VT1682 程序设计指南 V1.10

2. 修订历史

Revision	Date	Remark
V1.0	2005/08/	初始版
V1.1	2005/12/22	UART 状态端口由$2119 移动到$211B. 删除 CCIR 灰阶拍摄和蓝屏效果. 更改 8.2 ~ 8.6 IO 设置表格. 删除休眠/苏醒部份. 删除 Eye-Function IRQ. 删除 MCU 界面部份. 更改读取$2007 给卡通块 RAM 和$2004 给 VRAM 更改 $2110~$2113 读取端口. 删除卡通块纵向放大功能.
V1.2	2006/03/27	Page23:更改 $210B D7 TSYNEN 状态
V1.3	2006/05/02	Page12:修正在 EXT2421 地址模式的错误. Page23:修正定时器公式的错误. Page29,30:修正 BK1/BK2 VRAM 管理. Page22 and 68: 修正 ALU 除法器的余数.
V1.4	2006/06/01	Page18:修正 210D.D5
V1.5	2006/06/26	Page45:修正光枪的程序顺序
V1.6	2006/12/26	修正打字错误 Page25 CSPU->SCPU,Page73:0X4121->0X212E
V1.7	2006/04/10	修正打字错误 page68 14.8.2 Divider ALU_Muti_operand-→ALU_Div_operand
V1.8	2007/10/1	Page 13: 增加 CPU Vector → System Reset 和 0x7FFFC, 0x7FFFD Page 53: 修改列 XIOE[0,1,2]的 IOEOE=0 字段, INPUT floating → INPUT (12K ohms pull-down resistor) Page 56: 修改 XIOE0, 1, 2 的 OUTPUT 字段, --- → VR, VG, VB Page 57: 修正 0x2127® → 0x2127(R)
V1.9	2007/10/30	Page 62: 修改"CS_Control"的范围
V1.10	2009/01/19	Page 46: 取消 0x2004 和 0x2007 的读取功能

3. 一般功能描述

VT1682 包括 主 CPU, 图像处理器, 声音的 CPU, 内部的 SRAM(8K 字节给程序用,4K 字节给图像用),内部的 ROM(4K 字节)及 一些 I/O 控制装置. VT1682 可以分为两个系统,一个用于程序的,另一个用于影像的处理.

主 CPU(Main CPU)是整个程序系统的主要角色. 它可以对内部的随机存储器 PRAM 和外部的系统软件存储器(ROM 或是 Flash)进行寻址以取出需要的讯息进行运算处理. 系统软件的存储器(ROM 或是 Flash)被储存程序命令, 程序指引和一些声音数据.而 VT1682 内部的 8K 字节的程序动态随机存储器(PRAM)是第零页 RAM, 储存空间 及一些 CPU 的内存. 程序系统控制学习机的执行, 包括图案, 语音, 及字幕. 也就是说 CPU 将控制视频系统显示指定的图案.

图像单元是影像系统的主要角色. 它能够对内部的动态随机存储器(VRAM)和图形块(character)存储器(ROM 或是 Flash) 进行寻址以取出需要的讯息进行运算处理后自动地显示一些图案. 除了内部的 PRAM 之外,VT1682 内部有另外的 4K 字节的 VRAM, 影像的动态随机存储器(VRAM)存储许多指到图形块(Character)ROM 或是 Flash 内图形的图形序号. VRAM 储存图形序号,它可以在屏幕上作 2 层背景的显示. 图形块(Character) ROM 储存许多卡通块图形.

声音 CPU(Sound CPU)和主 C PU 共享内部的 ROM 和 4K 字节的程序 SRAM.它有各自的 IO 和 ALU.它的操作速度比主 CPU 快 4 倍而且它适合于不同的应用.

3.1 特点(Feature)

系统(System)

- 工作电压: 3.0~3.6 V
- 主 CPU: 6502 @5.3693MHz in NTSC and 5.3203MHz in PAL
- 内部的可随意调整程序内存(ROM): 4K 字节
- 内部的主 CPU 的程序 RAM: 8K 字节 (4K 字节为专用的 RAM 和 4K 字节为共享的 RAM)
- 内部的 Video RAM: 4K 字节
- 内存直接寻址(DMA) Sprite RAM / VRAM / Program RAM / 外部的存储器
- 单一的 16 位数据总线
- 扫瞄线 IRQ / 16-bits 定时器 IRQ / 外部的 IRQ
- 透过三条地址线(CSB)译码可扩充外部的存储器到 32M 字节.
- T.V. 信号输出(NTSC, PAL, PAL-M, PAL-N)
- 扩充 5 IRQ 伺服入口
- 56 个 GPIO 端口, 40 个给主 CPU, 其他 16 个给 Sound CPU.

周边(Peripheral)

- ADC: 8bits, 5 个时间分割复合(Times-Division-Multiplex)通道和声音增大(Voice Gain)控制
- 4 阶段低电压检查
- 主动装置(Master)/从动装置(Slave) SPI 界面
- UART 界面
- TFT LCD 界面.
- STN LCD 界面
- IIS 界面
- IIC 界面 (主动装置(Master)方式)
- CCIR656/601 界面
- 加强 ALU, 16 乘 16 乘法器和 32 除 16 除法器

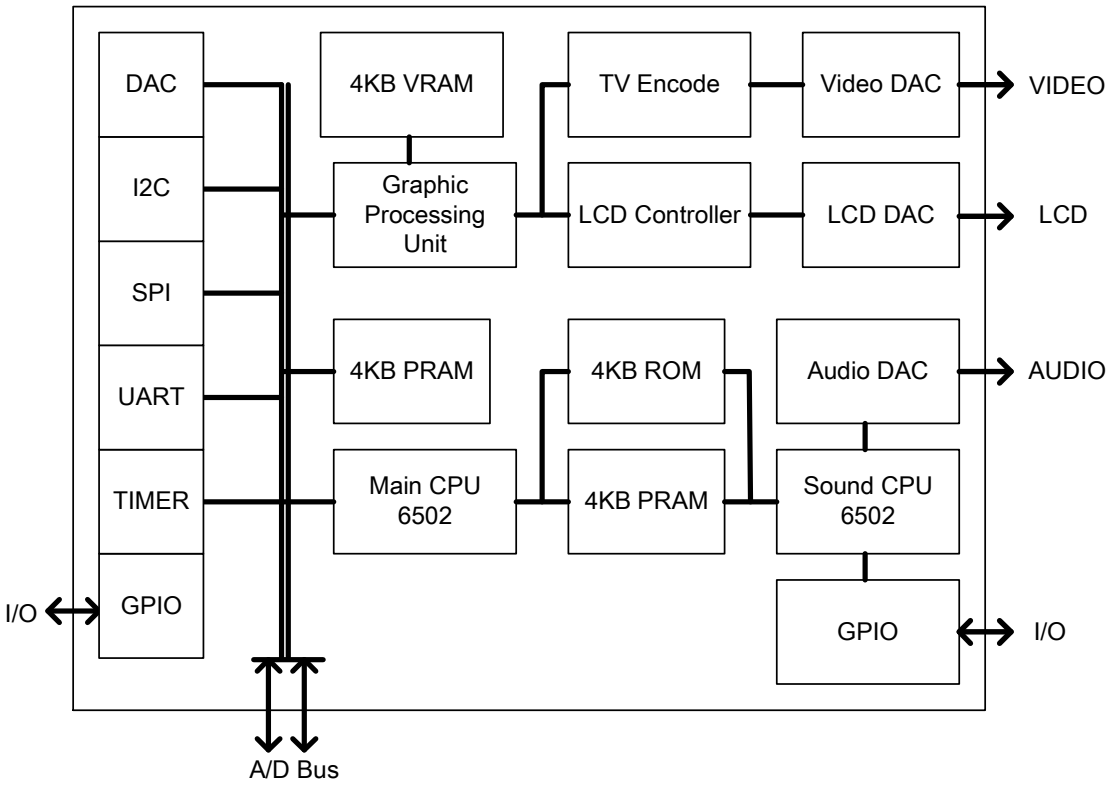
Graphic 处理器

- 分辨率: TV 256x240 点
- 在同一个画面(Frame)有 240 个卡通块,在横向最大的卡通块数是 16 个
- 2 个独立的背景层.
- 背景字符方式: 16/64/256 索引颜色方式.
- 背景数字映像的方式: 16/64/256 索引颜色方式.或是 32768 色直接颜色方式
- 卡通块(Sprites)为 16 色.
- 两个 256 上色颜色调色板, 最大显示的索引颜色: 512
- 背景纵向的宽度: x1/x1.5/x2
- 背景横向的线单独卷动: -128~+127

Sound CPU

- CPU 6502 @21.4772MHz in NTSC and 26.6017MHz in PAL
- 4K 字节共享 RAM
- 4K 字节可随意调整的内部的 ROM
- 16 GPIO 端口
- 16 bits Timer x2
- ALU, 16 乘 16 乘法器和 32 除 16 除法器

3.2 结构图(BLOCK DIAGRAM)



3.3 IC 脚位的描述

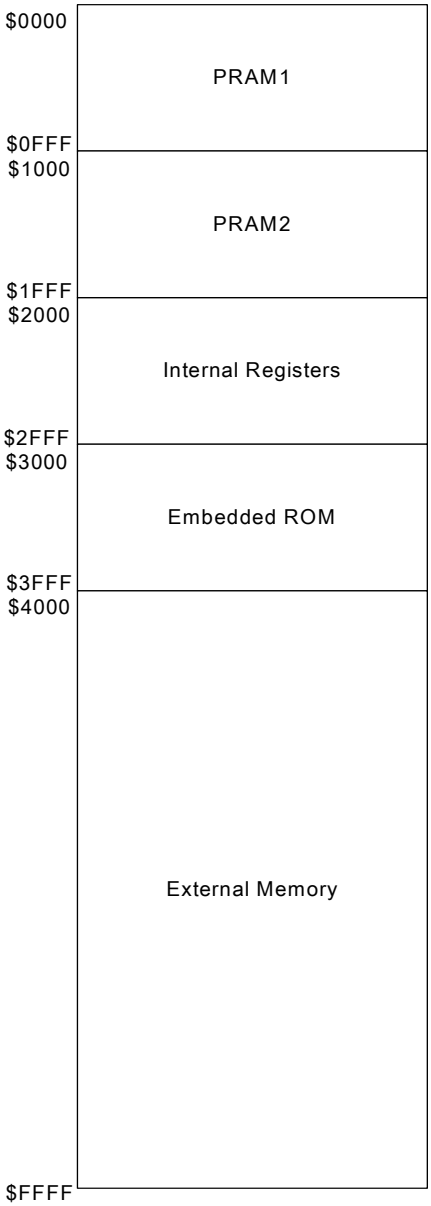
符　　号	型 式	描　　述
XA[23:0]	O	地址总线
XD[15:0]	I/O	数据总线
XROMCSB	O	1st 外部存储器蕊片片选讯号
XRAMCSB	O	2nd 外部存储器蕊片片选讯号
XROMOEB	O	外部存储器 OEB 讯号
XRAMRWB	O	外部存储器 RWB 讯号
XDEBUGNMI	I	除错方式的 NMI
XDEBUGCSB	O	除错方式存储器蕊片片选讯号
XIOA[3:0]	I/O	一般的 I/O
XIOB[3:0]	I/O	一般的 I/O
XIOC[3:0]	I/O	一般的 I/O
XIOD[3:0]	I/O	一般的 I/O
XIOE[3:0]	I/O	一般的 I/O
XIOF[3:0]	I/O	一般的 I/O
XUIOA[7:0]	I/O	一般的 I/O
XUIOB[7:0]	I/O	一般的 I/O
XSCPUIOA[7:0]	I/O	一般的 I/O
XSCPUIOB[7:0]	I/O	一般的 I/O
XTAL1	I	晶振脚位
XTAL2	O	晶振脚位
XBOOTINIT	I	内部的 ROM 启动方式
XPLLVCO	I/O	PLL 参考电压
XPLLVREF	I/O	PLL 参考电压
XVIDEO	O	混合视频讯号
XAUDIOR	O	音频讯号的右声道
XAUDIOL	O	音频讯号的左声道

4. 主 CPU

在 VT1682 的主 CPU 是 8-bits 的 6502,操作频率为 5MHz.

4.1 存储器映射(Memory Map)

存储器的映像如底下的图形所示. PRAM1 (Program RAM-1)是给主 CPU 部份的 Program RAM 为 4K 字节. PRAM2 是音频 CPU 和主 CPU 共享的 RAM 也是 4K 字节. 在 0x2000 和 0x20FF 之间的地址是 Graphic 端口,而其他的部份是由系统或是周边来控制. 内部有一个 4K 字节的 ROM(Embedded ROM) 给主 CPU 或是 Sound CPU 用. 它可以充当 BIOS, 加密, 程序的 ROM 或是数据的 ROM. $4000 ~ $FFFF 是映像到 6 个程序储存空间去扩充地址线到 8M 字节的外部的存储器.



4.2 地址方式(Address Mode)

这个 16 位 6502 CPU 地址线被扩充到 25 位的物理地址线是根据以下的编辑程序样式. 在下面的表格中, 6502 的地址线是 A[15:0] 和物理地址线是 {PA[24:13], A[12:0]}.

PQ2EN	COMR6	A15	A14	A13	TP20	TP19	TP18	TP17	TP16	TP15	TP14	TP13
0	0	1	0	0	Program_Bank0_Register0							
0	0	1	0	1	Program_Bank0_Register1							
0	0	1	1	0	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	1	1	1	0
0	1	1	0	1	Program_Bank0_Register1							
0	1	1	1	0	Program_Bank0_Register0							
0	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	0	0	Program_Bank0_Register0							
1	0	1	0	1	Program_Bank0_Register1							
1	0	1	1	0	Program_Bank0_Register2							
1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	Program_Bank0_Register2							
1	1	1	0	1	Program_Bank0_Register1							
1	1	1	1	0	Program_Bank0_Register0							
1	1	1	1	1	1	1	1	1	1	1	1	1
--	--	0	1	1	Program_Bank0_Register5							
--	--	0	1	0	Program_Bank0_Register4							

*TP[20:13]将会被转换在下面的表格

Program_Bank0_select[2:0]			PA20	PA19	PA18	PA17	PA16	PA15	PA14	PA13
0	0	0	PQ37	PQ36	TP18	TP17	TP16	TP15	TP14	TP13
0	0	1	PQ37	PQ36	PA35	TP17	TP16	TP15	TP14	TP13
0	1	0	PQ37	PQ36	PQ35	PQ34	TP16	TP15	TP14	TP13
0	1	1	PQ37	PQ36	PQ35	PQ34	PA33	TP15	TP14	TP13
1	0	0	PQ37	PQ36	PQ35	PQ34	PQ33	PQ32	TP14	TP13
1	0	1	PQ37	PQ36	PQ35	PQ34	PQ33	PQ32	PA31	TP13
1	1	0	PQ37	PQ36	PQ35	PQ34	PQ33	PQ32	PQ31	PQ30
1	1	1	TP20	TP19	TP18	TP17	TP16	TP15	TP14	TP13

* ProgramBank0_Register3 = {PQ37, PQ36, PQ35, PQ34, PQ33, PQ32, PQ31, PQ30,}.

EXT2421	PQ2EN	COMR6	A15	A14	A13	PA24	PA23	PA22	PA21
1	x	x	1	x	x	Program_Bank1_Register3			
0	0	0	1	0	0	Program_Bank1_Register0			
0	0	0	1	0	1	Program_Bank1_Register1			
0	0	0	1	1	0	Program_Bank1_Register3			
0	0	0	1	1	1	Program_Bank1_Register3			
0	0	1	1	0	0	Program_Bank1_Register3			
0	0	1	1	0	1	Program_Bank1_Register1			
0	0	1	1	1	0	Program_Bank1_Register0			
0	0	1	1	1	1	Program_Bank1_Register3			
0	1	0	1	0	0	Program_Bank1_Register0			
0	1	0	1	0	1	Program_Bank1_Register1			

0	1	0	1	1	0	Program_Bank1_Register2
0	1	0	1	1	1	Program_Bank1_Register3
0	1	1	1	0	0	Program_Bank1_Register2
0	1	1	1	0	1	Program_Bank1_Register1
0	1	1	1	1	0	Program_Bank1_Register0
0	1	1	1	1	1	Program_Bank1_Register3
X	X	X	0	1	1	Program_Bank1_Register5
x	x	X	0	1	0	Program_Bank1_Register4

参考寄存器(Reference Registers)

	D7	D6	D5	D4	D3	D2	D1	D0
0x2100(W)					Program_Bank1_Register3			
0x2100(R)					Program_Bank1_Register3			
0x2105(W)	---	COMR6						
0x2107(W)	Program_Bank0_Register0							
0x2107(R)	Program_Bank0_Register0							
0x2108(W)	Program_Bank0_Register1							
0x2108(R)	Program_Bank0_Register1							
0x2109(W)	Program_Bank0_Register2							
0x2109(R)	Program_Bank0_Register2							
0x210A(W)	Program_Bank0_Register3							
0x210A(R)	Program_Bank0_Register3							
0x210B		PQ2EN				Program_Bank0_select		
0x210C(W)					Program_Bank1_Register2			
0x210C(R)					Program_Bank1_Register2			
0x2110(W)					Program_Bank1_Register0			
0x2112(R)					Program_Bank1_Register0			
0x2111(W)					Program_Bank1_Register1			
0x2113(R)					Program_Bank1_Register1			
0x2112(W)	Program_Bank0_Register4							
0x2110(R)	Program_Bank0_Register4							
0x2113(W)	Program_Bank0_Register5							
0x2111(R)	Program_Bank0_Register5							
0x2118(W)	Program_Bank1_Register5				Program_Bank1_Register4			
0x2118(R)	Program_Bank1_Register5				Program_Bank1_Register4			
0x211C(W)			EXT2421EN					

4.3 CPU 序号(CPU Vectors)

在主 CPU 的序号包括 NMI, RESET 和 5 IRQ,如以下的表格.

Vector Name	vector address
NMI	0x7FFFA, 0x7FFFB
System Reset	0x7FFFC, 0x7FFFD
Ext_IRQ	0x7FFFE, 0x7FFFF
Timer_IRQ	0x7FFF8, 0x7FFF9
SCPU_IRQ	0x7FFF6, 0x7FFF7
UART_IRQ	0x7FFF4, 0x7FFF5
SPI_IRQ	0x7FFF2, 0x7FFF3

5. 界面

VT1682 主 CPU 控制的界面包括电视复合信号的输出, LCD, UART, SPI, IIC 和 CCIR 界面.

5.1 电视复合信号输出

VT1682 可以支持 4 种电视系统和准许调整电视信号的饱和度(Saturation)和光亮度(Luminance). 在 0x2106 写 "1" 到 TV_ON 和在 0x211D 写 "1" 到 VDAC_EN 去使这个复合视频输出功能致能.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2106	---	---	SCPURN	SCPU_ON	SPI_ON	UART_ON	TV_ON	LCD_ON
0x211D	LVDEN			VDAC_EN	ADAC_EN	PLL_EN	LCDACEN	---

5.1.1 电视系统的表面配置

NTSC, PAL, PAL-M and PAL-N 电视系统是利用于 VT1682 内设置 0x2105 端口和在 IC 脚位 XTAL1 和 XTAL2 更换相关的晶振(crystal). 它们的相关性如下面的表格所列.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2105	---	COMR6	TV_SYS_SE:[1:0]		CCIR_SEL	Dual_Speed	ROM_SEL	PRAM

TV_SYS_SEL[1:0]	TV system	Crystal Frequency
0	NTSC	21.4772MHz
1	PAL_M	21.453669MHz
2	PAL_N	21.492336MHz
3	PAL	26.601712MHz

5.1.2 电视的参数

在 VT1682 里面有 32 个明亮度(Brightness)和 8 个对比(Contrast)阶段可以经由设置 0x2021 和 0x2022 端口由程序来控制. 好好配合 0x2021 的设置可以让电视的画面产生渐现(渐隐)的效果.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2021	----	----	Luminance_offset					
0x2022	----	----	VCOMIO	RGB_DAC	CCIR_OUT	Saturation		

光亮度补偿值(Luminance_offset) : 电视输出光亮度调整控制, 代表 2's 非.

2's 非: -A = ~A + 1,

Ex: -5 = ~(000101) + 1 = 111010 + 1 = 111011 = 59

有效的值 -32--------- -16 ------ -1 –0 – 1-------- 15 ------- 31

最暗----------暗--------正 常--------明亮-----最亮

Data to 0x2021 32----------48---------63--0--1--------15---------31

饱和度(Saturation) = 0 : 灰阶输出, 默认值为 4.

在 0x2022 的值 1-----2-----3-----4------5------6------7

饱和度(Saturation) 高<----------正常----------→ 低

5.2 LCD 界面

VT1682 可以提供不同类型的 LCD 界面, 包括 AUO-051(数字), AUO-052(数字), 模拟 TFT LCD 和 CSTN 界面.在 0x200C 的 LCD_MODE 有 4 种主要的输出拟定可被选择.它们的映像信号如下面的表格所列.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2008	LCD_Y							
0x2009	LCD_X[7:0]							
0x200A	LCD_FR [7:0]							
0x200B							LCD_FR[8]	LCD_X[8]
0x200C	F_RATE		LCD_CLK		UPS052		LCD_MODE	
0x200D	LCDEN	Dot240	Reverse		Color_Sequence			
0x2022	----	----	VCOMIO	RGB_DAC	CCIR_OUT	Saturation		

LCD_Y : LCD 显示屏纵向的补偿,以横向的扫瞄线为基础.

