU MEM BLOCK 0
$0011 - MMU LUT0 – CPU MEM BLOCK 1
$0012- MMU LUT0 – CPU MEM BLOCK 2
$0013 - MMU LUT0 – CPU MEM BLOCK 3
$0014 - MMU LUT0 – CPU MEM BLOCK 4
$0015 - MMU LUT0 – CPU MEM BLOCK 5
$0016 - MMU LUT0 – CPU MEM BLOCK 6
$0017 - MMU LUT0 – CPU MEM BLOCK 7
b7 b6 b5 b4 b3 b2 b1 b0
XX A19 A18 A17 A16 A15 A14 A13
$0010 – $0017 – MMU LUT0
$0018 – $001F – MMU LUT1
$0020 – $0027 – MMU LUT2
$0028 – $002F – MMU LUT3
MMU LUT
MMU Address Decoding
[A19] = 0 – RAM / 1 – FLASH
[A18..A13] = 8K Block Number
[A12..A0] = Address Provided by CPU
If (ABS_ADDY[19] == 0)
RAM ADDY = ABS_ADDY[18:0]
$00000-$7FFFF
RAM – 512K
CPU ADDY[19:13]
MMU ADDY [19:13]
+
CPU ADDY[12:0]
CPU ADDY [12:0]
=
ABSOLUTE ADDY
ADDY[19:0]
If (ABS_ADDY[19] == 1)
FLASH ADDY = ABS_ADDY[18:0]
$80000-$FFFFF
FLASH – 512K
MMU MEMORY SPACE

We have overall, 1 Mbyte mapped in the system, 256K of RAM out of a window of 512K and a full 512K of FLASH that covers the second part of the overall 1Mbyte memory space.
So, as aforementioned, the value coming out of LUT will contain the high portion of the address so the system will know which bank of 8K the LUT entry will point to. So, we are going to use Bit5 to Bit0 from the LUT Entry to put in front of the already existing A[12:0] from the CPU, which will give us the full vector A[18:0] and finally the Bit6 in the LUT Entry will tell us if the entry is pointing towards FLASH memory or RAM memory. Then, the circuity will either insert an access in the RAM access block state-machine or let the CPU complete the transaction by simply read a value from the FLASH chips that sits on its bus.

There are 4 different LUT Tables which means that the configuration of the memory can be changed on the “FLY” by changing the system control register located @ $0001. With a simple bit switch you can map one or more table to have access to all the Video Memory if there is a need to transfer a bitmap image for example.

9 C256 FOENIX JR's MMU CONTROL - $0000

This section about the most important register in the system. The System Control register which has the function to control the different paging mechanisms in the system and the selection of the active MMU LUT out of the four that exists.

The reason this register is singled out is because it is a very important register that needs to be clearly understood.

@ Memory location $0000 resides a register that will used to control the MMU. The access and the control of the I/O Paging section is done @ $0001 with the IO Page Control Register.

$0000

b7	b6	b5	b4	b3	b2	b1	b0
Enable MMU LUT Access	RSVD	Edit MMU Page[1]	Edit MMU Page[0]	RSVD	RSVD	Active MMU LUT[1]	Active MMU LUT[0]

MMU CONTROL REGISTER

Bit[1:0]:	00b = Active MMU LUT Page0
	01b = Active MMU LUT Page1
	10b = Active MMU LUT Page2
	11b = Active MMU LUT Page3
Bit[2]:	RSVD
Bit[3]:	RSVD
Bit[5:4]:	00b = Edit MMU LUT0 – (Address $0008 – Address $000F)
	01b = Edit MMU LUT1 – (Address $0008 – Address $000F)
	10b = Edit MMU LUT2 – (Address $0008 – Address $000F)
	11b = Edit MMU LUT3 – (Address $0008 – Address $000F)
Bit[6]:	RSVD
Bit[7]:	0 = Address $0008 to $000F will read and write as RAM
	1 = Enable the Editing of the LUT Table Entries from $0008 to $000F

So, to summarize, bit0 and bit1 are the bits that will select the active MMU LUT. @ Reset, the value is set to 00b and each LUT Entry will match the 7 first blocks of the RAM which means the first 56K of the RAM, however, since the 7th block is setup to have the IO section, then only the first 48K of RAM (ABS $00000-$0C000) will be available. The 8th block will be mapped into the last 8K of FLASH from the external FLASH (ABS $FE000-$FFFFF)

Now, bit7 is there to activate the access to the MMU LUT itself, so when it is set to 1b, then, the memory location $0010 to $002F are occupied by the MMU register pages. So, you can read and write to them. To regain access to the RAM below then bit2 of the SYS_CTRL_REG will have to be cleared.

10 C256 FOENIX JR's IO PAGE CTRL - $0001

$0001

b7	b6	b5	b4	b3	b2	b1	b0
RSVD	RSVD	Text Memory Location	Text/ Color Slct	RSVD	Disable I/O Page	Active I/O Page[1]	Active I/O Page[0]

IO PAGE CONTROL REGISTER

Bit[1:0]: 00b = Active IO Page0
01b = Active IO Page1
10b = Active IO Page2
11b = Active IO Page3
Bit[2]: 0 = There is an Active IO Page@$C000 / 1 = Page @ $C000 is RAM or FLASH
Bit[3]: RSVD
Bit[4]: 0 = Text Memory / 1 = Color Memory
Bit[5]: 0 = Text/Color Mem is Located @ $0400 / 1= Text/Color Mem Resides in IO Page 2 & 3
Bit[6]: RSVD
Bit[7]: RSVD

The IO Page management register is there to select the IO that needs to be accessed and if the region of $C000 to $DFFF needs to disable for RAM and/or FLASH. Other bits that are not implemented yet have the usage of bringing the text memory @ $0400 to replicate the behavior of a C64 memory mapping. It hasn't been implemented yet and the jury is still out about if it is a good or bad idea to have a copy of the Screen memory to be located there.

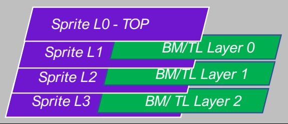
So bit0 and bit1 will determine the active IO page being accessed. See section about the IO Pages mapping for more information

Bit2 has been defined to either enable the IO Section or not. So, when the bit3 is set, then the RAM or FLASH depending on the active MMU LUT Block 6 entry will be accessible, it goes without saying that while this bit is set there can be no work done on any device in the system. So, one might one to keep this 8K block to store data only.

The bit3 to bit7 are presently RESERVED since the memory mapping logic to bring the Text memory down to $0400 is not yet implemented at the moment of writing these words.

11 C256 FOENIX JR's – IO PAGE 0 – I/O REGISTERS DEFINITION

11.1 Tiny VICKY - Master Control Registers - $D000 - $D0FF

<table>
<tr><th>Addy Start Offset</th><th>Addy End</th><th>Size</th><th>Type</th><th>R/W</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>MASTER CONTROL</td></tr>
<tr><td>0xD000</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>VICKY Master Control Register 0 Value @ Reset: 0x01
Control Bit Value:
b[0] – Text Mode Enable
b[1] – Text Mode Overlay Enable
b[2] – Graphic Mode Enable
b[3] – Bitmap Engine Enable
b[4] – Tile Map Engine Enable
b[5] – Sprite Engine Enable
b[6] – GAMMA Enable
b[7] – Not used</td></tr>
<tr><td>0xD001</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>VICKY Master Control Register 1 Value @ Reset: 0x00
Control Bit Value:
b[0] – Text (0 = 640x480@ 60Hz/ 1 = 640x480@ 70) / Graphic Resolution (0 = 320x240@ 60Hz / 1 = 320x200)
b[1] – Double Pixel in X direction (Text Mode Only) (1 = 320 / 0 = 640)
b[2] – Double Pixel in Y direction (Text Mode Only) (1 = 240 or 200 / 0 = 480 or 400)
b[3] – Not used
b[4] – Not used
b[5] – Not used
b[6] – Not used
b[7] – Not used</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>LAYER MANAGER</td></tr>
<tr><td>0xD002</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>VICKY Layer Order Control Register 0
Control Bit Value:
b[0] – Layer0[0] Type - TOP
b[1] – Layer0[1] Type
b[2] – Layer0[2] Type
b[3] – Not used
b[4] – Layer1[0] Type - MID
b[5] – Layer1[1] Type
b[6] – Layer1[2] Type
b[7] – Not used
<table>
<tr><th>Layer Type</th><th>[2]</th><th>[1]</th><th>[0]</th></tr>
<tr><td>Bitmap0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Bitmap1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>Tile Map 0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>Tile Map 1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>Tile Map 2</td><td>1</td><td>0</td><td>0</td></tr>
</table>

