VT1682 Programming Guide V1.5

2. REVISION HISTORY

Revision	Date	Remark
V1.0	2005/08/	First edition
V1.1	2005/12/22	Move UART status port $2119 to $211B. Remove CCIR gray capture and blue screen effect. Change 8.2 ~ 8.6 IO setting table. Remove sleep/wake up section. Remove 11.3.2 Eye-Function IRQ. Remove MCU interface section. Reading $2007 for sprite RAM and $2004 for VRAM Change $2110~$2113 read port. Remove Sprite vertical amplify function.
V1.2	2006/03/27	Correct the P23 error of $210B D7 TSYNEN
V1.3	2006/05/02	Correct the P12 Addressing Mode error at EXT2421. Correct the Timer formula error in P23. Correct the BK1/BK2 VRAM management in P29 and P30. Correct the remainder of the ALU Division in P22 and P68.
V1.4	2006/06/01	Correct the 201D.D5 in P18.
V1.5	2006/06/26	Correct the light gun programming sequence in P45

3. GENERAL DESCRIPTION

VT1682 includes the main CPU, Graphic Processor, Sound CPU, internal SRAM (8K bytes for program and 4K bytes for video) ROM (4Kbytes), and some I/O controllers. There are two main systems in VT1682, program system and video system.

Main CPU plays the key role in program system. It can access the internal and external program memories. The program memory stores the program command, instructions, and sound data. VT1682 is equipped with 8K Bytes SRAM as internal program memory. This program RAM will be the zero pages RAM, STACK and some memory of CPU. Program system controls the operations of education machine, including figure, voice, and the title. It means CPU will control the video system to display the specified figure.

Graphic Unit is the main role of the video system. It can access the video memory automatically to display some figures. In addition to the internal program SRAM, VT1682 is equipped the other 4K Bytes SRAM for Video RAM. Internal Video RAM stores pattern vectors for 2 layers of background. External Video memory stores the video characters to be pointed by the pattern vectors.

Sound CPU shared the internal ROM and 4K bytes program SRAM with main CPU. It has the individual IO and ALU. It operates four times faster than main CPU, and suits for different applications.

3.1 Feature

System

- Working Voltage 3.0~3.6 V
- Main CPU: 6502 @5.3693MHz in NTSC and 5.3203MHz in PAL
- Internal optional Program ROM: 4K Bytes
- Internal Main CPU Program RAM: 8K Bytes (4K bytes local RAM and 4K bytes shared RAM)
- Internal Video RAM: 4K Bytes
- Direct Memory Access (DMA) Sprite RAM / VRAM / Program RAM / External memory
- Single 16bits data bus
- Scan line IRQ / 16-bits Timer IRQ / External IRQ
- Expandable memory up to 32M bytes with 3 addresses decoder (CSB).
- T.V. signal output (NTSC, PAL, PAL-M, PAL-N)
- Extend 5 IRQ service entry
- 56 GPIO ports, 40 are for Main CPU, the other 16 are for Sound CPU.

Peripheral

- ADC: 8bits, 5 Times-Division-Multiplex channels with Voice Gain control
- 4 level low voltage detect
- Master/Slave SPI Interface:
- UART Interface
- TFT LCD Interface.
- STN LCD Interface
- IIS Interface
- IIC interface (Master mode)
- CCIR656/601 Interface
- Enhanced ALU, 16 by 16 multiplier and 32 by 16 divider

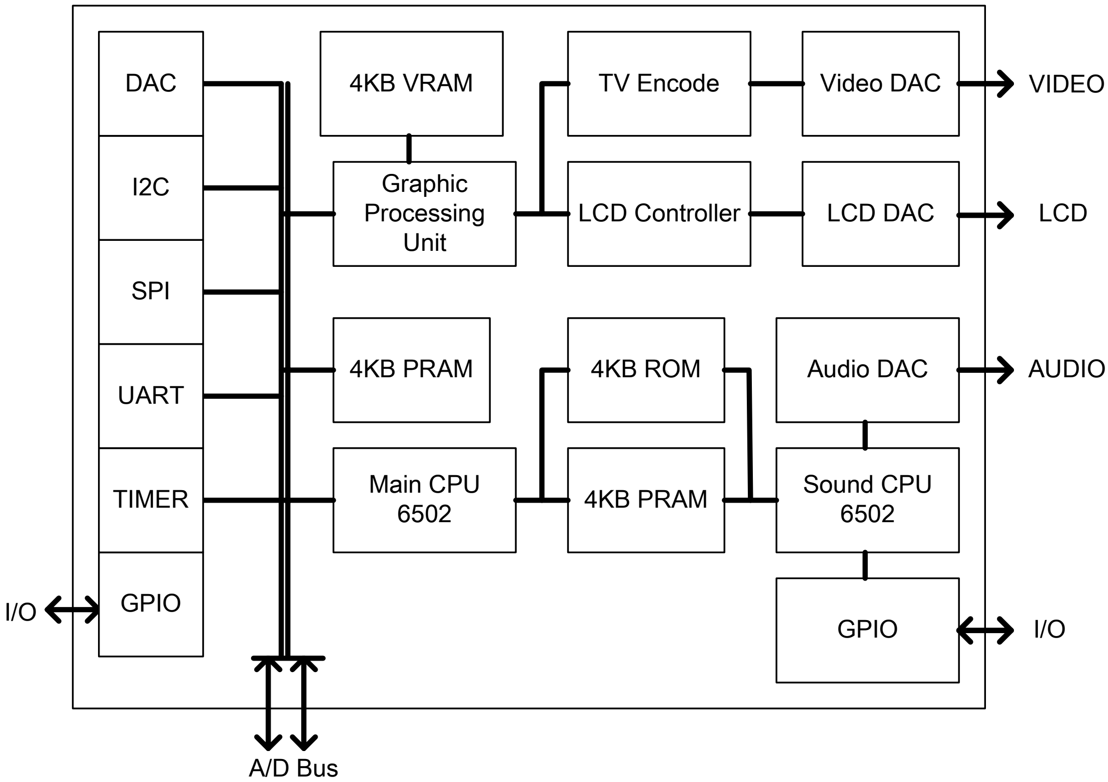
Graphic Processor

- Resolution: TV 256x240 pixels
- 240 sprites in one frame, 16 sprites in one horizontal line
- 2 independent background layers.
- Background character mode: 16/64/256 indexed color mode.
- Background bitmap mode: 16/64/256 indexed color mode or 32768 colors direct color mode
- Sprites are 16 colors.
- Two 256 colored-Color palettes, maximum display indexed color: 512
- Background vertical extension: x1/x1.5/x2
- Background horizontal line individual scrolling: -128~+127

Sound CPU

- CPU 6502 @21.4772MHz in NTSC and 26.6017MHz in PAL
- 4Kbytes Shared RAM
- 4Kbytes optional Internal ROM
- 16 GPIO ports
- 16 bits Timer x2
- ALU, 16 by 16 multiplier and 32 by 16 divider

3.2 BLOCK DIAGRAM



3.3 Pin Description

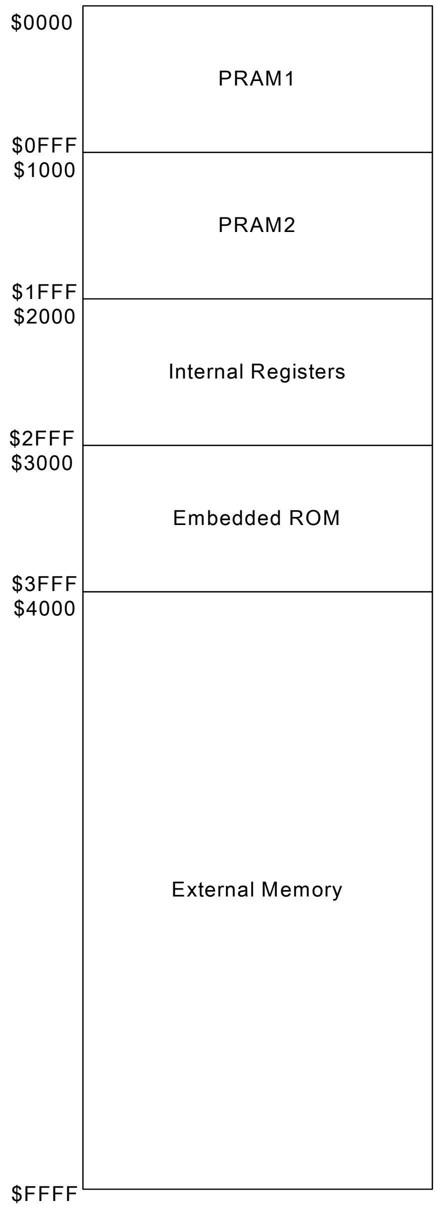
SYMBOL	TYPE	DESCRIPTION
XA[23:0]	O	Address bus
XD[15:0]	I/O	Data bus
XROMCSB	O	1st external memory CSB signal
XRAMCSB	O	2nd external memory CSB signal
XROMOEB	O	External memory OEB signal
XRAMRWB	O	External memory RWB signal
XDEBUGNMI	I	NMI for debug mode
XDEBUGCSB	O	Memory CSB for debug mode
XIOA[3:0]	I/O	Universal I/O
XIOB[3:0]	I/O	Universal I/O
XIOC[3:0]	I/O	Universal I/O
XIOD[3:0]	I/O	Universal I/O
XIOE[3:0]	I/O	Universal I/O
XIOF[3:0]	I/O	Universal I/O
XUIOA[7:0]	I/O	Universal I/O
XUIOB[7:0]	I/O	Universal I/O
XSCPUIOA[7:0]	I/O	Universal I/O
XSCPUIOB[7:0]	I/O	Universal I/O
XTAL1	I	Crystal pin
XTAL2	O	Crystal pin
XBOOTINIT	I	Internal ROM Boot up mode
XPLLVCO	I/O	PLL reference voltage
XPLLVREF	I/O	PLL reference voltage
XVIDEO	O	Composite video signal
XAUDIOR	O	Right channel audio signal
XAUDIOL	O	Left channel audio signal

4. MAIN CPU

Main CPU in the VT1682 is 8-bits 6502 operates at 5MHz.

4.1 Memory Map

The partition of the memory map is shown in the following diagram. PRAM1 (Program RAM-1) is 4KBytes for Main CPU local Program RAM. PRAM2 is 4KBytes shared by Main CPU and Sound CPU. Address between 0x2000 and 0x20FF is the Graphic ports and the others are for the system or peripheral control. There is a 4KBytes embedded ROM for either Main CPU or Sound CPU. It could be the BIOS, security, program ROM or data ROM. $4000 ~ $FFFF are mapped to 6 Program Bank to extend the address to 8MBytes external memory.



4.2 Address Mode

The 16 bits 6502 CPU address is extended to 25 bits physical address according to the following coding style. In the following table, the 6502 Address is A[15:0] and the Physical Address is {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] would be translated in the next table

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

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 Vectors

The vectors in main CPU include NMI, RESET and 5 IRQ as shown in the following table.

Vector Name	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

5. INTERFACE

Interfaces that controlled by VT1682 main CPU include TV composite output, LCD, UART, SPI, IIC and CCIR interface.

5.1 TV Composite Output

VT1682 can support up to 4 types of TV system, and allows adjusting the saturation and luminance of the TV signal. Write "1" to TV_ON in 0x2106 and VDAC_EN in 0x211D to enable the composite video output function.

	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 TV System Configuration

NTSC, PAL, PAL-M and PAL-N TV systems are valid in VT1682 by setting the port 0x2105 and replace the related crystal at the pin XTAL1 and XTAL2. Their relationship is listed in the following table.