备注: 0xFF 和 0 是不许用的.

LCD_X : LCD 显示屏横向的补偿,以 LCD dot clock 为基础.

备注: 0x1FF 和 0 是不许用的.

LCD_FR: STN LCD 交互的信号 toggle rate 以横向的扫瞄线为基础.

F_RATE: LCD 控制器参数, STN 框架速率(frame rate)控制.

0 : 60(N) / 50(P) FPS 1 : 120(N) / 100(P) FPS

LCD_CLK: LCD 控制器参数, LCD dot clock 选择

0 : 21.4772 / 4 MHz 1 : 21.4772 / 2 MHz

2 : 21.4772 MHz 3 : External clock*

UPS052 : LCD 控制器参数, AUO UPS052 TCON 方式选择

0 : 非 UPS052 方式 1 : UPS052 方式

LCDEN : LCD 致能控制

0 : 无作用. 1 : 致能.

Dot240 : LCD 依比率决定控制, 依比率决定比率到 15/16.

0 : No scaling 1 : Scaling

反转 : LCD 控制器参数,点阵数据输出反转

0 : 正常 1 : 反转

LCD_MODE : LCD 控制器参数, 控制器输出拟定选择

0 : ANALOG 方式 1 : CSTN 方式

2 : LTPS 方式 3 : DIGITAL 方式

VCOMIO : LCD 输入 VCOM 方式

0 : 无作用 1 : 致能

模拟方式(ANALOG Mode)

脚位名称	信号名称	设　　置	描　　述
XIOA0	DIO	IOA_ENB=1, IOA_OE=1	Vertical start pulse
XIOA1	XOE	IOA_ENB=1, IOA_OE=1	Output enable for scan driver
XIOA2	CPH1	IOA_ENB=1, IOA_OE=1	Sample and shift clock pulse for data driver
XIOA3	CPH3	IOA_ENB=1, IOA_OE=1	Sample and shift clock pulse for data driver
XIOB0	Q2H	IOB_ENB=1, IOB_OE=1	Analog rotate signal
XIOB1	VCOM	IOB_ENB=1, IOB_OE=1	Common electrode driving signal
XIOB2	CPV	IOB_ENB=1, IOB_OE=1	Shift clock for scan driver
XIOB3	INH	IOB_ENB=1, IOB_OE=1	Output enable for data driver
XIOC0	CPH2	IOC_ENB=1, IOC_OE=1	Sample and shift clock pulse for data driver
XIOC1	STH	IOC_ENB=1, IOC_OE=1	Start pulse of horizontal scan line
XIOE0	VR	IOE0OE = 0	Analog R
XIOE1	VG	IOE1OE = 0	Analog G
XIOE2	VB	IOE2OE = 0	Analog B

LTPS Mode

脚位名称	信号名称	设　　置	描　　述
XIOA0	DIO	IOA_ENB=1, IOA_OE=1	Frame start signal
XIOA1	XOE	IOA_ENB=1, IOA_OE=1	Inverse of frame start signal
XIOA2	CPH1	IOA_ENB=1, IOA_OE=1	Dot clock
XIOB0	Q2H	IOB_ENB=1, IOB_OE=1	Line clock
XIOB1	VCOM	IOB_ENB=1, IOB_OE=1	Common electrode driving signal
XIOB2	CPV	IOB_ENB=1, IOB_OE=1	Inverse of line clock
XIOB3	INH	IOB_ENB=1, IOB_OE=1	Inverse of line start signal
XIOC0	CPH2	IOC_ENB=1, IOC_OE=1	Inverse of dot clock
XIOC1	STH	IOC_ENB=1, IOC_OE=1	Line start signal
XIOE0	VR	IOE0OE = 0	Analog R
XIOE1	VG	IOE1OE = 0	Analog G
XIOE2	VB	IOE2OE = 0	Analog B

数字方式(Digital Mode)

脚位名称	信号名称	设　　置	描　　述
XIOA0	DIO	IOA_ENB=1, IOA_OE=1	Vertical SYNC
XIOA1	XOE	IOA_ENB=1, IOA_OE=1	Horizontal SYNC
XIOA2	CPH1	IOA_ENB=1, IOA_OE=1	Dot clock
XIOA3	CPH3	IOA_ENB=1, IOA_OE=1	DATA[0]
XIOB0	Q2H	IOB_ENB=1, IOB_OE=1	DATA[1]
XIOB1	VCOM	IOB_ENB=1, IOB_OE=1	DATA[3]
XIOB2	CPV	IOB_ENB=1, IOB_OE=1	DATA[2]
XIOB3	INH	IOB_ENB=1, IOB_OE=1	DATA[4]

CSTN 方式

脚位名称	信号名称	设　　置	描　　述
XIOA0	DIO	IOA_ENB=1, IOA_OE=1	Frame Start
XIOA1	XOE	IOA_ENB=1, IOA_OE=1	Line Data Load
XIOA2	CPH1	IOA_ENB=1, IOA_OE=1	DCLK
XIOA3	CPH3	IOA_ENB=1, IOA_OE=1	AC Signal
XIOB0	Q2H	IOB_ENB=1, IOB_OE=1	DATA[0]
XIOB1	VCOM	IOB_ENB=1, IOB_OE=1	DATA[2]
XIOB2	CPV	IOB_ENB=1, IOB_OE=1	DATA[1]
XIOB3	INH	IOB_ENB=1, IOB_OE=1	DATA[3]

5.3 UART 界面

在 VT1682 的 UART 控制器可以提供达到 11520 bps 接收/传递的 baud-rate, 可过程控制接收/传送的 IRQ, 同位检查(parity check). UART 传送 (TX) 和 UART 接收(RX) 界面是在 XIOC2 和 XIOC3.

5.3.1 UART 控制寄存器

在 0x2106 的 UART_ON 必须在任何一个寄存器设置到 UART 端口之前先被设置为致能. 当 TXIRQEn 或是 RXIRQEn, TXIRQ 和 RXIRQ 两者将会供给到 CPU IRQ3, UART_IRQ, with IRQ vectors 0x7FFF4, 0x7FFF5.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2106	---	---	SCPURN	SCPU_ON	SPI_ON	UART_ON	TV_ON	LCD_ON
0x2119	--	CarriEn	UARTEN	TXIRQEn	RXIRQEn	ParityEn	OddEven	9bitmode

UARTEN : UART 致能

0 : 无作用 1 : 致能

TXIRQEn : TX IRQ 致能.当 TXIRQEn=1 和 TX_Status=1 时, UART_IRQ 会出来.

0 :无作用 1 : 致能

RXIRQEn : RX IRQ 致能. 当 RXIRQEn=1 和 RX_Status=1 时,UART_IRQ 会出来.

0 : 无作用 1 : 致能

ParityEn : RX 同位检查致能,这个检查结果是于 UART 状态端口表示在 ParityErr.

0 : 无作用 1 : 致能

OddEven : 当 ParityEn 被设置, 单数(Odd) / 偶数(Even)同位必须被设置.

0 : 单数(Odd)同位 1 : 偶数(Even)同位

9bitmode : RX / TX 于 9 bits 方式下传送. 当 ParityEn 被设置, 这个 9th bit 会是同位 bit (odd / even). 不然的话, 这个 9th bit 会只是在 OddEven 的数据.

UART 状态 (只能读取(Read Only))

0x211B	--	--	RxError	TX_Status	RX_Status	ParityErr	--	--

RxError : RX 错误标志.

0 : 无错误 1 : Baudrate 不匹配或是超过停止位.

TX_Status : TX 的状态标志. 写 Tx_Data 会清除此状态标志.

0 : UART 正在传送 1 : UART 完成传送

RX_Status : RX 的状态标志. 读 Rx_Data 会清除此状态标志.

0 : 无数据在 Rx_Data 1 : 在 Rx_Data 数据是有效的

ParityErr : 同位检查错误标志. ParityEn 必须先被致能.

0 : 同位正确 1 : 同位错误

0x211A(W)	Tx_Data[7:0]
0x211A(R)	Rx_Data[7:0]

写 0x211A 会经由 TX 传送数据. 当 RX_Status=1 时,读 0x211A 会得到 RX 数据.

5.3.2 波特速率(Baud Rate) 设置

在 VT1682 的 UART 控制器可以提供达到 11520 bps 接收/传送的波特速率(baud-rate).此外, UART 界面也可以在 TX 输出提供载波参数给红外线(Infra-red)应用. 在 0x211B 端口设置 Carier_Frequency 来调整载波频率. 在 0x2119 设置 CarriEN 使载波功能致能.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2114	Baud_phase[7:0]							
0x2115	Baud_phase[15:8]							
0x211B	Carrier_frequency[7:0]							

For NTSC system,

Baud_phase[15:0] = 65536 * Baud_rate / 5.3693 x10^6

Carrier_Frequency[7:0] = 256 – (5.3693 x10^6 / $F_{carrier}$)

For PAL system,

Baud_phase[15:0] = 65536 * Baud_rate / 5.3203424 x10^6

Carrier_Frequency[7:0] = 256 – (5.3203 x10^6 / $F_{carrier}$)

哪里的波特速率(Baud_rate)是 115200, 57600, 38400, 19200, 9600, 4800, 2400 和 $F_{carrier}$ 是这个需要运送的频率.

5.3.3 UART 信号

UART 脚位, TX 和 RX 是共享 XIOC2 和 XIOC3.

	脚位名称	设 置	描 述
TX	XIOC2	IOC_ENB=1, IOC_OE=1, UART_ON	UART Transmit (OUTPUT)
RX	XIOC3	IOC_ENB=1, IOC_OE=1, UART_ON	UART Receive (INPUT)

程序设计批注:

UART_TX 会在 XIOC2 和 UART_RX 在 XOIC3. 以下的寄存器必须在 UART 寄存器之前被设置.

```
$2106.D2 = 1;          // UART 模块致能
$210D.D4 = 1;          // IOC 输出致能
$210D.D5 = 1;          // IOC 输出
```

5.4 SPI 界面

SPI (标准的周边界面)可以操作在主动装置(master)和从动装置(slave) 方式. 它可以操作在 4 种不同的速度,由 0x2116 的 CK_FREQ 控制. 有 4 个型式的通信协议,如下表所描述.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2116	16bitMode	SPIEN	SPI_RST	M/SB	CKPHASE	CKPOLAR	CK_FREQ[1:0]	
0x2117(W)	SPI_TX_Data							
0x2117(R)	SPI_RX_Data							

CK_FREQ : SPI clock 操作频率.

CK_FREQ	SPI SCK Frequency
0	2.5MHz
1	1.25MHz
2	639KHz
3	320KHz

CKPOLAR : SPI clock 极性控制.

CKPHASE : SPI clock 样本相位控制.

M/SB : SPI 主动装置(Master) / 从动装置(slave) 方式选择

SPI_RST : SPI 模块复位信号

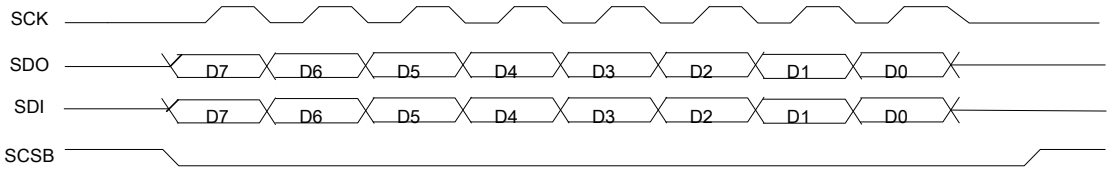
0 : 正常操作　　1 : SPI 复位

SPIEN : SPI 模块致能

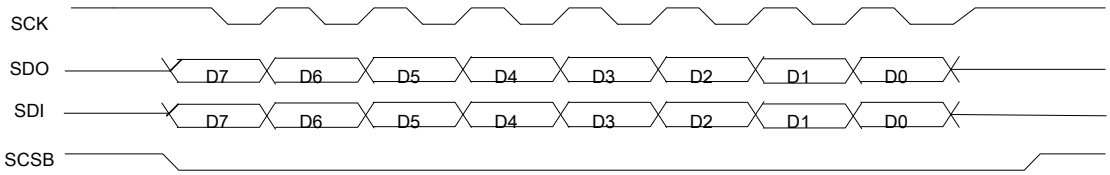
0 : 无作用　　1 : 致能

16bitMode : 8 / 16-bits SPI 方式选择

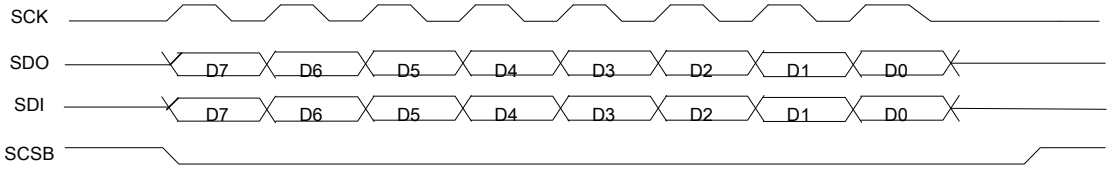
0 : 8-bits 方式　　1 : 16-bits 方式



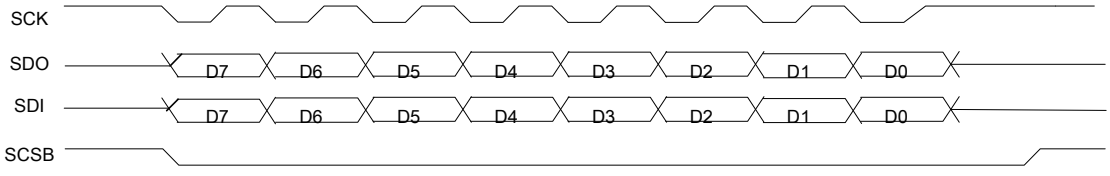
CKPOLAR=0, CKPHASE =0



CKPOLAR=1, CKPHASE =0



CKPOLAR=0, CKPHASE =1



CKPOLAR=1, CKPHASE =1

PIN NAME	Master Mode IOD_ENB=1, IOD_OE=1SPI_ON = 1	Slave Mode IOD_ENB=1, IOD_OE=0SPI_ON = 1	描 述
XIOD0	OUTPUT:SCK	INPUT:SCK	SPI CLOCK
XIOD1	INPUT:SDI	INPUT:SDI	SPI DATA INPUT
XIOD2	OUTPUT:SDO	OUTPUT:SDO	SPI DATA OUTPUT
XIOD3	OUTPUT:SCSB	INPUT:SCSB	SPI CSB

5.5 I2C

在 VT1682 主动装置(Master)方式的 I2C 功能是有效的. 每一个 I2C 命令包括 IIC_ID, IIC_ADDR 和 IIC_Data 如下面的图所示.当 IIC_ID 的低位(LSB)是偶数时,在 0x2142 的数据会输出在第三个相位.不然的话 IIC_DATA 会是在 0x2142 端口读到的流动输入数据.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2140	IIC_ID							
0x2141	IIC_ADDR							
0x2142(W)	IIC_DATA							
0x2142(R)	IIC_DATA							
0x2143							IIC_CLK_SEL	

IIC_CLK_SEL : I2C SCK 频率选择

0 : 1.34MHz 1 : 670KHz

2 : 335KHz 3 : 禁止使用

5.6 CCIR 通信协议(protocol)

VT1682 提供两组 CCIR 的界面,CCIR656 和 CCIR 601 界面, SET0 和 SET1 给影像输入.这个 CCIR 输入必须是 60 FPS for NTSC 或是 50 FPS for PAL system.

5.6.1 CCIR 输入脚位

这两组的 CCIR 输入是在 XUIOA, XUIOB, XSCPUIOA 和 XSCPUIOB ,如下面的表格所示.当这个 CCIR 界面被使用时,这些相关的 IO 脚必须被设置到输入方式. CCIR SET0 或是 SET1 是由 0x2105 的 CCIR_SEL 来选择.

SIGNAL NAME	SET0	SET1	Description
CCIR_D0	XUIOA0	XSCPUIOA0	DATA[0]
CCIR_D1	XUIOA1	XSCPUIOA1	DATA[1]
CCIR_D2	XUIOA2	XSCPUIOA2	DATA[2]
CCIR_D3	XUIOA3	XSCPUIOA3	DATA[3]
CCIR_D4	XUIOA4	XSCPUIOA4	DATA[4]
CCIR_D5	XUIOA5	XSCPUIOA5	DATA[5]
CCIR_D6	XUIOA6	XSCPUIOA6	DATA[6]
CCIR_D7	XUIOA7	XSCPUIOA7	DATA[7]

CCIR_CK	XUIOB0	XSCPUIOB0	CCIR CLOCK (27MHz)
CCIR_HS	XUIOB1	XSCPUIOB1	CCIR601 HSYNC
CCIR_VS	XUIOB2	XSCPUIOB2	CCIR 601 VSYNC

5.6.2 CCIR 控制寄存器

下面的寄存器是被使用来控制 CCIR 的界面.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2000				Capture	SLAVE	---	---	NMI_EN
0x2028	----	----	CCIR_Y					
0x2029	----	----	----	CCIR_X				
0x202A	VS_Phase	HS_Phase	YC_Swap	CbCrswap	SYNCMOD	YUV_RGB	Field_OEn	Field_On
0x2105	---	COMR6	TV_SYS_SE:[1:0]		CCIR_SEL	Double	ROM_SEL	PRAM

SLAVE : 从动装置(slave)方式致能控制, VT1682 画面和声音会与 CCIR 界面一致.

0 : 正常方式 1 : 从动装置(Slave)方式(CCIR 输入)

CCIR_Y : CCIR 纵向的补偿通信协议与横向的扫描线的起点.

Effective offset = 64 - CCIR_Y

CCIR_X : CCIR 横向的补偿通信协议与晶振周期的数目.

有效的补偿(Effective offset) = 32 - CCIR_H

注释 : 在 CCIR_V 和 CCIR_H 0 是禁止用的.

Field_On : CCIR 参数, 底色方式致能.

0 : 无作用 1 : 致能

Field_OEn : CCIR 参数,当 Field_On 是 1 时有效.

YUV_RGB : CCIR 参数,输入数据格式的选择.

0 : YUV 4:2:2 方式 1 : GBR 4:2:2 方式

SYNC_MOD : CCIR 参数, 同步的方式的选择.

0 : CCIR656 1 : CCIR601

CbCrSwap : CCIR 参数, Cb 和 Cr 命令的控制.

0 : CbYCrY 1 : CrYCbY

YC_Swap : CCIR 参数, Y 和 CrCb 命令的控制.

0 : CbYCrY 1 : YcbYCr

HS_Phase : CCIR 参数, 在 CCIR601 内 hsync 极性的控制.

0 : 低脉冲(Low pulse) 1 : 高脉冲(High pulse)

VS_Phase, CCIR 参数, 在 CCIR601 内 vsync 极性的控制.

0 : 低脉冲(Low pulse) 1 : 高脉冲(High pulse)

6. 周边(PERIPHERAL)

6.1 提高计算(Arithmetic)单元 –乘法器(Multiplier)和除法器(Divider)

VT1682 提供两个计算的单元,乘法器和除法器,一个是给主 CPU 而另外一个是给 Sound CPU. 这个乘法器是 16 位乘 16 位,这个除法器是 32 位除 16 位. 这个乘法需要 16 个 CPU clock cycle 才能完成操作,而除法需要 32 个 CPU clock cycle 才能完成操作.

6.1.1 乘法器 (16x16)

这个乘法操作是:

ALU_ Multi _operand6, ALU_ Multi _operand5

X) ALU_operand2, ALU_operand1

= ALU_out4, ALU_out3, ALU_out2, ALU_out1

当 ALU_Multi_operand6 被写入时这个操作开始.在乘法运算之后是在 ALU_Multi_operand5 和 ALU_Multi_operand6 的值被改变, 而不是在 ALU_operand1 和 ALU_operand2 的值.

0x2130(W)	ALU_operand1
0x2131(W)	ALU_operand2
0x2132(W)	ALU_operand3
0x2133(W)	ALU_operand4
0x2134(W)	ALU_Multi_operand5
0x2135(W)	ALU__Multi_operand6

0x2130(R)	ALU_out1
0x2131(R)	ALU_out2
0x2132(R)	ALU_out3
0x2133(R)	ALU_out4

6.1.2 除法器

这个除法操作是:

ALU_operand4, ALU_operand3, ALU_operand2, ALU_operand1

ALU__Multi_operand6, ALU__Multi_operand5

= { ALU_out4, ALU_out3, ALU_out2, ALU_out1} + { ALU_out6, ALU_out5}或是+{{ ALU_out6, ALU_out5}*2 - { ALU_out4, ALU_out3, ALU_out2, ALU_out1}}

当 LSB(Least Significant Bit) 为 “1”时, 余数会是 :

{ ALU_out6, ALU_out5}*2 - { ALU_out4, ALU_out3, ALU_out2, ALU_out1}

当 LSB 为 “0”时, 余数会是 { ALU_out6, ALU_out5}.

当 ALU__Div_operand6 被写入时这个操作开始. 在除法运算之后是在 ALU_operand1 和 ALU_operand2, ALU_operand3 和 ALU_operand4 的值被改变,而不是在 ALU_Div_operand5 和 ALU_Div_operand6 的值.