</td></tr>
<tr><td>0xD003</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>VICKY Layer Order Control Register 1
Control Bit Value:
b[0] – Layer2[0] Type - BOTTOM
b[1] – Layer2[1] Type
b[2] – Layer2[2] Type
b[3] – Not used
b[4] – Not used
b[5] – Not used
b[6] – Not used
b[7] – Not used
<table>
<tr><th>Layer Type</th><th>[2]</th><th>[1]</th><th>[0]</th></tr>
<tr><td>Bitmap0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Bitmap1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>Tile Map 0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>Tile Map 1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>Tile Map 2</td><td>1</td><td>0</td><td>0</td></tr>
</table>

Sprite L0 - TOP
Sprite L1
BM/TL Layer 0
Sprite L2
BM/TL Layer 1
Sprite L3
BM/ TL Layer 2

</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>BORDER</td></tr>
<tr><td>0xD004</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>Border Control Register
Control Bit Value:
b[0] – Border Enable
b[1] – Not used
b[2] – Not used
b[3] – Not used
b[4] – Border X Scroll[0]</td></tr>
</table>

					b[5] – Border X Scroll[1] b[6] – Border X Scroll[2] b[7] – Not used
0xD005		8	I/O	R/W	Border Color Register - BLUE
0xD006		8	I/O	R/W	Border Color Register - GREEN
0xD007		8	I/O	R/W	Border Color Register - RED
0xD008		8	I/O	R/W	Border X Size (0 to 31pixels) Border Size Bit Value: b[0] – Border X Size[0] b[1] – Border X Size[1] b[2] – Border X Size[2] b[3] – Border X Size[3] b[4] – Border X Size[4] b[5] – Not used b[6] – Not used b[7] – Not used
0xD009		8	I/O	R/W	Border Y Size (0 to 31pixels) Border Size Bit Value: b[0] – Border Y Size[0] b[1] – Border Y Size[1] b[2] – Border Y Size[2] b[3] – Border Y Size[3] b[4] – Border Y Size[4] b[5] – Not used b[6] – Not used b[7] – Not used
0xD00A		8			Reserved
0xD00B		8			Reserved
0xD00C		8			Reserved
					BACKGROUND COLOR
0xD00D		8	I/O	R/W	Background Color Register – BLUE (Graphic Mode Only)
0xD00E		8	I/O	R/W	Background Color Register – GREEN (Graphic Mode Only)
0xD00F		8	I/O	R/W	Background Color Register – RED (Graphic Mode Only)
					CURSOR
0x0010		8	I/O	R/W	Cursor Control Register Control Bit Value: b[0] – Cursor Enable b[1] – Cursor Flash Rate[0] b[2] – Cursor Flash Rate[1] b[3] – Cursor Flash Enable (1 = Cursor Flashes, 0 = No Flash) b[4] – Not used b[5] – Not used b[6] – Not used b[7] – Not used Flash Rate | [1] | [0] 1 Sec | 0 | 0 ½ sec | 0 | 1 ¼ sec | 1 | 0 1/5 sec | 1 | 1
0x0011		8	I/O	R/W	Reserved
0x0012		8	I/O	R/W	Cursor Character
0x0013		8	I/O	R/W	Reserved
0x0014		8	I/O	R/W	Cursor X Position Lo Register [7:0]
0x0015		8	I/O	R/W	Cursor X Position Hi Register [15:8]
0x0016		8	I/O	R/W	Cursor Y Position Lo Register [7:0]
0x0017		8	I/O	R/W	Cursor Y Position Hi Register [15:8]
					LINE INTERRUPT (SOL)
0x0018		8	I/O	W	Line Interrupt Control Register Control Bit Value: b[0] – Line Interrupt Enable b[1] – Not used b[2] – Not used b[3] – Not used b[4] – Not used b[5] – Not used b[6] – Not used b[7] – Not used
0x0019		8	I/O	W	Line Interrupt Line Number Lo [7:0]
0x001A		8	I/O	W	Line Interrupt Line Number Hi [3:0] = [11:8]
0x001B		8	I/O	W	Reserved

					PIXEL/LINE POSITION
0x0018		8	I/O	R	Pixel Position Lo [7:0] Position of the Pixel Pointer (0 – 799) (Includes Blanking)
0x0019		8	I/O	R	Pixel Position Hi [3:0] = [11:8]
0x001A		8	I/O	R	Active Line Number Lo [7:0] Actual Line that is being displayed (0 to 525 / 0 to 449) (Includes Blanking)
0x001B		8	I/O	R	Active Line Number Hi [3:0] = [11:8]

11.2 Tiny VICKY – Bitmap Control Registers - $D100 - $D1FF

Addy Start Offset	Addy End	Size	Type	R/W	Description
					BITMAP LAYER 0
0xD100		8	I/O	R/W	Bitmap Layer0 Control Register Control Bit Value: b[0] – Bitmap Layer 0 Enable b[1] – LUT0 b[2] – LUT1 b[3] – Not used b[4] – Not used b[5] – Not used b[6] – Not used b[7] – Not used
0xD101		8	I/O	R/W	Bitmap Layer0 VRAM Address Pointer Lo [7:0] (Points to the first 256K of SRAM)
0xD102		8	I/O	R/W	Bitmap Layer0 VRAM Address Pointer Med [15:8]
0xD103		8	I/O	R/W	Bitmap Layer0 VRAM Address Pointer Lo [17:16]
0xD104		8	I/O	R/W	Reserved
0xD105		8	I/O	R/W	Reserved
0xD106		8	I/O	R/W	Reserved
0xD107		8	I/O	R/W	Reserved
					BITMAP LAYER 1
0xD108		8	I/O	R/W	Bitmap Layer1 Control Register Control Bit Value: b[0] – Bitmap Layer 1 Enable b[1] – LUT0 b[2] – LUT1 b[3] – Not used b[4] – Not used b[5] – Not used b[6] – Not used b[7] – Not used
0xD109		8	I/O	R/W	Bitmap Layer1 VRAM Address Pointer Lo [7:0] (Points to the first 256K of SRAM)
0xD10A		8	I/O	R/W	Bitmap Layer1 VRAM Address Pointer Med [15:8]
0xD10B		8	I/O	R/W	Bitmap Layer1 VRAM Address Pointer Lo [17:16]
0xD10C		8	I/O	R/W	Reserved
0xD10D		8	I/O	R/W	Reserved
0xD10E		8	I/O	R/W	Reserved
0xD10F		8	I/O	R/W	Reserved

11.3 Tiny VICKY – Tile Map Control Registers - $D200 - $D2FF

Addy Start Offset	Addy End	Size	Type	R/W	Description
					TILE MAP LAYER0
0xD200		8	I/O	W	Tile Map Layer 0 Control Register Control Bit Value: b[0] – Tile Map Layer Enable b[1] – Not used b[2] – Not used b[3] – Not used b[4] – Tile Size (0 = 16x16, 1 = 8x8) b[5] – Not used b[6] – Not used

					b[7] – Not used Note: The LUT is defined in each tile Attributes Bit field [13:11] The Tile Set is also defined in each tile Attributes bit field [10:8]
0xD201		8	I/O	W	Tile Map Layer0 MAP Address Pointer Lo [7:0] (Points to the first 256K of SRAM)
0xD202		8	I/O	W	Tile Map Layer0 MAP Address Pointer Med [15:8]
0xD203		8	I/O	W	Tile Map Layer0 MAP Address Pointer Hi [17:16]
0xD204		8	I/O	W	Tile Map Layer0 X MAP Size [7:0]
0xD205		8	I/O	W	Reserved
0xD206		8	I/O	W	Tile Map Layer0 Y MAP Size [7:0]
0xD207		8	I/O	W	Reserved
0xD208		8	I/O	W	Tile Map Layer0 X MAP Position Lo Position Bit Value: b[0] – Smooth Scroll X[0] in 16x16, Not Used in 8x8 b[1] – Smooth Scroll X[1] in 16x16, X[0] in 8x8 b[2] – Smooth Scroll X[2] in 16x16, X[1] in 8x8 b[3] – Smooth Scroll X[3] in 16x16, X[2] in 8x8 b[4] – Position X[0] b[5] – Position X[1] b[6] – Position X[2] b[7] – Position X[3]
0xD209		8	I/O	W	Tile Map Layer0 X MAP Position Hi Position Bit Value: b[0] – Position X[4] b[1] – Position X[5] b[2] – Position X[6] b[3] – Position X[7] b[4] – Not Used b[5] – Not Used b[6] – Not Used b[7] – Position X[11] – Smooth Scroll Direction (Sign (1 = Negative, Going Left, 0 – Going Right))
0xD20A		8	I/O	W	Tile Map Layer0 Y MAP Position Lo Position Bit Value: b[0] – Smooth Scroll Y[0] in 16x16, Not Used in 8x8 b[1] – Smooth Scroll Y[1] in 16x16, Y[0] in 8x8 b[2] – Smooth Scroll Y[2] in 16x16, Y[1] in 8x8 b[3] – Smooth Scroll Y[3] in 16x16, Y[2] in 8x8 b[4] – Position Y[0] b[5] – Position Y[1] b[6] – Position Y[2] b[7] – Position Y[3]
0xD20B		8	I/O	W	Tile Map Layer0 Y MAP Position Hi Position Bit Value: b[0] – Position Y[4] b[1] – Position Y[5] b[2] – Position Y[6] b[3] – Position Y[7] b[4] – Not Used b[5] – Not Used b[6] – Not Used b[7] – Position Y[11] – Smooth Scroll Direction (Sign (1 = Negative, Going Left, 0 – Going Right))
					TILE MAP LAYER1
0xD20C		8	I/O	W	Tile Map Layer 1 Control Register Control Bit Value: b[0] – Tile Map Layer Enable b[1] – Not used b[2] – Not used b[3] – Not used b[4] – Tile Size (0 = 16x16, 1 = 8x8) b[5] – Not used b[6] – Not used b[7] – Not used Note: The LUT is defined in each tile Attributes Bit field [13:11] The Tile Set is also defined in each tile Attributes bit field [10:8]
0xD20D		8	I/O	W	Tile Map Layer1 MAP Address Pointer Lo [7:0] (Points to the first 256K of SRAM)
0xD20E		8	I/O	W	Tile Map Layer1 MAP Address Pointer Med [15:8]
0xD20F		8	I/O	W	Tile Map Layer1 MAP Address Pointer Hi [17:16]
0xD210		8	I/O	W	Tile Map Layer1 X MAP Size [7:0]
0xD211		8	I/O	W	Reserved
0xD212		8	I/O	W	Tile Map Layer1 Y MAP Size [7:0]

