

SPECIFICATION



SPCA554A

Advanced Mobile Imaging Controller

Preliminary

APR. 08, 2004

Version 0.2

SUNPLUS TECHNOLOGY CO. reserves the right to change this documentation without prior notice. Information provided by SUNPLUS TECHNOLOGY CO. is believed to be accurate and reliable. However, SUNPLUS TECHNOLOGY CO. makes no warranty for any errors which may appear in this document. Contact SUNPLUS TECHNOLOGY CO. to obtain the latest version of device specifications before placing your order. No responsibility is assumed by SUNPLUS TECHNOLOGY CO. for any infringement of patent or other rights of third parties which may result from its use. In addition, SUNPLUS products are not authorized for use as critical components in life support devices/ systems or aviation devices/systems, where a malfunction or failure of the product may reasonably be expected to result in significant injury to the user, without the express written approval of Sunplus.

Table of Contents

	<u>PAGE</u>
1. OVERVIEW	5
1.1. INTRODUCTION.....	5
1.2. TERMINOLOGY.....	5
2. INTERFACE DESCRIPTION	6
2.1. HOST INTERFACE.....	6
2.1.1. Pin description.....	6
2.1.2. Parallel Interface.....	8
2.1.3. I2C Interface.....	10
2.1.4. SPI Interface.....	11
2.1.5. RS232 Interface.....	13
2.2. SENSOR INTERFACE.....	14
2.2.1. Pin Description.....	14
2.2.2 I2C master port.....	17
2.2.3 Three-wired master serial port.....	18
2.2.4 Data transfer timing.....	20
2.3. STORAGE INTERFACE.....	21
2.4. USB INTERFACE.....	24
2.5. AUDIO INTERFACE.....	25
2.6. DISPLAY INTERFACE.....	28
2.6.1. 18-bit RGB interface.....	31
2.6.2. 18-bit Memory interface.....	32
2.6.3. 9-bit Memory interface.....	32
2.6.4. 16-bit Memory interface.....	33
2.6.5. 8-bit Memory interface.....	35
2.6.6. YUV data output.....	36
3. RESET AND IO-TRAP	37
4. BOOT SEQUENCE	38
4.1. BOOT FROM INTERNAL BOOT ROM.....	38
4.2. BOOT FROM THE HOST PROCESSOR.....	39
5. POWER MANAGEMENT	40
5.1. INTERNAL CLOCK CONTROL.....	40
5.2. DEEP POWER DOWN MODE.....	41
6. FUNCTION DESCRIPTION	42
6.1. OPERATION OVERVIEW.....	42
6.1.1. Camera preview.....	43
6.1.2. Still image capture.....	44
6.1.3. Still image playback.....	46
6.1.4. Video recoding.....	46
6.1.5. Video playback.....	48
6.1.6. Screen saver.....	48
6.1.7. Video conference.....	49
6.2. HOST BRIDGE.....	50

6.2.1. Host bridge control registers	51
6.2.2. Main Page Register and Mirror Page Register	57
6.2.3. Vendor command communication flow	58
6.2.4. Internal buffer access	59
6.2.5. Internal register access	60
6.3. EMBEDDED CPU	61
6.3.1. Addressing space partition	62
6.3.2. Priority interrupt controller	64
6.4. SENSOR CONTROL	65
6.4.1. Synchronization/ Valid signals generation	65
6.4.2. Image cropping	67
6.4.3. Exposure Time and Flash Light Control	68
6.5. COLOR DSP	69
6.6. JPEG AND VIDEO CODEC	71
6.6.1. Pre-Processing	71
6.6.2. JPEG engine	73
6.6.3. Video engine	73
6.7. MEMORY CONTROLLER	74
6.7.1. Camera Image of YUV422 format	76
6.7.2. Camera image of YUV420 format	76
6.7.3. Camera image of Raw data format	77
6.8. CAMERA IMAGE CONTROLLER	77
6.8.1. Scaling	78
6.8.2. Rotation and mirror	78
6.8.3. BitBit of camera image data	78
6.8.4. Bad Pixel Concealment	79
6.8.5. YUV420-to-YUV422 conversion	79
6.9. STORAGE MEDIA CONTROLLER	79
6.10. USB CONTROLLER	80
6.11. AUDIO CONTROLLER	81
6.12. DMA CONTROLLER	82
6.12.1. General purpose DMA channel	84
6.12.2. Secondary DMA channel	84
6.12.3. DRAM-to-DRAM DMA channel	85
6.12.4. FAT acceleration	85
6.13. 2D GRAPHICS ENGINE	85
6.13.1. Resolution and color system	86
6.13.2. BitBLT	88
6.13.3. Sprite	93
6.13.4. Line drawing	95
6.13.5. Gradient Fill	96
6.14. DISPLAY CONTROLLER	98
6.14.1. Geometrical Transformation	98
6.14.2. Color Adjust of Camera Image	100
6.14.3. Mixer	101

6.14.4. Image Data Format Conversion	104
6.14.5. LCM control options.....	106
6.15.PERIPHERALS	107
7. DISCLAIMER.....	109
8. REVISION HISTROY	110

Sunplus Confidential
For Edom Technology Use Only

ADVANCED MOBILE IMAGING CONTROLLER

1. OVERVIEW

1.1. Introduction

The SPCA554A is a multimedia processor targeting high-end mobile applications. The SPCA554A serves as a co-processor of the host (base-band) processor in a mobile phone. It handles all the multimedia functions required in a high-end mobile phone, such as digital camera functions, video camera functions, MP3 playback functions, JAVA functions, as well as rich OSD (on screen display) functions. Internal engines of the SPCA554A include a digital camera processor, MPEG4/H.263 CODEC, JPEG CODEC, 2D graphics engine and a powerful 32-bit RISC processor. The powerful all-in-one functions of the SPCA554A make the implementation of multimedia mobile applications a straightforward task.

Chapter 2 describes all the interfaces of the SPCA554A in terms of their corresponding functionalities, options and interconnections to other modules in a mobile phone system. Chapter 3 describes reset timing requirements and the power-on IO-TRAP mechanism. In Chapter 4 the internal clock distribution and power management of the SPCA554A will be discussed. Next, chapter 5 will explain SPCA554A boot flow and options. Finally, chapter 6 has detailed information about the function of individual internal modules.

1.2. Terminology

ADC:	Analog to digital converter
AE:	Auto exposure control
AWB:	Auto white balance control
BitBLT :	Bit Block Transfer
CODEC:	Encoder and decoder
DAC:	Digital to analog converter
I80:	Intel 80xx CPU protocols
I2C:	Inter IC Bus. I ² C bus is usually used to control the CMOS sensor. The SPCA554A can also be controlled through a separate I2C bus
I2S:	Inter IC Sound. The I ² S serial bus is used in SPCA554A to transfer to/from the external audio CODEC
IO-TRAP:	In the reset period, SPCA554A latches IO-pin values for configuration settings
LCM:	Liquid crystal display module
M68:	Motorola 68000 CPU data protocols
ROPs:	Raster operations
SPI:	serial peripheral interface
VLC:	Variable length coding. VLC is used in JPEG, MPEG and H.263 CODEC
VSYNC:	Vertical synchronization signal
VVALID:	Vertical valid signal of image
VOP:	Video object plane
GOP:	Group of object plane

2. INTERFACE DESCRIPTION

The SPCA554A has six interfaces to be connected to external modules. They include the following:

- Host interface: To communicate with the host processor
- Sensor interface: Connects the SPCA554A to a CMOS sensor, CCD sensor module or TV decoder to get image input
- Display interface: Connect to an LCM
- Audio interface: Allows the SPCA554A to get audio input.
- Storage media interface: Allows the SPCA554A to store image data, audio data and video data to external storage cards.
- USB interface: communicates with a PC or other USB devices.

The following table lists all IO buffer types used in the SPCA554A:

IO-buffer	Description
ID	Input buffer with programmable internal pull-down
IS	Schmitt trigger input buffer
B	Bi-directional buffer
BD	Bi-directional buffer with programmable internal pull-down
BDS	Bi-directional buffer with limited slew rate and programmable internal pull-down
O	Output buffer
XI	Crystal input buffer
XO	Crystal output buffer
P	Power
G	Ground

Table 2-1: IO-buffer types used in the SPCA554A

2.1. Host Interface

The host interface can be serial or parallel, depending on the requirement of applications. It is used by host processor controls through the host interface. The host processor can access internal registers of the SPCA554A, send host commands and data to the SPCA554A, or get status and data from the SPCA554A. Instead of programming the internal registers, the operational flow of the SPCA554A is controlled by a set of host commands. The host commands are a complete set of APIs for multimedia applications executed by the on-chip 32-bit RISC CPU. The RISC CPU interprets the host commands and performs corresponding operations and then sends the status back to the host processor. The SPCA554A supports the following types of host interfaces:

- I2C
- SPI interface mode 0, 1, 2 and 3
- RS232
- 8/16-bit parallel interface M68 mode
- 8/16-bit parallel interface i80 mode

The configuration of the host interface is selected by IO-TRAP settings. IO-TRAP mapping can be found in the SPCA554A datasheet. The following table lists the signal names in the various modes of the host interface. A detailed description of signal functions and timing is given in sections 2.1.1 through 2.1.5 and is as follows:

2.1.1. Pin description

The following table lists the host interface pin usage in different host types:

Pin Name	Pin Allocation In Different Host Interface Types												IO		
	I2C		SPI		RS232		8/16-bit Parallel bus		8-bit bus and RS232		8-bit bus and I2C			8-bit bus and SPI	
HGPIO0	SCL	I	Cs_n	I	Rxd	I	Cs_n	I	Cs_n	I	Cs_n	I	Cs_n	I	BDS

Pin Name	Pin Allocation In Different Host Interface Types														IO
	SDA	B	SCK	I	Txd	O	A0	I	A0	I	A0	I	A0	I	
HGPIO1							A0	I	A0	I	A0	I	A0	I	BDS
HGPIO2			SI	I			Rd_n / En	I	Rd_n / En	I	Rd_n / En	I	Rd_n / En	I	BDS
HGPIO3			SO	O			Wr_n / RW	I	Wr_n / RW	I	Wr_n / RW	I	Wr_n / RW	I	BDS
HGPIO4							D0	B	D0	B	D0	B	D0	B	BDS
HGPIO5							D1	B	D1	B	D1	B	D1	B	BDS
HGPIO6							D2	B	D2	B	D2	B	D2	B	BDS
HGPIO7							D3	B	D3	B	D3	B	D3	B	BDS
HGPIO8							D4	B	D4	B	D4	B	D4	B	BDS
HGPIO9							D5	B	D5	B	D5	B	D5	B	BDS
HGPIO10							D6	B	D6	B	D6	B	D6	B	BDS
HGPIO11							D7	B	D7	B	D7	B	D7	B	BDS
HGPIO12							D8	B	Rxd	I	SCL	I	Cs_n	I	BDS
HGPIO13							D9	B	Txd	O	SDA	B	SCK	I	BDS
HGPIO14							D10	B					SI	I	BDS
HGPIO15							D11	B					SO	O	BDS
HGPIO16							D12	B							BDS
HGPIO17							D13	B							BDS
HGPIO18							D14	B							BDS
HGPIO19							D15	B							BDS
BYPASS															BDS

Table 2-2: Pin list of host interface

Note: BYPASS pin is input only.

HGPIO[19:0] are shared between different types of host interfaces. The SPCA554A always acts as a slave device of the selected bus type. For example, if the I2C bus type is selected, the SPCA554A is a slave device on the I2C bus. The host interfaces allows the SPCA554A to be controlled by an 8-bit parallel bus and a serial bus at the same time.

Note that there are certain pins not used in a specific types of host interfaces. These pins are automatically set to the GPIO function once the host type is selected by the IO-TRAP configuration. For example, HGPIO[2:19] are not used in I2C type host interfaces.

They are automatically set to GPIO pins after IO-TRAP terminates.

Pin 21 of the host interface is the bypass mode control signal. It is an active high signal. In the bypass mode, the LCM is accessed by the host processor. Otherwise, the LCM is accessed by the SPCA554A. The following diagram shows a typical interface connection between the SPCA554A, LCM and the host processor. Note that the SPCA554A is put into the suspend state in the bypass mode to reduce power consumption. Also note that internal modules do not respond to host commands in the bypass mode.

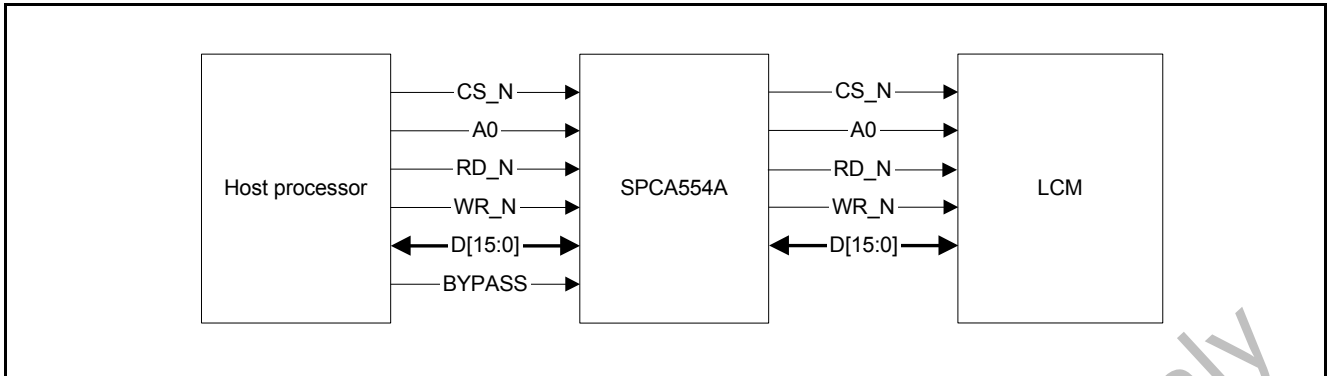


Figure 2-1: Interface connection example between the SPCA554A, LCM and the host processor

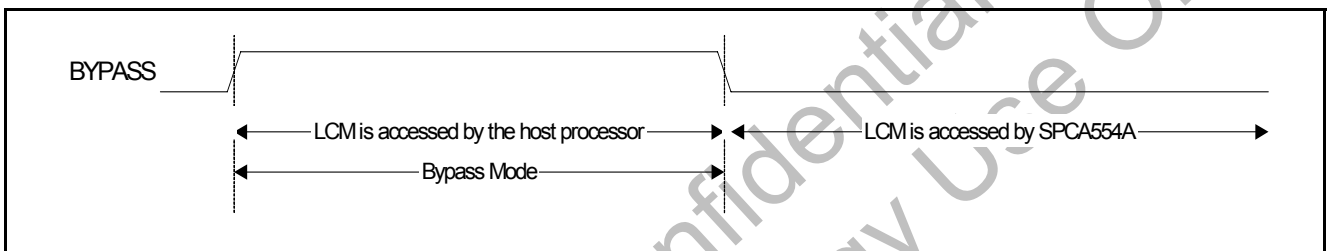


Figure 2-2: Bypass timing diagram

2.1.2. Parallel Interface

Sections 2.1.2 to section 2.1.5 describe the host interface in terms of signal functions and hardware protocols.

i80 Mode:

The signals of the i80 mode parallel interface are described below:

- CS_n : Chip select to the SPCA554A. Can be active high or active low, depending on the configuration of IO-trap (Refer to Section 4.3).
- A0 : Address/data indicator. When low (logic 0), it indicates that the host is sending an address via the data bus. When High (logic 1), it means that the host is sending data.
- Wr_n : write pulse.
- Rd_n : Read pulse.
- D[15:0]: While the host is sending addresses to the SPCA554A, only D[6:0] are used as address bits. While transferring data, either from the host to the SPCA554A or from the SPCA554A to the host, both 8-bit data and 16-bit data are supported. If the 8-bit data bus is selected, data should be placed in low bytes; D[7:0].

Write cycle timing

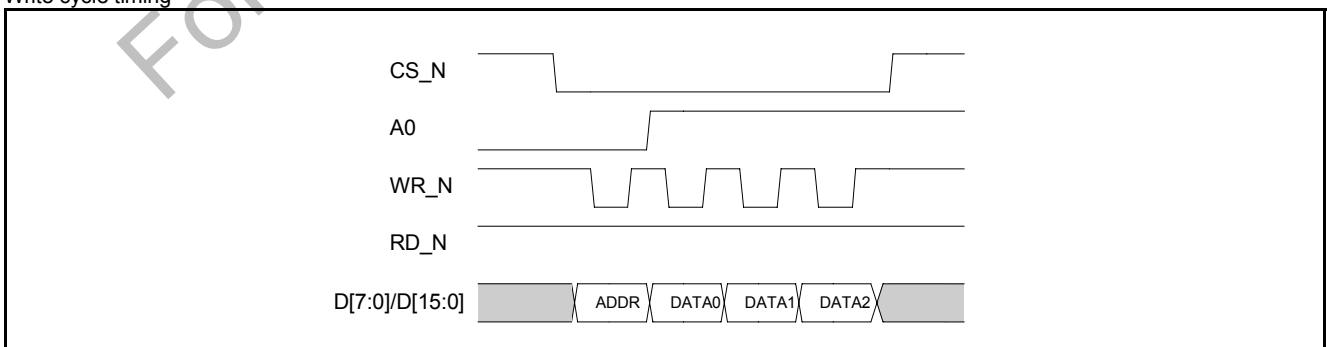


Figure 2-3: Write cycle timing of the i80 mode

Read cycle timing

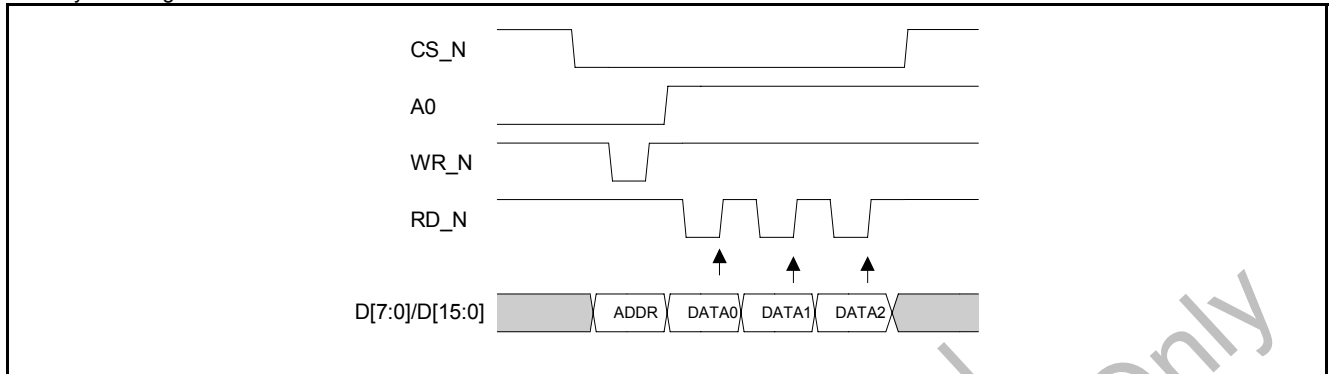


Figure 2-4: Read cycle timing of i80 mode

M68 Mode:

The signals of the m68 mode parallel interface are described below:

CS_n: Chip select to the SPCA554A. Its function is the same as in the i80 mode.

A0: Address /data indicator. The function is the same as in the i80mode.

RW: A read/write command indicator. Logic 0 indicates a write cycle, and logic 1 indicates a read cycle.

E: Enable signal. This signal is used in conjunction with the RW signal to generate the SPCA554 internal read/write pulse.

D[15:0]: Same as in the i80 mode.

Write cycle timing:

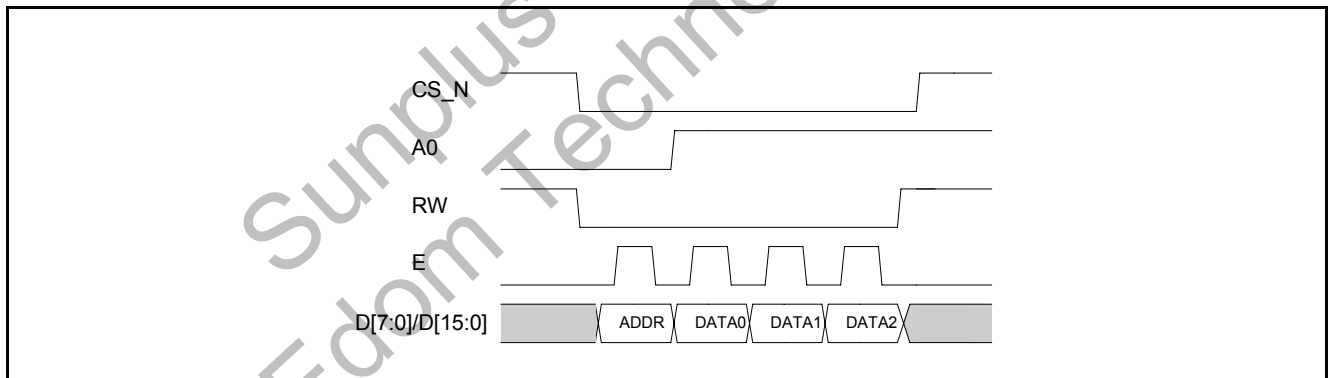


Figure 2-5: Write cycle timing of the M68 mode

Read cycle timing

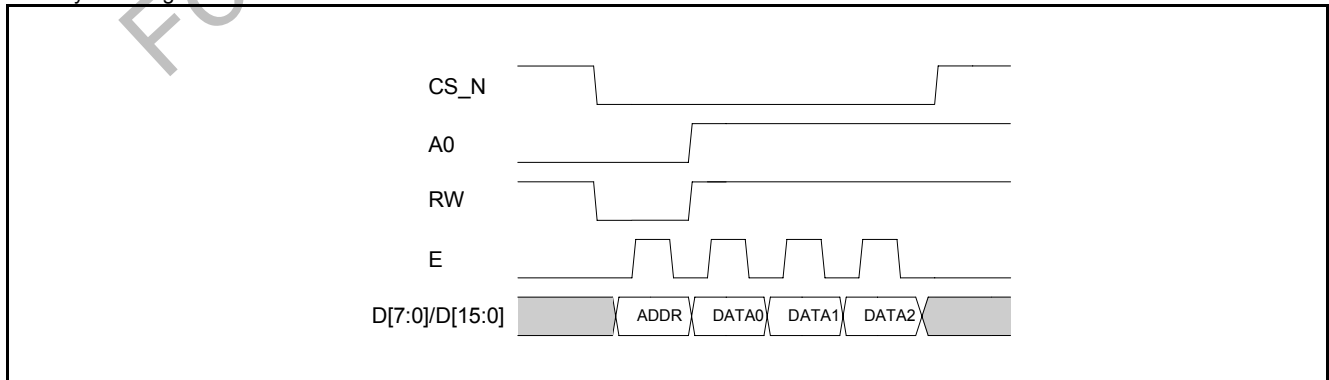


Figure 2-6: Read cycle timing of the M68 mode

2.1.3. I2C Interface

The host processor can also communicate with the SPCA554A through a I2C bus. In this case, the SPCA554A is a I2C slave from the host processor's point of view. As a I2C slave, the SPCA554A supports two alternative I2C slave addresses, depending on the IO-trap setting. The address can be 0X56 or 0X65. The address can be changed in case the mobile system has another I2C slave with a conflicting slave address. The following diagram shows the access timing. Note that the

register address field contains only 7 valid bits. Bit-7 of the address is ignored. The data field may contain multiple bytes when necessary. When multiple bytes follow the address byte, the data is written into (or read from) the same address. This is useful in some SPCA554A registers which are mapped to data ports of the internal memory. The multiple-byte data access enables burst access to the internal memory.

Write timing

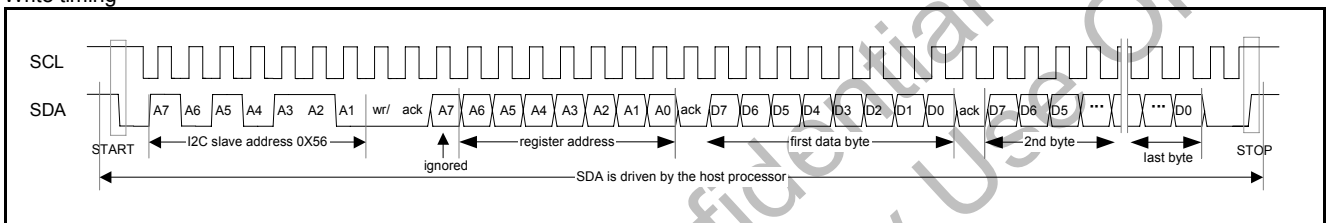


Figure 2-7: I2C write timing

Read timing: To read a register value from the SPCA554A using a I2C bus, two I2C cycles are required. The first cycle sets the register address. The second cycle reads data from the register (or port) of the indicated address.

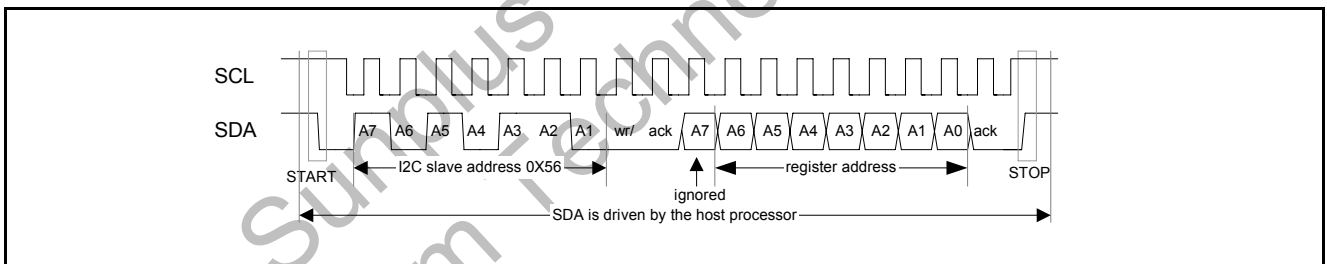


Figure 2-8: First I2C cycle of I2C read timing

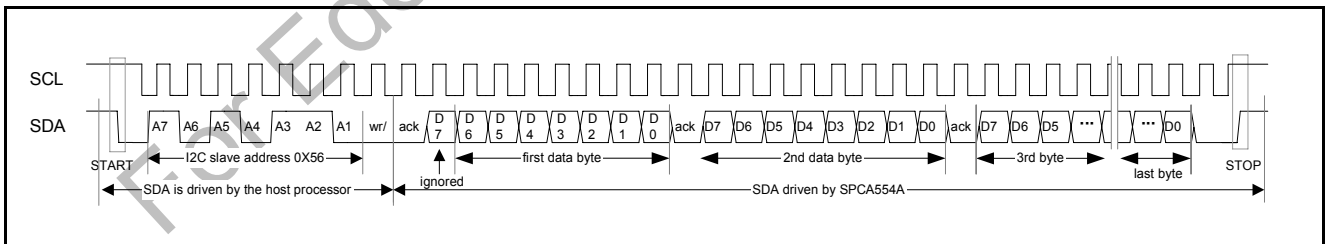


Figure 2-9: Second I2C cycle of I2C read timing

As long as the clocks are supplied to the SPCA554A, the data is written to the same address repeatedly. The SPCA554A automatically detects the stop condition of the I2C protocol and stops the burst read (or write) operation. Figure 2-7 ~Figure 2-9

shows when the SDA pin is driven by the host processor and when it is driven by the SPCA554A. Note that the "ACK" handshake is always driven by the I2C slave, i.e. by the SPCA554A.

2.1.4. SPI Interface

The following diagram shows the timing of SPCA554A internal register access via the SPI serial interface. The SPCA554A supports all SPI protocols, including mode-0, mode-1, mode-2 and mode-3.

A command transfer defines the access address and the access mode. The access mode can be read access or write access, depending on bit-7 of the transfer. Multiple data bytes can be transferred, following a command. For example, transferring a JFIF file to the SPCA554A requires all the data bytes to be written to the same image data port.

SPI mode 0 (clock stops at logic 0 when idle, rising edge latch data):

Write cycle timing

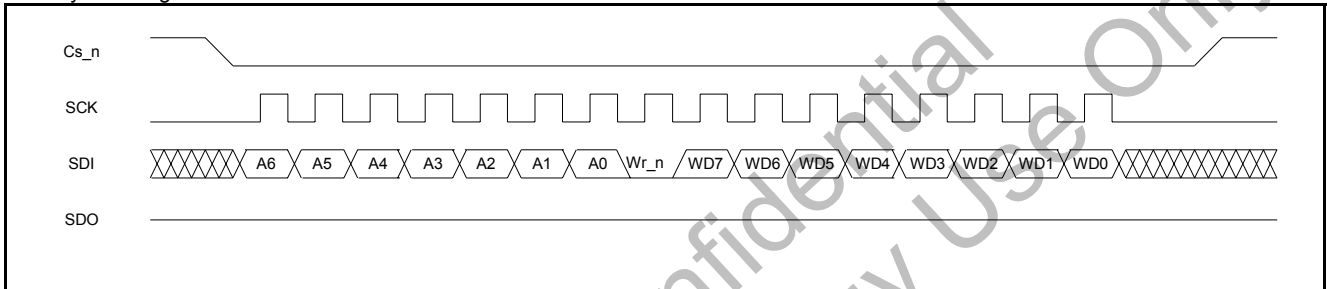


Figure 2-10: SPI mode-0 write timing

Read cycle timing

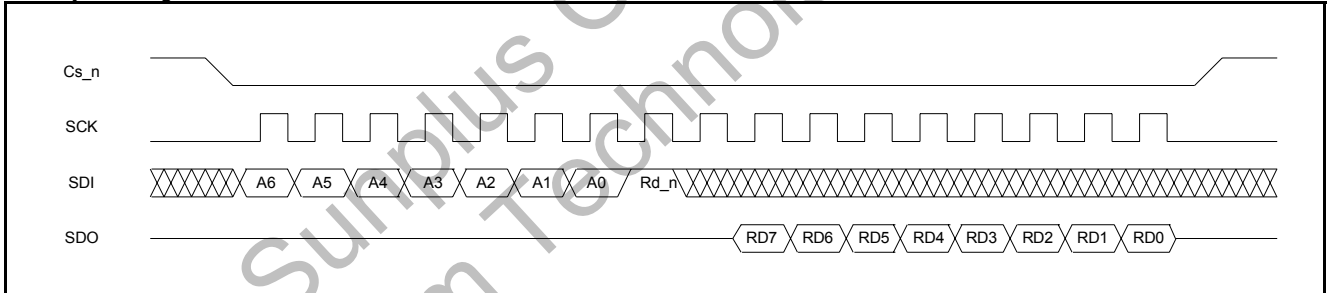


Figure 2-11: SPI mode-0 read timing

SPI mode 1 (clock stops at logic 0 when idle, falling edge latch data):

Write cycle timing

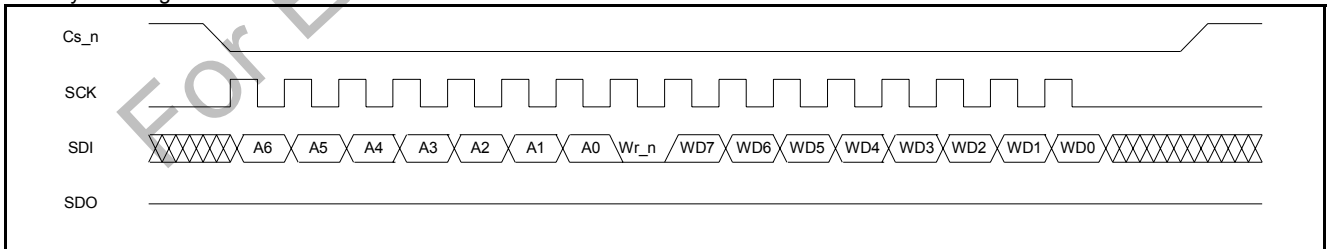


Figure 2-12: SPI mode-1 write timing

Read cycle timing

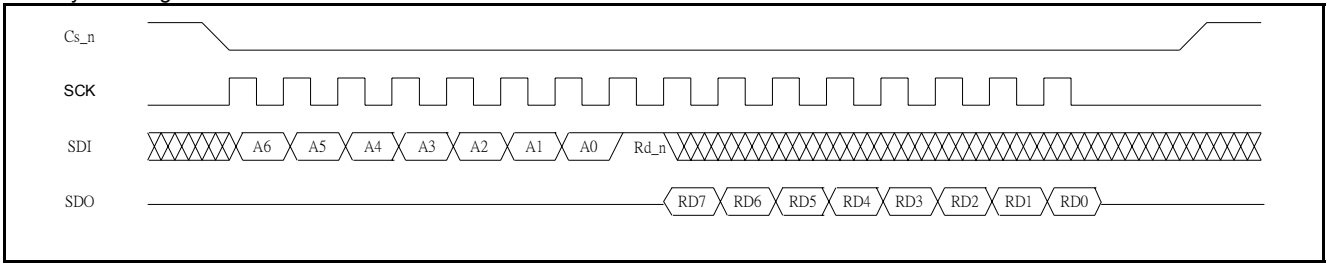


Figure 2-13: SPI mode-1 read timing

SPI mode 2 (clock stops at high when idle, falling edge latch data):

Write cycle timing

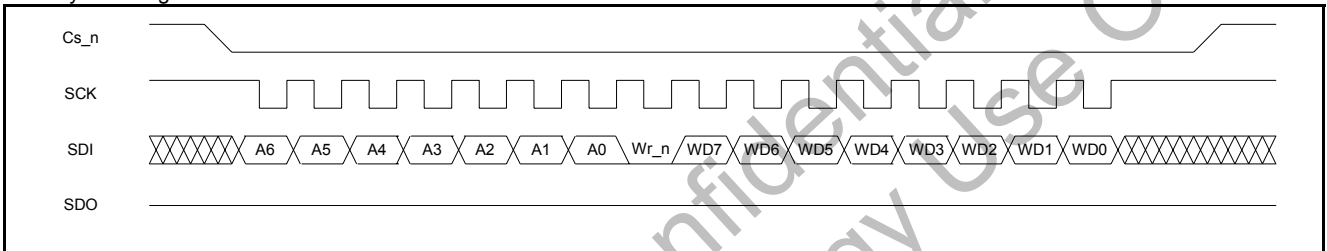


Fig 2-14: SPI mode-2 writes timing

Read cycle timing

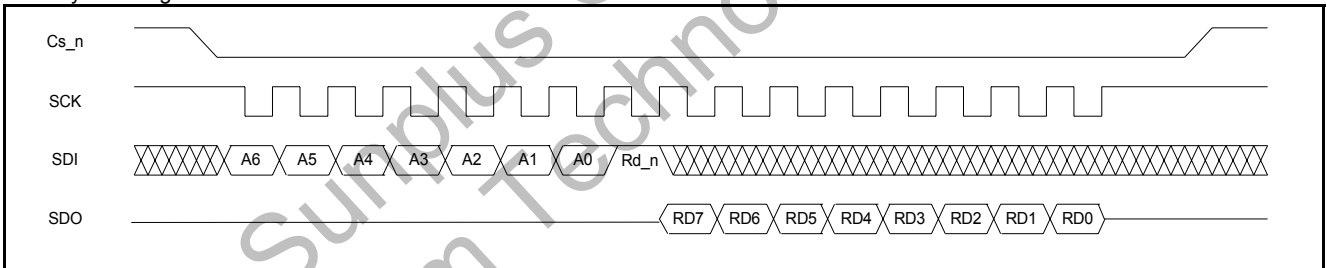


Figure 2-15: SPI mode-2 read timing

SPI mode 3 (clock stops at logic 1 when idle, rising edge latch data):

Write cycle timing

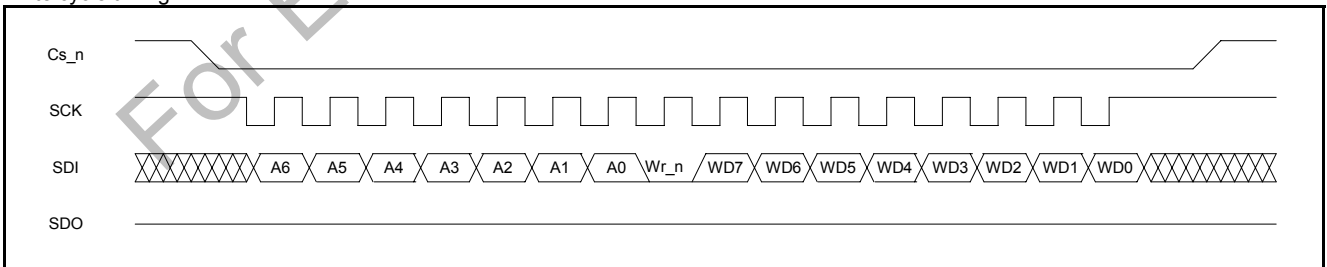


Figure 2-16: SPI mode-3 write timing

Read cycle timing

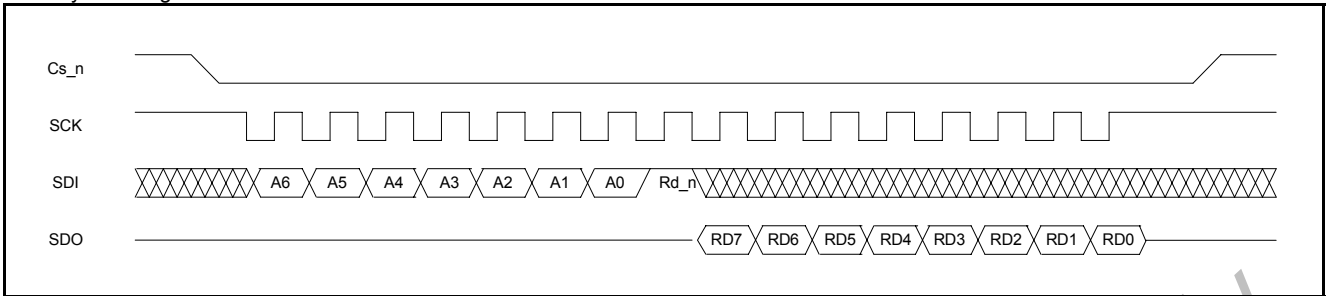


Figure 2-17: SPI mode-3 read timing

2.1.5. RS232 Interface

The following diagram shows the timing of SPCA554A internal register access via the RS232 serial interface. Nine-bit data transfer protocol is adopted by the SPCA554A. Bit-8 defines whether the transfer is a command transfer or a data transfer. A command transfer defines the access address and the access mode. The access mode can be read access or write access, depending on bit-7 of the transfer. Multiple data bytes can be transferred, following a command. The baud rates are programmable from 1.2 Kb/s to 115.2 Kb/s. The default baud

rate is 9600 bit/sec after power on. The time interval between each read data transfer is also programmable from 0 to 255 bits, which can also be programmed by the SPCA554A internal register.

The following diagrams show the timing. The read transfer command is sent from the host to the SPCA554A. After SPCA554A receives the read command, it transfers data back to the host.

Write cycle timing

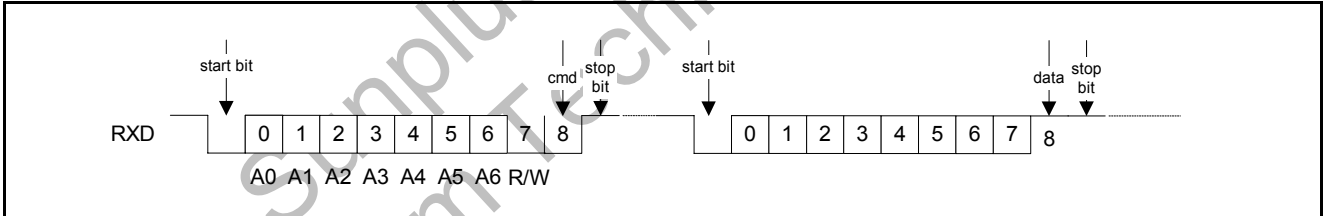


Figure 2-18: RS232 write timing

Read cycle timing

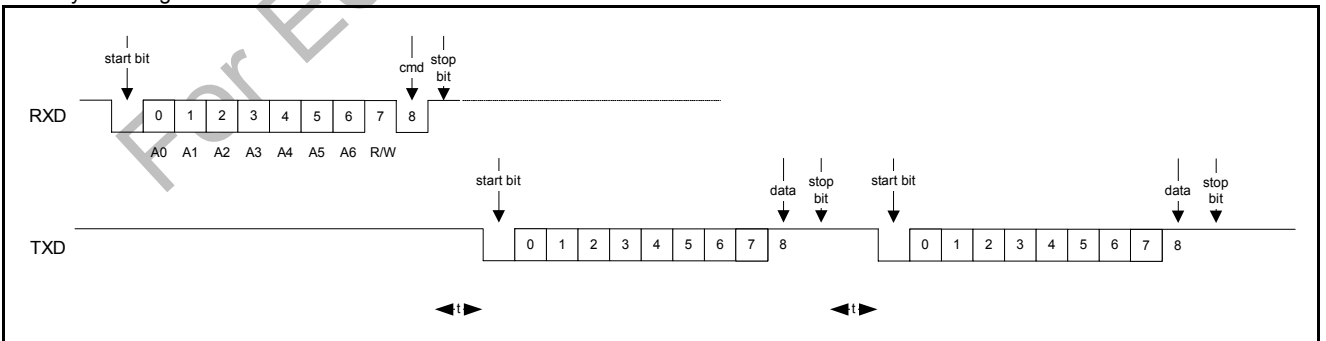


Figure 2-19: RS232 read timing

Note: The symbol “t” represents the time interval between each read data transfer.

2.2. Sensor interface

The sensor interface is used by the SPCA554A to acquire image input. Due to the wide range of data formats supported, the SPCA554A is able to obtain image data from a CMOS sensor, CCD sensor module or a TV decoder. The SPCA554A supports the following data types:

- Bayer pattern raw data: Usually used by CMOS sensors.
- CCIR601 YUV data: Usually used by CMOS sensors, CCD sensor modules and TV decoders.
- CCIR656 YUV data: Usually used by CMOS sensors, CCD sensor modules and TV decoders.

2.2.1. Pin Description

The following table lists sensor interface pin usage in different types of image data inputs:

Pin Name	Bayer Raw Data Input			CCIR601		CCIR656		IO Buffer	
TGGPIO0	FLASHCTRL						O	BD	
TGGPIO1	TGGPIO1			DVALID	I	TGGPIO1		B	BD
TGGPIO2	SCL (I2C)						B	BD	
TGGPIO3	SDA (I2C)						B	BD	
TGGPIO4	HD (horizontal synchronization)	B	HD	I	TGGPIO4		B	BD	
TGGPIO5	VD (vertical synchronization)	B	VD	I	TGGPIO5		B	BD	
TGGPIO6	Pixel clock						B	BD	
TGGPIO7	Master clock						B	BD	
TGGPIO8	SEN						O	BD	
TGGPIO9	SCK						O	BD	
TGGPIO10	SD						O	BD	
RGB0	Bayer raw data input, bit 0.	I	VVALID	I				ID	
RGB1	Bayer raw data input, bit 1.	I	HVALID	I				ID	
RGB2	Bayer raw data input, bit 2.	I	YUV data bit 0			I		ID	
RGB3	Bayer raw data input, bit 3.	I	YUV data bit 1			I		ID	
RGB4	Bayer raw data input, bit 4.	I	YUV data bit 2			I		ID	
RGB5	Bayer raw data input, bit 5.	I	YUV data bit 3			I		ID	
RGB6	Bayer raw data input, bit 6.	I	YUV data bit 4			I		ID	
RGB7	Bayer raw data input, bit 7.	I	YUV data bit 5			I		ID	
RGB8	Bayer raw data input, bit 8.	I	YUV data bit 6			I		ID	
RGB9	Bayer raw data input, bit 9.	I	YUV data bit 7			I		ID	

Table 2-3: Pins description for sensor interface

The typical function of TGGPIO0 is flash light strobe control. It can be used as a GPIO pin if the target system does not implement the flash light function. TGGPIO2 and TGGPIO3 constitute a I2C master. The control is used to program the registers in the CMOS sensor, the registers in the CCD module, or the registers in a TV decoder. TGGPIO8, TGGPIO9, and TGGPIO10 can operate in the three-wire mode. The three-wire control bus is normally used in a CCD module.

TGGPIO4 is a horizontal synchronization signal, which indicates the start of an image line. TGGPIO5 is the vertical synchronization signal, which indicates the start of a new image frame. Both GPIO4 and GPIO5 can be controlled by the SPCA554A or by the sensor, meaning that the SPCA554A CMOS sensor interface can be operated in the master mode or the slave mode.

TGGPIO6 is the pixel clock, which is usually driven by the sensor. The SPCA554A uses the pixel clock to sample input image data. TGGPIO7 is the master clock supplied to the sensor by the SPCA554A. The master clock serves as the clock input to the CMOS sensor. The frequency of the master clock is normally multiples of the pixel-clock frequency. The SPCA554A sets the master-clock frequency according to sensor specifications by default. However, in extremely low-light condition, the SPCA554A might reduce the frequency of the master clock to get

longer exposure time.

The SPCA554 supports 10-bit raw data input. The data bus is also used to input YUV data in CCIR601 and CCIR656 standards. The CCIR601 601 standard defines both the 8-bit and 16-bit data bus, however the SPCA554A only supports the 8-bit mode. The following figures are interconnection examples between the SPCA554A and Omnivision OV9640 CMOS sensors. OV9640 can output Bayer pattern raw data, CCIR656 YUV data, and CCIR601 YUV data.

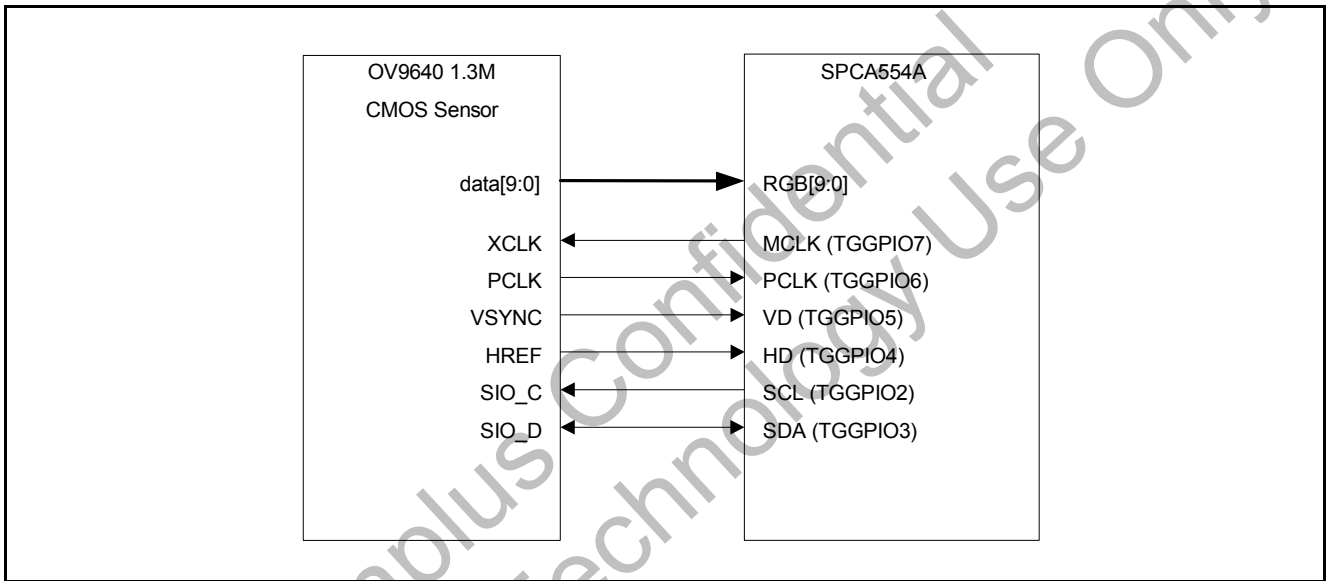


Figure 2-20: Interface connection between the SPCA554A and the OV9640 CMOS sensor; raw data

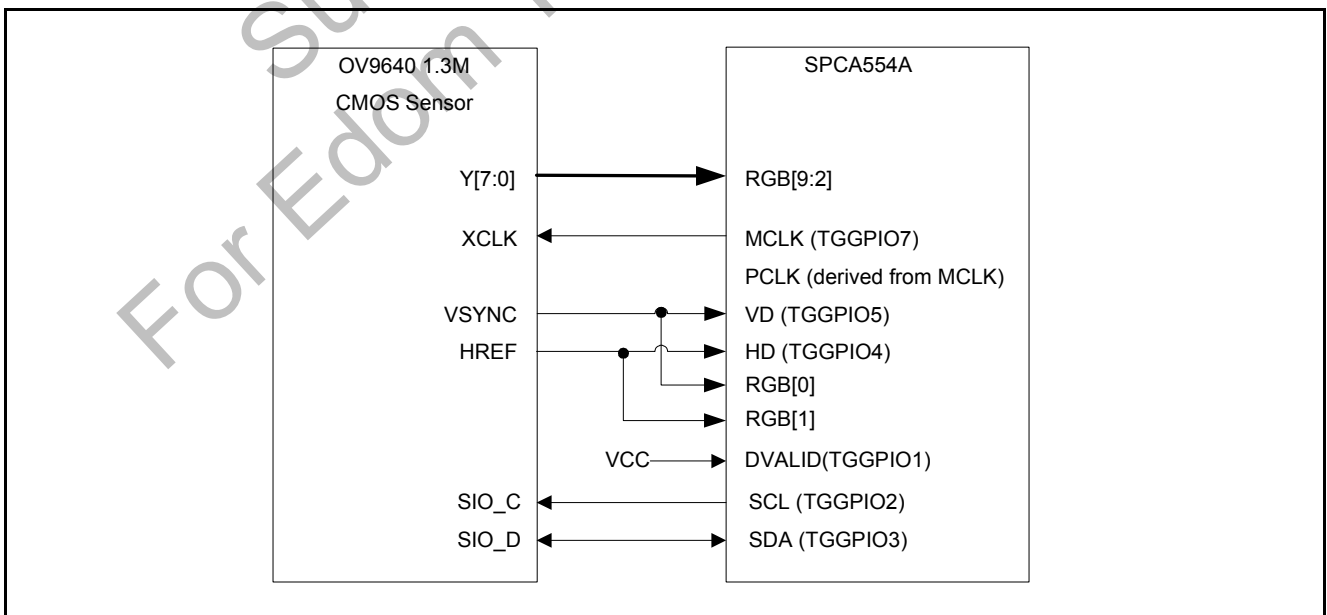


Figure 2-21: Interface connection between the SPCA554A and the OV9640 CMOS sensor; CCIR601 YUV data

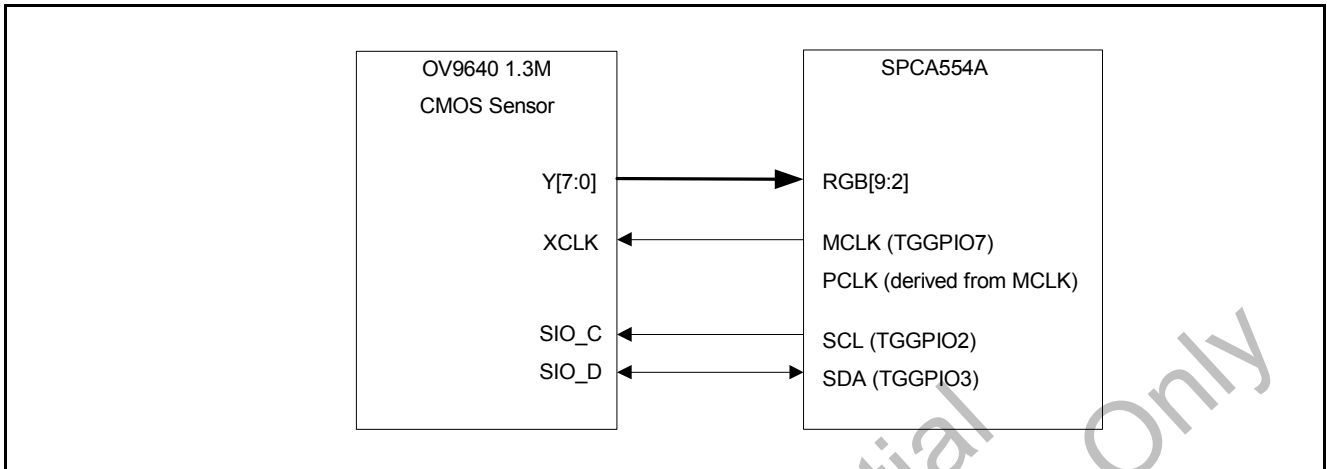


Figure 2-22: Interface connection between SPCA554A and OV9640 CMOS sensor; CCIR656 YUV data

Figure 2-23 and Figure 2-24 shows the interconnection between the SPCA554A and a TV decoder using a digital TV standard interface. A 27MHz external clock source is required in TV-input applications.

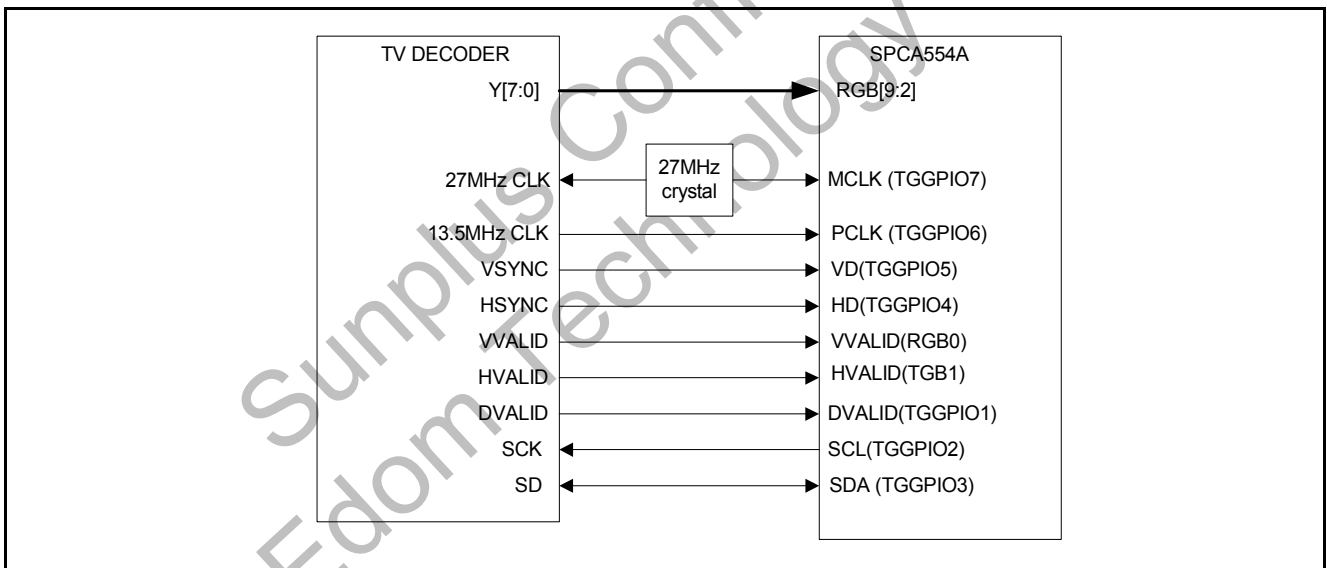


Figure 2-23: Interface connection between the SPCA554A and a CCIR601 TV Decoder

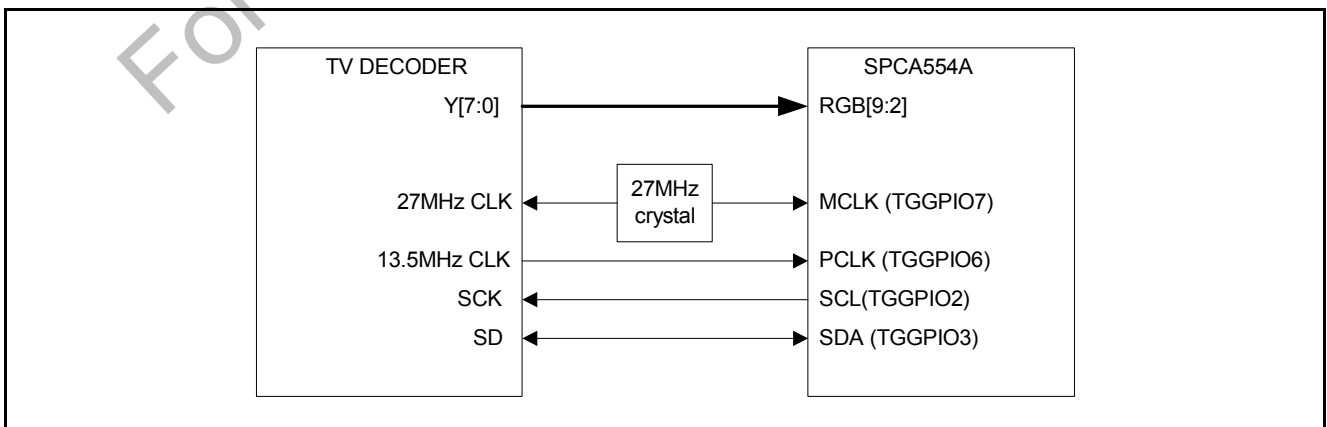


Figure 2-24: Interface connection between the SPCA554A and a CCIR656 TV Decoder

2.2.2 I2C master port

The SPCA554A has an internal I2C master controller; TGGPIO[3:2]. Eight bytes of internal buffer are integrated with the I2C master controller, so that up to 8-bytes of burst access to the CMOS sensor can be done by the I2C master controller. The fast mode data transfer of I2C specifications is also supported, which provides up to a 400K bits/sec data rate. According to I2C standards, there are a lot of variations in terms of timing. The SPCA554A supports the following timings (just which timing is appropriate depends on the requirement of the CMOS sensor):

- Normal-write timing
- Burst-write timing
- Normal-read timing
- Read-without-stop-code timing
- Read-without sub-address timing

The slave address in I2C specifications indicates which I2C device is to be accessed. The slave address of the CMOS sensor can

be found in CMOS sensor specifications. The SPCA554A supports a programmable I2C slave address. The sub-address indicates which internal register of the CMOS sensor is to be accessed. For example, the CMOS sensor gain register and the exposure register have different sub-addresses in the CMOS sensor.