0x2130(W)	ALU_operand1
0x2131(W)	ALU_operand2
0x2132(W)	ALU_operand3

0x2133(W)	ALU_operand4
0x2136(W)	ALU_Div_operand5
0x2137(W)	ALU_div_operand6

0x2130(R)	ALU_out1
0x2131(R)	ALU_out2
0x2132(R)	ALU_out3
0x2133(R)	ALU_out4
0x2134(R)	ALU_out5
0x2135(R)	ALU_out6

6.2 定时器(Timer)

写 Timer_PreLoad 去初始化这个定时器的频率. 写 0x2104 会重新装 Timer_Preload 到定时器, 所以 0x2101 必须要在 0x2104 之前被写入.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2101(W)	Timer_Preload[7:0]							
0x2101(R)	Timer_Preload[7:0]							
0x2102							TMR_IRQ	TMR_EN
0x2103	Timer_IRQ_Clear							
0x2104(W)	Timer_Preload[15:8]							
0x2104(R)	Timer_Preload[15:8]							
0x210B	TSYNEN							

Timer_PreLoad[15:0], Timer IRQ 周期定义.

TSYNEN : 定时器基础频率的选择

	TSYNEN	定时器基础频率
NTSC	1	15.746KHz
	0	5.3693 MHz
PAL	1	15.602KHz
	0	5.32034 MHz

当 TSYNEN = 0 时,

For NTSC,

Period = (65536 - Timer_PreLoad) / 5.3693MHz

Timer_PreLoad = 65536 – (Period(sec) * 5.3693 * 10^6)

For PAL

Period = (65536 - Timer_PreLoad) / 5.32034MHz

Timer_PreLoad = 65536 – (Period (sec) * 5.32034 * 10^6)

TMR_En : 定时器致能控制.

0 : 无作用 1 : 致能

TMR_IRQ, 定时器 IRQ 致能控制.

0 : 无作用 1 : 致能

Timer_IRQ_Clear : 定时器 IRQ 清除控制, 写任何数据去清除定时器 IRQ.

6.3 随机数产生器(Random Number Generator)

在 VT1682 有一个 8-bits 虚拟的随机数产生器.它需要去写一个非 0 的种子数字到 0x212C 开头. 这个随机数在读取 0x212C 时会是有效的. 每一次一个新的种子数字被写入,一个新的虚拟的随机数序号会被产生出来.请注意当你开始对这个随机数寻址时是不需要每次去写这个种子数字的.只有在初始的顺序或是改变这个随机数的顺序时你必须去更新这个发送数字.

	D7	D6	D5	D4	D3	D2	D1	D0
0x212C(W)	Pseudo_random_number_seed							
0x212C(R)	Pseudo_random_number							

6.4 模拟到数字转换器 (ADC)

在 VT1682 有一个 ADC,它拥有 5 个时间分割复合(timing-division-multiplex)通道和 8 位的分辨率. 其中的 4 个是标准的 ADC 端口.输入电压在(VCC-1) volt 和 GND 之间会被转换到 255~0. 另外一个端口是给麦克风应用的,它包括有一个声音增大控制器(VGC). 这个 VGC 增大是在 0.5 倍和 127.5 倍之间.当这些端口 XIOF0, XIOF1, XIOF2, XIOF3 或是 XIOE3 是 ADC 的输入时,记得设置 IOFOE 到低电平.

	D7	D6	D5	D4	D3	D2	D1	D0
0x211E(W)	ADCEN	ADCSEL		UNUSE	IOFOE3	IOFOE2	IOFOE1	IOFOE0
0x211E®	ADC_Data[7:0]							
0x211F	VGCEN	VGCGAIN						

VGC Gain = VGCGAIN + 0.5

VGCEN : VGC 致能控制

0： 无作用　　1： 致能

ADCEN : ADC 致能控制

0： 关掉　　1： 正常操作

ADCSEL	ADC Input Source
0	XIOF0
1	XIOF1
2	XIOF2
3	XIOF3

程序设计注释:

1. IOFOE0 = 0, ADCSEL = 0　　// 选择 XIOF0
2. ADAC_EN = 1,　　// ADC 致能
3. 定期的读取 ADC_Data　　// 取得 ADC 数据

6.5 相位锁定回路(Phase Lock Loop (PLL))

在 VT1682 内建一个 PLL 作为 LCD 应用. 它可以产生下面的效果

	D7	D6	D5	D4	D3	D2	D1	D0
0x211D	LVDEN	LVDS1	LVDS0	VDAC_EN	ADAC_EN	PLL_EN	LCDACEN	---
0x212D	SCALE_B				SCALE_M	SCALE_A		

F_{SCALE} = Fosc * (SCALE_A + 2) * (SCALE_M+1) / (SCALE_B + 2)

6.6 数字到模拟转换器(DAC)

在 VT1682 内有 6 个 DAC (数字到模拟转换器). 一个给视频(Video), 两个给音频(Audio),另外三个给模拟(Analog)LCD 界面. 然而,在应用时不是所有的 DAC 都可以被使用. 例如,如果你使用数字(Digital)LCD 界面,这三个 LCD ADC 将不能使用. 所以你可以把它们当额外的应用.

名称	功 能	分辨率	反应时间	输出 PAD	控制端口
Video DAC	Composite Video	6	50ns	XVIDEO	0x2030
LCD DAC1	Analog LCD	5	50ns	XIOE0	0x2031
LCD DAC2	Analog LCD	5	50ns	XIOE1	0x2032
LCD DAC3	Analog LCD	5	50ns	XIOE2	0x2033
Audio DAC1*	Audio	12	5us	XAUDIOL	0x2118, 0x2119(SCPU)
Audio DAC2*	Audio	12	5us	XAUDIOR	0x211A, 0x211B(SCPU)

* Audio DAC 端口 0x2118~0x211B 是只给 Sound CPU 寻址.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2030	----	VDACSW	VDAC_OUT[5:0]					
0x2031	----	----	RDACSW	RDAC_OUT[4:0]				
0x2032	----	----	GDACSW	GDAC_OUT[4:0]				
0x2033	----	----	BDACSW	BDAC_OUT[4:0]				

VDACSW : Video DAC 输出数据开辟.

0 : 电视合成的输出(默认) 1: 输出 VDAC_OUT

RDACSW : LCD DAC 输出数据开辟

0 : 输出 LCD 控制器数据(默认) 1: 输出 RDAC_OUT

GDACSW : LCD DAC 输出数据开辟

0 : 输出 LCD 控制器数据(默认) 1: 输出 GDAC_OUT

BDACSW : LCD DAC 输出数据开辟

0 : 输出 LCD 控制器数据(默认) 1: 输出 BDAC_OUT

注释 1 : 如果你需要电视合成的输出, VDACSW 必须设置在低电平(LOW).

注释 2 : 如果模拟的(Analog) RGB 数据被使用在 LCD 界面, 这些开关必须设置在低电平(LOW).

注释 3 : 音频(Audio)DAC 控制请参考“Sound CPU” 部份.

6.7 低电压检查 (LVD)

VT1682 有四个阶段的低电压检查功能. 它们为 3.65V, 3.18V, 2.71V 和 2.24V. 当 IC 的电源比检查出的标准低时, 在 0x211D 的 LVD 标志会进入 “high”. 写 “1” 到 LVDEN 去使 LVD 功能致能.

	D7	D6	D5	D4	D3	D2	D1	D0
0x211D(W)	LVDEN	LVDS1	LVDS0	VDAC_EN	ADAC_EN	PLL_EN	LCDACEN	---
0x211D(R)	----	----	----	----	----	----	----	LVD

LVDEN : LVD 致能控制

0： 无作用　　1： 致能

6.8 内部的 ROM(Embedded ROM)

在 VT1682 内部有一个 4K 字节的 ROM(Embedded ROM) .它可以给主 CPU 或是 Sound CPU 两者任一来寻址,由 0x2105 内的 ROM_SEL 来选择而且这个 ROM 它的地址是在 0x3000 和 0x3FFF 之间.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2105(W)	---	COMR6	TV_SYS_SE:[1:0]		CCIR_SEL	Dual_Speed	ROM_SEL	PRAM

ROM_SEL	Main CPU	Sound CPU
0	0x3000 ~ 0x3FFF	Invalid
1	Invalid	0x3000 ~ 0x3FFF

这个 ROM 也可以是一个 Boot ROM. 当“XBOOTINIT” 这个脚位被置为高电平时, 这些 RESET, NMI, IRQ 序号(vectors)会从 0x7FFF0 ~ 0x7FFFF 改变到 0x00FF0 ~0x00FFF.

7. GRAPHIC –图形处理单元(PPU)

7.1 特点(Feature)

- 分辨率: TV 256x240 点
- 在同一个画面(Frame)有 240 个卡通块，在横向最大的卡通块数是 16 个
- 2 个独立的背景层.
- 背景字符方式: 16/64/256 索引颜色方式.
- 背景数字映像的方式: 16/64/256 索引颜色方式.或是 32768 色直接颜色方式
- 卡通块(Sprites)为 16 色.
- 两个 256 上色颜色调色板，最大显示的索引颜色: 512
- 背景纵向的宽度: x1/x1.5/x2
- 背景横向的线单独卷动: -128~+127

7.2 画面结构(Screen Structure)

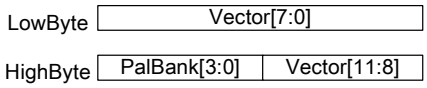
在 VT1682 显示画面(在电视机)图像的分辨率是 256 x 240 点.然而这个 VRAM 画面(这个图形块定义在给背景层 (background layer)的 VRAM)是 256 x256 点.

7.2.1 VRAM 画面结构(Screen Structure)

去显示一个背景在电视机的画面,这个在 VRAM 的图形块必须要先被定义. 在 VRAM 有两类的画面结构,它们是字符(Character)和数字映像(Bitmap)方式.在字符(Character)方式，这个画面分割成字符块.在数字映像方式,这个画面分割成一条一条线.

7.2.1.1 字符方式(Character Mode)

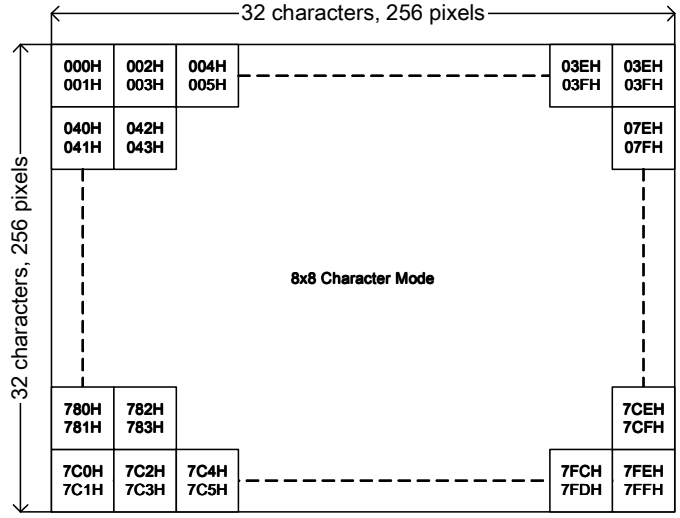
在背景的每一个字符需要两个字节来定义. 如下面的图形所示.这两个字节包括 12 bits 的字符序号(character vector)和 4 bits 的调色板储存空间参数(palette bank parameters). 序号(Vector) = 0 表示为透明的字符(transparent character).请参考图像地址方式(Graphic Addressing)部份和颜色调色板部份有关于这些参数的详细描述.



不同的字符大小会需要不同数量的字符图形块来定义一个画面,如下面的表格所示:

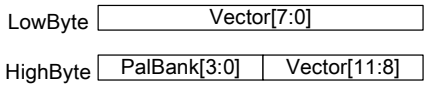
字符大小 (H x V)	需要的图形块数量	需要的 VRAM (字节)
8 x 8	32 x 32	2048
8 x 16	32 x 16	1024
16 x 8	16 x 32	1024
16 x 16	16 x 16	512

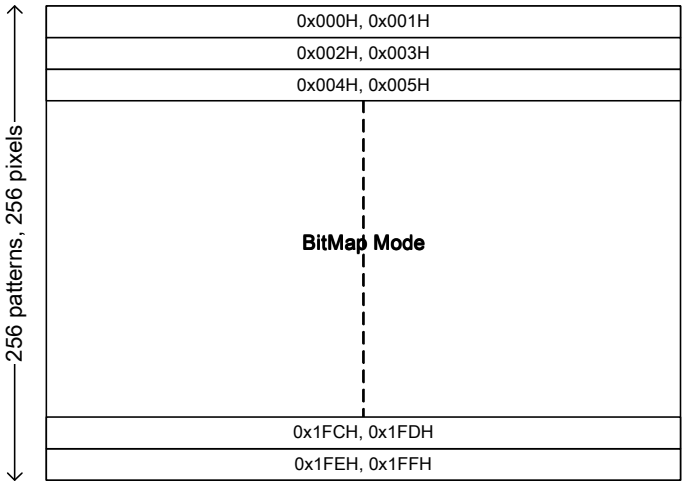
下面的例子是一个 8x8 字符(Character)方式它需要 32 x 32 的图形块去定义一个画面.



7.2.1.2 数字映像(Bitmap)方式

在这个方式，画面分割到 256 条线(图形块) 而且每一条线需要两个字节来定义，如下面的图表所示. 序号(Vector)= 0 表示在数字映像方式的透明线,但是不是在高颜色方式.





7.2.1.3 VRAM 管理(VRAM Management)

有一个 4K 字节的 VRAM 给背景层用. 在字符方式，不同的字符大小会需要不同的 VRAM 区域 (**7.2.1.1 字符方式**). 底下的表格所列是 BK1 和 BK2 在不同的字符大小和滚动条方式的存储器映像 (memory map).

BK1

	V_scroll_en	H_scroll_en	Y[8]	X[8]	BK1
8x8	0	0	0	0	0x000 ~ 0x7FF
			0	1	0x800 ~ 0xFFF
			1	0	0x800 ~ 0xFFF
			1	1	0x800 ~ 0xFFF
	0	1	0	0	0x000 ~ 0x7FF, 0x800 ~ 0xFFF
			0	1	0x800 ~ 0xFFF, 0x000 ~ 0x7FF
			1	0	0x000 ~ 0x7FF, 0x800 ~ 0xFFF
			1	1	0x800 ~ 0xFFF, 0x000 ~ 0x7FF
	1	0	0	0	0x000 ~ 0x7FF, 0x800 ~ 0xFFF
			0	1	0x000 ~ 0x7FF, 0x800 ~ 0xFFF
			1	0	0x800 ~ 0xFFF, 0x000 ~ 0x7FF
			1	1	0x800 ~ 0xFFF, 0x000 ~ 0x7FF
	1	1	0	0	Forbidden
			0	1	Forbidden
			1	0	Forbidden
			1	1	Forbidden
16x16	0	0	0	0	0x000 ~ 0x1FF
			0	1	0x200 ~ 0x3FF
			1	0	0x400 ~ 0x5FF
			1	1	0x600 ~ 0x7FF
	0	1	0	0	0x000 ~ 0x1FF, 0x200 ~ 0x3FF
			0	1	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
			1	0	0x000 ~ 0x1FF, 0x200 ~ 0x3FF
			1	1	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
	1	0	0	0	0x000 ~ 0x1FF, 0x200 ~ 0x3FF
			0	1	0x000 ~ 0x1FF, 0x200 ~ 0x3FF
			1	0	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
			1	1	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
	1	1	0	0	0x000 ~ 0x1FF, 0x200 ~ 0x3FF, 0x400 ~ 0x5FF, 0x600 ~ 0x7FF,
			0	1	0x200 ~ 0x3FF, 0x000 ~ 0x1FF, 0x600 ~ 0x7FF, 0x400 ~ 0x5FF,
			1	0	0x400 ~ 0x5FF, 0x600 ~ 0x7FF, 0x000 ~ 0x1FF, 0x200 ~ 0x3FF,
			1	1	0x600 ~ 0x7FF, 0x400 ~ 0x5FF, 0x200 ~ 0x3FF, 0x000 ~ 0x1FF,

BK2

	V_scroll_en	H_scroll_en	Y[8]	X[8]	BK2
8x8	0	0	0	0	0x000 ~ 0x7FF
			0	1	0x800 ~ 0xFFF
			1	0	0x800 ~ 0xFFF
			1	1	0x800 ~ 0xFFF
	0	1	0	0	0x000 ~ 0x7FF, 0x800 ~ 0xFFF
			0	1	0x800 ~ 0xFFF, 0x000 ~ 0x7FF
			1	0	0x000 ~ 0x7FF, 0x800 ~ 0xFFF
			1	1	0x800 ~ 0xFFF, 0x000 ~ 0x7FF
	1	0	0	0	0x000 ~ 0x7FF, 0x800 ~ 0xFFF
			0	1	0x000 ~ 0x7FF, 0x800 ~ 0xFFF
			1	0	0x800 ~ 0xFFF, 0x000 ~ 0x7FF
			1	1	0x800 ~ 0xFFF, 0x000 ~ 0x7FF
	1	1	0	0	Forbidden
			0	1	Forbidden
			1	0	Forbidden
			1	1	Forbidden
16x16	0	0	0	0	0x800 ~ 0x9FF
			0	1	0xA00 ~ 0xBFF
			1	0	0xC00 ~ 0xDFF
			1	1	0xE00 ~ 0xFFF
	0	1	0	0	0x800 ~ 0x9FF, 0xA00 ~ 0xBFF
			0	1	0xA00 ~ 0xBFF, 0x800 ~ 0x9FF
			1	0	0x800 ~ 0x9FF, 0xA00 ~ 0xBFF
			1	1	0xA00 ~ 0xBFF, 0x800 ~ 0x9FF
	1	0	0	0	0x800 ~ 0x9FF, 0xA00 ~ 0xBFF
			0	1	0x800 ~ 0x9FF, 0xA00 ~ 0xBFF
			1	0	0xA00 ~ 0xBFF, 0x800 ~ 0x9FF
			1	1	0xA00 ~ 0xBFF, 0x800 ~ 0x9FF
	1	1	0	0	0x800 ~ 0x9FF, 0xA00 ~ 0xBFF, 0xC00 ~ 0xDFF, 0xE00 ~ 0xFFF,
			0	1	0xA00 ~ 0xBFF, 0x800 ~ 0x9FF, 0xE00 ~ 0xFFF, 0xC00 ~ 0xDFF,
			1	0	0xC00 ~ 0xDFF, 0xE00 ~ 0xFFF, 0x800 ~ 0x9FF, 0xA00 ~ 0xBFF,
			1	1	0xE00 ~ 0xFFF, 0xC00 ~ 0xDFF, 0xA00 ~ 0xBFF, 0x800 ~ 0x9FF,

在数字映像方式, BK1 只有 240/256 序号(vectors)是被需要的.

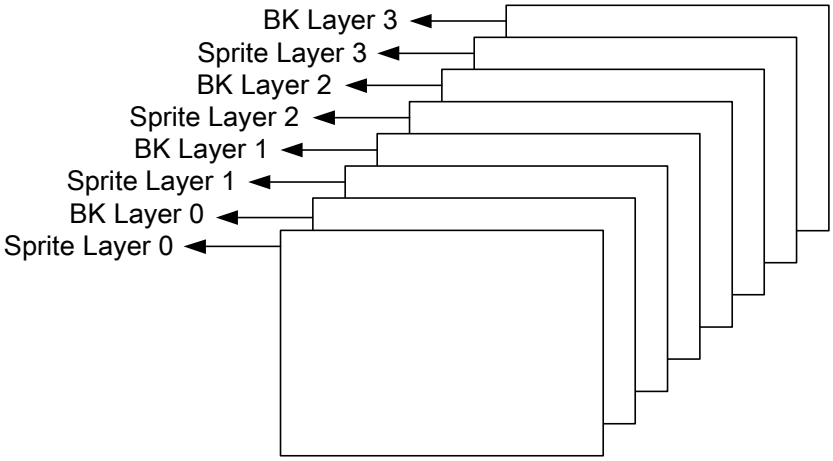
	V_scroll_en	H_scroll_en	Y[8]	X[8]	BK1
16x16	0	0	0	0	0x000 ~ 0x1FF
			0	1	0x200 ~ 0x3FF
			1	0	0x400 ~ 0x5FF
			1	1	0x600 ~ 0x7FF
	0	1	0	0	0x000 ~ 0x1FF, 0x200 ~ 0x3FF
			0	1	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
			1	0	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
			1	1	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
	1	0	0	0	0x000 ~ 0x1FF, 0x200 ~ 0x3FF
			0	1	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
			1	0	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
			1	1	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
	1	1	0	0	0x000 ~ 0x1FF, 0x200 ~ 0x3FF, 0x400 ~ 0x5FF, 0x600 ~ 0x7FF,
			0	1	0x200 ~ 0x3FF, 0x000 ~ 0x1FF, 0x600 ~ 0x7FF, 0x400 ~ 0x5FF,
			1	0	0x400 ~ 0x5FF, 0x600 ~ 0x7FF, 0x000 ~ 0x1FF, 0x200 ~ 0x3FF,
			1	1	0x600 ~ 0x7FF, 0x400 ~ 0x5FF, 0x200 ~ 0x3FF, 0x000 ~ 0x1FF,

7.2.2 显示画面结构(Display Screen Structure)

TV / LCD 显示画面的分辨率为 256 x 240 点.即使 VRAM 画面的分辨率为 256 x 256 点,部份的背景在电视画面是看不到的.请参考“Graphic 坐标部份”有显示画面和 VRAM 画面相关性的描述.

