VC0703

TV Camera Processor

使用指南

(UART Interface)

Preliminary†

Version 0.9

November 5, 2007

Notes1: The information is subject to change without notice. Before using this document, please confirm that this is the latest version.
Notes2: Not all products and/or types are available in every country. Please check with a Vimicro sales representative for availability and additional information.

Important Notice

Vimicro and its subsidiaries reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. Customers should contact Vimicro's sales department before purchasing the product described in this document. All products are sold subject to Vimicro's terms and conditions of sale supplied at the time of order acknowledgment.

Vimicro does not warrant or represent that any license, either explicit or implied, is granted under any Vimicro patent right, copyright, mask work right, or other Vimicro intellectual property right relating to any combination, machine, or process in which Vimicro products or services are used. Information published by Vimicro regarding third-party products or services does not constitute a license from Vimicro to use such products or service or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from Vimicro under the patents or other intellectual property of Vimicro.

Vimicro semiconductor devices are intended for standard uses (such as office equipment, computers, industrial/communications/measuring equipment, and personal/home equipment). Customers using semiconductor devices for special applications (including aerospace, nuclear, military and medical applications) in which a failure or malfunction might endanger life or limb and which require extremely high reliability must contact our Sales Department first. If damage is caused by such use of our semiconductor devices without first consulting our Sales Department, Vimicro will not assume any responsibility for the loss.

The contents of this document must not be reprinted or duplicated without written permission of Vimicro. Information and circuit diagrams in this document are only examples of device application. They are not intended to be used in actual equipment. Vimicro accepts no responsibility for infringement of patents or other rights owned by third parties caused by use of the information and circuit diagrams in this document.

Reproduction of information in Vimicro data books or data sheets is permissible only if preproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. Vimicro is not responsible or liable for such altered documentation. Resale of Vimicro products or services with statements different from or beyond the parameters stated by Vimicro for that product or service voids all explicit and any implied warranties for the associated Vimicro product or service and is an unfair and deceptive business practice. Vimicro is not responsible or liable for any such statements.

Table of Content

1. 简介

本文档就 VC0703 的 UART (Universal Asynchronous Receiver/Transmitter) 使用加以说明，以帮助用户在其设计中通过 UART 与 VC0703 进行通信。

为获知 VC0703 芯片的基本说明，请参考 *VC0703 Datasheet*.

本文档以运行调试工具的 PC 或者外部主控 MCU 为发送端，以 VC0703 为接收端。

2. UART 接口

2.1. 接口概要

VC0703 的 UART 接口是基于标准 UART 接口设计的，可以用于连接 PC 或者外部主控 MCU 的串行通信端口。VC0703 支持 RXD，TXD，但是不支持所有与 Modem 功能相关的引脚。

通过串口通讯协议可以获取 VC0703 的信息，并对其进行控制。

2.2. 通信协议

串口通讯协议格式如下所示：

接收的命令格式：协议标志（1 字节）+序列号（1 字节）+命令字（1 字节）+数据长度（1 字节）+数据（0~16 字节）

回复的命令格式：协议标志（1 字节）+序列号（1 字节）+命令字（1 字节）+状态字节（1 字节）+数据长度（1 字节）+数据（0~16 字节）；

其中：

协议标志： 用于标志该协议为 VC0703 串口通讯协议；接收的协议标志为 0x56（‘V’），回复的协议标志为 0x76（‘v’）。

序列号： 用于在多个设备同时连接时区分各个设备，只有串口命令的序列号与设备的序列号相同，设备才会接受这些命令。序列号的取值范围为 0~255。

命令字： 用于标识具体的串口通讯命令。

数据长度： 表示后面数据的长度，不包括协议标志、序列号、命令字和数据长度。取值范围为 0~16。

数据： 命令使用到的数据，不同的命令，数据的长度和格式有所不同，最大为 16 字节。

状态字节： 对于控制命令、数据写命令，数据的第一个字节表示命令是否成功，0 表示成功。0x01~0xFF 表示失败。

状态类型含义如下所示：

状态代码	错误说明
0	命令执行正确
1	非法的命令字，表示命令字系统不支持

2	非法的数据长度，表示该命令不支持该长度的数据长度值
3	数据格式非法
4	当前的系统状态下命令无法执行
5	命令接受，但是命令执行失败

说明：

- 对于多字节数据类型，数据排列方式为高位数据在前，低位数据在后。
- 如果是序列号错误，则系统不返回任何错误报告。
- 命令中的数据的长度最大为 16 字节。
- 如果命令格式或者命令执行失败，则返回的格式固定为 1 字节的错误代码，0 字节数据长度。