	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 TV Parameter

There are 32 brightness and 8 contrast levels are programmable in VT1682 by setting the port 0x2021 and 0x2022. Well-cooperate with 0x2021 could generate the fade effect.

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

Luminance_offset : TV output luminance adjustment control, represented in 2's complement.

2's complement : -A = ~A + 1,

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

Effective value -32--------- -16 ------ -1 –0 – 1-------- 15 ------- 31

darkest---darker----------Normal-----Lighter-----Brightest

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

Saturation = 0 : Gray output, default value is 4,

Value in 0x2022 1-----2-----3-----4------5------6------7

Saturation High <---------------Normal---------------→ Low-

5.2 LCD Interface

VT1682 could provides kinds of LCD interfaces, include AUO-051, AUO-052, analog TFT LCD and CSTN interface. There are four major output protocol selected by LCD_MODE in 0x200C. Their mapped signals are listed in the following table.

	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 : Vertical offset of LCD display screen with base of horizontal scan line.

Note: 0xFF and 0 is forbidden.

LCD_X : Horizontal offset of LCD display screen with base of LCD dot clock.

Note: 0x1FF and 0 is forbidden.

LCD_FR: STN LCD alternate signal toggle rate with the base of horizontal scan line.

F_RATE: LCD controller parameters, STN frame rate control.

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

LCD_CLK: LCD controller parameter, LCD dot clock select

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

2 : 21.4772 MHz 3 : External clock*

UPS052 : LCD controller parameter, AUO UPS052 TCON mode select

0 : Non UPS052 mode 1 : UPS052 mode

LCDEN, LCD enable control

0 : Disable, 1 : Enable

Dot240: LCD Scaling control, scaling to the ratio 15/16.

0 : No scaling 1 : Scaling

Reverse: LCD controller parameter, pixel data output reverse

0 : Normal, 1 : Reverse

LCD_MODE : LCD controller output protocol select

0 : ANALOG mode 1 : CSTN mode

2 : LTPS mode 3 : DIGITAL mode

VCOMIO : LCD input VCOM mode

0 : disable 1 : enable

ANALOG Mode

PIN NAME	SIGNAL NAME	SETTING	DESCRIPTION
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

PIN NAME	SIGNAL NAME	SETTING	DESCRIPTION
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

Digital Mode

PIN NAME	SIGNAL NAME	SETTING	DESCRIPTION
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 Mode

PIN NAME	SIGNAL NAME	SETTING	DESCRIPTION
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 interface

UART controller in VT1682 could provide up to 11520 bps receive/transmit baud-rate, programmable receive/transmit IRQ, parity check. UART transmit (TX) and UART receive (RX) is at XIOC2 and XIOC3.

5.3.1 UART Control Registers

UART_ON in 0x2106 should be set first to enable the UART before any registers setting to UART ports. When the TXIRQEn or RXIRQEn, both of the TXIRQ and RXIRQ would feed to 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 enable

0 : Disable 1 : Enable

TXIRQEn : TX IRQ Enable. UART_IRQ would occurs when TXIRQEn=1 and TX_Status=1.

0 : Disable 1 : Enable

RXIRQEn : RX IRQ Enable. UART_IRQ would occurs when RXIRQEn=1 and RX_Status=1.

0 : Disable 1 : Enable

ParityEn : RX parity check enable, the check result is presented at ParityErr in UART status port.

0 : Disable 1 : Enable

OddEven : When the ParityEn is set, odd / even parity should be set.

0 : Odd parity 1 : Even parity

9bitmode : RX / TX transmit at 9 bits mode. When the ParityEn is set, the 9^{th} bit would be the parity bit (odd / even). Otherwise, the 9^{th} bit would be just the data in OddEven.

UART Status (Read Only)

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

RxError : RX error flag.

0 : no error 1 : Baudrate mis-match or out of stop bit.

TX_Status : The status flag of the TX. Writing Tx_Data would clear the status flag.

0 : UART is transmitting 1 : UART complete the transmitting

RX_Status : The status of the RX. Reading Rx_Data would clear the status flag.

0 : no data in Rx_Data 1 : data is valid in Rx_Data

ParityErr : Parity check error flag. ParityEn should be enabled first.

0 : Parity correct 1 : Parity error

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

Writing 0x211A would transmit data through TX. Reading 0x211A would get the RX data when RX_Status=1.

5.3.2 Baud Rate Setting

UART controller in VT1682 could provide up to 11520 bps receive/transmit baud-rate. Besides, UART interface also provides the carrier parameter on TX output for the Infra-red application. Setting the port Carier_Frequency in0x211B to adjust the carrier frequency. Setting CarriEN in 0x2119 to enable the carrier function.

	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 $x10^6$ / $F_{carrier}$)

Where Baud_rate is 115200, 57600, 38400, 19200, 9600, 4800, 2400 and $F_{carrier}$ is the required carrier frequency

5.3.3 UART Signal

UART pins, TX and RX are shared with XIOC2 and XIOC3.

	PIN POSITION	SETTING	Description
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)

Programming Notes

UART_TX would be at XIOC2 and UART_RX at XOIC3. Following registers should be set before the UART registers.

$2106.D2 = 1; // UART module enable

$210D.D4 = 1; // IOC Output Enable

$210D.D5 = 1; // IOC Output

5.4 SPI interface

SPI (Standard Peripheral Interface) could operate at master and slave mode. It could operate at 4 different speeds, controlled be CK_FREQ in 0x2116. Also there are four types of protocols as shown in the following diagram.

	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 operation frequency.

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

CKPOLAR : SPI clock polarity control.

CKPHASE : SPI clock sample phase control.

M/SB : SPI Master / slave mode selection

SPI_RST : SPI module reset signal

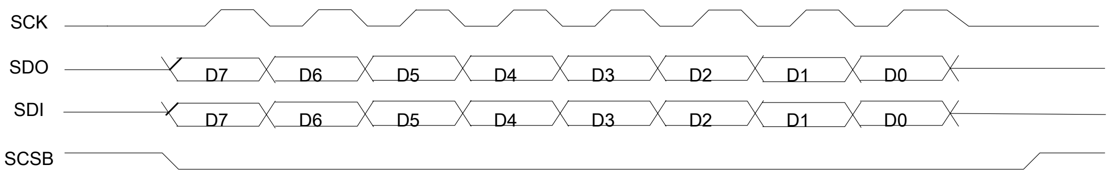
0 : Normal operation 1 : Reset SPI

SPIEN : SPI module enable

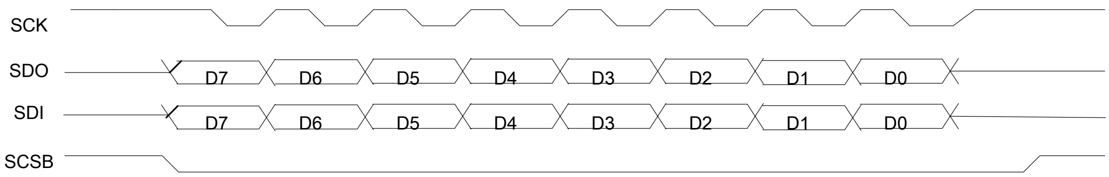
0 : Disable 1 : Enable

16bitMode : 8 / 16-bits SPI mode selection

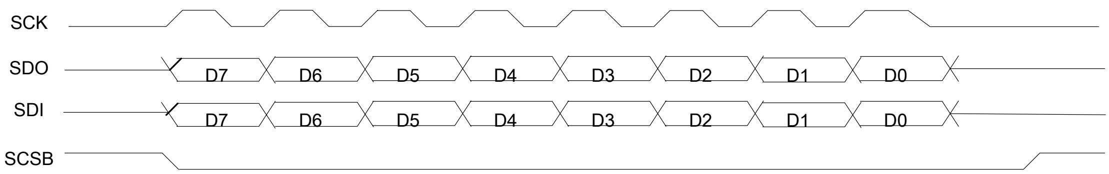
0 : 8-bits mode 1 : 16-bits mode



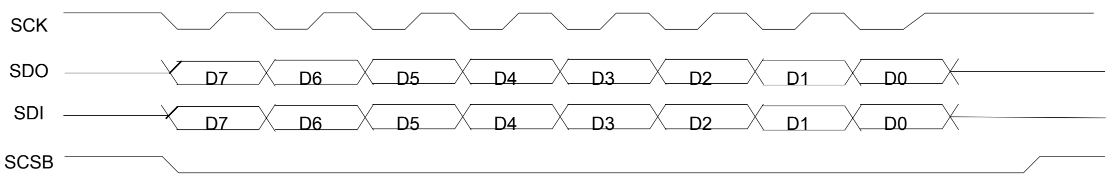
CKPOLAR=0, CKPHASE =0



CKPOLAR=1, CKPHASE =0



CKPOLAR=0, CKPHASE =1

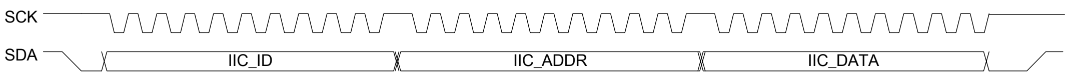


CKPOLAR=1, CKPHASE =1

PIN NAME	Master Mode IOD_ENB=1, IOD_OE=1SPI_ON = 1	Slave Mode IOD_ENB=1, IOD_OE=0SPI_ON = 1	Description
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

Master mode I2C function is valid in VT1682. Each I2C command includes IIC_ID, IIC_ADDR and IIC_Data as shown in the following diagram. When the LSB of IIC_ID is even, the data in 0x2142 would be output in the third phase. Otherwise the IIC_DATA would be the input data stream read at the port 0x2142.



	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 frequency select

0 : 1.34MHz 1 : 670KHz

2 : 335KHz 3 : Forbidden

5.6 CCIR protocol

VT1682 provides 2 sets of CCIR656 / 601 interfaces, SET0 and SET1, for the image input. The input CCIR should be 60 FPS for NTSC or 50 FPS for PAL system.

5.6.1 CCIR Input Pins

These two sets of CCIR input are at XUIOA, XUIOB, XSCPUIOA and XSCPUIOB as shown in the following table. When the CCIR interface is used, the related IO pins should be set to the input mode. CCIR SET0 or SET1 is selected by CCIR_SEL in 0x2105.

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 Control Registers

Following registers are used to control the CCIR interface.

	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 : slave mode enable control, VT1682 would synchronize with CCIR interface.

0 : Normal mode 1 : Slave mode (CCIR input)

CCIR_Y : Vertical offset of CCIR protocol with base of horizontal scan line.

Effective offset = 64 - CCIR_Y

CCIR_X : Horizontal offset of CCIR protocol with number of crystal cycles.

Effective offset = 32 - CCIR_H

Note : 0 is forbidden in CCIR_V and CCIR_H.

Field_On : CCIR parameter, field mode enable.

0 : Disable 1 : Enable

Field_OEn : CCIR parameter, valid on when Field_On is 1.

YUV_RGB : CCIR parameter, input data format select.

0 : YUV 4:2:2 mode 1 : GBR 4:2:2 mode

SYNC_MOD : CCIR parameter, synchronous mode select.

0 : CCIR656 1 : CCIR601

CbCrSwap : CCIR parameter, Cb and Cr order control

0 : CbYCrY 1 : CrYCbY

YC_Swap : CCIR parameter, Y and CrCb order control

0 : CbYCrY 1 : YcbYCr

HS_Phase : CCIR parameter, hsync in CCIR601 polarity control

0 : Low pulse 1 : high pulse

VS_Phase, CCIR parameter, vsync in CCIR601 polarity control

0 : Low pulse 1 : high pulse

6. PERIPHERAL

6.1 Enhanced Arithmetic Unit --Multiplier and Divider

VT1682 provide two multiplier/divider arithmetic Units, one is for main CPU and the other is for Sound CPU. The multiplier is 16 bit by 16 bits and the division is 32 bits by 16 bits. The multiplication requires 16 CPU clock cycle to complete the operation while the division requires 32.

