1. UART INTERFACE

1.1. Interface instructions

The UART and High Speed UART interface of VC0706 is based on the design of standard UART interface. It can be connected to the serial communication ports of a PC and an external MCU. VC0706 supports RXD and TXD,but not support the all pins that are related to MODEM functions.

Using Serial communicate Protocol,we can get information of vc0706 or control it,such as taking photo ,reading photo ,etc.

1.1.1.UART Interface pins

Out of the Serial Communication, the UART and High Speed UART interface pins of VC0706 and GPIO pins are in common.

Pin No.	GPIO	UART	Instructions
76	GPIO0	UART_TX	Send data through VC0706 UART interface
77	GPIO1	UART_RX	Receive data through VC0706 UART interface
80	GPIO14	HSUART_TX	Send data through VC0706 HS_UART interface
81	GPIO15	HSUART_RX	Receive data through VC0706 HS_UART interface

1.1.2.Select Serial Communication Interface

VC0706 supports three interfaces, UART, HS_UART, SPI, communication with the external MCU.But whenever, only one can be used as the main communication interface to control system.

UART is VC0706 default interface, if using other interfaces as the main communication interface, it is necessary to set some specified values in EEPROM or SPI Flash.

Address: 0x0007
Length: 1 byte

Interface	configuration value
UART	0x01
HS_UART	0x02
SPI	0x03

1.1.3.UART baud rate setting

About VC0706 UART baud rate, the default is 38400, the highest is 115200, it can be set by the specified values in EEPROM or SPI Flash.
Address: 0x0008
Length: 2 bytes

baud rate	configuration value
9600	0xAEC8

19200	0x56E4
38400	0x2AF2
57600	0x1C4C
115200	0x0DA6

1.1.4.HS_UART baud rate setting

About VC0706 HS_UART baud rate, the default is 115200, the highest is 921600, it can be set by the specified values in EEPROM or SPI Flash.
Address: 0x000A
Length: 4 bytes

baud rate	configuration value
38400	0x002B 0x03C8
57600	0x001D 0x0130
115200	0x000E 0x0298
460800	0x0003 0x02A6
921600	0x0001 0x0353

1.2.Communication Protocol

Communication Protocol format are as follows ：

Receive command format ：
Protocol sign（1byte）+Serial number（1byte）+Command（1byte）+Data-lengths（1byte）+Data（0~16bytes）

Return command format ：
Protocol sign（1byte）+Serial number（1byte）+Command（1byte）+Status（1byte）+Data-lengths（1byte）+Data（0~16bytes）

Protocol sign ：it marks that the protocol is VC0706 Serial Communication Protocol, the receive sign is **0x56('V')**, return sign is **0x76('v').**

Serial number ：it specify one device when there are serval devices in communication at the same time,the value of this byte range from 0 to 255 .
Command ：it marks a special command.
Data-lengths ：it shows the data lengths behind itself., not include Protocol sign, Serial number, Command, Data length.
Data ：used in commands, every command has different data lengths and format, range from 0 to 16 bytes.
Status ：this byte shows whether the receive command is right or wrong , 0 is right, others are wrong.

Status code	Error instructions
0	Executing command right.
1	System don't receive the command.
2	The data-length is error.

3	Data format error.
4	The command can not execute now .
5	Command received,but executed wrong.

●To multi-byte data type, the lower bytes follow the higher bytes.

●If serial number is wrong, the system will not return any content.

●The max communication data lengths are 16 bytes

●If the command format is wrong or command executes wrong , the status byte will be 1 byte and the data length byte will be 0.

1.3.communication command

1.3.1.command form

Command Definition	Command Byte	Instructions
GEN_VERSION	0x11	Get Firmware version information
SET_SERIAL_NUMBER	0x21	Set serial number
SET_PORT	0x24	Set port
SYSTEM_RESET	0x26	System reset
READ_DATA	0x30	Read data regisvter
WRITE_DATA	0x31	Write data register
READ_FBUF	0x32	Read buffer register
WRITE_FBUF	0x33	Write buffer register
GET_FBUF_LEN	0x34	Get image lengths in frame buffer
SET_FBUF_LEN	0x35	Set image lengths in frame buffer
FBUF_CTRL	0x36	Control frame buffer register
COMM_MOTION_CTRL	0x37	Motion detect on or off in comunication interface
COMM_MOTION_STATUS	0x38	Get motion monitoring status in comunication interface
COMM_MOTION_DETECTED	0x39	Motion has been detected by comunication interface
MIRROR_CTRL	0x3A	Mirror control
MIRROR_STATUS	0x3B	Mirror status
COLOR_CTRL	0x3C	Control color
COLOR_STATUS	0x3D	Color status
POWER_SAVE_CTRL	0x3E	Power mode control
POWER_SAVE_STATUS	0x3F	Power save mode or not
AE_CTRL	0x40	Control AE
AE_STATUS	0x41	AE status
MOTION_CTRL	0x42	Motion control
MOTION_STATUS	0x43	Get motion status
TV_OUT_CTRL	0x44	TV output on or off control
OSD_ADD_CHAR	0x45	Add characters to OSD channels
DOWNSIZE_CTRL	0x54	Downsize Control
DOWNSIZE_STATUS	0x55	Downsize status
GET_FLASH_SIZE	0x60	Get SPI flash size
ERASE_FLASH_SECTOR	0x61	Erase one block of the flash
ERASE_FLASH_ALL	0x62	Erase the whole flash
READ_LOGO	0x70	Read and show logo
SET_BITMAP	0x71	Bitmap operation
BATCH_WRITE	0x80	Write mass data at a time