0xD213		8	I/O	W	Reserved
0xD214		8	I/O	W	Tile Map Layer1 X MAP Position Lo Position Bit Value: b[0] – Smooth Scroll X[0] in 16x16, Not Used in 8x8 b[1] – Smooth Scroll X[1] in 16x16, X[0] in 8x8 b[2] – Smooth Scroll X[2] in 16x16, X[1] in 8x8 b[3] – Smooth Scroll X[3] in 16x16, X[2] in 8x8 b[4] – Position X[0] b[5] – Position X[1] b[6] – Position X[2] b[7] – Position X[3]
0xD215		8	I/O	W	Tile Map Layer1 X MAP Position Hi Position Bit Value: b[0] – Position X[4] b[1] – Position X[5] b[2] – Position X[6] b[3] – Position X[7] b[4] – Not Used b[5] – Not Used b[6] – Not Used b[7] – Position X[11] – Smooth Scroll Direction (Sign (1 = Negative, Going Left, 0 – Going Right))
0xD216		8	I/O	W	Tile Map Layer1 Y MAP Position Lo Position Bit Value: b[0] – Smooth Scroll Y[0] in 16x16, Not Used in 8x8 b[1] – Smooth Scroll Y[1] in 16x16, Y[0] in 8x8 b[2] – Smooth Scroll Y[2] in 16x16, Y[1] in 8x8 b[3] – Smooth Scroll Y[3] in 16x16, Y[2] in 8x8 b[4] – Position Y[0] b[5] – Position Y[1] b[6] – Position Y[2] b[7] – Position Y[3]
0xD217		8	I/O	W	Tile Map Layer1 Y MAP Position Hi Position Bit Value: b[0] – Position Y[4] b[1] – Position Y[5] b[2] – Position Y[6] b[3] – Position Y[7] b[4] – Not Used b[5] – Not Used b[6] – Not Used b[7] – Position Y[11] – Smooth Scroll Direction (Sign (1 = Negative, Going Left, 0 – Going Right))
					TILE MAP LAYER2
0xD218		8	I/O	W	Tile Map Layer 2 Control Register Control Bit Value: b[0] – Tile Map Layer Enable b[1] – Not used b[2] – Not used b[3] – Not used b[4] – Tile Size (0 = 16x16, 1 = 8x8) b[5] – Not used b[6] – Not used b[7] – Not used Note: The LUT is defined in each tile Attributes Bit field [13:11] The Tile Set is also defined in each tile Attributes bit field [10:8]
0xD219		8	I/O	W	Tile Map Layer2 MAP Address Pointer Lo [7:0] (Points to the first 256K of SRAM)
0xD21A		8	I/O	W	Tile Map Layer2 MAP Address Pointer Med [15:8]
0xD21B		8	I/O	W	Tile Map Layer2 MAP Address Pointer Hi [17:16]
0xD21C		8	I/O	W	Tile Map Layer2 X MAP Size [7:0]
0xD21D		8	I/O	W	Reserved
0xD21E		8	I/O	W	Tile Map Layer2 Y MAP Size [7:0]
0xD21F		8	I/O	W	Reserved
0xD220		8	I/O	W	Tile Map Layer2 X MAP Position Lo Position Bit Value: b[0] – Smooth Scroll X[0] in 16x16, Not Used in 8x8 b[1] – Smooth Scroll X[1] in 16x16, X[0] in 8x8 b[2] – Smooth Scroll X[2] in 16x16, X[1] in 8x8 b[3] – Smooth Scroll X[3] in 16x16, X[2] in 8x8 b[4] – Position X[0] b[5] – Position X[1] b[6] – Position X[2] b[7] – Position X[3]

0xD221		8	I/O	W	Tile Map Layer2 X MAP Position Hi Position Bit Value: b[0] – Position X[4] b[1] – Position X[5] b[2] – Position X[6] b[3] – Position X[7] b[4] – Not Used b[5] – Not Used b[6] – Not Used b[7] – Position X[11] – Smooth Scroll Direction (Sign (1 = Negative, Going Left, 0 – Going Right))
0xD222		8	I/O	W	Tile Map Layer2 Y MAP Position Lo Position Bit Value: b[0] – Smooth Scroll Y[0] in 16x16, Not Used in 8x8 b[1] – Smooth Scroll Y[1] in 16x16, Y[0] in 8x8 b[2] – Smooth Scroll Y[2] in 16x16, Y[1] in 8x8 b[3] – Smooth Scroll Y[3] in 16x16, Y[2] in 8x8 b[4] – Position Y[0] b[5] – Position Y[1] b[6] – Position Y[2] b[7] – Position Y[3]
0xD223		8	I/O	W	Tile Map Layer2 Y MAP Position Hi Position Bit Value: b[0] – Position Y[4] b[1] – Position Y[5] b[2] – Position Y[6] b[3] – Position Y[7] b[4] – Not Used b[5] – Not Used b[6] – Not Used b[7] – Position Y[11] – Smooth Scroll Direction (Sign (1 = Negative, Going Left, 0 – Going Right))
					TILE MAP GRAPHICS
0xD280		8	I/O	W	Tile Graphics Set Addy Pointer 0 Lo[7:0]
0xD281		8	I/O	W	Tile Graphics Set Addy Pointer 0 Med[15:8]
0xD282		8	I/O	W	Tile Graphics Set Addy Pointer 0 Hi[17:16]
0xD283		8	I/O	W	Reserved
0xD284		8	I/O	W	Tile Graphics Set Addy Pointer 1 Lo[7:0]
0xD285		8	I/O	W	Tile Graphics Set Addy Pointer 1 Med[15:8]
0xD286		8	I/O	W	Tile Graphics Set Addy Pointer 1 Hi[17:16]
0xD287		8	I/O	W	Reserved
0xD288		8	I/O	W	Tile Graphics Set Addy Pointer 2 Lo[7:0]
0xD289		8	I/O	W	Tile Graphics Set Addy Pointer 2 Med[15:8]
0xD28A		8	I/O	W	Tile Graphics Set Addy Pointer 2 Hi[17:16]
0xD28B		8	I/O	W	Reserved
0xD28C		8	I/O	W	Tile Graphics Set Addy Pointer 3 Lo[7:0]
0xD28D		8	I/O	W	Tile Graphics Set Addy Pointer 3 Med[15:8]
0xD28E		8	I/O	W	Tile Graphics Set Addy Pointer 3 Hi[17:16]
0xD28F		8	I/O	W	Reserved
0xD290		8	I/O	W	Tile Graphics Set Addy Pointer 4 Lo[7:0]
0xD291		8	I/O	W	Tile Graphics Set Addy Pointer 4 Med[15:8]
0xD292		8	I/O	W	Tile Graphics Set Addy Pointer 4 Hi[17:16]
0xD293		8	I/O	W	Reserved
0xD294		8	I/O	W	Tile Graphics Set Addy Pointer 5 Lo[7:0]
0xD295		8	I/O	W	Tile Graphics Set Addy Pointer 5 Med[15:8]
0xD296		8	I/O	W	Tile Graphics Set Addy Pointer 5 Hi[17:16]
0xD297		8	I/O	W	Reserved
0xD298		8	I/O	W	Tile Graphics Set Addy Pointer 6 Lo[7:0]
0xD299		8	I/O	W	Tile Graphics Set Addy Pointer 6 Med[15:8]
0xD29A		8	I/O	W	Tile Graphics Set Addy Pointer 6 Hi[17:16]
0xD29B		8	I/O	W	Reserved
0xD29C		8	I/O	W	Tile Graphics Set Addy Pointer 7 Lo[7:0]
0xD29D		8	I/O	W	Tile Graphics Set Addy Pointer 7 Med[15:8]
0xD29E		8	I/O	W	Tile Graphics Set Addy Pointer 7 Hi[17:16]
0xD29F		8	I/O	W	Reserved