6.1.1 Multiplier (16x16)

The multiply operation is,

ALU_ Multi _operand6, ALU_ Multi _operand5

X) ALU_operand2, ALU_operand1

= ALU_out4, ALU_out3, ALU_out2, ALU_out1

The operation is started when the ALU_Multi_operand6 is written. The value in ALU_Multi_operand5 and ALU_Multi_operand6 are changed, but not in ALU_operand1 and ALU_operand2, after the multiply.

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 Divider

When the division, {ALU_operand4, ALU_operand3, ALU_operand2, ALU_operand1}

is divided by {ALU__Multi_operand6, ALU__Multi_operand5}.

The **quotient** would be { ALU_out4, ALU_out3, ALU_out2, ALU_out1} and

When the LSB(Least Significant Bit) is "1", the **remainder** would be :

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

When the LSB is "0", the **remainder** would be { ALU_out6, ALU_out5}.

The operation is started when the ALU__Div_operand6 is written. The value in ALU_operand1 and ALU_operand2, ALU_operand3 and ALU_operand4are changed, but not in ALU_Div_operand5 and ALU_Div_operand6, after the division.

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 Timer

Write Timer_PreLoad to initialize the timer frequency. Writing 0x2104 would reload the Timer_Preload into timer, so 0x2101 should be written before 0x2104.

	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 period definition.

TSYNEN : Timer base frequency selection

	TSYNEN	Timer Base Frequency
NTSC	1	15.746KHz
	0	5.3693 MHz
PAL	1	15.602KHz
	0	5.32034 MHz

When TSYNEN = 0,

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 : Timer enable control.

0 : disable 1 : enable

TMR_IRQ, Timer IRQ enable control.

0 : disable 1 : enable

Timer_IRQ_Clear : Timer IRQ clear control, write any data to clear Timer IRQ.

6.3 Random Number Generator

There is an 8-bits pseudo-random number generator in VT1682. It requires to write a non-zero seed number into the 0x212C initially. The random number would be valid by reading 0x212C. Every time a new seed number is written, a new pseudo random number sequence would be generated. Please note that it's not necessary to write the seed number every time when you are going to access the random number. Only in the initial sequence, or changing the random sequence, you have to update the send number.

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

6.4 Analog to Digital Converter (ADC)

There is an ADC with 5 timing-division-multiplex channels and 8 bits resolution in VT1682. Four of them are the typical ADC ports, input voltage between (VCC-1) volt and GND would be converted into 255~0. The other one is for the microphone application which includes a Voice Gain Controller (VGC). The gain of this VGC is between 0.5 and 127.5 times. When the ports, XIOF0, XIOF1, XIOF2, XIOF3 or XIOE3 is the ADC input, remember to set IOFOE to the low.

	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 enable control

0 : disable 1 : enable

ADCEN : ADC enable control

0 : power down 1 : normal operation

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

Programming Notes:

1. IOFOE0 = 0, ADCSEL = 0 // Choose XIOF0
2. ADAC_EN = 1, // Enable ADC
3. Periodically read ADC_Data // Get ADC Data

6.5 Phase Lock Loop (PLL)

There is an embedded PLL in VT1682 for the LCD applications. It could generate

	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 DAC (Digital-to-Analog-Converter)

There are 6 DAC (Digital-to-Analog-Converter) in VT1682. One is for Video, two for audio and three for analog LCD interface. However, not all DAC would be used in the application. Fox example, if you use the digital LCD interface, the three LCD DAC would be useless. So you can take them as the extra applications.

Name	Function	Resolution	Response Time	Output PAD	Control Port
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(CSPU)

* Audio DAC port 0x2118~0x211B are Sound CPU access only.

	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 output data switch.

0 : TV composite output(default) 1: output the VDAC_OUT

RDACSW : LCD DAC output data switch

0 : Output the LCD controller data(default) 1: output the RDAC_OUT

GDACSW : LCD DAC output data switch

0 : Output the LCD controller data(default) 1: output the GDAC_OUT

BDACSW : LCD DAC output data switch

0 : Output the LCD controller data(default) 1: output the BDAC_OUT

Note1 : If you need the TV composite output, the VDACSW should be set LOW.

Note2 : If the analog RGB data is used in LCD interface, these switch should be set LOW.

Note3 : Please refer to the "Sound CPU" section for the Audio DAC control.

6.7 Low Voltage Detect (LVD)

There are four levels for low-voltage-detect function. They are 3.65V, 3.18V, 2.71V and 2.24V. When the IC power is lower than the detection level, the LVD flag in 0x211D would turn into “high”. Writing “1” to LVDEN to enable the LVD function.

	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 enable control

0 : disable 1 : enable

6.8 Embedded ROM

There is a 4KB embedded ROM in the VT1682. It can be accessed either for main CPU or Sound CPU selected by ROM_SEL in 0x2105 and its address of this ROM is between 0x3000 and 0x3FFF.

	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

This ROM can also be a boot ROM. When the pin “XBOOTINIT” is set to high, the RESET, NMI, IRQ vectors would change from 0x7FFF0 ~ 0x7FFFF to 0x00FF0 ~0x00FFF.

7. GRAPHIC – PICTURE PROCESSING UNIT (PPU)

7.1 Feature

- Resolution: TV 256x240 pixels
- 240 sprites in one frame, 16 sprites in one horizontal line
- 2 independent background layers.
- Background character: 16/64/256 indexed color mode.
- Background bitmap: 16/64/256 indexed color mode or 32768 colors for direct color mode
- Sprites are 16 colors.
- Two 256 colored-Color palettes, maximum display indexed color: 512
- Background vertical extension: x1/x1.5/x2
- Background horizontal line individual scrolling: -128~+127

7.2 Screen Structure

The display screen (displayed on the TV screen) resolution of the Graphic in VT1682 is 256 x 240 pixels. While the VRAM screen (the pattern defined in VRAM for background layer) is 256 x256 pixels.

7.2.1 VRAM Screen Structure

To display a background on the TV screen, the pattern in the VRAM should be defined first. There are two kinds of screen structure in VRAM, and they are character and bitmap mode. In character mode, the screen is partitioned into character blocks. In bitmap mode, the screen is partitioned line by line.

7.2.1.1 Character Mode

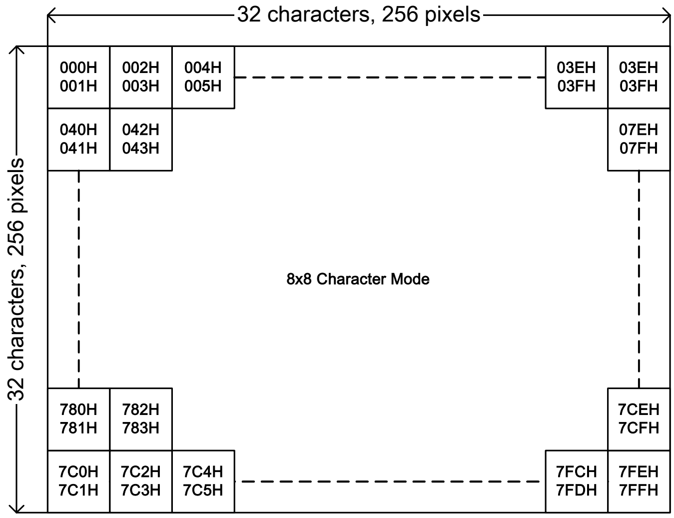
Each character in the background requires two bytes to define. As shown in the following diagram, these two bytes include the 12 bits character vector and 4 bits palette bank parameters. Vector = 0 means the transparent character. Please reference the Graphic Addressing Mode section and Color Palette section for the detail description about these parameters.

LowByte	Vector[7:0]	
HighByte	PalBank[3:0]	Vector[11:8]

Different character size would require different numbers of character pattern to define a screen as shown in the following table.

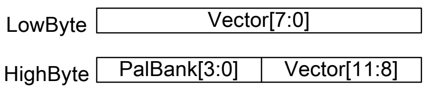
Charcter Size (H x V)	Required Pattern Number	Required VRAM (Byte)
8 x 8	32 x 32	2048
8 x 16	32 x 16	1024
16 x 8	16 x 32	1024
16 x 16	16 x 16	512

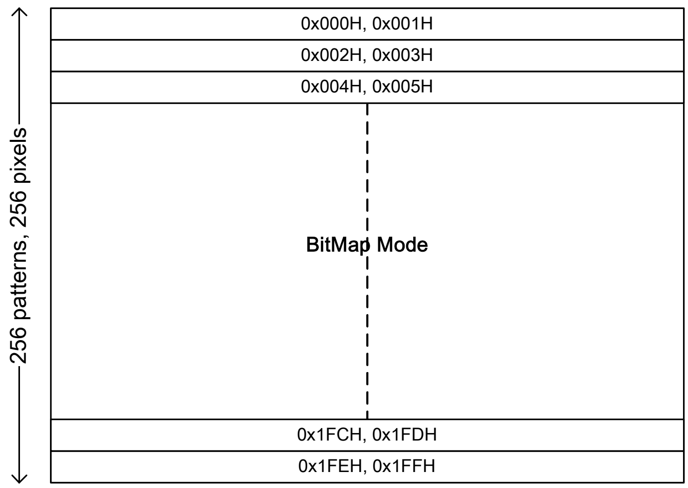
Following example is 8x8 character mode which requires 32 x 32 patterns to define a screen.



7.2.1.2 Bitmap Mode

In this mode, the screen in partitioned into 256 lines (patterns) and each line requires 2 bytes to define, as shown in the following diagram. Vector = 0 means transparent line in bitmap mode but not in high color mode.





7.2.1.3 VRAM Management

There is a 4Kbytes VRAM for the background layers. In character mode, different character size would require different VRAM area (**7.2.1.1 Character Mode**). Following table is the list of the memory map for the BK1 and BK2 with different character size and scrolling mode.

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,

In bitmap mode, there are only 240/256 vectors are required for BK1.

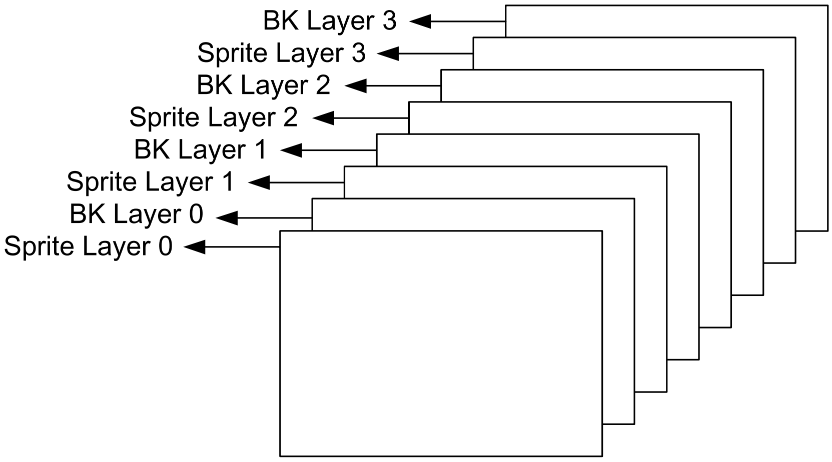
	V_scroll_en	H_scroll_en	Y[8]	X[8]	BK1
16x16	0	0	0	0	0x000 ~ 0x1FF
			0	1	0x200 ~ 0x3FF
			1	0	0x400 ~ 0x5FF
			1	1	0x600 ~ 0x7FF
	0	1	0	0	0x000 ~ 0x1FF, 0x200 ~ 0x3FF
			0	1	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
			1	0	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
			1	1	0x200 ~ 0x3FF, 0x000 ~ 0x1FF
	1	0	0	0	0x000 ~ 0x1FF, 0x200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

The TV / LCD display screen resolution is 256 x 240 pixels. Since the VRAM screen resolution is 256 x 256 pixels, partial of background is not visible on the TV screen. Please reference the "Graphic coordinate section to have the relationship between the Display Screen and VRAM screen.