Normal-write timing

The sub address is transmitted only once, where multiple bytes of data are written into the same CMOS sensor register.

Slave address: The I2C device address, i.e. the CMOS sensor I2C address.

W: 0 for write operation, 1 for read operation

ACKs: Acknowledge from CMOS sensor, 0 for acknowledge, 1 for non-acknowledge

ACKm: Acknowledge from the SPCA554A, 0 for acknowledge, 1 for non-acknowledge

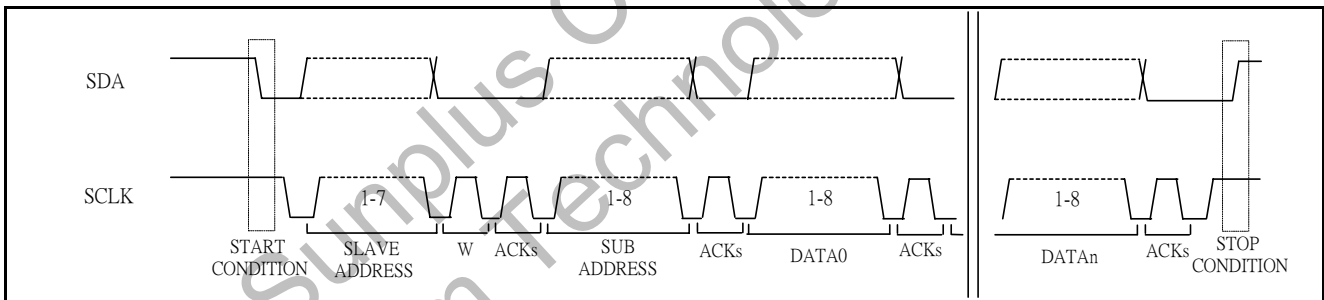


Figure 2-26: I2C normal write timing

Burst-write timing:

Note that each byte is written with a different sub-address.

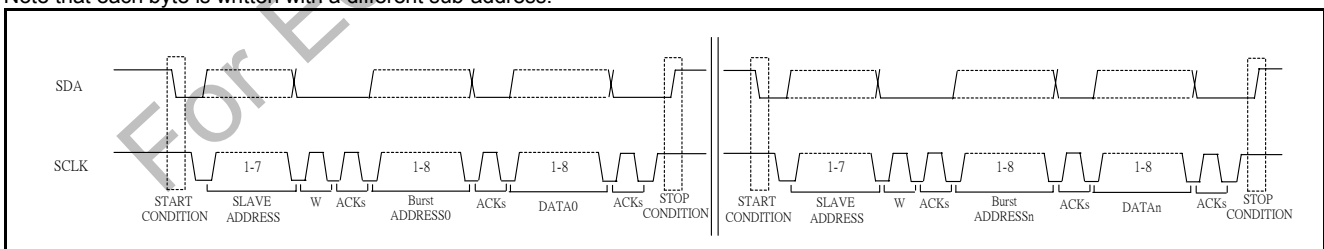


Figure 2-27: I2C burst-write timing

The normal data read protocol:

The SPCA554A sends a STOP code to the CMOS sensor for each byte of data transfer.

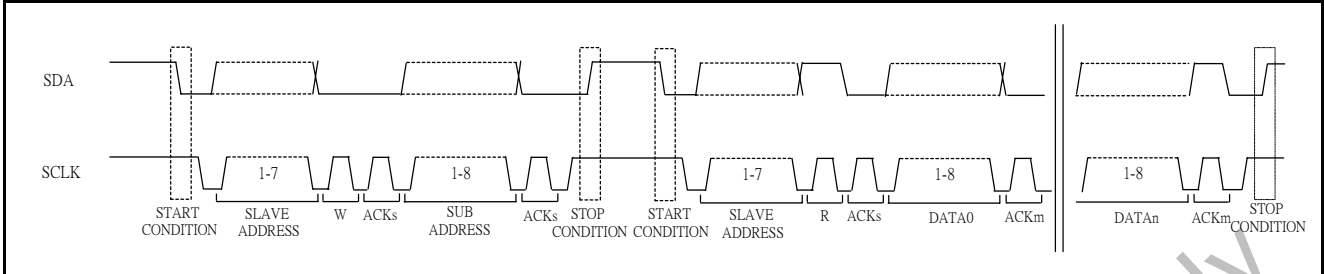


Figure 2-28: I2C normal read timing

Data-Read without Stop Code Protocol

Note: No STOP code is sent between bytes.

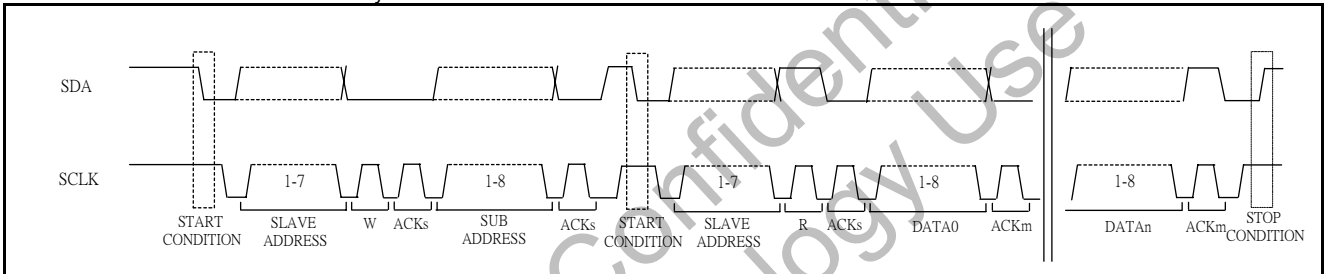


Figure 2-29: I2C data read without stop code timing

Data-Read without Sub-Address Protocol:

Some CMOS sensors allow the omission of sub-address transmission if the sub-address is the same as the previous I2C transfer.

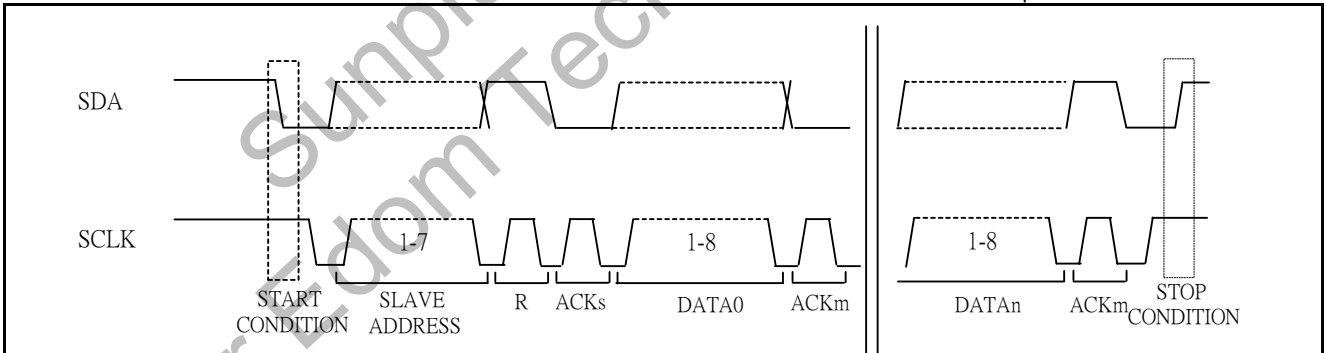


Figure 2-30: I2C data read without sub-address timing

2.2.3 Three-wired master serial port

CCD modules use a proprietary control bus for internal register programming. The SPCA554A supports a three-wire control bus with four different timing options. Using the three-wire control bus, the SPCA554A can transmit up to 64 bits of data to the CCD module, however, there is no read-back path. The bit clock

frequency is programmable. The SPCA554A transmits low byte first and LSB (Least Significant Bit) first (Refer to the diagram below). Bit 0 of DATA0 is the first bit to be sent to the sensors. Figure 2-31 shows the data transfer sequence.

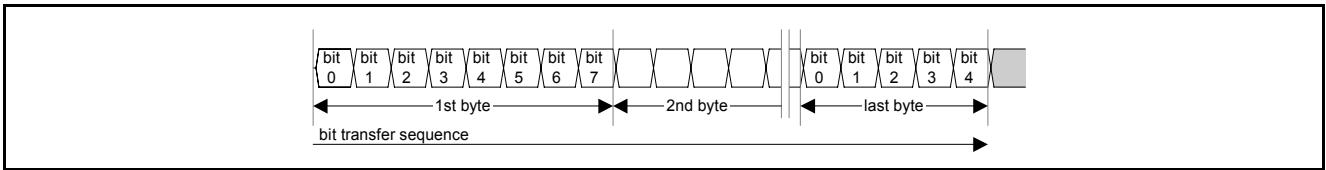


Figure 2-31: Three-wired control bus data transfer sequence

The SPCA554A supports 4 types of three-wire protocols as described below:

Type-0 protocol : (use only two signals)

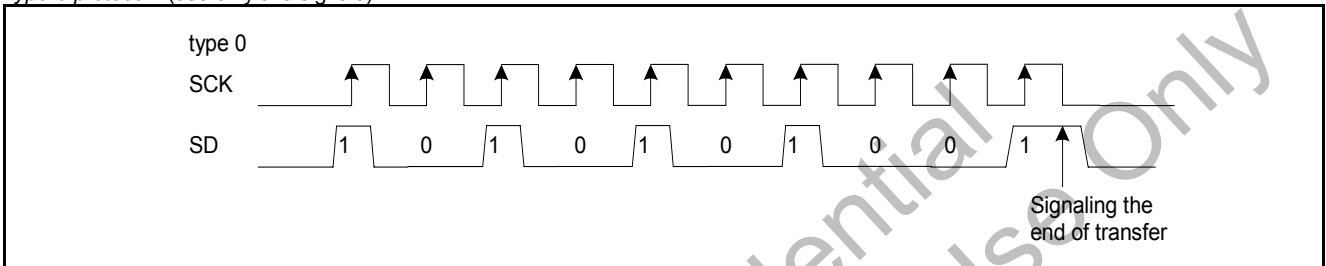


Figure 2-32: Three-wired master serial interface type-0 protocol

In type-0 protocol, the SEN signal is not used. The serial data is latched at the rising edge of SCK by the CMOS sensor. The SD signal stays high at the last falling edge of SCK. The SPCA554A drives DS high at the falling edge of the last falling edge of SCK to signal the end of serial transfer. The example below illustrates a 10-bit data transfer. Note that the data is 10_01010101b (LSB-first).

Type-1 protocol

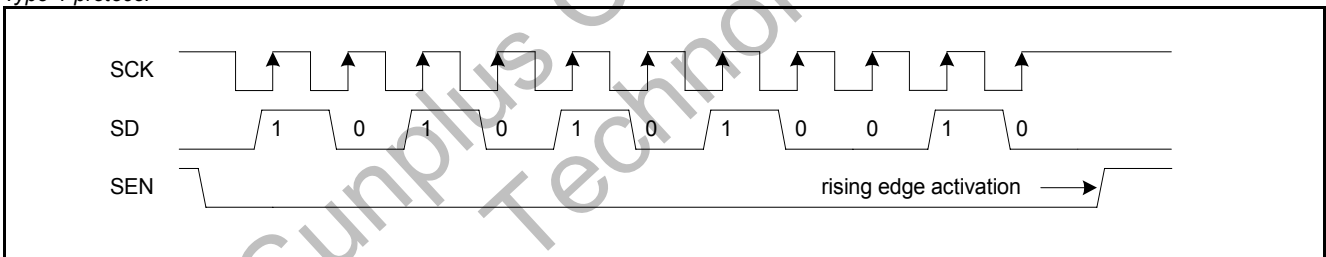


Figure 2-33: Three-wired master serial interface type-1 protocol

In Type-1 protocol, the data is latched at the rising edge of SCK. SEN is driven low at the start of entire data transfer cycle and a rising edge of SEN indicates the end of transfer. Figure 2-33 shows an example of an 11-bit data transfer, with data 010_01010101b, using type-1 protocol.

Type-2 protocol

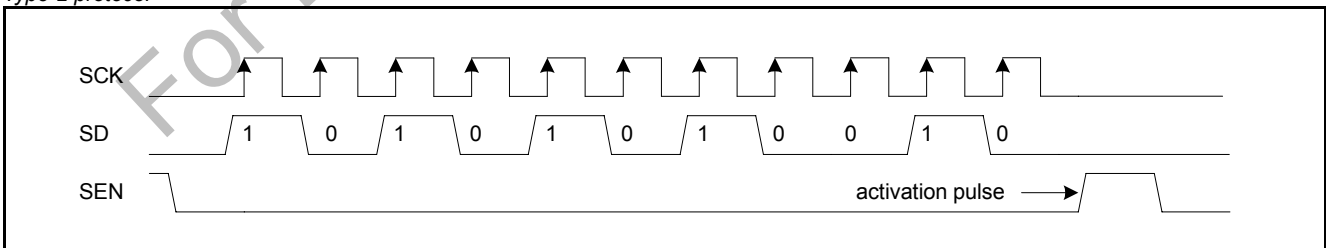


Figure 2-34: Three-wired master serial interface type-2 protocol

Type-2 protocol is similar to type-1 protocol, except for SEN signal behavior. A positive pulse on the SEN signal is used to indicate the end of data transfer in type-2 protocol. Figure 2-34 shows an 11-bit data transfer using type-2 protocol. The data is 010_01010101b.

Type-3 protocol

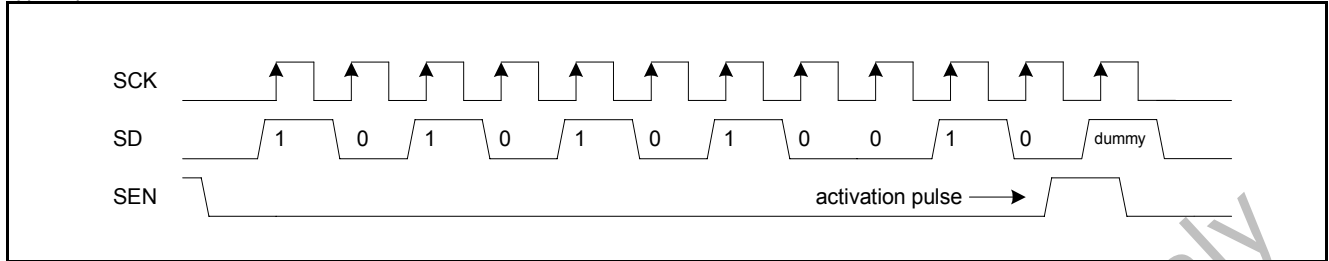


Figure 2-35: Three-wire master serial interface type 3 timing

Type-3 protocol is similar to type-2 protocol except that one dummy bit is attached at the end of the data transfer. Figure 2-35 shows an 11-bit data transfer using type-3 protocol. The data is 010_01010101b.

2.2.4 Data transfer timing

This sub-section describes the timings in raw data transfer, CCIR601 data transfer and CCIR656 data transfer.

■ **Raw Data Transfer Timing**

The SPCA554A requires HSYNC and VSYNC signals to construct the entire timing of an image frame. These signals can be generated by the SPCA554A internal counter or fed back from the sensors. Figure 2-36 illustrates the color filter array in a typical CMOS sensor.

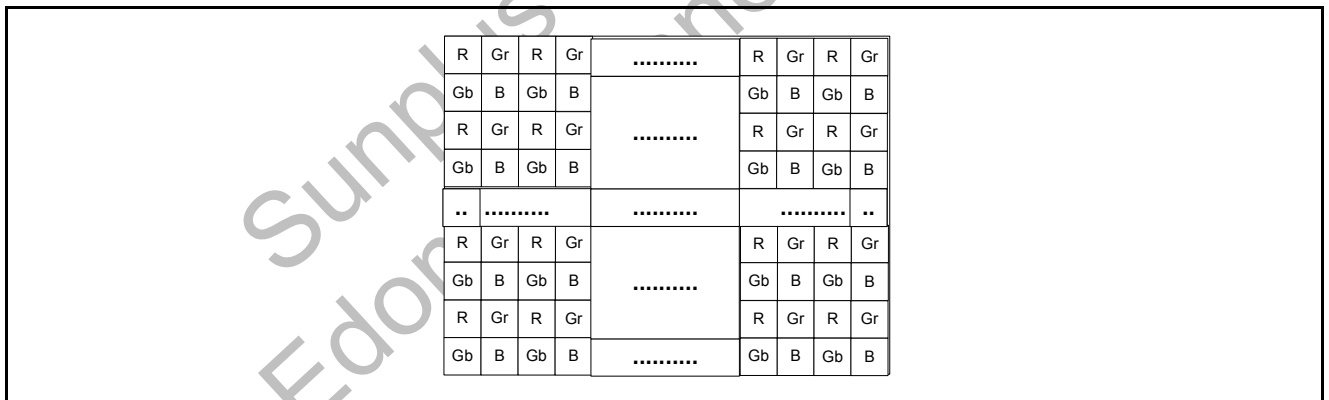


Figure 2-36: Typical Bayer Color Filter Array of a CMOS sensor

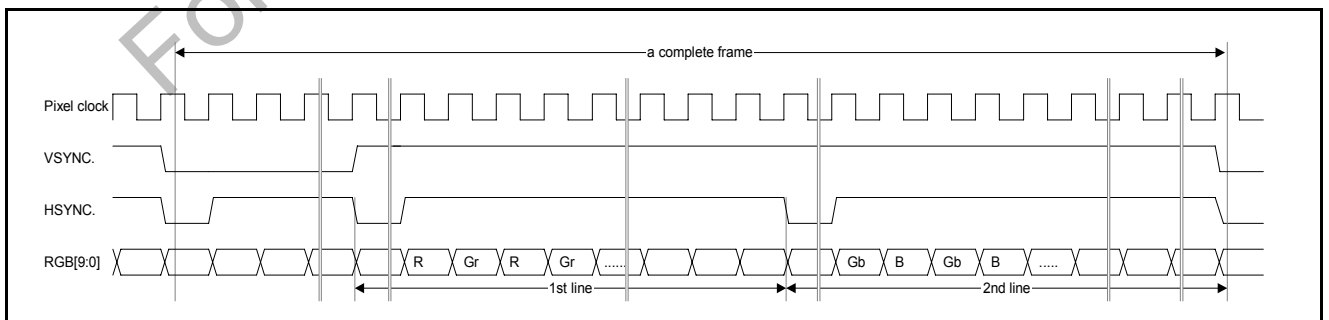


Figure 2-37: Timing diagram for bayer pattern data

■ CCIR601 Data Transfer Timing

The data is in YUV format and transmitted in YUYV interleaved sequence.

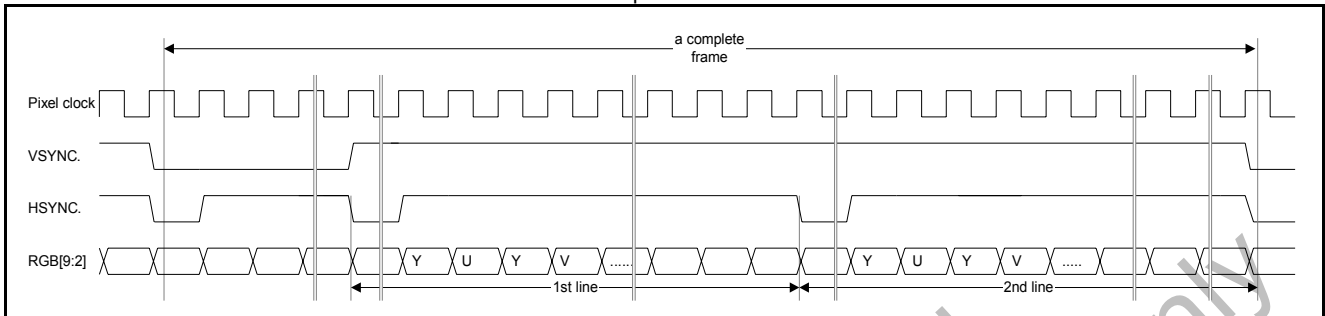


Figure 2-38: Timing diagram for CCIR601 data

■ CCIR656 Data Transfer Timing

Synchronization signals are encoded and inserted into the video YUV data. The encoded synchronization signal (the marker) starts with an EAV code (FFh, 00h, 00h, XYh) and stops with an SAV code (FFh, 00h, 00h, XYh). The SPCA554A decodes the markers and generates internal synchronization signals to construct video frame timing. The fourth byte of the marker has the HSYNC, VSYNC, field, blanking information, and so on as described below:

- Bit 7 fixed at 1.
- Bit 6 F, 0 : FIELD 0, 1: FIELD 1.
- Bit 5 V, 0: non-blanking period, 1: blanking period.
- Bit 4 H, 0: start of active video, 1: end of active video
- Bit 3 P3, V xor H
- Bit 2 P2, F xor H
- Bit 1 P1, F xor V
- Bit 0 P0, F xor V xor H

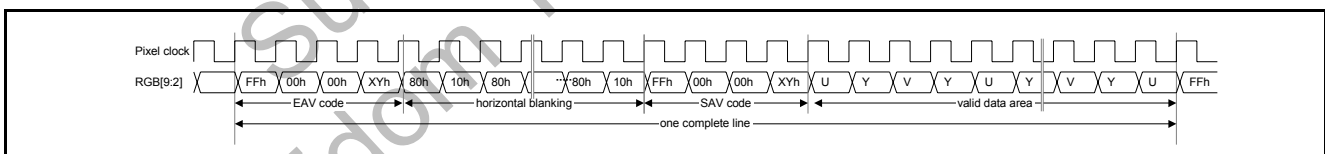


Figure 2-39: Timing diagram for CCIR656 data

SAV: Start position of horizontal valid data.

EAV: End position of horizontal valid data.

The YUV data sequence is programmable inside SPCA554A, allowing the SPCA554A to adapt to different input sources.

2.3. Storage Interface

The storage media interface is used by the SPCA554A to access external storage media. The SPCA554A always acts as a master to these storage medias. It is not possible for another bus master to control the SPCA554A through the storage media interface. The SPCA554A supports three types of storage media;

- SPI-type serial flash: Supports mode0, mode1, mode2 and mode3.
- Security Digital card: Both 1-bit and 4-bit modes. As MMC can be accessed by the 1-bit SD mode, SPCA554A also supports MMC.
- Memory Stick Card: Supports 1-bit mode for Memory Stick cards and 4-bit mode for Memory Stick Pro cards.

Table 2-4 lists pin functions according to different storage media types:

Pin Name	SD Card / MMC		Memory Stick		SPI-Type Serial Flash		IO Buffer
FMGPIO0	DAT3	B	BS	O	WP/	O	BD
FMGPIO1	CMD	B	SDIO1	B	RDY	I	BD
FMGPIO2	CLK	O	SDIO0	B	SCK	O	BD
FMGPIO3	DAT0	B	SDIO2	B	SI	O	BD
FMGPIO4	DAT1	B	SDIO3	B	RST/	O	BD
FMGPIO5	DAT2	B	SCK	O	SO	I	BD
FMGPIO6	FMGPIO6	B	FMGPIO6	B	CS/	O	BD

Table 2-4: Pins descriptions for storage interface.

Most storage media interface pins are multi-functional pins, except for the CS/ (chip select) pin for serial flash memory. The SPCA554A has a dedicated pin for CS/ of serial flash memory, allowing serial flash memory to co-exist with an SD card or an MS

card. An SD card cannot co-exist with an MS card in an SPCA554A application. Five feasible interconnections with external storage media are shown in the following figures:

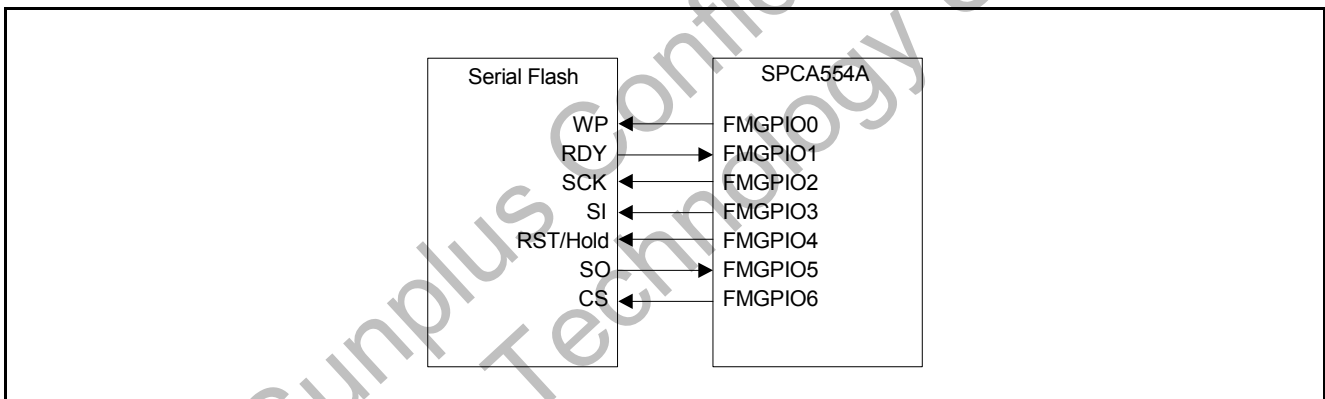


Figure 2-40: Interconnection between serial flash memory and the SPCA554A

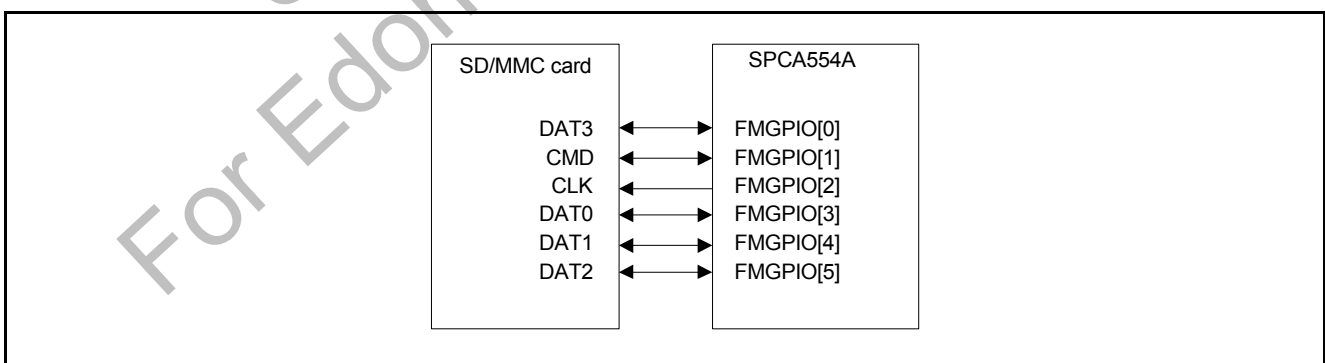


Figure 2-41: Interconnection between SD/MMC and the SPCA554A

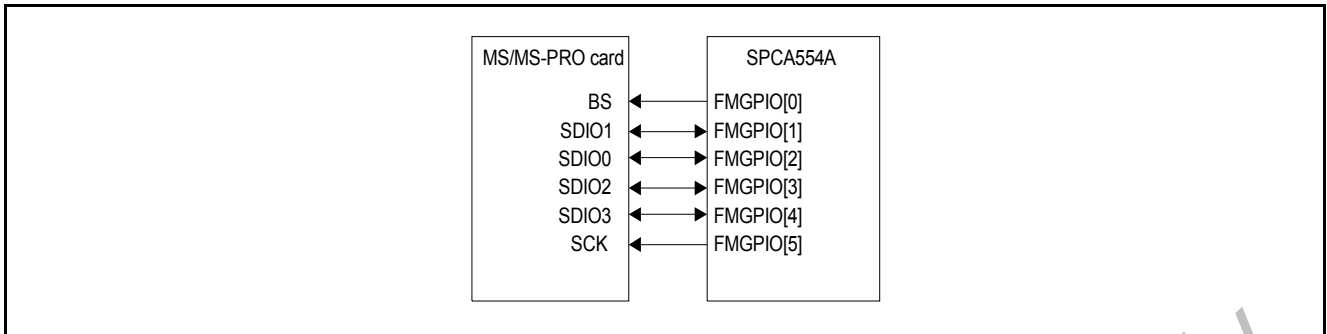


Figure 2-42: Interconnection between MS/MS-pro and the SPCA554A

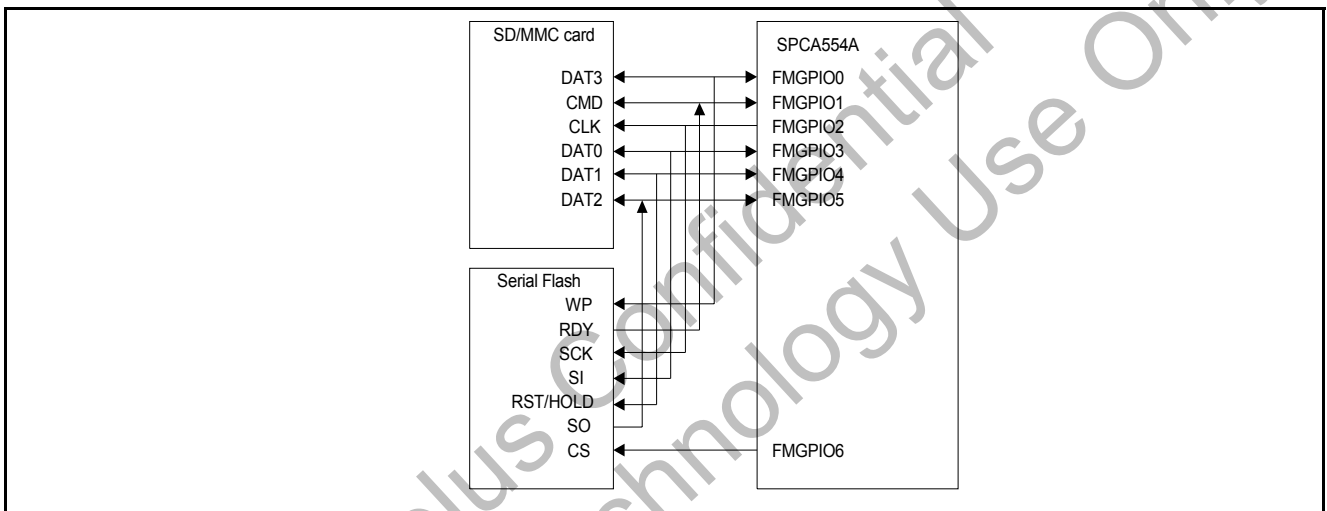


Figure 2-43: Interconnection between serial flash memory, SD/MMC and the SPCA554A

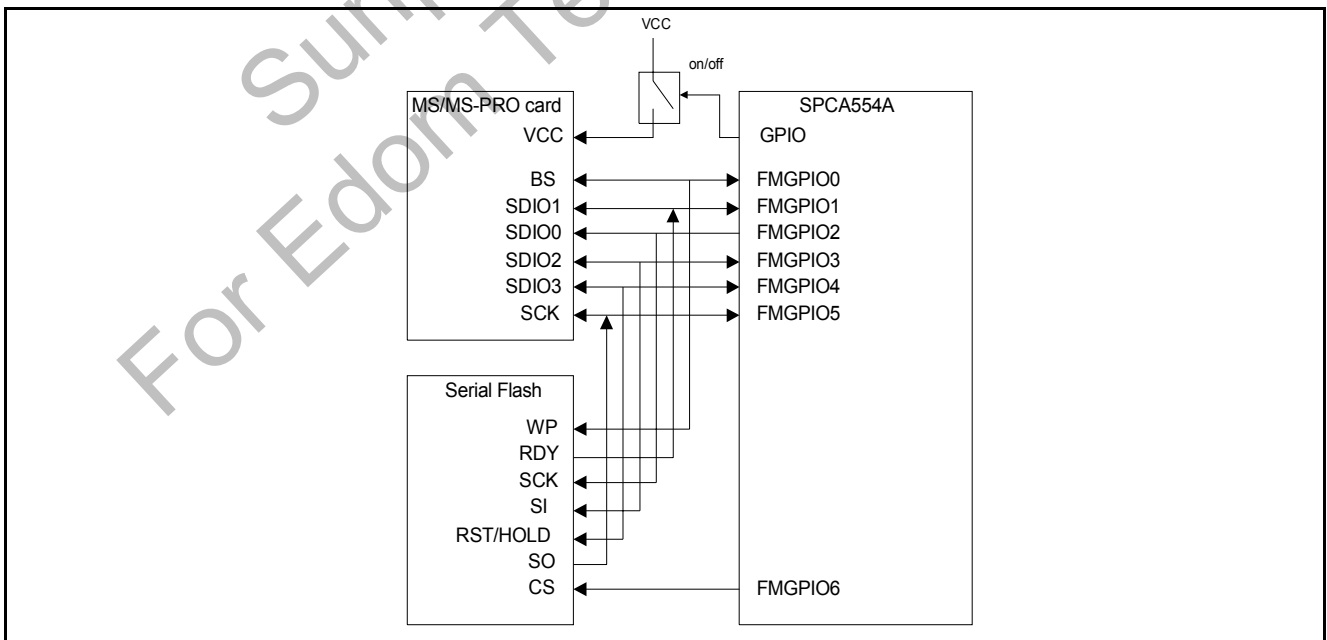


Figure 2-44: Interconnection between serial flash memory, MS/MS-pro and the SPCA554A

Note: To avoid bus conflict, MS power supply must be cut off before accessing serial flash memory. The power switch control can use any SPCA554A GPIO pin.

2.4. USB interface

The USB interface enables the SPCA554A to communicate with PCs. Data can be uploaded to PCs or downloaded from PCs via the USB bus. The ISP (In-System-Programming) function can also take advantage of the USB bus, which allows new firmware codes to be downloaded to the SPCA554A and then programmed to external serial flash memory. Another benefit of the USB interface is allowing the SPCA554A to be used as a PC-camera. The sensor module calibration in the mass production stage can

also be done via a PC host.

By setting SPCA554A internal registers, the built-in USB controller can easily be configured to three different modes; USB function mode, USB host mode and OTG mode. The USB host mode and the USB function mode share an on-chip USB 1.1 full speed transceiver, while the USB OTG mode requires an external OTG transceiver. With the external OTG transceiver, the SPCA554A turns into a USB OTG dual-role device.

Pin Name	USB Device		USB Host		USB OTG		IO Buffer
	DATA D+	B	DATA D+	B			
DP	DATA D+	B	DATA D+	B	Not used	-	USB transceiver
DM	DATA D-	B	DATA D-	B	Not used	-	USB transceiver
OTGGPIO0	OTGGPIO0	B	OTGGPIO0	B	OTG interrupt	I	BD
OTGGPIO1	OTGGPIO1	B	OTGGPIO1	B	OTG data input	I	BD
OTGGPIO2	OTGGPIO2	B	OTGGPIO2	B	OTG DATA D+	B	BD
OTGGPIO3	OTGGPIO3	B	OTGGPIO3	B	OTG DATA D-	B	BD
OTGGPIO4	OTGGPIO4	B	OTGGPIO4	B	OTG DATA OE	O	BD

Table 2-5: Pin descriptions for storage interface

Three feasible USB applications of the SPCA554A are illustrated below.

SPCA554A as a USB device:

A 1.5K Ohm pull high resistor must be connected to DP pin to the SPCA554A as a USB full speed device.

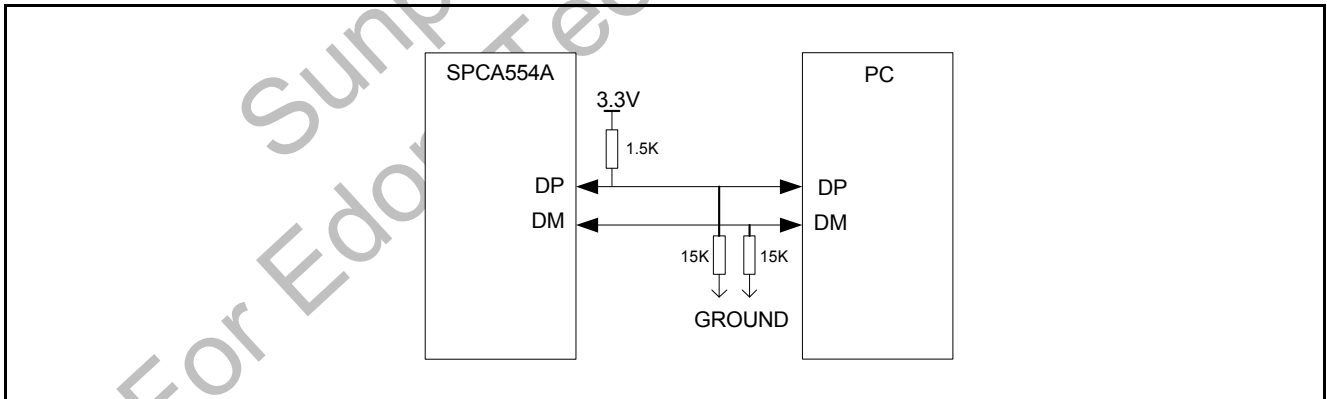


Figure 2-45: The SPCA554A as a USB device

SPCA554A as an USB host:

As a USB host, both DP and DM signals must be pulled-low by 15K Ohm resistors.

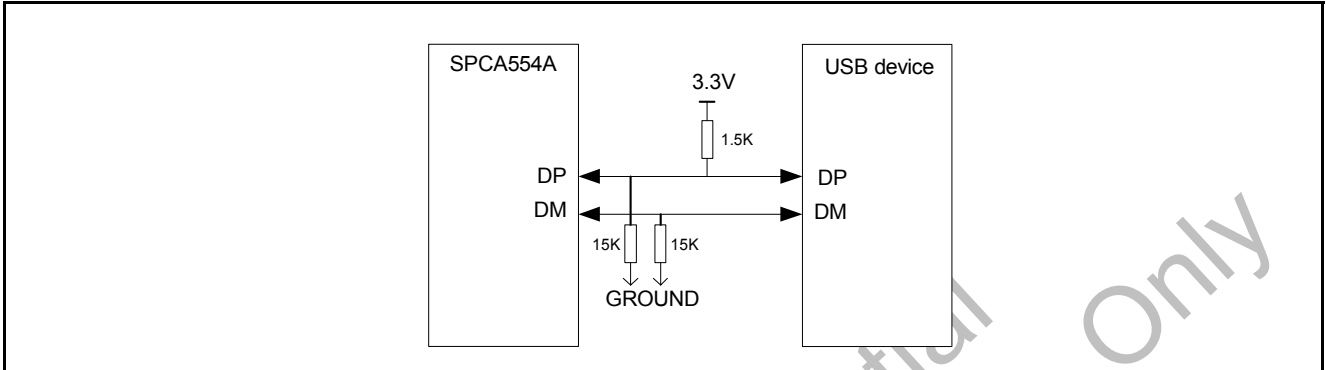


Figure 2-46: The SPCA554A as an USB host

The SPCA554 as an OTG dual-role device:

An external OTG transceiver should be connected to the OTGGPIO bus to enable the SPCA554A to communicate with other OTG dual-role devices. The OTG transceiver is controlled by an I²C bus. This I²C bus can be shared with the I²C bus

described in the sensor interface section because the I²C standard allows multiple I²C devices to be connected serially. The SPCA554A is able to gate different I²C slave addresses for each I²C devices.

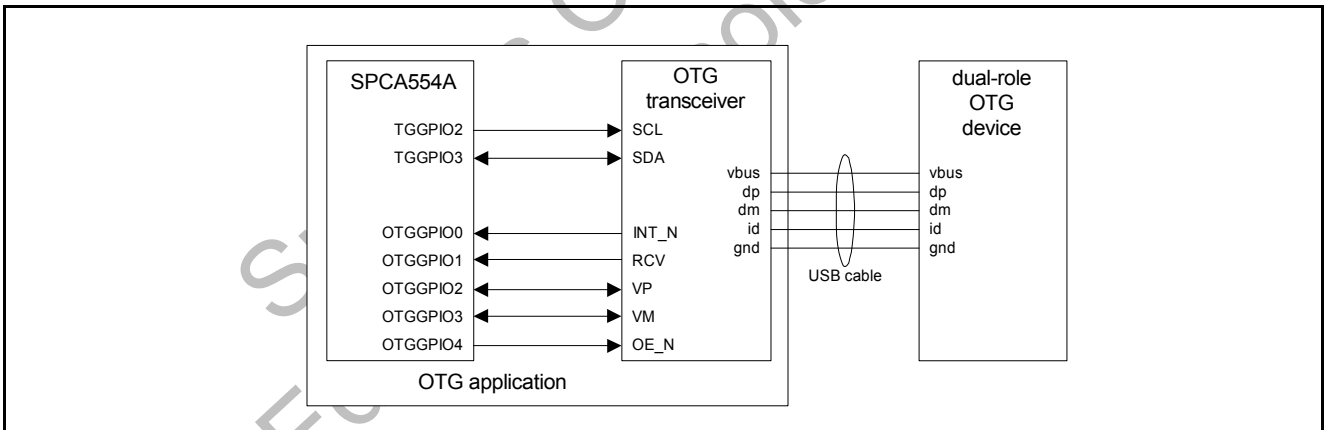


Figure 2-47: The SPCA554A as a USB OTG device

2.5. Audio Interface

The SPCA554A has both an analog and a digital audio interface. The analog interface is connected to an on-chip audio ADC and DAC. The digital audio interface is a I²S bus used to exchange data with external audio CODEC. The configuration of the audio interface is summarized below:

- Analog audio: On-chip 12-bit audio ADC and DAC.
- I²S digital audio interface for audio record: The SPCA554A can be an I²S mater or an I²S slave.
- I²S digital audio interface for audio playback: The SPCA554A can be an I²S mater or an I²S slave.
- I²S digital audio interface for audio record and playback: The SPCA554A is an I²S slave.

Figure 2-48 illustrates an analog audio application.

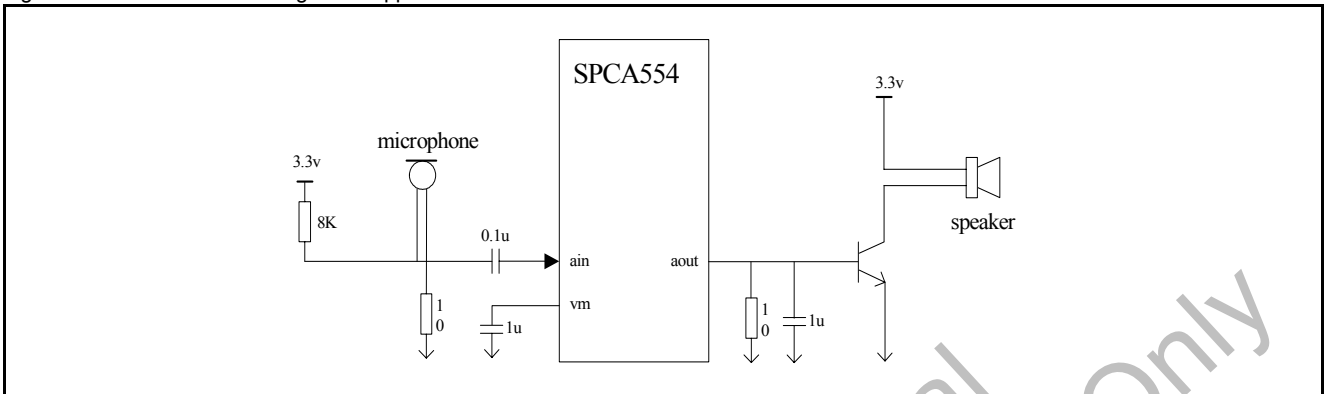


Figure 2-48: The reference circuit of the built-in ADC and DAC.

I²S-related pins and feasible modes are listed in Table 2-6 below:

Pin Name	Audio Record Only		Audio Playback Only		Audio Record And Playback		IO Buffer
AGPIO0	I2S_BCLK	B	I2S_BCLK	B	I2S_BCLK	I	BD
AGPIO1	I2S_LRC	B	I2S_LRC	B	I2S_LRC	I	BD
AGPIO2	I2S_DI	I	I2S_DO	O	I2S_DO	O	BD
GPIO5	GPIO5	B	GPIO5	B	ADCLRC	I	BD
GPIO6	GPIO6	B	GPIO6	B	I2S_DI	I	BD

Table 2-6: Pin descriptions for audio input (I²S)

In uni-directional audio applications, either in the record-only or in the playback-only mode, the SPCA554A can act as an I²S slave or an I²S master. If the SPCA554A is used as an I²S master, both I2S_BCLK and I2S_LRC are driven by the SPCA554A. On the other hand, if the SPCA554A is used as an I²S slave, both signals are driven by external audio CODEC. Also, in uni-directional applications, GPIO5 and GPIO6 are not used for the audio function.

I²S bi-directional applications require two extra pins; GPIO5 and GPIO6. With these two extra pins, the SPCA554A can record

and playback audio at the same time. In this type of application, the SPCA554A can only be an I²S slave. AGPIO2 is used to send audio data to the external audio CODEC. And GPIO6 is used to receive audio data from external audio CODEC.

I2S Audio Record-Only

The following diagram shows the connection between the SPCA554A and an external I²S audio CODEC in a record-only application. The SPCA554A uses its I²C master port to control external audio CODEC. Audio data is sent from the external I²S CODEC to the SPCA554A.

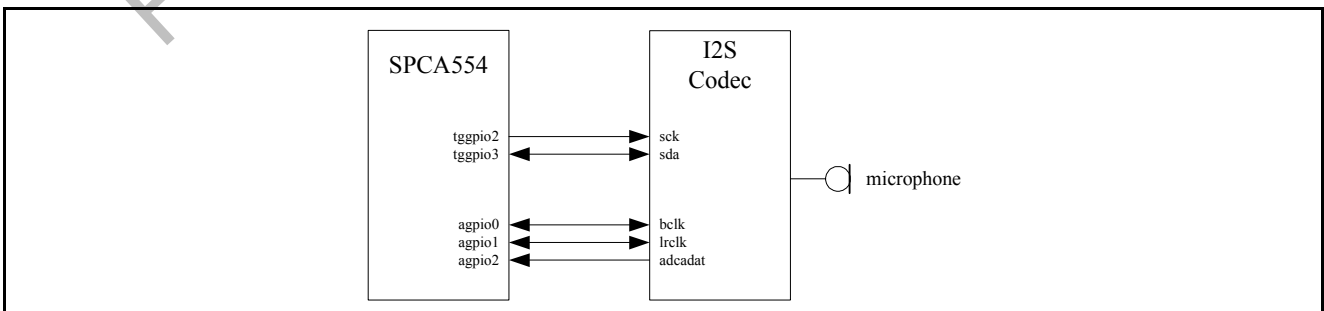


Figure 2-49: The connection between the SPCA554A and an external ADC in a record-only application

I2S Audio Playback-Only

The following diagram shows the connection between the SPCA554A and an external I²S audio CODEC in a playback-only

application. Like in the record-only configuration, the SPCA554A controls the I²S via its master I²C port. Audio data is sent from the SPCA554A to the I²S CODEC.

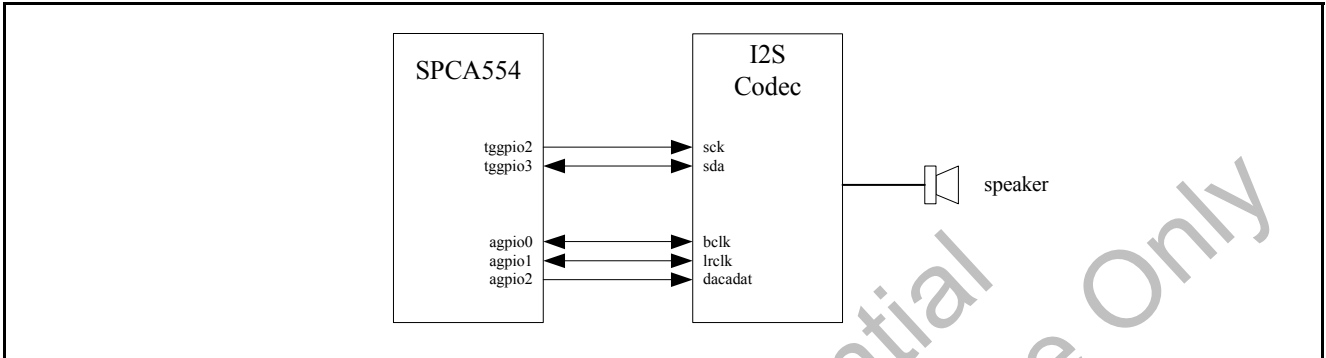


Figure 2-50: The connection between the SPCA554A and an external ADC in a playback-only application

I2S as Audio Record and Playback at the Same Time

The following diagram shows the connection between the SPCA554A and an external I²S CODEC in a bi-directional application. In this application, audio record and playback are

done at the same time. This example assumes that record and playback are operated at the same sample rate, which is controlled by LRCK. Both LRCK and BCLK signals are driven by external CODEC.

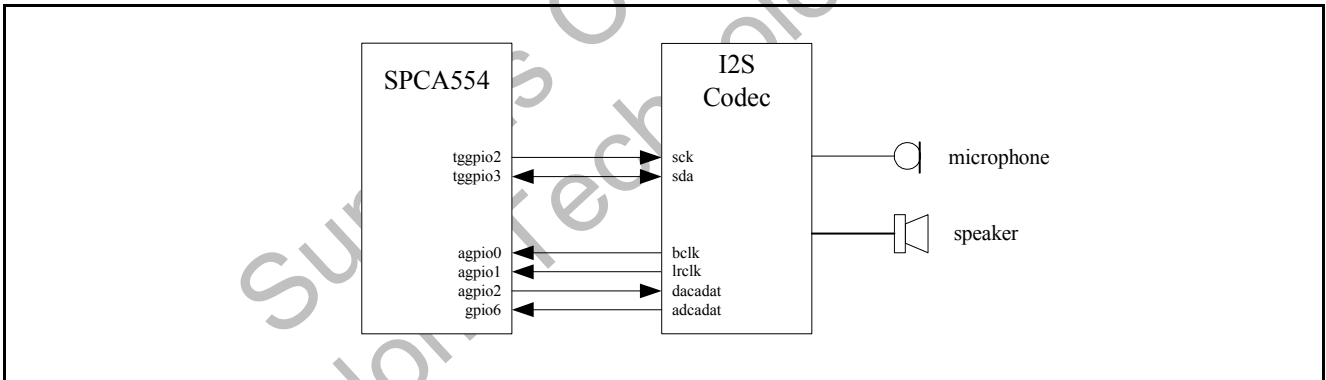


Figure 2-51: Interface connection for record and playback at the same time (same sample rate)

If audio record and playback are operated at different sample rates, then record sample rate is controlled by GPIO5 and

playback sample rate is controlled by AGPIO1. See the following illustration:

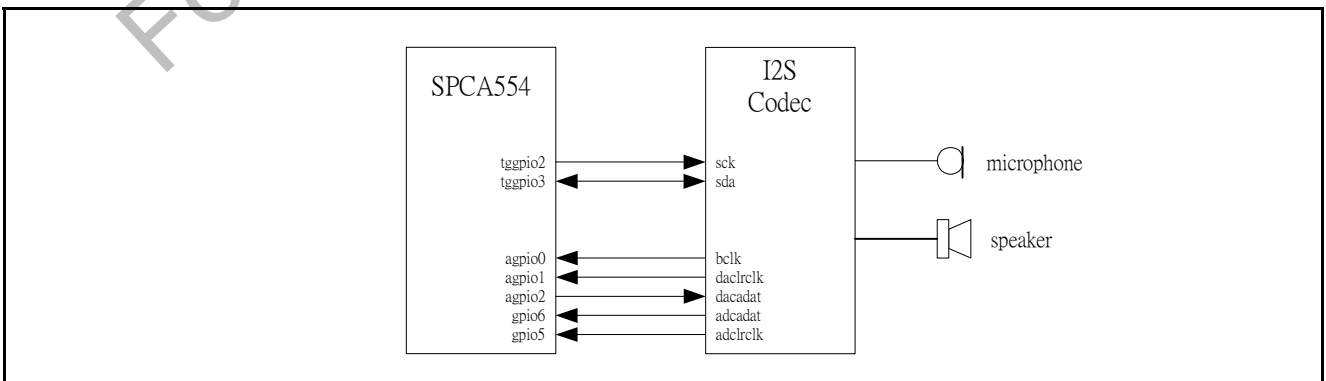


Figure 2-52: Interface connection for record and playback at the same time (different sample rate)

2.6. Display interface

The SPCA554A supports a variety of LCM interface types as described below:

- 18-bit RGB interface RGB666
- 18-bit memory RGB666
- 16-bit memory protocol RGB666, RGB565, RGB444
- 9-bit memory interface RGB666
- 8-bit memory RGB666, RGB565, RGB444, RGB332
- YUV output

Besides the color depths described above, the SPCA554A supports a very flexible RGB data sequence. All possible feasible sequences are described in this section. The following table lists pin functions according to the interface type. Note that unused pins in a specific type of LCM interface are assigned as GPIO pins.

c type of LCM interface are assigned as GPIO pins.