11.4 SID Left Channel - $D400 - $D4FF

Addy Start	Addy End	Size	Type	R/W	Description
					LEFT Channel (Registers summarized)
0xD400		8	I/O	W	Voice 1 - FREQ LOW
0xD401		8	I/O	W	Voice 1 - FREQ HI
0xD402		8	I/O	W	Voice 1 – PW LO
0xD403		8	I/O	W	Voice 1 – PW HI
0xD404		8	I/O	W	Voice 1 - Control
0xD405		8	I/O	W	Voice 1 – Attack / Decay
0xD406		8	I/O	W	Voice 1 – Sustain / Release
0xD407		8	I/O	W	Voice 2 - FREQ LOW
0xD408		8	I/O	W	Voice 2 - FREQ HI
0xD409		8	I/O	W	Voice 2 – PW LO
0xD40A		8	I/O	W	Voice 2 – PW HI
0xD40B		8	I/O	W	Voice 2 - Control
0xD40C		8	I/O	W	Voice 2 – Attack / Decay
0xD40D		8	I/O	W	Voice 2 – Sustain / Release
0xD40E		8	I/O	W	Voice 3 - FREQ LOW
0xD40F		8	I/O	W	Voice 3 - FREQ HI
0xD410		8	I/O	W	Voice 3 – PW LO
0xD411		8	I/O	W	Voice 3 – PW HI
0xD412		8	I/O	W	Voice 3 - Control
0xD413		8	I/O	W	Voice 3 – Attack / Decay
0xD414		8	I/O	W	Voice 3 – Sustain / Release
0xD415		8	I/O	W	Filter – FC LOW
0xD416		8	I/O	W	Filter – FC HI
0xD417		8	I/O	W	Filter – RES / FILT
0xD418		8	I/O	W	Filter – Mode / VOL
0xD419		8	I/O	W	POT X (not Supported)
0xD41A		8	I/O	W	POT Y (not Supported)
0xD41B		8	I/O	W	OSC3 / RANDOM
0xD41C		8	I/O	W	ENV3
0xD41D		8	I/O	W	Reserved
0xD41E		8	I/O	W	Reserved
0xD41F		8	I/O	W	Reserved

11.5 SID Right Channel - $D500 - $D5FF

Addy Start	Addy End	Size	Type	R/W	Description
					RIGHT Channel (Registers summarized)
0xD500		8	I/O	W	Voice 1 - FREQ LOW
0xD501		8	I/O	W	Voice 1 - FREQ HI
0xD502		8	I/O	W	Voice 1 – PW LO
0xD503		8	I/O	W	Voice 1 – PW HI
0xD504		8	I/O	W	Voice 1 - Control
0xD505		8	I/O	W	Voice 1 – Attack / Decay
0xD506		8	I/O	W	Voice 1 – Sustain / Release
0xD507		8	I/O	W	Voice 2 - FREQ LOW
0xD508		8	I/O	W	Voice 2 - FREQ HI
0xD509		8	I/O	W	Voice 2 – PW LO
0xD50A		8	I/O	W	Voice 2 – PW HI
0xD50B		8	I/O	W	Voice 2 - Control
0xD50C		8	I/O	W	Voice 2 – Attack / Decay
0xD50D		8	I/O	W	Voice 2 – Sustain / Release
0xD50E		8	I/O	W	Voice 3 - FREQ LOW
0xD50F		8	I/O	W	Voice 3 - FREQ HI
0xD510		8	I/O	W	Voice 3 – PW LO
0xD511		8	I/O	W	Voice 3 – PW HI
0xD512		8	I/O	W	Voice 3 - Control
0xD513		8	I/O	W	Voice 3 – Attack / Decay
0xD514		8	I/O	W	Voice 3 – Sustain / Release
0xD515		8	I/O	W	Filter – FC LOW
0xD516		8	I/O	W	Filter – FC HI
0xD517		8	I/O	W	Filter – RES / FILT
0xD518		8	I/O	W	Filter – Mode / VOL
0xD519		8	I/O	W	POT X (not Supported)
0xD51A		8	I/O	W	POT Y (not Supported)
0xD51B		8	I/O	W	OSC3 / RANDOM
0xD51C		8	I/O	W	ENV3
0xD51D		8	I/O	W	Reserved
0xD51E		8	I/O	W	Reserved
0xD51F		8	I/O	W	Reserved

11.6 PSG Left Channel - $D600 - $D60F

Addy Start	Addy End				
Offset		Size	Type	R/W	Description
0xD600		8	I/O	W	Internal (FPGA) PSG – LEFT Channel

11.7 PSG Right Channel - $D610 - $D61F

Addy Start	Addy End				
Offset		Size	Type	R/W	Description
0xD610		8	I/O	W	Internal (FPGA) PSG – RIGHT Channel

11.8 CODEC - $D620 - $D62F

Addy Start	Addy End	Size	Type	R/W	Description
0xD620		8	I/O	W	DATA Register [7:0]
0xD621		8	I/O	W	DATA Register [15:8] (Data is serialized to the CODEC after the Write Transaction is completed)
0xD620		8	I/O	R	STATUS Status Bit Value: b[0] - Not Used b[1] - Not Used b[2] - Not Used b[3] - Not Used b[4] - Not Used b[5] - Not Used b[6] - Not Used b[7] - BUSY Check for the busy flag to go back to '0' before sending another command

11.9 UART - $D630 - $D63F

Addy Start Offset	Addy End	Size	Type	R/W	Description
					Simple UART (Compatible with 16550)
0xD630		8	I/O	R/W	(RHR) Receiver Holding Register (R) (THR) Transmitter Holding Register (W)
0xD631		8	I/O	R/W	(IER) Interrupt Enable Register
0xD632		8	I/O	R/W	(ISR) Interrupt Status Register (R) (FCR) FIFO Control Register (FIFO is 16 Bytes Deep) (W)
0xD633		8	I/O	R/W	(LCR) Line Control Register
0xD634		8	I/O	R/W	(MCR) Modem Control Register
0xD635		8	I/O	R	(LSR) Line Status Register
0xD636		8	I/O	R/W	(MSR) Modem Status Register
0xD637		8	I/O	R/W	(SPR) Scratch Pad Register
					When DLAB = 1
0xD630		8	I/O	R/W	(DLL) Baud rate Divisor's Constant LSB
0xD631		8	I/O	R/W	(DLM) Baud rate Divisor's Constant MSB
0xD632		8	I/O	W	(PSD) Pre-Scaler Division

11.10 PS2 - $D640 - $D64F

Addy Start Offset	Addy End	Size	Type	R/W	Description
					PS2 Controller
0xD640		8	I/O	R/W	Keyboard/Mouse Output Buffer (W) Keyboard/Mouse Input Buffer (R) Keyboard/Mouse Data Buffer (R/W)
0xD644		8	I/O	R/W	Keyboard/Mouse Status Port (R) Keyboard/Mouse CMD Port (W)