2.3. 通信命令

2.3.1. 命令列表

命令定义	命令字	命令含义
GET_VERSION	0x11	获取 Firmware 程序版本
SET_SERIAL_NUMBER	0x21	设置序列号
SET_PORT	0x24	设置串口属性
SYSTEM_RESET	0x26	系统复位
READ_DATA	0x30	读取设备的寄存器
WRITE_DATA	0x31	向设备寄存器写入数据
COMM_MOTION_CTRL	0x37	启动或停止串口 Motion 监测
COMM_MOTION_STATUS	0x38	获取串口 Motion 监测状态
COMM_MOTION_DETECTED	0x39	启动了串口 Motion 检测时，用于串口发送检测到了 Motion
MIRROR_CTRL	0x3A	设置图像是否要 Mirror 显示
MIRROR_STATUS	0x3B	获取当前图像是否 Mirror 的状态
COLOR_CTRL	0x3C	颜色控制
COLOR_STATUS	0x3D	获取当前颜色控制模式和当前的颜色
POWER_SAVE_CTRL	0x3E	控制进入和退出节能模式
POWER_SAVE_STATUS	0x3F	检查是否进入节能模式
AE_CTRL	0x40	设置 AE 相关属性
AE_STATUS	0x41	获取 AE 相关属性
MOTION_CTRL	0x42	Motion 控制
MOTION_STATUS	0x43	获取 Motion 状态
TV_OUT_CTRL	0x44	控制打开和关闭视频输出
OSD_ADD_CHAR	0x45	添加 OSD 字符
GET_FLASH_SIZE	0x60	获取 SPI Flash 的大小
ERASE_FLASH_SECTOR	0x61	擦除 Flash 的一个块
ERASE_FLASH_ALL	0x62	擦除整个 Flash
READ_LOGO	0x70	读取 LOGO 并显示
SET_BIZTMAP	0x71	对 Bitmap 进行操作
BATCH_ WRITE	0x80	批量写数据

2.3.2. 命令详细说明

2.3.2.1. GET_VERSION

命令功能： 获取 Firmware 版本号

命令格式： 0x56+序列号+0x11+0x00

回复格式： 0x76+序列号+0x11+0x00+0x0B+"VC0703 1.00"

说明：

版本号的格式为：VC0703+空格+主版本号(1 字节)+'.'+次版本号(2 字节)；共 11 个字节。

版本号为字符串格式，如：VC0703 1.00。

2.3.2.2. SET_SERIAL_NUMBER

命令功能： 设置设备序列号

命令格式： 0x56+序列号+0x21+0x01+新序列号

回复格式： 0x76+序列号+0x21+0x00+0x00

系统在回复命令时还是使用旧的序列号，在回复完命令后改为新的序列号。

举例： 0x56+0x00+0x21+0x01+0x10 将设备序列号从 0x00 改为 0x10

说明：

设置了新的序列号之后，以后再发送命令时，需要使用新的序列号。

2.3.2.3. SET_PORT

命令功能： 设置串口的属性

命令格式： 0x56+序列号+0x24+数据长度+设备类型（1 字节）+配置数据

MUC 串口： 0x56+序列号+0x24+0x03+0x01+S1RELH(1 字节)+S1RELL(1 字节)

设备类型：

0x01：表示 MCU 串口；

回复格式：

如果运行正确，则回复：0x76+序列号+0x24+0x00+0x00

如果设备类型设置错误，则回复：0x76+序列号+0x24+0x03+0x00

举例： 0x56+0x00+0x24+0x03+0x01+0x0D+0xA6 设置 MCU 串口波特率为 115200

说明：

MCU 串口波特率：

- S1RELH 和 S1RELL 表示要写入 S1RELH 和 S1RELL 寄存器的值。
- 波特率与波特率寄存器的关系如下所示：

波特率	S1RELH	S1RELL
9600	0xAE	0xC8
19200	0x56	0xE4
38400	0x2A	0xF2
57600	0x1C	0x4C
115200	0x0D	0xA6