7.3 深度层次(Depth Layer)

每一个卡通块或是背景有一个深度层次参数.它定义适当的位置的显示物体.在 VT1682 有 8 个深度层次可以利用, 4 个给背景而其他的给卡通块,它们的关系如下面图表所示.当数个物体重迭在一起时,这个层次有较小的深度数会被显示. 当两个背景层有相同的深度层次时, 背景 1(Background1)会被显示. 当两个卡通块有相同的深度层次时,有较低卡通块数的卡通块(这个映像地址在卡通块 RAM, 0~239)会被显示.



7.4 图像图形块格式(Graphic Pattern Format)

7.4.1 字符(Character)方式

字符(Character) 8x8 方式

在 8x8 方式,在横向的数据顺序为 a,b,c,….h 如下面的图表所示.

a	b	c	d	e	f	g	h
i	j	k	l	m	n	o	p

字符(Character) 8_H x 16_V 方式

这个数据顺序和图形块数据除了这个图形块数据的长度之外跟 8x8 方式是一样的.

字符(Character) 16_H x 8_V 方式

在横向的数据顺序为 a,b,c,….h 如下面的图表所示.

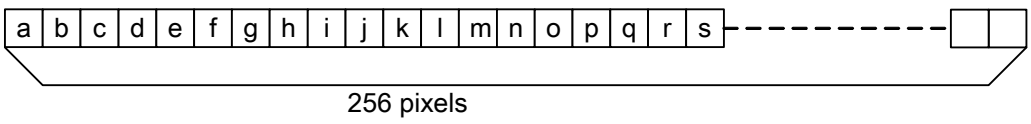
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
q	r	s	t	u	v	w	x								

字符(Character) 16_H x 16_V 方式

这个数据顺序和图形块数据除了这个图形块数据的长度之外跟 16_H x 8_V 方式是一样的.

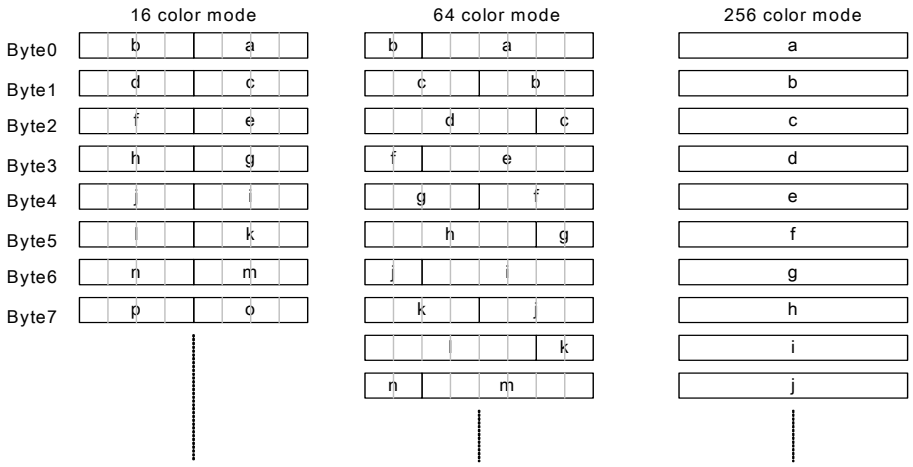
7.4.2 数字映像(Bitmap)方式

在数字映像方式, 这个图形块数据的定义是线靠线(line-by-line),它的顺序如下面的图所示.



7.4.3 颜色方式(Color Mode)

在 VT1682 有 4 种颜色方式,它们是 16, 64, 256 和 32768 色方式. 背景 1(Background 1)可以是 16, 64, 256 和 32768 色方式之中的一种. 背景 2(Background 2)可以是 16, 64, 和 256 色方式之中的一种.而卡通块(Sprite)只有在 16 色方式有效. 这个图形块对应不同的颜色方式如下面的图表所示:



7.4.4 透明的颜色(Transparent Color)

在索引颜色方式,有 16/64/256 色方式,这个透明的索引会是 0. 例如,在 16 色方式下带有数据 0x00 的点会是透明的.

7.4.5 高颜色方式(High Color Mode)

在高颜色(32768 色) 方式, 每一个点需要两个字节,它的格式如下:

低字节(Low Byte)

D7 – D5	D4 – D0
Green[2:0]	Blue[4:0]

高字节(High Byte)

D15	D14 – D10	D9 – D8
TRPT	Red[4:0]	Green[4:3]

在这个方式,如在颜色调色板内的内容,每一个点有 5 bits R, G, B 成分. 这个 MSB TRPT 是给透明的标志使用的.当这个 TRPT 标志为“1” 时,这个点将会变成透明的.当一个点被设置为透明的表示这些其他的具有最少深度层次的点将会被显示.如果没有其他的层次在它的下面,这个 R, G, B 颜色会被显示.

7.5 颜色调色板(Color Palette)

颜色调色板是被使用来转换索引颜色为物理的 RGB 颜色. 在 VT1682 有两个独立的颜色调色板. 在单一个显示画面每一个调色板可以从 32768 色选择 256 色来显示.

7.5.1 调色板内容(Palette Content)

这个颜色主要是 R, G, B, 每一个成分分辨率为 5 bits. 如下面的图表所示,每一个颜色在调色板需要两个字节来定义, 包括 5-bits Red, 5-bits Green, 5-bits Blue 和一个 DG 标志给特殊效果.关于这个 DG 的应用请参考图像特殊效果部份(Graphic Special Effect).

低字节(Low Byte):

D7 – D5	D4 – D0
Green[2:0]	Blue[4:0]

高字节(High Byte):

D7	D6 – D2	D1 – D0
DG	Red[4:0]	Green[4:3]

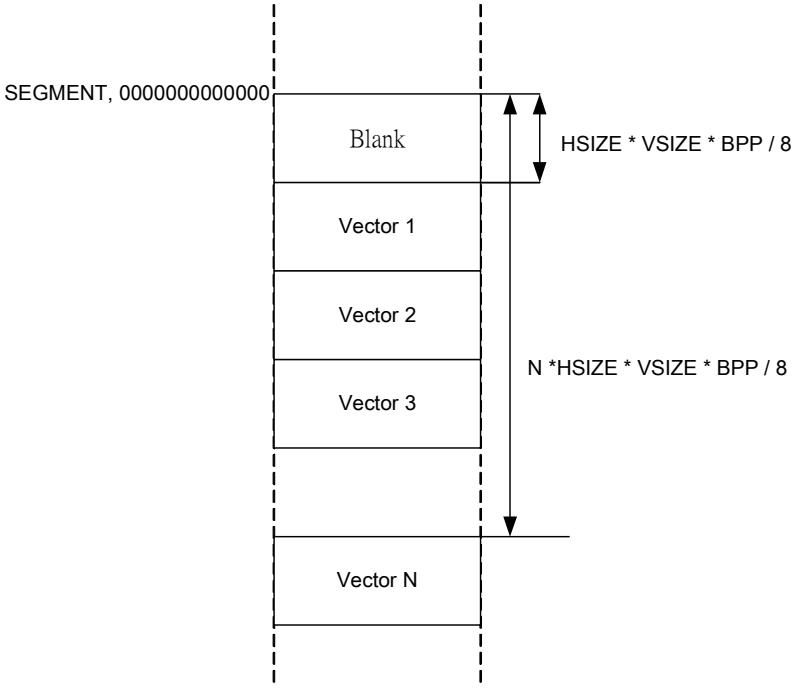
7.5.2 调色板储存空间(Palette Bank)

在调色板内有 256 色.在 16 色方式,它可以分割成 16 个储存空间(banks).在 64 色方式, 它可以分割成 4 个储存空间(banks) .这个储存空间(bank)是由“PalBank” 参数来定义.每一个图形块在 VRAM 有个别的“PalBank” ,定义在 D7:D4 的高字节图形块(High byte pattern).

	D7	D6	D5	D4	D3	D2	D1	D0
16 colors mode	PalBank[3:0]				D3 : D0			
64 colors mode	**PalBank[3:2]**		D5 : D0					
256 colors mode	D7 : D0							

7.6 图像地址方式(GRAPHIC ADDRESSING MODE)

这个图像图形的地址是以 SEGMENT 为起点,序号(Vector)号码如下面的图标.有三组的 SEGMENT 给卡通块(sprites),背景 1(background1 (BK1))和背景 2(background2 (BK2)).



物理地址= (Segment << 13) + Offset

Offset = Vector x H_Size x V_Size x Color_Mode / 8

V_Size 是 8 或是 16,

H_size 是 8 或是 16

Color_Mode 是 4 为 16 色方式, 6 为 64 色方式, 8 为 256 色方式.

Segment = {Segment_H, Segment_L}(Sprite/BK1/BK2)

例如:

如果 BK1 字符数据被开始具有地址 0x20000,

那么 segment = (0x20000) >> 13 = 0x10

如果 BK1 字符为 8x8 具有256色具有 vector 3,那么

Offset = 3 x 8 x 8 x 8 / 8 = 0xC0,

这个字符数据会从地址 0x200C0开始.

注释 1： 在数字映像方式或是高颜方式, V_Size 和 H_Size 两者都是 16.

注释 2： 在高颜色方式, Color_Mode 是 8.

注释 3： 所有的序号(vectors)在高颜色方式都是偶数.

	D7	D6	D5	D4	D3	D2	D1	D0
0x201A	SP_SEGMENT[7:0]							
0x201B					SP_SEGMENT[11:8]			
0x201C	BK1_SEGMENT[7:0]							
0x201D					BK1_SEGMENT[11:8]			
0x201E	BK2_SEGMENT[7:0]							
0x201F	----	----	----	----	BK2_SEGMENT[11:8]			

SP_SEGMENT : Segmentation number (8KB bank) 代表卡通块(Sprites)

BK1_SEGMENT : Segmentation number (8KB bank) 代表背景 1(BK1)

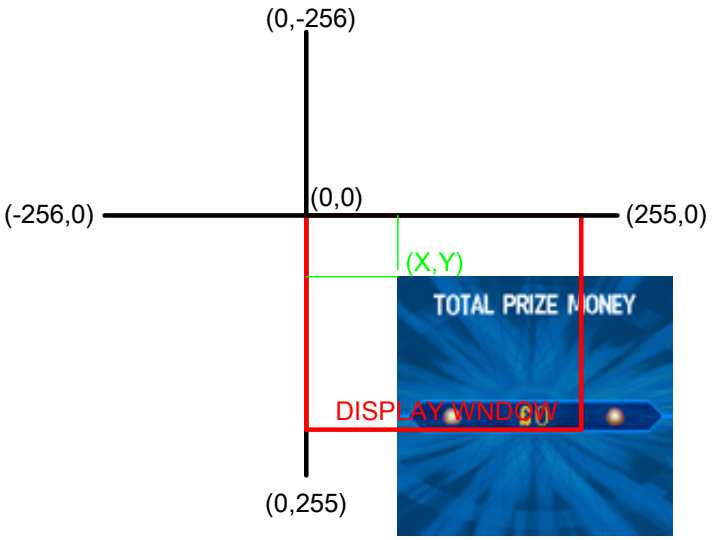
BK2_SEGMENT : Segmentation number (8KB bank) 代表背景 2 (BK2)

7.7 背景层(Background Layer)

在 VT1682 有两个背景层. 它们是背景 1(background1 (BK1))和背景 2(background2 (BK2)).

7.7.1 坐标(coordinate)

背景层的 X-轴是介于 –256 和 255 之间, Y-轴也是. (X, Y) 是表示在 2's 非.当这个 X 是正的(X[8] =0)时,这个显示背景会卷到右边. 当这个 Y 是正的(Y[8] = 0)时,这个显示背景会卷到下面.如下面的图示:



7.7.2 滚动条(Scrolling)

BKx_X[8:0] 和 BKx_Y[8:0] 是被使用来控制背景层的滚动条. BKx_Scroll_En 参数在每一个背景层提供 4 种滚动条方式.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2010	BK1_X[7:0]							
0x2011	BK1_Y[7:0]							
0x2012				BK1_HCLR	BK1_Scroll_En		BK1_Y8	BK1_X8
0x2014	BK2_X[7:0]							
0x2015	BK2_Y[7:0]							
0x2016					BK2_Scroll_En		BK2_Y8	BK2_X8

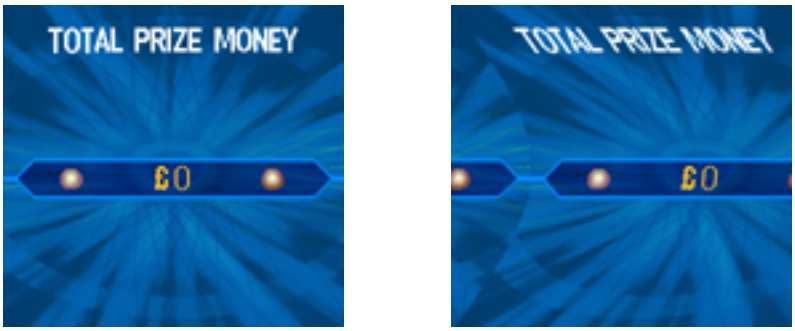
7.7.2.1 画面滚动条(Frame Scroll)

每一个背景(background)有 9-bits X 和 Y 坐标，提供达到 512x512 滚动条功能.取决于滚动条方式, BK1_Scroll_En 和 BK2_Scroll_En，它们定义在 0x2012 和 0x2016，这个滚动条效果将会不同.

滚动条方式(Scrolling Mode)	BK_Scroll_En	Effective X	Effective Y
固定页方式	0	0 ~ 255	0 ~ 255
横向两页方式	1	-256 ~ 255	0 ~ 255
纵向两页方式	2	0 ~ 255	-256 ~ 255
四页方式	3	-256 ~ 255	-256 ~ 255

7.7.2.2 线滚动条(Line Scroll)

VT1682 提供背景层线对线(line-by-line)横向的滚动条方式，如下面的图示:



线滚动条方式可以应用在 BK1 或是 BK2 两个之中任一个,或是两个都用程序来写寄存器 0x2020 的 BK1_L_EN 和 BK2_L_EN . 240 个字节的滚动条序号会在这个功能致能之前会被存放在介于{ Scroll_Bank, 8'b0000_0000} 和 { Scroll_Bank, 8'b1110_1111}之间的 PRAM. 每一个序号映射到每一条横向的扫描线. 这个序号值会是介于 –128 和 +127 之间而且它表示在 2's 非.这个序号会被加到背景层的 X.请注意这跟背景纵向的坐标 BK_Y 是独立的.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2020	----	----	BK2_L_EN	BK1_L_En	Scroll_Bank			

BK1_L_En : BK1 线滚动条方式控制

BK2_L_En : BK2 线滚动条方式控制

0： 无作用　　1： 致能

Scroll_Bank： 当作线滚动条方式序号的 PRAM 存储空间.

{ Scroll_Bank, 8'b0000_0000} 是当作 1st 线 (上面)，和 { Scroll_Bank, 8'b1111_0111} 是当作 240th 线 (下面).

7.7.3 颜色方式(Color Mode)

在背景层的颜色方式可以是 16, 64 或是 256 色.换句话说，在背景层的每一个点是表示在 4, 6 或是 8 位. 每一个字符(线)有各自的调色板存储空间,那个定义在这个 VRAM 的高字节 bit7~bit4,如下面的表格所示.调色板存储空间(Palette Bank) (PalBank，参考 **7.5.2 Palette Bank**),有两种组合当作不同的应用,如下面的表格所示:

背景 1(Background1)

BK1_Pal_Sel	16 色方式	64 色方式	256 色方式
0	Depth = 0x2013[D5:D4]	Depth = 0x2013[D5:D4]	Depth = 0x2013[D5:D4]
	PalBank = VRAM[15:12]	PalBank = VRAM[15:14]	----
1	Depth = VRAM[13:12]	Depth = VRAM[13:12]	Depth = VRAM[13:12]
	PalBank = {0x2013[5:4],VRAM[15:14] }	PalBank = VRAM[15:14]	----

背景 2(Background2)

BK2_Pal_Sel	16 色方式	64 色方式	256 色方式
0	Depth = 0x2017[5:4]	Depth = 0x2017[5:4]	Depth = 0x2017[5:4]
	PalBank = VRAM[15:12]	PalBank = VRAM[15:14]	----
1	Depth = VRAM[13:12]	Depth = VRAM[13:12]	Depth = VRAM[13:12]
	PalBank = { 0x2017[5:4],VRAM[15:14] }	PalBank = VRAM[15:14]	----

7.7.4 字符方式(Character Mode)

BK1_Line = 0; BK1_HCLR=0;

每一个背景层可以被 8x8 字符或是 16x16 字符之中任何一个来形成.当这个字符方式被选择时，这个在 0x2013 和 0x2017 的 BKx_Line 必须被置为 0，而 BKx_Size 是当作字符大小的选择.在 VRAM 它会拿 32x32 字(words)来定义一个 8x8 字符方式而在 16x16 字符方式为 16x16 字(words).

	D7	D6	D5	D4	D3	D2	D1	D0
0x2013	BK1_EN	BK1_Pal	BK1_Depth		BK1_Color		BK1_Line	BK1_Size
0x2017	BK2_EN	BK2_Pal	BK2_Depth		BK2_Color		----	BK2_Size

7.7.5 数字映像方式(Bitmap Mode)

BK1_Line = 1; BK1_Size = 1; BK1_EN = 1; BK1_HCLR=0

在数字映像方式，这个背景层被分割成 256 条横向的(水平的)线.在背景层需 256 条横向的(水平的)线要 256 字(words) .请注意这个数字映像方式只有在背景 1(Background1)是有效的.

7.7.6 高颜色方式(High Color Mode)

BK1_Line = 1; BK1_Size = 1; BK1_EN = 1; BK1_HCLR=1; BK2_EN = 0;

在背景 1(Background1)高颜色方式是去显示达到 32768 色.在上面的数字映像描述,给每一个背景层的这个颜色方式可以是 16, 64 或是 256 色.在高颜色方式,每一个点被用两个字节定义,它们的数据格式如下面的图标.请注意这个数字映像方式只有在背景 1(Background1)是有效的.当背景 1(background1)是在高颜色方式,背景 2(background2)将会无作用.除此之外,这个在 VRAM 的序号必须是偶数.

数据格式(Data Format):

低字节(Low Byte)

D7 – D5	D4 – D0
Green[2:0]	Blue[4:0]

高字节(High Byte)

D15	D14 – D10	D9 – D8
TRPT	Red[4:0]	Green[4:3]

7.7.7 深度(Depth)

深度参数是去定义显示的优先级.有较低优先级的对象会被显示在这个画面.每一个背景层有四个深度层次.在背景的所有的字符(线)会有一个共同的深度参数或是各自的深度,如下面的图示:

背景 1(Background1)

BK1_Pal_Sel	16 色方式	64 色方式	256 色方式
0	Depth = 0x2013[D5:D4]	Depth = 0x2013[D5:D4]	Depth = 0x2013[D5:D4]
	PalBank = VRAM[15:12]	PalBank = VRAM[15:14]	----
1	Depth = VRAM[13:12]	Depth = VRAM[13:12]	Depth = VRAM[13:12]
	PalBank = {0x2013[5:4],VRAM[15:14] }	PalBank = VRAM[15:14]	----

背景 2(Background2)

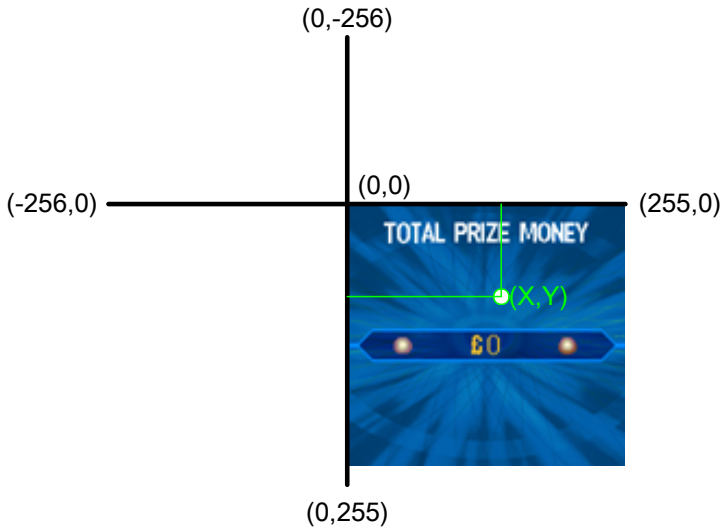
BK2_Pal_Sel	16 色方式	64 色方式	256 色方式
0	Depth = 0x2017[5:4]	Depth = 0x2017[5:4]	Depth = 0x2017[5:4]
	PalBank = VRAM[15:12]	PalBank = VRAM[15:14]	----
1	Depth = VRAM[13:12]	Depth = VRAM[13:12]	Depth = VRAM[13:12]
	PalBank = { 0x2017[5:4],VRAM[15:14] }	PalBank = VRAM[15:14]	----

7.8 卡通块层(Sprite Layer)

在一个显示画面有 240 个卡通块. 每一个横向的(水平的)方向线可以显示 16 个卡通块,每一个卡通块有各自的深度, 序号, X, Y, 调色板存储空间, 调色板选择参数.