7.3 Depth Layer

Each sprite or background has a depth layer parameter. It defines the relevant position of displayed objects. There are eight depth layers are valid in VT1682, 4 are for background and the other for sprites, their relationship is shown in the following diagram. The layer with less depth number would be displayed when several objects is overlapped. When two background layers have the same depth layer, background1 would be displayed. Sprites with lower sprite number (the mapped address in Sprite RAM, 0~239) would be displayed when they have the same layer number.



7.4 Graphic Pattern Format

7.4.1 Character Mode

Character 8x8 Mode

In 8x8 mode, the data sequence in horizontal is a,b,c,….h as the following diagram shown.

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

Character 8_H x 16_V Mode

The data sequence and Pattern Data are the same as 8x8 mode except the length of Pattern Data.

Character 16_H x 8_V Mode

The data sequence in horizontal is a,b,c,….h as the following diagram shown.

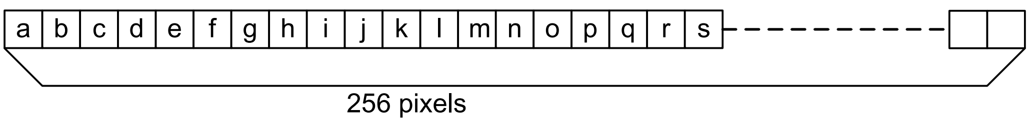
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
q	r	s	t	u	v	w	x								

Character 16_H x 16_V Mode

The data sequence and Pattern Data are the same as 16_H x 8_V mode except the length of Pattern Data.

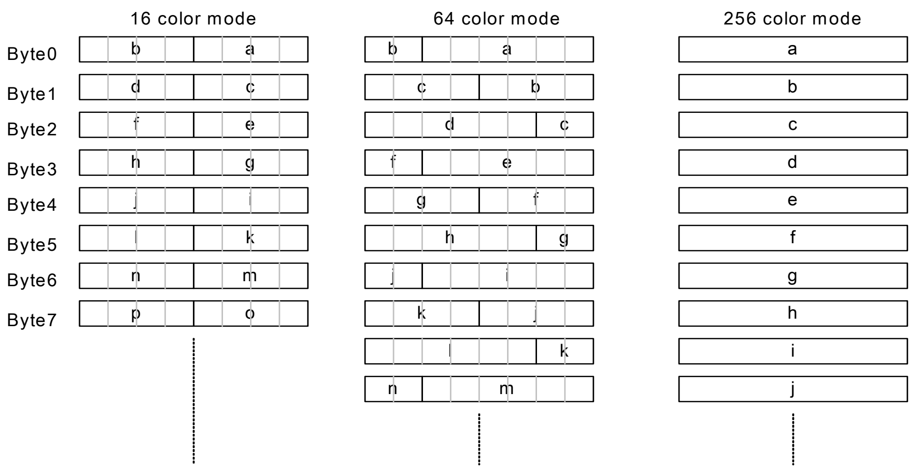
7.4.2 Bitmap Mode

In bitmap mode, the pattern data is defined line-by-line and its sequence is shown in the following diagram.



7.4.3 Color Mode

There are four kinds of color modes in VT1682, they are 16, 64, 256 and 32768 color modes. Background 1 could be one of 16, 64, 256 and 32768 color modes. Background2 could be one of 16, 64, 256 colors modes. And sprites are only valid in 16 colors mode. The Pattern Data with different color mode is shown in the following diagram.



7.4.4 Transparent Color

In the index color mode, 16/64/256 color modes, the transparent index would be 0. For example, the pixel with data 0x00 would be transparent in 16 color mode.

7.4.5 High Color Mode

In the High Color (32768 colors) Mode, each pixel requires two bytes, and its format is,

Low Byte

D7 – D5	D4 – D0
Green[2:0]	Blue[4:0]

High Byte

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

In this mode, like the content in Color Palette, each pixel has 5 bits R, G, B component. The MSB TRPT is used to be the transparent flag. When the TRPT flag is “1”, this pixel would become transparent. When a pixel is set to transparent, it means that the other pixel with lower depth layer would be display. If no other layer is under it, the color R, G, B would be display.

7.5 Color Palette

Color Palette is used to translate the indexed color into physical RGB color. There are two independent color palettes in VT1682. Each palette could display 256 colors selected from 32768 colors in a single display screen.

7.5.1 Palette Content

The color domain is R, G, B, and each component has 5 bits resolution. As shown in the following diagram, each color in Palettes requires 2 bytes to define, including 5-bits Red, 5-bits Green, 5-bits Blue and 1 DG flag for special effect. Please refer to the Graphic Special Effect section for the DG application.

Low Byte:

D7 – D5	D4 – D0
Green[2:0]	Blue[4:0]

High Byte:

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

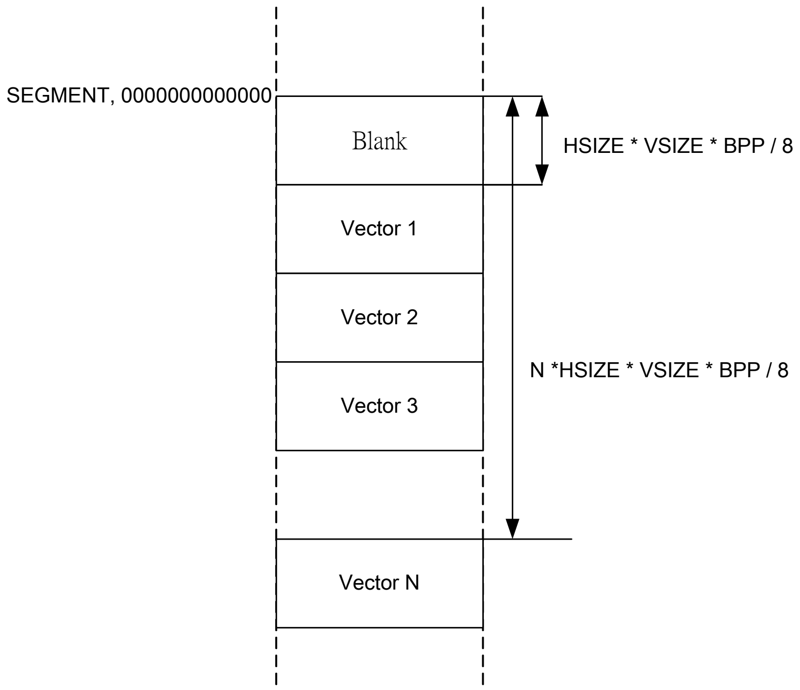
7.5.2 Palette Bank

There are 256 colors in Palette. For 16 colors mode, it could be partitioned into 16 banks. For 64 colors mode, it could be partitioned in 4 banks. The bank is defined by "PalBank" parameter. Each pattern in VRAM has the individual "PalBank" defined in D7:D4 of 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 GRAPHIC ADDRESSING MODE

The address of the Graphic pattern is based on the SEGMENT and Vector Number as shown in the following diagram. There are three sets of SEGMENT for sprites, background1 (BK1), and background2 (BK2).



Physical Address = (Segment << 13) + Offset

Offset = Vector x H_Size x V_Size x Color_Mode / 8

V_Size is 8 or 16,

H_size is 8 or 16

Color_Mode is 4 for 16 color mode, 6 for 64 color mode and 8 for 256 color mode

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

For example,

If BK1 character data is started with address 0x20000,

then segment = (0x20000) >> 13 = 0x10

If BK1 character is 8x8 with 256 colors with vector 3, then

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

The character data would be started from address 0x200C0.

Note1 : In bitmap or high color mode, both V_Size and H_Size are 16.

Note2 : In high color mode, Color_Mode is 8.

Note3 : All vectors in high color mode are even.

	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) for sprites

BK1_SEGMENT : Segmentation number (8KB bank) for BK1

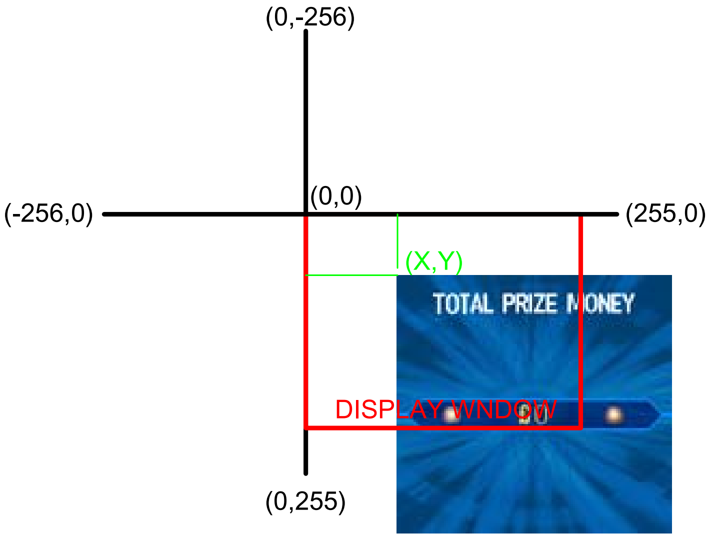
BK2_SEGMENT : Segmentation number (8KB bank) for BK2

7.7 Background Layer

There are two background layers in VT1682. They are background1 (BK1) and background2 (BK2).

7.7.1 Coordinate

X-axis of background layer is between –256 and 255, so is the Y-axis. (X, Y) is presented in 2's complement. When the X is positive (X[8] =0), the displayed background would scroll to right side. When the Y is positive (Y[8] = 0), the displayed background would scroll to bottom side, as shown in the following diagram.



7.7.2 Scrolling

BKx_X[8:0] and BKx_Y[8:0] is used to control the scroll of background layer. BKx_Scroll_En parameters provides 4-types of scrolling modes on each background layer.

	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 Frame Scroll

Each background has 9-bits X and Y coordinate, provides up to 512x512 scrolling function. Depending on the Scroll mode, BK1_Scroll_En and BK2_Scroll_En, defined in 0x2012 and 0x2016, the scrolling effect would be different.

Scrolling Mode	BK_Scroll_En	Effective X	Effective Y
Fix Page Mode	0	0 ~ 255	0 ~ 255
Horizontal Two Pages Mode	1	-256 ~ 255	0 ~ 255
Vertical Two Pages Mode	2	0 ~ 255	-256 ~ 255
Four Pages Mode	3	-256 ~ 255	-256 ~ 255

7.7.2.2 Line Scroll

VT1682 provides the background layer line-by-line horizontal scroll mode, as shown in the following diagram.

Line scroll mode could be applied on either BK1 or BK2 or both by programming the BK1_L_EN and BK2_L_EN in register 0x2020. 240 bytes scrolling vectors should be stored in the PRAM between { Scroll_Bank, 8'b0000_0000} and { Scroll_Bank, 8'b1110_1111} before enable the function. Each vector maps to each horizontal scan line. The value of the vectors should be between –128 and +127 and presented in 2's complement. The vector would be added with the Backgound layer's X. Please note that the independent with the vertical coordinate BK_Y of the background.