2.3.2.4. SYSTEM_RESET

命令功能： 系统复位命令，用于复位整个系统软件和硬件。

命令格式： 0x56+序列号+0x26+0x00

回复格式： 0x76+序列号+0x26+0x00+0x00

举例： 0x56+0x00+0x26+0x00 系统复位

说明：

返回回复后延迟大约 10 毫秒后复位整个系统，VC0703 系统复位后会通过 UART 返回一些基础配置信息到命令发送端。

2.3.2.5. READ_DATA

命令功能： 读取 chip 寄存器、Sensor 寄存器、I2C EEPROM、SPI 设备或者 CCIR656 设备的数据

命令格式： 0x56+序列号+0x30+数据长度+设备类型（1 字节）+要读取的数据个数(1 字节)+配置信息

设备类型：要访问的设备的类型；

1：Chip 寄存器

2：Sensor 寄存器

3：CCIR656 寄存器

4：I2C EEPROM

5：SPI EEPROM

6：SPI Flash

不同设备的命令格式如下所示：

Chip 寄存器： 0x56+序列号+0x30+0x04+0x01+读取数据个数+寄存器地址(2 字节)

Sensor 寄存器：0x56+序列号+0x30+0x05+0x02+读取数据个数+寄存器数据宽度(1 字节)+寄存器地址(2 字节)

CCIR656 设备：0x56+序列号+0x30+0x05+0x03+读取数据个数+寄存器数据宽度(1 字节)+寄存器地址(2 字节)

I2C EEPROM： 0x56+序列号+0x30+0x04+0x04+读取数据个数+数据地址(2 字节)

SPI EEPROM： 0x56+序列号+0x30+0x04+0x05+读取数据个数+数据地址(2 字节)

SPI Flash： 0x56+序列号+0x30+0x06+0x06+读取数据个数+数据地址(4 字节)

读取的数据个数：表示从指定的地址开始，连续读取的数据的个数。读取的数据个数有限制，“读取的数据个数”x“数据宽度”不能大于 16 个字节，否则可能会导致系统出错。

对于 Sensor 和 CCIR656 设备，寄存器的数据宽度可能不同，寄存器宽度表示每个寄存器的数据宽度，取值为 1-3 字节，用来表示后面的寄存器数据。数据的个数为：前面的数据个数*寄存器宽度。

回复格式：

如果运行正确，则回复：0x76+序列号+0x30+0x00+读取数据个数+读取的寄存器的值

如果命令中的设备类型不对或者当操作 Sensor 和 CCIR656 时，寄存器地址宽度与系统中设置的不同，则回复：0x76+序列号+0x30+0x03+0x00

回复命令中包含的数据个数为：“要读取的数据个数” x“数据宽度”。

举例：

- 0x56+0x00+0x30+0x04+0x01+0x02+0x18+0x00

  从 Chip 寄存器的地址 0x1800 开始读取 2 个字节的数据

- 0x56+0x00+0x30+0x05+0x02+0x01+0x01+0x00+0x20

  从 Sensor 寄存器的地址 0x0020 开始读取 1 个字节的数据（Sensor 寄存器的数据宽度为 1 字节）

- 0x56+0x00+0x30+0x05+0x02+0x02+0x02+0x00+0x20
  从 Sensor 寄存器的地址 0x0020 开始读取 2 个字的数据（Sensor 寄存器的数据宽度为 2 字节）
- 0x56+0x00+0x30+0x05+0x03+0x03+0x01+0x00+0x40
  从 CCIR656 寄存器的地址 0x0040 开始读取 3 个字节的数据（CCIR656 寄存器的数据宽度为 1 字节）
- 0x56+0x00+0x30+0x04+0x04+0x04+0x02+0x40
  从 I2C EEPROM 的地址 0x0240 开始读取 4 个字节的数据
- 0x56+0x00+0x30+0x04+0x05+0x03+0x08+0x00
  从 SPI EEPROM 的地址 0x0800 开始读取 3 个字节的数据
- 0x56+0x00+0x30+0x06+0x06+0x08+0x00+0x20+0x40+0x60
  从 SPI Flash 的地址 0x00204060 开始读取 8 个字节的数据