7.8.1 卡通块坐标(Sprite Coordinate)

每一个卡通块有 9-bits X 和 Y 坐标.这个起始坐标是在这个显示画面的左上角,如下的图示:



7.8.2 卡通块大小(Sprite Size)

卡通块的大小可以是 8x8, 8x16, 16x8 或是 16x16. 所有的卡通块有相同的字符大小由 0x2018 内的 SP_Size 来定义.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2018					SPALSEL	SP_EN	SP_SIZE	

SP_size : 卡通块大小(V x H)

0 : 8x8 1 : 8x16

2 : 16x8 3 : 16x16

7.8.3 卡通块控制(Sprite Control)

在 0x2018 设置 SP_EN 使卡通块显示致能.如果在横向的在线的卡通块数目超过 16 个, 这个在 0x2001 的超过标志 SP_ERR 会是 “1”. 这个标志是线对线来修改,而且只有在横向的空白期间是有效的.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2001(R)	VBLANK	SP_ERR						

SP_ERR : 在单一的横向的在线超过 16 个卡通块.

0 : 没超过 1 :超过

7.8.4 卡通块调色板选择(Sprite Palette Selection)

在 VT1682 有两个颜色调色板(参考 **7.10 Palette Select**).每一个卡通块有各自的参数,在卡通块 RAM 的 PalSel 来选择颜色的调色板.设置这个 SPALSEL 可以容许所有的卡通块在同一时间显示在 Palette1 和 Palette2 .

SPALSEL	PalSel in SpriteRAM	Palette1	Palette2
0	0	Yes	No
	1	No	Yes
1	0/1	Yes	Yes

7.8.5 卡通块 RAM 的格式(Sprite RAM data format)

每一个卡通块需要 6 个字节来定义它的属性,包括序号, X, Y, 深度层次,纵向的快速翻动,横向的快速翻动,调色板选择内部的图形块方式.它的形态如下面的表格所示.当这个 DMA 被使用来传送卡通块 RAM 数据时,这个传送来源地址顺序会是

00—01—02—03—04—05—06—07—08—09—0A—0B—0C--……

目的地卡通块 RAM 地址顺序会是

00—01—02—03—04—05—08—09---0A--0B---0C---0D—10--……

在卡通块 RAM 的卡通块数据形态

	D7	D6	D5	D4	D3	D2	D1	D0
SP*8	Vector[7:0]							
SP*8 + 1	Palette[3:0]				Vector[11:8]			
SP*8 + 2	X[7:0]							
SP*8 + 3				Depth[1:0]		Flip[1:0]		X[8]
SP*8 + 4	Y[7:0]							
SP*8 + 5						VRCH	PalSel	Y[8]

* SP 是在卡通块 RAM 的卡通块索引数,介于 0 和 239 之间.

7.9 CCIR 层(CCIRLayer)

VT1682 有两组 CCIR 通信协议给输入这个图像的影像,称之为 CCIR 层.请参考"CCIR 通信协议(Protocol)" 部份有这个信号的相关链接和寄存器的设置. CCIR 会把它当作第三个背景来看,但是只有在 Palette2 是有效的.如背景, CCIR 层有 8 个深度层次(CCIR_Depth) ,它是卡通块和背景深度层次的合成.

7.9.1 CCIR 颜色效果(Color Effects)

有几个颜色的特殊效当作 CCIR 层, 例如灰阶颜色, 半调色颜色和颜色的罩幕.这个灰阶颜色功能是去让在 CCIR 层的色度(Chrominance)无作用,只有当在 0x202A 的 YUV_RGB 为 0 时,这个功能是有效的. 半调色功能是一半的色度成分.在颜色罩幕功能有三个 RGB 颜色罩幕去产生不同的颜色输出. CCIR 层也有一些其他的特殊应用,例如影像传感器,这些功能会各自描述在下面的表格.

	D7	D6	D5	D4	D3	D2	D1	D0

0x202A	VS_Phase	HS_Phase	YC_Swap	CbCrswap	SYNCMOD	YUV_RGB	Field_OEn	Field_On
0x202B	R_MSK	G_MSK	B_MSK	HalfTone	Gray	CCIR_Depth		
0x202E	TRC_EN	CCIR_EN	BlueScr_EN	Touch_EN	CCIR_TH			

R_MSK : CCIR 特殊的效果，红色(red)成分罩幕控制.

0： 正常 1： 无红色成分

G_MSK : CCIR 特殊的效果，绿色(green)成分罩幕控制.

0： 正常 1： 无绿色成分

B_MSK : CCIR 特殊的效果，蓝色(blue)成分罩幕控制.

0： 正常 1： 无蓝色成分

HalfTone : CCIR 特殊的效果，输入颜色分成一半.

0： 正常 I 1 :一半

Gray : CCIR 特殊的效果，改变输入影像为灰阶(gray-levels).

0： 彩色 1： 灰阶

7.9.2 CCIR 影像拍摄(Image Capture)

CCIR 影像可以拍摄和储存在外部的 SRAM.在这个拍摄期间,卡通块不会被影响.只有当 VT1682 在从动装置(SLAVE)方式时,拍摄功能可以被正确的操作.当在 0x2000 的拍摄命令被设置时,这个影像透过 CCIR 界面会是闪光灯和被储存在外部的 SRAM.这个拍摄命令可以在 VBLANK 期间 致能/无作用.这个拍摄影像可以是高颜色或是 16 阶灰阶影像两者任一.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2000				Capture	SLAVE	---	---	NMI_EN

Capture： 在 CCIR 界面拍摄影像

0： 无操作 1: 影像拍摄

7.9.2.1 高颜色影像(High Color Image)

这个拍摄数据的格式是 256 x 240 点阵，每一个点使用一个字(word) (两个字节),而每一个字(word)包括红 (D14-D10)，绿(D9-D5) 和 蓝(D4-D0). 这些下面的寄存器必须在进入拍摄功能之前要被设置.

BK1_Line = 1; BK1_Size = 1; BK1_EN = 1; BK1_HCLR=1; BK2_EN = 0; BK1_X=9; BK1_Y=0; BK1_Scroll_En=0;

这个拍摄影像将会被储存在背景 1(Background1)高颜色方式的图形块区域.所以在 VRAM 的背景 1(BK1)序号会被设置在高颜色方式的格式, 240 个线序号.

7.10 调色板选择(Palette Select)

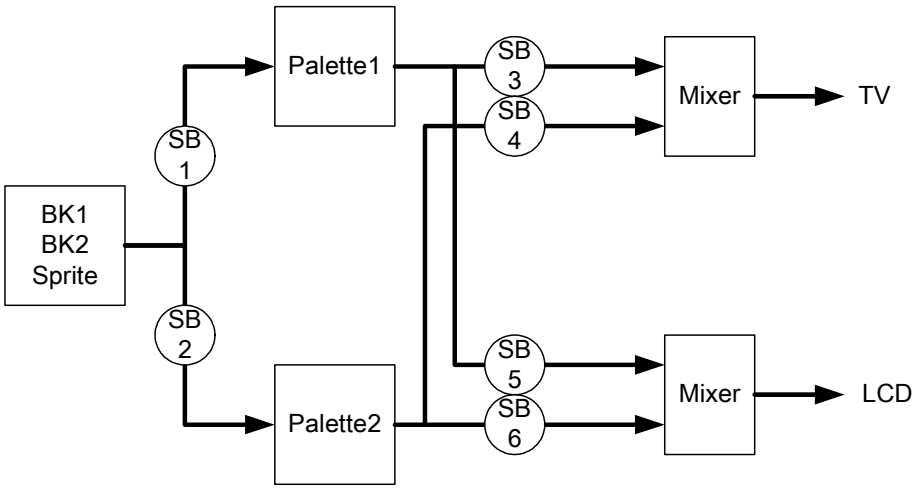
有两个 256 色的颜色调色板可以支持单一显示画面达到 512 色.

背景 1(BK1),背景 2(BK2)和卡通块(Sprite)可以选择各自的调色板.

如下面的图示, SB1 和 SB2 是被使用来选择调色板.

当 SB1 致能时，这个调色板 1(Palette 1)被选到.

当 SB2 致能时，这个调色板 2(Palette 2)被选到.



背景(Background) :

BK1	SB1	0x200F.D0
	SB2	0x200F.D1
BK2	SB1	0x200F.D2
	SB2	0x200F.D3

	D7	D6	D5	D4	D3	D2	D1	D0
0x200F					BK2_Pal_Sel		BK1_Pal_Sel	

BK1_Pal_Sel: 背景 1(BK1)调色板选择

0 : BK1 无作用　　1 : BK1 使用调色板 1(palette1)

2 : BK1 使用调色板 2(palette2)　　3 : BK1 使用调色板 1 和调色板 2

BK2_Pal_Sel:背景 2(BK2)调色板选择

0 : BK2 无作用　　1 : BK2 使用调色板 1(palette1)

2 : BK2 使用调色板 2(palette2)　　3 : BK2 使用调色板 1 和调色板 2

卡通块(Sprite) :

每一个卡通块有各自的调色板选择位,那是被在卡通块 RAM 的 “PalSel” 控制,而这个 SPALSEL 是在 0x2018.当 SPALSEL 是高电平时,所有的 SB1 和 SB2 的卡通块会有作用,这个 PalSel 会无作用.当 SPALSEL 为低电平时,这个 PalSel 将会定义 SB1 和 SB2,当 PalSel=0 时, SB1 被选择,反之 SB2 被选择.

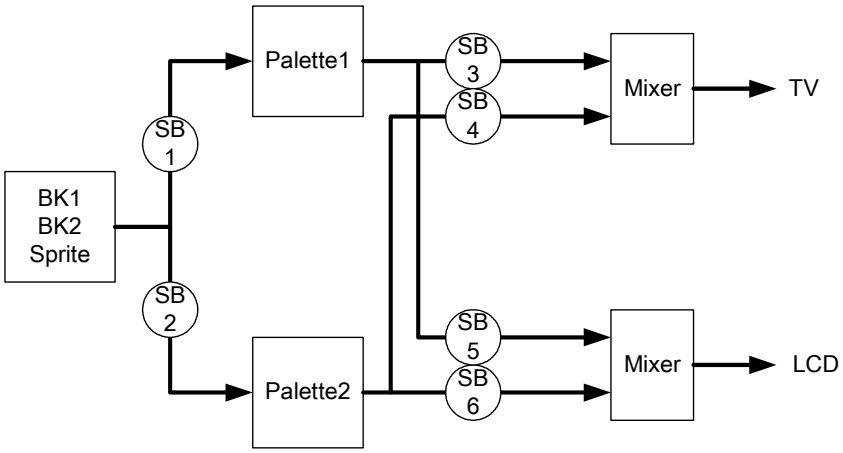
	D7	D6	D5	D4	D3	D2	D1	D0
0x2018					SPALSEL	SP_EN	SP_SIZE	

在卡通块 RAM 的数据格式

	D7	D6	D5	D4	D3	D2	D1	D0
SP*8	Vector[7:0]							
SP*8 + 1	Palette[3:0]				Vector[11:8]			
SP*8 + 2	X[7:0]							
SP*8 + 3				Layer[1:0]		Flip[1:0]		X[8]
SP*8 + 4	Y[7:0]							
SP*8 + 5						VRCH	PalSel	Y[8]

7.11 输出选择(Output Select)

由于有两个 256 色颜色调色板输出(调色板 1(Palette1) 和调色板 2(Palette2)),它们可以改方向到两个各自的输出通信协议(LCD 或是 TV) ,如下面的图示. SB3, SB4, SB5 和 SB6 是被用来选择这个输出的组件. 当 SB3 被致能时,所有用调色板 1(Palette1)的数据会被显示在电视上.当 SB5 被致能时,所有用调色板 1(Palette1)的数据会被显示在 LCD 上. SB3 和 SB5 可以各自被控制.同样的 SB4 和 SB6 也是.当 SB3 和 SB4 同时被致能时,用调色板 1 和调色板 2 的数据会在电视的调整装置上被混合.两种混合方法是有效的, 一种是起点的混合(重迭),是以深度层次为起点.另外一种是颜色的混合.它会从调色板 1 和调色板 2 用 50%的混合比重来混合这些数据.请参考 “图像混合控制(Graphic Blend Control)” 有详细的描述.



开关名称	端口
SB3	0x200E.D1
SB4	0x200E.D3
SB5	0x200E.D0
SB6	0x200E.D2

	D7	D6	D5	D4	D3	D2	D1	D0
0x200E			Blend2	Blend1	Pal2_Out_Sel		Pal1_Out_Sel	

Pal1_Out_Sel: 调色板 1 选择

0： 输出无作用　　1： 只有输出到 LCD

2： 只有输出到电视　　3： 输出到电视和 LCD

Pal2_Out_Sel:调色板 2 输出选择

0： 输出无作用　　1： 只有输出到 LCD

2： 只有输出到电视　　3： 输出到电视和 LCD

Blend1： 电视输出混合致能

0： 重迭　　1 :混合

Blend2 : LCD 输出混合致能

0： 重迭　　1 :混合

7.12 图像纵向的放大(Graphic Vertical Scaling)

有两个各自的纵向的放大系数对于背景 1,和背景 2 是有效的.它们是 x1, x1.5 和 x2 ,如下面的图示,由 0x2019 端口来选择.

Gain = 1　　Gain = 1.5　　Gain = 2

	D7	D6	D5	D4	D3	D2	D1	D0
0x2019					BK2_Gain		BK1_Gain	

BK2_Gain : 背景 2(BK2)纵向的放大倍率

BK1_Gain : 背景 1(BK1)纵向的放大倍率

0 : x1　　1 : x1

2 : x1.5　　3 : x2

7.13 光枪界面(Light Gun Interface) (脉冲锁定(Pulse Latch))

VT1682 提供两组的光枪界面(Light-Gun-Interface) (脉冲锁定功能(Pulse Latch function))作为双光枪应用.这两个输入的脉冲是经由 **XIOE0** 和 **XIOE1**.在 0x2023 被写入之后这个 **XIOE0** 右边的第一个上升边缘那个坐标会被锁定在 0x2024 和 0x2025.对于 **XIOE1** 也是一样,那个坐标会被锁定在 0x2026 和 0x2027. 0x2024 和 0x2026 的值会是介于 0 和 119 之间,然而 0x2025 和 0x2027 的值会是介于 0 和 126 之间.当这个坐标是(0,0)时,它代表在侦测时无输入上升边缘.

这个操作流程会是

1. 设置 **XIOE0** 为输入方式.(设置 **XIOE1** 为输入方式,如果需要的话)
2. 在 VBLANK NMI 期间读取 0x2024 和 0x2025 (0x2026 和 0x2027 如果需要的话).
3. (0x2024 *2) 是 Gun1 的物理的 Y 位置而 (0x2025 *2) 是 X 位置.
4. (0x2026 *2) 是 Gun2 的物理的 Y 位置而 (0x2027 *2) 是 X 位置.
5. 写任一个值到 0x2023.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2023	Light_Gun_Reset							
0x2024	Light_Gun1_Y							
0x2025	Light_Gun1_X							
0x2026	Light_Gun2_Y							
0x2027	Light_Gun2_X							

7.14 图像存储器寻址(Graphic Memory Access)

图像存储器包括卡通块 RAM(卡通块属性工作区) 和 VRAM (背景和调色板).它可以由 DMA 功能快速的更新或是由 CPU 用字节-方式来更新.这个 DMA 功能将不会在这里描述,请参考 “DMA” 部份.

7.14.1 卡通块 RAM(Sprite RAM)

卡通块存储器是用来储存卡通块属性的工作区. SPRAM_ADDR[10:0] 是给 CPU 寻址的卡通块 RAM 的地址.写数据到 0x2004 会写在卡通块 RAM 具有地址 SPRAM_ADDR.当 0x2004 被写入时, SPRAM_ADDR 会自动增加.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2002						SPRAM_ADDR[2:0]		
0x2003	SPRAM_ADDR[10:3]							
0x2004(W)	SPRAM_DATA[7:0]							

在卡通块 RAM 的数据格式

SPRAM_ADDR	D7	D6	D5	D4	D3	D2	D1	D0
SP*8	Vector[7:0]							
SP*8 + 1	Palette[3:0]				Vector[11:8]			
SP*8 + 2	X[7:0]							
SP*8 + 3				Layer[1:0]		Flip[1:0]		X[8]
SP*8 + 4	Y[7:0]							
SP*8 + 5						VRCH	PalSel	Y[8]
SP*8 + 6	Reserved							
SP*8 + 7	Reserved							

Note : SP 是介于 0 和 239 之间的数.

7.14.2 VRAM

VRAM 的寻址跟卡通块 RAM 一样.写数据到 0x2007 会写在 VRAM 具有地址 VRAM_ADDR. 当 0x2007 被写入时,VRAM_ADDR 会自动增加.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2005	VRAM_ADDR[7:0]							
0x2006	VRAM_ADDR[15:8]							
0x2007(W)	VRAM_DATA[7:0]							

VRAM 地址映像(VRAM address map)

7.15 特殊效果(Special Effect)

7.15.1 掘取颜色(Dig Color)

在 VT1682 的颜色调色板提供掘取颜色功能,那个功能会从调色板 1 移除颜色去显示相关的点到调色板 2. 同样地也可以从调色板 2 移除颜色去显示相关的点到调色板 1. DG 标志会在调色板之后表现这个掘取功能在图像上.不同于透明颜色, DG 标志是去移除所有层次的颜色和显示这些颜色在其他的调色板.透明颜色是去显示这个具有最低层次的颜色在同一个调色板.当这个 DG 为 “0”时,这个点会被显示具有颜色[R, G, B] 和相关的索引.当 DG 为 “1”时,这个显示颜色会是[R, G, B] 或是这个从其他调色板颜色的点颜色.参考 t “调色板选择(Palette Select)” 部份.如果没有显示点来自其他的调色板,这个颜色[R, G, B] 会被显示,不然的话,这个来自其他调色板颜色的点颜色会被显示.

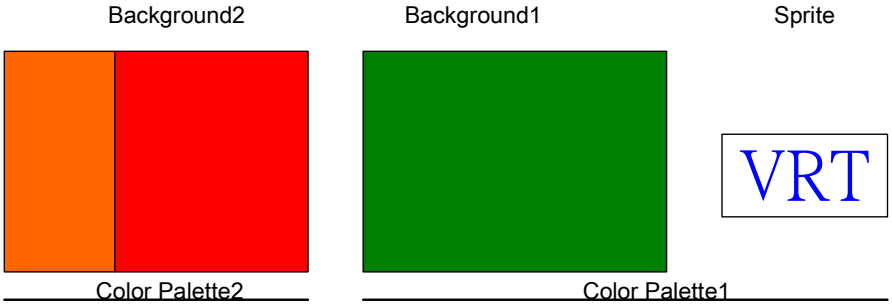
低字节(Low Byte):

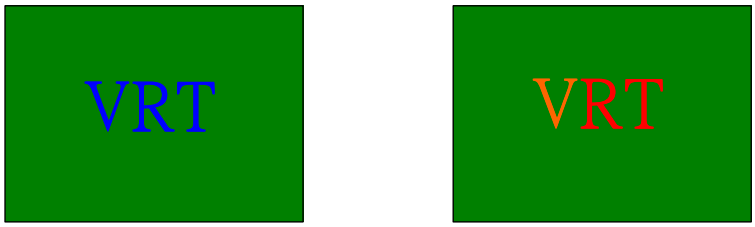
D7 – D5	D4 – D0
绿[2:0]	蓝[4:0]

高字节(High Byte):

D7	D6 – D2	D1 – D0
DG	红[4:0]	绿[4:3]

下面的图示是一个描述掘取颜色效果的例子.假定那个背景 1(BK1)和卡通块(Sprite)是来自于颜色调色板 1(Color Palette1),而背景 2(BK2)来自于颜色调色板 2(Color Palette2).当这个在颜色调色板 1 的卡通块蓝色掘取颜色标志为 0 时,这个颜色调色板 1 的输出会是蓝色的 “VRT”.当这个掘取颜色标志为 1 时,这个“VRT” 会变成颜色调色板 2 的颜色(BK2 在这个例子).





DG = 0　　DG = 1

7.15.2 混合效果(Blending Effect)

混合效果是颜色调色板 1 和颜色调色板 2 的颜色混合.这个混合的比率是 50% 的颜色调色板 1 和 50% 的颜色调色板 2.有两种混合的效果,一个是给电视输出,另一个是给 LCD.