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

BK1_L_En : BK1 line scroll mode control

BK2_L_En : BK2 line scroll mode control

0 : Disable 1 : Enable

Scroll_Bank : PRAM bank for line scroll mode vectors.

{ Scroll_Bank, 8'b0000_0000} is for 1st line (top side), and { Scroll_Bank, 8'b1111_0111} is for the 240th line(bottom side).

7.7.3 Color Mode

Color mode in background layer could be 16, 64 or 256 colors. In other word, each pixel in background layer is presented in 4, 6 or 8 bits. Each character (line) has a individual Palette Bank that defined in the VRAM high byte bit7~bit4 as shown in the following table. Palette Bank (PalBank, refer to 7.5.2 Palette Bank), has two combination for different application as shown in the following table.

Background1

BK1_Pal_Sel	16 colors mode	64 colors mode	256 color mode
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]	----

Background2

BK2_Pal_Sel	16 colors mode	64 colors mode	256 color mode
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 Character Mode

BK1_Line = 0; BK1_HCLR=0;

Each background layer could be formed by either 8x8 character or 16x16 character. When the character mode is selected, the BKx_Line in 0x2013 and 0x2017 should be set to 0, and BKx_Size is for the selection of character size. It would take 32x32 words to define a 8x8 character mode in VRAM and 16x16 words for 16x16 character mode.

	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 Bitmap Mode

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

In bitmap mode, the background layer is separated into 256 horizontal lines. 256 words are required to define 256 horizontal lines in background layer. Please note that Bitmap mode is valid only in Background1.

7.7.6 High Color Mode

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

High Color mode is to display up to 32768 colors in background1 layer. In bitmap mode described above, the color modes would be 16, 64 or 256 colors for each background layer. In high color mode, each pixel is defined with 2-bytes whose format is shown in the following diagram. Please note that high color mode is valid only in background1 layer. When background1 is in high color mode, background2 should disable. Besides, the vectors in VRAM must be even.

Data Format:

Low Byte

D7 – D5	D4 – D0
Green[2:0]	Blue[4:0]

High Byte

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

7.7.7 Depth

Depth parameter is to define the display priority. Object with lower priority would be displayed on the screen. Each background layer has four depth layers. All characters (lines) in background could have a common depth parameter or individual depth as shown in the following table.

Background1

BK1_Pal_Sel	16 colors mode	64 colors mode	256 color mode
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]	----

Background2

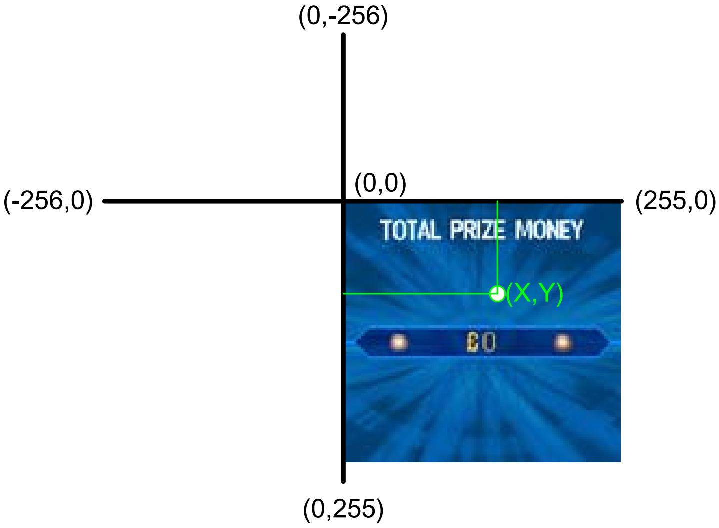
BK2_Pal_Sel	16 colors mode	64 colors mode	256 color mode
0	Depth = 0x2017[5:4]	Depth = 0x2017[5:4]	Depth = 0x2017[5:4]
	PalBank = VRAM[15:12]	PalBank = VRAM[15:14]	----
1	Depth = VRAM[13:12]	Depth = VRAM[13:12]	Depth = VRAM[13:12]
	PalBank = { 0x2017[5:4],VRAM[15:14] }	PalBank = VRAM[15:14]	----

7.8 Sprite Layer

There are 240 sprites in display screen. Each horizontal line could display up to 16 sprites. Each sprite has individual depth, vector, X, Y, palette bank, palette selection parameters.

7.8.1 Sprite Coordinate

Each sprite has 9-bits X and Y coordinate. The origin of the coordinate is the left-top corner of the display screen as shown in the following diagram



7.8.2 Sprite Size

Sprite size could be 8x8, 8x16, 16x8 or 16x16. All sprites have the equivalent character size defined by SP_Size in 0x2018.

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

SP_size : Sprite block size(V x H)

0 : 8x8   1 : 8x16

2 : 16x8   3 : 16x16

7.8.3 Sprite Control

Setting SP_EN in 0x2018 to enable the sprite display. If the display sprite number in horizontal line is more than 16, the overflow flag SP_ERR in 0x2001 would be “1”. This flag is modified line by line, and valid only in the horizontal blanking period.

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

SP_ERR : More than 16 sprites in single horizontal line.

0 : Not exceed,   1 : Exceed

7.8.4 Sprite Palette Selection

There are two color palettes in VT1682 (refer to **7.10 Palette Select**). Each sprite has the individual parameter, PalSel in Sprite RAM to select the Color Palette. Setting the SPALSEL would allow all sprites to display on Palette1 and Palette2 at the same time.

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

7.8.5 Sprite RAM data format

Each sprite requires 6 bytes to define its attribute, include vector, X, Y, depth layer, vertical flip, horizontal flip, palettes select internal pattern mode. Its format is shown in the following table. When the DMA is used to transfer the sprite RAM data, the transmit source address sequence would be

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

And the destination Sprite RAM address sequence would be

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

Sprite data format is Sprite RAM

	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 is the sprite index number in the Sprite RAM between 0 and 239.

7.9 CCIR Layer

VT1682 has two sets of CCIR protocol to input the graphic image named CCIR layer. Please reference the "CCIR Protocol" section for the signals interconnection and register setting. CCIR could be treated as the third background layer but only valid in Palette2. Like background, CCIR layer has 8 depth layers (CCIR_Depth) which is the combination of sprite and background depth layer.

7.9.1 CCIR Color Effects

There are several color special effects for the CCIR layer, such as gray color, halftone color, and color mask. Gray color function is to disable the chrominance in the CCIR layer, and this function is valid only when YUV_RGB in 0x202A is 0. Halftone function is to half the chrominance component. There are three RGB color mask to generate different color output in color mask function. CCIR layer also has some other special applications such as image sensor the function would be described individually in the following section.

	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	Unused	Touch_EN	CCIR_TH			

R_MSK : CCIR special effect, red component mask control.

0 : Normal 1 : No red component

G_MSK : CCIR special effect, green component mask control.

0 : Normal 1 : No green component

B_MSK : CCIR special effect, blue component mask control.

0 : Normal 1 : No blue component

HalfTone : CCIR special effect, half the input color.

0 : Normal 1 : Half

Gray : CCIR special effect, change the input image to gray-levels.

0 : Color 1 : Gray

7.9.2 CCIR Image Capture

CCIR image could be captured and stored in the external SRAM. In the capture period, the sprite image would not be affected. Capture function would operate correctly only when the VT1682 is in SLAVE mode. When the Capture command in 0x2000 is set, the image through CCIR interface would be strobe and store in external SRAM. The Capture command should be enable/disable in the VBLANK period. The captured image could be either high color or 16-gray-level image.

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

Capture : Capture the image at CCIR interface

0 : No operation 1: image capture

7.9.2.1 High Color Image

The captured file format is 256 x 240 pixels, each pixel uses one word (two bytes), and each word includes red (D14-D10), green (D9-D5) and blue (D4-D0). The following registers should be set before entering capture function.

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;

The captured image would be stored in the Background1 high color mode pattern area. So the BK1 vectors in VRAM should be set in the high color mode format, 240 line vectors.

7.10 Palette Select

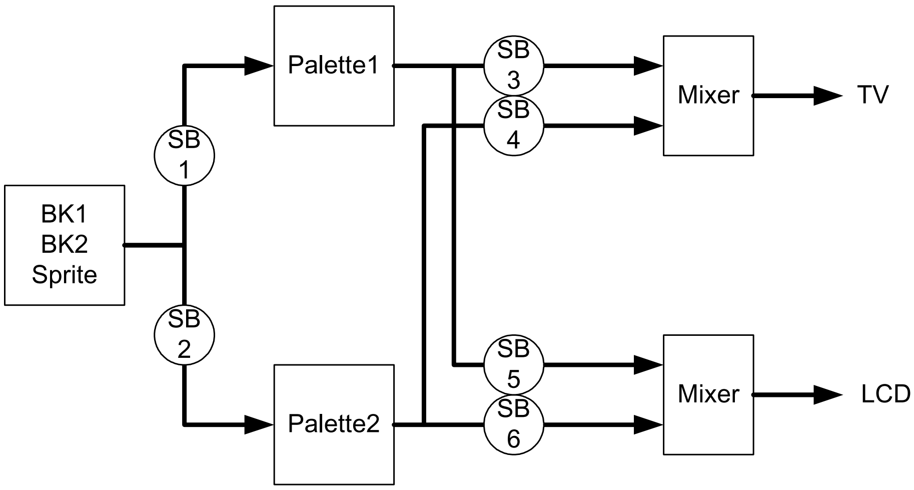
There are two 256 colors palettes can support up to 512 colors display on a single screen.

BK1, BK2 and sprites can choose the palette individually.

As shown in the following diagram, SB1 and SB2 are used to select the palette.

When the SB1 is enabled, the palette 1 is selected.

When the SB2 is enabled, the palette 2 is selected.



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: BK1 Palette select

0 : BK1 disable　　1 : BK1 uses palette1

2 : BK1 uses palette2　　3 : BK1 uses palette1 and palette2

BK2_Pal_Sel, BK2 Palette select

0 : BK2 disable　　1 : BK2 uses palette1

2 : BK2 uses palette2　　3 : BK2 uses palette1 and palette2

Sprite :

Each sprite has individual palette select bit , that is controlled by “PalSel” in the Sprite RAM and SPALSEL in 0x2018. When SPALSEL is high, all SB1 and SB2 of sprites would be active, the PalSel would disable. When SPALSEL is low, the PalSel would define the SB1 and SB2. When PalSel=0, the SB1 is selected, otherwise the SB2 is selected.