		18-bit RGB	18-bit Memory	16-bit Memory	9-bit Memory	8-bit Memory	YUV Output	IO Buffer
LGPIO0	O		XCS	XCS	XCS	XCS		BD
LGPIO1	O		XRD	XRD	XRD	XRD		BD
LGPIO2	O		XWR	XWR	XWR	XWR		BD
LGPIO3	B		A0	A0	A0	A0		BD
LGPIO4	B	R0	D0	D0	D0	D0	YUV0	BD
LGPIO5	B	R1	D1	D1	D1	D1	YUV1	BD
LGPIO6	B	R2	D2	D2	D2	D2	YUV2	BD
LGPIO7	B	R3	D3	D3	D3	D3	YUV3	BD
LGPIO8	B	R4	D4	D4	D4	D4	YUV4	BD
LGPIO9	B	R5	D5	D5	D5	D5	YUV5	BD
LGPIO10	B	G0	D6	D6	D6	D6	YUV6	BD
LGPIO11	B	G1	D7	D7	D7	D7	YUV7	BD
LGPIO12	B	G2	D8	D8	D8		YUV8	BD
LGPIO13	B	G3	D9	D9			YUV9	BD
LGPIO14	B	G4	D10	D10			YUV10	BD
LGPIO15	B	G5	D11	D11			YUV11	BD
LGPIO16	B	B0	D12	D12			YUV12	BD
LGPIO17	B	B1	D13	D13			YUV13	BD
LGPIO18	B	B2	D14	D14			YUV14	BD
LGPIO19	B	B3	D15	D15			YUV15	BD
LGPIO20	O	B4	D16					BD
LGPIO21	O	B5	D17					BD
LGPIO22	O	MCK					YUVCK	BD
LGPIO23	O	HS					HREF	BD
LGPIO24	O	VS					VREF	BD
LGPIO25	O	DE	XCS2	XCS2	XCS2	XCS2		BD

Table 2-7: Pin descriptions for LCM interface

The 18-bit RGB protocol uses vertical and horizontal synchronization signals (VS and HS) to construct timing of a display frame. 18-bit RGB data is sent to the LCM per clock.

All memory protocols have READ/WRITE pulses, which access

the LCM panel in the same way as a standard SRAM interface. The SPCA554A supports dual-panel applications. LCMGPIO0 is used as the chip select signal of the main LCM panel and LCMGPIO25] is used as the chip select signal of the sub-LCM panel.

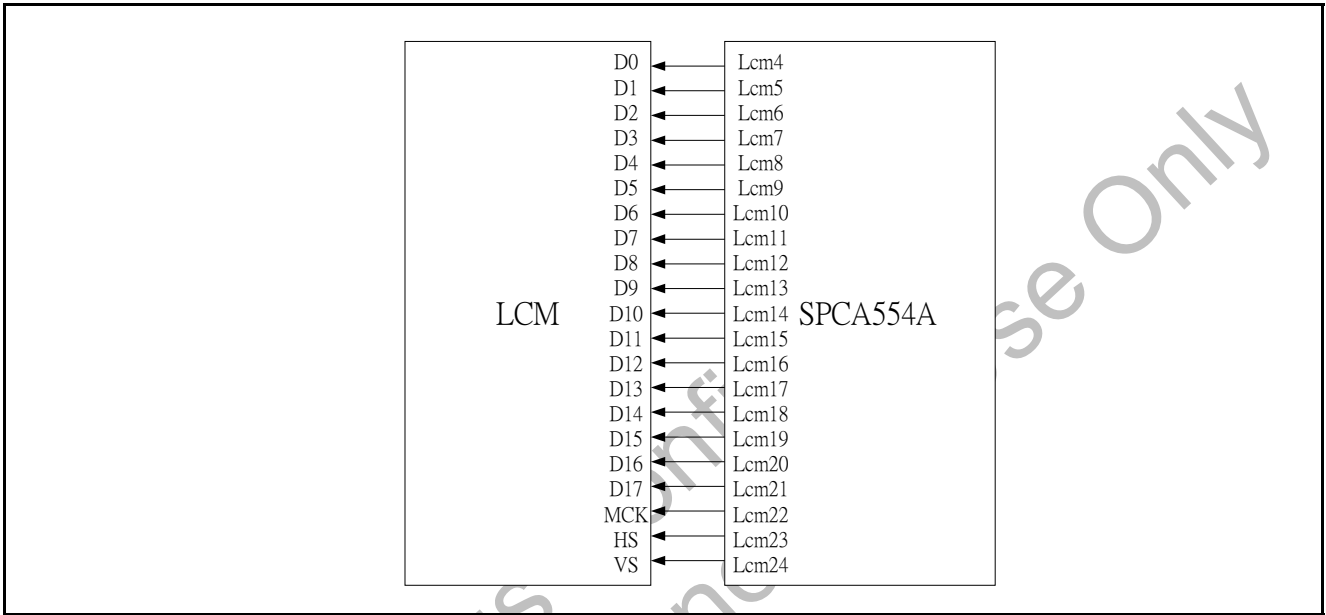


Figure 2-53: 18-bit RGB interface connection

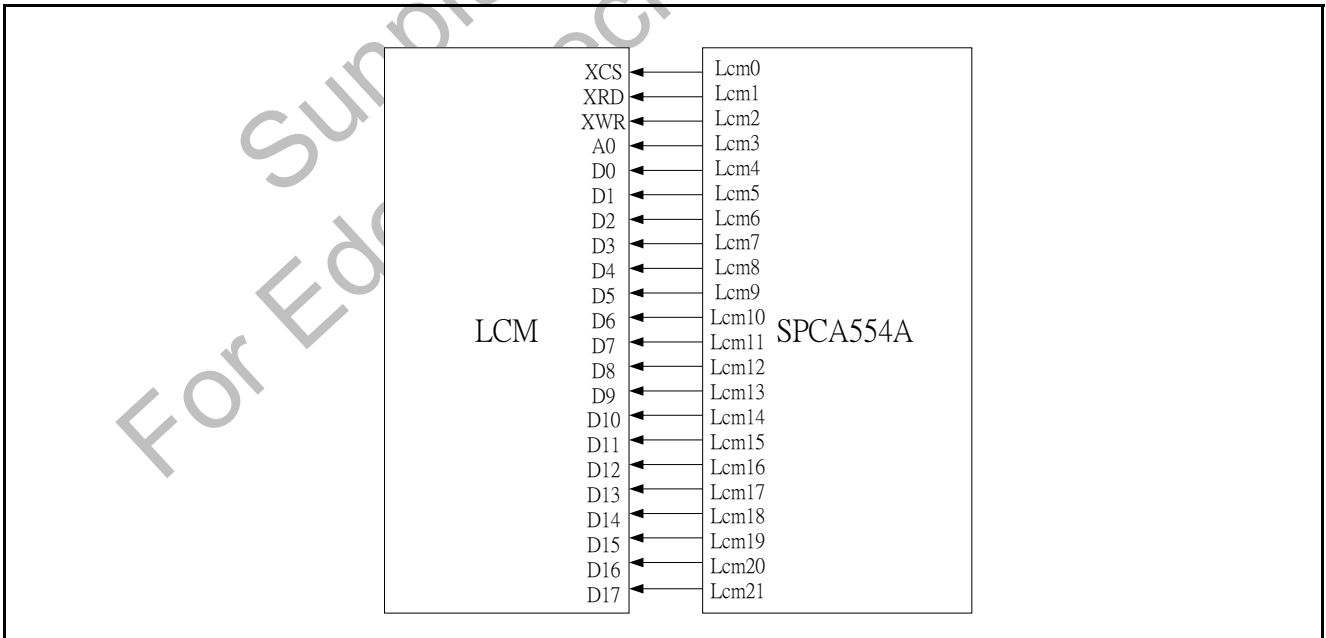


Figure 2-54: 18-bit memory interface connection

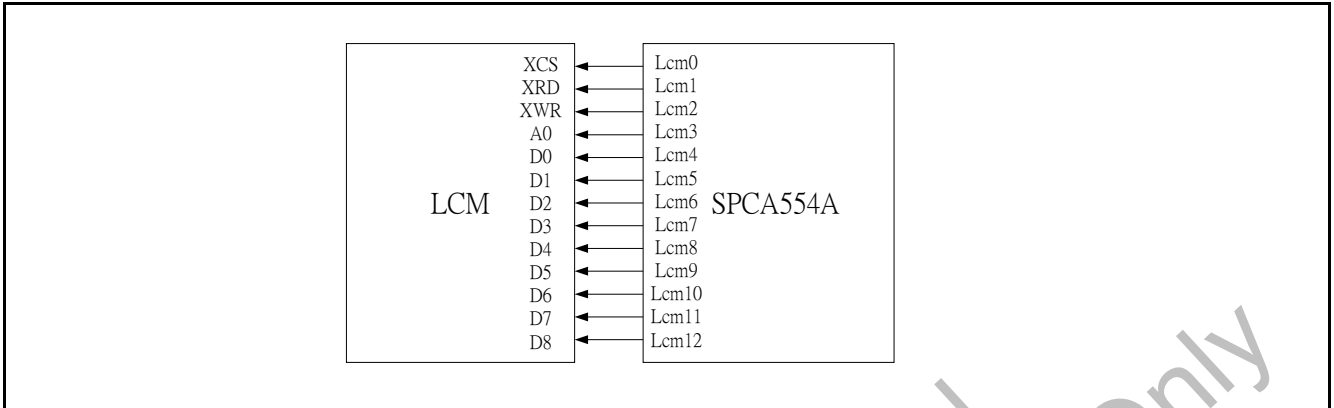


Figure 2-55: 9-bit memory interface connection

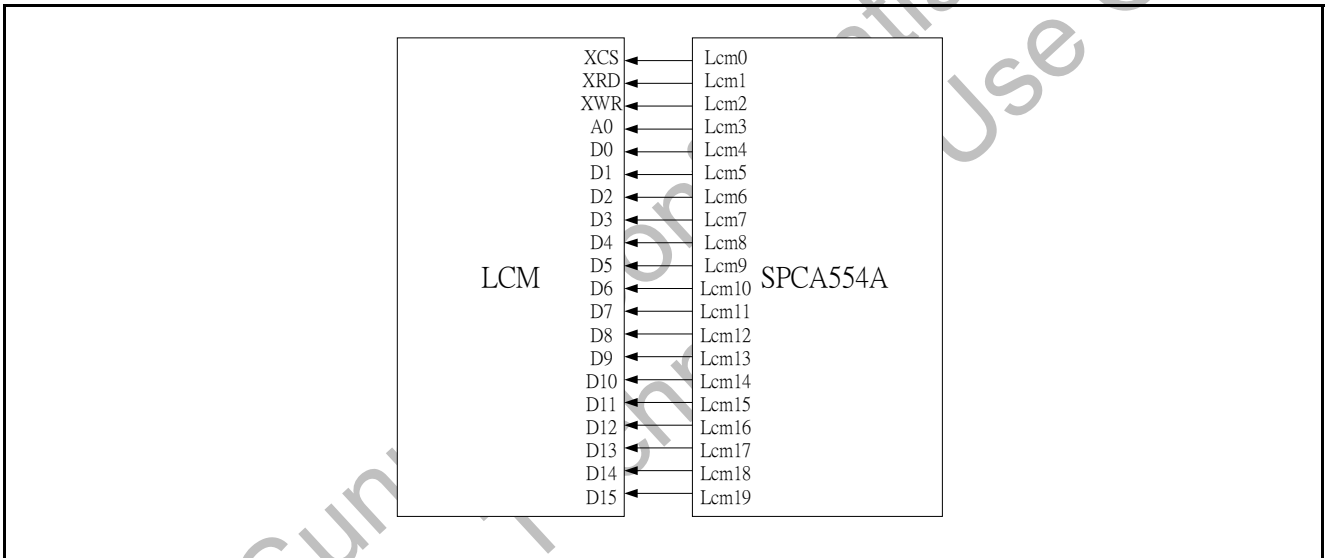


Figure 2-56: 16-bit memory interface connection

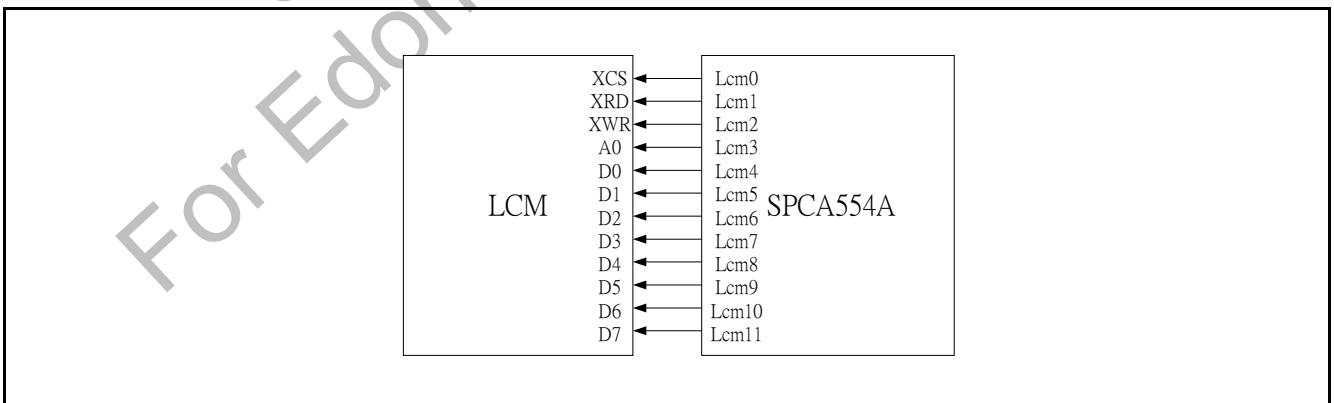
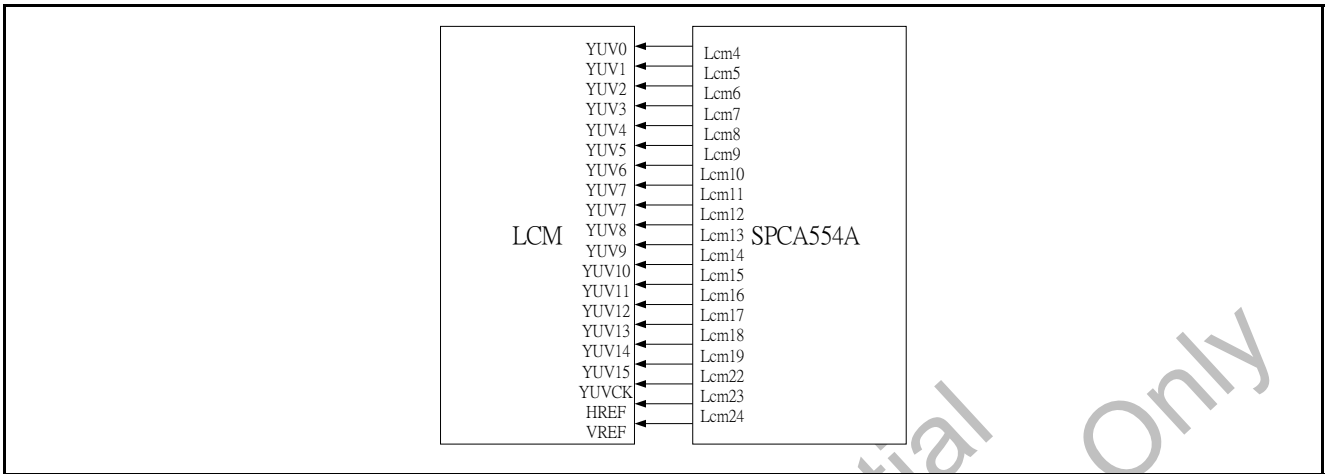
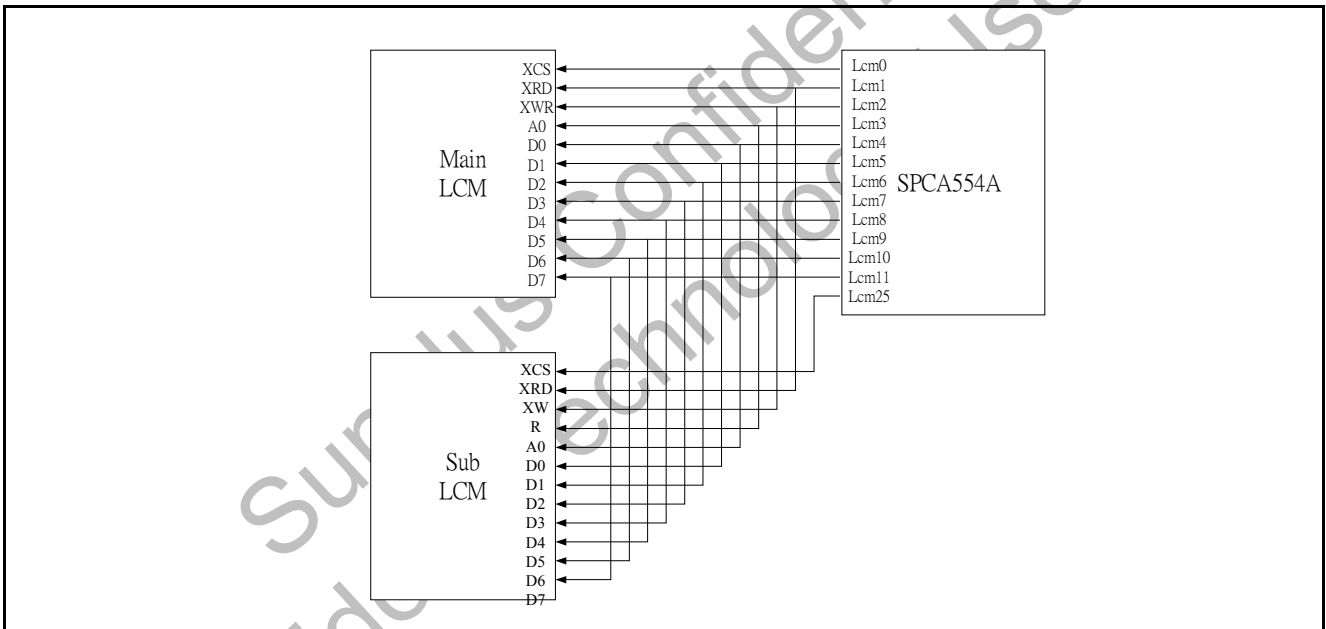


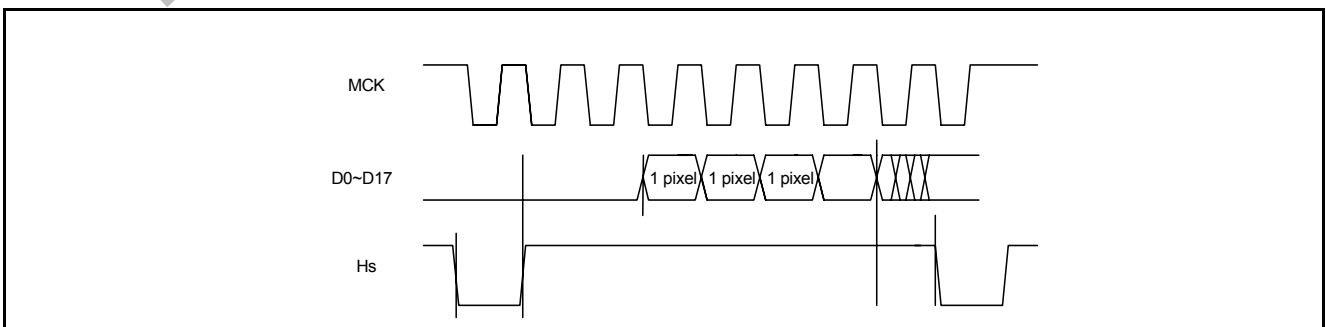
Figure 2-57: 8-bit memory interface connection


Figure 2-58: YUV interface connection

Figure 2-59: Main and sub panel interface connection

2.6.1. 18-bit RGB interface

The 18-bit RGB interface transfers data by parallel data bus D0~D17 (R0 ~ R5, G0 ~ G5, B0 ~ B5), VS, HS, and MCK. The

data sequence of D0~D17 is programmable. Waveform and data sequence are shown in the following figures:


Figure 2-60: Horizontal timing of 18-bit RGB protocol

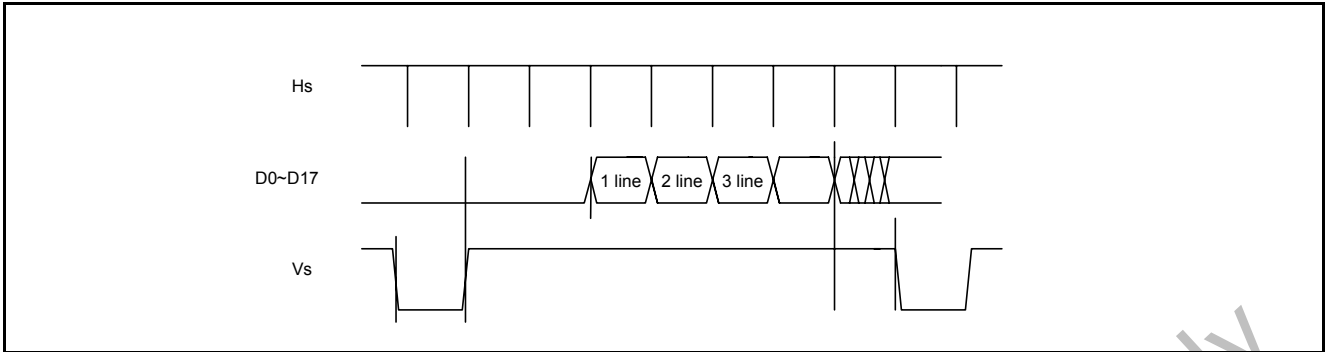


Figure 2-61: Vertical timing of 18-bit RGB protocol

Color Format	Data Sequence Option
18 bit RGB interface	{B[5:0],G[5:0],R[5:0]}
	{R[5:0],G[5:0],B[5:0]}

Table 2-8: Data sequence of the 18 bit RGB interface

2.6.2. 18-bit Memory interface

RGB data is carried by data bus D[17:0]. The A0 pin allows the SPCA554A to change the address of the register to be accessed. Typical LCMs with memory protocol allow the SPCA554A to program internal registers for initialization. In addition, there is

address mapping to the LCM data port, allowing the SPCA554A to access frame buffer inside the LCM. Waveform and data sequence are shown as follows:

Note: The SPCA554 can also read LCM data by internal CPU.

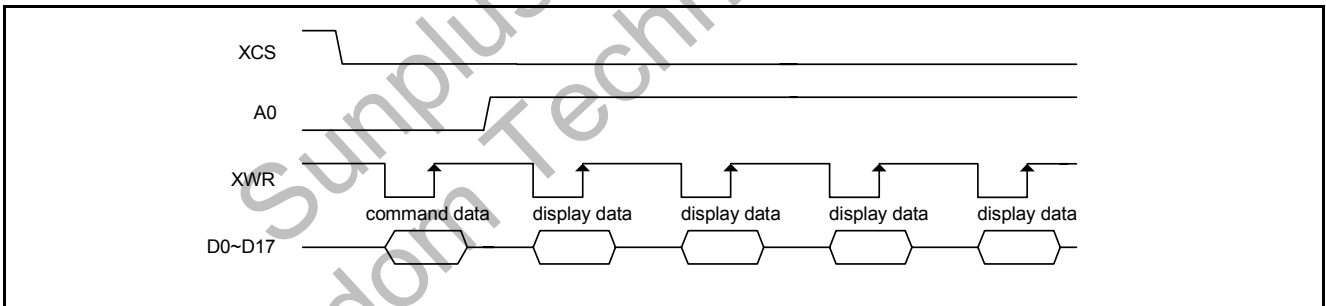


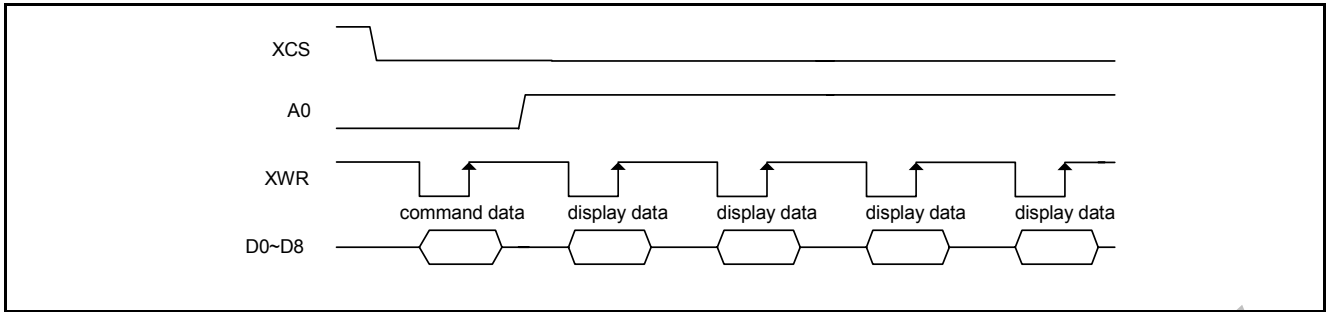
Figure 2-62: LCM 18-bit memory protocol

Color Format	Data Sequence Option
18 bit RGB 666	{B[5:0],G[5:0],R[5:0]}
	{R[5:0],G[5:0],B[5:0]}

Table 2-9: Data sequence of the 18 bit RGB 666 format

2.6.3. 9-bit Memory interface

The 9-bit memory interface transfers data by parallel data bus (D0 ~ D8), XCS, A0, XWR, and XRD. Waveform and data sequence are shown as follows:

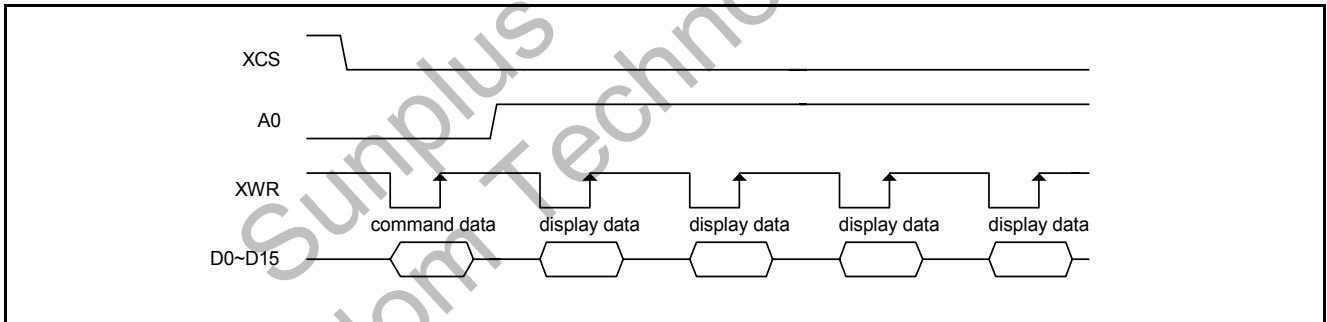

Figure 2-63: LCM 9-bit memory protocol

Color Format	Data Sequence Option
9 bit RGB 666	{G[2:0],R[5:0]}
	{B[5:0],G[5:3]}
	{R[5:0],G[2:0]}
	{G[5:3],B[5:0]}

Table 2-10: Data sequence of 9 bit RGB 666 format

2.6.4. 16-bit Memory interface

The 16-bit memory interface transfers data by parallel data bus (D0 ~ D15), XCS, A0, XWR, and XRD. Waveform and Data sequence is shown as follows:


Figure 2-64: LCM 16-bit memory protocol

Color Format	Data Sequence Option
16 bit RGB 666 NR	{B1[3:0],G1[5:0],R1[5:0]}
	{B2[1:0],G2[5:0],R2[5:0],B1[5:4]}
	{G3[5:0],R3[5:0],B2[5:2]}
	{G4[3:0],R4[5:0],B3[5:0]}
	{G5[1:0],R5[5:0],B4[5:0],G4[5:4]}
	{R6[5:0],B5[5:0],G5[5:2]}
	{R7[3:0],B6[5:0],G6[5:0]}
	{R8[1:0],B7[5:0],G7[5:0],R7[5:4]}
	{B8[5:0],G8[5:0],R8[5:2]}

Color Format	Data Sequence Option
	{R1[5:0],G1[5:0],B1[3:0]} {B1[5:4],R2[5:0],G2[5:0],B2[1:0]} {B2[5:2],R3[5:0],G3[5:0]} {B3[5:0],R4[5:0],G4[3:0]} {G4[5:4],B4[5:0],R5[5:0],G5[1:0]} {G5[5:2],B5[5:0],R6[5:0]} {G6[5:0],B6[5:0],R7[3:0]} {R7[5:4],G7[5:0],B7[5:0],R8[1:0]} {R8[5:2],G8[5:0],B8[5:0]}
16 bit RGB 565	{B[4:0],G[5:0],R[4:0]} {R[4:0],G[5:0],B[4:0]}
16 bit RGB 444	{4'b0,B[3:0],G[3:0],R[3:0]} { B[3:0],G[3:0],R[3:0],4'b0} {4'b0,R[3:0],G[3:0],B[3:0]} { R[3:0],G[3:0],B[3:0],4'b0}
16 bit RGB 666	{4'b0,G1[5:0],R1[5:0]} {4'b0,R2[5:0], B1[5:0]} {4'b0, B2[5:0], G2[5:0]} {2'b0,G1[5:0], 2'b0,R1[5:0]} {2'b0,R2[5:0], 2'b0, B1[5:0]} {2'b0, B2[5:0], 2'b0, G2[5:0]} {G1[5:0],R1[5:0], 4'b0} {R2[5:0], B1[5:0], 4'b0} {B2[5:0], G2[5:0], 4'b0} {G1[5:0], 2'b0,R1[5:0], 2'b0} {R2[5:0], 2'b0, B1[5:0], 2'b0} {B2[5:0], 2'b0, G2[5:0], 2'b0} {4'b0,R1[5:0],G1[5:0]} {4'b0,B1[5:0], R2[5:0]} {4'b0, G2[5:0], B2[5:0]} {2'b0,R1[5:0], 2'b0,G1[5:0]} {2'b0,B1[5:0], 2'b0, R2[5:0]} {2'b0, G2[5:0], 2'b0, B2[5:0]} {R1[5:0],G1[5:0], 4'b0} {B1[5:0],R2[5:0], 4'b0} { G2[5:0], B2[5:0], 4'b0} {R1[5:0], 2'b0,G1[5:0], 2'b0} {B1[5:0], 2'b0, R2[5:0], 2'b0} {G2[5:0], 2'b0,B2[5:0], 2'b0}

Table 2-11: Data sequence of 16 bit RGB format

2.6.5. 8-bit Memory interface

The 8-bit memory interface transfers data by parallel data bus (D0 ~ D7), XCS, A0, XWR, and XRD. The waveform and data sequence is shown as follows:

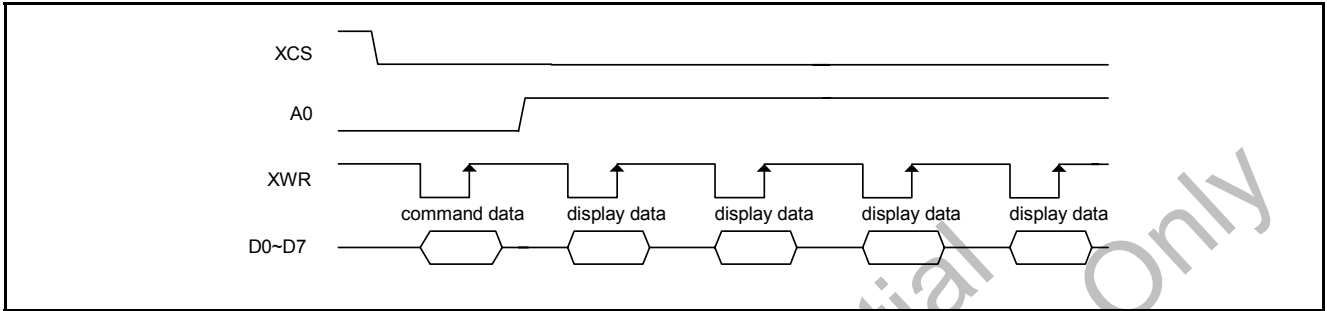


Figure2-65: LCM 8-bit memory protocol

Color Format	Data Sequence Option
8 bit RGB 666 NR	{G1[1:0],R1[5:0]} {B1[3:0],G1[5:2]} {R2[5:0],B1[5:4]} {B2[1:0],G2[5:0]} {R3[3:0],B2[5:2]} {G3[5:0],R3[5:4]} {R4[1:0],B3[5:0]} {G4[3:0],R4[5:2]} {B4[5:0],G4[5:4]}
	{R1[5:0],G1[1:0]} {G1[5:2],B1[3:0]} { B1[5:4], R2[5:0]} { G2[5:0], B2[1:0]} { B2[5:2], R3[3:0]} { R3[5:4], G3[5:0]} { B3[5:0], R4[1:0]} { R4[5:2], G4[3:0]} { G4[5:4], B4[5:0]}
8 bit RGB 332	{B[1:0],G[2:0],R[2:0]}
	{R[2:0],G[2:0],B[1:0]}
8 bit RGB 444	{G1[3:0],R1[3:0]} {R2[3:0],B1[3:0]} {B2[3:0],G2[3:0]}
	{R1[3:0],G1[3:0]} {B1[3:0],R2[3:0]} {G2[3:0],B2[3:0]}
8 bit RGB 565	{G1[2:0],R1[4:0]} {B1[4:0],G1[5:3]}
	{R1[4:0],G1[5:3]} {G1[2:0],B1[4:0]}

Color Format	Data Sequence Option
8 bit RGB 666	{2'b0,R[5:0]} {2'b0,G[5:0]} {2'b0,B[5:0]}
	{R[5:0], 2'b0} {G[5:0], 2'b0} {B[5:0], 2'b0}

Table 2-12: Data sequence of 8 bit RGB format

2.6.6. YUV data output

The SPCA554A supports continuous YUV image data output. In this mode, the SPCA554A outputs data with its accompanying clock, vertical reference signal and horizontal reference signal. Data should be sampled at the rising edge of clock. The data bus width can be 8-bits wide or 16-bits wide. The clock frequency is automatically adjusted by the SPCA554A according to the frame rate of the sensor. The maximum frequency is 24MHz. The YUV data output waveform is shown below. The

valid data is sampled when both horizontal and vertical reference signals are high, and horizontal and vertical reference signals may be longer than one clock cycle but they always change their state at the falling edge of clock. YUV output is not normally used to interface with an LCM panel even though it shares LCM interface pins. This YUV output function is used to convert Bayer pattern raw data into YUV data, taking advantages of the SPCA554A internal image enhancement function.

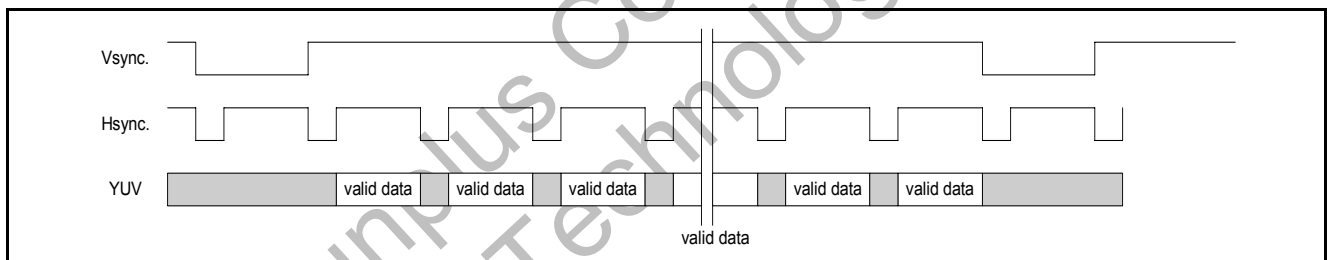


Figure 2-66: YUV data output

8-bit YUV output timing

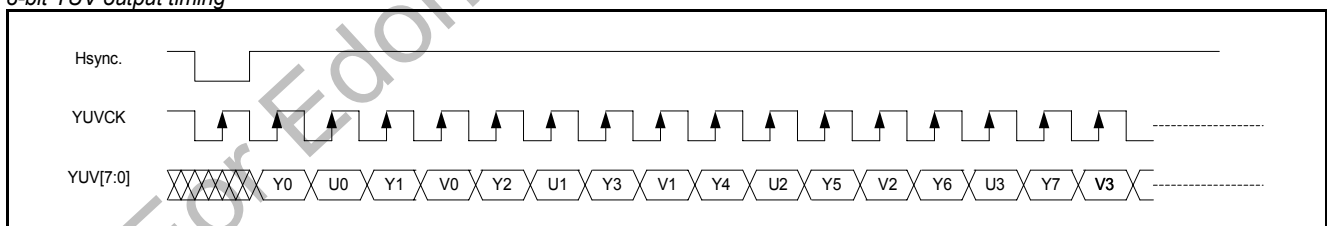


Figure 2-67: 8-bit YUV data output

16-bit YUV output timing

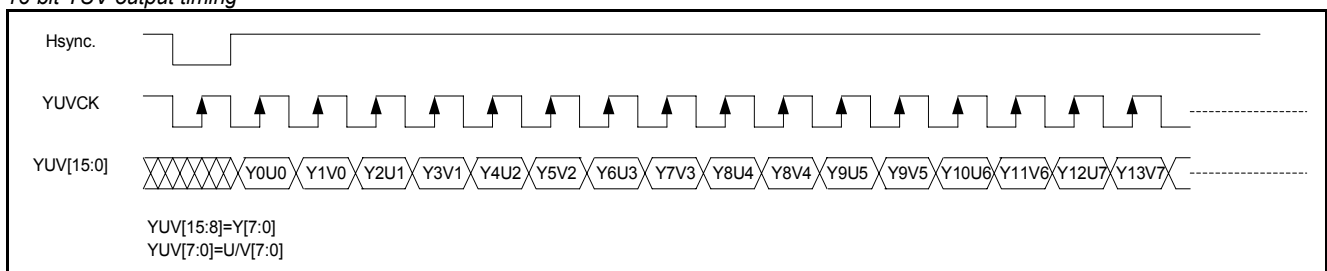


Figure 2-68: 16-bit YUV data output

3. RESET AND IO-TRAP

Power-on reset of the SPCA554 can be designed by a simple RC-delay circuit. The SPCA554 reset pin connects to an internal Schmitt-Triggered buffer. The SPCA554 internal reset circuitry delays the external reset for 10 ms, allowing the stabilization of the on chip phase lock loop. The delayed reset signal (i.e., internal reset signal) is used to reset all internal modules of the SPCA554A. At the end of the internal reset signal, the SPCA554A latches values on the LGPIO bus to determine the

IO-TRAP configuration. Core power must be supplied before IO power to avoid accidental signal conflicts. During the IO-TRAP working period (10 ms after external reset), the LCMGPIO bus must be stable because the SPCA554A latches the bus value for its power-on configuration. If the IO-TRAP latches false values, the SPCA554A cannot function normally. The trap signal is an observable IO pin useful for IO-TRAP function debug.

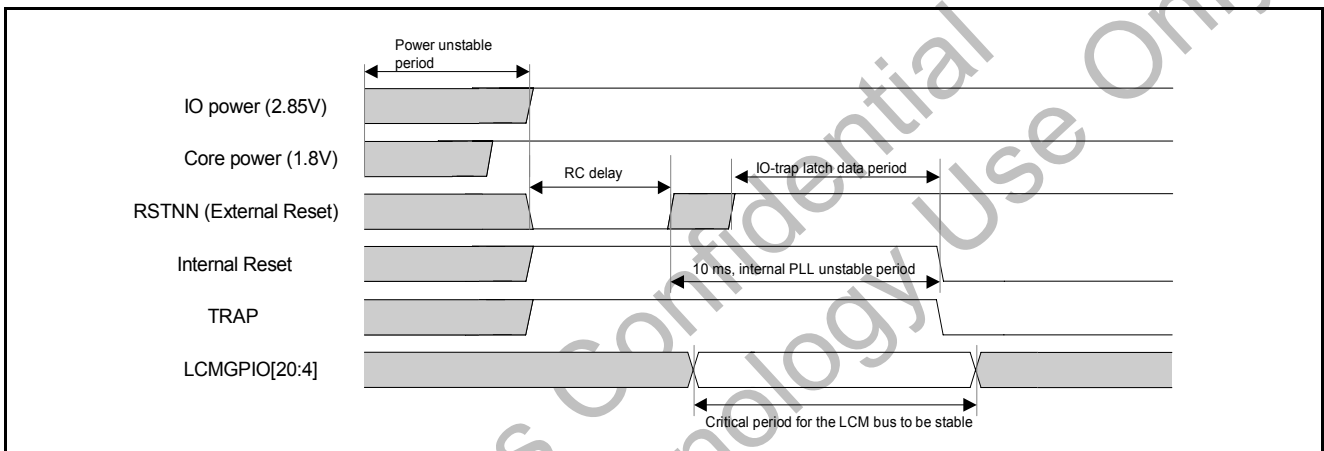


Table 3-1: Description of Figure: 3-1

The following figure shows a conceptual internal reset circuit. The RSTNN pin can also be driven by the host processor (with GPIO). But the host processor must guarantee that RSTNN is de-asserted after power is stable.

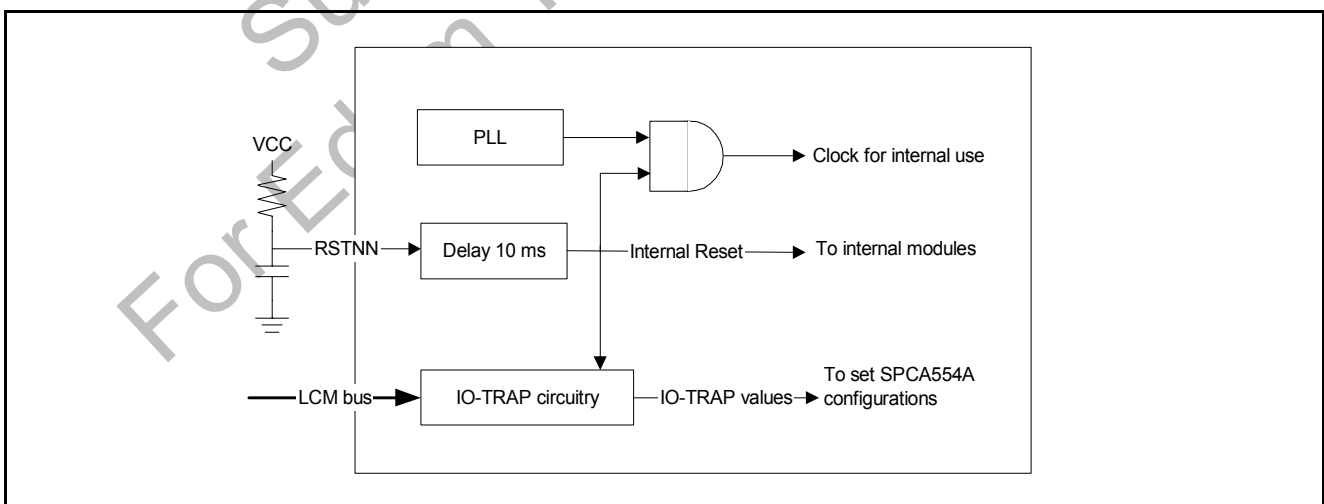


Figure 3-2: SPCA554A internal reset circuit

Besides hardware reset, the SPCA554A has software set to all the internal modules. Internal modules, which can be reset by software, include sensor controller, color DSP, Display controller, host bridge, 2D graphics engine, USB controller, MPEG/JPEG

engine, SDRAM controller, storage media controller and DMA controller. These modules can be reset independently by the internal RISC CPU or by the host controller. The RISC CPU can also be reset by the host processor.

4. BOOT SEQUENCE

The SPCA554A supports two boot sources. The first is boot from the host processor, the other is boot from the internal boot ROM. The boot source is selected by IO-TRAP[7].

4.1. Boot From Internal Boot ROM

The SPCA554A internal boot ROM is a 16K-byte mask ROM, which contains the boot code. The boot code includes the

program shadow function from the external serial flash memory to the stacked low power SDRAM. The boot code also includes the ISP (In-system-programming) function for host processor update of code in external serial flash. Program shadowing is code moved from external serial flash memory to stacked low power SDRAM, and then being executed. Figure 4.1 shows the flow of booting from the internal boot ROM.

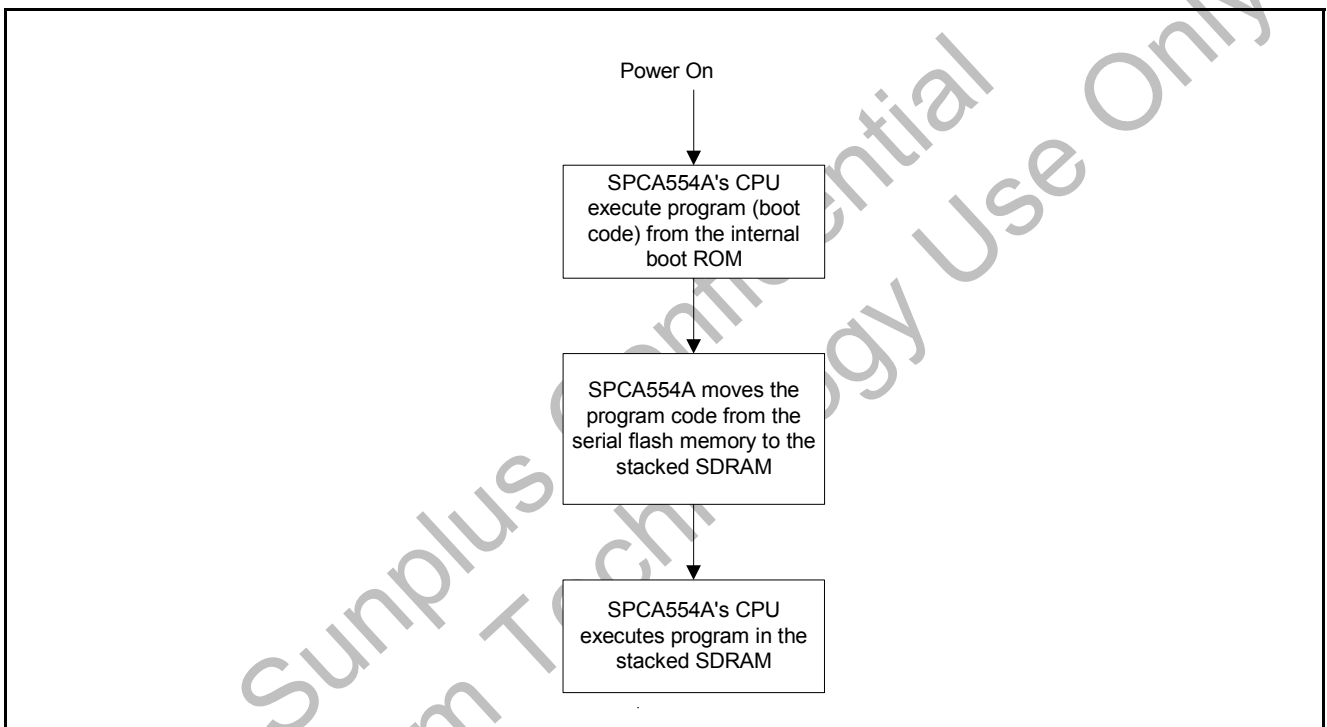


Figure 4-1: Boot from internal boot ROM

4.2. Boot From The Host Processor

If this configuration is selected, the SPCA554A internal CPU stays in the reset state after power-on. The internal CPU does not start to execute program in the stacked SDRAM until after the "PgmStMark" register in the host bridge is set to 0x5A and the "hcpurst" register is cleared to 0. From the host processor point-of-view, the host processor must download the program code to the stacked SDRAM of the SPCA554A, set the "PgmStMark" register to 0x5A, and set the "hcpurst" register to 0,

and then the SPCA554A internal CPU will then start running. Figure 4.2 shows the operational flow from the host processor point-of-view. Note that the SPCA554A automatically generates check sum from the program code it receives from the host processor. And that the total byte count of the program code is also recorded in the host bridge register. After the code is downloaded, the host processor should check the check sum and byte count of the program code for data integrity.

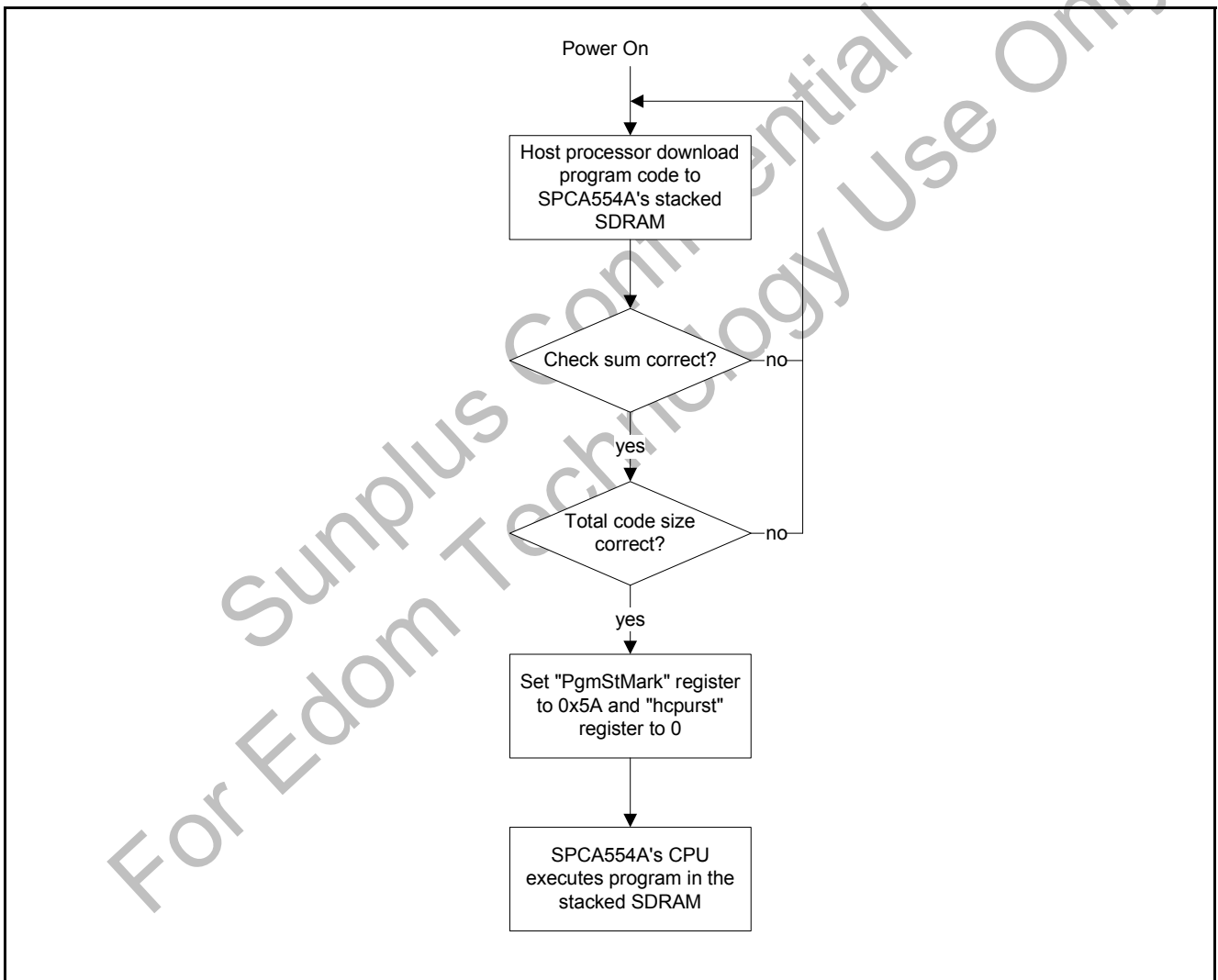


Figure 4-2: Boot from the host processor

5. POWER MANAGEMENT

5.1. Internal Clock Control

All major clocks to the internal building blocks of the SPCA554A can be turned off by software. The following diagram shows the internal clock structure of the SPCA554A. To achieve minimum

power consumption, these clocks must be manipulated according to the tasks that the SPCA554A executes. Unused clocks should be turned off to minimize power consumption.

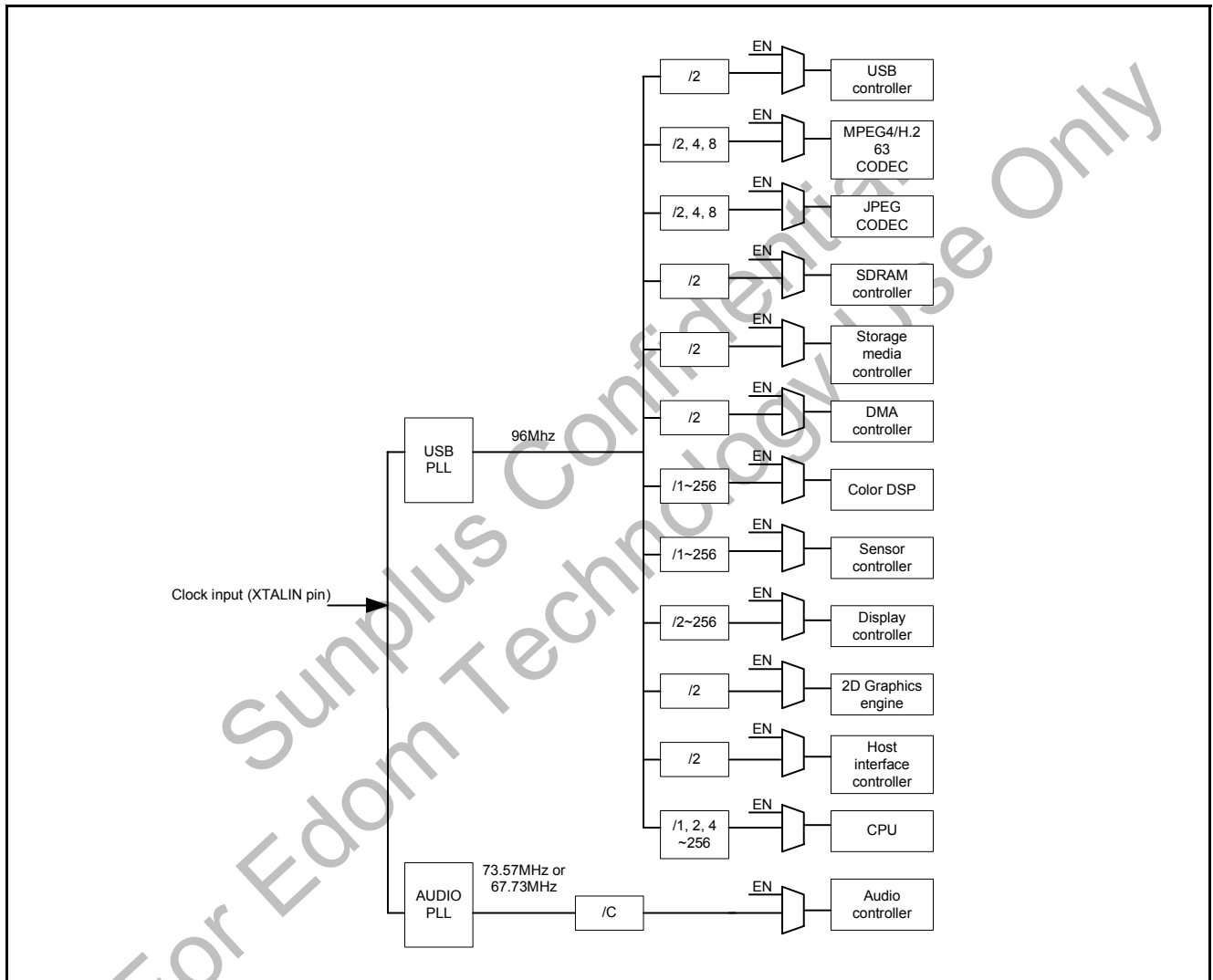


Figure 5-1: Internal clocks of the SPCA554A

The clock usages in different operation modes are show in the following table:

	USB Controller	Audio Controller	MPEG CODEC	JPEG CODEC	SDRAM Controller	Storage media Controller	DMA Controller	Color DSP	Sensor Controller	Display Controller	2D Graphics Host Bridge	RISC CPU
Idle	N	N	N	N	Y	N	N	N	N	N	N	Y
Reset	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SPCA554A Boot	N	N	N	N	Y	O(1)	O(1)	N	N	N	N	O(1)
Camera Preview	N	N	N	N	Y	N	N	Y	Y	Y	O(3)	Y

	USB Controller	Audio Controller	MPEG CODEC	JPEG CODEC	SDRAM Controller	Storage media Controller	DMA Controller	Color DSP	Sensor Controller	Display Controller	2D Graphics	Host Bridge	RISC CPU
PC-Camera	Y	N	N	Y	Y	N	Y	Y	Y	Y	N	Y	Y
Still image Capture	N	N	N	Y	Y	Y	Y	Y	Y	Y	O(3)	Y	Y
Still Image Playback	N	N	N	Y	Y	Y	Y	N	N	Y	O(3)	Y	Y
Video Recording	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	O(3)	Y	Y
Video Playback	N	Y	Y	Y	Y	Y	Y	Y	N	Y	O(3)	Y	Y
Audio Record/Playback	N	Y	N	N	Y	Y	Y	N	N	Y	N	Y	Y
JAVA Application	N	Y	N	N	Y	Y	Y	N	N	Y	Y	Y	Y
Host Compression	N	N	Y	Y	Y	N	N	N	N	Y	Y	Y	Y
Screen Saver	N	N	Y	Y	Y	N	N	N	N	Y	Y	Y	Y
LCM Bypass Mode (Suspend)	N	N	N	N	N	N	N	N	N	N	N	N	N

Table 5-1: Power management for the SPCA554A

N: Clock to the module should be turned-off.