	D7	D6	D5	D4	D3	D2	D1	D0
0x200E			Blend2	Blend1	Pal2_Out_Sel		Pal1_Out_Sel	

Pal1_Out_Sel: 调色板 1 选择

0： 输出无作用　　1： 只有输出到 LCD

2： 只有输出到电视　　3： 输出到电视和 LCD

Pal2_Out_Sel:调色板 2 输出选择

0： 输出无作用　　1： 只有输出到 LCD

2： 只有输出到电视　　3： 输出到电视和 LCD

Blend1： 电视输出混合致能

0： 重迭　　1 :混合

Blend2 : LCD 输出混合致能

0： 重迭　　1 :混合

7.16 纵向的空白(Vertical Blanking) (NMI)

主 CPU 和 Sound CPU 的 NMI,两者都是在纵向的空白处.去使 Sound CPU 或是主 CPU 的 NMI 致能, 必须要设置在 0x2000 内的 NMI_EN.当这个 NMI 无作用时,主 CPU 会读取在 0x2001 内的 VBLANK 标志来分辨识这个纵向的空白期间.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2000				Capture	SLAVE	---	---	NMI_EN
0x2001(R)	VBLANK	SP_ERR						

NMI_EN : VBLANK NMI 致能控制

0： 无作用 1： 致能

VBLANK： 纵向的空白处有效期间和读取来清除 NMI

0： 非-VBLANK 期间 1 : VBLANK 期间

7.17 图像显示画面大小(Graphic Display Screen Size)

在正常的操作之下,这个显示图像的分辨率是 256 x 240 点.当作一些特殊的横向的滚动条控制时, VT1682 提供另一种显示画面大小为 248 x 240 点.这个最左边的 8 点, 1st ~ 8th 将不会被显示在画面上.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2001(W)					EXT_CLK_DIV		SP_INI	BK_INI

SP_INI： 卡通块显示画面大小

BK_INI： 背景显示画面大小

0 : 256 x 240 点 1 : 248 x 240 点

8. I/O

在 VT1682 有 56 个 I/O 可以使用, 40 个由主 CPU 来控制,其他的 16 个是给 Sound CPU 用的.这 40 个 I/O 是 IOA[3:0], IOB[3:0], IOC[3:0], IOD[3:0], IOE[3:0], IOF[3:0], UIOA[7:0] 和 UIOB[7:0]. 它们每一个都有不同的属性,如下面的表格所示.除此之外,所有的这些 I/O 端口是共享的脚位,它们可以是 GPIO 或是 LCD, UART, SPI, I2C, …… 界面.

	输出 High	输出 Low	输入 Floating	Input w/ Pull High 电阻	Input w/ Pull Low 电阻	共享功能
IOA	O	O	X	O	O	LCD
IOB	O	O	X	O	O	LCD
IOC	O	O	X	O	O	LCD / UART
IOD	O	O	X	O	O	SPI
IOE	O	O	O	X	X	LCD / GunPort
IOF	O	O	O	X	X	ADC / I2C
UIOA	O	O	O	O	O	LCD / CCIR
UIOB	O	O	O	O	O	CSB / Ext IRQ / JoyStick

8.1 IOA

IOA 是和LCD 界面共享.当这个LCD 控制器被使用时,这个IOA_ENB 和 IOA_OE 必须被置为 1.在这个 GPIO(IOA_ENB = 0) 方式, IOA 可以是输出高电平(High),输出低电平(Low),有拉到高电平(pull-high)电阻的输入或是有拉到低电平(pull-low)电阻的输入方式的其中之一,如下面的表格所列:

XIOA[0,1,2,3]	IOAOE = 1	IOAOE = 0
IOAENB = 0	OUTPUT IOA_O	IOA_O[x] = 0 : INPUT w/ pull-low 电阻 IOA_O[x] = 1 : INPUT w/ pull-high 电阻
IOAENB = 1	OUTPUT LCD signals	IOA_O[x] = 0 : INPUT w/ pull-low 电阻 IOA_O[x] = 1 : INPUT w/ pull-high 电阻

	D3	D2	D1	D0
0x210D(W)	IOPBENB	IOBOE	IOAENB	IOAOE
0x210E(W)	IOA_O[3:0]			
0x210E(R)	IOA_I[3:0]			

8.2 IOB

IOB 是和 LCD 界面共享. 当这个 LCD 控制器被使用时, 这个 IOB_ENB 和 IOB_OE 必须被置为 1.在这个 GPIO(IOB_ENB = 0) 方式, IOB 可以是输出高电平(High),输出低电平(Low),有拉到高电平(pull-high)电阻的输入或是有拉到低电平(pull-low)电阻的输入方式的其中之一,如下面的表格所列:

XIOB[0,2,3]	IOBOE = 1	IOBOE = 0
IOBENB = 0	OUTPUT IOB_O	IOB_O[x] = 0 : INPUT w/ pull-low 电阻 IOB_O[x] = 1 : INPUT w/ pull-high 电阻
IOBENB = 1	OUTPUT LCD	IOB_O[x] = 0 : INPUT w/ pull-low 电阻 IOB_O[x] = 1 : INPUT w/ pull-high 电阻

XIOB[1]	IOBOE = 1	IOBOE = 0
IOBENB = 0	VCOM_OEN=0, OUTPUT IOB_O	IOB_O[1] = 0 : INPUT w/ pull-low 电阻 IOB_O[1] = 1 : INPUT w/ pull-high 电阻
IOBENB = 1	VCOM_OEN=0, OUTPUT LCD VCOM_OEN=1, INPUT LCD	IOB_O[1] = 0 : INPUT w/ pull-low 电阻 IOB_O[1] = 1 : INPUT w/ pull-high 电阻

Note : VCOM_OEN : 0x2022.D5

	D7	D6	D5	D4	D3	D2	D1	D0
0x210D(W)					IOPBENB	IOBOE		
0x210E(W)	IOB_O[3:0]							
0x210E(R)	IOB_I[3:0]							

8.3 IOC

IOC 是和 LCD 及 UART 界面共享. 当这个 LCD 控制器是在 LTPS 或是 ANALOG 方式时或是 UART 界面被使用时, 这个 IOC_ENB 和 IOC_OE 必须被置为 1.在这个 GPIO(IOC_ENB = 0) 方式, IOC 可以是输出高电平(High),输出低电平(Low),有拉到高电平(pull-high)电阻的输入或是有拉到低电平(pull-low)电阻的输入方式的其中之一,如下面的表格所列:

XIOC[0,1,2]	IOCOE = 1	IOCOE = 0
IOCENB = 0	OUTPUT IOC_O	IOC_O[x] = 0 : INPUT w/ pull-low 电阻 IOC_O[x] = 1 : INPUT w/ pull-high 电阻
IOCENB = 1	OUTPUT LCD/UART	IOC_O[x] = 0 : INPUT w/ pull-low 电阻 IOC_O[x] = 1 : INPUT w/ pull-high 电阻

XIOC[3]	IOCOE = 1	IOCOE = 0
IOCENB = 0	UART_ON=0, OUTPUT IOC_O	IOC_O[3] = 0 : INPUT w/ pull-low 电阻 IOC_O[3] = 1 : INPUT w/ pull-high 电阻
IOCENB = 1	UART_ON=1, INPUT UART:RX	IOC_O[3] = 0 : INPUT w/ pull-low 电阻 IOC_O[3] = 1 : INPUT w/ pull-high 电阻

	D7	D6	D5	D4	D3	D2	D1	D0
0x210D(W)			IOCENB	IOCOE				
0x210F(W)					IOC_O[3:0]			
0x210F(R)					IOC_I[3:0]			

8.4 IOD

IOD 是和 SPI 界面共享. 当这个 SPI 界面是在 Master 方式时, 这个 IOD_ENB 和 IOD_OE 必须被置为 1. 当这个 SPI 界面是在 Slave 方式时, 这个 IOD_OE 必须被置为 0.在这个 GPIO(IOD_ENB = 0)方式, IOD 可以是输出高电平(High),输出低电平(Low),有拉到高电平(pull-high)电阻的输入或是有拉到低电平(pull-low)电阻的输入方式的其中之一,如下面的表格所列:

XIOD[0,3]	IODOE = 1	IODOE =0
IODENB = 0	OUTPUT IOD_O	IOD_O[x] = 0 : INPUT w/ pull-low 电阻 IOD_O[x] = 1 : INPUT w/ pull-high 电阻
IODENB = 1	OUTPUT SPI	IOD_O[x] = 0 : INPUT w/ pull-low 电阻 IOD_O[x] = 1 : INPUT w/ pull-high 电阻

XIOD[1]	IODOE = 1	IODOE = 0
IODENB = 0	SPI_ON=0, OUTPUT IOD_O SPI_ON=1, INPUT:SPI:DI	IOD_O[1] = 0 : INPUT w/ pull-low 电阻 IOD_O[1] = 1 : INPUT w/ pull-high 电阻
IODENB = 1	SPI_ON=0, OUTPUT IOD_O SPI_ON=1, INPUT:SPI:DI	IOD_O[1] = 0 : INPUT w/ pull-low 电阻 IOD_O[1] = 1 : INPUT w/ pull-high 电阻

XIOD[2]	IODOE = 1	IODOE = 0
IODENB = 0	OUTPUT IOD_O	IOD_O[2] = 0 : INPUT w/ pull-low 电阻 IOD_O[2] = 1 : INPUT w/ pull-high 电阻
IODENB = 1	OUPUT:SPI:DO	IOD_O[2] = 0 : INPUT w/ pull-low 电阻 IOD_O[2] = 1 : INPUT w/ pull-high 电阻

	D7	D6	D5	D4	D3	D2	D1	D0
0x210D(W)	IODENB	IODOEN						
0x210F(W)	IOD_O[3:0]							
0x210F(R)	IOD_I[3:0]							

8.5 IOE

IOE 是和 LCD DAC 输出及光枪界面共享. 当这个 SPI 界面是在 Master 方式时, 这个 IOD_ENB 和 IOD_OE 必须被置为 1. 当这个 LCD 控制器是在 LTPS 或是 ANALOG 方式或是 LCD DAC 时,光枪功能被使用时, 这个 IOE_OE 必须被置为 0.当这个 ADC 麦克风方式被使用时, 在 0x211E 的 IOEOE3 必须被置为 0,不然的话 IOE 可以是输出高电平(High),输出低电平(Low),输入漂浮(floating)方式的其中之一,如下面的表格所列:

	IOEOE = 1	IOEOE = 0
XIOE[0,1,2]	LCDACEN=0, OUTPUT IOE_O LCDACEN =1, OUTPUT LCDAC	LCDACEN=0, INPUT (12K ohms pull-down resistor) LCDACEN =1, OUTPUT LCDAC

	IOEOE3 = 1	IOEOE3 = 0
XIOE[3]	OUTPUT IOE_O	INPUT floating

	IOEOE = 1, LCDACEN = 0	IOEOE = 0, LCDACEN = 0	LCDACEN = 1
XIOE0	OUTPUT / IOE_O[0]	INPUT / IOE_I[0] /GunPort[0]	OUTPUT / LCD:VR
XIOE1	OUTPUT / IOE_O[1]	INPUT / IOE_I[1] / GunPort[1]	OUTPUT / LCD:VG
XIOE2	OUTPUT / IOE_O[2]	INPUT / IOE_I[2]	OUTPUT / LCD:VB

	IOEOE3 = 1,	IOEOE3 = 0,
XIOE3	OUTPUT / IOE_O[3]	INPUT / IOE_I[3]

	D7	D6	D5	D4	D3	D2	D1	D0
0x214C						----	----	IOEOE
0x214D(W)					IOE_O[3:0]			
0x214D®					IOE_I[3: 0]			
0x211E(W)				IOEOE3				

8.6 IOF

IOF 是和 ADC 输入及 IIC 界面共享. 当这个 ADC 被使用时, 这些相关的 IOF_OE[x] 必须被置为 0. 当 IIC 被使用时, IOFENB 必须被置为 1. 不然的话, IOF 可以是输出高电平(High),输出低电平(Low),输入漂浮(floating)的方式的其中之一,如下面的表格所列:

	IOF_OE[X] = 1	IOF_OE[X] = 0
XIOF0	OUTPUT:IOF_O[0]	INPUT FLOAT / ADC0
XIOF1	OUTPUTLIOF_O[1]	INPUT FLOAT / ADC1

	IOF_OE[2] = 1	IOF_OE[2] = 0
XIOF2	IOF_ENB=0, OUTPUT:IOF_O[2] IOF_ENB=1, OUTPUT: IIC:SCK	INPUT FLOAT / ADC2

	IOF_OE[3] = 1	IOF_OE[3] = 0
XIOF3	IOF_ENB=0, OUTPUT:IOF_O[3] IOF_ENB=1, INPUT: IIC:SDA	INPUT FLOAT / ADC3

	D7	D6	D5	D4	D3	D2	D1	D0
0x214C					IOFENB			
0x214D(R)	IOF[3:0]				IOE[3:0]			
0x211E(W)					IOFOE3	IOFOE2	IOFOE1	IOFOE0

8.7 UIOA

UIOA 是 bit-方式属性控制,每一个 UIOA 有一个各自的方向性(IN / OUT)和属性(拉到高电平(pull high)或是拉到低电平(pull low),漂浮(floating)) 参数,如下面的表格所列. XUIOA 是跟 LCD 和 CCIR 界面共享. 当这个 CCIR 界面有作用时, UIOA_DIR 必须置为输入漂浮的方式.

DIR	ATTR	DATA_OUT	Description
0	0	0	Input floating
0	0	1	Input floating
0	1	0	Input with pull-low resistor
0	1	1	Input with pull-high resistor
1	0	0	Output low
1	0	1	Output high
1	1	0	Output low
1	1	1	Output high

	D7	D6	D5	D4	D3	D2	D1	D0
0x2129(W)	UIOA_DATA_OUT							
0x2129(R)	UIOA_DATA_IN							
0x212A(W)	UIOA_DIR							
0x212B	UIOA_ATTR							
0x2148(W)								UIOA_MODE

8.7.1 UIOA 共享功能(UIOA Shared Function)

输出方式(OUTPUT MODE)

	UIOA_MODE[0]=0	UIOA_MODE[0]=1
XUIOA0	OUTPUT:UIOA_DATA_OUT[0]	OUTPUT:LCD:D0
XUIOA1	OUTPUT:UIOA_DATA_OUT[1]	OUTPUT:LCD:D1
XUIOA2	OUTPUT:UIOA_DATA_OUT[2]	OUTPUT:LCD:D2
XUIOA3	OUTPUT:UIOA_DATA_OUT[3]	OUTPUT:LCD:D3

XUIOA4	OUTPUT:UIOA_DATA_OUT[4]	OUTPUT:LCD:D4

	UIOA_MODE[1]=0	UIOA_MODE[1]=1
XUIOA5	OUTPUT:UIOA_DATA_OUT[5]	OUTPUT:LCD:D5*
XUIOA6	OUTPUT:UIOA_DATA_OUT[6]	OUTPUT:LCD:D6*
XUIOA7	OUTPUT:UIOA_DATA_OUT[7]	OUTPUT:LCD:D7*

*LCD 的详细描述，请参考 LCD 部份.

输入方式(INPUT MODE)

	Function
XUIOA0	UIOA_DATA_IN[0] / CCIR_D0
XUIOA1	UIOA_DATA_IN[1] / CCIR_D1
XUIOA2	UIOA_DATA_IN[2] / CCIR_D2
XUIOA3	UIOA_DATA_IN[3] / CCIR_D3
XUIOA4	UIOA_DATA_IN[4] / CCIR_D4
XUIOA5	UIOA_DATA_IN[5] / CCIR_D5
XUIOA6	UIOA_DATA_IN[6] / CCIR_D6
XUIOA7	UIOA_DATA_IN[7] / CCIR_D7

8.8 UIOB

UIOB 是 bit-方式属性控制,每一个 UIOB 有一个各自的方向性(IN / OUT)和属性(拉到高电平(pull high)或是拉到低电平(pull low),漂浮(floating)) 参数,如下面的表格所列. XUIOB 是跟外部的 IRQ,LCD 和 CCIR 界面共享.当这个 CCIR 界面有作用时, UIOB_DIR [2:0] 必须置为输入漂浮的方式.当外部的 IRQ 有作用时, XUIOB3 必须被置为带有拉到高电平(pull high)电阻的输入方式.

0x2148(W)	UIOB_SEL[7:3]		
0x2149(W)	UIOB_DATA_OUT		
0x2149®	UIOB_DATA_IN		
0x214A	UIOB_DIRECTION		
0x214B	UIOB_ATTRIBUTE		

8.8.1 UIOB 共享功能(UIOB Shared Function)

输出(OUTPUT)

	UIOA_MODE[1]=0	UIOA_MODE[1]=1
XUIOB0	OUTPUT:UIOB_DATA_OUT[0]	OUTPUT:LCD:D8*
XUIOB1	OUTPUT:UIOB_DATA_OUT[1]	OUTPUT:LCD:D9*

	UIOB_MODE[3]=0	UIOB_MODE[3]=1
XUIOB3	OUTPUT:UIOB_DATA_OUT[3]	OUTPUT:JOY_CK

	UIOB_MODE[4]=0	UIOB_MODE[4]=1
XUIOB4	OUTPUT:UIOB_DATA_OUT[4]	OUTPUT:JOY_CK2

	UIOB_MODE[5]=0	UIOB_MODE[5]=1
XUIOB5	OUTPUT:UIOB_DATA_OUT[5]	OUTPUT:CSYNC*

	UIOB_MODE[7]=0	UIOB_MODE[7]=1
XUIOB7	OUTPUT:UIOB_DATA_OUT[7]	OUTPUT:ROMCSB2

8.9 IO 共享脚位表(IO Shared Pin Table)

PIN NAME	OUTPUT	INPUT
XIOA0	LCD	----
XIOA1	LCD	----
XIOA2	LCD	----
XIOA3	LCD	----
XIOB0	LCD	----
XIOB1	LCD	----
XIOB2	LCD	----
XIOB3	LCD	----
XIOC0	LCD	----
XIOC1	LCD	----
XIOC2	UART_TX	----
XIOC3	----	UART_RX
XIOD0	SPI_CKO	SPI_CKI
XIOD1	----	SPI_DI
XIOD2	SPI_DO	----
XIOD3	SPI_CSBO	SPI_CSBI
XIOE0	VR	GunPort
XIOE1	VG	GunPort
XIOE2	VB	----
XIOE3	----	----
XIOF0	----	ADC0
XIOF1	----	ADC1
XIOF2	IIC_SCK	ADC2
XIOF3	IIC_SDA	ADC3
XUIOA0	LCD_D0 /	CCIR_D0
XUIOA1	LCD_D1 / CSTN_D0	CCIR_D1
XUIOA2	LCD_D2 / CSTN_D1	CCIR_D2
XUIOA3	LCD_D3 / CSTN_D2	CCIR_D3
XUIOA4	LCD_D4 / CSTN_D3	CCIR_D4
XUIOA5	LCD_D0 / CSTN_CP	CCIR_D5
XUIOA6	LCD_D1 / CSTN_LP	CCIR_D6

XUIOA7	LCD_D2 / CSTN_FP	CCIR_D7
XUIOB0	LCD_D3 / CSTN_FM	CCIR_CK
XUIOB1	LCD_D4 /	CCIR_HS
XUIOB2	----	CCIR_VS
XUIOB3	JOY_CK	EXT_IRQ
XUIOB4	JOY_CK2	----
XUIOB5	CSYNC	----
XUIOB6	----	----
XUIOB7	ROMCSB2	----

9. DMA

VT1682 提供 4 种存储器直接寻址(DMA)路径来加速数据的传送.它们是外部的存储器到程序的 RAM, 外部的存储器到程序的 VRAM,程序的 RAM 到 VRAM 和程序的 RAM 到外部的存储器.程序的 RAM 包括 4K 字节的主 CPU 的专用的 PRAM 和 4K 字节的共享 RAM,而 VRAM 包括卡通块 RAM,颜色调色板和背景的 VRAM.它们是由寄存器 0x2122 ~ 0x2128 来控制,如下面的表格所示.这些 DMA_DT_Addr, DMA_SR_Addr 和 DMA_SR_Bank 是以"字节(byte)"为基础,但是 DMA_Number 是以"字(Word)"为基础"(两个字节(Bytes)). DMA 的传送是由写 DMA_Number 所发送.这个 DMA 它的目的地是 VRAM 一直到这个纵向的空白(VBLANK) 期间时将不会被开始.换句话说,如果你在非 VBLANK 期间发送一个 VRAM DMA,这个 DMA 将会无作用一直到 VBLANK 开始时.在 DMA 期间这个 CPU 将会停止.这个 DMA_Status 是 DMA 状态标志作为你追踪 DMA 的操作.当你传送数据到 VRAM 时,不只有这里的这些 DMA 寄存器你必须去写它们而且这些寄存器 0x2002, 0x2003, 0x2005 和 0x2006 也要.当这个 DMA 的目的地是卡通块(Sprite) RAM 时,寄存器端口 0x2002 和 0x2003 是被使用来定义这个目的地地址.当这个 DMA 的目的地是 VRAM 时,寄存器端口 0x2005 和 0x2006 必须被定义.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2122	DMA_DT_Addr[7:0]							
0x2123	DMA_DT_Addr[15:8]							
0x2124	DMA_SR_AddrT[7:0]							
0x2125	DMA_SR_Addr[15:8]							
0x2126	DMA_SR_Bank[22:15]							
0x2127(W)	DMA_Number							
0x2127(R)								DMA_Status
0x2128(W)							DMA_SR_Bank[24:23]	

DMA_DT_Addr : DMA 目的地地址.

DMA_SR_Addr : DMA 来源地址.

注释 : DMA_SR_Addr[0] 和 DMA_DT_Addr[0] 必须为 0.

DMA_Number : DMA 传送"字(word)"的数目.

DMA_Status : DMA 状态标志.

0 : DMA 准备就续(ready) 1 : DMA 忙碌的(busy.)