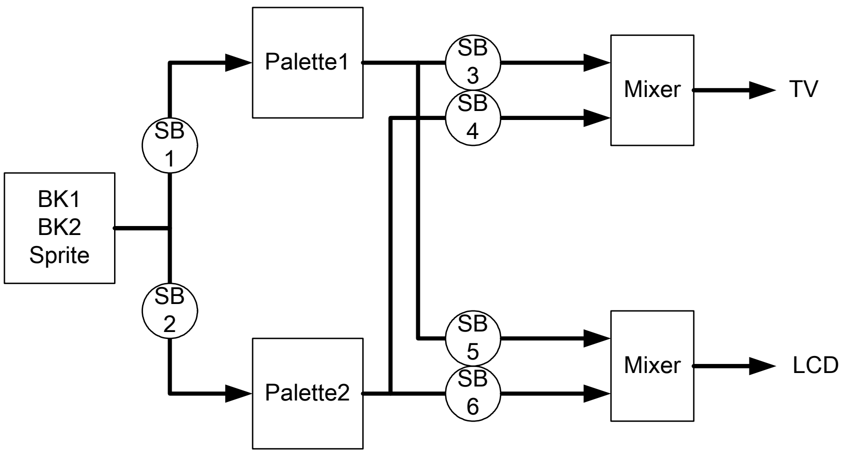
	D7	D6	D5	D4	D3	D2	D1	D0
0x2018					SPALSEL	SP_EN	SP_SIZE	

Sprite data format is Sprite RAM

	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 Output Select

Since there are two 256 colors palettes output (Palette1 and Palette2), they can be redirected to two output protocols (LCD or TV) individually. As shown in the following diagram, SB3, SB4, SB5 andSB6 are used to select the output device. When SB3 is enabled, all data through Palette1 would be displayed on TV. When SB5 is enabled, all data through Palette1 would be displayed on LCD. SB3 and SB5 could be controlled individually. It's the same for the SB4 and SB6. When the SB3 and SB4 are both enabled, data through Palette1 and Palette 2 would be mixed at the TV mixer. Two mixing methodologies are valid. One is the basic mixing (overlap), based on the "depth layer". The other is the color blending. It would blend the data from Palette1 and Paletted2 with 50% blending weight. Please reference the "Graphic Blend Control" for detail description.



SWITCH NAME	PORT
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: Palette1 output select

0 : output disable, 1 : output to LCD only

2 : output to TV only 3 : output to TV and LCD

Pal2_Out_Sel: Palette2 output select

0 : output disable, 1 : output to LCD only

2 : output to TV only 3 : output to TV and LCD

Blend1 : TV output blending enable

0 : Overlapped, 1 : blending

Blend2 : LCD blending enable

0 : Overlapped, 1 : blending

7.12 Graphic Vertical Scaling

Two vertical scaling factors are valid for BK1 and BK2 individually. They are x1, x1.5 and x2 as shown in the following diagrams, selected by the port 0x2019.

Gain = 1 Gain = 1.5 Gain = 2

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

BK2_Gain : BK2 (vertical) amplification gain

BK1_Gain : BK1 (vertical) amplification gain

0 : x1 1 : x1

2 : x1.5 3 : x2

7.13 Light Gun Interface (Pulse Latch)

VT1682 provides two sets of Light-Gun-Interface (Pulse Latch function) for dual light gun applications. These two input pulse are through **XIOE0** and **XIOE1**. The first rising edge of the **XIOE0** right after the 0x2023 is written whose coordinate would be latched in 0x2024 and 0x2025. It's the same for the **XIOE1**, 0x2026 and 0x2027. The value of 0x2024 and 0x2026 should be between 0 and 119 while 0x2025 and 0x2027 should be between 0 and 126. When the coordinate is (0,0), it means that no input rising edge in detected.

The operation procedure should be

1. Set **XIOE0** to input mode. (Set **XIOE1** to input mode, if necessary)
2. Read 0x2024 and 0x2025 (0x2026 and 0x2027 if necessary) in VBLANK NMI period.
3. (0x2024 *2) is physical Y position and (0x2025 *2) is the X position of Gun1.
4. (0x2026 *2) is physical Y position and (0x2027 *2) is the X position of Gun2.
5. Write any value to 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 Graphic Memory Access

Graphic memory includes the Sprite RAM (Sprite pool) and VRAM (Background and Palette). It could be updated rapidly by DMA function or byte-wised by CPU. The DMA function would not be illustrated here, please refer to the "DMA" section.

7.14.1 Sprite RAM

Sprite Memory is the pool to store the sprite attributes. SPRAM_ADDR[10:0] is the Sprite RAM address for the CPU access. Writing data into 0x2004 would write the data in the Sprite RAM with address SPRAM_ADDR. Reading 0x2004 would get the data in Sprite RAM with address SPRAM_ADDR. SPRAM_ADDR would increment automatically while 0x2004 is written.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2002						SPRAM_ADDR[2:0]		
0x2003	SPRAM_ADDR[10:3]							
0x2004(W)	SPRAM_DATA[7:0]							
0x2007(R)	SPRAM_DATA[7:0]							

Data Format in the Sprite RAM

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 is sprite number between 0 and 239.

7.14.2 VRAM

VRAM access is similar to the Sprite RAM. Writing data into 0x2007 would write the data in the VRAM with address VRAM_ADDR. Reading 0x2007 would get the data in VRAM with address VRAM_ADDR. VRAM_ADDR would increment automatically while 0x2007 is written.

	D7	D6	D5	D4	D3	D2	D1	D0
0x2005	VRAM_ADDR[7:0]							
0x2006	VRAM_ADDR[15:8]							
0x2007(W)	VRAM_DATA[7:0]							
0x2004(R)	VRAM_DATA[7:0]							

VRAM address map

Address	Content
$0000 – $0FFF	BacKground Page
$1000 – $1BFF	Reserved
$1C00 – $1DFF	Color Palette2
$1E00 – $1FFF	Color Palette1

7.15 Special Effect

7.15.1 Dig Color

Color Palette in VT1682 provides dig color function that would remove the color from Color Palette1 to display the related pixel on color Palette2. It's the same for Color Palette2 and Color Palette1. DG flag could be treated as the dig function on the graphic after Color Palette. Different from the transparent color, DG flag is to remove colors of all layers and show the color on the other palette. Transparent color is to show the color on the same palette with lower depth layer. When the DG is "0", the pixel would be displayed with color[R, G, B] with related index. When DG is "1", the displayed color would be [R, G, B] or the pixel color from the other palette, refer to the "Palette Select" section. If there is no display pixel from the other palette, the color [R, G, B] would be displayed. Otherwise, the pixel from the other palette would be displayed.

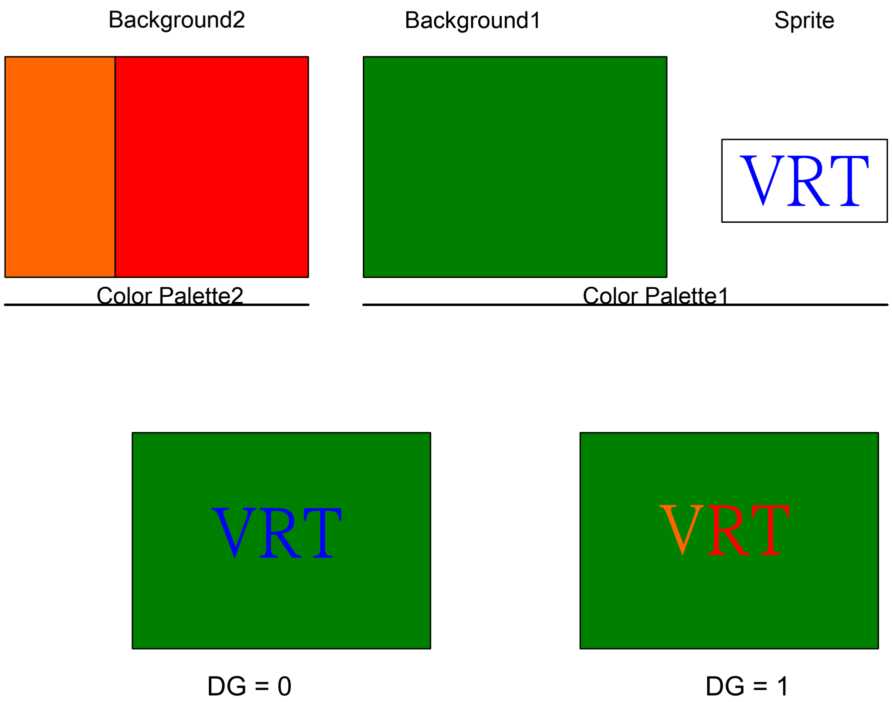
Low Byte:

D7 – D5	D4 – D0
Green[2:0]	Blue[4:0]

High Byte:

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

Following diagram is an example to illustrate the Dig Color Effect. Assume that BK1 and sprite are from Color Palette1, and BK2 from Color Palette2. When the Dig Color flag of the sprite blue color on Color Palette1 is zero, the output of the Color Palette1 would be blue word “VRT”. When the Dig Color is one, the “VRT” would become the color of Color Palette2 (BK2 in this example).



7.15.2 Blending Effect

Blending effect is the color mixing of Color Palette1 and Color Palette2. The mixing ratio is 50% Color Palette1 and 50% Color Palette2. There are two blending effect, one is for TV out, the other is for LCD.

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

Pal1_Out_Sel: Palette1 output select

0 : output disable, 1 : output to LCD only

2 : output to TV only 3 : output to TV and LCD

Pal2_Out_Sel: Palette2 output select

0 : output disable, 1 : output to LCD only

2 : output to TV only 3 : output to TV and LCD

Blend1 : TV output blending enable

0 : Overlapped, 1 : blending

Blend2 : LCD blending enable

0 : Overlapped, 1 : blending

7.16 Vertical Blanking (NMI)

The NMI of the main CPU and Sound CPU are both vertical blanking. To enable the NMI of Sound CPU or CPU, the NMI_EN in 0x2000 should be set. When the NMI disables, main CPU could read the VBLANK flag in 0x2001 to recognize the vertical blanking period

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

NMI_EN : VBLANK NMI enable control

0 : Disable, 1 : Enable

VBLANK : Vertical blank active period, and read to clear NMI

0 : Non-VBLANK period, 1 : VBLANK period

7.17 Graphic Display Screen Size

The display graphic resolution is 256 x 240 pixels in normal operation. For some particular horizontal scrolling control, VT1682 provides another display screen size, 248 x 240. The left 8-pixels, 1^{st} ~ 8^{th} would not be displayed on the screen.

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

SP_INI : Sprite display screen size

BK_INI : Background display screen size

0 : 256 x 240 pixels 1 : 248 x 240 pixels

8. I/O

There are 56 I/Os are valid in VT1682, 40 of them are controlled by the main CPU, and the others are for Sound CPU. These 40 I/O are IOA[3:0], IOB[3:0], IOC[3:0], IOD[3:0], IOE[3:0], IOF[3:0], UIOA[7:0] and UIOB[7:0]. Each of them has different attribute as shown in the following table. Besides, all these I/O ports are shared pin, they could be GPIO or the LCD, UART, SPI, I2C, …… interface.

	Output High	Output Low	Input Floating	Input w/ Pull High Resistor	Input w/ Pull Low Resistor	Shared Function
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 is shared with LCD interface. When the LCD controller is used, the IOA_ENB and IOA_OE should be set to 1. In the GPIO(IOA_ENB = 0) mode, IOA could be either output high, output low, input with pull-high resistor or input with pull-low resistor mode as listed in the following table.

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

	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 is shared with LCD interface. When the LCD controller is used, the IOB_ENB and IOB_OE should be set to 1. In the GPIO(IOB_ENB = 0) mode, IOB could be either output high, output low, input with pull-high resistor or input with pull-low resistor mode as listed in the following table.

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

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

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 is shared with LCD and UART interface. When the LCD controller is in LTPS or ANALOG mode or UART interface is used, the IOC_ENB and IOC_OE should be set to 1. In the GPIO(IOC_ENB = 0) mode, IOC could be either output high, output low, input with pull-high resistor or input with pull-low resistor mode as listed in the following table.

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

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

	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 is shared with SPI interface. When the SPI interface is in Master mode, IOD_ENB and IOD_OE should be set to 1. When the SPI is Slave mode, IOD_OE should be set to 0. In the GPIO (IOD_ENB = 0) mode, IOD could be either output high, output low, input with pull-high resistor or input with pull-low resistor mode as listed in the following table.