Y: Clock to the module must be turn-on.

O(1): Clock supply depends on the boot source. If the SPCA554A boots from serial flash memory, the storage media controller, the DMA controller and the CPU must be supplied with clocks.

O(2): If the SPCA554A boots from the host, the clock must be turned-on.

O(3): Whether the 2D graphics engine clock must be turned on depends on the application. For example, in still image capture, the 2D graphics engine is required only to create photo sticker. Plain image capture does not require the 2D graphics engine to function.

5.2. Deep Power Down Mode

The SPCA554A supports the deep power down mode, in which all the internal clocks stop. The PLL also stops working in the deep power down mode. The deep power down mode is often referred to as the suspend mode in this specification. All functions of the SPCA554A do not work in the suspend mode except the bypass function. The host processor can still access the LCM by asserting the BYPASS signal.

In the suspend mode, the stacked SDRAM can be put into the self-refresh mode or simply cut off its power supply. If the SDRAM power supply is cut off, all data in SDRAM is lost, including the program code. The SPCA554A will need to reload the program code from the serial flash memory or from the host processor. If the SDRAM is put into the self-refresh mode in the suspend mode, the data in the SDRAM is retained. In this case,

waking-up the SPCA554A is faster, but the suspend mode power consumption is higher. All SDRAM interface pins are driven low by the SPCA554A to avoid current leakage.

To avoid current leakage in the suspend mode, all interface pins of the SPCA554A must be carefully managed. Depending on the application circuit, the suspend management firmware must put each pin into "driven-high", "driven low" or "tri-state" to avoid bus conflict and DC current.

The SPCA554A is put into the suspend mode by internal register programming. It can only be awakened by register access from the host interface. Nevertheless, access to the LCM in the suspend mode does not wake-up the SPCA554A because the access cycle is re-directed to the LCM.

6. FUNCTION DESCRIPTION

6.1. Operation Overview

This chapter describes the data processing flows of major functions provided by the SPCA554A. It includes digital camera preview (view finder), still image capture, still image playback, video capture and video playback. The SPCA554A has built in

plenty of scaling functions and rotation/flipping functions in many internal modules. These functions are carefully designed to meet the best image quality and should be activated according to different operating flows.

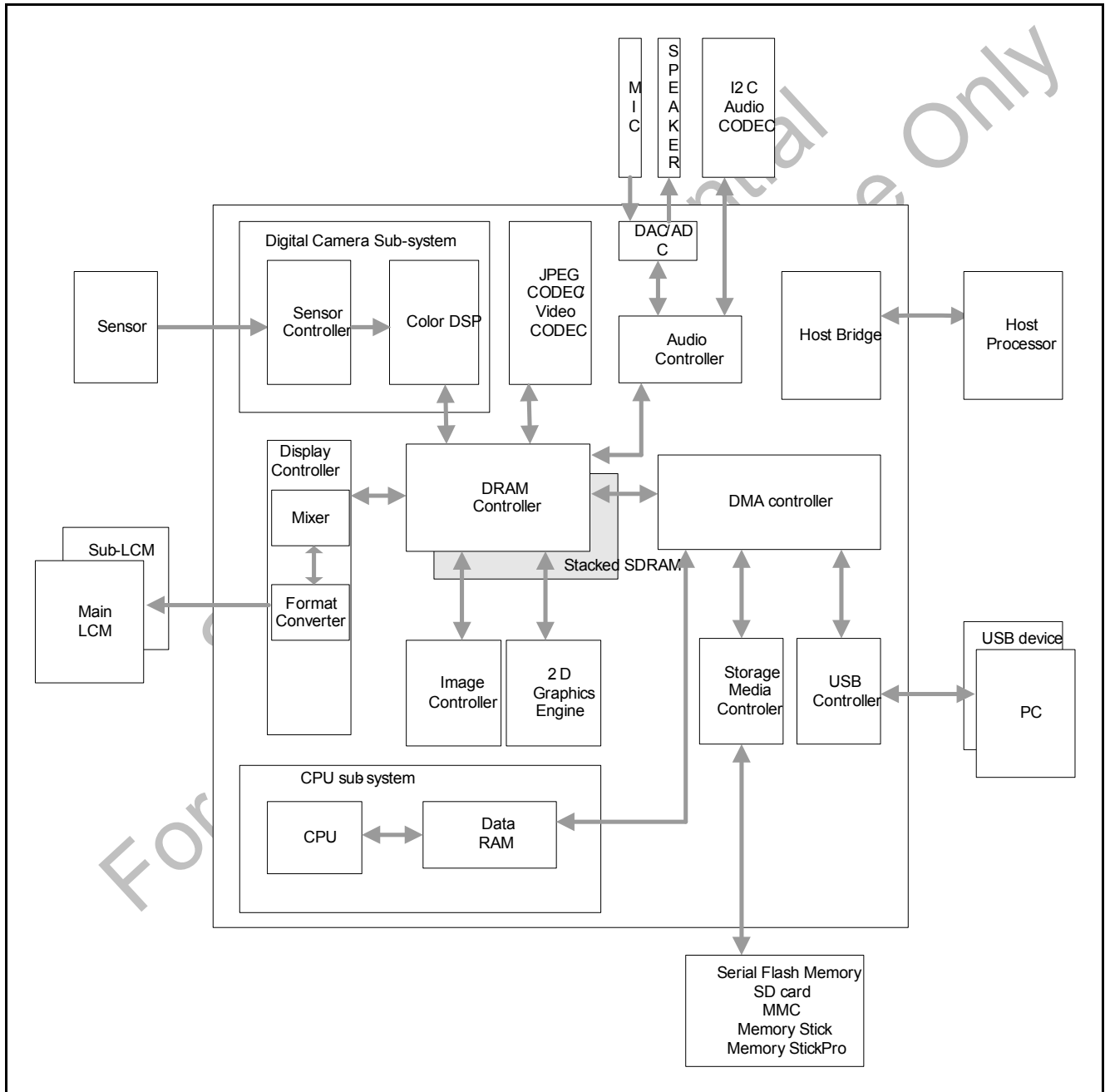


Figure 6-1-1: SPCA554A architecture

6.1.1. Camera preview

In the digital camera preview mode, the image data from the sensor goes through the image processing pipeline, image-post processing pipeline and display processing pipeline in sequence. The image processing pipeline is handled by the SPCA554A sensor controller and color DSP. The image post-processing pipeline is handled by the SPCA554A image controller, 2D

graphics engine and mixer (Display controller). The display processing pipeline is handled by the SPCA554A format converter (display controller). In the preview mode, the embedded CPU performs AE and AWB according to lighting conditions. Figure 6-1-2 shows the preview pipeline and applicable functions in each stage.

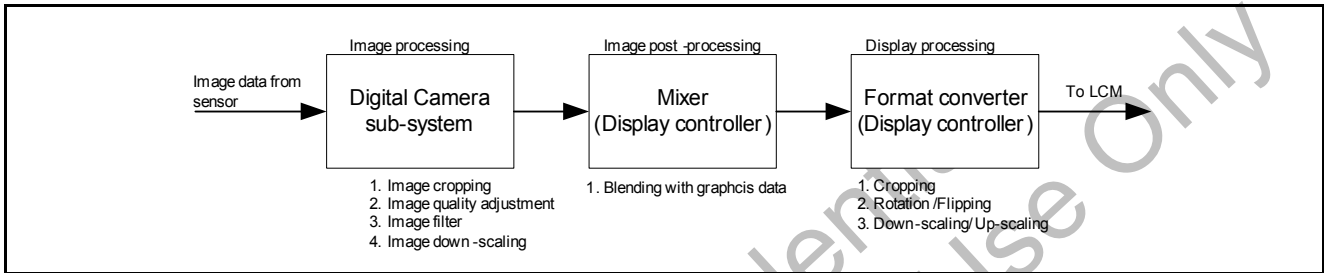


Figure 6-1-2: The function blocks of preview mode.

Digital camera sub-system:

- 1.) Image cropping is done in digital zoom, in which a smaller ROI (region of interest) is desired. Only the center part of the image is captured by the SPCA554A when the digital zoom function is activated.
- 2.) Image quality adjustment by Color DSP engine (refer to section 6.5). Also, the data is converted to YUV color space.
- 3.) With the color DSP, the SPCA554A can apply different filters to produce fancy image effects as shown in Figure 6-1-3.
- 4.) Image downscaling is necessary to fit the LCM resolution. Normally the sensor resolution is much higher than the LCM resolution. The horizontal resolution of the downscaled image must be multiples of 8-pixels.

Mixer:

The graphics data is blended (or overlaid) with the image data in the post-processing pipeline. This function is used to enable the OSD (on screen display) on the LCM. The SPCA554A supports a variety of blending modes. Reference to section 6.14.5 for more detailed information.

Format converter:

- 1.) Cropping the image after post-processing, which is usually used to resolve the digital zoom issue (Refer to section 6.14.1).
- 2.) Rotation/flipping, in which the display engine supports 90, 180, 270 rotation, horizontal and vertical flipping (Refer to section 6.14.2).
- 3.) Image Up/Down scaling, in which the display engine supports 1.0x to 4.0x scaling up, and smooth scale down for digital zoom function.

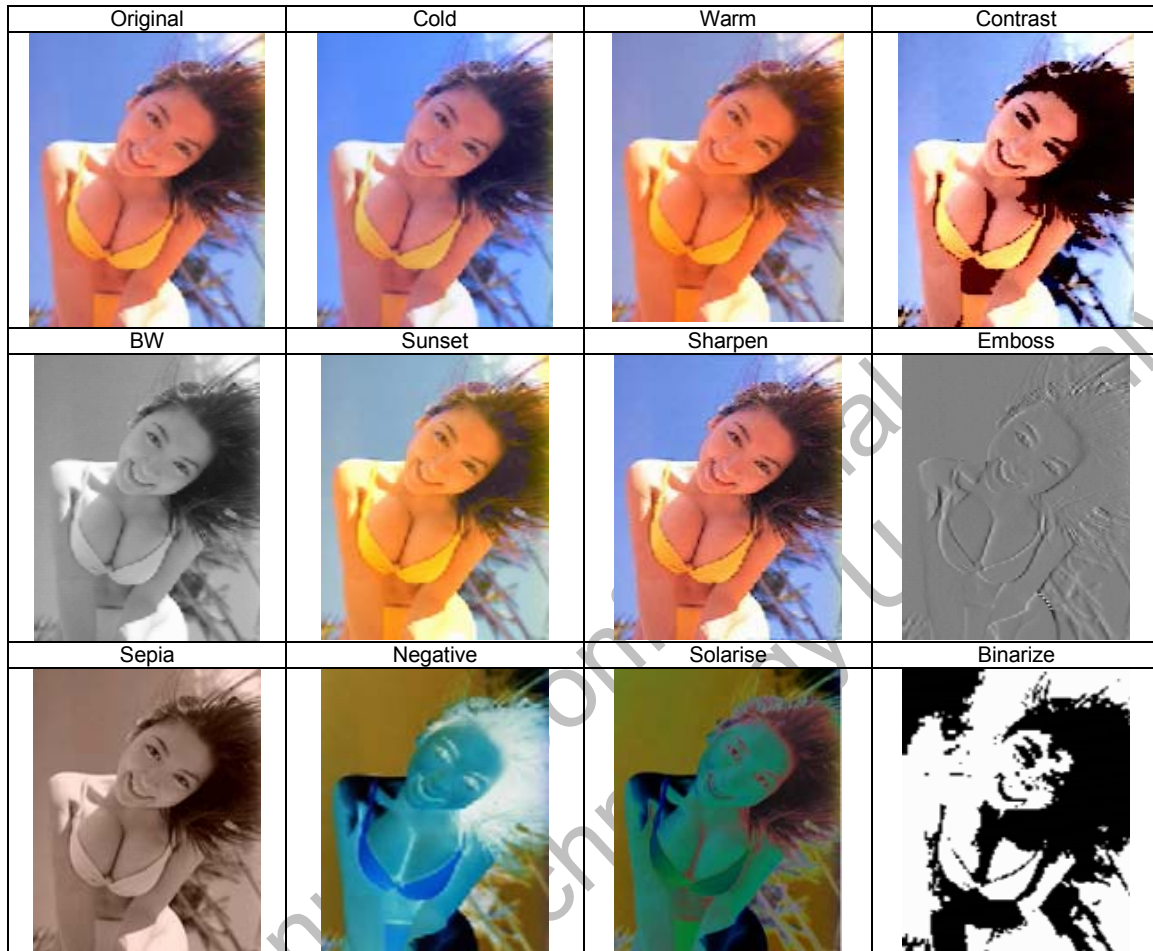


Figure 6-1-3: Image filter in the preview mode

6.1.2. Still image capture

For still image capture, the image data goes through the image processing pipeline and the image post-processing pipeline as in the preview mode. After post processing, the YUV image data is sent to the image compression pipeline, which is handled by the

SPCA554A JPEG engine. After JPEG compression, the final JPEG file is sent to the storage media. Or optionally, it can be sent to the host processor.

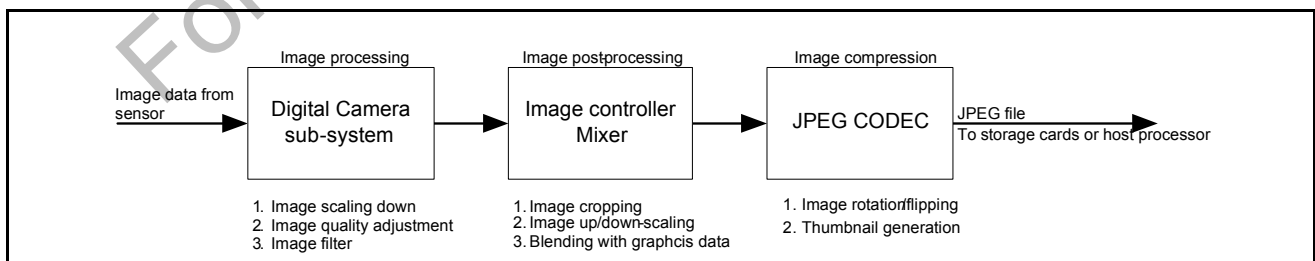


Figure 6-1-4: Function blocks of still image capture

Digital Camera sub-system:

- 1.) Image scaling down, which is usually used to control COMS sensor output image scaling down via the I²C interface.
- 2.) As in the preview mode, the image quality adjustment and filters can be applied to incoming image data.
- 3.) Image downscaling is activated when a photo-sticker is to be created or to resolve the storage card capacity issue. Otherwise, the full size image data is captured.

Image controller and Mixer:

- 1.) Cropping the ROI (region of interest), which is usually for digital zoom applications (reference to section 6.8.3).
- 2.) Image up/down scaling, which is used for scaling the source image to the destination size (reference to section 6.8.1).
- 3.) Additional filters can be applied to YUV image data as shown in figure 6-5. This is achieved by the SPCA554A internal CPU. Note that the filter in figure 6-1-3 is handled by the

image processing pipeline. They are available in both the preview and the still image capture modes. The filters in figure 6-1-5 are only available in the still image capture mode.

- 4.) Graphic data is blended with the image data, especially for photo-sticker generation.

JPEG CODEC:

- 1.) The SPCA554A's JPEG engine supports 90, 180, 270 rotation and horizontal, vertical flipping when compressing the image data. These functions are done at the same time when the image is compressed (Reference to section 6.6.2).
- 2.) The SPCA554A supports YUV420 and YUV422 JPEG compression (Reference to section 6.6.2).
- 3.) The thumbnail picture can be optionally generated after JPEG compression. The resolution of the thumbnail picture is 1/8 of the original picture.

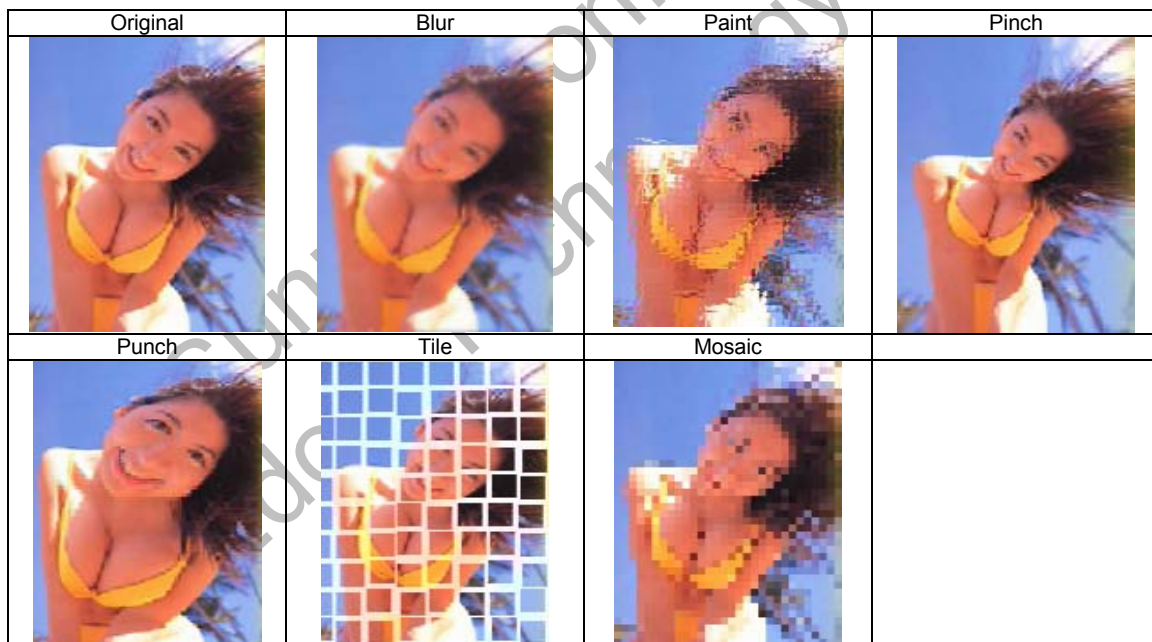


Figure 6-1-5: Additional filter in still image capture mode

6.1.3. Still image playback

The SPCA554A also has a built-in JPEG decoder to support the image playback function. To play back a compressed image, the SPCA554A receives the JFIF file from the host processor or storage card. Then the SPCA554A decodes the file, scales down the image, and sends the decompressed image to the LCM. The downscaling is needed only when the image resolution is higher than the LCM resolution. Besides being sent to the LCM, the decompressed image can be optionally sent back to the host processor as well. In the playback mode, the SPCA554A can

also perform image rotation, flipping and mirroring by LCM. Digital scaling down can be achieved by JPEG engine and image controller. The JPEG engine scales down the image first (ratio can be 7/8, 6/8, ..., 2/8, 1/8), then the image controller scales the image that was decompressed by JPEG to the destination size (usually the panel size). But when the playback image is smaller than the panel size, the image controller must scale up the image to panel size.

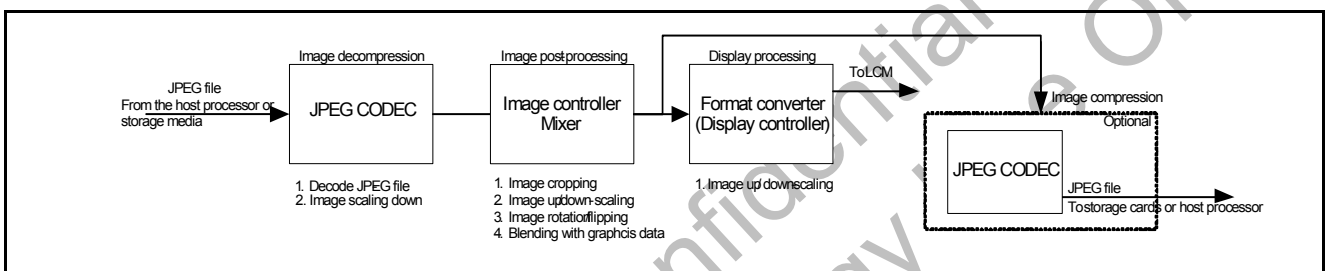


Figure 6-1-6: The function block of still image playback.

JPEG CODEC (Image decompression):

The JPEG engine decompresses the VLC stream to YUV422 data, and scales down the image to a suitable size. The scaling ratio can be 7/8, 6/8, ..., 2/8, 1/8 (refer to section 7.11.2).

Image controller and Mixer:

- 1.) Cropping the ROI (region of interest) is usually for digital zoom applications (reference to section 6.8.3).
- 2.) Image up/down scaling is used to scale image to destination size (panel size or target size define by user) (refer to section 6.8.1)
- 3.) It supports 90, 180, 270 rotation, horizontal and vertical flipping by image controller (refer to section 6.8.2).
- 4.) Supports blending (α blending, color key) for playback image and graphic data (refer to section 6.14.5)

Format converter:

It is used to scale the image to fit the panel size (refer to section 6.14.3).

JPEG CODEC (Image compression):

Compresses the image to JPEG YUV422, YUV420 format bit stream.

6.1.4. Video recoding

The SPCA554A supports MPEG-4, H.263 and Motion-JPEG video encoding algorithms. The output file of the video recording includes the 3GP format, ASF format and AVI format. Choose the appropriate encoding algorithm and output file format depending on requirements. For 352x288 resolutions, the maximum frame rate is 15 frames per second. For 176x144 resolutions, the maximum frame rate is 30 frames per second. The video recoding time depends on the capacity of the external storage media. Due to the nature of the MPEG CODEC algorithm, the video resolution must be multiples of 16 pixels in

both the horizontal and vertical directions. For example, 160x120 is not a legal video resolution for the video CODEC engine. During video recording, it is also able to record the audio input. The encoded audio bit stream will be multiplexed with the encoded video bit stream by the SPCA554A. For the audio types supported by the SPCA554A, please refer to section 2.5. The video encoder of the SPCA554A also has a built-in geometry transform function, which can perform image scaling, image flipping and image rotation during compression.

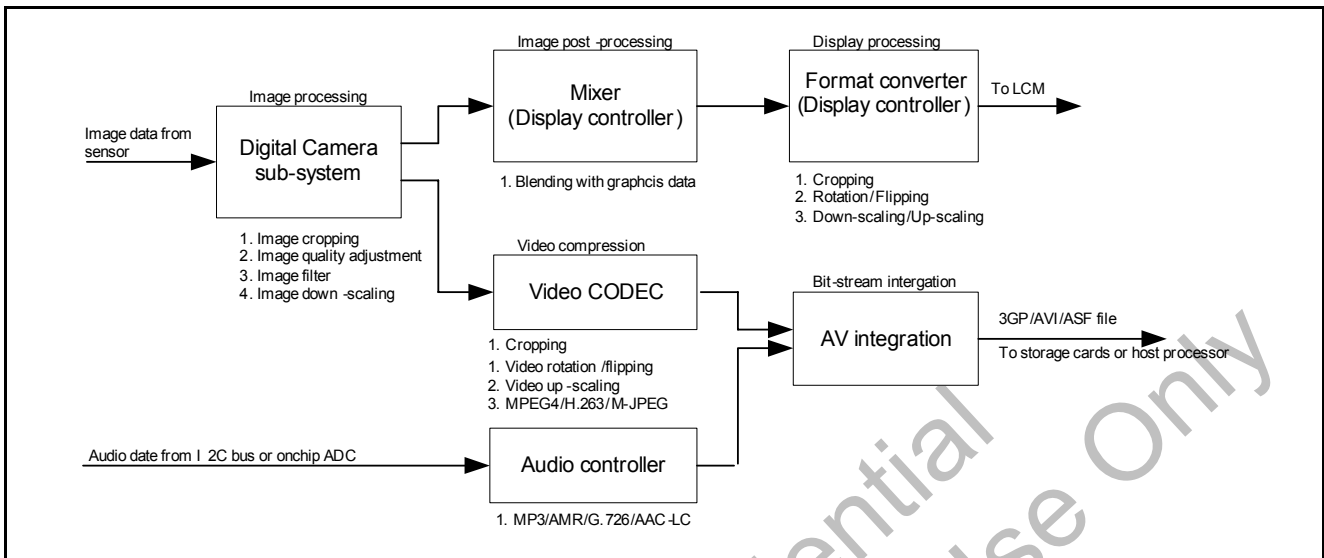


Fig 6-1-7: The function block of video recording

Digital Camera sub-system:

- 1.) Sensor image cropping (refer to section 6.4.2) is usually done for digital zoom.
- 2.) Image quality adjustment by Color DSP engine (refer to section 6.5).
- 3.) Image filter (the filter that the SPCA554A supports is the same as preview mode).
- 4.) 2D-scaling down is usually the image processing scaling down of the sensor image to recording image size. After scaling down, the horizontal size must be multiples of 16 (for the MPEG compression issue).

Mixer

SPCA554A blending (α blending, color key) where the sensor image and graphic data is the same as the preview mode.

Format converter

- 1.) Cropping the image after post-processing is usually for digital zoom applications (reference to section 6.14.1).
- 2.) Rotation/flipping. The display engine supports 90, 180, 270 rotation and horizontal, vertical flipping (reference to section 6.14.2)
- 3.) 2D scaling down/up. There are two reasons to use the

display scaling engine; one is for the LCM panel size issue, and the other is for digital zoom applications.

Video CODEC

- 1.) Cropping the image after post-processing is usually for digital zoom applications. The zoom factor must be the same as display processing (refer to section 6.6.1).
- 2.) Rotation/flipping. The MPEG engine supports 90, 180, 270 rotation and horizontal, vertical flipping. The setting must be the same as display processing (refer to section 6.6.1).
- 3.) Scaling up for digital zoom applications. It must co-operate with the cropping function (refer to section 6.6.1).
- 4.) The SPCA554A supports MPEG-1, MPEG-4, H.263 and Motion-JPEG video coding standards.

Audio controller

The SPCA554A can compress the audio raw data to MP3, AMR, AAC-LC or G.726 bit stream.

AV integration

The embedded CPU packs the video and audio bit stream to 3GP, AVI, ASF file formats.

6.1.5. Video playback

During video playback, the SPCA554A receives video files from the host CPU or storage card. All video files in 3GP format, ASF format and AVI format can be decoded by the SPCA554A. The SPCA554A internal CPU is in charge of file format parsing. The

parsed video bit stream is sent to the MPEG decoder for decoding. After decoding, the image is sent to the LCM through the LCM interface. The scaling up/down, rotation and flipping function are also supported by the LCM.

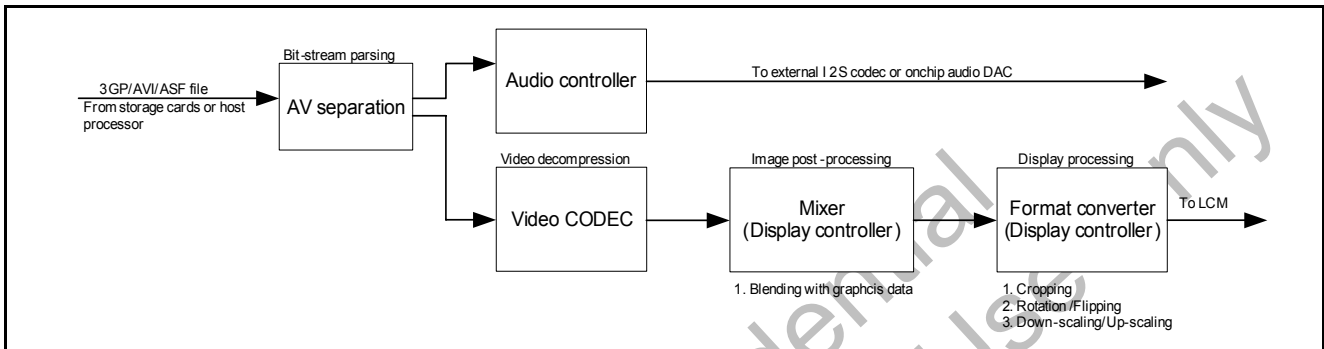


Figure 6-1-8: The function block of video playback

■ AV separation

The embedded CPU parses 3GP, AVI or ASF file, and separates the video and audio bit stream, then sends the audio bit stream to audio decomposition, and video to video decomposition.

■ Audio controller

The embedded CPU decompresses the audio bit stream (MP3, AMR, AAC-LC, G.726) to audio raw data, and sends to external I2S codec or on chip audio DAC.

■ Video CODEC

The MPEG engine can decompress the video stream (MPEG4, H.263) to YUV422, YUV420 data, then send the video data to image post-processing.

■ Mixer

The SPCA554A can blend (α blending, color key) the sensor image and graphic data for user applications, and send the blending data to display processing.

■ Format converter

1.) Cropping the image after post-processing is usually for digital

zoom applications (reference to section 6.14.1).

2.) Rotation/flipping. The display engine supports 90, 180, 270 rotation, horizontal and vertical flipping (refer to section 6.14.2).

3.) 2D scaling down/up. There are two reasons to use the display scaling engine; one is for the LCM panel size issue, and the other is for digital zoom applications.

6.1.6. Screen saver

The SPCA554A is able to playback a short sequence of images automatically without host intervention. This is often used to implement the screen-saver function when the host processor is in the stand-by mode. To do this, the compressed bit stream of the image sequence is downloaded to the SPCA554A internal memory, then the SPCA554A will be able to playback them one by one according to the specified frame rate. The image resolution can be up to 352x288, and the SPCA554A internal memory is able to accommodate at least 10 images for the screen saver function. The host sets the frame rate by a host command before starting the screen saver function. The SPCA554A internal CPU controls the play back time.

6.1.7. Video conference

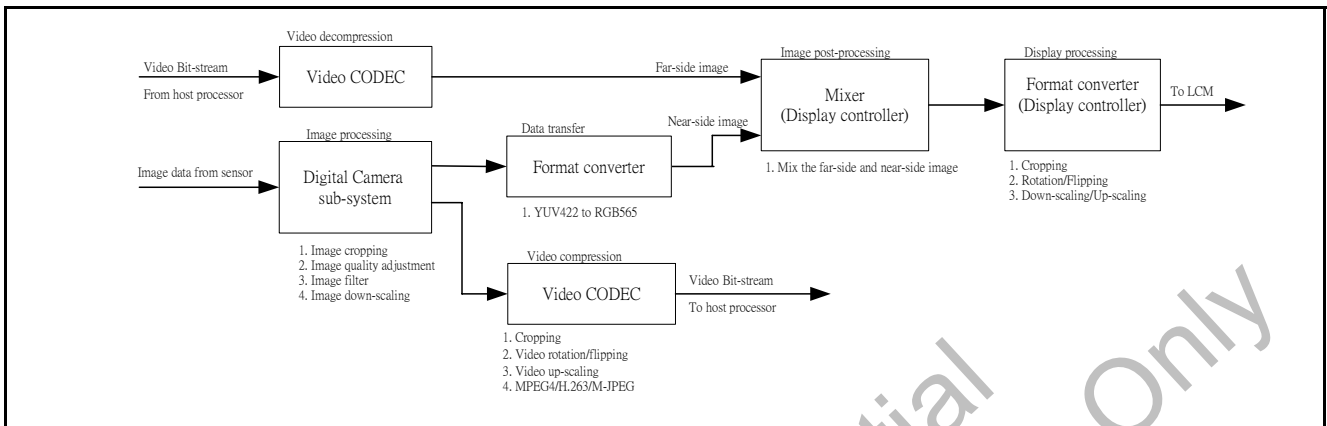


Figure 6-1-10: The function block of video playback

Digital Camera sub-system:

- 5.) Sensor image cropping (refer to section 6.4.2) is usually done for digital zoom.
- 6.) Image quality adjustment by Color DSP engine (refer to section 6.5)
- 7.) Image filter (the filter that the SPCA554A supports is the same as preview mode).
- 8.) 2D-scaling down is usually the image processing scaling down of the sensor image to recoding image size. After scaling down, the horizontal and vertical size must be multiples of 16 (for the MPEG compression issue).

Video CODEC (Video decompression)

The MPEG engine can decompress the video stream (MPEG4, H.263) to YUV422, YUV420 data, and then send the video data to the format converter.

Video CODEC (Video compression)

- 5.) Cropping the image after post-processing is usually for digital zoom applications. The zoom factor must be the same as display processing (refer to section 6.6.1).
- 6.) Rotation/flipping. The MPEG engine supports 90, 180, 270 rotations and horizontal and vertical flipping. The setting

must be the same as display processing (refer to section 6.6.1).

- 7.) Scaling up for digital zoom applications. It must co-operate with the cropping function (reference to section 6.6.1).
- 8.) The SPCA554A supports MPEG-1, MPEG-4, H.263 and Motion-JPEG video coding standards.

Format converter (Data transfer)

The format converter transfers the YUV422 image data to RGB 565 graphic image data.

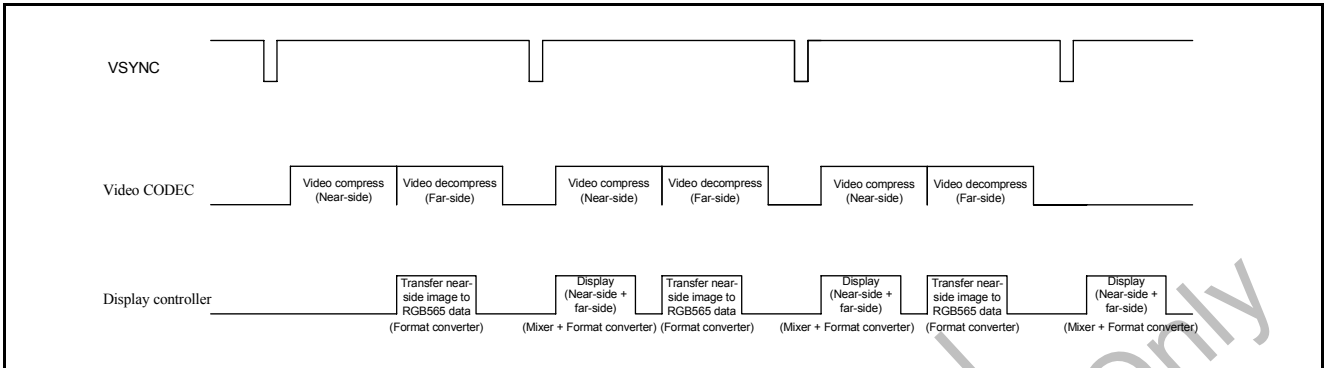
Mixer

The SPCA554A mixes the far-side and near-side images together.

Format converter (Display processing)

- 4.) Cropping the image after post-processing is usually for digital zoom applications (refer to section 6.14.1).
- 5.) Rotation/flipping. The display engine supports 90, 180, 270 rotation, horizontal and vertical flipping (refer to section 6.14.2).
- 6.) 2D scaling down/up. There are two reasons to use the display-scaling engine; one is for the LCM panel size issue, and the other is for digital zoom applications.

The following diagram shows the far-side and near-side images of video conferencing and processing in time domain:



Note: The max Framerate is 7.5 fps for 352x288 image size, 15 fps for 176x144

6.2. Host Bridge

The host bridge of SPCA554A provides an interface for the host processor to control and transfer data to/from the SPCA554A. No matter which type of host interface protocol is selected, the host bridge always accepts 7-bit addresses in an indirect access manner. For example, in the 8-bit parallel bus protocol, the first byte is transferred from the host processor to the SPCA554A with A0 pin driven low and it is interpreted as the address byte. Seven bits out of the address byte are used by SPCA554A to address its internal host bridge registers.

With seven bit addresses, 128 registers can be addressed. The SPCA554A partitions the addressing space into the host bridge control register section and the communication section. The host bridge section provides control registers, allowing the host bridge to handshake with the SPCA554A. The communication section is reserved for data communication between the host processor and the SPCA554A. The following figure shows the addressing space partition:

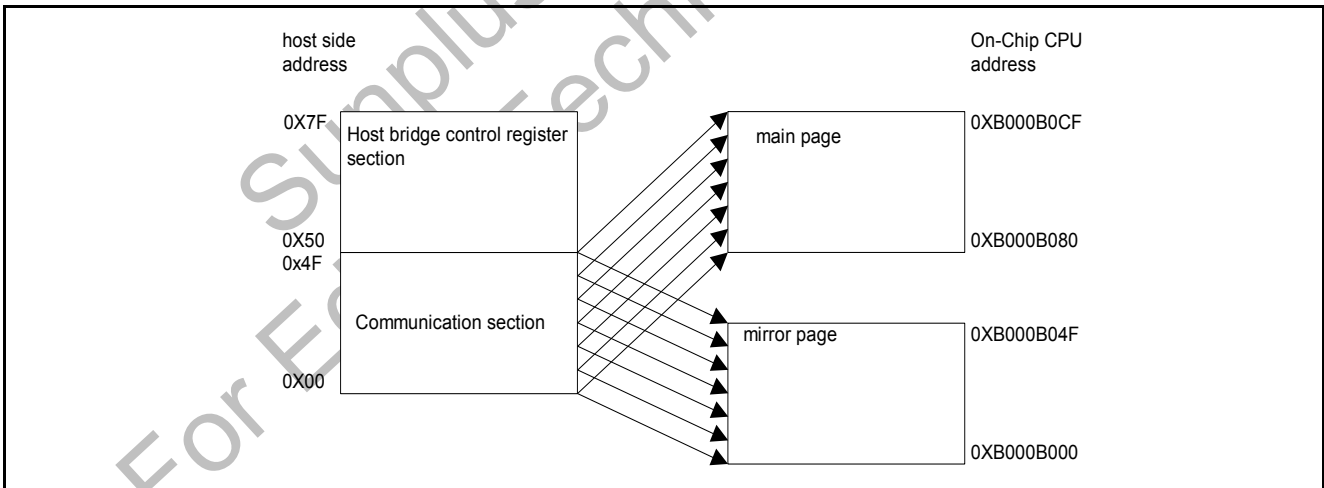


Figure 6-2-1: Addressing space mapping

The communication section has two pages, the main-page and the mirror-page. It is selected by page switching bit, Reg0X7F[0]. If the main page is selected, the host processor can write data to the communication section and the data is referenced by SPCA554A on-chip CPU later. If the mirror-page is selected, the host processor can read data from the communication section, which contains the data returned from SPCA554A.

In a real application, the host processor writes vendor commands to the main page. The commands are then processed by the SPCA554A on-chip CPU. If there is data to be returned, the SPCA554A on-chip CPU writes data to the mirror page and then the host processor reads the returned data from the mirror page. The host processor can read/write data from/to the main page, but can only read data from the mirror page. The SPCA554A's on-chip CPU can read/write data from/to the mirror page, but it can only read data from the main page.

Address	Register "CPUPsel" = 0	Register "CPUPsel" = 1
0x00 – 0x4F	Main page	Mirror page
0x50 – 0x7F	Other host bridge control register	

Table 6-2-1: Three groups of Host bridge registers

6.2.1. Host bridge control registers

This section describes the control registers in the host bridge. Major control flows, like internal register access, vendor command communication and internal buffer access are described in further details in sections 6.2.3 ~ 6.2.5.

Address	Name	Bit	Type	Function
0X50	HIDWData	7:0	R/W	Data write port for the host processor to write data to SPCA554A internal registers. SPCA554A allows the host processor to access its internal registers. HIDWDATA is the data write port used by the host processor to write data to SPCA554A internal registers. The registers include all the control and data registers of internal modules, like ColorDSP, MJPEG,... etc. Registers 0X50 ~ 0x54 are used for internal register access. Internal register access is prohibited when internal CPU is in operation. It is necessary to reset the internal CPU (writing 1 to HCPURST register) before accessing the internal registers.

Address	Name	Bit	Type	Function
0X51	HIDDO	7:0	R	Data read port for the host processor to read data from the SPCA554A internal registers

Address	Name	Bit	Type	Function
0X52	HIDStrAddrL	7:0	R/W	Low byte address to be used in the internal register access. Internal registers are addressed by 16-bit address.

Address	Name	Bit	Type	Function
0X53	HIDSTRAddrH	7:0	R/W	High byte address to be used in the internal register access. Bit[7:4] are used to address individual functional block. Each internal module is allocated with 4K-bytes of address space.

Address	Size	Description
0x0XXX	4KB	Global control registers
0x1XXX	4KB	CPU control registers
0x2XXX	4KB	Color DSP control registers
0x3XXX	4KB	DMA control registers
0x4XXX	4KB	Storage media controller registers
0x5XXX	4KB	USB controller registers
0x6XXX	4KB	Audio controller registers
0x7XXX	4KB	Memory Controller Registers
0x8XXX	4KB	MJPEGCODEC registers
0x9XXX	4KB	Sensor controller Registers
0xAxxx	4KB	Display controller Registers
0xBxxx	4KB	Host Bridge Registers
0xCxxx	4KB	2D Graphic Engine Registers

Table 6-2-2: Addressing partition

Address	Name	Bit	Type	Function
0X54	HIDAutoInc	0	R/W	Automatic address increment mode used in the internal register address. If this bit is set, the host processor can read/write HIDD0/ HIDWDATA continuously with automatic register address increment. Writing 1 turns on this feature. Writing 0 turns off this feature.

Address	Name	Bit	Type	Function
0X56	HPIOWDataL	7:0	R	Data read-back check port used in Internal buffer access. Low byte. The SPCA554A provides five internal buffer access to the host processor via the host interface. The host processor can burst-access these internal buffers. The accessible buffers include stacked low power SDRAM, data RAM in the CPU sub-system, display data buffer after data mixing (graphic image and camera image mixing) and the 2D engine command buffer. All the other data buffers are accessed with a 16-bit data width. If the 8-bit host interface is selected, low byte of the data is sent first, followed by high byte. HPIOWDATA provides a read-back check port when the host writes data to these data buffers.

Address	Name	Bit	Type	Function
0X57	HPIOWDataH	7:0	R	Data read-back check port used in Internal buffer access. High byte. This register is not used when 8-bit host interface is selected.

Address	Name	Bit	Type	Function
0X5A	RSVCtrl	7:0	R/W	Reserved control register.

Address	Name	Bit	Type	Function
0X5C	SDRAMRdy	0	R	SDRAM data is ready for read. The registers are used to check if the buffer data is ready to be read by the host processor. When accessing the stacked SDRAM and the Display data by buffer access method, these status bits must be checked before reading data from DATAPORT (register 0X70).
	DispHRdy	1	R	Display buffer data is ready for read.

Address	Name	Bit	Type	Function
0X5D	HCPURst	0	R/W	On-chip CPU software reset. Writing 1 to reset the on-chip CPU. The default value of this register is 0 when selecting host as the boot source. And its default value is 1 if internal mask ROM is selected as the boot source. It means that when booting from the host, the on-chip CPU is in the reset state after power on. After program code downloading, the host processor must write 0 to HCPURST to activate the on-chip CPU.
	HCPUSPEND	1	R/W	Reserved bit, must be kept 0.

Address	Name	Bit	Type	Function
0X60	hvcint	1	R/W	After the host processor writes the vendor command to the main page, it writes 1 to this bit to inform the on-chip CPU that the vendor command is issued. Writing 1 to this bit generates an interrupt to the on-chip CPU, then the on-chip CPU is to parse the vendor command in the main page. This bit is not used in the final firmware implementation. Do not program this bit.

Address	Name	Bit	Type	Function
				SPCA554A resume interrupt enable. Once SPCA554A is put into suspend state, it can only be awakened by the interrupt induced by the CS/ pin state change. CS/ pin the chip select signal of the host interface.
0X61	HBIntEn	0	R/W	Rising edge of CS/ signal generates a resume interrupt.
		1	R/W	Falling edge of CS/ signal generates a resume interrupt. To be able to wake up SPCA554A after suspend, either bit 0 or bit one must be set.

Address	Name	Bit	Type	Function
0X62	HTGPIOSEL0	7:0	R/W	GPIO pins control arbitration. SPCA554A has accompanied GPIO function for each interface pins. When these pins are not used as their default function,

they can be used as GPIO pins. The control of the GPIO function can be from the internal function blocks or directly from the host processor. Register 0X62~0X6A are used for this GPIO function. Write 1 to enable host control of the GPIO pins. Writing 0 to keep the control to SPCA554A internal modules.

- 0X63 HTGPIOSEL1 7:0 R/W Writing 1 to enable host control of the GPIO pins. Write 0 to keep the control to SPCA554A internal modules.
- 0X64 HTGPIOSEL2 1:0 R/W Writing 1 to enable host control of the GPIO pins. Write 0 to keep the control to SPCA554A internal modules.

Registers	Mapped GPIO Pin
HTGPIOSEL0 [0]	Cpugpio [0]
HTGPIOSEL0 [1]	cpugpio [1]
HTGPIOSEL0 [2]	cpugpio [2]
HTGPIOSEL0 [3]	cpugpio [3]
HTGPIOSEL0 [4]	cpugpio [4]
HTGPIOSEL0 [5]	cpugpio [5]
HTGPIOSEL0 [6]	cpugpio [6]
HTGPIOSEL0 [7]	cpugpio [7]
HTGPIOSEL1 [0]	cpugpio [8]
HTGPIOSEL1 [1]	cpugpio [9]
HTGPIOSEL1 [2]	suspend
HTGPIOSEL1 [3]	trap
HTGPIOSEL1 [4]	lcmgpio [20]
HTGPIOSEL1 [5]	lcmgpio [21]
HTGPIOSEL1 [6]	lcmgpio [22]
HTGPIOSEL1 [7]	lcmgpio [23]
HTGPIOSEL2 [0]	lcmgpio [24]
HTGPIOSEL2 [1]	lcmgpio [25]

Table 6-2-3: Host GPIO pins mapping

Address	Name	Bit	Type	Function
0X65	HTGPIOOE0	7:0	R/W	GPIO pins output enable control. This register is used to turn on the output enable and make the GPIO pins as output pins. It is effective only when the GPIO pins are selected to be controlled by the host processor. Write 1 to turn the GPIO pins to output mode. Write 0 to turn the GPIO pins to input mode.
0X66	HTGPIOOE1	7:0	R/W	Write 1 to turn the GPIO pins to output mode. Write 0 to turn the GPIO pins to input mode.
0X67	HTGPIOOE2	1:0	R/W	Writing 1 to turn the GPIO pins to output mode. Write 0 to turn the GPIO pins to input mode.

Address	Name	Bit	Type	Function
0X68	HTGPIOOV0	7:0	R/W	Output value of the GPIO pins. This register is effective only when the GPIO is controlled by the host processor and is programmed to be output pins.
0X69	HTGPIOOV1	7:0	R/W	Output value of the GPIO pins.
0X6A	HTGPIOOV2	1:0	R/W	Output value of the GPIO pins.

Address	Name	Bit	Type	Function
0X70	DataPort	7:0	R/W	The data port for the host to access the internal buffer of SPCA554A. Data written to DATAPORT is re-directed to the internal buffer, like stacked SDRAM, data memory of the CPU sub-system, etc.

Address	Name	Bit	Type	Function
0X71	PortSel	2:0	R/W	Data buffer selection. This register is used to select which data buffer is to be accessed. 0: Data memory of the CPU sub-system 1: Stacked low power SDRAM 2: 2D engine command buffer 3: Display data after data mixing. 4: Huffman table or Q-table of JPEG engine. This path is not used in the final firmware implementation.

Address	Name	Bit	Type	Function
0X71	CEndinCtrl	4	R/W	Data bytes swapping control. It determines whether to swap the low byte and the high byte when host accesses the data buffer. 0: Keep the byte order in internal buffer access. 1: Swap the high/low byte when host access internal buffer via the DATAPORT.

Address	Name	Bit	Type	Function
0X71	BEndianCtrl	5	R/W	Data bytes swapping control. It determines whether to swap the high/low bytes when host accesses the LCM in the by-pass mode. 0: Keep the byte order in by-pass mode. 1: Swap high/low byte when host accesses the LCM in the by-pass mode.

Address	Name	Bit	Type	Function
0X72	H16Bit	0	R/W	This register is used to set the host interface data width and the LCM control pulse timing. The re-shape timing diagram for read pulse is shown in figure 6-10. 0: 8-bit data bus. 1: 16-bit data bus.
	BPOrgShpEn	1	R/W	Keeps the access timing as original as from the host processor 0: Re-shape the LCM access timing in by-pass mode. 1: Keep the timing original.
	PoeDlySel	3:2	R/W	CS/ signal of the LCM interface timing setting 0: Delay active period by 10 ns and inactive period by 10ns. 1: Delay active period by 10 ns and inactive period by 5ns.. 2: Delay active period by 10 ns and inactive period by 0ns. 3: Delay active period by 0 ns and inactive period by 0ns.
	PRDDlysel	5:4	R/W	Read pulse delay selection 0: Delay 15 ns. 1: Delay 10 ns. 2: Delay 5 ns. 3: Unchanged.
	PRDShtSel	7:6	R/W	Read pulse front porch re-shape 0: 15 ns short. 1: 10 ns short. 2: 5 ns short. 3: Unchanged.

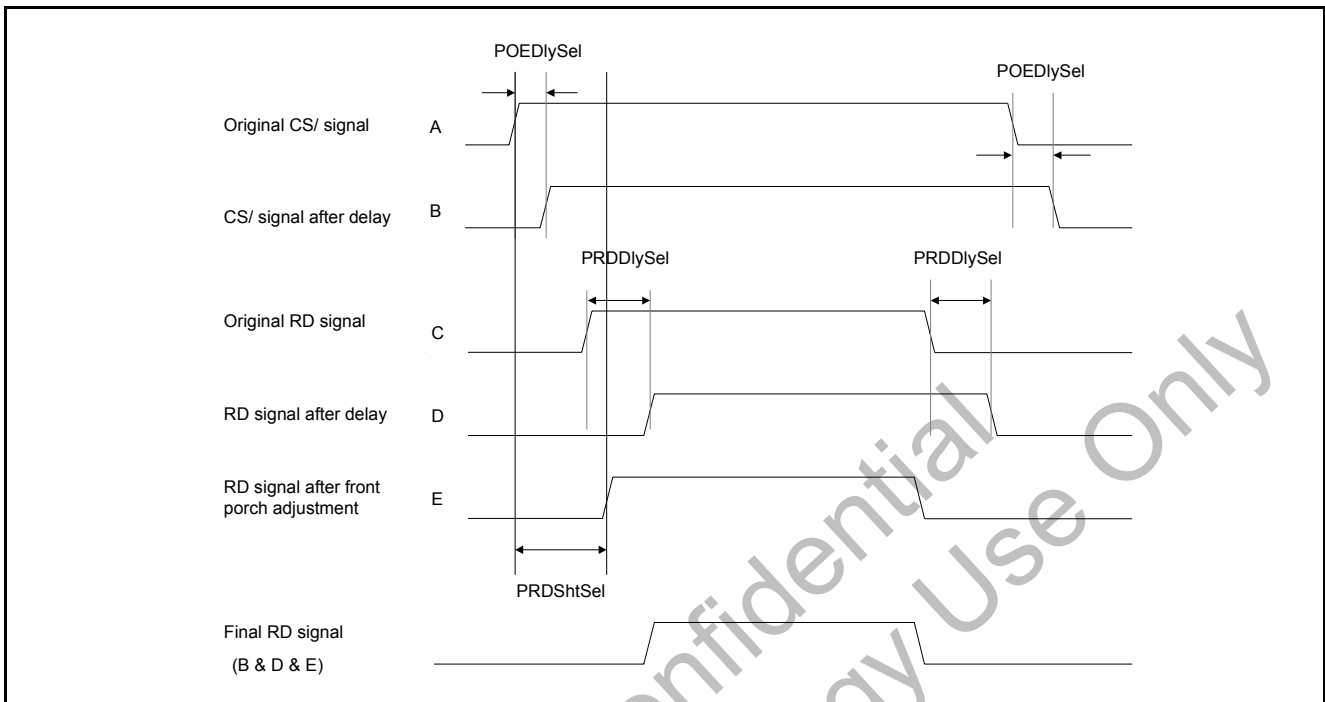


Figure 6-2-2: re-shape for read pulse

Address	Name	Bit	Type	Function
0X73	HPWrStart	0	R/W	This register is used in internal buffer access. Write 1 to this bit before an internal buffer write access.
	HPRdStart	1	R/W	Write 1 to this bit before an internal buffer read access.
	HPEndSel	2	R/W	Write 1 to this bit to terminate an internal buffer access.

Address	Name	Bit	Type	Function
0X74	PortSumClr	0	R/W	For each internal buffer access, the SPCA554A automatically generates the check sum and records the byte-count of the access. Writing 1 to clear the check sum register. Writing 0 to resume to normal check sum function.
	PortSumREn	1	R/W	0: Enable the check sum function in internal buffer read access. 1: Disable the check sum function in internal buffer read access.
	PortSumWn	2	R/W	0: Enable the check sum function in internal buffer write access. 1: Disable the check sum function in internal buffer write access.
	PortCntClr	4	R/W	Writing 1 to clear the byte counter of the internal buffer access. Writing 0 to resume the byte-counting function.
	PortCntREn	5	R/W	0: Enable the byte-counting function in internal buffer read access. 1: Disable the byte-counting function in internal buffer read access.
	PortCntWEn	6	R/W	0: Enable the byte-counting function in internal buffer write access. 1: Disable the byte-counting function in internal buffer write access.

Address	Name	Bit	Type	Function
				The check sum in an internal buffer access is 16-bits long. It can be read back by the host processor for integrity check. This register is cleared by PortSumClr, HPWrStart or HPRdStart.
0X75	PortSumL	7:0	R	Low byte of the check sum.
0X76	PortSumH	7:0	R	High byte of the check sum.

Address	Name	Bit	Type	Function
				The byte-count in an internal buffer access is 24-bit long, meaning a 16Mbyte data can be transferred in a single buffer access. The byte-count can be read back by the host processor for integrity check. This register is cleared by PortCntClr, HPWrStart or HPRdStart.
0X77	PortCntL	7:0	R	Low byte of the byte-count.
0X78	PortCntM	7:0	R	Middle byte of the byte-count.
0X78	PortCnH	7:0	R	High byte of the byte-count.

Address	Name	Bit	Type	Function
0X7A	PGMStMark	7:0	R/W	Marker to activate the on-chip CPU. 0x5A: start CPU execution. Others: Keep the on-chip CPU in the reset state. The power on value of this register is 0X5A. SPCA554A reference firmware does not use this register to control the CPU. Instead, HCPURst (Reg0X5D[0]) is used. It is recommended to keep this register as 0X5A.