11.11 TIMER CONTROLLER - $D650 - $D65F

Addy Start Offset	Addy End	Size	Type	R/W	Description
					Timer 0 – Clock Source – Tiny VICKY Internal Clock = 25.175Mhz
0xD650		8	I/O	W	Control Register 0 Value @ Reset: 0x00 Control Bit Value: b[0] – Timer 0 Enable b[1] – Timer 0 Clear b[2] – Timer 0 Load b[3] – Timer 0 Up/Down b[4] – Timer 0 Reclear b[5] – Timer 0 Reload b[6] – Not Used b[7] – Timer 0 Interrupt Enable
0xD650		8	I/O	R	Status Register 0 Value @ Reset: 0x00 Status Bit Value: b[0] – Timer 0 Compare (1 = Equal, 0 = Not Equal) b[1] – Not Used b[2] – Not Used b[3] – Not Used b[4] – Not Used b[5] – Not Used b[6] – Not Used b[7] – Not Used
0xD651		8	I/O	R/W	TIMER0 – VALUE – Low [7:0]
0xD652		8	I/O	R/W	TIMER0 – VALUE – Med [15:8]
0xD653		8	I/O	R/W	TIMER0 – VALUE – Hi [23:16]
0xD654		8	I/O	R/W	TIMER0 Compare Control Register
0xD655		8	I/O	R/W	TIMER0 – COMPARE VALUE – Low [7:0]
0xD656		8	I/O	R/W	TIMER0 – COMPARE VALUE – Med [15:8]
0xD657		8	I/O	R/W	TIMER0 – COMPARE VALUE – Hi [23:16]
					Timer 1 – Clock Source – SOF (60Hz or 70Hz) "Be Careful if you change the Resolution"
0xD658		8	I/O	W	Control Register 1 Value @ Reset: 0x00 Control Bit Value: b[0] – Timer 1 Enable b[1] – Timer 1 Clear b[2] – Timer 1 Load b[3] – Timer 1 Up/Down b[4] – Timer 1 Reclear b[5] – Timer 1 Reload b[6] – Not Used b[7] – Timer 1 Interrupt Enable
0xD658		8	I/O	R	Status Register 1 Value @ Reset: 0x00 Status Bit Value: b[0] – Timer 1 Compare (1 = Equal, 0 = Not Equal) b[1] – Not Used b[2] – Not Used b[3] – Not Used b[4] – Not Used b[5] – Not Used b[6] – Not Used b[7] – Not Used
0xD659		8	I/O	R/W	TIMER1 – VALUE – Low [7:0]
0xD65A		8	I/O	R/W	TIMER1 – VALUE – Med [15:8]
0xD65B		8	I/O	R/W	TIMER1 – VALUE – Hi [23:16]
0xD65C		8	I/O	R/W	TIMER1 Compare Control Register
0xD65D		8	I/O	R/W	TIMER1 – COMPARE VALUE – Low [7:0]
0xD65E		8	I/O	R/W	TIMER1 – COMPARE VALUE – Med [15:8]
0xD65F		8	I/O	R/W	TIMER1 – COMPARE VALUE – Hi [23:16]

11.12 INTERRUPT CONTROLLER - $D660 - $D66F

<table>
<tr><th>Addy Start Offset</th><th>Addy End</th><th>Size</th><th>Type</th><th>R/W</th><th>Description</th></tr>
<tr><td>0xD660</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>Interrupt Pending Register [7:0]
<table>
<tr><th>Interrupt Source</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>VICKY INT0 (SOF)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>VICKY INT1 (SOL)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>PS2 Keyboard</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>PS2 Mouse</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Timer0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Timer1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>DMA INT</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Reserved</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>
Value @ Reset: 0x0000</td></tr>
<tr><td>0xD661</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>Interrupt Pending Register [15:8]
<table>
<tr><th>Interrupt Source</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th></tr>
<tr><td>UART</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>VICKY INT2 (TBD)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>VICKY INT3 (TBD)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>VICKY INT4 (TBD)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>RTC</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>VIA (65C22)</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>IEC</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>SD Card Insert</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>
Value @ Reset: 0x00</td></tr>
<tr><td>0xD662</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>Polarity Register [7:0] (Not in Use) Value @ Reset: 0x00</td></tr>
<tr><td>0xD663</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>Polarity Register [15:8] (Not in Use) Value @ Reset: 0x00</td></tr>
<tr><td>0xD664</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>EDGE Register [7:0] (Not in Use) Value @ Reset: 0xFF</td></tr>
<tr><td>0xD665</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>EDGE Register [15:8] (Not in Use) Value @ Reset: 0xFF</td></tr>
<tr><td>0xD666</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>MASK Register [7:0] Value @ Reset: 0xFF</td></tr>
<tr><td>0xD667</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>MASK Register [15:8] Value @ Reset: 0xFF</td></tr>
</table>

11.13 DIP SWITCH - $D670 - $D67F

<table>
<tr><th>Addy Start</th><th>Addy End</th><th>Size</th><th>Type</th><th>R/W</th><th>Description</th></tr>
<tr><td>0xD670</td><td></td><td>8</td><td>I/O</td><td>R</td><td>Dip Switch Value
(by default, all values are '1') Pressing a switch force the input to ground.
<table>
<tr><th>Switch Bits Value</th></tr>
<tr><td>b[0] - BOOT0</td></tr>
<tr><td>b[1] - BOOT1</td></tr>
<tr><td>b[2] - BOOT2</td></tr>
<tr><td>b[3] - BOOT3</td></tr>
<tr><td>b[4] - USER0</td></tr>
<tr><td>b[5] - USER1</td></tr>
<tr><td>b[6] - USER2</td></tr>
<tr><td>b[7] - GAMMA CORRECTION</td></tr>
</table></td></tr>
</table>

11.14 IEC CONTROLLER - $D680 - $D68F

Addy Start	Addy End	Size	Type	R/W	Description
					WRITE
0xD680		8	I/O	W	TALKER CMD ($28, $48, $6x, $Ex, $Fx, Etc…) (Not $3F or $5F)
0xD681		8	I/O	W	TALKER CMD LAST ($3F, $5F Only)
0xD682		8	I/O	W	TALKER DATA (All Data ought to be written here, save the last Byte)
0xD683		8	I/O	W	TALKER DATA LAST (Last Byte or 1 Byte to Send should be written here)
					READ
0xD680		8	I/O	R	LISTENER DATA (FIFO Output, FIFO Holds 512Bytes)
0xD681		8	I/O	R	LISTENER FIFO STAT Status Bit Value: b[0] – Read FIFO Empty Flag (1= Empty, 0 = Data in FIFO) b[1] – Read FIFO Full Flag (1 = FIFO, 0 = Space Left) – If you get FIFO Full while writing, you will lose data b[2] – Last Byte Flag (1 = This is the last byte of the packet) - Check after reading the Data from FIFO - (updated at the time of read) b[3] – Not Used b[4] - Device Not Present (Send the Command $28 or $48, then check this flag before sending the next command) (1 = No Device Attached) b[5] - Not Used b[6] - Not Used b[7] - Not Used
0xD682		8	I/O	R	LISTENER FIFO COUNT LOW [7:0] Bit [7:0] = READ FIFO Count[7:0]
0xD683		8	I/O	R	LISTENER FIFO COUNT HI [8] Bit [0] = READ FIFO Count[8]

11.15 RTC - $D690 - $D69F

<table>
<tr><th>Addy Start</th><th>Addy End</th><th>Size</th><th>Type</th><th>R/W</th><th>Description</th></tr>
<tr><td>0xD690</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Seconds Register
<table>
<tr><td></td><td></td><td colspan="3">10 Sec Digit</td><td colspan="4">1 Sec Digit</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Seconds (00-59)</td><td>0</td><td>T</td><td>T</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr>
</table>
- T = Tens, U = Units</td></tr>
<tr><td>0xD691</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Seconds Alarm
<table>
<tr><td></td><td></td><td colspan="3">10 Sec Digit</td><td colspan="4">1 Sec Digit</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Seconds alarm</td><td rowspan="2">ALM1</td><td>ALM0</td><td>T</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr>
<tr><td>Seconds alarm</td><td>T</td><td>T</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr>
</table>
- T = Tens, U = Units</td></tr>
<tr><td>0xD692</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Minutes Register
<table>
<tr><td></td><td></td><td colspan="3">10 Min Digit</td><td colspan="4">1 Min Digit</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Minutes (00-59)</td><td>0</td><td>T</td><td>T</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr>
</table>
- T = Tens, U = Units</td></tr>
<tr><td>0xD693</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Minutes Alarm
<table>
<tr><td></td><td></td><td colspan="3">10 Min Digit</td><td colspan="4">1 Sec Digit</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Minutes alarm</td><td rowspan="2">ALM1</td><td>ALM0</td><td>T</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr>
<tr><td>Minutes alarm</td><td>T</td><td>T</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr>
</table>
- T = Tens, U = Units</td></tr>
<tr><td>0xD694</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Hours Register
<table>
<tr><td></td><td></td><td colspan="3">10 Hour Digit</td><td colspan="4">1 Hour Digit</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Hours (01-12AM)
Hours (81-92PM)</td><td>PM/AM</td><td>0</td><td>T</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr>
</table>
- T = Tens, U = Units</td></tr>
<tr><td>0xD695</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Hours Alarm
<table>
<tr><td></td><td></td><td colspan="3">10 Hour Digit</td><td colspan="4">1 Hour Digit</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Hours (01-12AM)
Hours (81-92PM)</td><td>PM/AM
ALM1</td><td>ALM0</td><td>T</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr>
</table>
- T = Tens, U = Units</td></tr>
<tr><td>0xD696</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Day
<table>
<tr><td></td><td></td><td colspan="3">10 Day Digit</td><td colspan="4">1 Day Digit</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Day (01-31)</td><td>0</td><td>0</td><td>T</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr>
</table>
- T = Tens, U = Units</td></tr>
<tr><td>0xD697</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Day Alarm
<table>
<tr><td></td><td></td><td colspan="3">10 Day Digit</td><td colspan="4">1 Day Digit</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Day (01-31)</td><td>ALM1</td><td>ALM0</td><td>T</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr>
</table>
- T = Tens, U = Units</td></tr>
<tr><td>0xD698</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Day of Week
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="3">Day of Week</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Day of Week
(01-07)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>U</td><td>U</td><td>U</td></tr>
</table>
- U = Units</td></tr>
<tr><td>0xD699</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Month
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td colspan="4">Month Digit</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Month (01-12)</td><td>0</td><td>0</td><td>0</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr>
</table>
- T = Tens, U = Units</td></tr>
<tr><td>0xD69A</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Year
<table>
<tr><td></td><td colspan="4">10 Year Digit</td><td colspan="4">1 Year Digit</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Years (99-00)</td><td>T</td><td>T</td><td>T</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr>
</table>
- T = Tens, U = Units</td></tr>
<tr><td>0xD69B</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Rates
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Rates</td><td>0</td><td>WD2</td><td>WD1</td><td>WD0</td><td>RES3</td><td>RES2</td><td>RES1</td><td>RES0</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>0xD69C</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Enables<table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>Enables</td><td>0</td><td>0</td><td>0</td><td>0</td><td>AIE</td><td>PIE</td><td>PWRIE</td><td>ABE</td></tr></table></td></tr>
<tr><td>0xD69D</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Flags<table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>Flags</td><td>0</td><td>0</td><td>0</td><td>0</td><td>AF</td><td>PF</td><td>PWRF</td><td>BVF</td></tr></table></td></tr>
<tr><td>0xD69E</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Control<table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>Control</td><td>0</td><td>0</td><td>0</td><td>0</td><td>UTI</td><td>STOPn</td><td>24/12</td><td>DSE</td></tr></table></td></tr>
<tr><td>0xD69F</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>RTC – Century<table><tr><td></td><td colspan="4">10 Year Digit</td><td colspan="4">1 Year Digit</td></tr><tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>Century (99-00)</td><td>T</td><td>T</td><td>T</td><td>T</td><td>U</td><td>U</td><td>U</td><td>U</td></tr></table>T = Tens, U = Units</td></tr>
</table>