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

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

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

	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 is shared with LCD DAC output and light Gun interface. When the LCD controller ANALOG, LTPS mode or the LCD DAC, Light Gun function is used, IOE_OE should be set to 0. When the ADC microphone mode is used, IOEOE3 in 0x211E should be set to 0. Otherwise, IOE could be either output high, output low, input floating mode as listed in the following table.

	IOEOE = 1	IOEOE = 0
XIOE[0,1,2]	LCDACEN=0, OUTPUT IOE_O LCDACEN =1, OUTPUT LCDAC	LCDACEN=0, INPUT floating 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 is shared with ADC input and IIC interface. When the ADC is used, the related IOF_OE[x] should be set to 0. When the IIC is used, IOFENB should be set to 1. Otherwise, IOF could be either output high, output low, input floating mode as listed in the following table.

	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, INOUT: 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 is bit-wise attribute control, each UIOA has an individual direction (IN / OUT) and attribute (pull-high, pull-low / floating) parameters as listed in the following table. XUIOA are shared with the LCD and CCIR interface. When the CCIR interface is active, UIOA_DIR should be set to input float mode.

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 Shared Function

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*

*For the detail description of LCD:Dx, please reference the LCD section.

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 is bit-wise attribute control, each XUIOBx has an individual direction (IN / OUT) and attribute (pull-high, pull-low / floating) parameters as listed in the following table. XUIOB are shared with the External IRQ, LCD and CCIR interface. When the CCIR interface is active, UIOB_DIR[2:0] should be set to input float mode. When External IRQ is active, XUIOB3 should be set to input with pull-high resistor mode.

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 Shared Function

OUTPUT

	UIOB_MODE[1]=0	UIOB_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 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 provides 4 Direct Memory Access (DMA) paths to speed up the data transfer. They are external memory to program RAM, external memory to VRAM, program RAM to VRAM and program RAM to external memory. Program RAM includes the 4KB main CPU local PRAM and 4KB shared RAM and the VRAM include the Sprite RAM, Color Palette and Background VRAM. They are controlled by the registers 0x2122 ~ 0x2128 as shown in the following table. The base of DMA_DT_Addr, DMA_SR_Addr, and DMA_SR_Bank is "byte", but the DMA_Number is base on the "Word"(Double Byte). DMA transfer is issued by the writing of the DMA_Number. The DMA whose destination is VRAM would not be started until the vertical blank (VBLANK) period. In other word, if you issue a VRAM DMA in non-VBLANK period, the DMA would not active until the beginning of the VBLANK. The CPU would be halt in the DMA period. The DMA_Status is the DMA status flag for you to monitor the operation of the DMA. When you transfer data to VRAM, not only the DMA registers here you have to program but also the register 0x2002, 0x2003, 0x2005 and 0x2006. When the DMA destination is Sprite RAM, port 0x2002 and 0x2003 is used to define the destination address. While destination is VRAM, the 0x2005 and 0x2006 should be defined.

	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®								DMA_Status
0x2128(W)							DMA_SR_Bank[24:23]	

DMA_DT_Addr : DMA destination address.

DMA_SR_Addr : DMA source address.

Note : DMA_SR_Addr[0] and DMA_DT_Addr[0] should be zero.

DMA_Number : the number of DMA transfer "word".

DMA_Status : DMA status flag.

0 : DMA ready 1 : DMA busy.

DMA Source	DMA Target	Source Start Address	Target 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. POWER SAVING

To save the power consumption, most modules in VT1682 could be turned off separately, include Sound CPU, SPI, UART, TV encoder, LCD controller, LVD module, Video DAC, Audio DAC, LCD DAC, PLL and ADC. In the default mode, all these modules are disable.

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

LCD_ON : LCD controller module enable control.

TV_ON : TV encoder module enable control

UART_ON : UART enable control

SPI_ON : SPI module enable control

LVDEN : Low Voltage Detect module enable control

VDAC_EN : Video DAC module enable control

ADAC_EN, :Audio DAC module enable control

PLL_EN : Phase Lock Loop module enable control

LCDDACEN : analog LCD DAC module enable control

SCPU_ON : Sound CPU enable control

0 : Disable 1 : Enable

SCPRN : Sound CPU Reset control

0 : Reset Sound CPU 1 : Normal operation

11. INTERRUPT

There are 6 IRQ source with 5 IRQ vectors in VT1682. These IRQ source are external IRQ, Timer IRQ, SPCU IRQ, UART IRQ and SPI IRQ. Their IRQ vectors are listed in the following table. When IRQ occurs at the time, the service priority would be Ext_IRQ > Timer_IRQ > SCPU_IRQ > UART_IRQ > SPI_IRQ.

Vector Name	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 Non Maskable Interrupt (NMI)

The NMI source is the graphic vertical blanking (VBLANK). The detail description could be referred in "7.17 Vertical Blanking (NMI)" section.

11.2 External IRQ

External IRQ is the falling edge trigger from XUIOB3. Before the External IRQ is enabled, remember to set the XUIOB3 to input with pull-high resistor mode. When the Ext_MSK in 0x2121 is set to 1, falling edge on the XUIOB3 would make CPU enter the IRQ service routine defined in 0x7FFFE and 0x7FFFF. Writing "1" to Ext_MSK in service routine would clear the IRQ flag to wait for the next trigger.

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

Ext_MSK, external IRQ enable control

0 : IRQ enable 1 : IRQ disable

Writing "1" to Ext_MSK would clear the IRQ flag.

11.3 Timer IRQ

Timer IRQ vector is 0x7FFF8 and 0x7FFF9, and this IRQ is mask-able by IRQ1_MSK in 0x2121.

11.3.1 Timer IRQ

Please refer to **"6.2 Timer"** section for the detail timer operation. To enable the Timer IRQ, both TMR_IRQ in 0x2102 and IRQ1_MSK in 0x2121 should be set to "1" first. When the timer IRQ occurs, writing 0x2103 (Timer_IRQ_Clear) would clear the current timer IRQ flag to wait for the next timer 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_IRQ: Timer IRQ enable control.

0 : disable 1 : enable

Timer_IRQ_Clear : Timer IRQ clear control, write any data to clear Timer IRQ.

11.4 Sound CPU IRQ

IRQ is used for the communication between Sound CPU and main CPU. Both of them could send on IRQ to each other.

11.4.1 Receive Sound CPU IRQ

When Sound CPU sends the IRQ to main CPU, the CPU would enter the service routine defined by 0x7FFF6 and 0x7FFF7, if the SPU_MSK in 0x2121 is enabled. Please refer to "14.3.2.2 **Communicate with Main CPU**" section for the Sound CPU IRQ. In the Sound CPU IRQ service routine, writing "1" to SPU_MSK would clear the IRQ flag till the next Sound CPU IRQ.

11.4.2 Transmit Sound CPU IRQ

When main CPU is going to have the request from Sound CPU, it could send an IRQ to interrupt Sound CPU. Writing a positive pulse to SCPU_IRQ in 0x211C generates this IRQ pulse.

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

11.5 UART IRQ

There are two IRQ source from UART, one is RX (receive) IRQ and the other is TX (transmit) IRQ. RXIRQEn, TXIRQEn in 0x2119, controls these two IRQ. UART IRQ is also mask-able by the UART_MSK in 0x2121. Only when UART_MSK is "1" and one of the RXIRQEn and TXIRQEn is enabled, the UART_IRQ would occur. To distinguish from RX and TX IRQ, just read the flag TX_Status and RX_Status from 0x211B. Please refer to "**5.3 UART interface**" for detail UART setting.

	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 is mask-able by SPI_MSK in 0x2121. When the SPI is ready to transmit or ready to receive, the IRQ would occur.

12. EXTERNAL MEMORY CONTROL

External memory bus, XA[23:0], XD[15:0], XROMCSB, XRAMCSB, XRAMRWB, XROMOEB are programmable to enter the tri-state mode for particular application. Two types of external memory access time, 180ns and 80ns, are allowed in VT1682. There are up to three external memory CSB are valid to save the application BOM cost.

12.1 External Memory Bus Access Time

In the default mode, the CPU performance would be interrupted by the Graphic. Higher graphic quality would lead to the lower CPU performance. To solve this problem, VT1682 allows user to change the higher speed external memory device (SRAM / FLASH / ROM) to improve the CPU performance. Besides, VT1682 can change the memory bus (XA, XD, XROMCSB, XRAMCSB, XRAMRWB and XROMOEB) into tri-state to allow the external device access the memory. In this period, CPU can work on the PRAM, and show the simple graphic on the VRAM without turning off the display.

Bus Access Frequency

Two kinds of access speed are valid in VT1682 selected by "Double" in 0x2105.

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

Double	External Memory Access Time
0	180 ns
1	80 ns

12.2 Bus Tri-state Control

VT1682 could isolate the system bus, XA, XD, XROMCSB, XRAMCSB, XRAMRWB and XROMOEB. In this mode, external device is allows to access the VT's external memory. When bus is in tri-sate, there is a 50Kohm pull-high resistor in XA, XROMCSB, XRAMCSB, XRAMRWB and XROMOEB.

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

0 : Normal operation 1 : XA tri-state mode

12.3 External Memory Chip Select Signal Control

VT1682 could provide up to 3 CSB control pins for external memory, they are XROMCSB, XRAMCSB and XUIOB7. There are four memory map modes for these three pins selected by CS_Control in 0x210B. When the XUIOB7 is used as External memory CSB, remember to modify the attribute, UIOB_SEL[7] = 1 and 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	$000000 ~ $FFFFFF
2	$000000 ~ $3FFFFF	$40000 ~ $7FFFFF	$800000 ~ $FFFFFF
3	$000000 ~ $1FFFFF	$200000 ~ $3FFFFF	$400000 ~ $7FFFFF

13. JOYSTICK PROTOCOL

VT1682 could provide up to two sets of 3-wired joystick protocol. Each set includes SCK, SDO and SDI. The clock SCK would be output at XUIOB4 and XUIOB5. When the port 0x2129 is read, a 200ns-wide positive pulse would be generated at XUIOB4. It's the same for 0x212A and XUIOB5. The setting of the XUIOB4 and XUIOB5 are listed in the following table. SDI and SDO could be anyone of the UIOA or UIOB except UIOB[5:4].