DMA Source	DMA Target	来源(Source) 开始地址(Start Address)	目标(Target) 地址(Address)
EXT	PRAM	{DMA_SR_Bank, DMA_SR_Addr[14:0]} (*1)	DMA_DT_Addr
EXT	SpriteRAM	{DMA_SR_Bank, DMA_SR_Addr[14:0]} (*1)	DMA_DT_Addr(*3)
EXT	VRAM	{DMA_SR_Bank, DMA_SR_Addr[14:0]} (*1)	DMA_DT_Addr(*4)
PRAM	SpriteRAM	DMA_SR_Addr[14:0] (*2)	DMA_DT_Addr(*3)
PRAM	VRAM	DMA_SR_Addr[14:0] (*2)	DMA_DT_Addr(*4)
PRAM	PRAM	DMA_SR_Addr[14:0] (*2)	DMA_DT_Addr
PRAM	EXT	DMA_SR_Addr[14:0] (*2)	{DMA_SR_Bank, DMA_DT_Addr[14:0]}(*5)

*1 : DMA_SR_Addr[15] = 1.

*2 : DMA_SR_Addr[15] = 0.

*3 : DMA_DT_Addr = 0x2004.

*4 : DMA_DT_Addr = 0x2007.

*5 : DMA_DT_Addr[15] = 1.

10. 省电功能(POWER SAVING)

为了节省电源损耗,在 VT1682 大部份的模块都是可以分开的来关掉,包括 Sound CPU, SPI, UART, 电视编码器(TV encoder), LCD 控制器, LVD 模块, Video DAC, Audio DAC, LCD DAC, PLL 和 ADC.在默认的方式下,所有这些模块都是无作用的.

0x2106	---	---	SCPURN	SCPU_ON	SPI_ON	UART_ON	TV_ON	LCD_ON
0x211D	LVDEN			VDAC_EN	ADAC_EN	PLL_EN	LCDACEN	---

LCD_ON : LCD 控制器模块致能控制.

TV_ON : 电视编码器(TV encoder) 模块致能控制

UART_ON : UART 致能控制

SPI_ON : SPI 模块致能控制

LVDEN : 低电压检测(Low Voltage Detect)模块致能控制

VDAC_EN : Video DAC 模块致能控制

ADAC_EN, :Audio DAC 模块致能控制

PLL_EN : 相位锁定回路(Phase Lock Loop)模块致能控制

LCDDACEN : 模拟(analog)LCD DAC 模块致能控制

SCPU_ON : Sound CPU 致能控制

0 : 无作用 1 : 致能

SCPRN : Sound CPU Reset 控制

0 : Reset Sound CPU 1 : 正常操作

11. 中断(INTERRUPT)

在 VT1682 有 6 个 IRQ 来源和 5 个 IRQ 序号(vectors).这些 IRQ 来源是外部的 IRQ,定时器(Timer) IRQ, SPCU IRQ, UART IRQ 和 SPI IRQ.它们的 IRQ vectors 如下面的表格所示.当在 IRQ 发生时期当中,这个伺服的优先级会是 Ext_IRQ > Timer_IRQ > SCPU_IRQ > UART_IRQ > SPI_IRQ.

序号名称(Vector Name)	序号地址(vector address)
NMI	0x7FFFA, 0x7FFFB
Ext_IRQ	0x7FFFE, 0x7FFFF
Timer_IRQ	0x7FFF8, 0x7FFF9
SCPU_IRQ	0x7FFF6, 0x7FFF7
UART_IRQ	0x7FFF4, 0x7FFF5
SPI_IRQ	0x7FFF2, 0x7FFF3

11.1 非罩幕式中断(Non Maskable Interrupt (NMI))

这个 NMI 来源是这个图像纵向的空白(VBLANK).这些详细的描述请参考 “7.17 纵向的空白(Vertical Blanking)(NMI)” 部份.

11.2 外部的 IRQ(External IRQ)

外部的 IRQ 是来自于 XUIOB3 的下降边缘触发.在外部的 IRQ 被致能之前,记得去设置这个 XUIOB3 为带有 Pull-high 电阻的输入方式.当这个在 0x2121 的 Ext_MSK 被置为 1 时,在 XUIOB3 的下降边缘会使 CPU 进入这个 IRQ 伺服惯例定义在 0x7FFFE 和 0x7FFFF.在伺服惯例写“1” 到 Ext_MSK 会清除这个 IRQ 标志去等待下一个触发.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2121	--	--	--	SPI_MSK	UART_MSK	SPU_MSK	IRQ1_MSK	Ext_MSK

Ext_MSK: 外部的 IRQ 致能控制

0 : IRQ 致能　　1 : IRQ 无作用

写 “1” 到 Ext_MSK 会清除 IRQ 标志.

11.3 定时器 IRQ(Timer IRQ)

定时器 IRQ(Timer IRQ)的 IRQ 序号(vector), 0x7FFF8 和 0x7FFF9,而且这个 IRQ 是可以罩幕的,是由 0x2121 的 IRQ1_MSK 控制.

请参考 **“6.2 定时器(Timer)”** 部份有详细的操作.去使这个 Timer IRQ 致能,必须要把在 0x2102 的 TMR_IRQ 和在 0x2121 的 IRQ1_MSK 两个都先置为 “1” .当这个定时器 IRQ 发生时, 写 0x2103 的 (Timer_IRQ_Clear) 会清除这个现在的定时器 IRQ 标志去等待下一个定时器 IRQ.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2102							TMR_IRQ	TMR_EN
0x2121	--	--	--	SPI_MSK	UART_MSK	SPU_MSK	IRQ1_MSK	Ext_MSK
0x2103	Timer_IRQ_Clear							

TMR_IR: 定时器 IRQ 致能控制.

0 : 无作用　　1 : 致能

Timer_IRQ_Clear : 定时器 IRQ 清除控制, 写入任一数据去清除定时器 IRQ.

11.4 Sound CPU IRQ

IRQ 是被使用来当作 Sound CPU 和主 CPU 之间的通讯.它们两个都可以在 IRQ 传送给彼此.

11.4.1 接收 Sound CPU IRQ

当 Sound CPU 送 IRQ 到主 CPU 时, 如果在 0x2121 的 SPU_MSK 被致能,主 CPU 将进入伺服惯例, 由 0x7FFF6 和 0x7FFF7 来定义. 请参考“**14.3.2.2 与主 CPU 通讯**” 部份作为 Sound CPU IRQ.在 Sound CPU IRQ 伺服惯例, 写“1” 到 SPU_MSK 将会清除 IRQ 标志一直到下一个 Sound CPU IRQ.

11.4.2 传送 Sound CPU IRQ

当主 CPU 正准备有来自于 Sound CPU 的需求时,它会送一个 IRQ 去中断 Sound CPU.在 0x211C 的 SCPU_IRQ 写一个正的脉冲产生这个 IRQ 脉冲.

	D7	D6	D5	D4	D3	D2	D1	D0
0x211C(W)				SCPUIRQ				

11.5 UART IRQ

有两个 IRQ 来源来自于 UART,一个是 RX (接收) IRQ 和另一个是 TX (传送) IRQ.在 0x2119 的 RXIRQEn, TXIRQEn 控制这两个 IRQ. UART IRQ 也是可以罩幕的是由在 0x2121 的 UART_MSK 控制. 只有当 UART_MSK 为 "1" 时和 RXIRQEn 和 TXIRQEn 其中之一被致能时,这个 UART_IRQ 会发生.只要从 0x211B 去读取 TX-Status 和 RX_Status 的标志就可以区分来自于 RX 和 TX 的 IRQ.请参考"**5.3 UART 界面**" 作详细的 UART 设置.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2119-W	--	CarriEn	UARTEN	TXIRQEn	RXIRQEn	ParityEn	OddEven	9bitmode
0x211B-R	--	--	RxError	TX_Status	RX_Status	ParityErr	--	--
0x2121	--	--	--	SPI_MSK	UART_MSK	SPU_MSK	IRQ1_MSK	Ext_MSK

11.6 SPI IRQ

SPI IRQ 是可以罩幕的,是由在 0x2121 的 SPI_MSK 控制.当这个 SPI 准备就绪去传送或是准备就绪去接收时,这个 IRQ 会发生.

12. 外部的存储器控制(EXTERNAL MEMORY CONTROL)

外部的存储器总线, XA[23:0], XD[15:0], XROMCSB, XRAMCSB, XRAMRWB, XROMOEB 当做特殊应用时是可以用过程控制去进入三态(tri-state)方式的.外部的存储器寻址时间有两种型式, 180ns 和 80ns 在 VT1682 都是容许的.为了节省应用时的 BOM 费用,有多达三个外部的存储器 CSB 是可以利用的.

12.1 外部的存储器总线寻址时间(External Memory Bus Access Time)

在默认的方式,CPU 的效能会被图像所妨碍.更高的图像质量会导致更低的 CPU 效能.为了解决这个问题, VT1682 容许使用者去更换更快速的外部的存储器组件(SRAM / FLASH / ROM)来改善 CPU 的效能.除此之外, VT1682 可以改变存储器总线 (XA, XD, XROMCSB, XRAMCSB, XRAMRWB 和 XROMOEB) 进入三态(tri-state)去允许外部的组件对这个存储器寻址.在这个期间, CPU 可以在 PRAM 工作而且不用关掉显示画面并且显示简单的图像在 VRAM.

总线寻址频率(Bus Access Frequency)

两种类型的寻址速度可利用在 VT1682,由在 0x2105 的 "Double" 来控制.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2105(W)	---	COMR6	TV_SYS_SE:[1:0]		CCIR_SEL	Double	ROM_SEL	PRAM

Double	外部的存储器寻址时间(Access Time)
0	180 ns
1	80 ns

12.2 总线三态控制(Bus Tri-state Control)

VT1682 可以隔离系统总线 XA, XD, XROMCSB, XRAMCSB, XRAMRWB 和 XROMOEB.在这个方式下,外部的组件是被允许来对 VT1682 的外部存储器进行寻址的.当总线是在三态(tri-sate)时,在 XA, XROMCSB, XRAMCSB, XRAMRWB 和 XROMOEB 有一个 50K ohm 的拉到高电平(Pull-high)电阻.

	D7	D6	D5	D4	D3	D2	D1	D0
0x210B	TSYNEN	PQ2EN	BUSTRI	CS_Control[1:0]		Program_Bank0_select		

0： 正常操作　　1 : XA 三态(tri-state)方式

12.3 外部的存储器蕊片选择信号控制(External Memory Chip Select Signal Control)

VT1682 可以提供达 3 个 CSB 的控制脚给外部的存储器,它们是 XROMCSB, XRAMCSB 和 XUIOB7. 有四种存储器映像(memory map)方式给这三支脚,它们是由在 0x210B 的 CS_Control 来控制.当这个 XUIOB7 是被使用当做外部存储器的 CSB 时,记得去更改它的属性,设置 UIOB_SEL[7] = 1 和 UIOB_DIR[7] = 1.

	D7	D6	D5	D4	D3	D2	D1	D0
0x210B	TSYNEN	PQ2EN	BUSTRI	CS_Control[1:0]		Program_Bank0_select		

CS_Control	XROMCSB	XRAMCSB	XUIOB7
0	$000000 ~ $FFFFFF	----	----
1	$000000 ~ $7FFFFF	$800000 ~ $FFFFFF	----
2	$000000 ~ $3FFFFF	$40000 ~ $7FFFFF	$800000 ~ $FFFFFF
3	$000000 ~ $1FFFFF	$200000 ~ $3FFFFF	$400000 ~ $7FFFFF

13. 游戏杆通信协议(JOYSTICK PROTOCOL)

VT1682 可以提供达到两组的三条线游戏杆通信协议.每一组包括 SCK, SDO 和 SDI.这个 SCK 时钟会输出在 XUIOB4 和 XUIOB5.当这个 0x2129 端口被读取时,一个 200ns 宽的正脉冲会被产生在 XUIOB4. 同样在 0x212A 端口被读取时,一个 200ns 宽的正脉冲会被产生在 XUIOB5. XUIOB4 和 XUIOB5 的设置如下面表格所列. SDI 和 SDO 会是任何一个 UIOA 或是除了 UIOB[5:4]之外的任何一个 UIOB.

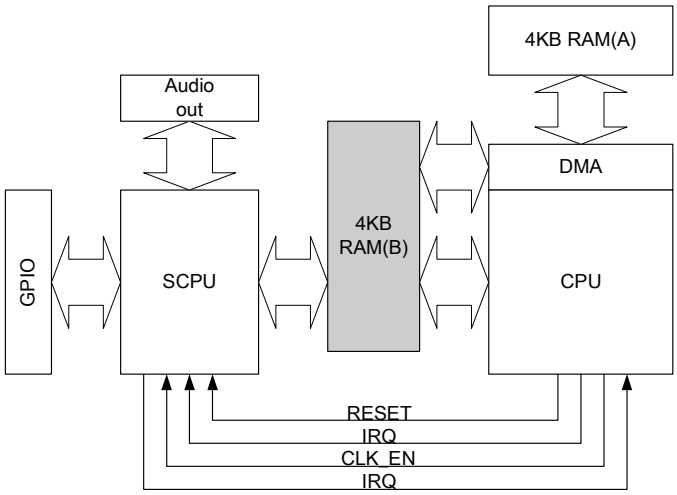
0x2129®	送_JOY_CLK		
0x2129®	UIOA_DATA_IN		
0x2148(W)	UIOB_SEL[7:3]		UIOA_MODE

脚位	信号名称	设置
XUIOB4	JOY_CK	UIOB_SEL[4] = 1, UIOB_DIR[4] = 1
XUIOB5	JOY_CK2	UIOB_SEL[5] = 1, UIOB_DIR[5] = 1

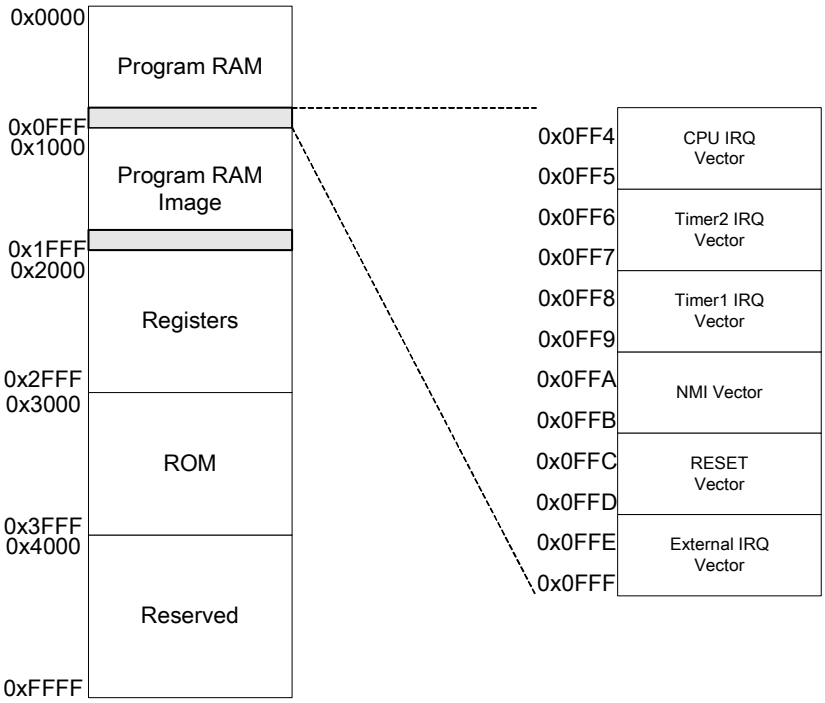
14. SOUND CPU

在 VT1682 有两个 6502 CPU，一个是主 CPU,另一个是 Sound CPU (SCPU). SCPU 于 NTSC 系统下操作在 21.4772MHz 和于 PAL 系统下操作在 26.6027MHz. SCPU 和主 CPU 共享 4K 字节的 SRAM.主 CPU 和 SCPU 可以透过这个共享的 SRAM 或是 IRQ 信号来通讯.

14.1 结构图(Block Diagram)



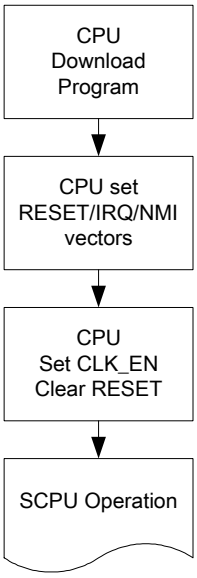
14.2 存储器映射(Memory Map)



14.3 操作程序(Operation Procedure)

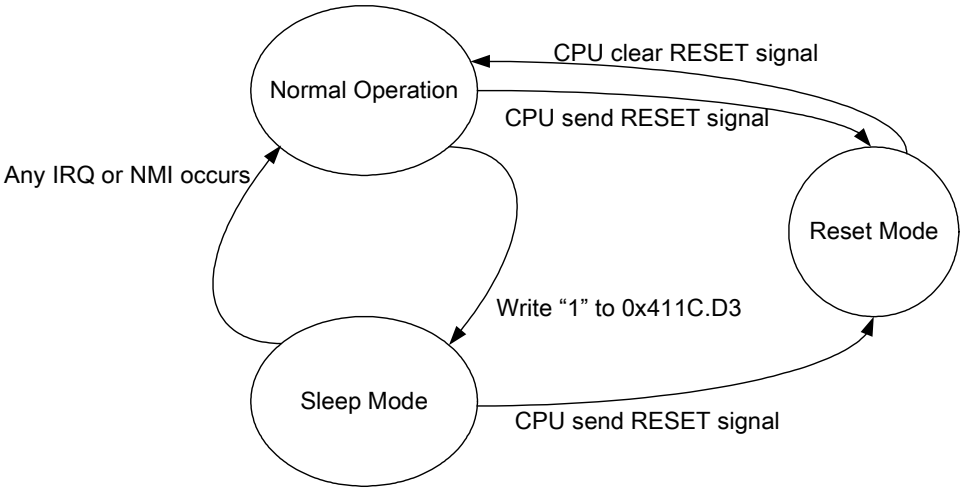
14.3.1 Power-On / Reset 程序

自主 CPU 控制 SCPU,所有在下面图标的指令,在这一部份是给主 CPU 的.这个 SCPU 的 power-on (reset) 流程如下面的图所示.主 CPU 必须先下载(download)SCPU 的程序到这个共享的 SRAM,并且在主 CPU 的存储器设置这个 SCPU 序号(vector)介于 0x1FF4 ~ 0x1FFF 之间.然后使在 0x2106 的 SCPU 时钟致能,清除这个 reset 标志, SCPU 会开始工作.



14.3.2. SCPU 操作流程

在 SCPU 有三种操作方式,如下面的图示.



14.3.2.1 休眠方式(Sleep Mode)

SCPU 可以进入休眠方式去节省电源损耗. 这个 SCPU 的苏醒是由 IRQ 控制,它们是定时器 IRQ, CPU IRQ, NMI 和外部的 IRQ.

注意这个苏醒 IRQ(wakeup IRQ)罩幕必须要打开.

	D7	D6	D5	D4	D3	D2	D1	D0

0x211C(W)				IRQ_OUT	SLEEP	ExtIRQSel	NMI_EN	ExtMask

SLEEP : 写 “1” 去进入休眠方式

NMI_EN : NMI 罩幕

0 : NMI 不是苏醒(wakeup)来源　　1 : NMI 是苏醒(wakeup)来源

ExtMask : 外部的 IRQ 罩幕

0 :外部的 IRQ 不是苏醒(wakeup)来源　　1 : 外部的 IRQ 是苏醒(wakeup)来源

14.3.2.2 与主 CPU 通讯(Communicate with Main CPU)

有两种方法作为 SCPU 来与主 CPU 通讯.第一个方法是在共享的 RAM 去定义一个整体的存储器区域.这个 SCPU 和主 CPU 的共享的存储器区域是介于 0x1000 和 0x1FFF 之间.另一个方法是 IRQ. 主 CPU 可以在 0x211C 的 SCPUIRQ 写一个正的脉冲给 SCPU.同样地, SCPU 也可以在 0x211C 的 IRQ_OUT 写一个正的脉冲给主 CPU.在 SCPU 的 CPU_IRQ 伺服惯例,为了下一个 CPU IRQ,SCPU 必须要读取 0x211C 端口去清除 IRQ 标志.

14.4 定时器(Timer)

在 SCPU 有两组定时器.

14.4.1 TimerA

写 Timer_A_PreLoad 来初始化定时器的频率.写 0x2101 会重新下载 TimerA_Preload 进入定时器,所以 0x2100 必须要在 0x2101 之前被写入.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2100	Timer_A_PreLoad[7:0]							
0x2101	Timer_A_PreLoad[15:8]							
0x2102							TMRA_IRQ	TMRA_EN
0x2103	TimerA_IRQ_Clear							

Timer_PreLoad[15:0]: 定时器 IRQ 期间(period)定义.

For NTSC,

Period = (65536 - Timer_A_PreLoad) / 21.4772MHz

Timer_A_PreLoad = 65536 – (Period(sec) * 21.4772 * 10^6)

For PAL

Period = (65536 - Timer_A_PreLoad) / 26.601712MHz

Timer_A_PreLoad = 65536 – (Period(sec) * 26.601712 * 10^6)

TMRA_En : TimerA 致能控制.

0 : 无作用　　1 : 致能

TMRA_IRQ: TimerA IRQ 致能控制.

0 : 无作用　　1 : 致能.

TimerA_IRQ_Clear : TimerA IRQ 清除控制, 写任一个值来清除 TimerA IRQ.

14.4.2 TimerB

写 Timer_B_PreLoad 来初始化定时器的频率.写 0x2111 会重新下载 TimerB_Preload 进入定时器,所

以 0x2110 必须要在 0x2111 之前被写入.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2110	Timer_B_PreLoad[7:0]							
0x2111	Timer_B_PreLoad[15:8]							
0x2112							TMRB_IRQ	TMRB_EN
0x2113	TimerB_IRQ_Clear							

Timer_PreLoad[15:0]: 定时器 IRQ 期间(period)定义.