1.3.2.command detailed description

1.3.2.1.GEN_VERSION

Command function ：Get Firmware version information
Command format ：0x56+Serial number+0x11+0x00
Return format ：0x76+Serial number+0x11+0x00+0x0B+"VC0706 1.00"
Additionally ：
Version format：VC0706+space+first version(1 byte)+'.'+second version(2 bytes);11 bytes in all
Character-string format：e.g. VC0706 1.00

1.3.2.2.SET_SERIAL_NUMBER

Command function ：Set serial number
Command format ：0x56+Serial number+0x21+0x01+New serial number
Return format ：0x76+Serial number+0x21+0x00+0x00
The serial number in the return command is the older one, but from now on system will use the new number.
E.g. 0x56+0x00+0x21+0x01+0x10 change the serial number 0x00 to 0x10
After setting new serial number, you need the new one to send command.

1.3.2.3.SET_PORT

Command function ：Set the property of communication interface
Command format ：0x56+Serial number+0x24+Data-length+interface type1byte）
+configuration data
Such as set **MCU UART**：
0x56+Serial number+0x24+0x03+0x01+S1RELH(1byte)+S1RELL(1byte)
interface type：
0x01：MCU UART
Return format ：
OK: 0x76+Serial number+0x24+0x00+0x00
ERROR: 0x76+Serial number+0x24+0x03+0x00
E.g.
● 0x56+0x00+0x24+0x03+0x01+0x0D+0xA6 The baud rate will be 115200.
●S1RELH and S1RELL are the values that be written to S1RELH register and S1RELL register.
●Baud rate and baud rate register：

Baud rate	S1RELH	S1RELL
9600	0xAE	0xC8
19200	0x56	0xE4
38400	0x2A	0xF2
57600	0x1C	0x1C
115200	0x0D	0xA6

1.3.2.4.SYSTEM_RESET

Command function ：System reset
Command format ：0x56+Serial number+0x26+0x00
Return format ：0x76+Serial number+0x26+0x00+0x00
E.g. 0x56+0x00+0x26+0x00 The system will be reset.
When receiving return command, ten millisecond later, system will reset and receive some basic configuration information.

1.3.2.5.READ_DATA

Command function ：Read Chip-register, Sensor-register, I2C EEPROM, SPI device or CCIR656's data
Command format ：0x56+serial number+0x30+Data-length+device-type(1 byte)+the data num ready to read(1 byte)+configuration information.
Device-type ：

1. Chip register
2. Sensor register
3. CCIR656register
4. I2C EEPROM
5. SPI EEPROM
6. SPI Flash

The command format of different devices are as follows：
Chip register：0x56+serial number+0x30+0x04+0x01+the data num ready to read +register address(2 bytes).
Sensor register：0x56+serial number+0x30+0x05+0x02+the data num ready to read+register data width(1 byte)+register address(2 bytes).
CCIR656register：0x56+serial number+0x30+0x05+0x03+the number of ready reading+register data width(1 byte)+register address(2 bytes).
I2C EEPROM：0x56+serial number+0x30+0x04+0x04+the data num ready to read+register address(2 bytes).
SPI EEPROM：0x56+serial number+0x30+0x04+0x05+the number of ready reading+register address(2 bytes).
SPI Flash：0x56+serial number+0x30+0x06+0x06+the data num ready to read+register address (4 bytes).
The data num ready to read：Begin with the register address, the number of which you are ready to read.But the number is restricted.If that the number multiplied by data width is more than 16 bytes, it could be wrong.
To sensor and CCIR656 devices, the register data width may be different. The register data width means the data width of every register, values range from 1 to 3 bytes. The whole data number equals to that the front data number multiplied by register widths.
Return format ：
Ok：0x76+serial number+0x30+0x00+the number of reading+register address
Error： if the device-type is wrong, or register address width is different from the one of system on controlling sensor and CCIR656, we will receive：0x76+serial number+0x30+0x03+0x00
In returning command, the data number is that the number of reading multiplied by data width.

E.g.

- 0x56+0x00+0x30+0x04+0x01+0x02+0x18+0x00
  Begin from chip register address 0x1800, read 2 bytes data.
- 0x56+0x00+0x30+0x05+0x02+0x01+0x01+0x00+0x20
  Begin from sensor register address 0x0020, read 1 byte data. The sensor register data width is one byte.
- 0x56+0x00+0x30+0x05+0x02+0x02+0x02+0x00+0x20
  Begin from sensor register address 0x0020, read 2 bytes data. The sensor register data width is 2 bytes.
- 0x56+0x00+0x30+0x05+0x03+0x03+0x01+0x00+0x40
  Begin from CCIR656 register address 0x0040, read 3 bytes data. The CCIR656 register data width is 1 byte.
- 0x56+0x00+0x30+0x04+0x04+0x04+0x02+0x40
  Begin from I2C EEPROM address 0x0240, read 4 bytes data.
- 0x56+0x00+0x30+0x04+0x05+0x03+0x08+0x00
  Begin from SPI EEPROM address 0x0800, read 3 bytes data.
- 0x56+0x00+0x30+0x06+0x06+0x08+0x00+0x20+0x40+0x60
  Begin from SPI Flash address 0x00204060, read 8 bytes data.