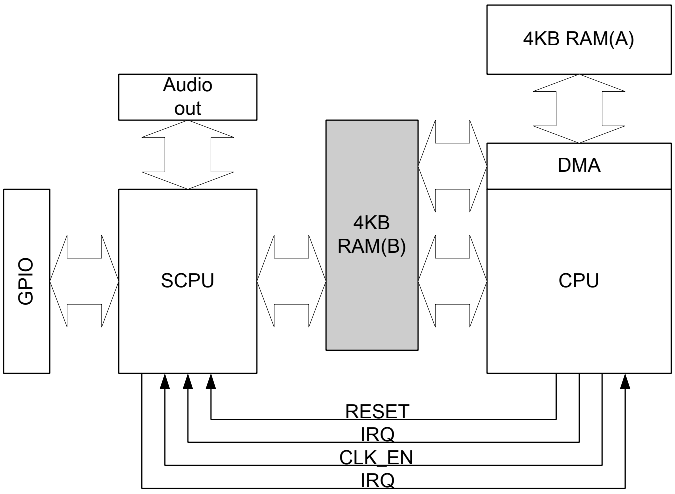
0x2129®	Send_JOY_CLK		
0x2129®	UIOA_DATA_IN		
0x2148(W)	UIOB_SEL[7:3]		UIOA_MODE

PIN	SIGNAL NAME	SETTING
XUIOB4	JOY_CK	UIOB_SEL[4] = 1, UIOB_DIR[4] = 1
XUIOB5	JOY_CK2	UIOB_SEL[5] = 1, UIOB_DIR[5] = 1

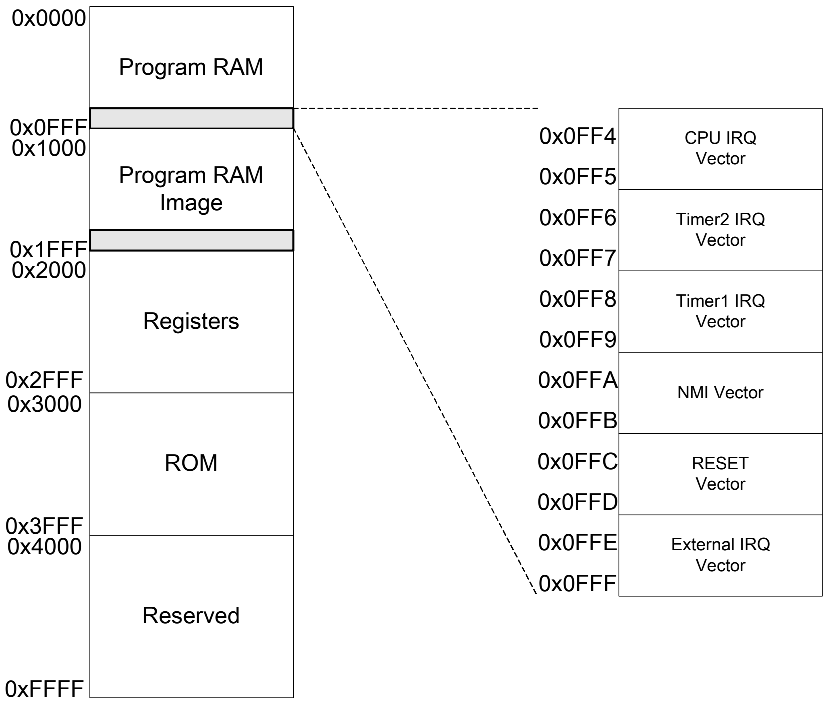
14. SOUND CPU

There are two 6502 CPU in VT1682, one is the main CPU; the other is Sound CPU (SCPU). SCPU operates at 21.4772MHz in NTSC and 26.6027MHz in PAL. SCPU shares a 4KB SRAM with main CPU. Main CPU and SCPU could communicate through the shared RAM or the IRQ signals.

14.1 Block Diagram



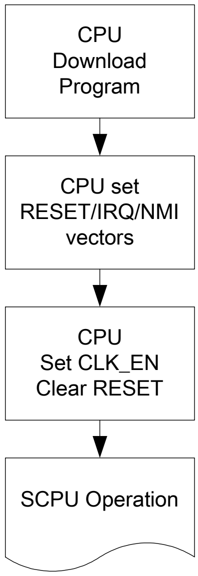
14.2 Memory Map



14.3 Operation Procedure

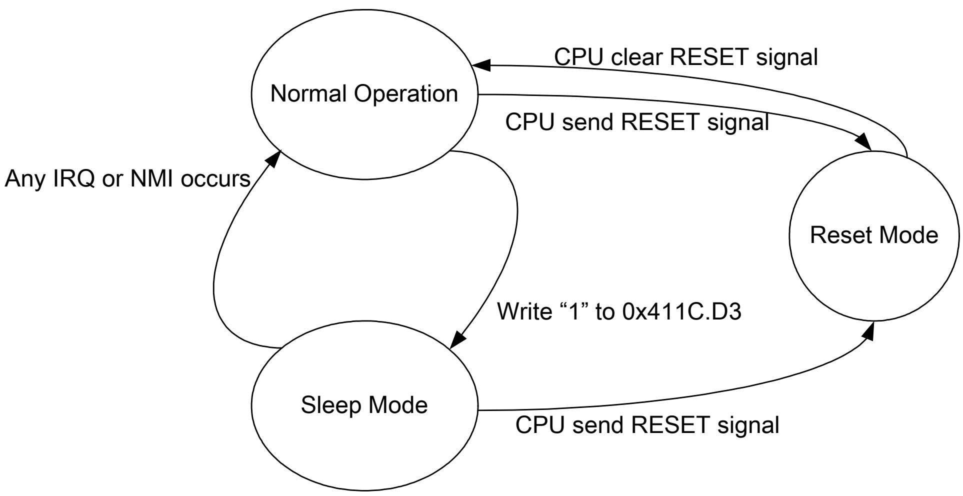
14.3.1 Power-On / Reset procedure

Since the main CPU controls SCPU, all instructions in following diagram are for main CPU in this section. The power-on (reset) procedure of the SCPU is shown in the following diagram. Main CPU has to download the SCPU program into the Share RAM first, and set the SCPU vector between 0x1FF4 ~ 0x1FFF in main CPU memory. Then enable the SCPU clock in 0x2106, clear the reset flag, SCPU would start to work.



14.3.2. SCPU Operation Flow

There are three operation modes in SCPU as shown in the following diagram.



14.3.2.1 Sleep Mode

SCPU can enter the sleep mode to save the power consumption. The SCPU is wakeup by the IRQ, they are Timer IRQ, CPU IRQ, NMI and external IRQ.

Note that the wakeup IRQ mask should be open.

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

SLEEP : Write “1” to enter sleep mode

NMI_EN : NMI mask

0 : NMI is not wakeup source 1 : NMI is wakeup source

ExtMask : External IRQ mask

0 : External IRQ is not wakeup source 1 : External IRQ is wakeup source

14.3.2.2 Communicate with Main CPU

There are two ways for the SCPU to communicate with main CPU. The first one is to define a global memory area in the share RAM. The shared memory area is between 0x1000 and 0x1FFF of SCPU and main CPU. The other is IRQ. Main CPU could write a positive pulse to SCPUIRQ in 0x211C to SCPU. Similarly, SCPU could write a positive pulse to IRQ_OUT in 0x211C to main CPU. On the SCPU’s CPU_IRQ service routine, SCPU must read the port 0x211C to clear the IRQ flag for the next CPU IRQ.

14.4 Timer

There are two sets of timer for SCPU.

14.4.1 TimerA

Write Timer_A_PreLoad to initialize the timer frequency. Writing 0x2101 would reload the TimerA_Preload into timer, so 0x2100 should be written before 0x2101.

	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]: Timer IRQ period definition.

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 enable control.

0 : disable 1 : enable

TMRA_IRQ: TimerA IRQ enable control.

0 : disable 1 : enable

TimerA_IRQ_Clear : TimerA IRQ clear control, write any data to clear TimerA IRQ.

14.4.2 TimerB

Write Timer_B_PreLoad to initialize the timer frequency. Writing 0x2111 would reload the TimerB_Preload into timer, so 0x2110 should be written before 0x2111.

	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]: Timer IRQ period definition.

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 enable control.

0 : disable 1 : enable

TMRB_IRQ: TimerB IRQ enable control.

0 : disable 1 : enable

TimerB_IRQ_Clear : TimerB IRQ clear control, write any data to clear TimerB IRQ.

14.5 Audio Output

There are two ways to output the Audio from VT1682. The first one is the IIS interface as shown in the next section. The other is the embedded audio DAC. It is 12-bits precision, so only the 12 MSB [15:4] would be outputted. Please remember to turn on the audio DAC by main CPU before you are going to output the audio through 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 Left channel output data.

Audio_DAC_R[15:0] : Audio DAC Right channel output data.

14.6 IIS Interface

VT1682 provides the 16-bits dual channel IIS output for the higher audio quality output applications. IIS signals are output at XSCPUIOB6, XSCPUIOB5 and XSCPUIOB4. To enable the IIS interface, please make sure that IIS_EN has to set to “1” and the SCPUIOB[6:4] have to set to output mode.

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

IIS_Mode : IIS format selection

0 : MSB justified format 1 : IIS-bus format

IIS_EN : IIS interface enable control.

0 : Disable 1 : Enable

14.7 IRQ Control

There are four IRQ in SCPU; they are external IRQ, Timer-A IRQ, Timer-B IRQ and CPU IRQ. Their vector addresses are listed in the following table.

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

Each IRQ has a mask flag except the CPU_IRQ. The mask of Timer-A and Timer-B is in 0x2102 and 0x2112, and the mask of NMI and External IRQ is in 0x211C. When the mask is "0", the IRQ would not be detected by SCPU. IRQ_OUT is the IRQ output to main CPU for the communication between two CPUs. Writing a positive pulse to IRQ_OUT would interrupt the main CPU. Reading the 0x211C would clear the CPU IRQ, and this must be done in the CPU IRQ service routine. Otherwise, the next CPU IRQ would not occur. EXTIRQSel in 0x211C is to select the external IRQ source; it could be the low level trigger from pin XSCPUIOB7. NMI in main CPU (0x2000) must enable if the NMI in SCPU is used.

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

IRQ_OUT : send IRQ to CPU.

0 : clear IRQ 1 : set IRQ

SLEEP : sleep mode enable, write "1" to enter sleeping mode.

Note : Any issue of IRQ would wake up CPU.

EXTIRQSel : External IRQ source select control

0 : SCPUIOB[7] falling edge 1 : External CPU write transfer complete

NMI_EN : NMI enable control

0 : disable 1 : enable

ExtMask : IRQ mask for External IRQ source

0 : disable External IRQ 1 : enable External IRQ

Clear_CPU_IRQ : clear IRQ flag from CPU, write any data to clear IRQ flag.

14.8 Enhanced Arithmetic Unit --Multiplier and Divider

SCPU has a dedicated ALU for the multiplication and division. The multiplier is 16 bit by 16 bits and the division is 32 bits by 16 bits. The multiplication requires 16 CPU clock cycle to complete the operation while the division requires 32.

14.8.1 Multiplier (16x16)

The multiply operation is,

ALU_ Multi _operand6, ALU_ Multi _operand5

XI) ALU_operand2, ALU_operand1

= ALU_out4, ALU_out3, ALU_out2, ALU_out1

The operation is started when the ALU_Multi_operand6 is written. The value in ALU_Multi_operand5 and ALU_Multi_operand6 are changed, but not in ALU_operand1 and ALU_operand2, after the multiply.

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 Divider

When the division, {ALU_operand4, ALU_operand3, ALU_operand2, ALU_operand1}

is divided by {ALU__Multi_operand6, ALU__Multi_operand5}.

The **quotient** would be { ALU_out4, ALU_out3, ALU_out2, ALU_out1} and

When the LSB(Least Significant Bit) is "1", the **remainder** would be :

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

When the LSB is "0", the **remainder** would be { ALU_out6, ALU_out5}.

The operation is started when the ALU__Div_operand6 is written. The value in ALU_operand1 and ALU_operand2, ALU_operand3 and ALU_operand4are changed, but not in ALU_Div_operand5 and ALU_Div_operand6, after the division.

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

There are 16 IO in SCPU. Each of them is bit-wise-controlled, it could be either pull-high input, pull-low input, floating input, output high or output low. Secondary CCIR input, SCPU external IRQ and IIS interface are all shared with these IO pins.

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 could be either GPIO or the secondary CCIR interface. When it's the secondary CCIR interface, remember to set the IOA to input floating mode.

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

Table of shared pins

PIN NAME	SIGNAL NAME
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 could be GPIO, secondary CCIR interface, IIS or external IRQ. When it is CCIR, the related pins have to set to input floating mode. XSCPUIOB7 has to set to the input with pull-high resistor mode when External IRQ is used. When IIS is active, XSPUIOB6, XSPUIOB5 and XSPUIOB4 has to set to the output mode.

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

Table of shared pins

	OUTPUT	INPUT
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. REGISTER TABLE

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]				

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
0x4121						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 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