For NTSC,

Period = (65536 - Timer_B_PreLoad) / 21.4772MHz

Timer_B_PreLoad = 65536 – (Period(sec) * 21.4772 * 10^6)

For PAL

Period = (65536 - Timer_B_PreLoad) / 26.601712MHz

Timer_B_PreLoad = 65536 – (Period(sec) * 26.601712 * 10^6)

TMRB_En : TimerB 致能控制.

0： 无作用 1： 致能

TMRB_IRQ: TimerB IRQ 致能控制.

0： 无作用 1： 致能

TimerB_IRQ_Clear : TimerB IRQ 清除控制, 写任一个值来清除 TimerB IRQ.

14.5 音频输出(Audio Output)

有两种方法从 VT1682 来输出音频(Audio).第一种方法是 IIS 界面,如下一个部份的表示.另一种方式是内部的音频(Audio) DAC.它有 12 位的精度,所以只有这 12 个 MSB [15:4] 会被输出.请记得在你准备透过音频(Audio) DAC 输出音频之前经由主 CPU 来打开这个 audio DAC.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2118	Audio_DAC_L[7:0]							
0x2119	Audio_DAC_L[15:8]							
0x211A	Audio_DAC_R[7:0]							
0x211B	Audio_DAC_R[15:8]							

Audio_DAC_L[15:0] : Audio DAC 左信道输出数据.

Audio_DAC_R[15:0] : Audio DAC 右信道输出数据.

14.6 IIS 界面

VT1682 为更高的音频质量输出应用提供 16 位双工通道的 IIS 输出. IIS 信号是输出在 XSCPUIOB6, XSCPUIOB5 和 XSCPUIOB4. 给 IIS 界面致能,请记得确认 IIS_EN 必须置为"1"和 SCPUIOB[6:4]必须设置为输出方式.

	D7	D6	D5	D4	D3	D2	D1	D0
0x211D							IIS_Mode	IIS_EN

IIS_Mode : IIS 格式选择

0 : MSB 认可格式　　1 : IIS-bus 格式

IIS_EN : IIS 界面致能控制.

0 : 无作用　　1 : 致能

14.7 IRQ 控制

在 SCPU 有四个 IRQ,它们是外部的 IRQ, Timer-A IRQ, Timer-B IRQ 和 CPU IRQ. 它们的序号(vector) 地址如下表所列.

IRQ	IRQ vector 地址 (字节)
NMI	0x0FFA, 0x0FFB
Ext_IRQ	0x0FFE, 0x0FFF
TimerA_IRQ	0x0FF8, 0x0FF9
TimerB_IRQ	0x0FF6, 0x0FF7
CPU_IRQ	0x0FF4, 0x0FF5
RESET	0x0FFC, 0x0FFD

除了 CPU_IRQ 之外,每一个 IRQ 有一个罩幕标志(mask flag). Timer-A 和 Timer-B 的罩幕是在 0x2102 和 0x2112, 而 NMI 和外部的 IRQ 的罩幕是在 0x211C. 当这个罩幕是"0"时,这个 IRQ 不会被 SCPU 侦测到. IRQ_OUT 是 IRQ 输出到主 CPU 作为两个 CPU 之间的通讯用.写一个正的脉冲到 IRQ_OUT 中断主 CPU.读取寄存器 0x211C 会清除 CPU IRQ,而且这个动作必须要在 CPU IRQ 伺服惯例来做.不然的话,下一个 CPU IRQ 将不会发生.在 0x211C 的 EXTIRQSel 是用来选择外部的 IRQ 来源,它可以是来自于脚位 XSCPUIOB7 的低准位触发两者其中之一.如果在 SCPU 的 NMI 被使用,在主 CPU(0x2000) 的 NMI 必须被致能.

	D7	D6	D5	D4	D3	D2	D1	D0
0x211C(W)				IRQ_OUT	SLEEP	ExtIRQSel	NMI_EN	ExtMask
0x211C(R)	Clear_CPU_IRQ							

IRQ_OUT :送 IRQ 到 CPU.

0 : 清除 IRQ　　1 : 设置 IRQ

SLEEP :休眠方式致能, 写 "1" 去进入休眠方式.

注释 : 任一个发出的 IRQ 会苏醒 CPU.

EXTIRQSel : 外部的 IRQ 来源选择控制

0 : SCPUIOB[7] 下降边缘　　1 : 外部的 CPU 写传送完成

NMI_EN : NMI 致能控制

0 : 无作用　　1 : 致能

ExtMask : IRQ 罩幕给外部的 IRQ 来源

0 : 外部的 IRQ 无作用　　1 : 外部的 IRQ 致能

Clear_CPU_IRQ : 从 CPU 清除 IRQ 标志, 写任一值来清除 IRQ 标志.

14.8 提高运算单元(Enhanced Arithmetic Unit) –乘法器(Multiplier)和除法器(Divider)

SCPU 有一个专门的 ALU 作为乘法和除法. 这个乘法器是 16 位乘以 16 位,这个除法器是 32 位除以

16 位. 这个乘的运算需要 16 个 CPU 时钟周期来完成此操作而除的运算需要 32 个.

14.8.1 乘法器(Multiplier) (16x16)

这个乘法的操作是,

ALU_ Multi _operand6, ALU_ Multi _operand5

XI) ALU_operand2, ALU_operand1

= ALU_out4, ALU_out3, ALU_out2, ALU_out1

当 ALU_Multi_operand6 被写入时,这个操作开始.在乘法运算之后是在 ALU_Multi_operand5 和 ALU_Multi_operand6 的值被改变，而不是在 ALU_operand1 和 ALU_operand2 的值.

0x2130(W)	ALU_operand1
0x2131(W)	ALU_operand2
0x2132(W)	ALU_operand3
0x2133(W)	ALU_operand4
0x2134(W)	ALU_Multi_operand5
0x2135(W)	ALU__Multi_operand6

0x2130(R)	ALU_out1
0x2131(R)	ALU_out2
0x2132(R)	ALU_out3
0x2133(R)	ALU_out4

14.8.2 除法器(Divider)

这个除法操作是:

ALU_operand4, ALU_operand3, ALU_operand2, ALU_operand1

ALU__Div_operand6, ALU__Div_operand5

= { ALU_out4, ALU_out3, ALU_out2, ALU_out1} + { ALU_out6, ALU_out5}或是+{{ ALU_out6, ALU_out5}*2 - { ALU_out4, ALU_out3, ALU_out2, ALU_out1}}

当 LSB(Least Significant Bit) 为 “1”时，余数会是 :

{ ALU_out6, ALU_out5}*2 - { ALU_out4, ALU_out3, ALU_out2, ALU_out1}

当 LSB 为 “0”时，余数会是 { ALU_out6, ALU_out5}.

当 ALU__Div_operand6 被写入时这个操作开始. 在除法运算之后是在 ALU_operand1 和 ALU_operand2, ALU_operand3 和 ALU_operand4 值被改变，而不是在 ALU_Div_operand5 和 ALU_Div_operand6 的值.

0x2130(W)	ALU_operand1
0x2131(W)	ALU_operand2
0x2132(W)	ALU_operand3
0x2133(W)	ALU_operand4
0x2136(W)	ALU_Div_operand5
0x2137(W)	ALU_Div_operand6

0x2130(R)	ALU_out1
0x2131(R)	ALU_out2
0x2132(R)	ALU_out3
0x2133(R)	ALU_out4
0x2134(R)	ALU_out5
0x2135(R)	ALU_out6

14.9 IO

在 SCPU 有 16 个 IO 口.它们每一个都是位方式控制,它可以是拉到高电平(pull-high)输入, 拉到低电平(pull-low)输入,漂浮输入(floating input), 高电平输出(output high)或是低电平输出(output low).次要的 CCIR 输入, SCPU 外部的 IRQ 和 IIS 界面,都是共享这些 IO 脚位.

DIR	ATTR	DATA	Operation Mode
0	0	0	Input floating
0	0	1	Input floating
0	1	0	Input with pull-low resistor
0	1	1	Input with pull-high resistor
1	0	0	Output low
1	0	1	Output high
1	1	0	Output low
1	1	1	Output high

14.9.1 IOA

IOA 可以是 GPIO 或是次要的 CCIR 界面.当它是作为次要的 CCIR 界面时, 记得要去设置 IOA 为输入漂浮(input floating)方式.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2140(W)	IOA_Data							
0x2140(R)	IOA_Data							
0x2141	IOA_DIR							
0x2142	IOA_ATTR							

共享脚位表

脚位名称	信号名称
XSCPUIOA0	CCIR_D0
XSCPUIOA1	CCIR_D1
XSCPUIOA2	CCIR_D2
XSCPUIOA3	CCIR_D3
XSCPUIOA4	CCIR_D4
XSCPUIOA5	CCIR_D5
XSCPUIOA6	CCIR_D6
XSCPUIOA7	CCIR_D7

14.9.2 IOB

IOB 可以是 GPIO,次要的 CCIR 界面, IIS 或是外部的 IRQ. 当它是 CCIR 时, 这些相关的脚位必须被设置为输入漂浮方式(input floating). 当外部的 IRQ 被使用时,XSCPUIOB7 必须设置为带有 Pull-high 电阻的输入方式. 当 IIS 有作用时, XSPUIOB6, XSPUIOB5 和 XSPUIOB4 必须被设置为输出方式.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2144(W)	IOB_Data							
0x2144(R)	IOB_Data							
0x2145	IOB_DIR							
0x2146	IOB_ATTR							

共享脚位表

	输出	输入
SCPU_IOB1	----	EXT_CPU_CSB / CCIR_HS
SCPU_IOB2	----	CCIR_VS
SCPU_IOB3	----	----
SCPU_IOB4	IIS_CK	----
SCPU_IOB5	IIS_DA	----
SCPU_IOB6	IIS_SW	----
SCPU_IOB7	----	SCPU_IRQN

15. 寄存器表(REGISTER TABLE)

图像的寄存器(Graphic Registers)

	D7	D6	D5	D4	D3	D2	D1	D0
0x2000				Capture	SLAVE	---	---	NMI_EN
0x2001(W)					EXT_CLK_DIV		SP_INI	BK_INI
0x2001(R)	VBLANK	SP_ERR						
0x2002						SPRAM_ADDR[2:0]		
0x2003	SPRAM_ADDR[10:3]							
0x2004	SPRAM_DATA[7:0]							
0x2005	VRAM_ADDR[7:0]							
0x2006	VRAM_ADDR[15:8]							
0x2007	VRAM_DATA[7:0]							
0x2008	LCD_VS_DELAY							
0x2009	LCD_HS_DELAY							
0x200A	LCD_FR_DELAY[7:0]							
0x200B	CH2_Odd_line_color		CH2_Even_line_color		CH2_SEL	CH2_REV	LCD_FR[8]	LCD_HS[8]
0x200C	F_RATE	DotODR	LCD_CLK		UPS052	Field_AC	LCD_MODE	
0x200D	LCDEN	Dot240	Reverse	VCOM	Odd_line_color		Even_line_color	
0x200E			Blend2	Blend1	Pal2_Out_Sel		Pal1_Out_Sel	
0x200F					BK2_Pal_Sel		BK1_Pal_Sel	
0x2010	BK1_X[7:0]							
0x2011	BK1_Y[7:0]							
0x2012				BK1_HCLR	BK1_Scroll_En		BK1_Y[8]	BK1_X8
0x2013	BK1_EN	BK1_Pal	BK1_Depth		BK1_Color		BK1_Line	BK1_Size
0x2014	BK2_X[7:0]							
0x2015	BK2_Y[7:0]							
0x2016					BK2_Scroll_En		BK2_Y[8]	BK2_X8
0x2017	BK2_EN	BK2_Pal	BK2_Depth		BK2_Color		----	BK2_Size
0x2018					SPALSEL	SP_EN	SP_SIZE	
0x2019					BK2_Gain		BK1_Gain	
0x201A	SP_SEGMENT[7:0]							
0x201B					SP_SEGMENT[11:8]			
0x201C	BK1_SEGMENT[7:0]							
0x201D					BK1_SEGMENT[11:8]			
0x201E	BK2_SEGMENT[7:0]							
0x201F	----	----	----	----	BK2_SEGMENT[11:8]			
0x2020	----	----	BK2_L_EN	BK1_L_En	Scroll_Bank			
0x2021	----	----	Luminance_offset					
0x2022	----	----	VCOMIO	RGB_DAC	CCIR_OUT	Saturation		
0x2023	Light_Gun_Reset							
0x2024	Light_Gun1_Y							
0x2025	Light_Gun1_X							
0x2026	Light_Gun2_Y							
0x2027	Light_Gun2_X							
0x2028	----	----	CCIR_Y					
0x2029	----	----	----	CCIR_X				

0x202A	VS_Phase	HS_Phase	YC_Swap	CbCrswap	SYNCMOD	YUV_RGB	Field_OEn	Field_On
0x202B	R_EN	G_EN	B_EN	HalfTone	B/W	CCIR_Depth		
0x202E	TRC_EN	CCIR_EN	BlueScr_EN	Touch_EN	CCIR_TH			
0x2030	----	VDACSW	VDAC_OUT[5:0]					
0x2031	----	----	RDACSW	RDAC_OUT[4:0]				
0x2032	----	----	GDACSW	GDAC_OUT[4:0]				
0x2033	----	----	BDACSW	BDAC_OUT[4:0]				

系统的寄存寄存器(System Registers)

	D7	D6	D5	D4	D3	D2	D1	D0
0x2100(W)					Program_Bank1_Register3			
0x2100(R)					Program_Bank1_Register3			
0x2101(W)	Timer_Preload							
0x2101(R)	Timer_Preload							
0x2102							TMR_IRQ	TMR_EN
0x2103	Timer_IRQ_Clear							
0x2104(W)	Timer_Preload[15:8]							
0x2104(R)	Timer_Preload[15:8]							
0x2105(W)	---	COMR6	TV_SYS_SE:[1:0]		CCIR_SEL	Double	ROM_SEL	PRAM
0x2106	---	---	SCPURN	SCPU_ON	SPI_ON	UART_ON	TV_ON	LCD_ON
0x2107(W)	Program_Bank0_Register0							
0x2107(R)	Program_Bank0_Register0							
0x2108(W)	Program_Bank0_Register1							
0x2108(R)	Program_Bank0_Register1							
0x2109(W)	Program_Bank0_Register2							
0x2109(R)	Program_Bank0_Register2							
0x210A(W)	Program_Bank0_Register3							
0x210A(R)	Program_Bank0_Register3							
0x210B	TSYNEN	PQ2EN	BUSTRI	CS_Control[1:0]		Program_Bank0_select		
0x210C(W)					Program_Bank1_Register2			
0x210C(R)					Program_Bank1_Register2			
0x210D	IODENB	IODOEN	IOCENB	IOCOE	IOPBENB	IOBOE	IOAENB	IOAOE
0x210E(W)	IOB_O[3:0]				IOA_O[3:0]			
0x210E(R)	IOB_I[3:0]				IOA_I[3:0]			
0x210F(W)	IOD_O[3:0]				IOC_O[3:0]			
0x210F(R)	IOD_I[3:0]				IOC_I[3:0]			
0x2110(W)					Program_Bank1_Register0			
0x2112(R)					Program_Bank1_Register0			
0x2111(W)					Program_Bank1_Register1			
0x2113(R)					Program_Bank1_Register1			
0x2112(W)	Program_Bank0_Register4							
0x2110(R)	Program_Bank0_Register4							
0x2113(W)	Program_Bank0_Register5							
0x2111(R)	Program_Bank0_Register5							
0x2114	Baud_rate[7:0]							
0x2115	Baud_rate[15:8]							
0x2116	16bitMode	SPIEN	SPI_RST	M/SB	CKPHASE	CKPOLAR	CK_FREQ[1:0]	
0x2117(W)	SPI_TX_Data							

0x2117(R)	SPI_RX_Data							
0x2118(W)	Program_Bank1_Register5				Program_Bank1_Register4			
0x2118(R)	Program_Bank1_Register5				Program_Bank1_Register4			
0x2119(W)	--	CarriEn	UARTEN	TxIRQEn	RxIRQEn	ParityEn	OddEven	9bitmode
0x211A(W)	Tx_Data[7:0]							
0x211A(R)	Rx_Data[7:0]							
0x211B	Carrier_frequency[7:0]							
0x211B(R)	--	--	RxError	Tx_Status	Rx_Status	ParityErr	--	--
0x211C	AutoWake	KeyWake	EXT2421EN	SCPUIRQ	SLEEPM	----	SLEEPSel	CLKSEL
	Clear_SCPU_IRQ							
0x211D(W)	LVDEN	LVDS1	LVDS0	VDAC_EN	ADAC_EN	PLL_EN	LCDACEN	---
0x211D®	----	-----	----	----	----	----	----	LVD
0x211E(W)	ADCEN	ADCS1	ADCS0	UNUSE	IOFOEN3	IOFOEN2	IOFOEN1	IOFOEN0
0x211E®	ADC_Data[7:0]							
0x211F	VGCEN	VGCA6	VGCA5	VGCA4	VGCA3	VGCA2	VGCA1	VGCA0
0x2120	SLEEP_PERIOD							
0x2121	--	--	--	SPI_MSK	UART_MSK	SPU_MSK	TMR_MSK	Ext_MSK
0x212E						Clear_Ext		Clear_SPU
0x2122	DMA_DT_Addr[7:0]							
0x2123	DMA_DT_Addr[15:8]							
0x2124	DMA_SR_AddrT[7:0]							
0x2125	DMA_SR_Addr[15:8]							
0x2126	DMA_SR_Bank[22:15]							
0x2127(W)	DMA_Number							
0x2127®								DMAStatus
0x2128							DMA_SR_Bank[24:23]	
0x2129®	Send_JOY_CLK							
0x2129(W)	UIOA_DATA_OUT							
0x2129®	UIOA_DATA_IN							
0x212A®	Send_JOY_CLK2							
0x212A(W)	UIOA_DIRECTION							
0x212B	UIOA_ATTRIBUTE							
0x212C(W)	Pseudo_random_number_seed							
0x212C®	Pseudo_random_number							
0x212D(W)	PLL_B				PLL_M	PLL_A		
0x2130(W)	ALU_operand1							
0x2131(W)	ALU_operand2							
0x2132(W)	ALU_operand3							
0x2133(W)	ALU_operand4							
0x2134(W)	ALU_Multi_operand5							
0x2135(W)	ALU__Multi_operand6							
0x2136(W)	ALU_Div_operand5							
0x2137(W)	ALU_div_operand6							
0x2130®	ALU_out1							
0x2131®	ALU_out2							
0x2132®	ALU_out3							
0x2133®	ALU_out4							

0x2134®	ALU_out5						
0x2135®	ALU_out6						
0x2140	IIC_ID						
0x2141	IIC_ADDR						
0x2142(W)	IIC_DATA						
0x2142(R)	IIC_DATA						
0x2143							IIC_CLK_SEL
0x2148(W)	UIOB_SEL[7:3]						UIOA_MODE
0x2149(W)	UIOB_DATA_OUT						
0x2149®	UIOB_DATA_IN						
0x214A	UIOB_DIRECTION						
0x214B	UIOB_ATTRIBUTE						
0x214C			KeyChangeEN	IOFEN	----	----	IOEOEN
0x214D(W)	IOF[3:0]			IOE[3:0]			
0x214D®	IOF[3:0]			IOE[3:0]			

SCPU 寄存器(SCPU Registers)

	D7	D6	D5	D4	D3	D2	D1	D0
0x2100	Timer_A_PreLoad[7:0]							
0x2101	Timer_A_PreLoad[15:8]							
0x2102							TMRA_IRQ	TMRA_EN
0x2103	Timer_A_IRQ_Clear							
0x2110	Timer_B_PreLoad[7:0]							
0x2111	Timer_B_PreLoad[15:8]							
0x2112							TMRB_IRQ	TMRB_EN
0x2113	Timer_B_IRQ_Clear							
0x2118	Audio_DAC_L[7:0]							
0x2119	Audio_DAC_L[15:8]							
0x211A	Audio_DAC_R[7:0]							
0x211B	Audio_DAC_R[15:8]							
0x211C(W)				IRQ_OUT	SLEEP	ExtIRQSel	NMI_EN	ExtMask
0x211C(R)	Clear_CPU_IRQ							
0x211D							IIS_Mode	IIS_EN
0x211E(W)								
0x211E(W)								
0x2130(W)	ALU_operand1							
0x2131(W)	ALU_operand2							
0x2132(W)	ALU_operand3							
0x2133(W)	ALU_operand4							
0x2134(W)	ALU_Multi_operand5							
0x2135(W)	ALU__Multi_operand6							
0x2136(W)	ALU_Div_operand5							
0x2137(W)	ALU_div_operand6							
0x2130(R)	ALU_out1							
0x2131(R)	ALU_out2							
0x2132(R)	ALU_out3							
0x2133(R)	ALU_out4							

0x2134(R)	ALU_out5
0x2135(R)	ALU_out6
0x2140(W)	IOA_Data
0x2140(R)	IOA_Data
0x2141	IOA_IO_DIR
0x2142	IOA_R_PLH
0x2144(W)	IOB_Data
0x2144(R)	IOB_Data
0x2145	IOB_IO_DIR
0x2146	IOB_R_PLH