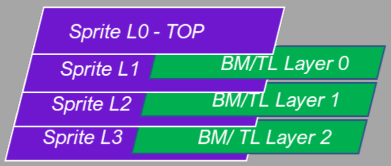
11.16 System Control Registers (Leds, Buzzer, LFSR, Etc…) - $D6A0 - $D6AF

Addy Start	Addy End	Size	Type	R/W	Description
0xD6A0		8	I/O	R/W	System Control Register 0 Control Register 0 b[0] – Power LED (1 – On, 0 – Off) b[1] – SD Card LED (1 – On, 0 – Off) b[2] – Status LED0 (1 – On, 0 – Off) When in manual Control Mode b[3] – Status LED1 (1 – On, 0 – Off) When in manual Control Mode b[4] – Buzzer (1 – On, 0 – Off) – Needs to Modulated to get Sound (No Internal Modulator) b[5] – Not Used b[6] – Not Used b[7] – Manual Reset if SCR2 and SCR3 = 0xDEAD Value @ Reset: 0x01
0xD6A1		8	I/O	R/W	System Control Register 1 Control Register 1 b[0] – Status LED 0 – Auto Flash / Manual (0 – Auto, 1 – Manual) b[1] – Status LED 1 – Auto Flash / Manual (0 – Auto, 1 – Manual) b[2] – Not Used b[3] – Not Used b[4] – LED0 Flash Rate 0 (00 = 1sec, 01=0.5sec, 10=0.4sec, 11=0.2sec) b[5] – LED0 Flash Rate 1 b[6] – LED1 Flash Rate 0 (00 = 1sec, 01=0.5sec, 10=0.4sec, 11=0.2sec) b[7] – LED1 Flash Rate 1 Value @ Reset: 0x40
0xD6A2		8	I/O	R/W	System Control Register 2 Value @ Reset: 0x00 Write the Value: 0xDE to unlock the Manual Reset
0xD6A3		8	I/O	R/W	System Control Register 3 Value @ Reset: 0x00 Write the Value: 0xAD to unlock the Manual Reset
0xD6A4		8	I/O	R/W	READ: LFSR DATA OUT [7:0] – WRITE: SEED VALUE [7:0] Every time you read this register after the LFSR has been enabled and the Seed setup, you will get a new random value.
0xD6A5		8	I/O	R/W	READ: LFSR DATA OUT [15:8] – WRITE: SEED VALUE[15:8] Every time you read this register after the LFSR has been enabled and the Seed setup, you will get a new random value.
0xD6A6		8	I/O	R/W	READ: LFSR Status Register b[0] – Not Used b[1] – Not Used b[2] – Not Used b[3] – Not Used b[4] – Not Used b[5] – Not Used b[6] – Not Used b[7] – LFSR Done WRITE - LFSR Control Register LFSR Control Register b[0] – Enable LFSR (1 – Enable, 0 – Disabled) b[1] – SEED Value Write, After Writing a Seed Value, toggle that bit for it to be registered in the LFSR b[2] – Not Used b[3] – Not Used b[4] – Not Used b[5] – Not Used b[6] – Not Used b[7] – Not Used
0xD6A7		8	I/O	R	Machine ID Possible Value for Machine ID:

Machine	Bit 3	Bit 2	Bit 1	Bit 0
FMX	0	0	0	0
C256U	0	0	0	1
C256 JR	0	0	1	0
A2560 Dev	0	0	1	1
GEN X	0	1	0	0
C256U+	0	1	0	1

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td><table><tr><td>Reserved</td><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>Reserved</td><td>0</td><td>1</td><td>1</td><td>1</td></tr><tr><td>A2560X</td><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>A2560U</td><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>A2560M</td><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>A2560K</td><td>1</td><td>0</td><td>1</td><td>1</td></tr></table>Returned Value (Hardcoded) : 0x02</td></tr>
<tr><td>0xD6A8</td><td></td><td>8</td><td>I/O</td><td>R</td><td>PCB Hardware Version in ASCII – Returned Value (Hardcoded) : 0x41 ‘A’</td></tr>
<tr><td>0xD6A9</td><td></td><td>8</td><td>I/O</td><td>R</td><td>PCB Hardware Version in ASCII – Returned Value (Hardcoded) : 0x30 ‘0</td></tr>
<tr><td>0xD6AA</td><td></td><td>8</td><td>I/O</td><td>R</td><td>CHIP SUB_VERSION LO – Returned Value in BCD</td></tr>
<tr><td>0xD6AB</td><td></td><td>8</td><td>I/O</td><td>R</td><td>CHIP SUB_VERSION HI – Returned Value in BCD</td></tr>
<tr><td>0xD6AC</td><td></td><td>8</td><td>I/O</td><td>R</td><td>CHIP VERSION LO – Returned Value in BCD</td></tr>
<tr><td>0xD6AD</td><td></td><td>8</td><td>I/O</td><td>R</td><td>CHIP VERSION HI – Returned Value in BCD</td></tr>
<tr><td>0xD6AE</td><td></td><td>8</td><td>I/O</td><td>R</td><td>CHIP NUMBER LO – Returned Value in BCD – ‘0x99’</td></tr>
<tr><td>0xD6AF</td><td></td><td>8</td><td>I/O</td><td>R</td><td>CHIP NUMBER HI – Returned Value in BCD – ‘0x95’</td></tr>
</table>

11.17 Tiny VICKY – SPRITES REGISTERS - $D900 - $DAFF

<table>
<tr><th>Addy Start</th><th>Addy End</th><th>Size</th><th>Type</th><th>R/W</th><th>Description</th></tr>
<tr><td>0xD900</td><td></td><td>8</td><td>I/O</td><td>W</td><td>Sprite 0 Control Register
<table>
<tr><th>Control Bit Value:</th></tr>
<tr><td>b[0] – Sprite 0 Enable</td></tr>
<tr><td>b[1] – LUT[0]</td></tr>
<tr><td>b[2] – LUT[1]</td></tr>
<tr><td>b[3] – Depth[0]</td></tr>
<tr><td>b[4] – Depth[1]</td></tr>
<tr><td>b[5] – Size[0]</td></tr>
<tr><td>b[6] – Size[1]</td></tr>
<tr><td>b[7] – Reserved</td></tr>
</table>