Address	Name	Bit	Type	Function
0X7B	ChipRdy	0	R/W	It indicates SPCA554 is ready to accept another vendor command. Whenever SPCA554 finishes a corresponding task of a vendor command, it writes 1 to this bit. The host processor has to wait for this bit to be set before issuing the next vendor command. Before the host issues the next vendor command, the host processor must clear this bit first. This bit serves as a flag for the host processor to communicate with the SPCA554A with a vendor command. This bit is not used in the final firmware implementation.
	ChipErr	3	R	This bit is set by the SPCA554A on-chip CPU when the execution of a vendor command encounters an internal error. The host may check this bit to see if an error occurs with the execution of a vendor command it issues. This bit is not used in the final firmware implementation.
	CmdRdy	4	R/W	It indicates SPCA554 is ready to accept another vendor command. Whenever the SPCA554A finishes a corresponding task of a vendor command, it writes 1 to this bit. The host processor has to wait for this bit to be set before issuing the next vendor command. Before the host issues the next vendor command, the host processor must clear this bit first. This bit serves as a flag for the host processor to communicate with the SPCA554A with a vendor command. This bit IS USED in final firmware implementation.
	CmdErr	5	R	This bit is set by the SPCA554A on-chip CPU when the execution of a vendor command encounters an internal error. The host may check this bit to see if an error occurs with the execution of a vendor command it issues. This bit is not used in final firmware implementation.
	HOPrdySet	6	W	After the host processor writes the vendor command to the main page, it writes 1 to this bit to inform the on-chip CPU that the vendor command is issued. Writing 1 to this bit generates an interrupt to the on-chip CPU, then the on-chip CPU is to parse the vendor command in the main page. This bit is not used in final firmware implementation.
	HCmdRdySet	7	W	After the host processor writes the vendor command to the main page, it writes 1 to this bit to inform the on-chip CPU that the vendor command is issued. Writing 1 to this bit generates an interrupt to the on-chip CPU, then the on-chip CPU is parses the vendor command in the main page. This bit IS USED in final firmware implementation.

Address	Name	Bit	Type	Function
0X7C	ChipStsL	7:0	R/W	Low byte of SPCA554A chip status. This register is set by the on-chip CPU and referenced by the host CPU. It is used to report the current SPCA554A status. ChipSts is reserved for future extension and is not used in current firmware implementation.

0X7D ChipStsH 7:0 R/W High byte of SPCA554A chip status.

Address	Name	Bit	Type	Function
0X7E	VCMode	0	R/W	Internal register access mode selection. 0: Access SPCA554A's internal registers by indirect addressing mode. 1: Access SPCA554A's internal registers by vendor command.

Address	Name	Bit	Type	Function
0X7E	GUIVCTrl	7:4	R/W	Reserved for 2D engine performance check.

Address	Name	Bit	Type	Function
0X7F	CPUSel	5	R/W	Page selection of the communication section of the host interface addressing space. 0: Select the main page, for writing vendor commands to the communication section. 1: Select the mirror page, for reading data from the mirror page.

HQTSEL	6	R/W	JPEG buffer access port selection. This register is used together with PoerSel (Reg 0X71[2:0]) to access JPEG's Huffman table or Q-table in an internal buffer access. The Huffman table is 512 bytes and the Q-table is 128 bytes. This bit is not used in the final firmware implementation 0: Select Huffman table in JPEG buffer access 1: Select Q-table in JPEG buffer access
--------	---	-----	---

6.2.2. Main Page Register and Mirror Page Register

This section describes the communication section of the host bridge addressing space. The host processor can read/write data from/to the main page, but can only read data from the mirror page. The SPCA554A on-chip CPU can read/write data from/to the mirror page, but it can only read data from the main page.

Main Page Register

Address	Name	Bit	Type	Function
0X00	VCCmdIDLh	7:0	R/W	Vendor command ID, low bytes

Address	Name	Bit	Type	Function
0X01	VCCmdIDLh	7:0	R/W	Vendor command ID, high bytes

Address	Name	Bit	Type	Function
0X02	VCDDataPLh	7:0	R/W	Vendor command parameter size, low byte. It indicates number of parameter bytes.

Address	Name	Bit	Type	Function
0X03	VCCmdPLh	7:0	R/W	Vendor command parameter size, high byte.

Address	Name	Bit	Type	Function
0X04~0X3F	VCCmdParm	7:0	R/W	Vendor command parameters

Address	Name	Bit	Type	Function
0X40~0x4F	Resereved	7:0	R/W	Reserved

Mirror Page Register

Address	Name	Bit	Type	Function
---------	------	-----	------	----------

0X00~0X4F DATA 7:0 R SPCA554A returns data to the host processor in response to vendor commands. At most, 80 bytes can be returned to the host processor.

6.2.3. Vendor command communication flow

Two Registers, “HostCmdRdy” and “CmdRdy”, are used as hand-shake signals for vendor commands between the host processor and the SPCA554A. Register “HostCmdRdy” is set by the host processor and cleared by the SPCA554A. Register “CmdRdy” is set by the SPCA554A and cleared by the host processor.

A host command contains at most 80-bytes of content. The host processor writes a vendor command to the main page registers (0x00 to 0x4F). After SPCA554A finishes processing the vendor command, the host processor can read the response data from the mirror page registers (0x00 to 0x4F), if there is any. As the main page and the mirror page share the same addressing space, the host processor has to program the page selection bit (Reg0X7F[0]) to be able to access the desired page.

The following figure shows the vendor command flow. At first, the host bridge checks to see if the SPCA554A is ready to receive the next vendor command. If the SPCA554A is ready, then the host processor sets register CmdRdy to 0 and starts writing vendor commands and data to the main page registers. Next, it writes register HostCmdRdy to 1 and triggers a vendor command interrupt to the SPCA554A on-chip CPU. The on-chip CPU, then, sets the chip status to the vendor command mode, decodes vendor commands and processes them. During processing, the CPU will read/write other internal registers and forward the processed data to the Mirror CPU Page registers from those internal registers. Then, the CPU reset register “cmdrdy” is set to one and register “Hostcmdrdy” is set to zero. Once the host bridge detects the register, “cmdrdy” becomes one. It will start to read the response vendor data from the Mirror CPU Page registers to finish the whole flow. Figure 6-2-3 shows vendor command flow.

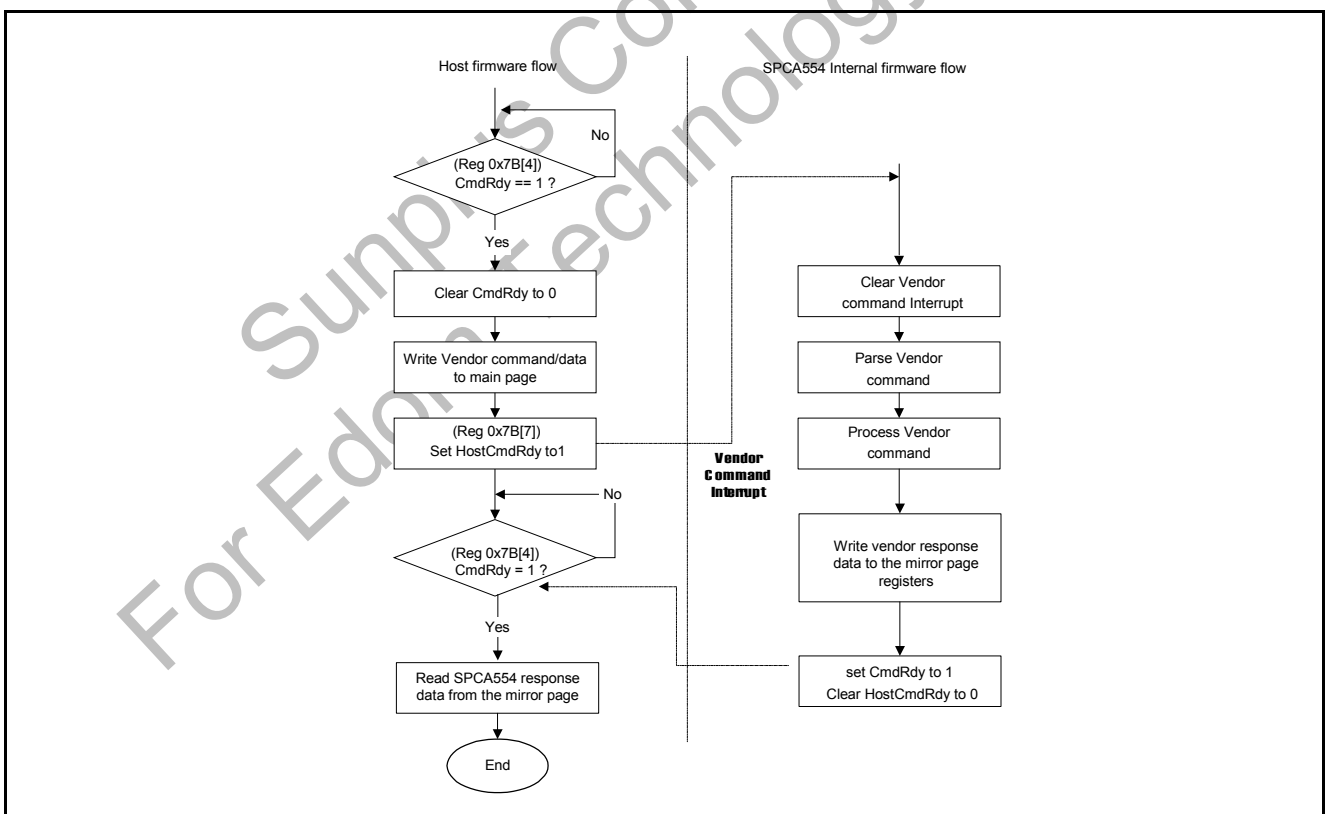


Figure 6-2-3: Vendor command operation flow

6.2.4. Internal buffer access

The host processor can access the SPCA554A internal buffers through DataPort (Reg0X70). Figure 6-2-4 shows the available paths for the host processor to access SPCA554A internal buffers.

- Path 1: Access stacked low power SDRAM
- Path 2: Access mixed data (camera image mixed with graphics image)
- Path 3: Access 2D Graphics Engine command buffer
- Path 4: Access DATA RAM in the CPU sub-system

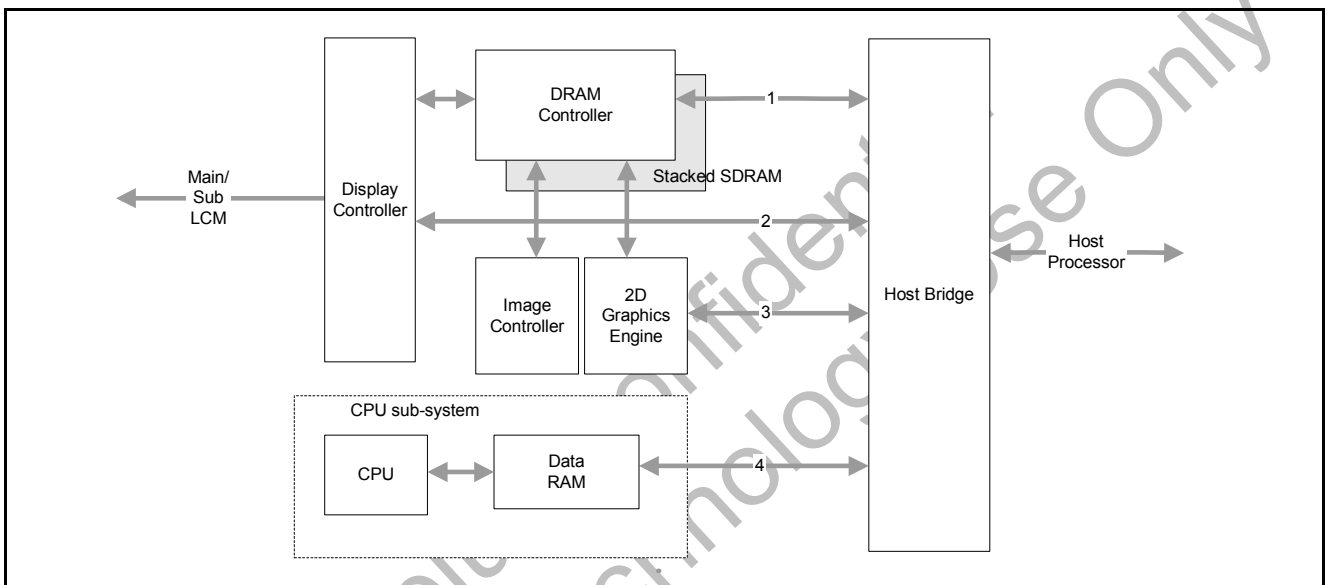


Figure 6-2-4: SPCA554A block diagram

When the host processor accesses data from the SPCA554A internal buffer, it must set the starting address of the data first. For example, if the host processor intends to read the data from the stacked SDRAM, the address register in SDRAM must be initialized first, so that the DRAM controller can decide where to start ready data from. Also the host processor has to select the corresponding port of the buffer. Port selection is done by programming the PortSel register (Reg0X71[2:0]). Then the host processor writes 1 to HPWrStart or HPRdstart register to trigger buffer access (Reg0x&3[1:0]). After the buffer access cycle is

triggered, the host processor can start to burst read/write DataPort(Reg0X70). One exception is for stacked SDRAM and display data. The host processor has to poll the SDRAMRdy and DispHRdy register (Reg 0X5C[1:0]) before burst reading corresponding buffers. The data in both buffers have a read latency time and require the host processor to make sure the data is ready for reading. To stop the buffer access cycle, the host processor has to set the HPEndSel bit to 1 (Reg 0X73[2]). Figure 6-2-5 shows host processor flow for accessing SPCA554A.

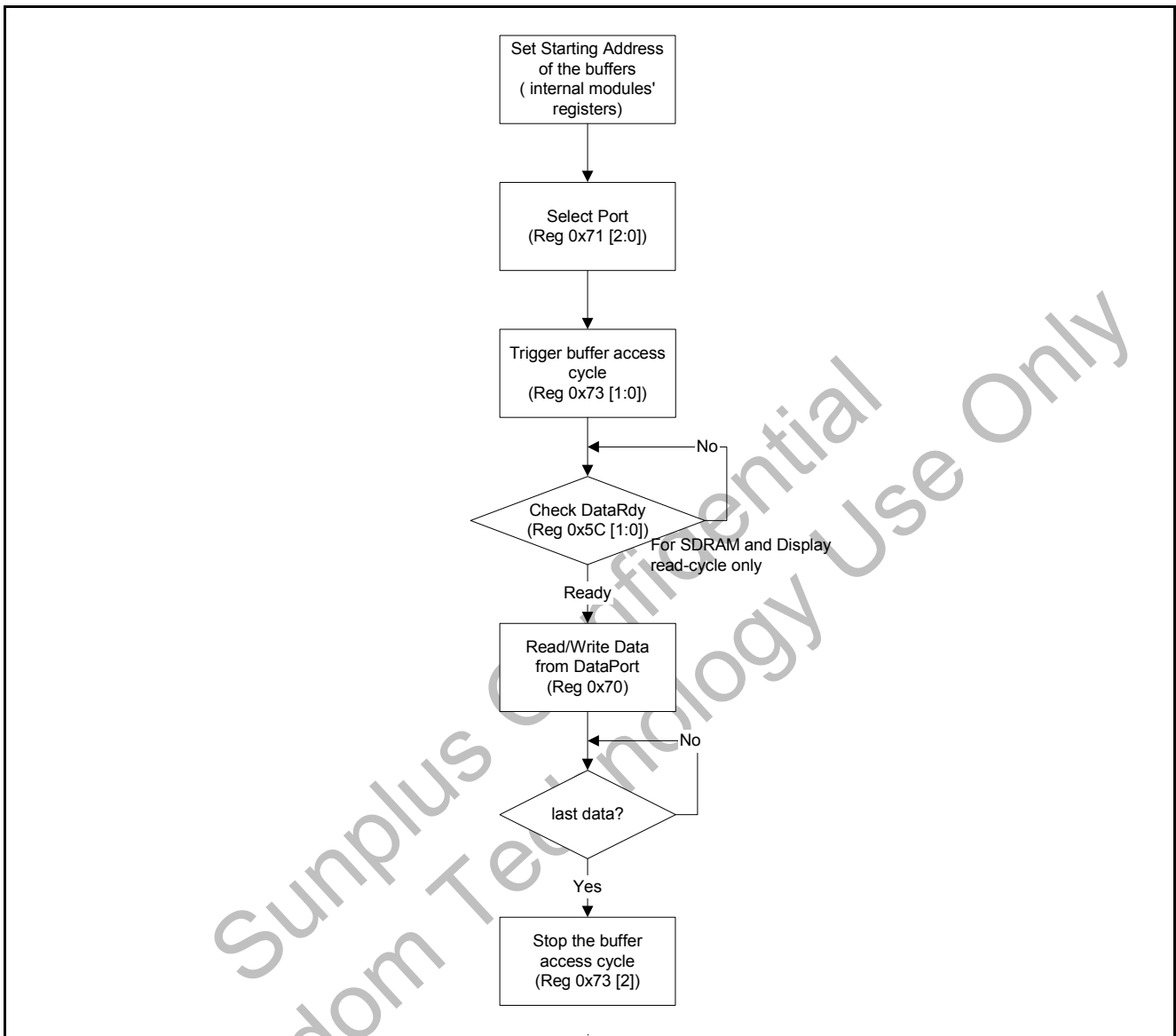


Figure 6-2-5: Internal buffer access flow

6.2.5. Internal register access

There are two methods for the host processor to access SPCA554A internal registers. All the control, data and status registers of internal functional blocks, such as Color DSP registers and MPEG registers, can be accessed via these methods. These internal registers are addressed by 16-bit address. The first method is through the host bridge control registers with an indirect addressing mode. This method works only when the SPCA554A internal CPU is in the reset state. And it is useful for SPCA554A initialization, such as for initializing the program code starting address in the stacked SDRAM. Once the SPCA554A internal CPU starts working, the only way to access the internal registers is through the register-access vendor command. The register access vendor command has a similar format to that

defined in the USB vendor command.

Access internal register by indirect address

Host bridge control registers 0X50~0X54 are used to access SPCA554A internal registers in an indirect addressing manner. The host processor should set the address first in registers Reg0X52 and Reg0X53, then read data from the Reg0X50 port or write data to the Reg0X51 port. Burst access to the internal registers is also supported, with optional address auto-increment. Figure 6-2-6 shows the flow of the indirect addressing mode. When accessing internal registers using the indirect access mode, register 0X7E[0] VCMODE should be set to 0 first.

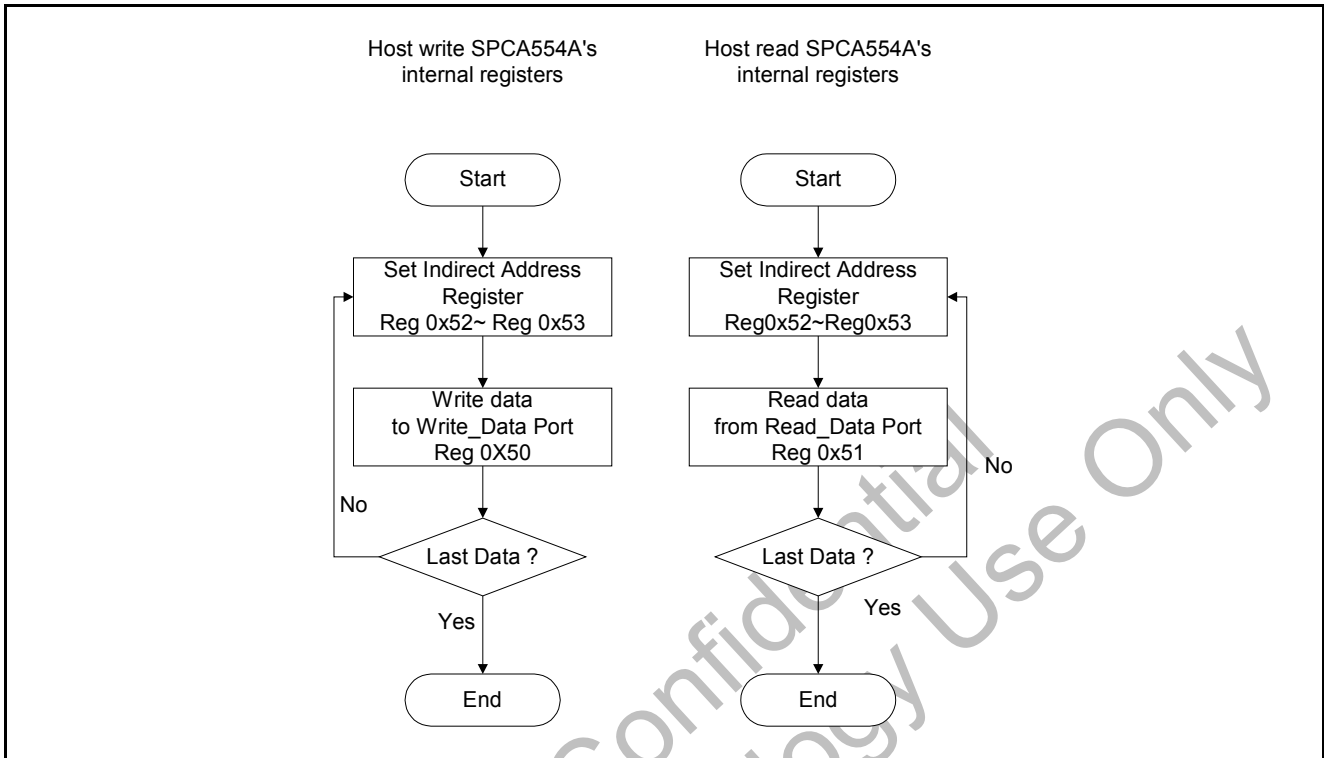


Figure 6-2-6: Flow of indirect address

Access internal register by vendor command

Accessing SPCA554A internal registers via vendor commands has the same vendor command operation flow as described in section 6.2.3. In a read access, the internal CPU reads the register value and puts it in the mirror page for the host to access. In a write access, the internal CPU gets the data from the main page and writes it to the internal register.

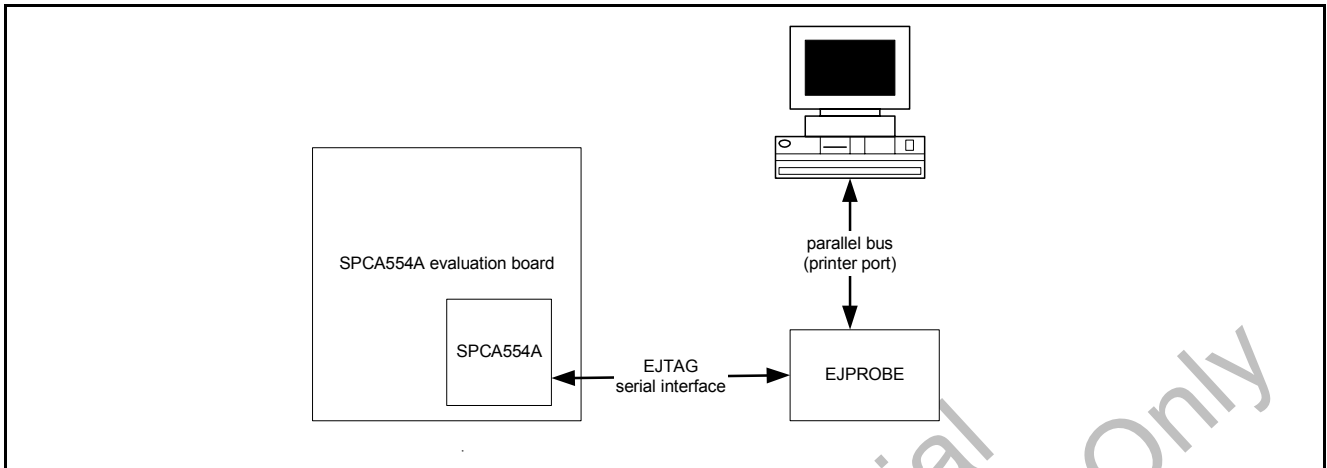
6.3. Embedded CPU

The on-chip 32-bit RISC CPU of the SPCA554A is in charge of camera parameter adjustments, like AE and AWB calculation. It is also in charge of running the audio CODEC, applying special digital effects to the capture camera image, parsing and coordinating the commands from the host, etc.

The CPU sub-system of SPCA554A integrates the CPU core, 2K-byte instruction cache, 4K-byte data cache, and 8K-byte scratch-pad memory. The scratch-pad memory is designed for high-speed data access of the CPU core. The contents of the scratch-pad memory can be swapped with contents of other buffer of SPCA554A through the DMA controller. For example, the data

in the stacked SRAM can be transferred to the scratch-pad memory, and vice versa. The scratch-pad memory is very useful for boosting system performance when the application requires intensive data access to SDRAM buffers. For example, when audio data encoding and decoding (MP3 or AMR), the scratch pad memory can be used as temporary data buffers to hold an entire audio frame data. CODEC efficiency improves drastically by reducing data swapping frequency. The scratch-pad memory can also be accessed by the host processor in a burst manner so it can also be used by the host processor to transfer data to the SPCA554A at higher speeds.

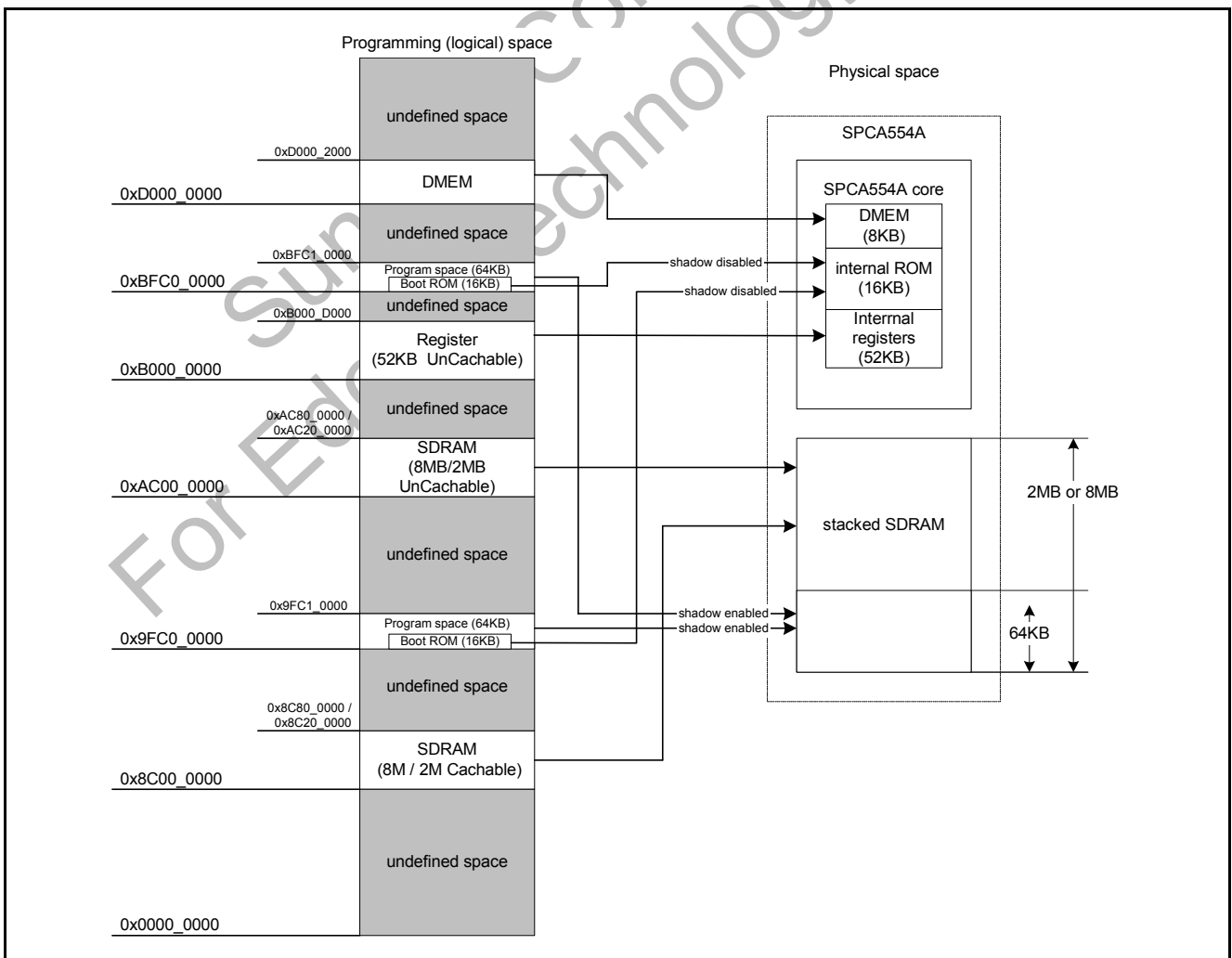
The SPCA554A features an EJTAG debugging interface. With an EJTAG probe, the SPCA554A internal CPU can be fully controlled by the host CPU. A step-by-step Firmware diagnostic is possible by using the IDE (integrating development environment. The IDE enables the user to set SPCA554A firmware hardware breakpoints, and it also enables the user to dump internal memory. The connection is shown in the following diagram:


Figure 6-2-7 EJTAG connection

6.3.1. Addressing space partition

The CPU core uses a 32-bit address to access its peripherals, like SDRAM buffer, scratch memory, internal registers, etc. The 32-bit address yields a 4G bytes addressing space, which is

partitioned as in the following figure. The internal memory management unit translates the logical address to the physical address for the CPU core to access all the peripherals.


Figure 6-3-1 Addressing space partition

The address space partitions are listed in the following table, which reflects Figure 6-3.2.

	Logical Address	Size	Description
Cachable SDRAM space	0x8C00_0000 ~ 0x8C7F_FFFF	8MB	SDRAM space
Program space	0x9FC0_0000 ~ 0x9FC0_FFFF	64KB	Firmware programming space
Uncachable SDRAM space	0xAC00_0000 ~ 0xAC7F_FFFF	8MB	SDRAM space
Register space	0xB000_0000 ~ 0xB000_0FFF	4KB	Global control registers
	0xB000_1000 ~ 0xB000_1FFF	4KB	CPU control registers
	0xB000_2000 ~ 0xB000_2FFF	4KB	CDSP control registers
	0xB000_3000 ~ 0xB000_3FFF	4KB	DMA control registers
	0xB000_4000 ~ 0xB000_4FFF	4KB	Storage media control registers
	0xB000_5000 ~ 0xB000_5FFF	4KB	USB control registers
	0xB000_6000 ~ 0xB000_6FFF	4KB	Audio control registers
	0xB000_7000 ~ 0xB000_7FFF	4KB	Memory Control Registers
	0xB000_8000 ~ 0xB000_8FFF	4KB	MJPEG Registers
	0xB000_9000 ~ 0xB000_9FFF	4KB	Sensor Interface Registers
	0xB000_A000 ~ 0xB000_AFFF	4KB	Display Controller Registers
	0xB000_B000 ~ 0xB000_BFFF	4KB	Host Bridge Registers
	0xB000_C000 ~ 0xB000_CFFF	4KB	2D Graphic Engine Registers
DMEM space	0xD000_0000~ 0xD000_1FFF	8KB	Scratch pad memory

Table 6-3-1 Addressing space partition

Internal register space

The internal registers of the SPCA554A occupy 52K-byte address space. These registers are used by internal function modules to control and report status of a variety of tasks. This space is divided into 13 regions corresponding to 13 internal function modules.

Cachable/Uncachable SDRAM space

The SPCA554A is shipped with a 2M-byte stacked SDRAM (or 8M-byte SDRAM). The stacked SDRAM stores firmware code necessary for on-chip CPU operations. It also holds a lot of image data, such as camera image data and 2D graphics image data. The entire SDRAM is mapped to CPU address space from address 0X8C00_0000 to 0X8C1F_FFFF (or 0X8C7F_FFFF for 8MB SDRAM). This addressing space is cachable. The same SDRAM spaced can also be accessed through the uncachable space, from address 0XAC00_0000 to 0XAC1F_FFFF (or 0XAC7F_FFFF).

Program space

The embedded boot 64K-byte program space is mapped to address 0XBFC0_0000 to 0XBFC0_FFFF, and also 9FC0_0000 to 9CF0_FFFF. The former one is uncachable and the latter one in cachable. Both spaces are mapped to the same physical address space. The boot address of SPCA554A is 0XBFC0_0000, which is in the uncachable program space. The exception (interrupt) entry point is 0XBFC0_0180. It is also in the

uncachable program space. When the program shadow function is turned on, the program space is mapped to a 64K-byte section of the stacked SDRAM. When the shadow function is disabled, the only usable program space is the on-chip 16K-byte boot ROM. For the final implementation of an SPCA554A-enabled device, the shadow function must be turned on if the SPCA554A boots from internal boot ROM or from the host processor. It is the only way to get a reasonable amount of programming space. Note that the firmware program size is not limited to 64K-bytes of the program space, and all the space in the stacked SDRAM can be used to store the firmware code. As described in section 6.7, a typical application of the SPCA554A reserves 512K-bytes for firmware code.

DMEM space

DMEM is the scratch memory described in the previous section. DMEM in the SPCA554A is partitioned into 3 regions; A-DMEM, B-DMEM, and Sp-DMEM. Sp-DMEM is used to store stack data and local variables. Data in this region grows downward from address 0XD0000_7FFF when the stack data grows. A-DMEM and B-DMEM are dual-buffers that can be accessed by the CPU, the DMA controller and the host processor. The dual-buffer architecture allows parallel data processing. For example, when the CPU is processing data in the A-buffer, the DMA can access B-buffer. The dual-buffer architecture increases the efficiency of memory access.

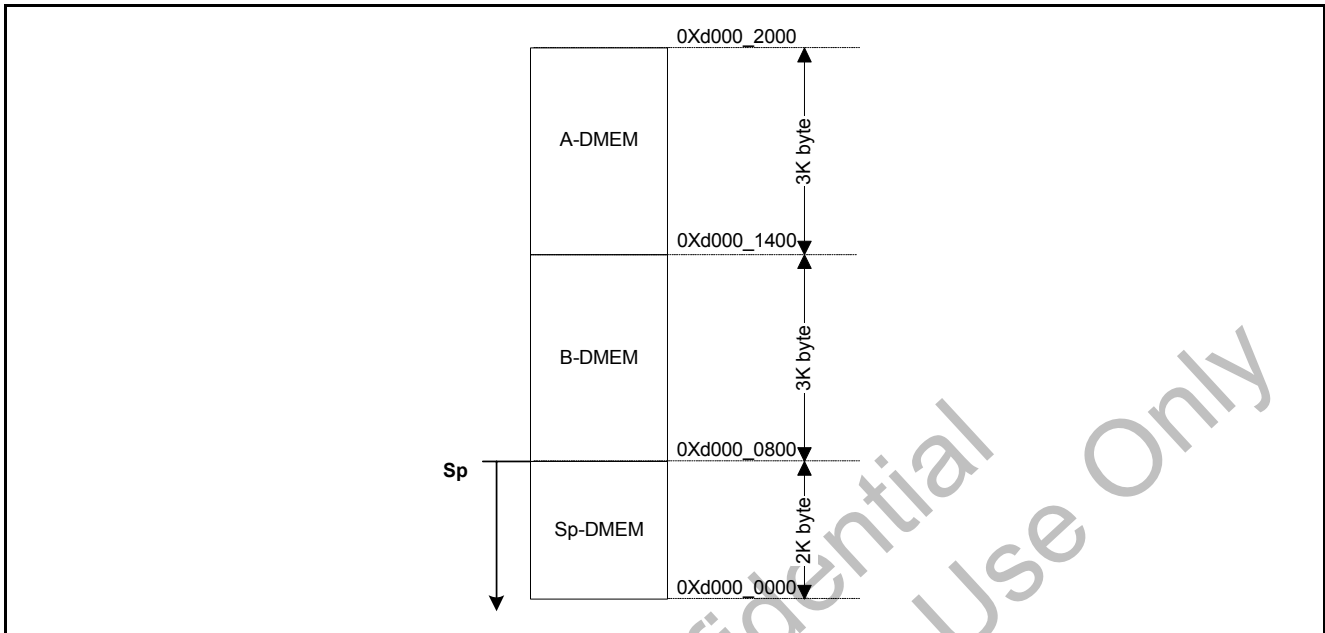


Figure 6-3-2 DMEM partition

6.3.2. Priority interrupt controller

SPCA554A has 23 interrupt sources. Each source is assigned a 5-bit interrupt priority level. Level 1 has the highest priority and level 31 has the lowest priority. Priority level 0 means that the interrupt is disabled. The priority level can be changed dynamically. When an interrupt occur, the SPCA554A interrupt controller reports the interrupt ID with the highest interrupt priority

level to the CPU core. The interrupt ID is reported by the interrupt controller with an internal register named “ Current Interrupt Status”. This register keeps reporting the interrupt ID with the highest priority until the interrupt is cleared. If two interrupts with the same priority level occur at the same time, the one with larger ID is reported.

Interrupt Name	Event Description	Int Num.
GPIOInt	GPIO interrupt is generated when any one of the pins (GPIO0~GPIO11) has a status change.	0x0
TGGPIOInt	When used as GPIO pins, the sensor interface pins can generate an interrupt in the same way as a typical GPIO (GPIO0~GPIO11) pin. TGGPIO interrupt is generated when any one of the TGGPIO has a status change.	0x1
Timer1Int	Global timer1 interrupt. This interrupt is generated when the global timer counts down to zero.	0x2
Timer2Int	Global timer2 interrupt. This interrupt is generated when the global timer counts down to zero.	0x3
Timer3Int	Global timer3 interrupt. This interrupt is generated when the global timer counts down to zero.	0x4
Timer4Int	Global timer4 interrupt. This interrupt is generated when the global timer counts down to zero.	0x5
USBInt	When SPCA554A is used as an USB device, this interrupt is generated whenever a USB data packet is received by the SPCA554A USB controller.	0x6
VDInt	VSYNC interrupt is the SPCA554A vertical synchronization signal of the CMOS sensor interface. This interrupt is generated whenever the signal goes from high to low or from low to high. This interrupt is generated for each incoming camera frame.	0x7
DMAInt	DMA interrupt is generated when a DMA transfer is completed.	0x8
MJPGInt	MJPG interrupt: This interrupt is generated by the MJPG Module to signal when JPEG decoding or encoding is completed.	0x9
DRAMInt	This interrupt is generated by the image controller to signal when a specific task is completed, such as image rotation, and scaling.	0xA
FMIInt	The flash memory interface, when used as GPIO pins, can generate an interrupt signal to the CPU when the input signal status changes. The function is exactly the same as a normal GPIO interrupt.	0xB

Interrupt Name	Event Description	Int Num.
LCMInt	The LCM controller can generate GPIO in two conditions. The first condition is after it writes all frame data to the LCM. An interrupt is generated to signal completion. The second condition is when the interface pins are used as GPIO. These pins can generate interrupts in the same way as a typical GPIO pin.	0xC
AudioInt	When being used as GPIO pins, the audio interface pins can generate an interrupt in the same way as a typical GPIO pin.	0xD
FrontInt	This interrupt is generated by the sensor controller. It signals one of the following: Vertical synchronization signal status change A still image is captured The color processing of a still image is completed A serial interface control protocol (to the sensor) is finished Periodically generated in a programmable timing spot of a frame	0xE
MshutterInt	This interrupt is generated to signal the timing for closing the mechanical shutter. It is used only in an application with a CCD module.	0xF
GpioCounterInt	This interrupt is generated when the GPIO pins are used to count the input pulses.	0x10
UARTInt	UART interrupt. This interrupt has four kinds of interrupts. Transmission FIFO empty Receiving FIFO not empty Receiving FIFO full When error occur in receiving stop bit	0x11
Lbuserrorint	CPU core accesses the unmapped address space. This interrupt is generated to signal an error condition.	0x12
OverBoundInt	Reserved.	0x13
Hostvcint	Reserved.	0x14
HBint	Reserved.	0x15
hostoprdyint	Reserved.	0x16
hostcmdrdyint	Host command ready interrupt. This interrupt is generated as a part of the host interface handshake protocol.	0x17

Table 6-3-2 Internal \ interrupt source list

6.4. Sensor Control

6.4.1. Synchronization/ Valid signals generation

The SPCA554A CMOS sensor controller supports both master-type CMOS sensors and slave-type CMOS sensors. The CMOS sensor interface has vertical synchronization (VD) and horizontal synchronization (HD) signals to construct image frame timing. These signals are driven by the CMOS sensor when SPCA554A interfaces to a master-type CMOS sensor. And they are driven by SPCA554A when SPCA554A interfaces to a slave-type CMOS sensor. The polarities of the synchronization signals are programmable. HD and VD are used to reset the

Color DSP pipeline at the end of each image line and the end of each frame. Due to the pipeline architecture of the Color DSP, pixel color processing has several clocks/lines of pipeline delay. The HD and VD active periods must be relocated to avoid accidentally resetting the Color DSP pipeline when the input image is not completely processed. The following diagram shows the re-location of HD and VD signals. The re-located HD and VD are used internally in the SPCA554A Color DSP pipeline. The timing is programmable through a set of registers.

FrameTotal: Defines total number of lines in an image frame.

FrameBlank: Defines the VD active period. It is used only when the slave mode CMOS sensor is connected to SPCA554A.

VFall: Defines the falling edge timing of the re-located VD.

VRise: Defines the rising edge of the re-located VD.

Voffset: Defines the starting line of the active lines.

Vsize: Defines the total number of active lines in a frame.

The VDValid signal indicates the active lines of an image frame.

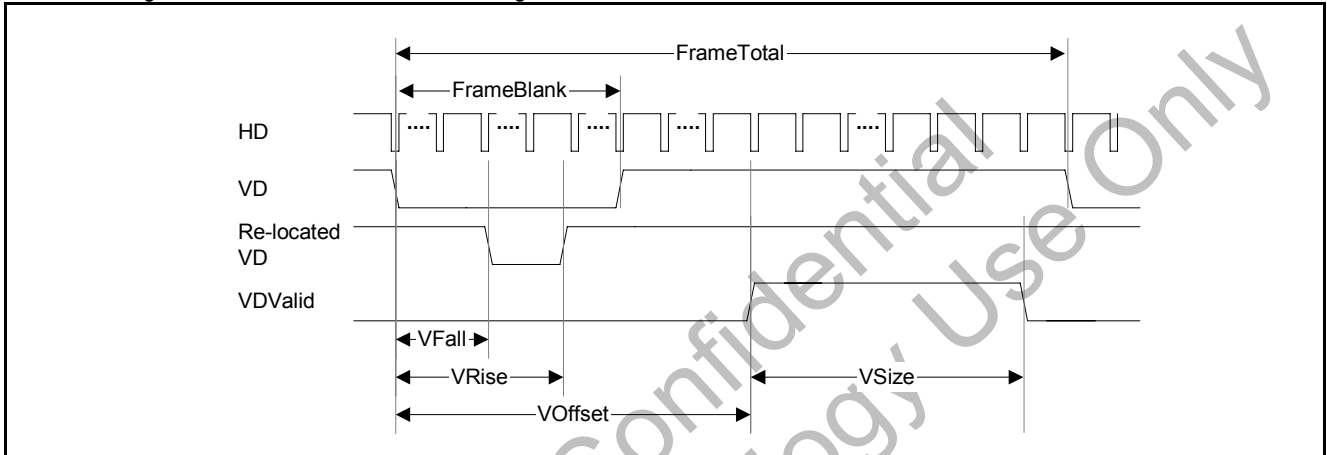


Figure 6-4-1: Vertical timing

LineTotal: Defines total number of pixels in an image line.

LineBlank: Defines the HD active period. It is used only when the slave mode CMOS sensor is connected to SPCA554A.

HFall: Defines the falling edge timing of the re-located HD.

HRise: Defines the rising edge of the re-located HD.

Hoffset: Defines the starting pixel of the active pixels.

Hsize: Defines the total number of active pixels in a line.

The HDValid indicates the active pixels of an image line.

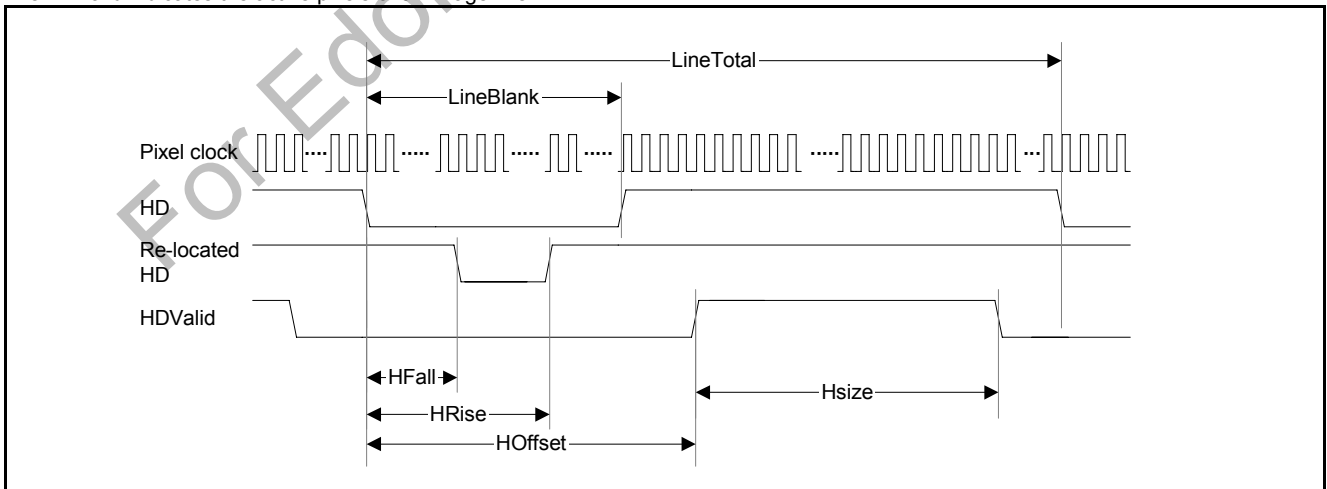


Figure 6-4-2: Horizontal timing

These parameters normally take effect immediately after they are programmed. An option to delay the effective time of these parameters is also implemented in SPCA554A, which delays the

parameter effective time until the next frame. Six parameters (LineTotal, FrameTotal, HOffset, HSize, VOffset and Vsize) can optionally delay their effective time to the next frame.

6.4.2. Image cropping

The first step for the sensor controller to process the incoming image is image cropping. By setting the Voffset, Hoffset, Vsize and Hsize registers, the active image region can be defined. The

active image region definition actually crops the image. Note that Voffset and Hoffset should be set by an even number of pixels because it needs to align to the Bayer pattern format.

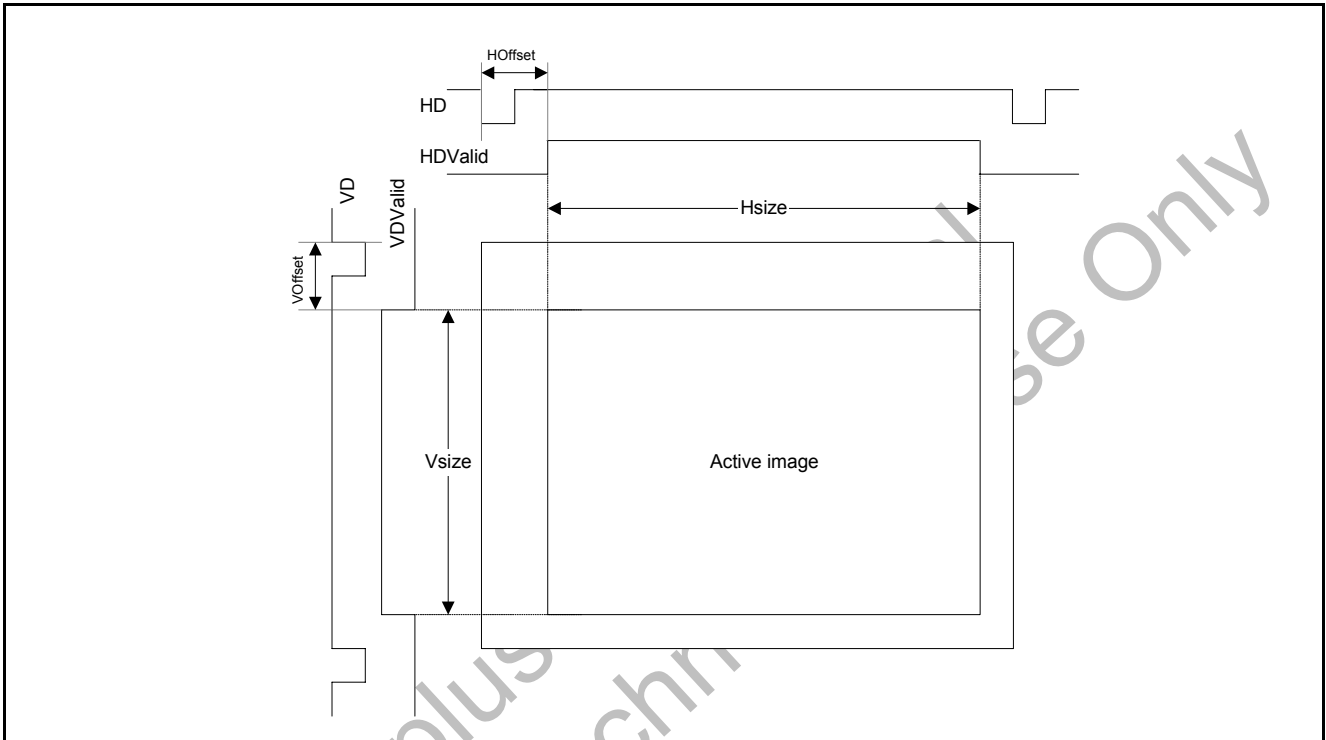
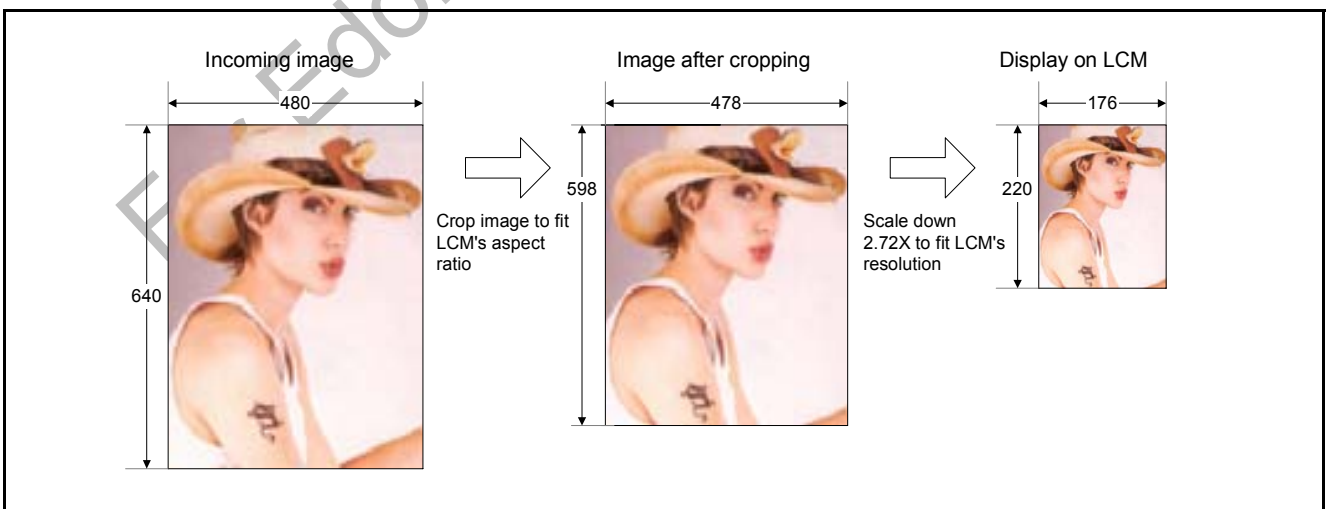


Figure 6-4-3: The cropping operation for sensor controller

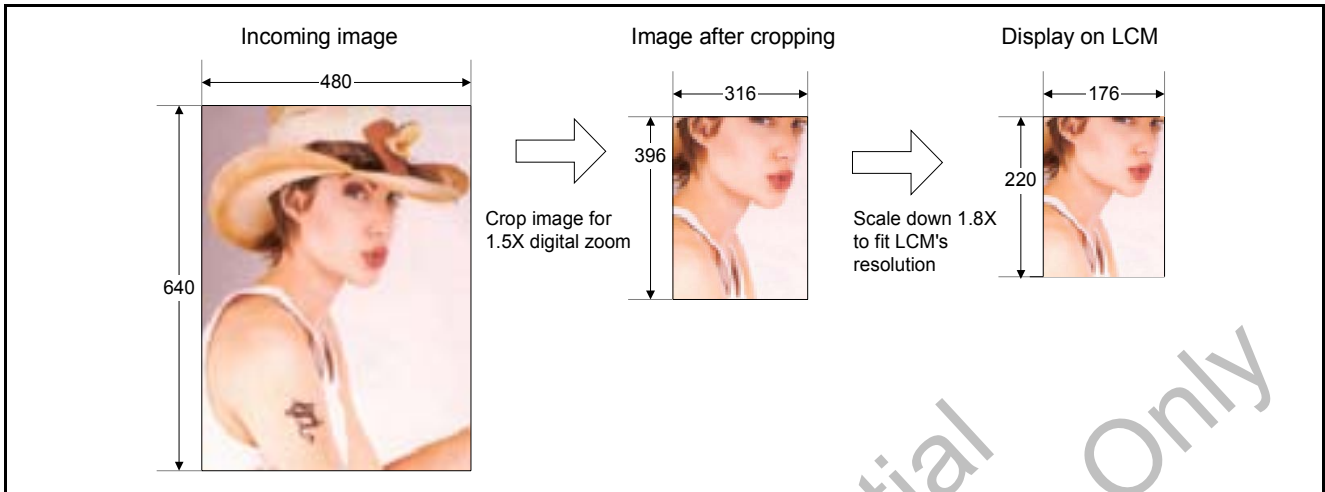
Image cropping can be used to fit the LCM aspect ratio. The following example shows a VGA resolution image to be shown on a 176x220 resolution LCM. The image needs to be cropped to a

478x598 resolution to meet the LCM 4/5 aspect ratio. The original image aspect ratio is 3/4. After cropping, the image is scaled down by 2.72X to fit the LCM resolution.



Another application of image cropping is digital zoom. The following example shows a 1.5X digital zoom. The digital zoom ratio is obtained by the coverage ratio of the cropped image over the original image. The cropped image still needs to be scaled

down to fit the LCM resolution. The digital zoom function described here applies to both the camera preview mode and the video recording mode.



6.4.3. Exposure Time and Flash Light Control

Typical CMOS sensor exposure time control uses a “rolling shutter” mechanism. When using rolling shutter, each line of the CMOS sensor takes turns getting exposure. Even though the total exposure time (from reset to readout) of all lines is the same, the pixels in different lines get different exposures in terms of time axis. Figure 2-40 shows how the rolling shutter mechanism works. The exposure time is actually controlled by when to reset a line, which is programmable inside the CMOS sensor. The SPCA554A has a serial bus to program the internal registers of the CMOS sensor.

Due to the nature of the rolling shutter, there is a window when all lines of the CMOS sensor get exposure at the same time. The shorter the exposure time is, the narrower the window is. The flash light strobe, once applied, must be applied within the window to make all lines exposed to the flash light. SPCA554A's flash light control relies on firmware to calculate the appropriate timing for flash light strobe. TGGPIO[0] is used as flash light strobe control signal, which sends a positive (or negative) pulse to the flash light module. The pulse is triggered by the CPU and its width is programmable.