1.3.2.6.WRITE_DATA

Command function ：Write Chip-register, Sensor-register, I2C EEPROM, SPI device or CCIR656's data

Command format ：0x56+serial number+0x31+Data-length+device-type(1 byte)+the data num ready to write(1 byte)+configuration information+data.

Device-type ：

1. Chip register
2. Sensor register
3. CCIR656register
4. I2C EEPROM
5. SPI EEPROM
6. SPI Flash

The concrete command format as follows ：

Chip register：0x56+serial number+0x31+Data-length+0x01+the data num ready to write+register address(2 bytes)+data.

Sensor register：0x56+serial number+0x31+Data-length+0x02+the data num ready to write+register data width(1 byte)+register address(2 bytes)+data.

CCIR656register：0x56+serial number+0x31+Data-length+0x03+the data num ready to write+register data width(1 byte)+register address(2 bytes)+data.

I2C EEPROM：0x56+serial number+0x31+Data-length+0x04+the data num ready to write+register address(2 bytes)+data.

SPI EEPROM：0x56+serial number+0x31+Data-length+0x05+the data num ready to write+register address(2 bytes)+data.

SPI Flash：0x56+serial number+0x31+Data-length+0x06+the data num ready to write+register

address (4 bytes)+data.
The data num ready to write：Begin with the register address, the number of which you are ready to write.But the number is limited by which the data number in command can't be more than 16 bytes.
To sensor and CCIR656 devices, if the data width is more than 1 byte, the lower byte follows the higher byte.
To SPI EEPROM and SPI Flash, the writing address is not more than 2 pages, otherwise it will be wrong. The page of every SPI device is different. Pay much attention on using multibyte data to control SPI device.
Return format ：
Ok：0x76+serial number+0x31+0x00+0x00
Error： if the device-type is wrong, or register address width is different from the one of system on controlling sensor and CCIR656, we will receive：0x76+serial number+0x31+0x03+0x00
In return command, the data number is that the number of reading multiplied by data width.
E.g.

- 0x56+0x00+0x31+0x06+0x01+0x02+0x18+0x00+0x11+0x22
  Begin from chip register address 0x1800, write 2 bytes data,0x11,0x22.
- 0x56+0x00+0x31+0x06+0x02+0x01+0x01+0x00+0x20+0x80
  Begin from sensor register address 0x0020, write 1 byte data,0x80. The sensor register data width is one byte.
- 0x56+0x00+0x31+0x09+0x02+0x02+0x02+0x00+0x20+0x01+0x00+0x02+0x00
  Begin from sensor register address 0x0020, write 2 bytes data,0x0100,0x0200. The sensor register data width is 2 bytes.
- 0x56+0x00+0x31+0x06+0x03+0x03+0x01+0x00+0x20+0x41+0x42+0x43
  Begin from CCIR656 register address 0x0020, write 3 bytes data,0x41,0x42,0x43. The CCIR656 register data width is 1 byte.
- 0x56+0x00+0x31+0x08+0x04+0x04+0x02+0x40+0x11+0x12+0x13+0x14
  Begin from I2C EEPROM address 0x0240, write 4 bytes data,0x11,0x12,0x13,0x14.
- 0x56+0x00+0x31+0x07+0x05+0x03+0x02+0x40+0x21+0x22+0x23
  Begin from SPI EEPROM address 0x0240, write 3 bytes data,0x21,0x22,0x23.
- 0x56+0x00+0x31+0x0D+0x06+0x07+0x00+0x20+0x40+0x60+0x30+0x31+0x32+0x33+ 0x34+ 0x35+0x36
  Begin from SPI Flash address 0x00204060, write 7 bytes data,0x30,0x31,0x32,0x33,0x34,0x35,0x36.

1.3.2.7.READ_FBUF

Command function ：read image data from FBUF.
Command format ：0x56+serial number+0x32+0x0C+FBUF type(1 byte)+control mode(1 byte) +starting address(4 bytes)+data-length(4 bytes)+delay(2 bytes)
FBUF type：current frame or next frame
0：current frame
1：next frame

Control mode：the mode by which image data transfer

Bit0：0：data transfer by MCU mode

1：data transfer by DMA mode

Bit[2:1]：2'b11

Bit3： 1'b11

Starting address： the address in fbuf to store the image data.

Data-length：the byte number ready to read, it must be the multiple of 4.

Delay：the delay time between command and data, the unit is 0.01 millisecond.

Return format ：

Ok：if execute right, return 0x76+serial number+0x32+0x00+0x00, the following is image data, at last, return 0x76+serial number+0x32+0x00+0x00 again.

Error：if execute wrong , 0x76+serial number+0x32+error code+0x01 will be received .

E.g.