说明：

2.3.2.6. WRITE_DATA

命令功能： 向 chip 寄存器、Sensor 寄存器、I2C 的 EEPROM、SPI 设备或者 CCIR 设备写入数据

命令格式： 0x56+序列号+0x31+数据长度+设备类型（1 字节）+要写入的数据个数(1 字节)+配置信息+数据

设备类型：要访问的设备的类型；

1：Chip 寄存器
2：Sensor 寄存器
3：CCIR656 寄存器
4：I2C EEPROM
5：SPI EEPROM
6：SPI Flash

不同设备的命令格式如下所示：

Chip 寄存器： 0x56+序列号+0x31+数据长度+0x01+写入的数据个数+寄存器地址(2 字节)+数据

Sensor 寄存器： 0x56+序列号+0x31+数据长度+0x02+写入的数据个数+寄存器宽度(1 字节)+寄存器地址(2 字节)+数据

CCIR656 设备： 0x56+序列号+0x31+数据长度+0x03+写入的数据个数+寄存器宽度(1 字节)+寄存器地址(2 字节)+数据

I2C EEPROM： 0x56+序列号+0x31+数据长度+0x04+写入的数据个数数+数据地址(2 字节)+数据

SPI EEPROM： 0x56+序列号+0x31+数据长度+0x05+写入的数据个数数+数据地址(2 字节)+数据

SPI 设备： 0x56+序列号+0x31+数据长度+0x06+写入的数据个数+数据地址(4 字节)+数据

要写入的数据个数：表示从指定的地址开始，连续写入的数据的个数。一次写入的数据个数有限制，写入的数据个数受限于命令中数据最多只能 16 个字节的限制。

对于 Sensor 和 CCIR656 设备进行写操作时，如果其数据宽度大于 1 字节，则每个数据中多个字节的存放方式为高位数据在前，低位数据在后。

对于 SPI 设备（SPI EEPROM 和 SPI Flash）来说，写入操作的地址不能跨越 2 个 Page，否则会导致写入错误；不同的 SPI 设备的 Page 大小不同，在对 SPI 设备进行多字节写操作是需要多加注意。

回复格式：

如果运行正确，则回复：0x76+序列号+0x31+0x00+0x00

如果命令中的设备类型不对或者当操作 Sensor 和 CCIR656 时，寄存器地址宽度与系统中设置的不同，则回复：0x76+序列号+0x31+0x03+0x00

举例：

- 0x56+0x00+0x31+0x06+0x01+0x02+0x18+0x00+0x11+0x22

  向 Chip 寄存器从 0x1800 开始的地址依次写入 0x11、0x22，共 2 个字节的数据
- 0x56+0x00+0x31+0x06+0x02+0x01+0x01+0x00+0x20+0x80

  向 Sensor 寄存器的 0x0020 地址写入 0x80，共 1 个字节的数据（Sensor 寄存器的数据宽度为 1 字节）
- 0x56+0x00+0x31+0x09+0x02+0x02+0x02+0x00+0x20+0x01+0x00+0x02+0x00

  向 Sensor 寄存器从 0x0020 开始的地址写入 0x0100、0x0200，共 2 个字的数据（Sensor 寄存器的数据宽度为 2 字节）
- 0x56+0x00+0x31+0x06+0x03+0x03+0x01+0x00+0x20+0x41+0x42+0x43

  向 CCIR656 寄存器从 0x0020 开始的地址写入 0x41、0x42、0x43，共 3 个字节的数据（CCIR656 寄存器的数据宽度为 1 字节）
- 0x56+0x00+0x31+0x08+0x04+0x04+0x02+0x40+0x11+0x12+0x13+0x14

  向 I2C EEPROM 从 0x0240 开始的地址依次写入 0x11、0x12、0x13、0x14，共 4 个字节的数据
- 0x56+0x00+0x31+0x07+0x05+0x03+0x02+0x40+0x21+0x22+0x23

  向 SPI EEPROM 从 0x0240 开始的地址依次写入 0x21、0x22、0x23，共 3 个字节的数据
- 0x56+0x00+0x31+0x0D+0x06+0x07+0x00+0x20+0x40+0x60+0x30+0x31+0x32+0x33+0x34+0x35+0x36 向 SPI Flash 从 0x00204060 开始的地址依次写入 0x30、0x31、0x32、0x33、0x34、0x35、0x36，共 7 个字节的数据