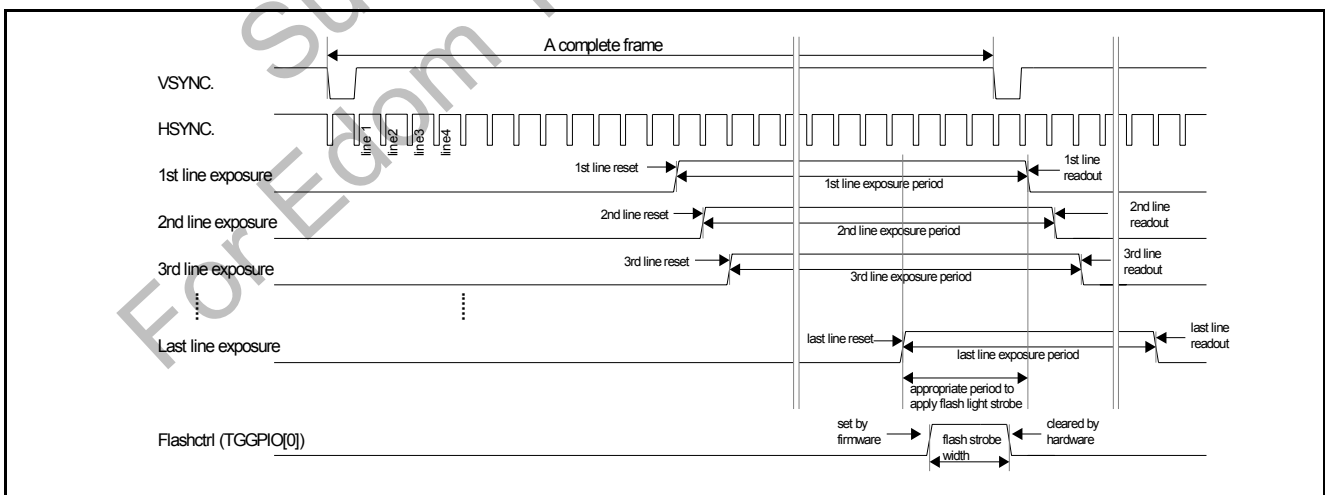


Figure 6-4-5: Timing diagram for flash light control

While adjusting CMOS sensor exposure time, it is important to keep the exposure time in multiples of 1/120 sec in 60Hz power system areas and 1/100 sec in 50Hz power system areas. If this condition is not met, different lines of the CMOS sensor will

integrate different energy even when their exposure times are equal. The following diagram shows an example. When a CMOS sensor image lines receives different integration energy, the banding effect emerges.

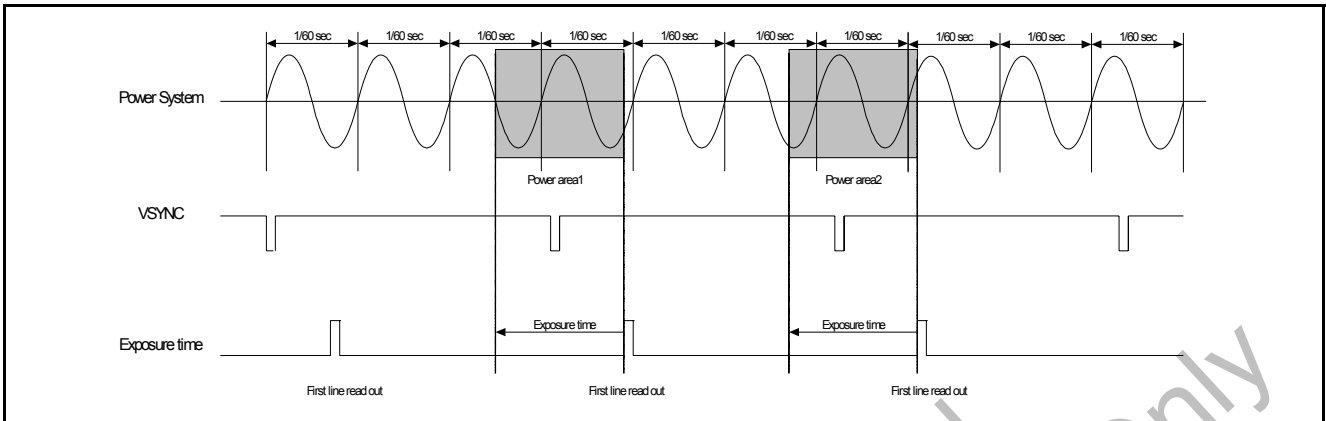


Figure 6-4-6: Banding occurs

The maximum exposure time is equal to one frame. In an extremely low light environment, the SPCA554A reduces the frame rate to obtain longer exposure times.

6.5. Color DSP

The color DSP is a hard-wired pipeline image-processing engine. It performs various image processing tasks on raw data from image sensor or YUV422 data and outputs YUV data to the memory controller. Each processing stage is controlled by a rich

set of color parameters, allowing the SPCA554A to fine-tune the image quality. Figure 7-19 shows the pipeline architecture in the raw data input mode. Figure 7-20 shows the pipeline architecture in the YUV data input mode.

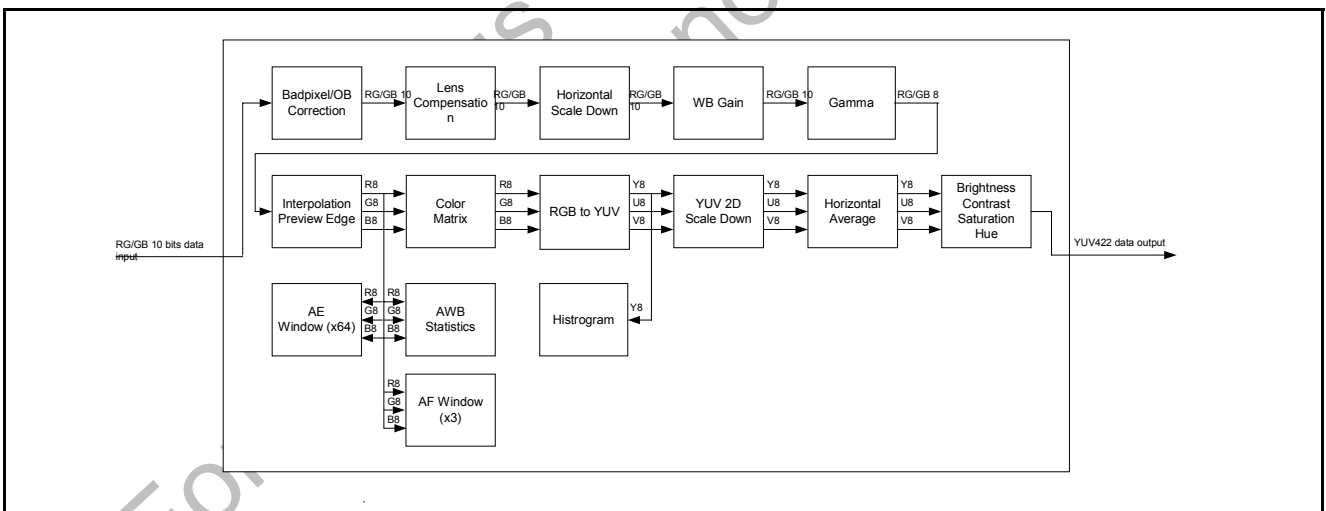


Figure 6-4-7: RGB Bayer pattern - input pipeline

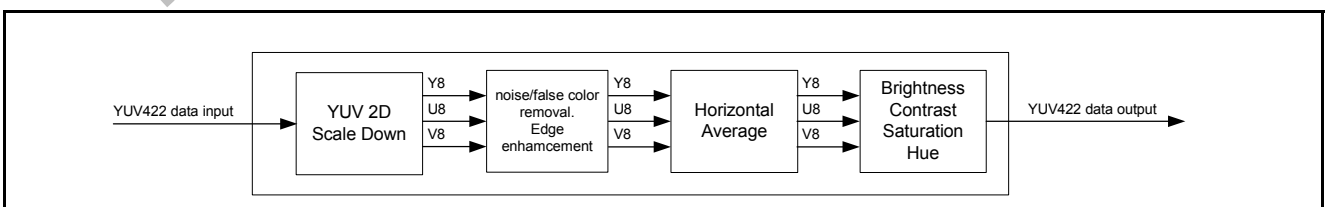


Figure 6-4-8: YUV422 - input pipeline

Bad pixel canceling

In the SPCA554A, two approaches have been implemented for bad pixel concealment. The first approach is designed for the camera preview and video recording modes. It adopts a modified median filter to conceal bad pixels in real time. The second approach adopts a more location accurate method and is used in the still image capture mode. In this mode, the bad pixel locations must be calibrated in advance. The SPCA554A conceals the bad pixel by referring to neighboring pixels. The SPCA554A is able to conceal unlimited bad pixels in still image capture.

Lens shading compensation

The SPCA554A uses a look-up-table for lens shading compensation. The spatial distance to the center point is used as the index for the look up the table. The table is programmable and has 256 entries. The center point coordinate is also programmable.

Horizontal scaling down (RGB domain)

This function is usually used in the preview or video recording mode. In these modes, the resolution of the sensor is usually higher than the resolution of the display device. The input image is required to be scaled down to fit the resolution of the display device. The most effective way is to scale down the image in the RGB Bayer-pattern domain because reducing the resolution in the early stage can also reduce the power consumption necessary for image processing. The scaling factor can be programmed through the internal registers of the Color DSP.

White balance gain

The white balance adjustment applies different gains to Gr, R, Gb and B components of the input raw data. The gain values are calculated by the SPCA554A AWB algorithm based on AWB window statistics.

Gamma correction

SPCA554A gamma correction is done utilizing a fully programmable look up table. After Gamma correction, the image data is converted to 8-bit RGB data.

Interpolation

SPCA554A color interpolation transforms Bayer-pattern raw data into full RGB data. The noise removal function is combined with the interpolation function to obtain cleaner images. The edge information can also be emphasized in SPCA554A interpolation.

Color correction

Color correction is done by multiplying the input image by a 3x3 matrix. The parameters in the matrix are programmable based on the characteristics of sensor components.

RGB-to-YUV conversion

Standard color space conversion from RGB domain to YUV domain.

YUV 2D scaling down

The function block scales down the image further before storing the image into stacked SDRAM. It is used together with the raw data horizontal down-scaling function to obtain the desired resolution.

Brightness, contrast, saturation and hue

The final stage of the color DSP pipeline is to adjust contrast, saturation and hue. These parameters are programmable.

AE window statistics

SPCA554A partitions the active image region into 8x8 equal-sized AE windows. A Total of 64 statistic values are reported to the on-chip CPU for AE adjustment.

AWB statistics

SPCA554A identifies the potential white pixels of the incoming image. All the Y value, (R-G) value and (B-G) of the potential white pixels are accumulated and reported to the on-chip CPU for AWB adjustment.

Digital special effects

Digital filters can be applied to the incoming image in the color DSP. The special image effects are feasible in the camera preview mode and still image capture. The SPCA554A has implemented 6 special filters; negative, solar, emboss, binary, sepia and black-white.

Histogram accumulation

Luminance histogram is also implemented in the SPCA554A color DSP. It assists the on-chip CPU in avoiding over-exposure and under-exposure in certain parts of the image.

Post Processing

For still image capture, the image can go through the color DSP pipeline multiple times. The parameters in the color DSP can be adjusted for each time. This feature allows the SPCA554A to perform post-processing on the image. Typical post-processing in the SPCA554A includes, edge enhancement, noise removal and false color suppression.

6.6. JPEG and Video CODEC

The still image CODEC and video CODEC in the SPCA554A are highly efficient and are low power CODECs designed for mobile application. For image capture and video recording, the output data (after compression) can be sent to the host processor or sent to the external storage media. For still image and video playback, the input source can be from the host processor or the external

storage media.

6.6.1. Pre-Processing

Before still image compression (JPEG) and video compression (MPEG4/H.263), several image pre-processing functions are implemented in SPCA554A. They include, data range conversion, image cropping, image up-scaling and image rotation.

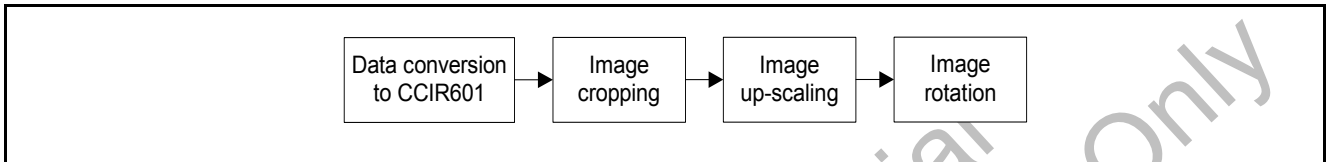


Figure 6-6-1 Pre-processing Flow

YUV to CCIR601 Conversion

The values of image data from the color DSP range from 0 to 255. To comply with MPEG standards, image data should be converted to the CCIR601 data range, from 16 to 235. The conversion formula is shown below:

$$Y = (Y * 219) / 256 + 16$$

$$U = (U * 224) / 256 + 16$$

$$V = (V * 224) / 256 + 16$$

Image Cropping

The MPEG/JPEG compression engine of the SPCA554A can be defined as a rectangular region out of its corresponding image buffer in the stacked SDRAM. And only the rectangular region is to be compressed. The rectangular region is defined by four

internal registers; Voffset, Hoffset, Height and Width. Both Height and Width of the region must be multiples of 16 pixels due to the block-based characteristics of JPEG/MPEG. There is no alignment requirement for Hoffset and Voffset.

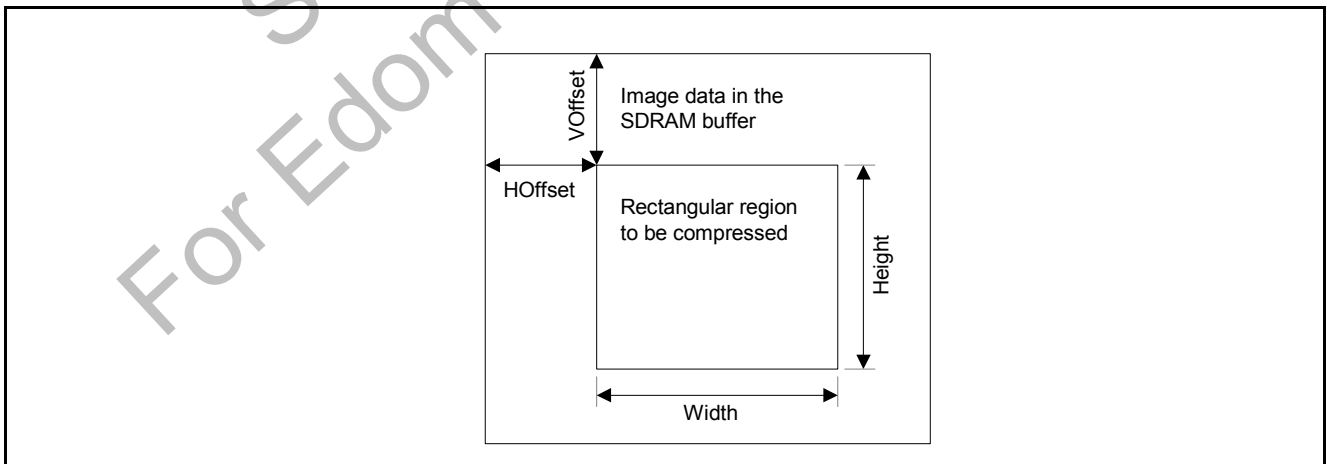


Figure 6-6-2 Image Cropping

Image Up-Scaling

The image up-scaling function in the MPEG CODEC is dedicated to digital zoom applications in the video recording mode. It is able to do real time image up-scaling. The up-scaling ratio is from 1 to 4 with a step of 0.01. The digital zoom in the video recording mode of SPCA554A is done by two steps. The first step is to

disable the down-scaling function in the color DSP pipeline. The second step is to use the image cropping and up-scaling function of the video CODEC. The later step is used in the high zoom ratio. This two-step scheme achieves the best video quality. The following figures illustrate the digital zoom functions of these

two steps.

In figure 6.6.3, the sensor resolution is 640x480, and the LCM resolution is 176x220. And the target resolution of the video recording is 240x320. For a digital zoom ratio of 2X and below, the color DSP downscaling function is used to control the digital zoom ratio. But for digital zoom ratio over 2X, the video CODEC

cropping and scaling function must be turned on. In this example, the image data in the SDRAM buffer is zoomed by 2X by turning off the downscaling engine in the color DSP. The view angle of the image data in the SDRAM buffer is 1/2 or the incoming image. The MPEG CODEC cropping and up-scaling function provides further 2X zoom. Overall, the video recorded is digitally zoomed by 4X.

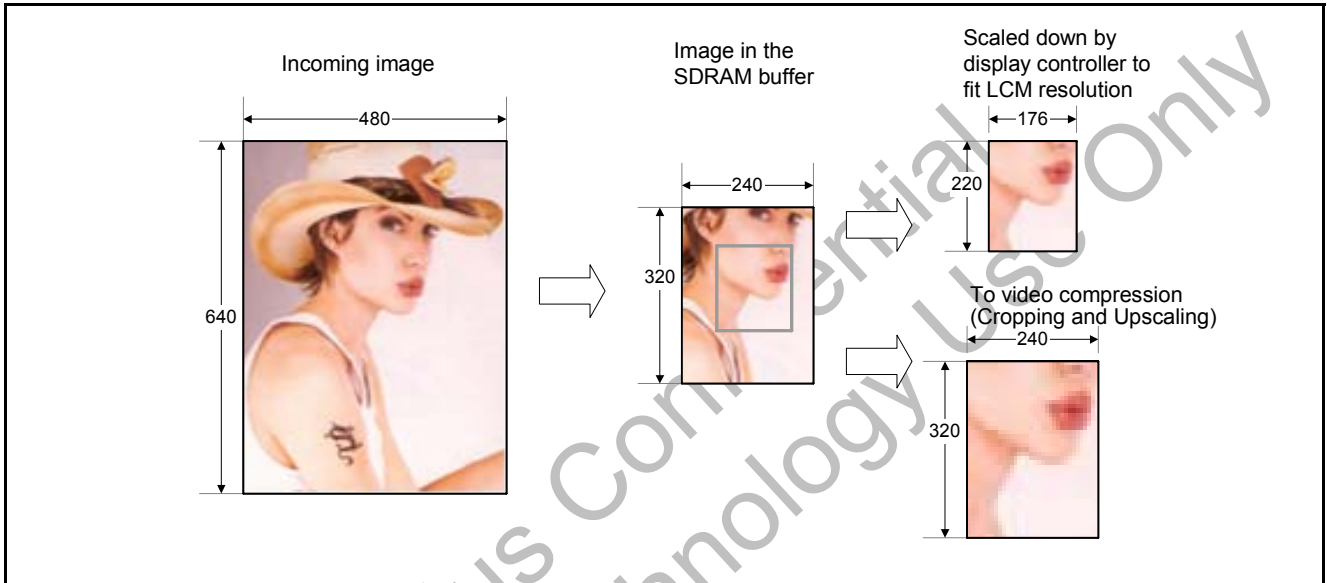


Figure 6-6-3 Video recording with high digital zoom ratio

Rotation

Both still image compression and video compression support the rotation and mirror operation when the compression engines fetches data from the SDRAM buffer. This function is done in

real-time and no extra operational delays are induced by rotation and mirror. The following figure shows the eight available rotation/mirror modes.









Normal	Rotation 90°	Rotation 180°	Rotation 270°
			
Horizontal mirror	Vertical mirror	Vertical mirror + Rotation 90°	Vertical mirror + Rotation 270°
			

Figure 6-6-4 JPEG/MPEG CODEC rotation modes

6.6.2. JPEG engine

The JPEG CODEC is for still image compression (image capture) and decompression (image playback). SPCA554A JPEG CODEC conforms to the ISO/IEC 10918-1 international standards. Quantization tables and Huffman tables are programmable in SPCA554A JPEG CODEC to improve compatibility. With the

assistance of firmware, the JPEG CODEC can generate (decode) standard JPEG files in JFIF format. All the pre-processing functions described in the previous sub-section are applicable to JPEG CODEC. The following table shows the data formats supported by SPCA554A JPEG CODEC.

Format	YUV420	YUV422	YUV444	YUV411
JPEG Compression	Yes	Yes	No	No
JPEG Decompression	Yes	Yes	Yes	Yes

Table 6-6-1 Data formats supported by SPCA554A JPEG CODEC

Still image Compression

Pre-processing can perform image cropping and up-scaling before still image compression. After compression, a thumbnail image is automatically generated. The resolution of the thumbnail image is 1/8 of the original image. For example, if the input image resolution is 640x480, then the thumbnail resolution is 80x60. The thumbnail can be attached to the compressed image to form a complete JFIF file. It is useful for accelerating playback speed when small image icons are to be viewed first.

Still Image De-compression (Playback)

In still image playback, The SPCA554A JPEG decoder is able to extract only a portion of the image. It works in the same way as cropping in the compression mode. Users are required to input Voffset, Hoffset, Width and Height parameters through internal registers. Also, the decoder is able to downscale the image at the same time it decodes the image. This feature is helpful in reducing power consumption and enhancing playback speed because the LCM resolution is normally lower than the image resolution. However, the downscaling ratio is limited. SPCA554A JPEG decoding is able to down scale the image from a ratio of 1/8 to 7/8 with a 1/8 step. Downscaling the image with JPEG decoder first is recommended, and then downscaling again using the display controller scaling engine. The final image resolution depends on the LCM resolution and application requirements.

A stand-alone JPEG CODEC

In addition to typical applications in which image data is compressed from the color DSP, JPEG CODEC can also be used as a stand-alone still image CODEC so as to conserve host processor power. Uncompressed images can be sent to the SPCA554A, then, the SPCA554A compresses the image and returns a JPEG file to the host processor. Alternatively, the JPEG file can be sent to the SPCA554A and then the SPCA554A returns a decompressed image to the host processor.

6.6.3. Video engine

For video compression and decompression, SPCA554A video CODEC supports motion JPEG, MPEG4 simple profile and H.263 (Profile 0) video coding standards. The file formats can be ASF, AVI, 3GP, MOV, or MP4. All the pre-processing functions described in section 6.6.1 are applicable to video applications.

Video Recording

To comply with the video coding standards, the video resolution must be multiples of 16 pixels in both the horizontal and vertical directions. For example, a 352x288 video resolution is feasible in SPCA554A, but 160x120 is not feasible. The SPCA554A tries to crop the image to a bigger resolution when their resolution is not multiples of 16 pixels. For example, a 160x128 video resolution is adopted instead of 160x120. The video recording frame rate is 30 fps at a 176x144 resolution and 15 fps at a 352x288 resolution, respectively. There is no limitation on the recording time as long as the external storage media provides enough storage space. When directing the encoded bit stream to the host processor, SPCA554A continues recording and sending data to the host processor until the host stops recording.

In an audio-enabled application, the encoded audio bit stream can be integrated with the video bit stream and then stored to external storage media (or sent to the host processor).

Video Playback

In video playback, the on-chip CPU is responsible for all bit-stream parsing in order to get rid of the file header and record related parameters in the header. All video file formats, ASF, AVI, 3GP, MOV, or MP4, are supported. Also the on-chip CPU is also in charge of the separation of audio and video bit streams. It sends the video part to the video CODEC for decoding and decodes the audio part independently.

Video Conference

SPCA554A supports full-duplex video CODEC functions for applications that require encoding and decoding at the same time, such as video-conference applications. In this type of application, the video CODEC utilizes the time-sharing scheme for decoding

and encoding. The frame rate is 7.5 fps at a 352x288 resolution and 15 fps at a 176x144 resolution, respectively. The far-side and near-side video can be rendered together on the LCM, as illustrated below.



Figure 6-6-5 Video Conference

6.7. Memory Controller

The memory controller of the SPCA554A interface is used to the stack low power SDRAM. All the internal module SDRAM access to SDRAM is through the memory controller. The stacked SDRAM holds all temporary data required during SPCA554A operation. The data in SDRAM includes program code, camera image buffer, graphics image buffer, display buffers, etc.

The arbiter in the memory controller decides the access priority between the host bridge, on-chip CPU, JPEG and MPEG CODEC, display controller, DMA controller, 2D graphic engine, color DSP,

and audio controller. The priority is programmable. The SDRAM access clock can be 48 MHz or 96MHz.

Fig6-26 illustrates memory space allocation, taking a 1.3M(1280x960) camera as an example. Note that all the buffer sizes and locations in SDRAM are programmable so that the SPCA554A can fit to different applications with different resolutions of cameras. Note that SPCA554A partitions memory space according to specific operations so as to keep the memory requirement at a minimum.

SDRAM space partition for 1.3M (1280x960) sensor

Preview mode	Still image capture mode (VGA)	Still image capture mode (1.3M)	Still image playback mode	Video recording mode	Video playback mode	Image editing mode	JAVA mode	Native game mode	
CPU memory (program & data) 512K bytes maximum	CPU memory (program & data) 512K bytes maximum	CPU memory (program & data) 512K bytes maximum	CPU memory (program & data) 512K bytes maximum	CPU memory (program & data) 512K bytes maximum	CPU memory (program & data) 512K bytes maximum	CPU memory (program & data) 512 K bytes maximum	Java class library 622 K bytes	OS 16 K bytes	
Bad pixel coordinates 1024	Bad pixel coordinates 1024	Bad pixel coordinates 1024	Bad pixel coordinates 1024	Bad pixel coordinates 1024	Bad pixel coordinates 1024	Bad pixel coordinates 1024		2D Graphics Driver 114 K bytes	
CDSPP window 128	CDSPP window 128	CDSPP window 128	CDSPP window 128	CDSPP window 128	CDSPP window 128	CDSPP window 128	KVMIDP Code 402 K bytes	Graphics Back Buffer A 176x220x2=76K	
Graphics Sprite Buffer 153600	raw frame buffer 640x480*2=614400	raw frame buffer & VLC buffer 1280x960=1228800	Graphics Sprite Buffer 153600	Graphics Sprite Buffer 153600	Graphics Sprite Buffer 153600	Graphics Sprite Buffer 153600		Graphics Back Buffer B 176x220x2=76K	
Graphics System Block 31744 bytes			Graphics System Block 31744 bytes	Graphics System Block 31744 bytes	Graphics System Block 31744 bytes	Graphics System Block 31744 bytes	Graphics System Block 31744 bytes	Handshake function 11K Bytes	
Graphics command ring 1024 bytes			Graphics command ring 1024 bytes	Graphics command ring 1024 bytes	Graphics command ring 1024 bytes	Graphics command ring 1024 bytes	Graphics command ring 1024 bytes	Graphics Back Buffer (176x220x2) 76K Bytes	
Graphics Back Buffer A 320x240x2=153600			Graphics Back Buffer A 320x240x2=153600	Graphics Back Buffer A 320x240x2=153600	Graphics Back Buffer A 320x240x2=153600	Graphics Back Buffer A 320x240x2=153600	Graphics Back Buffer A 320x240x2=153600		
Graphics Back Buffer B 320x240x2=153600			Graphics Back Buffer B 320x240x2=153600	Graphics Back Buffer B 320x240x2=153600	Graphics Back Buffer B 320x240x2=153600	Graphics Back Buffer B 320x240x2=153600	Graphics Back Buffer B 320x240x2=153600		
preview_buffer_A 320x240x2=153600			video_buffer_A 320x240x2=153600	video_buffer_B 320x240x2=153600	video_buffer_A 320x240x2=153600	video_buffer_E 320x240x2=153600	video_buffer_F 320x240x2=153600		Image buffer 640x480x2=614400 (max.)
preview_buffer_B 320x240x2=153600					video_buffer_C 320x240x2=153600	video_buffer_D 320x240x2=153600	VLC ring buffer 256k		
Free area (924544 bytes)	yuv frame buffer 640x480*2=614400	VLC buffer 1280x960x2/4=614400	VLC buffer_A 320x240x1.5/8=14400	VLC buffer_B 320x240x1.5/8=14400	audio ring buffer 32K (32768)	VLC buffer 640x480x2/4=153600	2D Graphics Driver 114K Bytes		
			ASF buffer 16K (16384)	Free area (460548 bytes)	Free area (310144 bytes)		OS 16K Bytes		
			VLC ring buffer 256k	Free area (156544 bytes)	Free area (415488 bytes)		Handshake function 11K Bytes		
	VLC buffer 640x480x2/4=153600	cdsp temporary buffer 36-lines A/B 1280x36x2x2=184320	Thumbnail Group Buffer 320x240x2=153600	audio ring buffer 32K (32768)	ASF buffer 16K (16384)	Free area (460548 bytes)	Free area (310144 bytes)	Native Game + Graphics Sprite Buffer 1755K Bytes	
	Thumbnail buffer 160x120x2=38400	Thumbnail buffer 160x120x2=38400	Free area (120192 bytes)	Free area (120192 bytes)	Free area (120192 bytes)	Free area (120192 bytes)	Free area (120192 bytes)	Java internal Heap 467K Bytes	
	Free area (35712 bytes)	Free area (120192 bytes)	Free area (120192 bytes)	Free area (120192 bytes)	Free area (120192 bytes)	Free area (120192 bytes)	Free area (120192 bytes)	Java internal Stack 100K Bytes	
							Graphics Sprite Buffer 240K Bytes		

Figure 6-7-1: SDRAM space allocation
Digital camera-related buffers:

- Bad pixel coordinates buffer: To store bad pixel coordinates of sensor
- CDSPP window buffer: To save the AE window statistics values
- Preview A/B buffers: To store YUV data of camera image from Color DSP in preview mode
- Raw frame buffer: To store RGB Bayer pattern raw data from sensor in still image capture mode
- YUV frame buffer: To store the YUV camera image data after color processing in still image capture mode
- Thumbnail buffer: Stores DC terms as the thumbnail camera image after still image compression
- VLC buffer (capture mode): Compressed JPEG bit streams in the still image capture mode.
- Color DSP temporary buffer: This buffer is used by the Color DSP as a temporary buffer to send processed data to JPEG encoder.
- JPEG decompression buffer: This buffer stores YUV camera image data after JPEG decompression.
- Thumbnail group buffer: For viewing multiple thumbnail images at the same time. This buffer is reserved for tailoring thumbnail image.
- Video A/B buffers: These buffers store YUV camera image data for video recording.
- Video C/D buffers: These buffers store reconstructed camera image data used in MPEG/H.263 compression.
- Video E/F buffers: These buffers store the current reconstructed frame and last reconstructed frame for MPEG/H.263 video de-compression.
- VLC A/B buffers: These buffers store VLC bit streams in video compression.
- Audio ring buffer: This buffer is used to store PCM data from/to audio CODEC.
- ASF buffer: This buffer is used to pack/unpack audio and video bit streams to/from ASF, 3GP files.
- VLC ring buffer: In the video recording mode, this buffer is used as a temporary buffer for VLC bit streams before packing

with the

audio data. In the video playback mode, this buffer is used to store VLC bit streams after unpacking audio and video data.

Source image buffer: This buffer stores original image for special effects editing on a play-back image.

Destination image buffer: This buffer stores the image processed by the image controller.

2D Graphics engine-related buffers:

Graphics Sprite buffer: This buffer is used to supply a number of graphic elements used as the materials to construct an image.

Graphics Back A/B buffers: Stores RGB data of the graphics drawn by 2D graphics engine. A/B buffers allow the operations of 2D graphics engine writing buffer and display controller reading buffer to be activated at the same time so that animation can be implemented.

Graphics System Block: This block is managed by the 2D graphics driver for house-keeping.

Graphics command ring buffer: 2D graphics command queue.

Java-related buffers:

Java Class Library: The library includes a number of packages of Java Classes that support J2ME

KVM/MIDP code: This buffer includes main body of Java virtual machine.

Hand-shaking functions: Firmware code for handshaking with the host processor.

Java Internal Heap: Heap is created by JAVA virtual machine for JAVA run time usage.

Java Internal Stack: Stack is created by JAVA virtual machine for JAVA run time usage.

6.7.1. Camera Image of YUV422 format

The camera image data format is illustrated in the following figure. The SPCA554A packs 2 pixels of Y (luminance) data into a single word to improve memory access efficiency for video CODEC. So

does the chrominance data. The following diagram shows an example of a (8x8) pixel of YUV422 image in the SDRAM buffer:

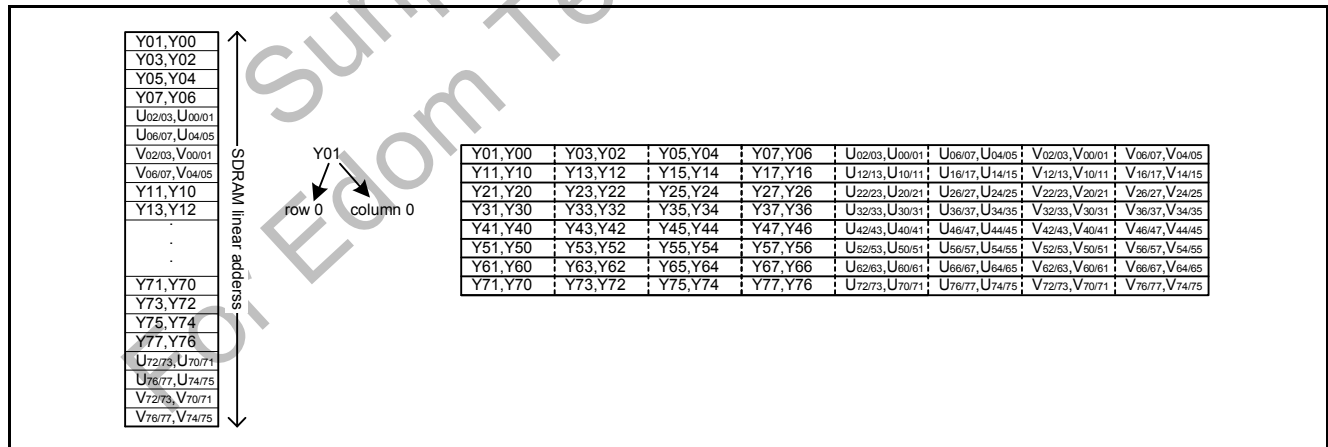


Figure 6-7-2: YUV422 image in the SDRAM buffer.

6.7.2. Camera image of YUV420 format

The YUV420 format is similar to the YUV422 format except that the chrominance of each odd line is not valid because UV is sub-sampled in the vertical direction. The following diagram shows an example of a (8x8) pixel of YUV420 image in the SDRAM buffer:

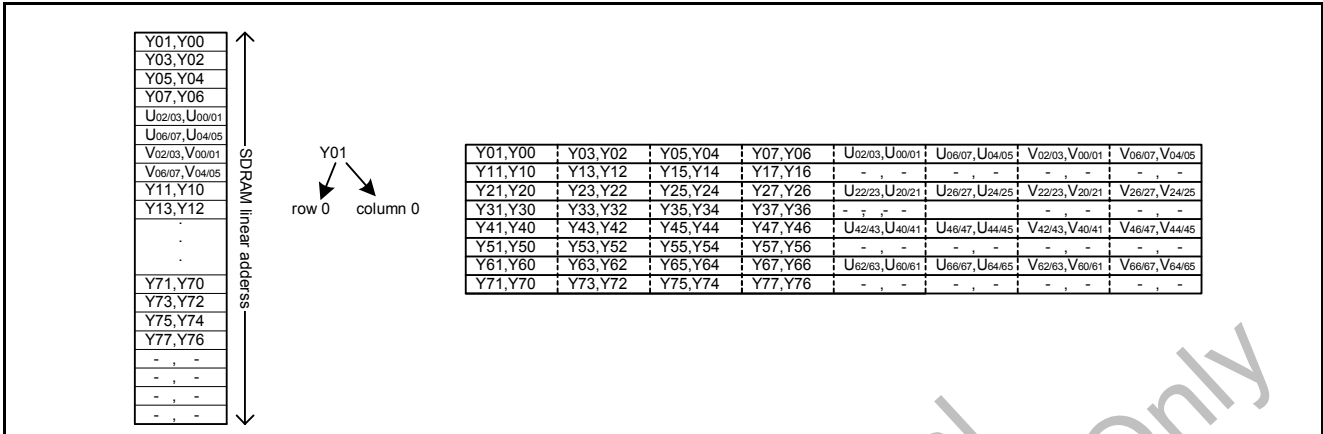


Figure 6-7-3: YUV420 image in the SDRAM buffer

6.7.3. Camera image of Raw data format

SPCA554A supports both 8-bit and 10-bit Bayer-pattern output sensors. If either 8-bit or 10-bit is captured, each pixel is stored in 10-bit memory. For 8-bit capture, two stuffing 0's are attached

in each pixel. 8 pixels of raw data (8 x 10 bits) are packed into 5 words (5 x 16 bits). The following figure shows an example of 64 (8x8) pixels of raw data image in the SDRAM buffer.

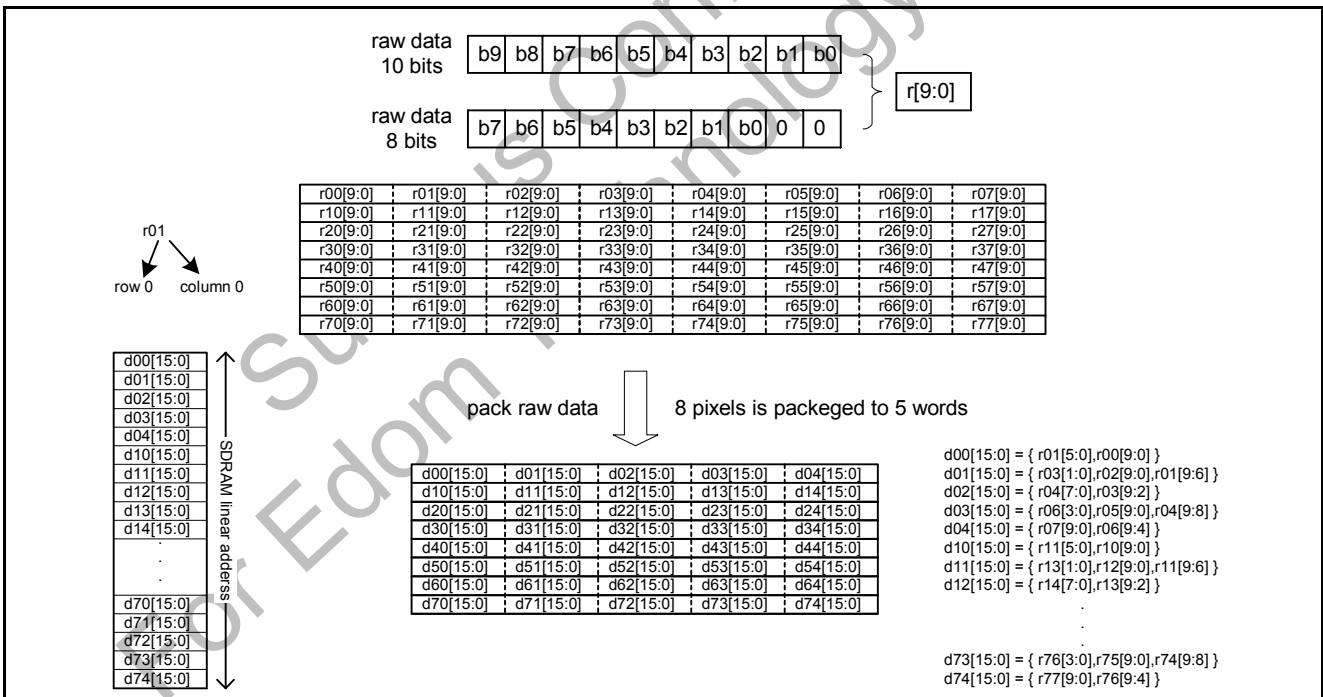


Figure 6-7-4: Raw data in the SDRAM buffer

6.8. Camera Image Controller

The image controller supports a variety of camera image operations, including image scaling up/down, rotation, mirror, BitBlt, bad-pixel concealment and YUV420-to-YUV422 conversion. It reads original image from source image buffer and writes

transformed image to destination image buffer. The camera image controller works only on still image and YUV camera image. It can be used to edit the still image after capture or playback.

6.8.1. Scaling

The scaling function in the camera image controller performs up-scaling or downscaling on a still image. It can be used after still image capture or playback (refer to section 6.1.2 and 6.1.3). Both YUV422 and YUV420 camera image can be scaled. The scaling factor is controlled by a 16-bit fractional number. For image scaling down, the scale factor represents the target-to-source image size ratio. For image scaling up, the scaling factor represents the source-to-target image size ratio.

The image width and height of source and destination images must be multiples of 8 pixels.

6.8.2. Rotation and mirror

In addition to original mode, the SPCA554 supports 7 different types of rotation and mirror. Totally, 7 modes are feasible. This function works on still camera image only. It can be used in the still image capture mode or the playback mode (Refer to section 6.1.2 and 6.1.3).

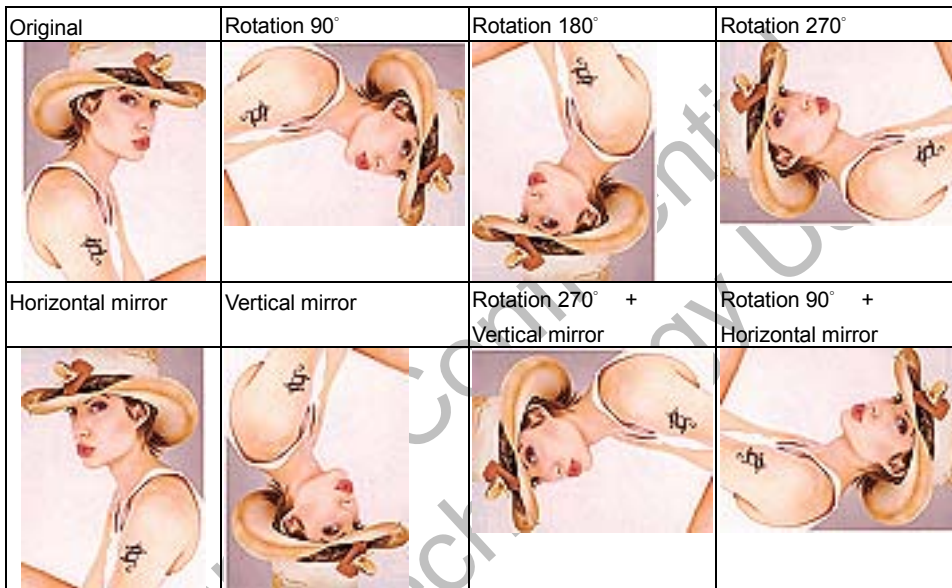


Figure 6-8-1: Rotation and flipping in image controller

6.8.3. BitBlit of camera image data

In this function, a rectangular region is copied from source camera image and pasted to destination camera image. The source width and height of the source image buffer and destination buffer

are programmable. Also the offsets and size of the copied region are programmable.

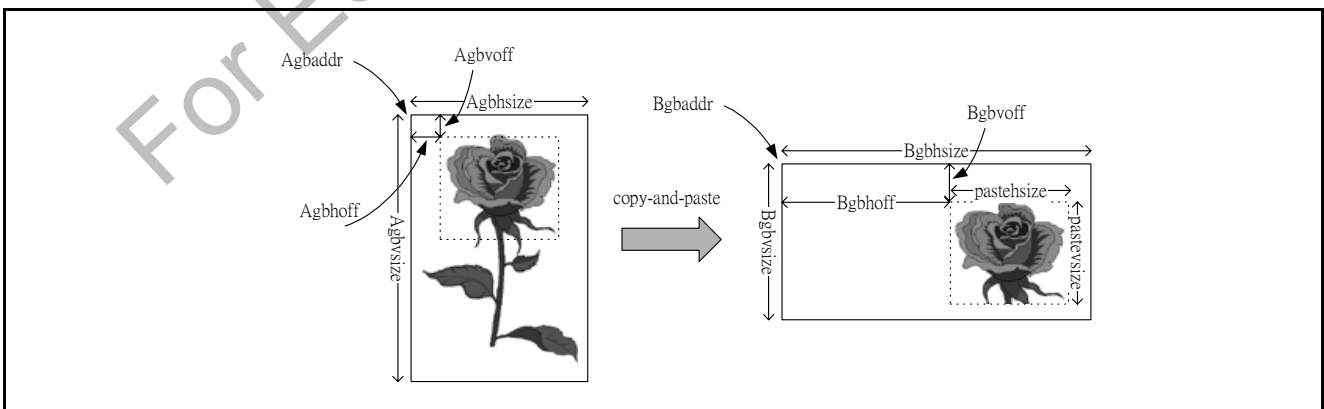


Figure 6-8-2: copy and paste

Note: The values of Agbhoff, Bgbhoff and pastehsz must be even numbers.

6.8.4. Bad Pixel Concealment

This function performs bad pixel concealment based on the bad pixel coordinates stored in SDRAM. 8 neighboring pixels are referenced for interpolating the value of the target pixel. And the interpolated value is written back to SDRAM to overwrite the bad pixel data. The bad pixel concealment function in the image controller is designed for still image capture. It can not be used in real time video, such as in the preview mode.

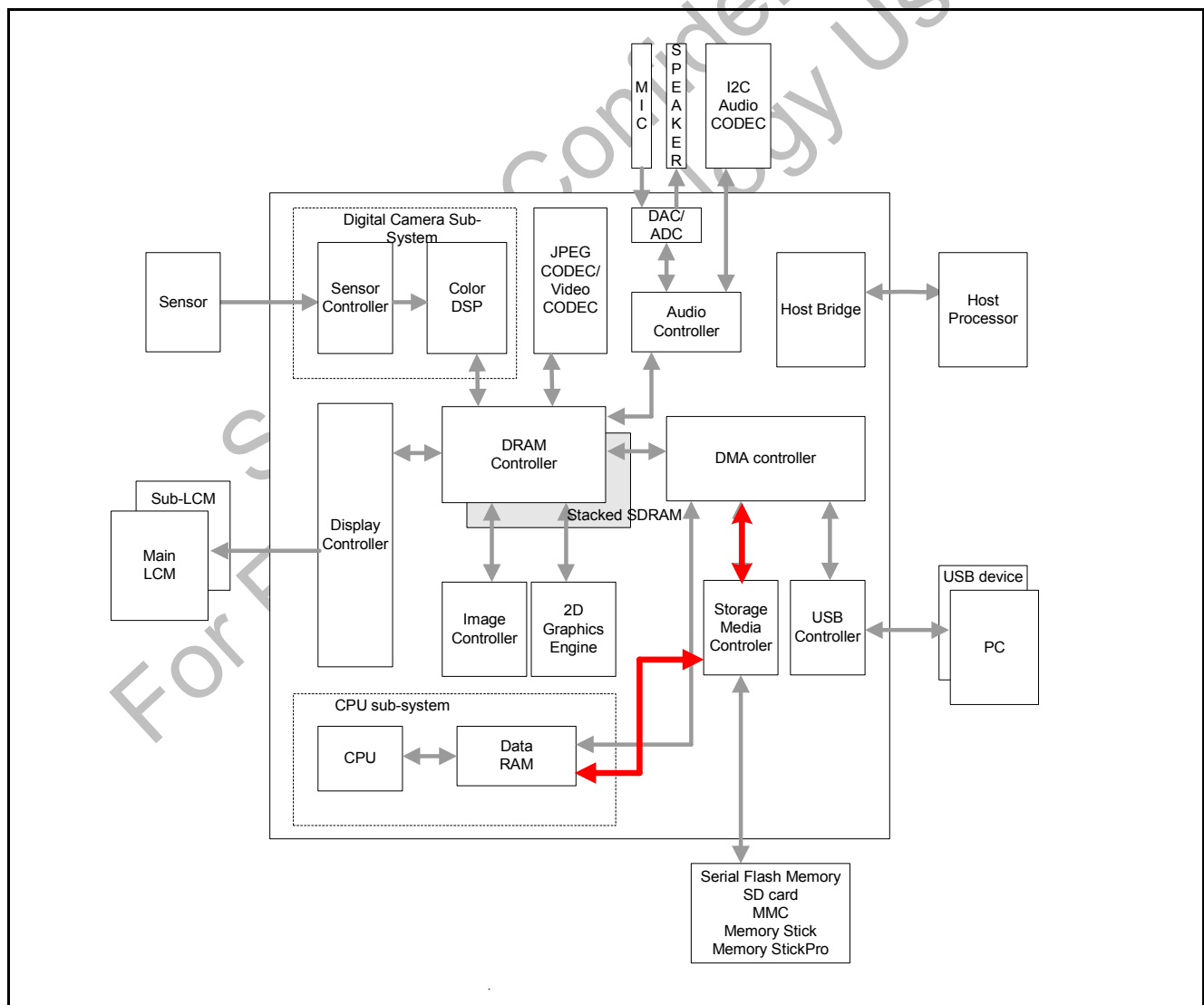
6.8.5. YUV420-to-YUV422 conversion

This function is used to convert the image from the YUV420 format to the YUV422 format. The display controller can only display an image in the YUV422 format. If a YUV420 image is to be play-back, it must be converted to YUV422 format first.

6.9. Storage Media Controller

Storage media is necessary in a mobile application for audio data, video data, and still image storage. SPCA554A supports SD (Security Digital) cards, MS (Memory Stick), MS-pro and SPI-type flash memory. The serial flash can also be used to store the program codes necessary for SPCA554A on-chip CPU operation. If the application chooses booting SPCA554A from the internal mask ROM, serial flash is necessary.

Combined with the on-chip USB controller, the SPCA554A can realize a standard USB device with MSDC (Mass Storage Media Class) compliance. Being a MSDC USB device, the mobile phone can be used as a portable hot-plug disk when being connected to the USB port of a PC.



The storage media controller interfaces to the internal DMA controller, the on-chip CPU and the external storage media. When accessed by the on-chip CPU, the storage media controller provides the PIO access mode. When accessed by the DMA controller, it is operated in the DMA transfer mode. In PIO mode access, the on-chip CPU can read (or write) storage media byte-by-byte. Although the storage media data bus is only 1-bit or 4-bits wide, the SPCA554A automatically converts bytes to bits (or bytes to nibbles). In the DMA mode, the DMA controller moves data between the storage media and the other clients of the DMA controller. For example, the DMA controller can transfer data between stacked SDRAM and the storage media. The DMA mode is normally used to achieve a high data rate. Through the DMA controller, the storage controller can move JPEG files or MPEG4 files from external storage media to the stacked SDRAM for playback. It can also move the captured image (or video) to the external storage media. The same data flow applies to audio applications.

6.10. USB Contoller

The USB interface enables the SPCA554A to communicate with a PC. Data can be uploaded to a PC or downloaded from PCs via the USB bus. The ISP (In-System-Programming) function can also take advantage of the USB bus, which allows a new firmware code to be downloaded to the SPCA554A and then programmed to the external serial flash memory. Another benefit of the USB interface is it allows the SPCA554A to be used as a PC-camera. The sensor module calibration in the mass production stage can also be done via a PC. The SPCA554A USB controller interfaces to the internal DMA controller only. All the data transfers on the USB bus is through the DMA controller.

By programming SPCA554A internal registers, the built-in USB controller can easily be configured in three different modes; USB function mode, USB host mode and OTG mode. The USB host mode and the USB function mode share an on-chip USB 1.1 full speed transceiver. Application of the USB OTG mode requires an external OTG transceiver. With an OTG transceiver, SPCA554A turns into an USB OTG dual-role device. The default mode after power-on is the USB device mode.

USB Device mode

SPCA554A acts as a slave to a USB host (PC). From the PC's viewpoint, the SPCA554A supports four endpoints; CONTROL endpoint, BULK-IN endpoint, BULK-OUT endpoint and an INTERRUPT-IN endpoint. The CONTROL endpoint processes all the commands from a PC host. The PC-side software can access all the internal registers through the control endpoint. The BULK-IN endpoint is responsible for transferring data from the SPCA554A to the PC. The BULK-OUT endpoint is responsible for transferring data from the PC to the SPCA554A. The INTERRUPT-IN endpoint reports SPCA554A's internal events to the PC. With all these four endpoints, the USB MSDC (Mass Storage Class Device) can be realized. SPCA554A kernel firmware implements fully compatible MSDC APIs, which make the SPCA554A a hot-plug disk to a PC. When plugged into a PC USB socket, the PC can access files in the storage media. The files on the base-band processor side can also be transferred to the PC via the USB interface. The maximum packet size of all endpoints is 64-bytes.

Configuration	Interface	Alternate setting	Logical Endpoint	Type	Maximum packet size	Function
NA	NA	NA	EP0	Default Control	64	Process both standard and vendor commands
1	1	0	EP2	BULK-IN	64	Transmits data to the host
			EP3	BULK-OUT	64	Receives data from the host
			EP4	INTERRUPT-IN	64	Transmits interrupt to the host

Table 6-10-1: SPCA554A's USB configuration

To be able to access internal registers of SPCA554A, the following USB vendor commands are defined. These vendor commands are widely used in SPCA554A's development software. The *wIndex* field denotes the address of the register to be accessed. In response to the *Read* vendor command, SPCA554A returns

1-byte of the register value. In the *Write* command, the value to be written to the register is specified in the low byte of the *wValue* field. The internal register accesses vendor commands described here enabling the user to debug an SPCA554A based application via a PC.

Command	bmReqType	bRequest	wValue	wIndex	wLength
Read	0XC0	0X00	Reserved	Address	1
Write	0X40	0X00	High byte: reserved Low byte: write value	Address	0

Table 6-10-2: SPCA554A's USB vendor command

USB Host mode and OTG mode

Being used as a USB host, SPCA554A can access other USB devices such as a USB disk or USB printer. The SPCA554A can easily access a USB disk that conforms to the USB MSDC standard. However, proprietary firmware should be implemented to enable communication between the SPCA554 and a USB printer. The USB printer normally has a proprietary data transfer protocol. The USB dual-role device is achieved by connecting an external OTG transceiver to the SPCA554A. The USB cable signals, including the ID pin, are connected to the transceiver. By reading the register in the OTG transceiver, the SPCA554A will know when to operate in the host mode or in the slave mode.

6.11. Audio Controller

SPCA554A supports both analog and digital interfaces to implement the audio function in a mobile device. The audio data is transmitted through these interfaces and normally requires further data processing by the internal CPU. For example, while recording audio data from an external microphone, the SPCA554A is able to perform AMR, AAC-LC, G.726 or MP3 encoding. On the other hand, the MP3, AMR, AAC-LC and G.726 decoding is supported in audio playback. The following diagram shows the sample rate and bit rate that SPCA554A supports:

	Encode	Decode
MP3	MPEG-1 Layer3 32~320 Kbps	MPEG1/2/2.5 8k~48KHz, 32-320 Kbps
AMR	4.75~12.2 Kbps	8KHz, 4.75~12.2 Kbps
AAC-LC	48K bps/ch	8-96 KHz, 48~96K bps/ch
G.726	16~40 K bps	8KHz, 16~40 K bps

Table 6-11-1: SPCA554A supported sample rates and bit rates

The SPCA554A audio controller serves as a bridge between the external audio CODEC and the audio buffer located in the stacked SDRAM. In audio recording, the audio data is sent to the audio buffer directly. Then the CPU reads the data, compresses the data and writes the result back to SDRAM. The compressed audio data is sent to the external storage media via the DMA controller. It can also be read by the host processor through the host bridge.

In the audio playback mode, source audio data is sent from the host processor or moved from the external storage media. The source data is sent to the stacked SDRAM first, then parsed and decompressed by the internal CPU. The decompressed audio data stream is sent back to the stacked SDRAM, waiting for the audio controller to read. The audio controller reads the digital audio data from SDRAM and sends them to an external CODEC.

ADC and DAC

Alternatively, the audio controller can interface to the on-chip audio ADC and DAC. SPCA554A has built in 12-bit ADC as an

analog interface. The ADC is connected to an external microphone for mono audio functions. There is also a 10-bit on-chip DAC for audio playback. The DAC is to be connected to an external speaker. Both the on-chip ADC and DAC can operate at the most commonly used audio sampling rates including 8KHz, 11.025KHz, 12KHz, 16KHz, 22.05KHz, 24KHz, 32KHz, 44.1KHz and 48KHz. The sampling rates of the on-chip ADC and DAC can be programmed independently. The following diagram shows the reference circuit of the built-in ADC and DAC.