0x56+0x00+0x32+0x0C+0x00+0x0F+0x00+0x00+0x00+0x10+0x00+0x00+0x02+0x00+0x10+0x00

Begin with the address 0x0100, read current frame data by DMA mode.The data-length is 0x0200, delay time is 40.96 millisecond.

After receiving command, return the ack command, then read the certain lengths data and send to the external MCU by SPI, the ack command will be sent again to indicate the data sending has finished.

Before using the command, we must stop the certain fbuf ,otherwise operation will fail because the image frames update all the time.

You can read the image data at one time or many times .

Suggest using DMA mode in order to get higher transmission speed.

1.3.2.8.WRITE_FBUF

Command function ：write image data to FBUF

Command format ：0x56+serial number+0x33+0x0B+control mode(1 byte)+starting address(4 bytes)+data-lengths(4 bytes)+delay(2 bytes)

Control mode：the mode by which image data transfer

Bit0：0：data transfer by MCU mode

1：data transfer by DMA mode

Bit[2:1]：2'b11

Bit3： 1'b11

Bit4： whether it is the first to write data to buffer.

0：no **1：**yes

If it need many times to write data, the bit4 is 1 at the first time, others are 0.

Starting address：the address in fbuf to store the image data.

Data-lengths：the byte number of ready writing, it must be the multiple of 4.

Delay：the delay time between command and data, the measure is 0.01 millisecond.

Return format ：

Ok：if execute right, return 0x76+serial number+0x33+0x00+0x00, then wait for receiving image data.Until the data receiving all have finished, the second return command 0x76+serial

number+0x33+0x00+0x00.
Error：0x76+serial number+0x33+error code+0x00
E.g.
●0x56+0x00+0x33+0x0B+0x0F+0x00+0x00+0x00+0x10+0x00+0x00+0x02+0x00+0x10+0x00
Begin from the address 0x0100, write data by DMA mode.The data-lengths is 0x0200.
After receiving command, wait for program responding and returning command, there will be the second return command till the data receiving all have finished.
Must stop the certain frame before using the command.
Suggest using DMA mode in order to get higher transmission speed.

1.3.2.9.GET_FBUF_LEN

Command function ：get byte-lengths inFBUF
Command format ：0x56+serial number+0x34+0x01+FBUF type(1 byte)
FBUF type：current frame or next frame
0：current frame
1：next frame
Return format ：
OK：0x76+serial number+0x34+0x00+0x04+FBUF data-lengths(4 bytes)
Error：0x76+serial number+0x34+0x03+0x00
E.g.
● 0x56+0x00+0x34+0x01+0x00
get current frame byte-lengths in buffer register
● 0x56+0x00+0x34+0x01+0x01
get next frame byte-lengths in buffer register
Generally, the command is used before reading FBUF.

1.3.2.10.SET_FBUF_LEN

Command function ：set byte-lengths in FBUF
Command format ：0x56+serial number+0x35+0x04+FBUF byte-lengths(4 bytes)
The largest length is 65535.
Return format ：
OK：0x76+serial number+0x35+0x00+0x00
E.g.
● 0x56+0x00+0x35+0x04+0x00+0x00+0xA0+0xB0
Set the FBUF byte-lengths with 0xA0B0.
The length and the byte-lengths of writing must be the same, otherwise the image will be error on TV.

1.3.2.11.FBUF_CTRL

Command function ：control frame buffer register
Command format ：0x56+serial number+0x36+0x01+control flag(1 byte)

control flag：

0：stop current frame

1：stop next frame

2：resume frame

3：step frame

Return format ：

OK：0x76+serial number+0x36+0x00+0x00

Error：0x76+serial number+0x36+0x03+0x00

E.g.

● 0x56+0x00+0x36+0x01+0x00 stop current frame

● 0x56+0x00+0x36+0x01+0x01 stop next frame

● 0x56+0x00+0x36+0x01+0x02 resume frame

● 0x56+0x00+0x36+0x01+0x03 step frame

1.3.2.12.COMM_MOTION_CTRL

Command function ：motion detect on or off in comunication interface

Command format ：0x56+serial number+0x37+0x01+control flag(1 byte)

control flag：

0：stop motion monitoring

1：start motion monitoring

Return format ：

OK：0x76+serial number+0x37+0x00+0x00

Error：0x76+serial number+0x37+0x03+0x00

E.g.

● 0x56+0x00+0x37+0x01+0x01 start motion monitoring

● 0x56+0x00+0x37+0x01+0x00 stop motion monitoring

After starting motion monitoring, once system detects motion, it will send COMM_MOTION_DETECTED command.

1.3.2.13.COMM_MOTION_STATUS

Command function ：get motion monitoring status in comunication interface.

Command format ：0x56+serial number+0x38+0x00

Return format ：

OK：0x76+serial number+0x38+0x00+0x01+control flag(1 byte)

control flag：

0：stop motion monitoring

1：start motion monitoring

E.g.

● 0x56+0x00+0x38+0x00 get motion monitoring presently status in comunication interface.

1.3.2.14.COMM_MOTION_DETECTED

Command function ： detect motion

Command format ：

After starting motion monitoring, once system detects motion, it will send the command.

Return format ：0x76+serial number+0x39+0x00

E.g.