说明：

2.3.2.7. COMM_MOTION_CTRL

命令功能： 用于控制是否打开串口的 Motion 监测

命令格式： 0x56+序列号+0x37+0x01+控制标志(1 字节)

控制标志：表示是否打开 Motion 监测;

0：表示关闭

1：表示打开

回复格式：

如果运行正确，则回复：0x76+序列号+0x37+0x00+0x00

如果控制标志设置错误，则回复：0x76+序列号+0x37+0x03+0x00

举例：

0x56+0x00+0x37+0x01+0x01	打开串口 Motion 监测
0x56+0x00+0x37+0x01+0x00	关闭串口 Motion 监测

说明：

打开串口的 Motion 检测后，如果检测到了 Motion，系统就会通过串口发送

COMM_MOTION_DETECTED 命令。

2.3.2.8. COMM_MOTION _STATUS

命令功能： 获取当前串口 Motion 监测是否打开

命令格式： 0x56+序列号+0x38+0x00

回复格式：

如果运行正确，则回复：0x76+序列号+0x38+0x00+0x01+控制标志(1 字节)

控制标志：表示是否打开 Motion 监测；

0：表示关闭

1：表示打开

举例：

0x56+0x00+0x38+0x00 获取当前串口 Motion 监测的状态

说明：

2.3.2.9. COMM_MOTION _DETECTED

命令功能： 表示串口监测到了 Motion

命令格式：

本命令用于在打开了串口监测后，当系统检测到 Motion 后，通过串口发送该命令通知监控程序。

回复格式： 0x76+序列号+0x39+0x00

举例：

0x76+0x00+0x39+0x00 检测到了 Motion

说明：

该命令是系统主动发送给控制端的。

2.3.2.10. MIRROR_CTRL

命令功能： 控制 Sensor 的 Mirror 和非 Mirror 显示

命令格式： 0x56+序列号+0x3A+0x02+控制模式(1 字节)+Mirror 模式(1 字节)

控制模式：用于表示通过串口还是通过 GPIO 来进行 Mirror 的控制；

0：表示通过 GPIO

1：表示通过串口

Mirror 模式：在通过串口控制时，该项表示是 Mirror 显示还是非 Mirror 显示；

0：表示非 Mirror

1：表示 Mirror 显示

该控制项只在控制模式为通过串口控制时有效，通过 GPIO 控制时，则是根据控制 Mirror 的 GPIO 的值来设置。

回复格式：

如果运行正确，则回复：0x76+序列号+0x3A+0x00+0x00

如果参数错误，则回复：0x76+序列号+0x3A+错误代码+0x00

举例：

0x56+0x00+0x3A+0x02+0x01+0x00 设置为通过串口控制 Mirror，并且设置为非 Mirror 显示

0x56+0x00+0x3A+0x02+0x01+0x01 设置为通过串口控制 Mirror，并且设置为 Mirror 显示

0x56+0x00+0x3A+0x02+0x00+0x00 设置为通过 GPIO 控制 Mirror

说明：

2.3.2.11. MIRROR_STATUS

命令功能： 获取 Sensor 是否 Mirror 显示，以及 Mirror 的控制方式

命令格式： 0x56+序列号+0x3B+0x00

回复格式： 0x76+序列号+0x3B+0x00+0x02+控制模式(1 字节)+Mirror 模式(1 字节)

控制模式：用于表示 Mirror 是通过串口还是通过 GPIO 来控制；

0：表示通过 GPIO

1：表示通过串口

Mirror 模式：表示当前是 Mirror 显示还是非 Mirror 显示；

0：表示非 Mirror

1：表示 Mirror 显示

举例：

0x56+0x00+0x3B+0x00 获取当前 Mirror 控制方式和 Mirror 状态

说明：

2.3.2.12. COLOR_CTRL

命令功能： 设置颜色的控制方式和颜色显示方式

命令格式： 0x56+序列号+0x3C+0x02+控制模式(1 字节)+颜色模式(1 字节)