I²S

The SPCA554A digital audio interface supports external DAC, ADC, or bi-directional CODEC, using I²S protocol. All of these external components communicate with SPCA554A through an I²S link. There are, at most, five signals that constitute the I²S link; AGPIO[0], AGPIO[1], AGPIO[2], GPIO[5] and GPIO[6]. The I²S link can be configured in the master mode or the slave mode. In the master mode, the clocks are driven by SPCA554A and all the commonly used sampling rates, including 8KHz, 11.025KHz, 12KHz, 16KHz, 22.05KHz, 24KHz, 32KHz, 44.1KHz and 48KHz,

can be achieved by programming SPCA554A internal registers. In the slave mode, the clocks are driven by the external CODEC and thus the operating sampling rate is determined by the external

CODEC. The data transmission sequence is MSB-first. The following diagram shows the timing of the I²S protocol.

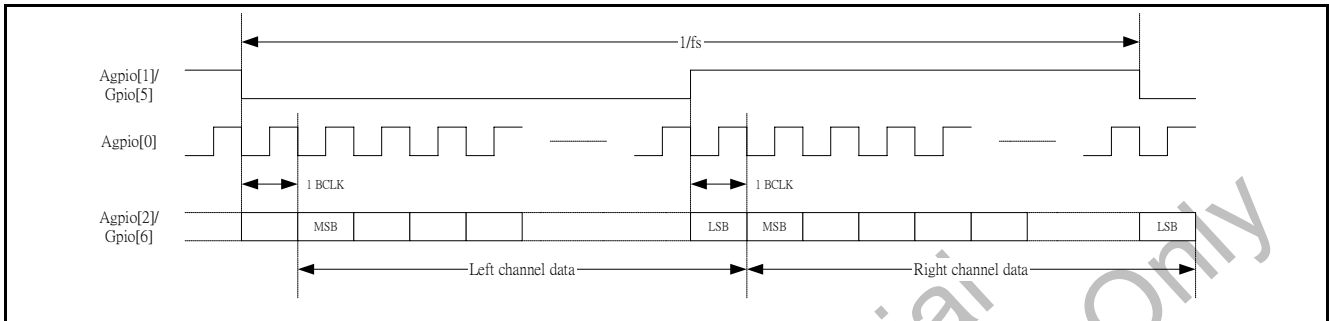


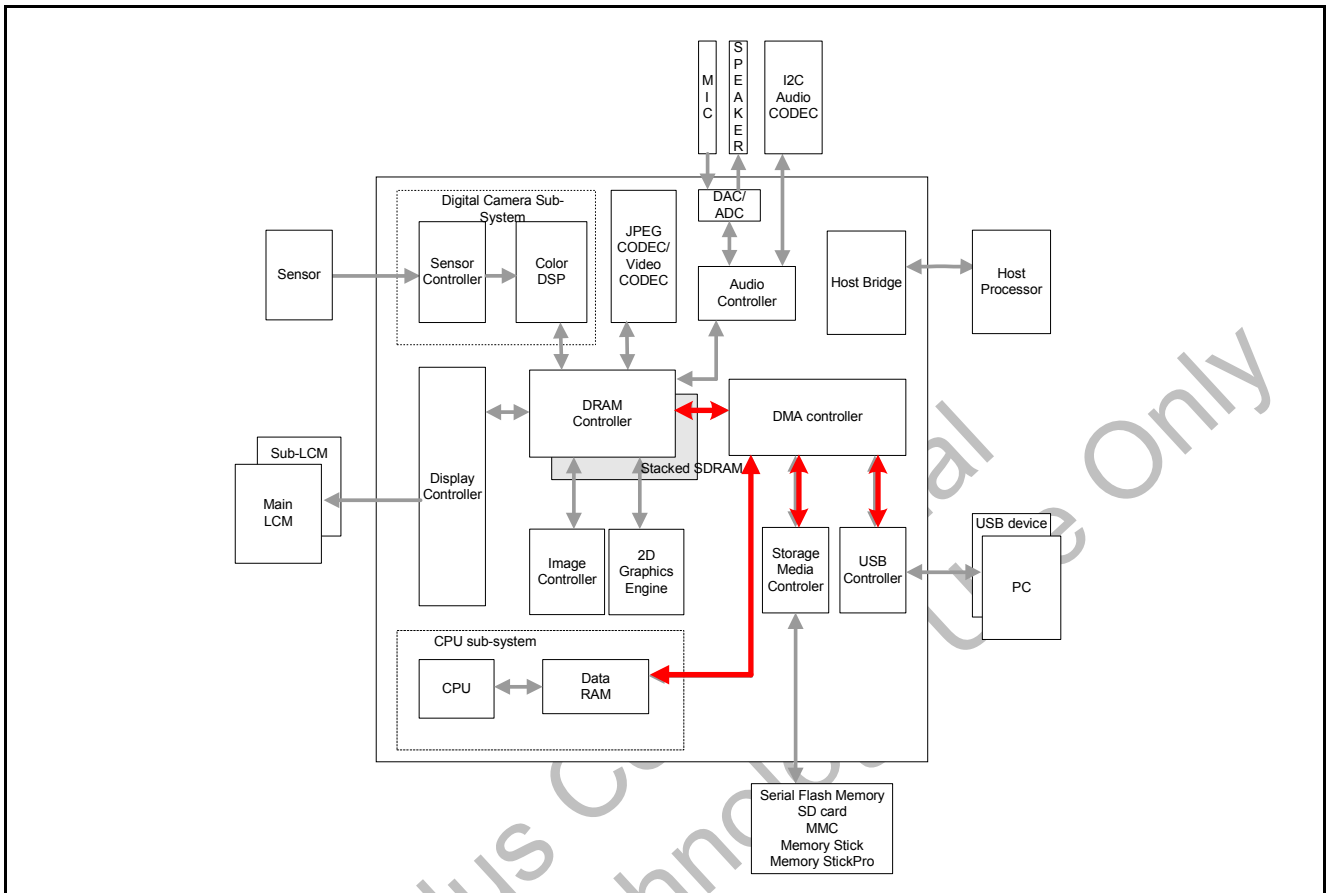
Fig 6-11-1: I²S protocol

6.12. DMA Controller

The DMA controller in the SPCA554A is in charge of moving data between different internal modules and external devices. The DMA function releases the embedded CPU from moving data using the PIO (programmable IO) method. It is very useful for moving the data transfer task to background execution so that CPU power can be reserved for computation-intensive operations. For example, after the SPCA554A captures a picture, the captured picture can be sent to an external storage card using the DMA controller. In the meantime, the CPU can keep working on AE/AWB for camera preview.

The DMA controller of the SPCA554A serves four DMA clients; external stacked SDRAM, data RAM in the CPU sub-system, USB controller and storage media controller. All of these clients can be utilized as a DMA source (data provider) and destination (data consumer). The programming model of the DMA controller is simply programming the source and destination address before triggering the DMA data transfer.

Sunplus Confidential For Edom Technology Use Only



The stacked low power SDRAM is the major temporary buffer in SPCA554A applications. All program code, captured image and video/audio CODEC buffers are allocated in the stacked SDRAM. The content in SDRAM demand massive data transfers to (and from) other modules.

The data RAM of the CPU sub-system can be accessed by the CPU-core at high-speed. Theoretically, all the data processing can be done by directly accessing stacked low power SDRAM. However, moving data to the DRAM of the CPU module in advance can often produce higher performance. Accessing data in DRAM is much faster than accessing data in stacked SDRAM and it consumes less power. For example, audio AMR CODEC, FAT cluster search, and data ECC correction can all be done inside DRAM of the CPU module.

The USB module in SPCA554A contains a 64-byte buffer, which can be used to exchange data between the USB and other internal modules. To upload data to a PC via the USB bus, the data is sent to the 64-byte buffer first. Then the USB controller sends the data by USB BULK-IN packets to the PC. To download data from the PC to the SPCA554A, the USB controller obtains data from the PC via the BULK-OUT pipe and stores data in the 64-byte buffer. The data, in turn, is sent to other modules via DMA transfer.

Storage media is necessary in a mobile application for audio data, video data, and still image storage. The content in the storage media requires massive data transfers between the storage media and other internal modules. For example, the DMA controller can move data between stacked SDRAM and external storage media.

The SPCA554A DMA controller has the following features:

- Three DMA channels.
- Page mode transfer with padding-bytes
- FAT acceleration
- 16-bit data transfer with automatic data bus width adjustment
- Provide auto-increment/decrement address and fixed address DMA transfer

6.12.1. General purpose DMA channel

The general purpose DMA channel moves data between all the clients of the DMA controller. The following table lists the most commonly used DMA data transfer paths:

Data source	Data destination	Function description
Stacked SDRAM	CPU data memory	--Audio data to be accessed by the CPU at high speed, such as AMR encoding and decoding. --High-speed FAT empty cluster search.
	Storage media	--Picture, audio, and video data store.
	USB	--Upload data from the SDRAM to the PC
CPU data memory	Stacked SDRAM	--No specific application
	Storage media	--Stores the files headers of images, audio and video into the storage media.
	USB	--No specific application
Storage media	Stacked SDRAM	--Restores image, audio and video files for decoding and playback. --Mirrors the FAT of the storage media into the stacked SDRAM at power-on stage.
	CPU data memory	--Part of the image, audio, video file decoding procedure. The file headers are uploaded to CPU data RAM for parsing.
	USB	--Uploads image, audio and video files to the PC.
USB	Stacked SDRAM	--ISP (in-system programming) is used to update firmware in the the storage media. Downloading program code from the PC to stacked low power SDRAM is part of the ISP procedure.
	CPU data memory	--No specific application
	Storage media	--Downloads image, audio and video files from the PC to the storage media.

Table 6-12-1: Clients of the general purpose DMA channel

The data size of a DMA transfer is programmable. The minimal data size of a DMA transfer is 2 bytes, while the maximum is 128K bytes. Most of the storage cards, like MMC and SD cards, are accessed on a block (page) bases. If the programmed DMA data size is not aligned to the page size of the storage media, the storage media might not be able to finish a complete access cycle. SPCA554A can automatically pad 0's to the last data in a DMA transfer, so that each page-write can be completed. In page-read, SPCA554A automatically issues dummy read cycles to the storage media when the programmed DMA data size is less than the storage media's page size. By doing the dummy read, the storage media receives a complete page-read cycle. The CPU DMA completion interrupt is activated only after the dummy read

or padding-write cycles are completed.

6.12.2. Secondary DMA channel

The efficiency of moving data between the storage media and other modules is often limited by the access time of the storage media. To maximize the utilization of the general purpose DMA channel, the SPCA554A has allocated a dedicated DMA channel between stacked SDRAM and the storage media. The secondary DMA channel can move data from stacked SDRAM to the storage media, and vice versa. While this secondary DAM channel is working, the general purpose DMA channel can work on other DMA transfers.

Even though the secondary DMA channel can work with the general-purpose DMA channel at the same time, there is still a limitation due to the block-access nature of storage media. Storage media cannot be assigned as the client of the general-purpose DMA channel when a secondary DMA channel is activated. Otherwise, the data in the storage media is corrupted.

6.12.3. DRAM-to-DRAM DMA channel

The SPCA554A supports a DRAM-to-DRAM DMA function. This function performs data transfer from source address to destination address in stacked DRAM. Source address, destination address and transfer byte count must be programmed before the transfer. The transfer is byte alignment, and the maximal transfer size is 4Kbyte. This DMA transfer sends an interrupt to the on-chip CPU upon DMA completion.

6.12.4. FAT acceleration

To comply with USB MSDC (Mass Storage Device Class) specifications, the FAT should be implemented in SPCA554A firmware. However, it takes a lot of time to construct/parse FAT by firmware, especially when the firmware is trying to search an empty cluster for new data storage. SPCA554A has a built-in

accelerator to speed up empty cluster search. The accelerator works seamlessly with the DMA controller. The accelerator works with the general-purpose DMA channel only.

In a FAT system, the empty cluster is normally represented by a special code. The SPCA554A FAT accelerator can report the number and location of the special code after a DMA transfer. This function saves the CPU from inspecting each byte of the FAT contents and will drastically improve FAT performance. There are two FAT acceleration operational modes. The first operating mode simply reports the result of code search after entire DMA transfer. The second operational mode pauses the DMA transfer whenever a matched code is encountered. And an interrupt is issued to the CPU for firmware intervention. The firmware might resume the DMA transfer after inspecting the location of the matched code.

SPCA554A supports 12-bits, 16-bits and 32-bits code search, which correspond to FAT12, FAT16 and FAT32 systems respectively. The code to be searched is programmable. The following is an example of the pattern-search.

DATA transfer sequence	0	1	2	3	4	5	6	7	8	9	10	11	12
DATA	00h	04h	0Fh	01h	04h	05h	01h	0Fh	08h	01h	0Fh	0Bh	0Ch

Assume a 16-bit code of 0x0F01 is to be searched and the data sequence is low-byte first, the FAT accelerator will report the first code match at location 7 and the matched count is 2. Byte 6 and Byte7 form a matched pair while byte 9 and byte 10 form another. Note that byte 2 and byte 3 do not match the pattern since the sequence is reversed.

6.13. 2D Graphics Engine

The 2D graphics engine in SPCA554A supports four main functions; BitBLT, Line Drawing, Gradient Fill and Sprite. It accelerates the graphics application in a mobile phone, such as MMI implementation and game playing. The 2D graphics engine

requires three working buffers, two destination buffers for display and one source buffer for temporary storage of 2D graphics images/icons. The dual-display buffer architecture allows SPCA554A to display the contents of one buffer while updating the contents of the other buffer. Also, with the assistance of the display controller, the graphics image can be overlaid with the camera image to create photo stickers. The overall 2D graphics engine operation is illustrated below (Figure 6-13-1). Note that the display buffers are also referred to as destination buffers because the 2D graphics engine always output data to the display buffers.

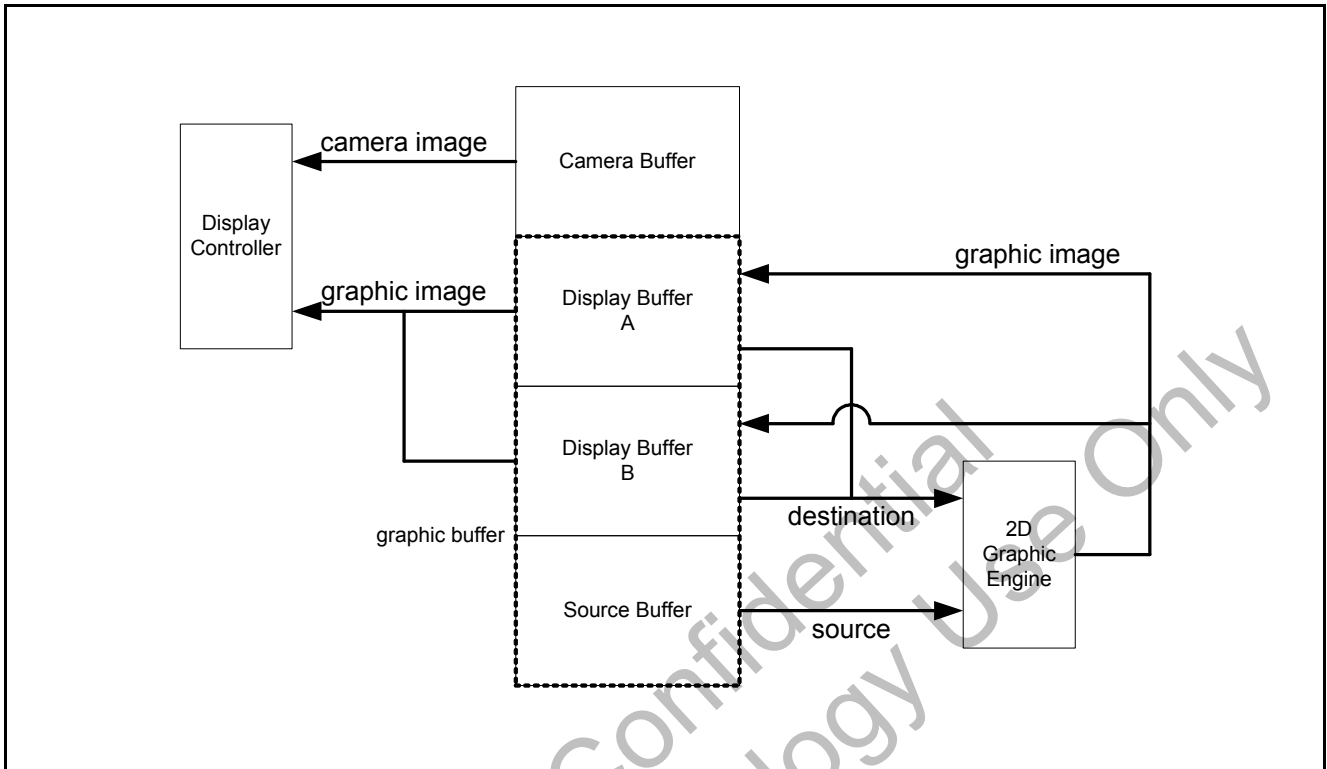


Figure 6-13-1: The operation of the graphic engine

6.13.1. Resolution and color system

The SPCA554A 2D graphics engine always outputs on 16-bpp (bit-per-pixel) graphics objects. That means the display buffers always holds graphics images utilizing a 16-bit color depth. The

maximum resolution the 2D graphics engine supports is 240x320. The following table shows the commonly used display resolutions and display buffer requirements for the graphic engine.

Display Resolution (Pixels*pixels)	240x320	176x220	128x160	120x160
Display Buffer Requirements (Per display buffer)	150 KBytes	76 KBytes	40 KBytes	38 KBytes

Table 6-13-1: Display resolution and display buffer requirements

Destination buffer color format

Table 6-13-2 shows the color format of the display buffer. The SPCA554A supports both RGB565 and BGR565 formats. The format must conform to the source buffer color format.

R= red color channel

G= green color channel

B= blue color channel

Color depth	Format	Bits	Description
16-bpp	RGB565	[15:0]	RRRR_RGGG_GGGB_BBBB
	BGR565	[15:0]	BBBB_BGGG_GGGR_RRRR

Table 6-13-2: Display buffer color format

Source buffer color format

In the source buffer color format, SPCA554A supports 16-bpp, 12-bpp, 8-bpp and 4-bpp. 16-bpp and 12-bpp formats can be in RGB or BGR sequence. 8-bpp and 4-bpp are indexed color formats.

- K= Opaque/Transparency switch bit
- A= per-pixel alpha channel
- R= red color channel
- G= green color channel
- B= blue color channel
- P= palette index

Table 6-13-3 shows the 16-bpp color formats of the source image. A total of four types of 16-bpp color formats are supported; RGB565, BGR565, KRGB1555 and KBGR1555. The K-bit is specially designed for overlay with the camera image. It is checked by the display controller to determine whether or not to

blend the graphics image with the camera image. A pixel with K-bit set to one will be overlaid on the camera image without any transparent effect. A pixel with K-bit set to zero will be blended with the camera image to create a semi-transparent effect. The blending level is controlled by the display controller.

Color depth	Format	Bits	Description
16-bpp	RGB565	[15:0]	RRRR_RGGG_GGGB_BBBB
		[15:0]	KRRR_RRGG_GGGB_BBBB
	BGR565	[15:0]	BBBB_BGGG_GGGR_RRRR
		[15:0]	KBBB_BBGG_GGGR_RRRR

Table 6-13-3: 16-bpp Source buffer color format

Table 6-13-4 shows the 16-bpp ARGB formats of the source buffer. ARGB is a per-pixel alpha format, meaning each pixel has its own alpha value. Both ARGB1555 and ARGB4444 formats are supported. ARGB1555 represents the RGB555 format with one

extra alpha bit for each pixel. ARGB4444 represents the RGB444 format with 4 alpha bits for each pixel. The alpha value here is referenced by the 2D graphics engine itself when performing various 2D operations.

Color depth	Format	Bits	Description
16-bpp	ARGB1555	[15:0]	ARRR_RRGG_GGGB_BBBB
	ARGB4444	[15:0]	AAAA_RRRR_GGGG_BBBB

Table 6-13-4: 16-bpp Source pixel's ARGB format

Table 6-13-5 shows the 12-bpp RGB formats of the source buffer:

Color depth	Format	Bits	Description
12-bpp	RGB444	[11:0]	RRRR_GGGG_BBBB
	BGR444	[11:0]	BBBB_GGGG_RRRR

Table 6-13-5: 12-bpp source pixel RGB format

Table 6-13-6 shows the 8-bpp Index format of the source buffer. The index color mode is useful in GIF file playback. SPCA554A has a programmable CLUT (color look up table) with 16-bit color for each entry

Color depth	Format	Bits	Description
8-bpp	256-color	[7:0]	PPPP_PPPP

Table 6-13-6: 8-bpp source pixel index format

Table 6-13-7 shows the 4-bpp Index format of the source buffer. The index color mode is useful in menu icon playback. SPCA554A has a programmable CLUT (color look up table) with 16-bit color for each entry.

Color depth	Format	Bits	Description
4-bpp	16-color	[3:0]	PPPP

Table 6-13-7: 4-bpp source pixel index format

6.13.2. BitBLT

BitBLT is the abbreviation for **BIT BLock Transfer**. This operation performs block to block graphics data transfer. Figure 6-13-2 shows the pipeline of the BitBLT function with associated features. This features includes ROP, alpha blending, stipple mask, rotation, mirror, and clipping.

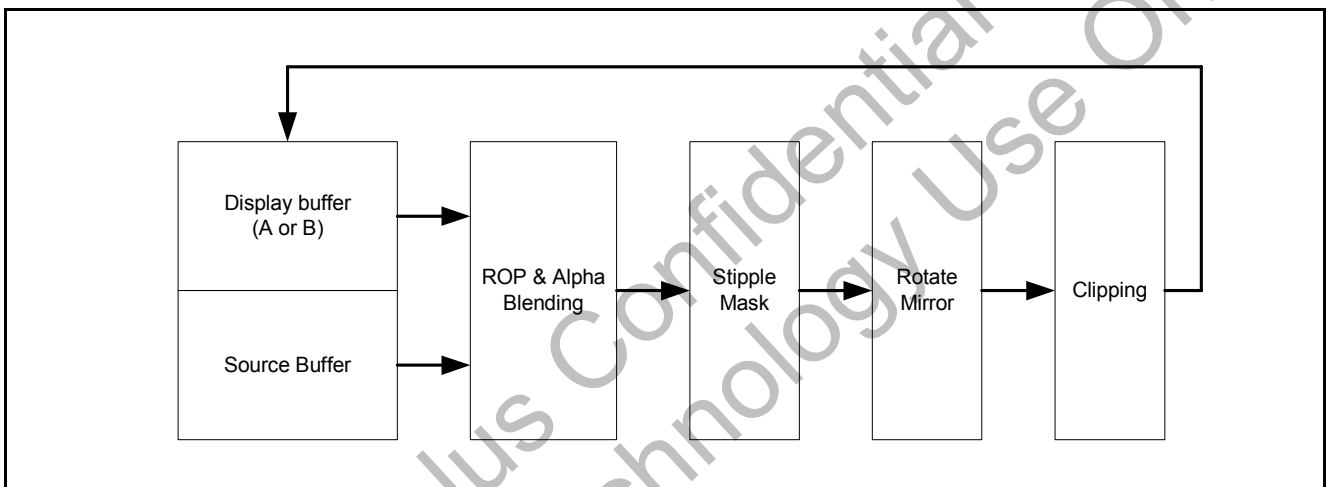


Figure 6-13-2: BitBLT function pipeline

ROP

ROP is the abbreviation for **Raster Operation**. It is a bit-wise Boolean operation with three operands, the source pixel color, the destination pixel color and a predefined pattern color. The pattern color is defined by a 16-bit register (RGB565). An 8-bit ROP register defines the ROP code and it determines which Boolean operation is used for mixing the colors.

Op0 ~ Op7 are defined if the source pixel color, the destination pixel color and the pattern color are used in the Boolean operation. The relationship is listed in the following table:

	Definition	Pattern Color	Source pixel color	Destination pixel color
		PAT[i]	SRC[i]	DST[i]
Op0	0	0	0	0
Op1	DST[i]	0	0	1
Op2	SRC[i]	0	1	0
Op3	SRC[i]&DST[i]	0	1	1
Op4	PAT[i]	1	0	0
Op5	PAT[i]&DST[i]	1	0	1
Op6	PAT[i]&SRC[i]	1	1	0
Op7	PAT[i]&SRC[i]&DST[i]	1	1	1

Table 6-13-8: ROP Boolean operation

Final Boolean operation is defined in the following formula:

$$Dst^{new}[i] = (Op\ 0 \& Rop\ [0]) \mid (Op\ 1 \& Rop\ [1]) \mid (Op\ 2 \& Rop\ [2]) \mid (Op\ 3 \& Rop\ [3]) \mid (Op\ 4 \& Rop\ [4]) \mid (Op\ 5 \& Rop\ [5]) \mid (Op\ 6 \& Rop\ [6]) \mid (Op\ 7 \& Rop\ [7]) \quad \text{where } i = 0, K, \dots, 15$$

Commonly used ROPs are listed in Table 6-13-9 below:

ROP Code	Bit-wise Operation	Description
0x00	0	Destination is set to black
0xFF	1	Destination is set to white
0xCC	S	Source copy to destination
0xF0	P	Pattern fill to destination
0x33	~S	Copy and invert source to destination
0x55	~D	Invert destination
0xEE	S D	Combine source with destination using OR
0x88	S&D	Combine source with destination using AND
0xC0	S&P	Combine source with pattern using AND
0xFB	~S D P	Combine inverted source with destination and pattern using OR
0xFE	S D P	Combine source with destination and pattern using OR

Table 6-13-9: ROP examples

Figure 6-13-3 shows an example of ROP (ROP code 0xEE, source image OR destination image):

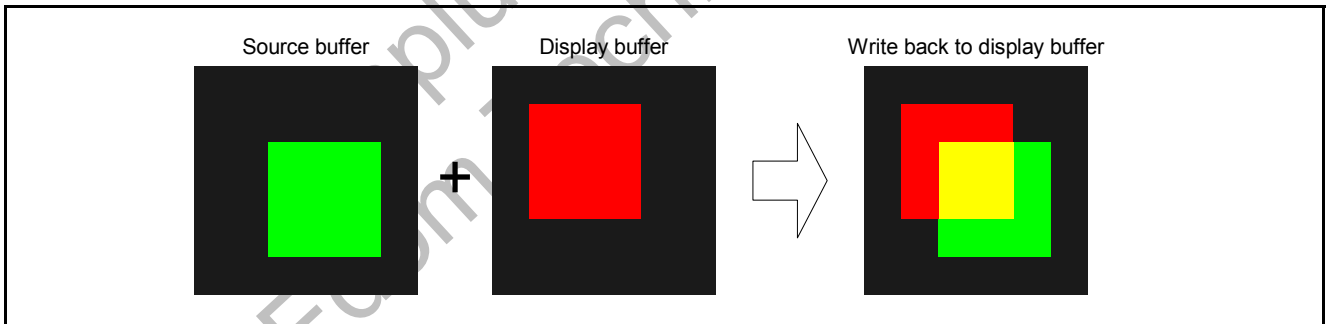


Figure 6-13-3: ROP example 1

Figure 6-13-4 shows another example of ROP (ROP code 0xFE, source image OR destination image OR pattern color).

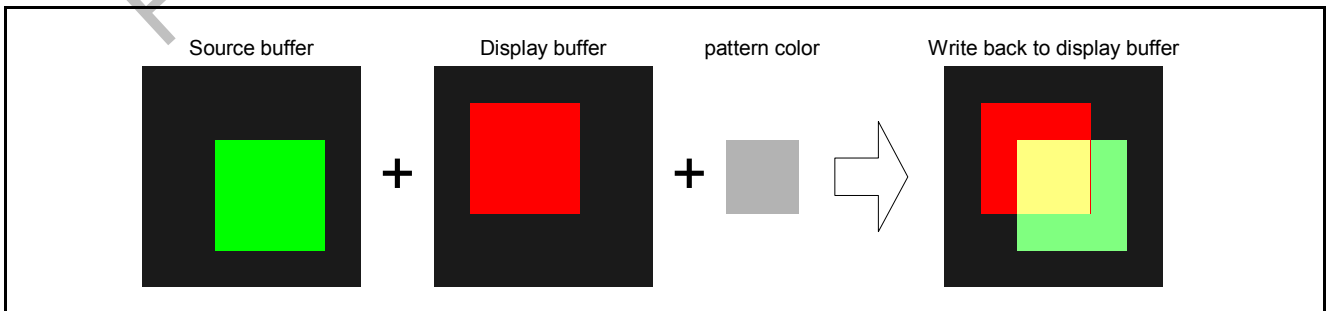


Figure 6-13-4: ROP example 2

Per-pixel Alpha Blending and Constant Alphas Blending

Alpha blending is a color interpolation operation between the source image and the destination image. It uses the alpha value to decide the weightings of the source image and destination image during interpolation. SPCA554A supports two types of alpha blending; per-pixel alpha blending and constant alpha blending. Per-pixel alpha blending allows each pixel to own its own alpha value and constant alpha blending restricts the whole

image to use of a single programmable alpha value. In per-pixel alpha blending applications, the source image must be in the ARGB1555 and ARGB4444 formats.

ARGB1555 has only one alpha value bit. It indicates either the fully opaque quality or fully transparent quality of a pixel. The following equation defines the ARGB1555 blending operation:

$$Dst_{new} = \begin{cases} Src & \text{if } Src_A = 1 \\ Dst & \text{otherwise} \end{cases} \quad \text{where } Src_A \text{ is 1-bit per-pixel alpha}$$

Figure 6-13-5 shows an example of ARGB1555 per-pixel alpha blending:

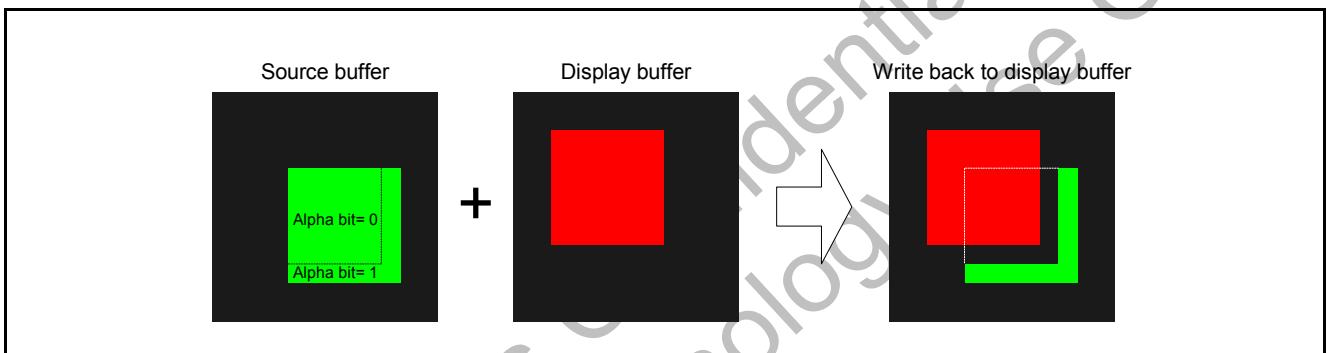


Fig 6-13-5: Per-pixel alpha blending example (ARGB1555)

ARGB4444 has 4 bits of alpha value for each pixel. It is capable of generating sixteen transparent levels. In addition, SPCA554A uses the following equation to calculate transparent levels:

$$Dst_C^{new} = \begin{cases} Src_C & \text{if } Src_A = 15 \\ Dst_C & \text{if } Src_A = 0 \\ \frac{Src_C * Src_A + Dst_C * (15 - Src_A)}{16} & \text{otherwise} \end{cases}$$

where Src_A is a 4-bit alpha value and $C = R, G, B$ is a color channel.

Figure 6-13-6 shows an example of ARGB4444 per-pixel alpha blending:

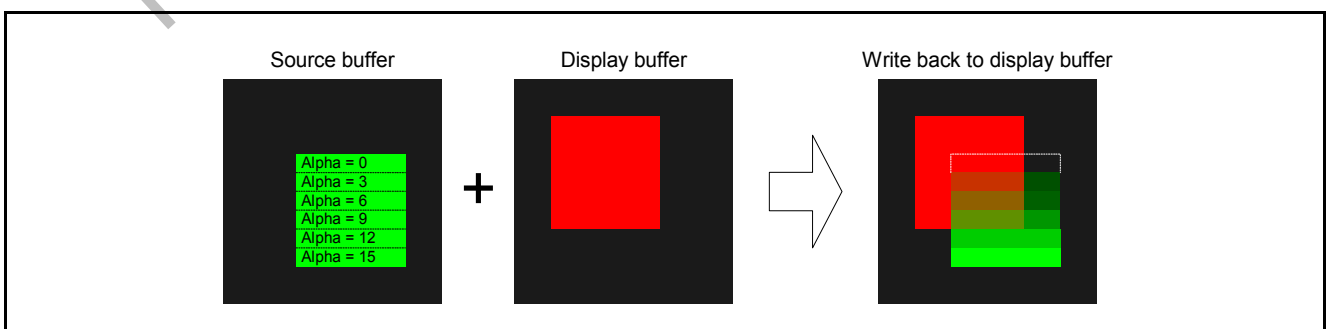


Figure 6-13-6: Per-pixel alpha blending example (ARGB4444)

Constant alpha blending is calculated utilizing the source alpha value and the destination alpha value. The constant alpha value is defined by a 5-bit register, which includes a sign bit and 4-bit magnitude. The constant alpha blending level is calculated using the following formula:

$$Dst_c^{new} = \frac{Src_c * A_c^{Src} + Dst_c * A_c^{Dst}}{16}$$

where A_c^{Src} is the global source alpha register and A_c^{Dst} is the global destination alpha register and $C = R, G, B$ is the color channel.

Figure 6-13-7 shows an example of constant alpha blending:

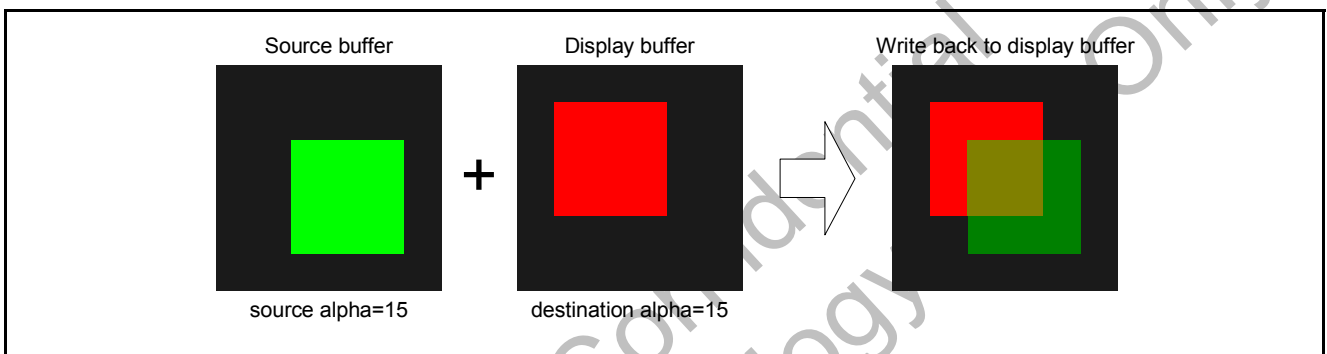


Figure 6-13-7: Constant Alpha example

Stipple Mask

In the SPCA554A 2D graphics engine, an 8x8 mask pattern can be defined to mask out the desired pixels during BitBLT operation. Each pixel in the mask pattern is defined by one bit. If the pixel in the 8x8 mask pattern is equal to 1, the corresponding destination pixel is replaced by a pre-defined pattern color. The calculation is as shown in the following equation:

$$Dst_{new}[x, y] = \begin{cases} Pat & \text{if } Mask_{stipple}[i, j] = 1 \\ Dst & \text{otherwise} \end{cases}$$

where $[x, y]$ is an XY-coordinate and $[i, j]$ is a coordinate rounded by eight.

Figure 6-13-8 shows an operational example of Stipple Mask (ROP code 0xF0, pattern fill).

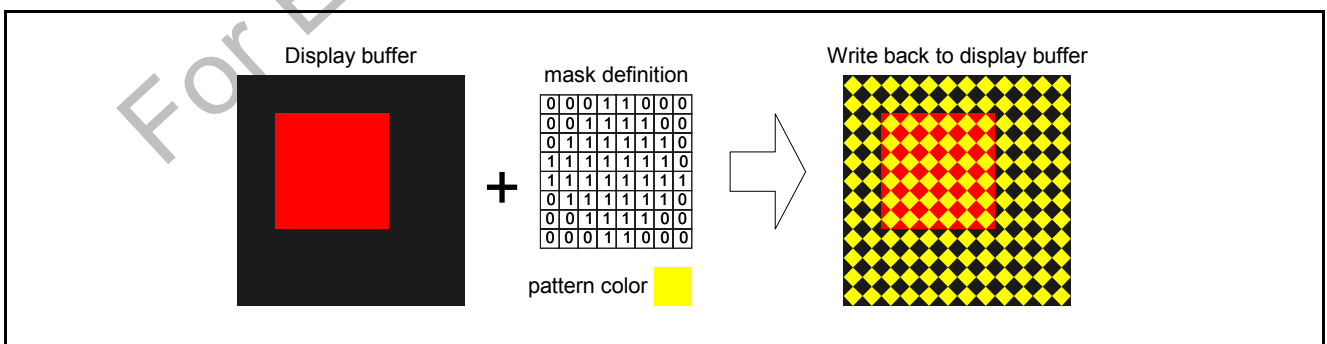


Figure 6-13-8: Stipple Mask operational example

Mirror and Rotation

Eight rotation/mirror modes are supported in the BitBLT function, as shown in the following figure:









	Rotate 0°	Rotate 90°	Rotate 180°	Rotate 270°
None				
Mirror				

Table 6-13-10: Eight kinds of Mirror Rotation Transforms

When a rotation (or mirror) is applied to the source image, SPCA554A allows the user to define a reference point and make the reference point fixed at the destination buffer. Hence, the

reference point acts as the center point of the rotation (mirror) operation. Figure 6-13-9 illustrates the application of rotation:

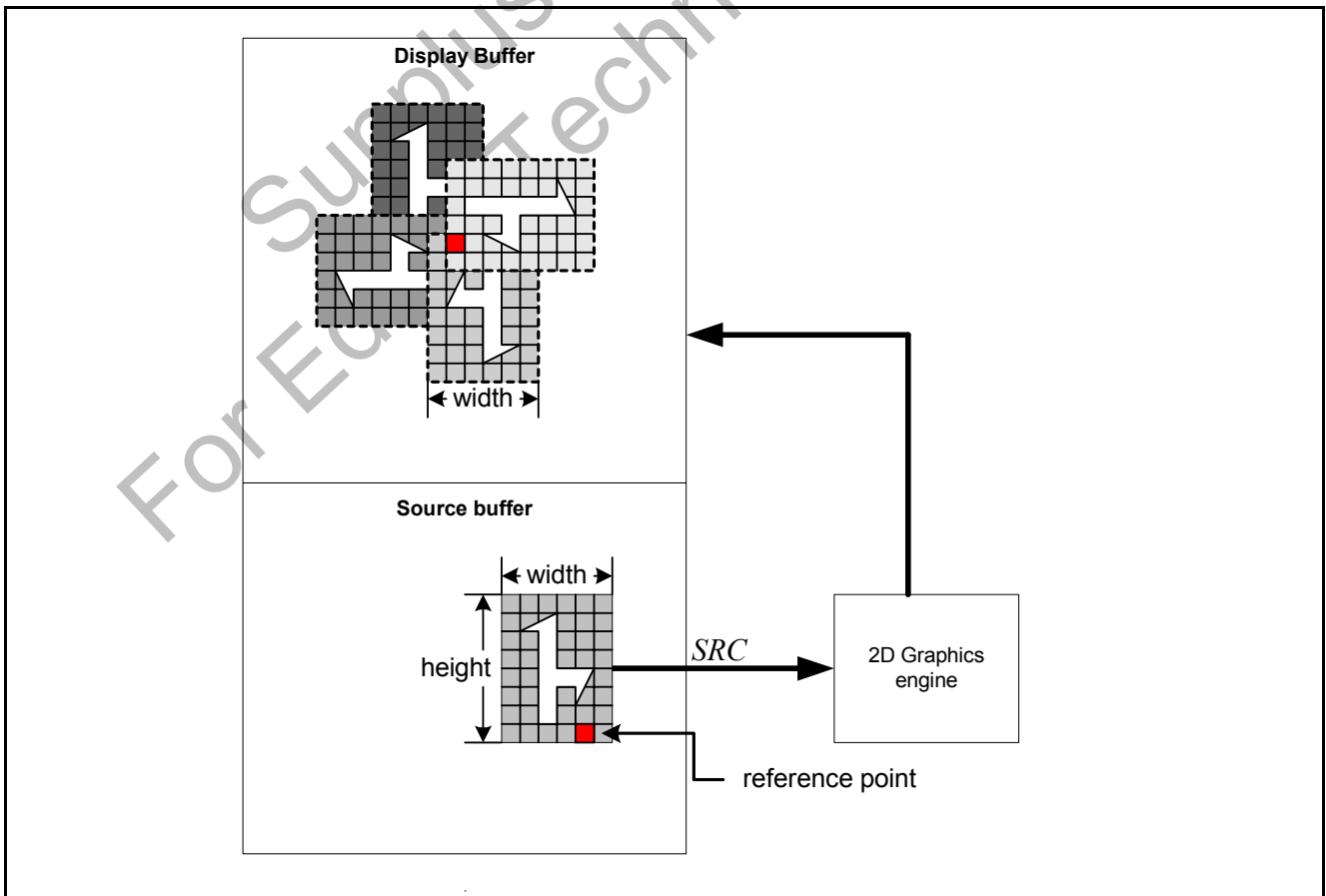


Figure 6-13-9: Rotation/Mirror operational example

Clipping

SPCA554A automatically clips the image outside of the display window area during the BitBLT operation. In addition, users may define a rectangular region, and the 2D graphics engine clips the image outside the window.

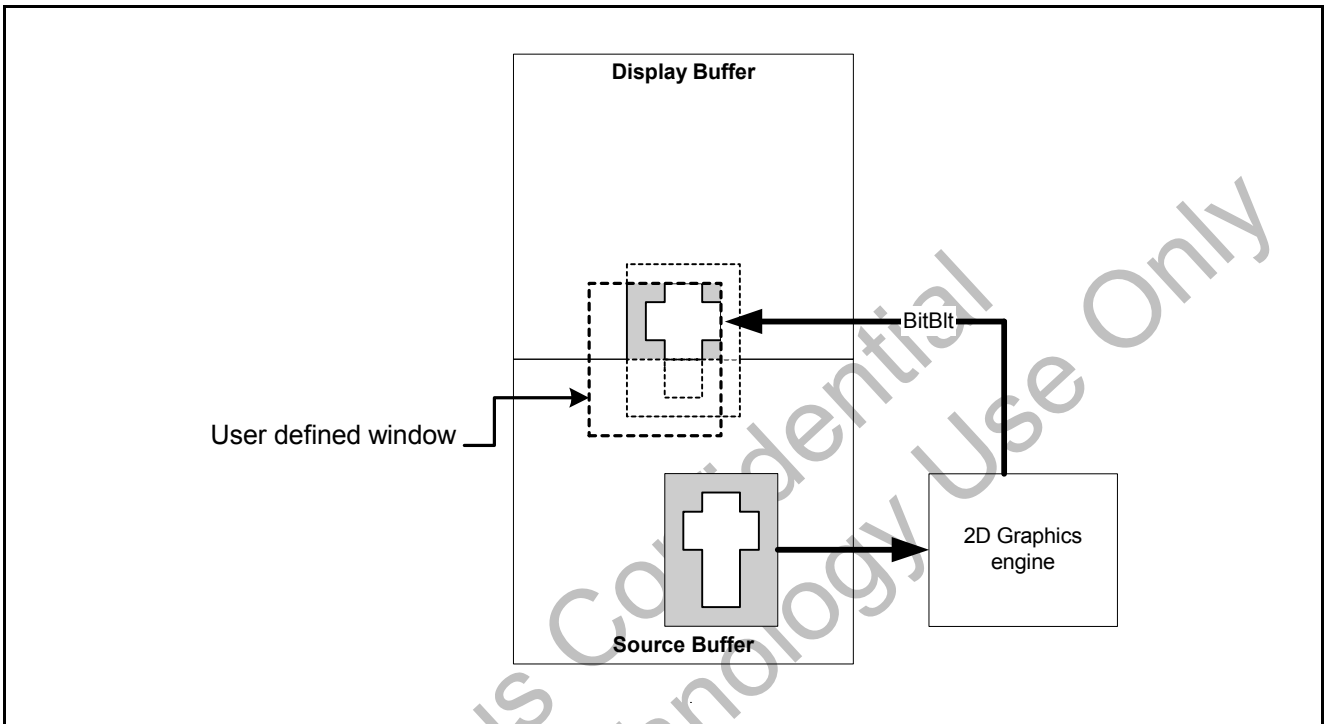


Figure 6-13-10: Rectangle Clipping combined with Window Clipping

6.13.3. Sprite

Sprite function features are similar to those in the BitBLT function and include transparent ROP, ROP, Alpha blending, rotation, mirror, and clipping. The Sprite function is applied to non-rectangular

shape manipulation utilizing the color key approach. Figure 6-13-11 shows the pipeline of the Sprite function with associated features:

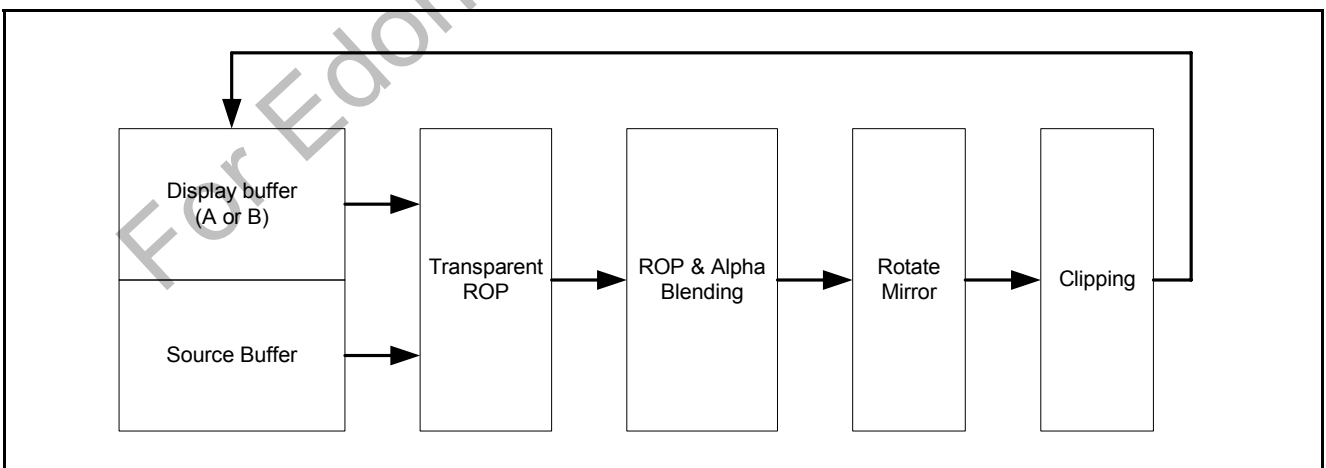


Figure 6-13-11: Sprite function pipeline

Transparent ROP

Transparent ROP (TROP) is accomplished by color comparison of source pixels and destination pixels. There are two sets of color-key boundaries for the source image and destination image respectively. If the pixel color is within the range defined by the registers of color-key boundaries, it passes the color-key comparison. Color comparison of source pixels and destination pixels is processed respectively. Depending on the results of

source pixel and destination pixel color comparison, 16 kinds of TROP, specified by a 4-bit register, are selected to generate various source pixel transparent effects.

Table 6-13-11 shows TROP Boolean operation mapping. Op0 ~ Op3 defines if the results of color comparison of source pixels and destination pixels used in Boolean operations. For example, $Op3 = DstCmp \& SrcCmp$

	Color Comparison	
	DstCmp	SrcCmp
Op0	0 (Fail)	0 (Fail)
Op1	0 (Fail)	1 (Pass)
Op2	1 (Pass)	0 (Fail)
Op3	1 (Pass)	1 (Pass)

Table 6-13-11: TROP Boolean operation

Final transparent operation formula is illustrated below. The operation result $Trapr$ decides whether the pixel of the source image is transparent or not. If $Trapr$ is equal to one, the pixel will be transparent. Applicable equation is shown below:

$$Trapr = \sim [(Op0 \& TRop [3]) | (Op1 \& TRop [2]) | (Op2 \& TRop [1]) | (Op3 \& TRop [0])]$$

Commonly used ROPs are listed in Table 6-13-12.

TROP Code	Transparent Operation	Description
0x0	1	All source pixels are transparent.
0x1	$\sim(DstCmp \& SrcCmp)$	If one of destination or source comparison fails, the source pixel is transparent.
0x3	$\sim DstCmp$	If destination comparison fails, the source pixel is transparent.
0x5	$\sim SrcCmp$	If source comparison fails, the source pixel is transparent.
0x7	$\sim(DstCmp SrcCmp)$	If both of destination and source comparison fail, the source pixel is transparent.
0x8	$DstCmp SrcCmp$	If one of destination or source comparison passes, the source pixel is transparent.
0xA	SrcCmp	If source comparison passes, the source pixel is transparent.
0xF	0	All source pixels are opaque.

Table 6-13-12: TROP example

Figure 6-13-12 shows an example of TROP (ROP code 0xCC, TROP code 0XA, SRC color range 0x07E0 ~ 0x0000). This operation performs the all green background of the source image that is to be transparent utilizing only the source comparison.

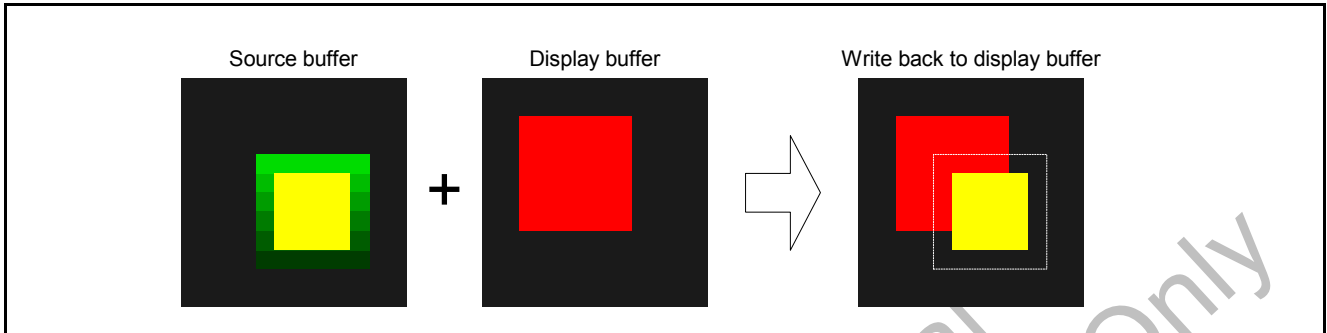


Figure 6-13-12: TROP example 1

Figure 6-13-13 shows another example of TROP (ROP code 0xCC, TROP code 0x8, SRC color range 0x07E0~0x0000, DST color range 0xF800 ~ 0xF800). This operation can be separated

into two steps. The first step is the same as example 1. The second step is to paste source pixels to the red area of the destination utilizing the destination comparison.

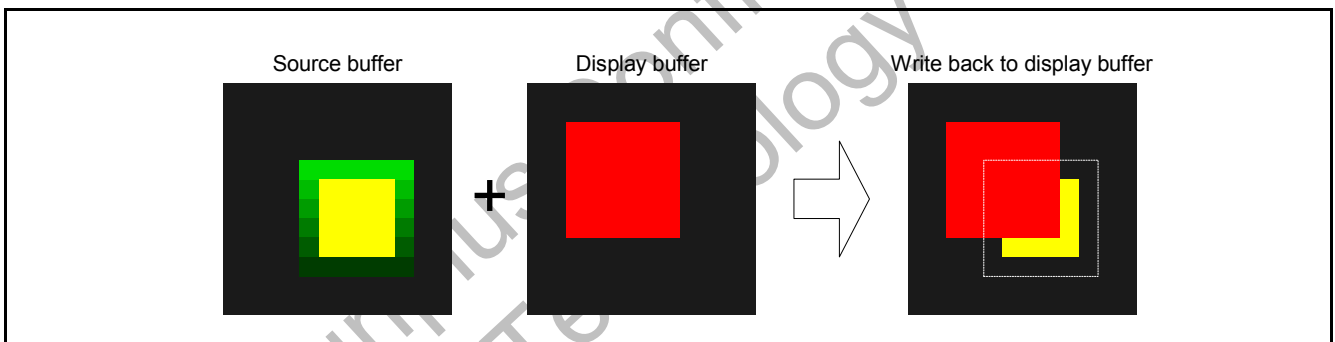


Figure 6-13-13: TROP example 2

Other features

ROP, alpha blending, rotation, mirror and clipping features of the Sprite function are the same as in the BitBLT function.

6.13.4. Line drawing

The SPCA554A 2D graphics engine supports the line drawing function utilizing the Bresenham algorithm. It allows free angle line drawing between two arbitrary points. Figure 6-13-14 shows

the pipeline of the Line Drawing function with related features. The features include ROP, constant alpha blending, stipple line, gradient line, and clipping.

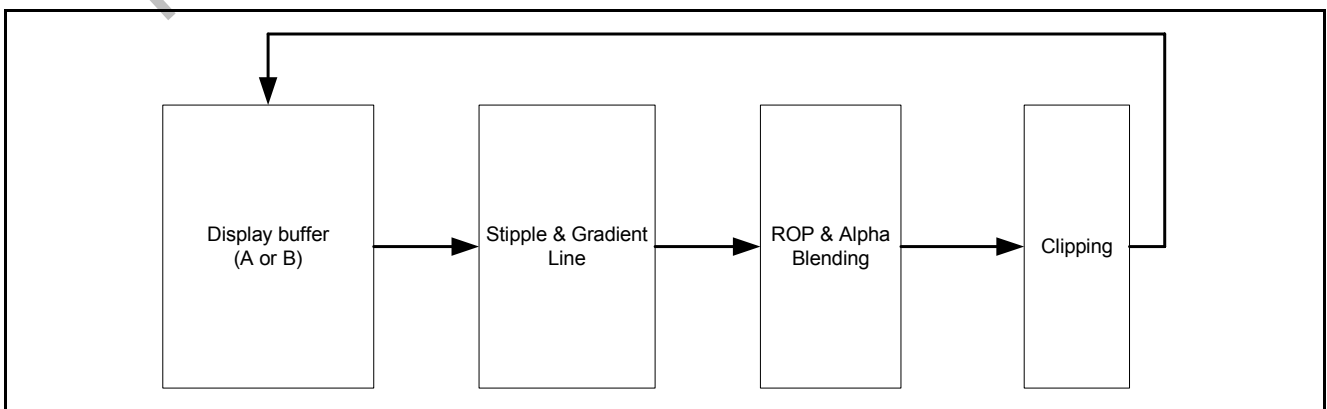


Figure 6-13-14: Line Drawing function pipeline

Drawing Line Segment

In the SPCA554A, line drawing is executed by specifying both the start and end point positions.

Stipple and Gradient Color Line

The pattern used to draw a line is programmable with a 64-bit pattern register which can result in a stipple line. The color used to draw the lines is programmable. It can be a fixed color or a gradient color. The following formula describes the stipple line operation and *Style* is the 64-bit line-style register.

$$Dst_{new} = \begin{cases} Pat & \text{if } Style [i] = 1 \\ Dst & \text{otherwise} \end{cases} \quad \text{where } i \text{ is the pattern period counter and rounding is based on a programmable size.}$$

Gradient Line is performed by defining the initial color value and the gradient value of each color channel. Line color renders color shading along the line.