● 0x76+0x00+0x39+0x00 detect motion

It is an active command that system send to control terminal.

1.3.2.15.MIRROR_CTRL

Command function ： control show status of sensor mirror.

Command format ：0x56+serial number+0x3A+0x02+control mode(1 byte)+Mirror mode(1 byte)

Control mode：

0：control mirror by GPIO.

1：control mirror by UART.

Mirror mode：whether show mirror by UART, it is effective only with UART.It needs GPIO value to set with GPIO control.

0：do not show mirror

1：show mirror

Return format ：

OK：0x76+serial number+0x3A+0x00+0x00

Error：0x76+serial number+0x3A+error code+0x00

E.g.

● 0x56+0x00+0x3A+0x02+0x01+0x00
Control mirror by UART and don't show mirror.

● 0x56+0x00+0x3A+0x02+0x01+0x01
Control mirror by UART and show mirror.

● 0x56+0x00+0x3A+0x02+0x00+0x00
Control mirror by GPIO.

1.3.2.16.MIRROR_STATUS

Command function ： get show status of sensor mirror.

Command format ：0x56+serial number+0x3B+0x00

Control mode：

0：control mirror by GPIO.

1：control mirror by UART.

Mirror mode：whether show mirror by UART, it is effective only with UART.It needs GPIO value to set with GPIO control.

0：do not show mirror

1：show mirror

Return format ：

0x76+serial number+0x3B+0x00+0x02+control mode(1 byte)+Mirror mode(1 byte)

E.g.

● 0x56+0x00+0x3B+0x00　　get mirror control mode and sensor mirror show status.

1.3.2.17.COLOR_CTRL

Command function ： color control mode and show mode

Command format ：0x56+serial number+0x3C+0x02+control mode(1 byte)+show mode(1 byte)

Control mode：

0：control color by GPIO.

1：control color by UART.

Show mode：show different color by UART, it is effective only with UART.It needs Mirror value to set with GPIO control.

0：automatically step black-white and colour.

1：manually step color, select colour.

2：manually step color, select black-white.

Return format ：

OK：0x76+serial number+0x3C+0x00+0x00

Error：0x76+serial number+0x3C+0x03+0x00

E.g.

● 0x56+0x00+0x3C+0x02+0x01+0x00

Control color by UART and automatically step black-white and colour.

● 0x56+0x00+0x3C+0x02+0x01+0x01

Control color by UART and select colour show.

● 0x56+0x00+0x3C+0x02+0x01+0x02

Control color by UART and select black-white show.

● 0x56+0x00+0x3C+0x02+0x00+0x00

Control color by GPIO.

1.3.2.18.COLOR_STATUS

Command function ： get color control mode and show mode

Command format ：0x56+serial number+0x3D+0x00

Control mode：

0：control color by GPIO.

1：control color by UART.

Show mode：show current color by UART.

0：automatically step black-white and colour.

1：manual step color, select colour.

2：manual step color, select black-white.

Return format ：

0x76+serial number+0x3D+0x00+0x03+control mode(1 byte)+show mode(1 byte)+current color(1 byte)

E.g.

● 0x56+0x00+0x3D+0x00　get color control mode and show mode

1.3.2.19.POWER_SAVE_CTRL

Command function ： whether select power-save mode
Command format ：0x56+serial number+0x3E+data-length+command-type(1 byte)+control item(multi-byte)
Command-type：
　　0：power-save control mode
　　1：power-save attribute configuration
Control item：
● When the command-type is 0：
　The **first** byte：
　　0：control by GPIO
　　1：control by UART
　The **second** byte：it is effective only with UART.It needs GPIO value to set with GPIO control.
　　0：stop power-save
　　1：start power-save
● When the command-type is 1：
　The **first** byte：power-save control mode
　　Bit[1:0]：select power-save mode
　　　2b'00：stop FBUF and relevant ouput
　　　2b'01：stop JPG and relevant output
　　Bit2：Whether the control mode and motion are relevant or not.If true, it will select power-save in a certain time until the motion is monitored.
　　　0：no relativity
　　　1：relativity
　　If the control mode and motion are relevant, it could not be controlled by GPIO.Or else,it will be based on GPIO standard.
　The **second** and **third** byte mean the time interval from no motion to starting power-save as power-save and motion are relevant.
Return format ：
OK：0x76+serial number+0x3E+0x00+0x00
Error：0x76+serial number+0x3E+0x03+0x00
E.g.
● 0x56+0x00+0x3E+0x03+0x00+0x01+0x01
　Start power-save by UART with power-save control command.
● 0x56+0x00+0x3E+0x03+0x00+0x01+0x00
　Stop power-save by UART with power-save control command.
● 0x56+0x00+0x3E+0x03+0x00+0x00+0x00
　Control power-save by GPIO with power-save control command.
● 0x56+0x00+0x3E+0x04+0x01+0x02+0x01+0x2C
　Power-save attribute configuration command.