<table>
<tr><th>Size</th><th>[1]</th><th>[0]</th></tr>
<tr><td>32x32</td><td>0</td><td>0</td></tr>
<tr><td>24x24</td><td>0</td><td>1</td></tr>
<tr><td>16x16</td><td>1</td><td>0</td></tr>
<tr><td>8x8</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>0xD901</td><td></td><td>8</td><td>I/O</td><td>W</td><td>Sprite 0 Graphics Pointer Addy Lo[7:0]</td></tr>
<tr><td>0xD902</td><td></td><td>8</td><td>I/O</td><td>W</td><td>Sprite 0 Graphics Pointer Addy Med[15:8]</td></tr>
<tr><td>0xD903</td><td></td><td>8</td><td>I/O</td><td>W</td><td>Sprite 0 Graphics Pointer Addy Hi[17:16]</td></tr>
<tr><td>0xD904</td><td></td><td>8</td><td>I/O</td><td>W</td><td>Sprite 0 X Position Lo[7:0]
Note: The position 0,0 of a sprite is -32, -32 offscreen</td></tr>
<tr><td>0xD905</td><td></td><td>8</td><td>I/O</td><td>W</td><td>Sprite 0 X Position Hi[8]
Note: The position 0,0 of a sprite is -32, -32 offscreen</td></tr>
<tr><td>0xD906</td><td></td><td>8</td><td>I/O</td><td>W</td><td>Sprite 0 Y Position Lo[7:0]
Note: The position 0,0 of a sprite is -32, -32 offscreen</td></tr>
<tr><td>0xD907</td><td></td><td>8</td><td>I/O</td><td>W</td><td>Sprite 0 Y Position Hi[8]
Note: The position 0,0 of a sprite is -32, -32 offscreen</td></tr>
<tr><td>0xD908</td><td>0xD90F</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 1</td></tr>
<tr><td>0xD910</td><td>0xD917</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 2</td></tr>
<tr><td>0xD918</td><td>0xD91F</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 3</td></tr>
<tr><td>0xD920</td><td>0xD927</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 4</td></tr>
<tr><td>0xD928</td><td>0xD92F</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 5</td></tr>
<tr><td>0xD930</td><td>0xD937</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 6</td></tr>
<tr><td>0xD938</td><td>0xD93F</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 7</td></tr>
<tr><td>0xD940</td><td>0xD947</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 8</td></tr>
<tr><td>0xD948</td><td>0xD94F</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 9</td></tr>
<tr><td>0xD950</td><td>0xD957</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 10</td></tr>
<tr><td>0xD958</td><td>0xD95F</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 11</td></tr>
<tr><td>0xD960</td><td>0xD967</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 12</td></tr>
<tr><td>0xD968</td><td>0xD96F</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 13</td></tr>
<tr><td>0xD970</td><td>0xD977</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 14</td></tr>
<tr><td>0xD978</td><td>0xD97F</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 15</td></tr>
<tr><td>0xD980</td><td>0xD987</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 16</td></tr>
<tr><td>0xD988</td><td>0xD98F</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 17</td></tr>
<tr><td>0xD990</td><td>0xD997</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 18</td></tr>
<tr><td>0xD998</td><td>0xD99F</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 19</td></tr>
<tr><td>0xD9A0</td><td>0xD9A7</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 20</td></tr>
<tr><td>0xD9A8</td><td>0xD9AF</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 21</td></tr>
<tr><td>0xD9B0</td><td>0xD9B7</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 22</td></tr>
<tr><td>0xD9B8</td><td>0xD9BF</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 23</td></tr>
<tr><td>0xD9C0</td><td>0xD9C7</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 24</td></tr>
<tr><td>0xD9C8</td><td>0xD9CF</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 25</td></tr>
<tr><td>0xD9D0</td><td>0xD9D7</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 26</td></tr>
<tr><td>0xD9D8</td><td>0xD9DF</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 27</td></tr>
<tr><td>0xD9E0</td><td>0xD9E7</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 28</td></tr>
<tr><td>0xD9E8</td><td>0xD9EF</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 29</td></tr>
<tr><td>0xD9F0</td><td>0xD9F7</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 30</td></tr>
<tr><td>0xD9F8</td><td>0xD9FF</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 31</td></tr>
<tr><td>0xDA00</td><td>0xDA07</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 32</td></tr>
<tr><td>0xDA08</td><td>0xDA0F</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 33</td></tr>
<tr><td>0xDA10</td><td>0xDA17</td><td>8</td><td>I/O</td><td>W</td><td>Sprite 34</td></tr>
</table>

0xDA18	0xDA1F	8	I/O	W	Sprite 35
0xDA20	0xDA27	8	I/O	W	Sprite 36
0xDA28	0xDA2F	8	I/O	W	Sprite 37
0xDA30	0xDA37	8	I/O	W	Sprite 38
0xDA38	0xDA3F	8	I/O	W	Sprite 39
0xDA40	0xDA47	8	I/O	W	Sprite 40
0xDA48	0xDA4F	8	I/O	W	Sprite 41
0xDA50	0xDA57	8	I/O	W	Sprite 42
0xDA58	0xDA5F	8	I/O	W	Sprite 43
0xDA60	0xDA67	8	I/O	W	Sprite 44
0xDA68	0xDA6F	8	I/O	W	Sprite 45
0xDA70	0xDA77	8	I/O	W	Sprite 46
0xDA78	0xDA7F	8	I/O	W	Sprite 47
0xDA80	0xDA87	8	I/O	W	Sprite 48
0xDA88	0xDA8F	8	I/O	W	Sprite 49
0xDA90	0xDA97	8	I/O	W	Sprite 50
0xDA98	0xDA9F	8	I/O	W	Sprite 51
0xDAA0	0xDAA7	8	I/O	W	Sprite 52
0xDAA8	0xDAAF	8	I/O	W	Sprite 53
0xDAB0	0xDAB7	8	I/O	W	Sprite 54
0xDAB8	0xDABF	8	I/O	W	Sprite 55
0xDAC0	0xDAC7	8	I/O	W	Sprite 56
0xDAC8	0xDACF	8	I/O	W	Sprite 57
0xDAD0	0xDAD7	8	I/O	W	Sprite 58
0xDAD8	0xDADF	8	I/O	W	Sprite 59
0xDAE0	0xDAE7	8	I/O	W	Sprite 60
0xDAE8	0xDAEF	8	I/O	W	Sprite 61
0xDAF0	0xDAF7	8	I/O	W	Sprite 62
0xDAF8	0xDAFF	8	I/O	W	Sprite 63 – Least Priority

11.18 VIA - $DC00 - $DCFF

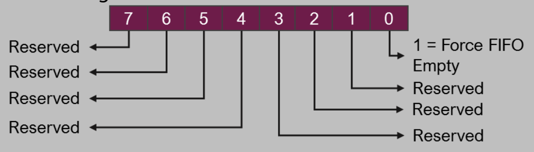
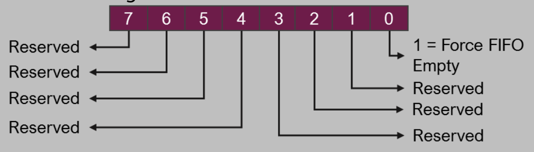
Addy Start	Addy End	Size	Type	R/W	Description
					W65C22 – Versatile Interface Adapter
0xDC00		8	I/O	R/W	(IRA) Input Register B (R) (ORA) Output Register B (W)
0xDC01		8	I/O	R/W	(IRA) Input Register A (R) (ORA) Output Register A (W)
0xDC02		8	I/O	R/W	(DDRB) Data Direction Register B
0xDC03		8	I/O	R/W	(DDRA) Data Direction Register A
0xDC04		8	I/O	R/W	(T1C-L) T1 Low-Order Latches / T1 Low-Order Counter
0xDC05		8	I/O	R/W	(T1C-H) T1 High-Order Counter
0xDC06		8	I/O	R/W	(T1L-L) T1 Low-Order Latches
0xDC07		8	I/O	R/W	(T1L-H) T1 High-Order Latches
0xDC08		8	I/O	R/W	(T2C-L) T2 Low-Order Latches / T2 Low-Order Counter
0xDC09		8	I/O	R/W	(T2C-H) T2 High-Order Counter
0xDC0A		8	I/O	R/W	(SR) Shift Register
0xDC0B		8	I/O	R/W	(ACR) Auxiliary Control Register
0xDC0C		8	I/O	R/W	(PCR) Peripheral Control Register
0xDC0D		8	I/O	R/W	(IFR) Interrupt Flag Register
0xDC0E		8	I/O	R/W	(IER) Interrupt Enable Register
0xDC0F		8	I/O	R/W	(ORA/IRA) Same as Reg1 except no “Handshake”