Figure 6-13-15 shows the three kinds of line drawings, including solid line, gradient line, and stipple line (ROP code 0xF0).

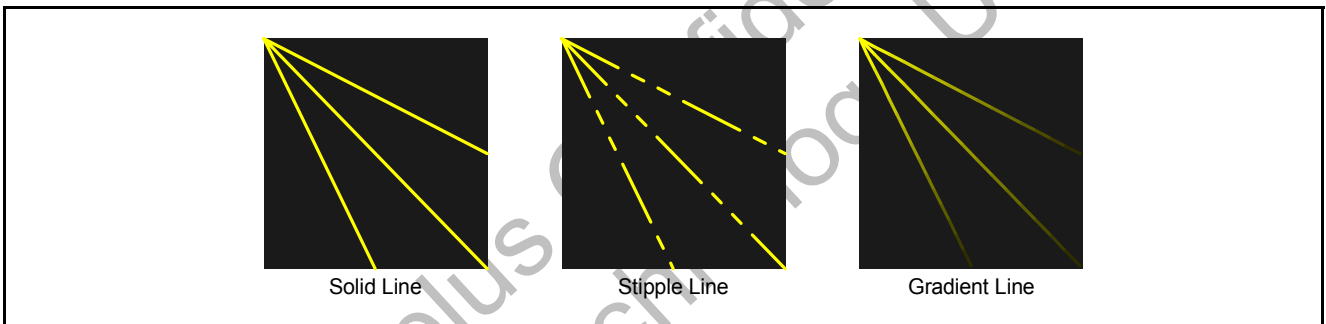


Figure 6-13-15: Three kinds of line drawing examples

Other features

ROP, constant alpha blending, and clipping features of the Line Drawing function are the same as in the BitBLT function. The ROP used in line drawing takes the pattern color and the destination color as operands, but not the source color. Pattern color is used in alpha blending to replace the source color.

6.13.5. Gradient Fill

This function is used to fill a rectangular area with gradient color shading. It is capable of creating heavy to light or dark to shine background effects.

Figure 6-13-16 shows the pipeline of the Gradient Fill function with associated features, including ROP, alpha blending, horizontal and vertical color shading, and clipping.

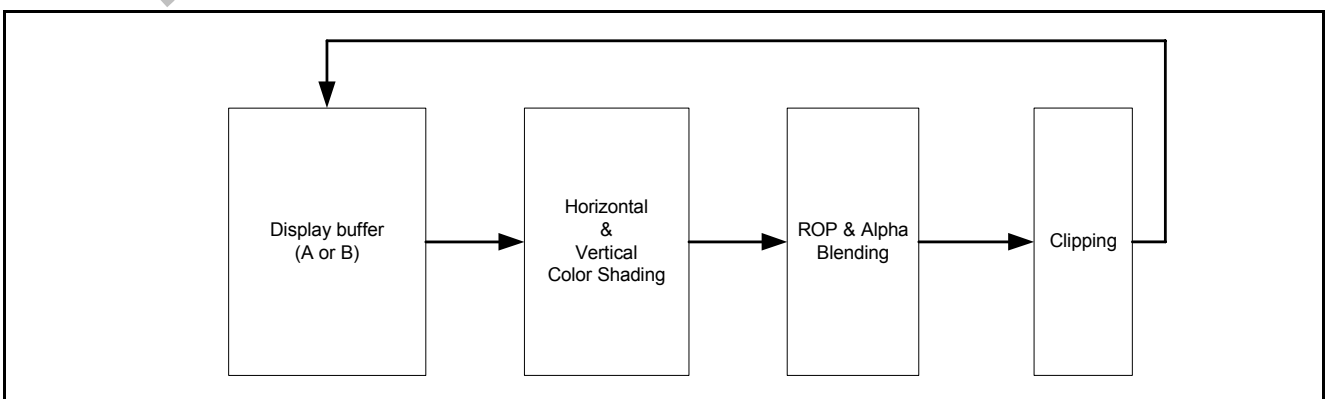


Figure 6-13-16: Gradient Fill function pipeline

Horizontal and vertical Color Shading

Gradient color shading can be applied in horizontal and vertical directions respectively. The initial color value and the gradient color value of R/G/B channel can be specified respectively. The gradient color value can be positive or negative values. Also, horizontal flipping and vertical flipping are supported. The operation formula of color shading is illustrated below, in which, *IniColor* is the initial color and $\Delta HColor$ and $\Delta VColor$ are the gradient colors of the horizontal and vertical direction individually. See the equation below:

$$Dst_C^{new}[\Delta x, \Delta y] = IniColor_C + \Delta x * \Delta HColor_C + \Delta y * \Delta VColor_C$$

where $[\Delta x, \Delta y]$ is the offset of XY-coordinate from the start point and $C = R, G, B$ color channel.

Figure 6-13-17 shows examples of horizontal shading, vertical shading, and a combination of former shading operations (ROP code 0xF0).

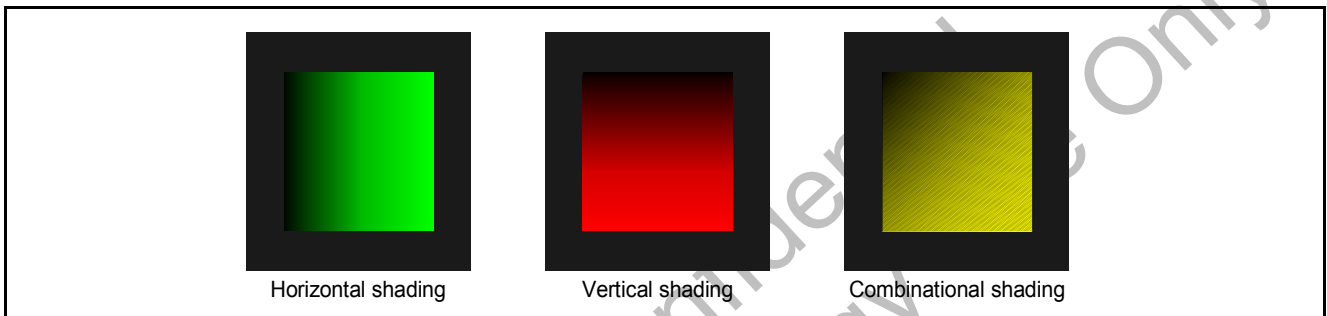


Figure 6-13-17: Gradient Fill Shading example

Figure 6-13-18 shows examples of the both horizontal and vertical shading with arrangement of four flipping operations (ROP code 0xF0).

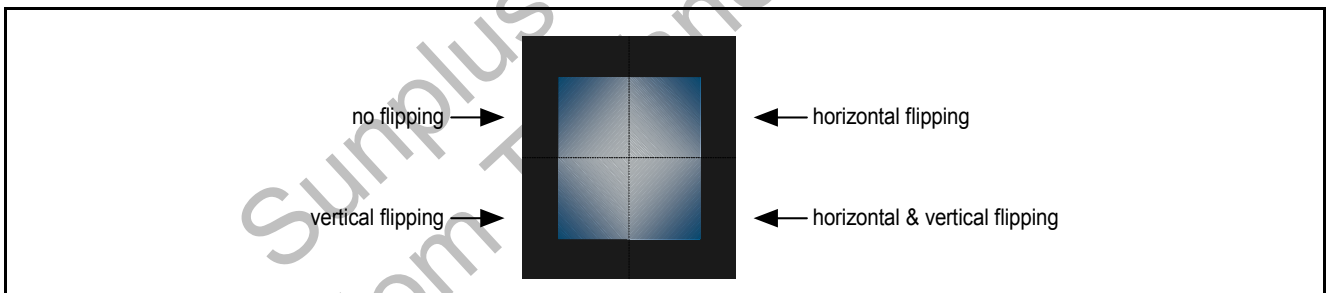


Fig 6-13-18: Gradient Fill Flipping example

Other features

ROP, constant alpha blending, and clipping features of the Gradient Fill function are the same as in the BitBLT function. ROP in the Gradient Fill function uses pattern color and destination color as operands, but not the source color. Pattern color is used in alpha blending to replace the source color.

Figure 6-13-19 shows an example of Gradient Fill with constant alpha blending. The destination image is blended with the horizontal shading of the green color channel.

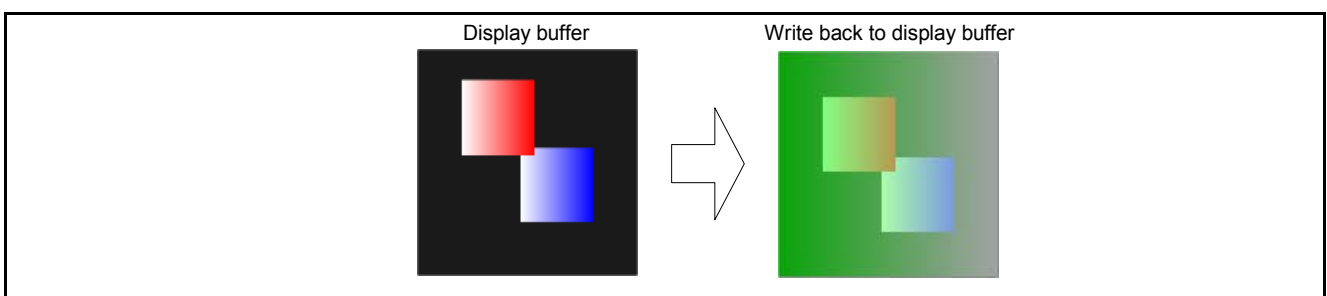


Figure 6-13-19: Gradient Fill alpha blending example

6.14. Display Controller

The display controller of the SPCA554A is in charge of moving image data to the LCM. The sources of image data are the camera image buffer and the graphics image buffer. The display controller has a built-in mixer to overlay/blend the two sources. Before data mixing and blending, the display controller is able to do image cropping, rotation, mirroring and scaling to each of the

two data sources independently. Also the mixed image data can be sent back to the graphics image buffer or the camera's image buffer. The mixed data can be read back by the host processor through the host interface. The following figure illustrates the overall operation concept of the display controller:

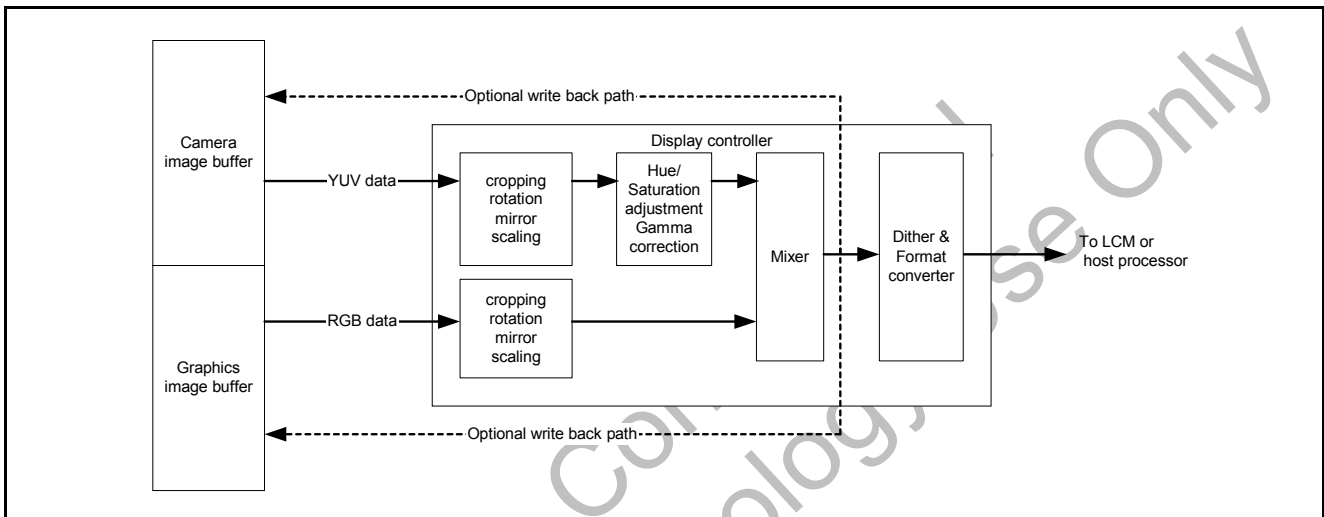


Figure 6-14-1: Display Controller architecture

Both camera image and graphics image can be cropped, rotated, mirrored and scaled. For the camera image, the display controller can also perform hue and saturation adjustment. Gamma correction is implemented in the camera image display path. Gamma is calculated by a 16-section piece-wise-linear curve. The last block in the display controller is dedicated to fit the requirements of different LCM panels. It converts image data to the format and data sequence of the LCM. It also integrates a dither function to enhance viewing quality.

6.14.1. Geometrical Transformation

The SPCA554A Display Controller can do geometrical transformation to camera image and graphics image independently. The transformation includes, image cropping, image rotation and image scaling.

Image cropping

Image cropping is available for both camera image and graphics image. The definition of the cropping area is done by the offsets of the left-top corner and the size of the cropping area. The cropping area of the camera image buffer and the graphics image buffer can be programmed independently. Note that the graphics image buffer referred to here are the display buffers used by the 2D graphics engine. After cropping, the images can be shown in any position on the LCM. The positions to be shown are also independently programmable. After projection to the LCM, and if camera image and graphics image have an overlap area, then the settings in the mixer are to be used to determine how to mix the camera image and the graphics image. It can be overlaid or alpha blended.

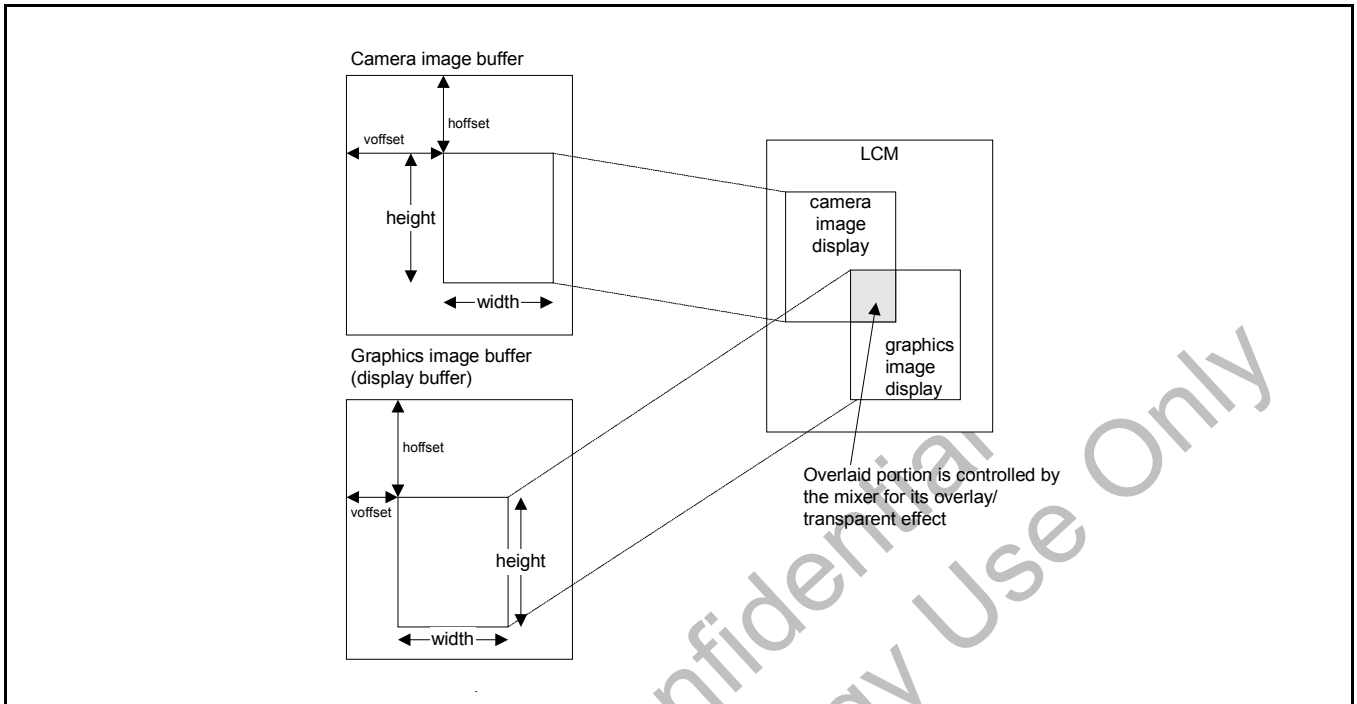


Figure 6-14-2: Image Cropping

Image Rotation

The rotation function of the display controller is useful when the sensor and the LCM have reverse aspect ratios (normally, sensor is a landscape type and the LCM is a portrait type). For example, if the sensor resolution is 640(H)X480(V) and the LCM resolution is 240x480, the rotation function helps the application gain the highest view angle.

The following figure shows capturing of a portrait image using a landscape sensor. There is no way to cover the full size of the portrait image. The image is copied and stored in SDRAM. On the LCM display side, approach A scales down the camera image to fit the horizontal resolution of the LCM. Approach B crops the camera image and then scales the image up to fit the full resolution of the LCM. Neither approach A, nor approach B can get full coverage of the portrait image.

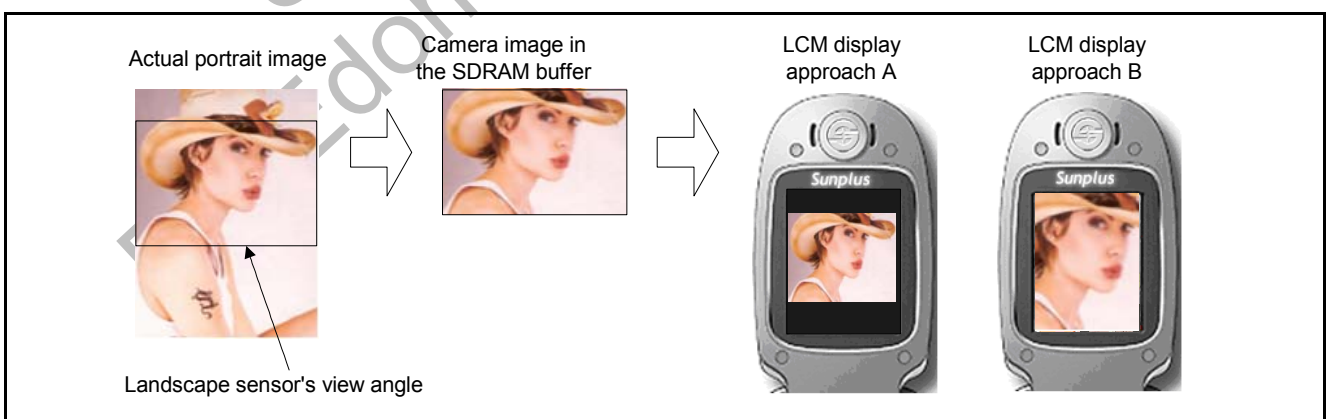


Figure 6-14-3: Camera view angle without rotation

If the mechanical design of the mobile camera rotates the sensor by 90°, then the portrait sensor can get a full coverage of the portrait image. The camera image stored in the SDRAM is also rotated by 90°. The display controller's rotation function can

turn the image upright again for viewing and it also preserves the maximum viewing angle of the portrait image. The rotation function is illustrated in the following diagram:

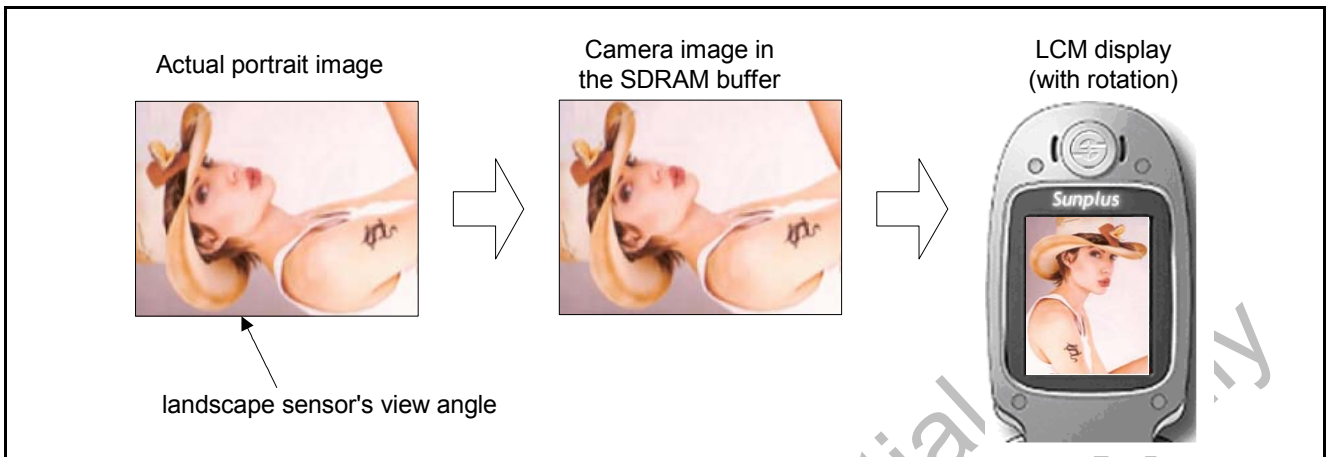


Figure 6-14-4: Maximum view angle using the rotation function

The display controller supports eight rotation modes as shown in the following table:






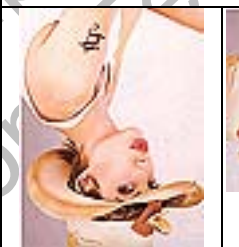


Normal	Rotation 90°	Rotation 180°	Rotation 270°
			
Horizontal mirror	Vertical mirror	Vertical mirror + Rotation 90°	Vertical mirror + Rotation 270°
			

Figure 6-14-5: Rotation in the display engine

Image scaling

SPCA554A's display controller support both up-scaling and down-scaling function after it fetches image data from the graphics image buffer and camera image buffer. The scaling ratio is programmable, ranging from 1.00X to 4.00X with 0.01 steps. As this is a real-time scaling engine, it can be used in the digital zoom for camera preview and video recording modes.

6.14.2. Color Adjust of Camera Image

The SPCA554A provides the brightness, contrast, hue, saturation adjustment and gamma correction functions for camera image adjustment. The brightness and contrast adjustment is done utilizing the following equation. The offset and gain are programmable and are determined as follows:

$$Y_{out} = Y_c \times [Y_{in} + Y_b]$$

$$Y \in [-128, 127]$$

Y_{in} : input Y value

Y_b : offset

Y_c : gain

Y_{out} : output Y value

The hue and saturation adjustment are done by the following equation. $huecosdata$ and $huesindata$ are programmable.

$$U_{out} = UV_{sat} \times [U_{in} \times hue\ cos\ data - V_{in} \times hue\ sin\ data]$$

$$V_{out} = UV_{sat} \times [U_{in} \times hue\ sin\ data + V_{in} \times hue\ cos\ data]$$

$$U, V \in [-128, 127]$$

$huesindata$: hue coefficient

$huecosdata$: hue coefficient

UV_{sat} : saturation coefficient

6.14.3. Mixer

The Mixer performs overlay and blending functions between the camera image and the graphic image. SPCA554A provides several modes to mix graphic images with camera images, including window overlay, color key overlay and alpha blending.

Overlay by Window

The mixer provides two methods for window overlay. The first method uses the camera image as the base image, the other method uses the graphics image as the base image. When using

the camera as the base image, users can define a window inside the camera image buffer and all the pixels inside the window are replaced by the graphics image during superimposing. Alternatively, if the graphics image is used as the base image, the window is defined in the graphics image buffer and all the pixels inside the window are replaced by the camera image during superimposing. The window size and position in the overlay are programmable.



Figure 6-16-6 Overlay with window camera as base



Figure 6-14-7 Overlay by window using graphics base

Overlay by Color-key

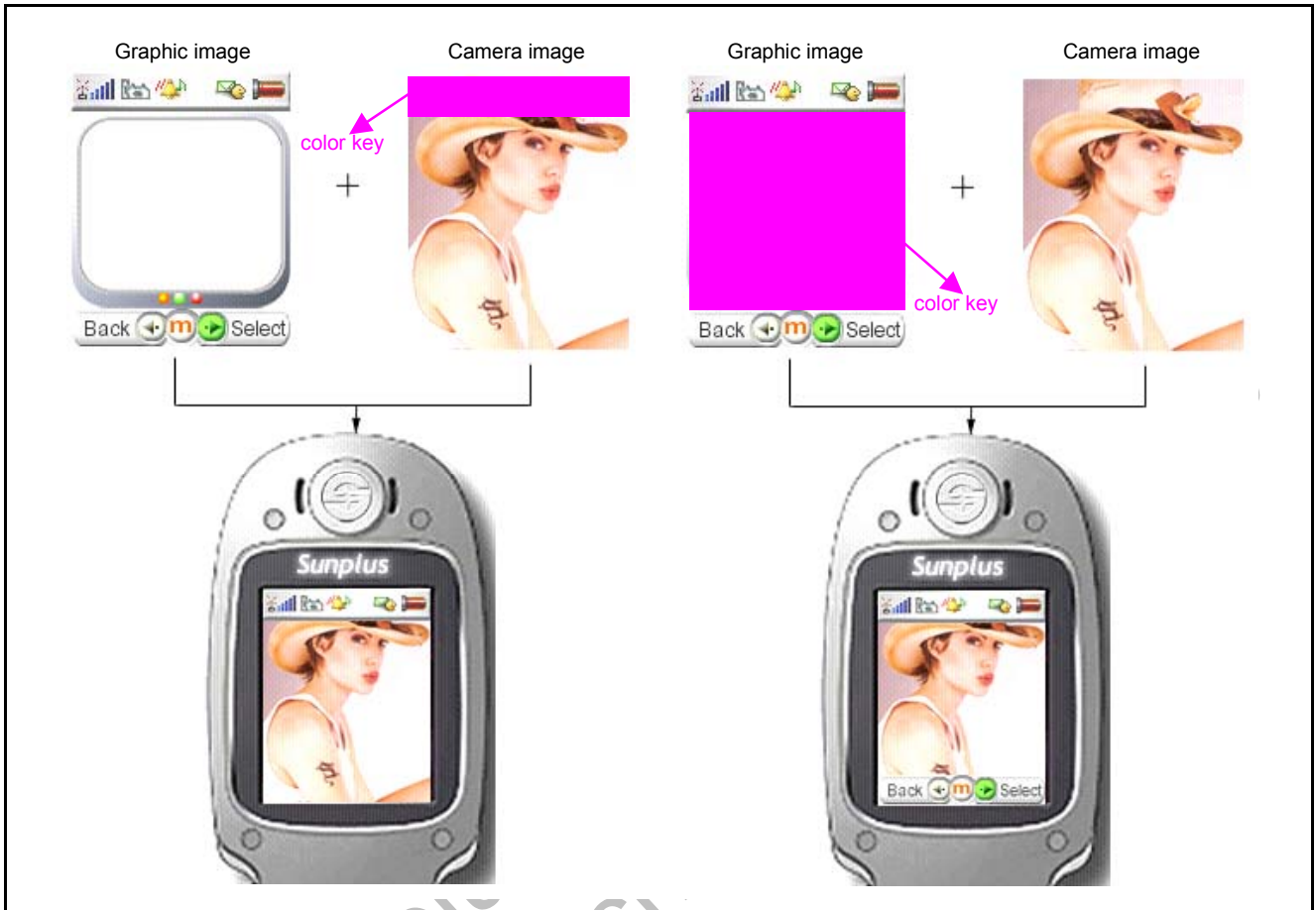
SPCA554 performs camera image color key and graphic image color key operations to achieve several blended image types.

The definition of each color key is described as follows:

- 1.) Graphic image color key 0: When graphic image data is the same as the graphic image color key 0 value, the graphic image is treated as transparent blending with camera image data.
- 2.) Graphic image color key 1: When graphic image data is the same as the graphic image color key 1 value, the camera image is treated as transparent blending with graphic image data.

- 3.) Camera image color key: When camera image data is between camera image color key, camera image is treated as transparent blending with graphic image data.

Using color key for graphic-image-overlay is based on the graphic image color key or camera image color key. The graphic image color key and camera image color key can be defined separately. For example, if the overlay is based on the camera image color-key, all the color key areas of the camera image data will be replaced by the graphic image data. While sub-window overlay is restricted to rectangular shapes, color-key overlay can operate on arbitrary shapes.



Overlay based on camera image color key Overlay based on graphic image color key 0

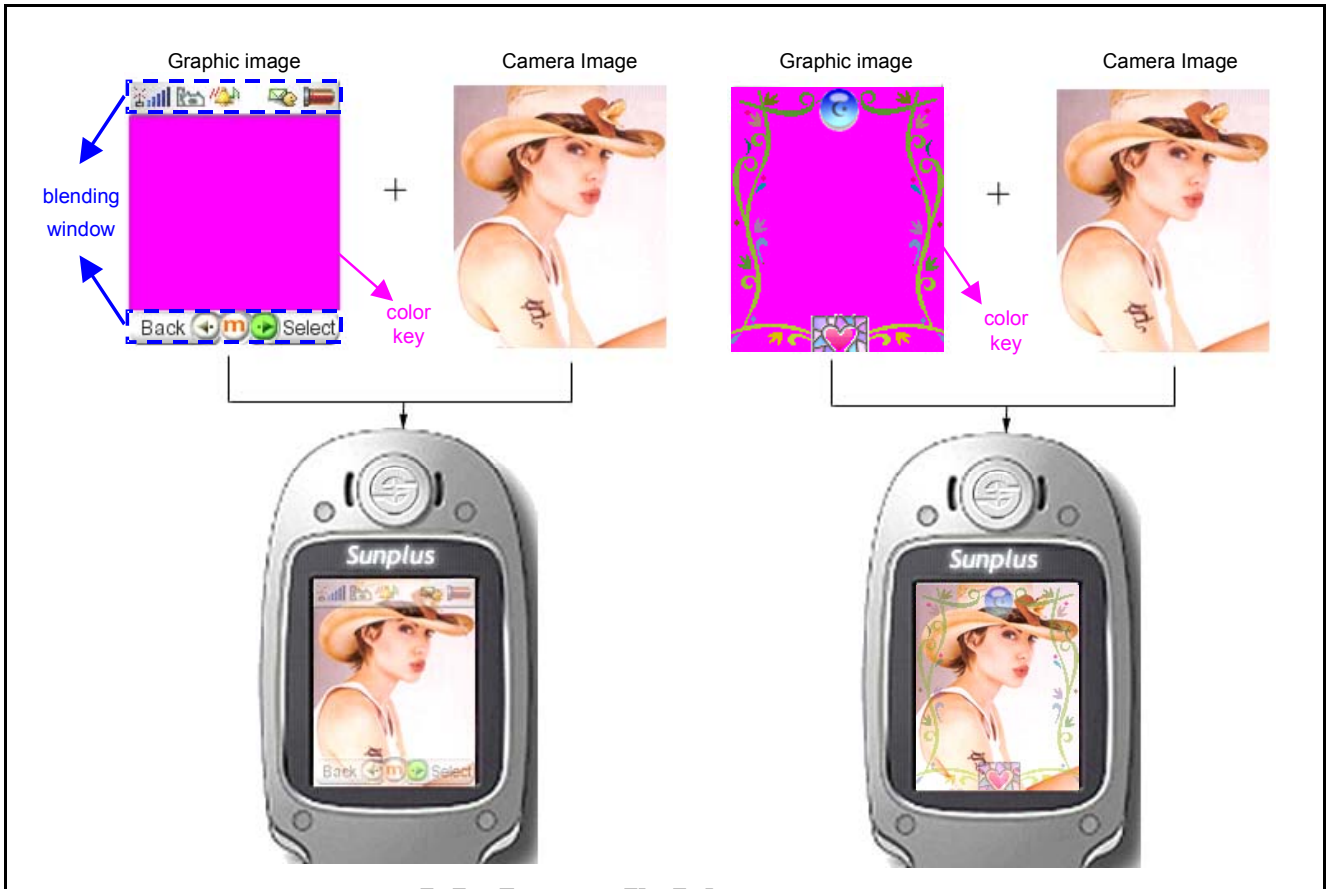
Figure 6-14-8: Examples of overlay by color key

Overlay by Alpha Blending

The SPCA554A also supports α blending between camera image data and graphic image data. If the α blending function is turned-on, the pixel will be blended using graphic image data and camera image data when it matches the following two conditions:

- 1.) If sub-window is defined, the pixel must be inside the sub-window area, and
- 2.) If graphic image color-key is defined, the pixel color does not match the color-key.

The blending ratio (α) can be programmed to 0/4, 1/4, 2/4, 3/4, or 4/4 and the blending area can be separated into 2 regions which have their own α to do α blending.



Define color-key and blending window Define color-key only

Figure 6-14-9: Examples of overlay by Alpha blending

Alpha Blending with Solid Graphic Icon

Using the graphic image color key for alpha blending has an additional feature that can preserve the graphic image data (refrain from being blended). This feature is available only when the graphic image data is in the RGB565 (RGB565 with color-key)

format. In the RGB555 graphic data format, an extra bit is used to check if the graphic image data should be preserved in the camera image to graphic image alpha blending. The following pictures show the difference between preserving the graphic image data and blending the graphic image data.

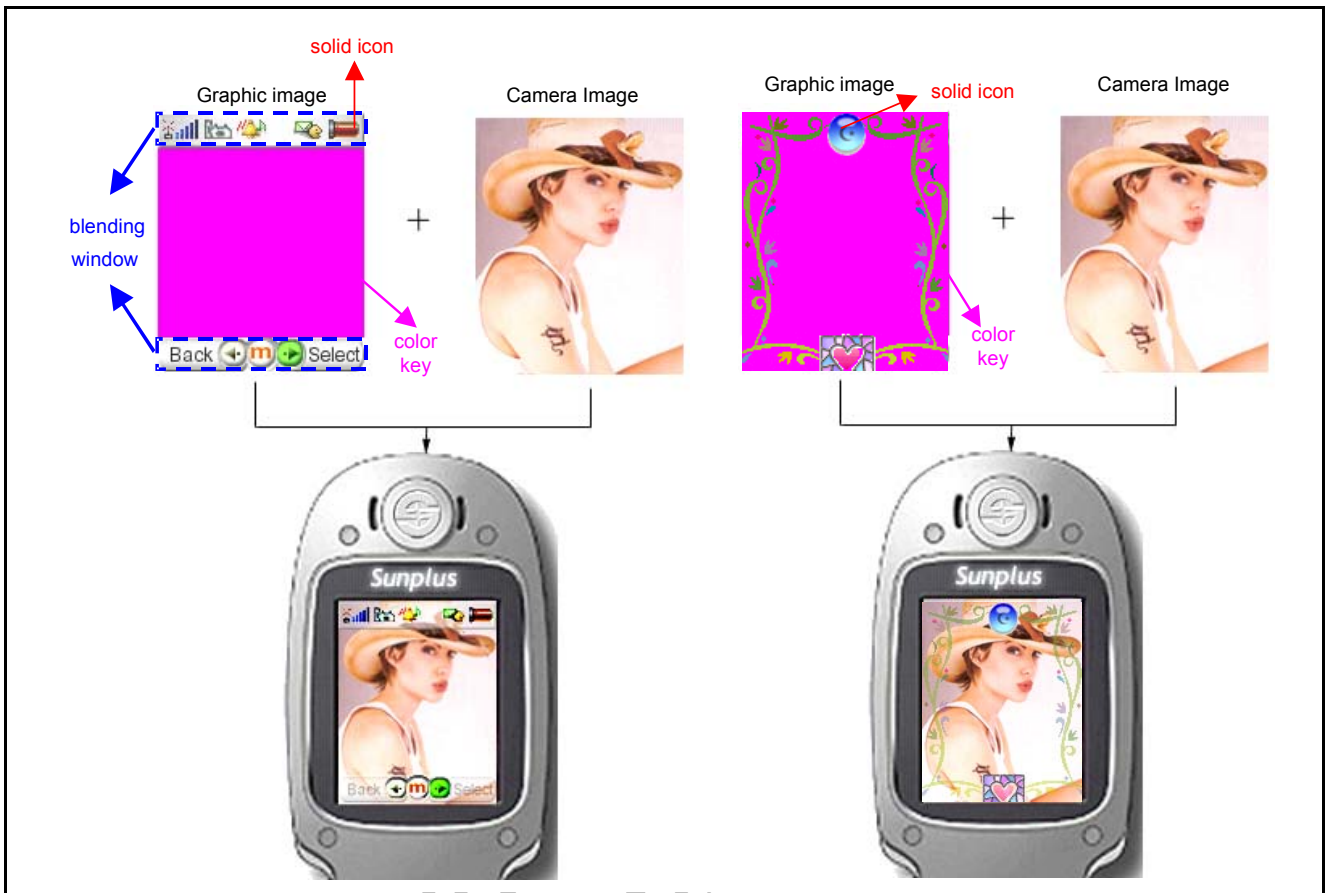


Figure 6-14-10: Examples of Alpha Blending with Solid Graphic Icon

6.14.4. Image Data Format Conversion

In SPCA554A, the graphics data buffer always holds RGB image and the camera image buffer always holds YUV image. Graphics image data can be converted by the display controller and written to the camera image buffer. Alternatively, camera image data can be converted into an RGB image and written to the graphics image buffer. RGB-to-YUV conversion and YUV-to-RGB conversion is part of the display controller function. So the JPEG

file, after being decoded, can be converted to RGB image and saved in the graphics buffer. The PNG and PNG image, after being decoded, can also be converted to a YUV image and saved in the camera image buffer. The following diagram illustrates four possible image data manipulation paths controlled by the display controller:

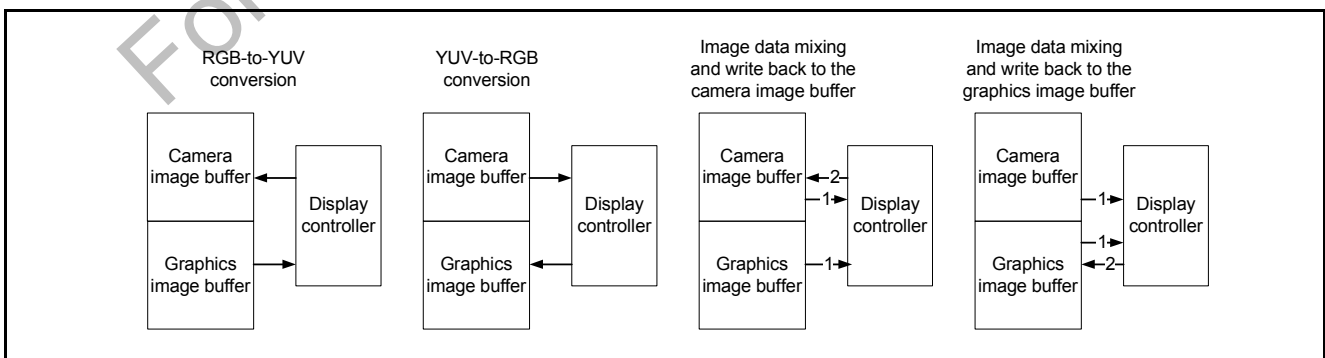


Figure 6-14-11

As the JPEG file size is normally much smaller than the graphics image file in GIF or PNG format, it is very helpful to choose the JPEG format for MMI icons so as to save the storage space.

Finally, these JPEG icons can be converted to RGB icons and saved in the graphics source buffer.

Fade-in/Fade-out effect in JPEG file playback

Converting the YUV camera image to an RGB graphics image is useful in generating fade-in and fade-out effects in JPEG file playback. The first JPEG file is decoded and converted to graphics image. Then, the second JPEG file is decoded and

blended with the previous JPEG file, which is already converted to a graphics image. The alpha blending function is implemented in the mixer module of the display controller. More blending function details are described in the next section.

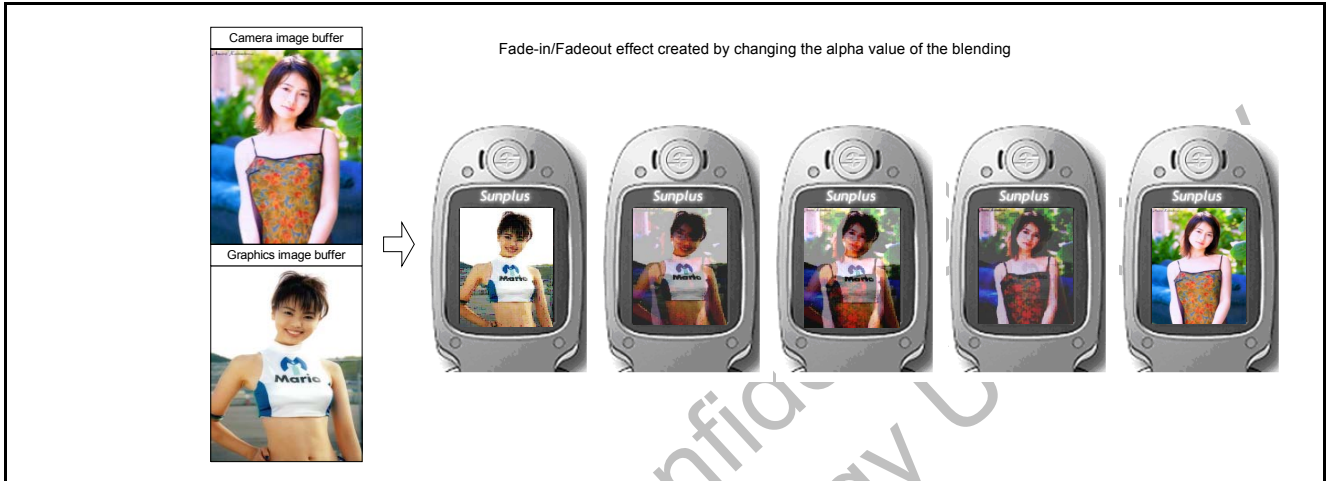


Figure 6-14-12

Video conference

Video conference is another application of image data format conversion. Video conference applications require encoding of incoming video (from the sensor) and decoding incoming video stream (from the host processor) at the same time. Each encoded frame (YUV format), is converted by the SPCA554A into an RGB graphics image. Each video decoded frame (YUV

format) is put into the camera image buffer by the SPCA554A. Then the display controller fetches image data from both the camera image buffer and the graphics image buffer and outputs the overlaid image to the LCM. The decoded YUV image can be shown on the LCM with graphics image (representing the near-side image) overlaid on top of it. The following diagram shows the video conference operating procedure:

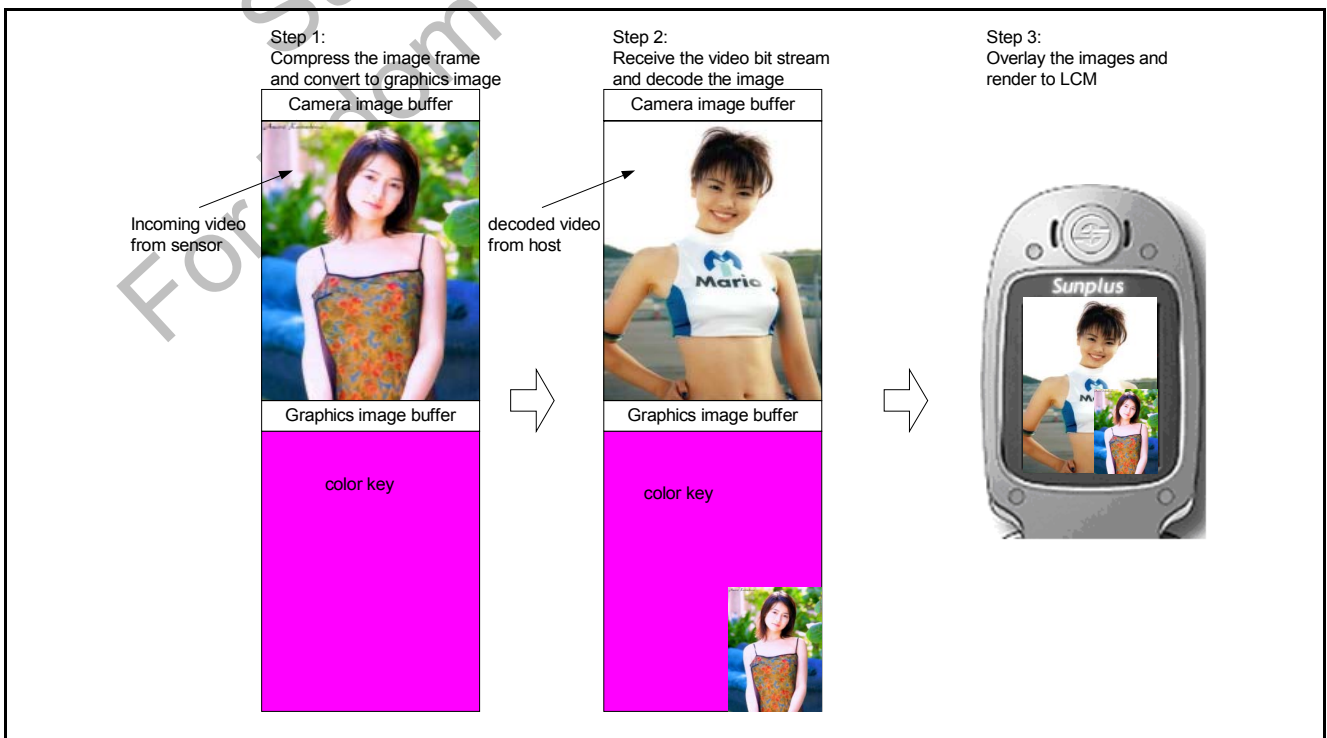


Figure 6-14-13

Photo-sticker image creation

Another powerful function the display controller provides write back of the blended image to the graphics image buffer or the camera image buffer. This function is useful in photo sticker creation. To create a photo sticker image, a JPEG file and a graphics image are decoded and loaded to the camera image buffer and graphics image buffer, respectively. Then, they are

mixed (blended or overlaid) and written back to the camera image buffer. The mixed image can then be compressed, with the SPCA554A JPEG encoder, to another JPEG file. The following figure shows the photo sticker image creation procedure. This example shows a photo sticker creation using color key to overlap image. Besides color key, all blending effects described in the mixer section can be used to create photo sticker images.



6.14.5. LCM control options

SPCA554A provides two chip select signals used to control the main LCM and sub-LCM. LGPIO0 (XCS) is the chip select signal for main LCM control. LGPIO25 (XCS2) is the second chip select signal for sub-LCM control. Two modes are designed to fit the application into SPCA554A's normal operation mode and by-pass mode.

Normal operation mode

When the SPCA554A is not in the by pass mode, both chip select signals are programmable through SPCA554A's internal registers. The application can choose to activate the main LCM or the sub-LCM. It is also possible to activate both LCMs at the same time. In the normal operation mode, the LCM is driven by an SPCA554A Display Controller. The Display Controller sends the data to main LCM or sub-LCM according to the register setting.

By-Pass Mode

When the SPCA554A is in the by-pass mode, the LCM is driven by the host processor. It's not possible for the host processor to

program SPCA554A internal registers. SPCA554A provides another method for the host controller to determine whether to send display data to the main LCM or the sub-LCM. In this method, OTGGPIO0 determines whether to activate the main LCM or sub-LCM. If OTGGPIO0 is low, the chip select signal from the host processor will be routed to XCS. The XCS2 signal remains high, meaning that the sub-LCM is not activated. If OTGGPIO0 is high, the chip select signal from the host processor is routed to the sub LCM. The XCS signal remains high, meaning that the main LCM is not activated. This method allows the host processor to access the main LCM by driving OTGGPIO0 low, and to access the sub-LCM by driving OTGGPIO0 high. This method does not support host access to main LCM and sub-LCM at the same time. Based on application requirements, user can choose to route an GPIO pin to OTGGPIO0 or just use A1 (bit 1 of the address bus) to connect with OTGGPIO0. The following table is the summary of XCS and XCS2 controls.

Pin	Name	In normal operation mode	In by-pass mode	
LGPIO0	XCS	Programmed by internal register	OTGGPIO0=0	Host bus CS signal
			OTGGPIO0=1	1
LGPIO25	XCS2	Programmed by internal register	OTGGPIO0=0	1
			OTGGPIO0=1	Host bus CS signal

Table 6-14-1 XCS and XCS2 control summary

The following diagram illustrates interconnection in the normal and the by-pass mode.

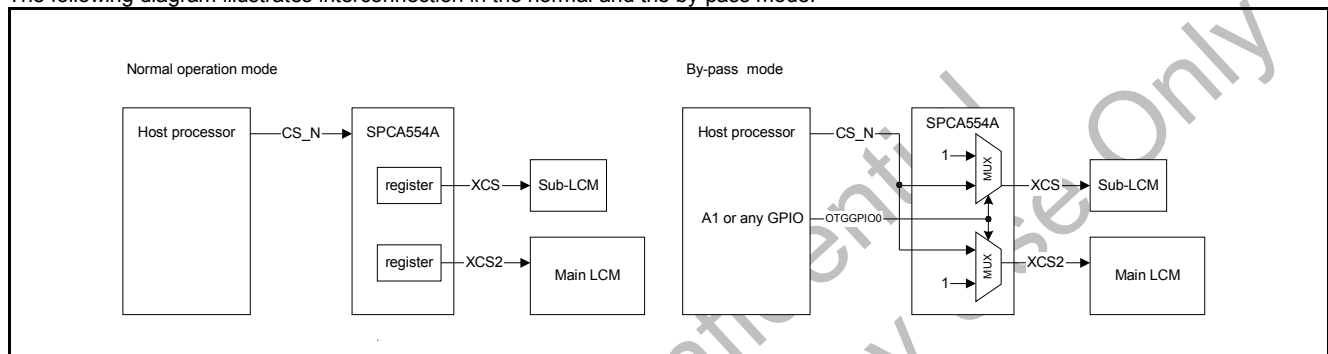


Figure 6-14-14 Chip select pin connection of LCMs

6.15. Peripherals

Timers

SPCA554A has four 24-bit timers. These timers are all general-purpose timers and they count based on a programmable time base. The time base can be 1 ms, 100us, 10us and 100ns. The time base of the timers can be set individually and the timers can be used to do time measurement or used as watch-dog timers. There are two operation modes, the up-count mode and the down-count mode.

In the up-count mode, the timer simply reports how much time has passed after it is triggered. The timer starts to count with an initial value. The counting value is incremental by 1 each time. The initial value is programmable and the counting value wraps around and reloads its initial value after reaching 0XFFFFFF. It also generates an interrupt to the CPU at the same time the counting value reaches 0XFFFFFF. This function is convenient when the system application requires a periodical interrupt source.

In the down-count mode, the timer starts to count with an initial value. The counting value decrements by one each time. After the counting value reaches zero, the timer generates an interrupt, reloads its initial value and continues to count. The down-count mode can be used in conjunction with internal CPU reset circuitry to form a watchdog timer. In watchdog timer applications, the firmware has to reset the timer to its initial value before the counting values reaches zero. Otherwise, the CPU will be reset automatically.

PWM pulse generator

SPCA554A's PWM pulse generator can output a predefined periodical waveform. This function can be used when a system application requires a PWM (Pulse Width Modulation) pulse control. SPCA554A supports four PWM generators. Each of these generators defines a maximum 32-bit basic waveform. For example, 11223344 (Hex) defines the timing waveform below.

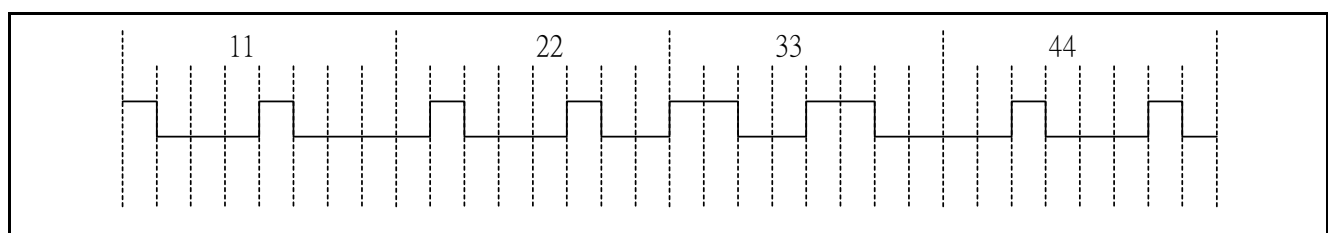


Figure 6-15-1 Basic waveform of the PWM pulse generator

A complete waveform is constructed by assembling the basic waveform, and can be divided into 4 sections; idle period, start period, waveform period and stop period. The waveform period has an active and inactive period. The basic waveform definition

applies to the active period. The signal levels (high or low) in the start period, inactive period and stop period are programmable. Users can also specify how many times the active/inactive period are repeated.

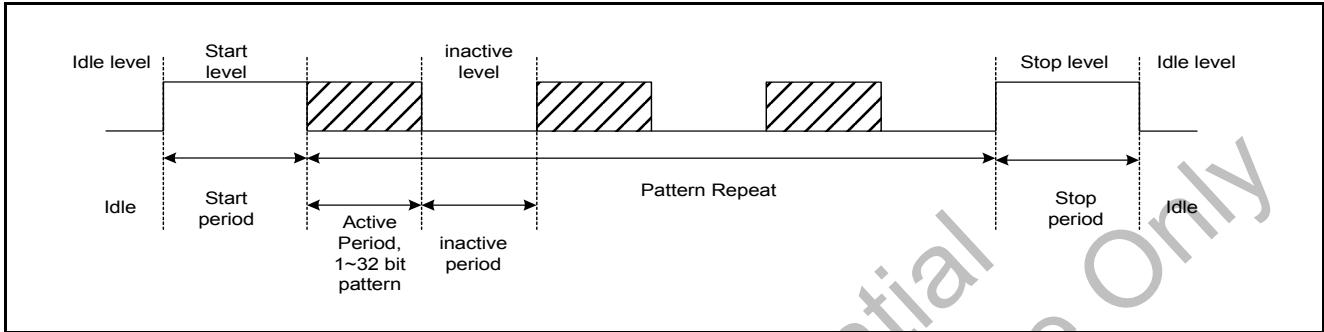


Figure 6-15-2 Complete waveform of the PWM pulse generator

GPIO0~GPIO7 can be used as the outputs of the PWM generator. When used as PWM generator output, GPIO0, GPIO2, GPIO4 and GPIO6 are fully programmable as in the above diagram. GPIO1, GPIO3, GPIO5 and GPIO7 output inverse waveforms of GPIO0, GPIO2, GPIO4 and GPIO6 respectively.

UART
The UART (Universal Asynchronous Receiver/Transmitter) of the SPCA554A is used to output debugging messages. It shares the GPIO pins of the SPCA554A. The baud rate of the UART is programmable, which ranges from 75Hz to 115200Hz.

Sunplus Confidential Use Only
For Edom Technology

7.DISCLAIMER

The information appearing in this publication is believed to be accurate.

Integrated circuits sold by Sunplus Technology are covered by the warranty and patent indemnification provisions stipulated in the terms of sale only. SUNPLUS makes no warranty, expressed, statutory implied or by description regarding the information in this publication or regarding the freedom of the described chip(s) from patent infringement. FURTHERMORE, SUNPLUS MAKES NO WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PURPOSE. SUNPLUS reserves the right to halt production or alter the specifications and prices at any time without notice. Accordingly, the reader is cautioned to verify that the data sheets and other information in this publication are current before placing orders. Products described herein are intended for use in normal commercial applications. Applications involving unusual environmental or reliability requirements, e.g. military equipment or medical life support equipment, are specifically not recommended without additional processing by SUNPLUS for such applications. Please note that application circuits illustrated in this document are for reference purposes only.

Sunplus Confidential
For Edom Technology Use Only

8. REVISION HISTORY

Date	Revision #	Description	Page
MAR. 05, 2004	0.1	Original	103
APR. 08, 2004	0.2	Syntax and related modifications.	All pages

Sunplus Confidential
For Edom Technology Use Only