1.3.2.20.POWER_SAVE_STATUS

Command function ： get current power-save status
Command format ：0x56+serial number+0x3F+0x01+command-type(1 byte)
Command-type：
0：power-save control mode
1：power-save attribute configuration
Control item：
● When the command-type is 0：
The **first** byte：
0：control by GPIO
1：control by UART
The **second** byte：it is effective only with UART.It needs GPIO value to set with GPIO control.
0：stop power-save
1：start power-save
● When the command-type is 1：
The **first** byte：power-save control mode
Bit[1:0]：select power-save mode
2b'00：stop FBUF and relevant output
2b'01：stop JPG and relevant output
Bit2：Whether the control mode and motion are relevant or not.If true, it will select power-save in a certain time until the motion is monitored.
0：no relativity
1：relativity
If the control mode and motion are relevant, it could not be controlled by GPIO.Or else,it will be based on GPIO standard.
The **second** and **third** byte mean the time interval from no motion to starting power-save as power-save and motion are relevant.
Return format ：
OK：0x76+serial number+0x3F+0x00+data-lengths+control item
Error：0x76+serial number+0x3F+0x03+0x00
E.g.
● 0x56+0x00+0x3F+0x01+0x00
Get current power-save status and control mode.
● 0x56+0x00+0x3F+0x01+0x01
Get power-save attribute configuration.

1.3.2.21.AE_CTRL

Command function ： control AE attribute
Command format ：0x56+serial number+0x40+0x03+AE attribute(1 byte)+control mode(1 byte)+control item(1byte)
AE attribute：
0：set frequency 50Hz or 60Hz

1：automatically step or forcibly step indoor

2：backlight compensation

control mode：

0：GPIO

1：UART

control item：it is effective only with UART.It needs GPIO values to set with GPIO control.

● When AE attribute is 0：

0：50Hz

1：60Hz

● When AE attribute is 1：

0：automatically step indoor and outdoor

1：forcibly step indoor

● When AE attribute is 2：

0：stop backlight compensation

1：start backlight compensation

Return format ：

OK：0x76+serial number+0x40+0x00+0x01

Error：0x76+serial number+0x40+0x03+0x00

E.g.

● 0x56+0x00+0x40+0x03+0x00+0x01+0x00

Set frequency of 50Hz by MCU UART.

● 0x56+0x00+0x40+0x03+0x00+0x01+0x01

Set frequency of 60Hz by MCU UART.

● 0x56+0x00+0x40+0x03+0x00+0x00+0x00

Set AE frequency by GPIO.

● 0x56+0x00+0x40+0x03+0x01+0x01+0x00

Step indoor and outdoor automatically by MCU UART.

● 0x56+0x00+0x40+0x03+0x01+0x01+0x01

Step indoor and outdoor by MCU UART, and forcibly step indoor.

● 0x56+0x00+0x40+0x03+0x01+0x00+0x00

Step indoor and outdoor by GPIO.

● 0x56+0x00+0x40+0x03+0x02+0x01+0x00

Stop backlight compensation by MCU UART.

● 0x56+0x00+0x40+0x03+0x02+0x01+0x01

Start backlight compensation by MCU UART.

● 0x56+0x00+0x40+0x03+0x02+0x00+0x00

Control backlight compensation by GPIO.

1.3.2.22.AE_STATUS

Command function ： get AE attribute

Command format ：0x56+serial number+0x41+0x01+AE attribute(1 byte)

AE attribute：

0：set frequency 50Hz or 60Hz

1：automatically step or forcibly step indoor

2：backlight compensation

control mode：

0：GPIO

1：UART

control item：it is effective only with UART.It needs GPIO values to set with GPIO control.

● When AE attribute is 0：

0：50Hz

1：60Hz

● When AE attribute is 1：

0：automatically step indoor and outdoor

1：forcibly step indoor

● When AE attribute is 2：

0：stop backlight compensation

1：start backlight compensation

Return format ：

OK： 0x76+serial number+0x41+0x00+0x03+AE attribute(1 byte)+control mode(1 byte)+control item(1byte)

Error：0x76+serial number+0x41+0x03+0x00

E.g.

● 0x56+0x00+0x41+0x01+0x00

Get AE attribute 0, control mode and configuration information.

● 0x56+0x00+0x41+0x01+0x01

Get AE attribute 1, control mode and configuration information.

● 0x56+0x00+0x41+0x01+0x02

Get AE attribute 2, control mode and configuration information.

1.3.2.23.MOTION_CTRL

Command function ： motion control

Command format ：0x56+serial number+0x42+data-lengths+motion attribute+control item

motion attribute：

0：motion control and enabling control

1：alarm-output attribute

2：alarm-output enabling control

3：alarm-output control

control item：

● motion control and enabling control

The **first** byte：

0：GPIO

1：UART

The **second** byte：

0：forbid motion monitoring

1：start motion monitoring

● alarm-output attribute

The **first** byte：

bit0：alarm type

0：stop alarming at a certain time.

1：alarm at all times.

Bit1：alarm electrical level

0：it is low level until alarm.

1：it is high level until alarm.

The second and third byte mean the alarm time, the lower byte follows the higher byte, the unit is 10 millisecond.

● alarm-output enabling control

The **first** byte：

0：forbid alarm-output

1：enable alarm-output

● alarm-output control

The **first** byte：

0：stop alarm-output

1：start alarm-output

Return format ：

OK： 0x76+serial number+0x42+0x00+0x00

Error：0x76+serial number+0x42+0x03+0x00

E.g.