11.19 SD CARD - $DD00 - $DDFF

<table>
<tr><th>Addy Start</th><th>Addy End</th><th>Size</th><th>Type</th><th>R/W</th><th>Description</th></tr>
<tr><td>0xDD00</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>Version Reg
7 6 5 4 3 2 1 0
Major Revision Number [3] — Minor Revision Number [0]
Major Revision Number [2] — Minor Revision Number [1]
Major Revision Number [1] — Minor Revision Number [2]
Major Revision Number [0] — Minor Revision Number [3]</td></tr>
<tr><td>0xDD01</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>Master Control Register
7 6 5 4 3 2 1 0
Reserved — Reset Controller (1 = Reset)
Reserved — Reserved
Reserved — Reserved
Reserved — Reserved</td></tr>
<tr><td>0xDD02</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>Transfer Type
7 6 5 4 3 2 1 0
Reserved — Transfer Type[0]
Reserved — Transfer Type[1]
Reserved — Reserved
Reserved — Reserved
<table>
<tr><th>Transfer Type</th><th>[1]</th><th>[0]</th></tr>
<tr><td>Direct Access</td><td>0</td><td>0</td></tr>
<tr><td>Init SD</td><td>0</td><td>1</td></tr>
<tr><td>R/W Read SD Block</td><td>1</td><td>0</td></tr>
<tr><td>R/W Write SD Block</td><td>1</td><td>1</td></tr>
</table></td></tr>
<tr><td>0xDD03</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>Transfer Control Register
7 6 5 4 3 2 1 0
Reserved — Start Transaction (Self-Clear)
Reserved — Reserved
Reserved — Reserved
Reserved — Reserved</td></tr>
<tr><td>0xDD04</td><td></td><td>8</td><td>I/O</td><td>R</td><td>Transfer Status Register
7 6 5 4 3 2 1 0
Reserved — 1 = Busy
Reserved — Reserved
Reserved — Reserved
Reserved — Reserved</td></tr>
<tr><td>0xDD05</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>Transfer Error Register
7 6 5 4 3 2 1 0
Reserved — SD_INIT_ERROR[0]
Reserved — SD_INIT_ERROR[1]
SD_WRITE_ERROR[1] — SD_READ_ERROR[0]
SD_WRITE_ERROR[0] — SD_READ_ERROR[1]
<table>
<tr><th>Init Error</th><th>[1]</th><th>[0]</th><th>Read Error</th><th>[1]</th><th>[0]</th><th>Write Error</th><th>[1]</th><th>[0]</th></tr>
<tr><td>No Error</td><td>0</td><td>0</td><td>No Error</td><td>0</td><td>0</td><td>No Error</td><td>0</td><td>0</td></tr>
<tr><td>Init CMD0</td><td>0</td><td>1</td><td>Read CMD</td><td>0</td><td>1</td><td>Write CMD</td><td>0</td><td>1</td></tr>
<tr><td>Init CMD1</td><td>1</td><td>0</td><td>Read Token</td><td>1</td><td>0</td><td>Write Data</td><td>1</td><td>0</td></tr>
<tr><td>Reserved</td><td>1</td><td>1</td><td>Reserved</td><td>1</td><td>1</td><td>Write Busy</td><td>X</td><td>X</td></tr>
</table></td></tr>
<tr><td>0xDD06</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>Direct Access Data Register
TX_Data[7:0] (W)
Set TX_DATA prior to starting a DIRECT_ACCESS transaction.

Note that the SPI bus has no concept of a read or write transaction. Thus every DIRECT_ACCESS transaction transmits data from the SPI master, and receives data from the SPI slave.

RX_Data[7:0] (R)
Read RX_DATA after completing a DIRECT_ACCESS transaction.</td></tr>
<tr><td>0xDD07</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>SD Address Register [7:0]
Normally set to zero, because memory accesses should occur on a 512 bytes boundary. Set the SD/MMC memory address before starting a block read or block write.</td></tr>
<tr><td>0xDD08</td><td></td><td>8</td><td>I/O</td><td>R/W</td><td>SD Address Register [15:8]
Normally set SD_ADDR[8] to zero, because memory accesses should occur on a 512 bytes boundary.</td></tr>
</table>

0xDD09		8	I/O	R/W	SD Address Register [23:16]
0xDD0A		8	I/O	R/W	SD Address Register [31:24]
0xDD0B		8	I/O	R/W	SPI Clock Del Register SPI_CLK_DEL controls the frequency of the SPI_CLK after SD initialization is completed. SPI_CLK_DEL = (spiSysClk / (SPI_CLK * 2)) – 1
0xDD10		8	I/O	R	Reception FIFO Data Register SD/MMC block read data. Note, FIFO size matches the SD/MMC block size of 512 bytes.
0xDD12		8	I/O	R/W	Reception FIFO Data Count Register [15:8] MSB of FIFO_DATA_COUNT. Indicates the number of data entries within the FIFO.
0xDD13		8	I/O	R/W	Reception FIFO Data Count Register [7:0] LSB of FIFO_DATA_COUNT. Indicates the number of data entries within the FIFO.
0xDD14		8	I/O	R/W	Reception FIFO Control Register  7 6 5 4 3 2 1 0 Reserved Reserved Reserved Reserved 1 = Force FIFO Empty Reserved Reserved Reserved Deletes all the data samples within the FIFO. Self clearing.
0xDD20		8	I/O	R/W	Transmission FIFO Data Register SD/MMC block write data. FIFO size matches the SD/MMC block size of 512 bytes.
0xDD24		8	I/O	R/W	Transmission FIFO Control Register  7 6 5 4 3 2 1 0 Reserved Reserved Reserved Reserved 1 = Force FIFO Empty Reserved Reserved Reserved Deletes all the data samples within the FIFO. Self clearing.

11.20 MATH - $DE00 - $DEFF

Addy Start	Addy End	Size	Type	R/W	Description
					WRITE
					SIGNED MULTPLICATION
0xDE00		8	I/O	W	Unsigned Mult Operand A [7:0]
0xDE01		8	I/O	W	Unsigned Mult Operand A [15:8]
0xDE02		8	I/O	W	Unsigned Mult Operand B [7:0]
0xDE03		8	I/O	W	Unsigned Mult Operand B [15:8]
					SIGNED MULTPLICATION
0xDE04		8	I/O	W	Signed Mult Operand A [7:0]
0xDE05		8	I/O	W	Signed Mult Operand A [15:8]
0xDE06		8	I/O	W	Signed Mult Operand B [7:0]
0xDE07		8	I/O	W	Signed Mult Operand B [15:8]
					UNSIGNED DIVISION
0xDE08		8	I/O	W	Unsigned Div Denominator [7:0]
0xDE09		8	I/O	W	Unsigned Div Denominator [15:8]
0xDE0A		8	I/O	W	Unsigned Div Numerator [7:0]
0xDE0B		8	I/O	W	Unsigned Div Numerator [15:8]
					SIGNED DIVISION
0xDE0C		8	I/O	W	Signed Div Denominator [7:0]
0xDE0D		8	I/O	W	Signed Div Denominator [15:8]
0xDE0E		8	I/O	W	Signed Div Numerator [7:0]
0xDE0F		8	I/O	W	Signed Div Numerator [15:8]
					READ
					UNSIGNED MULTPLICATION
0xDE00		8	I/O	R	Operand A [7:0]
0xDE01		8	I/O	R	Operand A [15:8]
0xDE02		8	I/O	R	Operand B [7:0]
0xDE03		8	I/O	R	Operand B [15:8]
0xDE04		8	I/O	R	Results [7:0]
0xDE05		8	I/O	R	Results [15:8]
0xDE06		8	I/O	R	Results [23:16]
0xDE07		8	I/O	R	Results [31:24]
					SIGNED MULTPLICATION
0xDE08		8	I/O	R	Operand A [7:0]
0xDE09		8	I/O	R	Operand A [15:8]
0xDE0A		8	I/O	R	Operand B [7:0]
0xDE0B		8	I/O	R	Operand B [15:8]
0xDE0C		8	I/O	R	Results [7:0]
0xDE0D		8	I/O	R	Results [15:8]
0xDE0E		8	I/O	R	Results [23:16]
0xDE0F		8	I/O	R	Results [31:24]
					UNSIGNED DIVISION
0xDE10		8	I/O	R	Denominator [7:0]
0xDE11		8	I/O	R	Denominator [15:8]
0xDE12		8	I/O	R	Numerator [7:0]
0xDE13		8	I/O	R	Numerator [15:8]
0xDE14		8	I/O	R	Quotient [7:0]
0xDE15		8	I/O	R	Quotient [15:8]
0xDE16		8	I/O	R	Remainder [7:0]
0xDE17		8	I/O	R	Remainder [15:8]
					SIGNED DIVISION
0xDE18		8	I/O	R	Denominator [7:0]
0xDE19		8	I/O	R	Denominator [15:8]
0xDE1A		8	I/O	R	Numerator [7:0]
0xDE1B		8	I/O	R	Numerator [15:8]
0xDE1C		8	I/O	R	Quotient [7:0]
0xDE1D		8	I/O	R	Quotient [15:8]

0xDE1E		8	I/O	R	Remainder [7:0]
0xDE1F		8	I/O	R	Remainder [15:8]

11.21 DMA CONTROLLER - $DF00 - $DFFF

To be documented Soon