● 0x56+0x00+0x42+0x03+0x00+0x01+0x01

Enable motion monitoring by MCU UART, and open it.

● 0x56+0x00+0x42+0x03+0x00+0x01+0x00

Enable motion monitoring by MCU UART, and stop it.

● 0x56+0x00+0x42+0x03+0x00+0x00+0x00

Enable motion monitoring by GPIO.

● 0x56+0x00+0x42+0x04+0x01+0x02+0x00+0x64

Set alarm-output attribute.

● 0x56+0x00+0x42+0x02+0x02+0x01

Enable alarm-output control.

● 0x56+0x00+0x42+0x02+0x02+0x00

Disallow alarm-output control.

● 0x56+0x00+0x42+0x02+0x03+0x01

Start alarm-output.

● 0x56+0x00+0x42+0x02+0x03+0x00

Stop alarm-output.

1.3.2.24.MOTION_STATUS

Command function ： get motion status

Command format ：0x56+serial number+0x43+0x01+motion attribute

motion attribute：

0：motion control and enabling control

1：alarm-output attribute

2：alarm-output enabling control

3：alarm-output control

control item：

● motion control and enabling control

The **first** byte：

0：GPIO

1：UART

The **second** byte：

0：forbid motion monitoring

1：start motion monitoring

● alarm-output attribute

The **first** byte：

bit0：alarm type

0：stop alarming at a certain time.

1：alarm at all times.

Bit1：alarm electrical level

0：it is low level until alarm.

1：it is high level until alarm.

The second and third byte mean the alarm time, the lower byte follows the higher byte, the unit is 10 millisecond.

● alarm-output enabling control

The **first** byte：

0：forbid alarm-output

1：enable alarm-output

● alarm-output control

The **first** byte：

0：stop alarm-output

1：start alarm-output

Return format ：

OK： 0x76+serial number+0x43+0x00+data-lengths+motion attribute+control item

Error：0x76+serial number+0x43+0x03+0x00

E.g.

● 0x56+0x00+0x43+0x01+0x00

Get current motion control and enabling control status.

● 0x56+0x00+0x43+0x01+0x01

Get current alarm-output attribute.

● 0x56+0x00+0x43+0x01+0x02

Get alarm-output enabling control status.

● 0x56+0x00+0x43+0x01+0x03

Whether started alarm-output control or not.

1.3.2.25.TV_OUT_CTRL

Command function ： control TV output
Command format ：0x56+serial number+0x44+0x01+control item(1 byte)
Control item：
0：stop TV output
1：start TV output
Return format ：
OK： 0x76+serial number+0x44+0x01+0x00
Error：0x76+serial number+0x44+0x03+0x00
E.g.
● 0x56+0x00+0x44+0x01+0x01 start TV output
● 0x56+0x00+0x44+0x01+0x00 stop TV output

1.3.2.26.OSD_ADD_CHAR

Command function ： add OSD characters to channels(channel 1)
Command format ：0x56+serial number+0x45+data-length+character number(1 byte)+starting address(1 byte)+characters(n characters)
character number：the number of characters which continuously are written to channels, the most is 14.
starting address：the starting place from which characters show. The format is as follows.
Bit[4-0]：Y-coordinate
Bit[6-5]：X-coordinate
Characters：the characters ready to show. It is VC0706 OSD characters.
Return format ：
OK： 0x76+serial number+0x45+0x00+0x00
Error：0x76+serial number+0x45+0x03+0x00
E.g.
● 0x56+0x00+0x45+0x07+0x06+0x24+0x1F+0x2C+0x30+0x2C+0x26+0x35+0x32
Write 7 characters, Vimicro, to channel on col 2 row 4 in turn.
VC0706 supports 80 characters, see the table below, every character has the index value of itself. For example, 0x00 is 0, 0x01 is 1, 0x10 is G,0x11 is H,etc.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
K	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	**a**	**b**	**c**	**d**
e	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**
y	**z**	**-**	**_**	**:**	**.**	/	*	**(**	**)**	**[**	**]**	@	**!**	+	|	\	#	■	

1.3.2.27.DOWNSIZE_SIZE

Command function ： control downsize attribute
Command format ：0x56+serial number+0x53+0x01+control item(1 byte)
control item：zooming image proportion
Bit[1:0]：width zooming proportion

2b'00：1:1, no zoom
2b'01：1:2, the proportion is 1/2.
2b'10：1:4, the proportion is 1/4.
2b'11：reservation
Bit[3:2]：height zooming proportion
2b'00：1:1, no zoom
2b'01：1:2, the proportion is 1/2.
2b'10：1:4, the proportion is 1/4.
2b'11：reservation

Notice：

1. The image width must be the multiple of 16 in FBUF, image height is the multiple of 8, so the configuration information could satisfy the condition.
2. The zooming proportion of image height is not more than the zooming proportion of width.

Return format ： 0x76+serial number+0x53+0x00+0x00

E.g.

● 0x56+0x00+0x53+0x01+0x05 the width and height will be the half of previous attribute.

1.3.2.28.DOWNSIZE_STATUS

Command function ： get downsize status

Command format ： 0x56+serial number+0x54+0x00

control item：zooming image proportion

Bit[1:0]：width zooming proportion
2b'00：1:1, no zoom
2b'01：1:2, the proportion is 1/2.
2b'10：1:4, the proportion is 1/4.
2b'11：reservation
Bit[3:2]：height zooming proportion
2b'00：1:1, no zoom
2b'01：1:2, the proportion is 1/2.
2b'10：1:4, the proportion is 1/4.
2b'11：reservation

Return format ： 0x76+serial number+0x54+0x00+0x01+control item(1 byte)

E.g.

● 0x56+0x00+0x54+0x00 get downsize configuration information.

1.3.2.29.GET_FLASH_SIZE

Command function ： get SPI Flash size

Command format ：0x56+serial number+0x60+0x00

Return format ：0x76+serial number+0x60+0x00+0x04+Flash size(4 bytes)

E.g.

● 0x56+0x00+0x60+0x00 get SPI Flash size.

1.3.2.30.ERASE_FLASH_SECTOR

Command function ： erase the certain sector in SPI Flash
Command format ：0x56+serial number+0x61+0x04+starting address(4 bytes)
Return format ：0x76+serial number+0x61+0x00+0x00
E.g.
● 0x56+0x00+0x61+0x04+0x00+0x01+0x00+0x00
erase the sector in SPI Flash address 0x10000.

1.3.2.31.ERASE_FLASH_ALL

Command function ： erase the whole SPI Flash
Command format ：0x56+serial number+0x62+0x00
Return format ：0x76+serial number+0x62+0x00+0x00
E.g.
● 0x56+0x00+0x62+0x00 erase the whole SPI Flash.

1.3.2.32.READ_LOGO

Command function ： read logo and show it
Command format ：0x56+serial number+0x70+0x06+logo-lengths(2 bytes)+starting address(4 bytes)
logo-length：the data-lengths of logo
starting address：the starting place to store the logo data.
Return format ：0x76+serial number+0x70+0x00+0x00
After reading OSD from control information, system will return the command.
E.g.
● 0x56+0x00+0x70+0x06+0x02+0x00+0x00+0x00+0x80+0x00
Begin with the address 0x8000, read logo data to OSD channel and show it.The data-lengths are 0x0200.

1.3.2.33.SET_BITMAP

Command function ： control bitmap and read it
Command format ：0x56+serial number+0x71+data-lengths+control-type(1 byte)+bitmap-lengths(2 bytes)+starting address(4 bytes)
control-type：
1：stop data-transfer and display of bitmap from SPI memory(SPI EEPROM or SPI FLASH)
2：start data-transfer and display of bitmap from SPI memory(SPI EEPROM or SPI FLASH)
bitmap-length：the lengths of bitmap ready to transfer.
starting address：the starting place to transfer the bitmap data in SPI memory. It is effective only when **control-type** is 2.
Return format ：

OK： 0x76+serial number+0x71+0x00+0x00
Error：0x76+serial number+0x71+0x03+0x00
E.g.
● 0x56+0x00+0x71+0x01+0x01
Stop bitmap show.
● 0x56+0x00+0x71+0x07+0x02+0x25+0x80+0x00+0x00+0x40+0x00
Begin with the address 0x4000, read bitmap data.The data-lengths are 0x2580. Then show it.
Once start data-transfer and display of bitmap from SPI memory, it can not read from SPI memory or write to SPI memory. Or else, you must stop bitmap show firstly.

2.Take Photo By MCU UART

2.1.About frame buffer

In this text, FBUF is frame buffer. VC0706 supports two frame buffers, current frame and next frame.

2.1.Operation steps

When the external MCU read images from VC0706 and write images data to FBUF of VC0706 by SPI, we need to follow the stated operation steps.

2.1.1.Read images from VC0706

● Send FBUF_CTRL command to stop current frame updating, the parameter is 0x00.
● Send GET_FBUF_LEN command to get image lengths in FBUF.
● Send READ_FBUF to read image data, the parameters are as follows in READ_FBUF command.
■ the FBUF frame type is 0x00
■ control mode is 0x0F
■ starting address is 0x00
■ the image data is the one that we get with GET_FBUF_LEN command.
■ delay time is used to prolong the time between image data and return command, the default value is 3000 , and can be modified.
● After sending READ_FBUF command, wait for responding, if execute right, receive the image data from VC0706.When the data sending has finished, VC0706 will send the return command secondly to the external MCU.
● After all have finished, we need to send FBUF_CTRL command to resume frame, the parameter is 0x02.

2.1.2.Write image data to VC0706

● Send FBUF_CTRL command to stop all frames updating, the parameter is 0x02.
● Send WRITE_FBUF command to write image data, the parameters are as follows in

WRITE_FBUF command.

■ control mode is 0x0F

■ starting address is 0x00

■ the image data is the one that be ready to write.

■ delay time is used to prolong the time between image data and return command, the default value is 3000 , and can be modified.

● After sending WRITE_FBUF command, wait for responding, if execute right, it will send the image data to VC0706.When the data sending has finished, VC0706 will send the return command secondly to the external MCU.

● Send SET_FBUF_LEN command to set image lengths, the parameter is the lengths of sending image.

● Send FBUF_CTRL command to step frame to current frame and show the image, the parameter is 0x03.