控制模式：用于表示通过串口还是通过 GPIO 来进行颜色的控制；

0：表示通过 GPIO

1：表示通过串口。

颜色模式：在通过串口控制时，该项表示是颜色显示模式；

0：表示自动黑白彩色切换

1：表示手动颜色切换，并且彩色显示

2：表示手动颜色切换，并且黑白显示

该控制项只在控制模式为通过串口控制时有效，通过 GPIO 控制时，则是根据控制颜色的 GPIO 的值来设置。

回复格式：

如果运行正确，则回复：0x76+序列号+0x3C+0x00+0x00

如果参数错误，则回复：0x76+序列号+0x3C+0x03+0x00

举例：

0x56+0x00+0x3C+0x02+0x01+0x00 设置为通过串口控制颜色，并且设置自动黑白彩色切换

0x56+0x00+0x3C+0x02+0x01+0x01 设置为通过串口控制颜色，并且设置彩色显示

0x56+0x00+0x3C+0x02+0x01+0x02 设置为通过串口控制颜色，并且设置黑白显示

0x56+0x00+0x3C+0x02+0x00+0x00 设置为通过 GPIO 控制颜色

说明：

2.3.2.13. COLOR_STATUS

命令功能： 获取颜色的控制方式和颜色显示方式

命令格式： 0x56+序列号+0x3D+0x00

回复格式： 0x76+序列号+0x3D+0x00+0x03+控制模式(1 字节)+颜色模式(1 字节)+当前颜色(1 字节)

控制模式：用于当前是通过串口还是通过 GPIO 来进行颜色的控制；

0：表示通过 GPIO

1：表示通过串口。

颜色模式：表示当前的颜色显示模式；

0：表示自动黑白彩色切换

1：表示手动颜色切换，并且彩色显示

2：表示手动颜色切换，并且黑白显示

举例：

0x56+0x00+0x3D+0x00 获取当前的颜色控制方式和颜色状态

说明：

2.3.2.14. POWER_SAVE_CTRL

命令功能： 控制是否进入节能模式

命令格式： 0x56+序列号+0x3E+数据长度+命令类型（1 字节）+控制项（多字节）

命令类型：表示命令类型；

0：节能模式控制

1：节能模式属性配置

控制项：

- 当命令为节能模式控制时：

  第 1 个字节用于表示通过串口还是通过 GPIO 来进行节能模式的控制；

  0：表示通过 GPIO

  1：表示通过串口。

  第 2 个字节用于表示是否进入 Power Save 状态；

  0：表示退出 Power Save 状态

  1：表示进入 power save 状态

  该控制项只在控制模式为通过串口控制时有效，通过 GPIO 控制时，则是根据控制节能模式的 GPIO 的值来设置。

- 当命令为节能模式属性配置：

  第 1 个字节表示节能模式控制；

  Bit[1:0]：节能方式选择；

  2b’00：关闭 FBUF 后面相关模块和输出；

  2b’01：关闭 JPG 以及输出的相关模块；

  Bit2： 控制节能模式是否与 Motion 联动，如果联动，则在没有 Motion 后指定时间进入节能模式，并且在监测到 Motion 的时候退出节能模式；

  0：节能模式跟 Motion 不联动；

1：节能模式跟 Motion 联动；

如果要设置节能模式与 Motion 联动，则节能模式控制不能设置为 GPIO 口控制，否则将以 GPIO 的控制为准。

第 2，3 字节表示在节能模式跟 Motion 联动时，从没有 Motion 到进入节能模式的时间间隔，单位为 10 毫秒。

回复格式：

如果运行正确，则回复：0x76+序列号+0x3E+0x00+0x00

如果参数错误，则回复：0x76+序列号+0x3F+0x03+0x00

举例：

0x56+0x00+0x3E+0x03+0x00+0x01+0x01　　节能模式控制命令，通过串口控制进入节能模式

0x56+0x00+0x3E+0x03+0x00+0x01+0x00　　节能模式控制命令，通过串口控制退出节能模式

0x56+0x00+0x3E+0x03+0x00+0x00+0x00　　节能模式控制命令，通过 GPIO 控制节能模式

0x56+0x00+0x3E+0x04+0x01+0x02+0x01+0x2C　　节能模式属性设置命令

说明：

2.3.2.15. POWER_SAVE _STATUS

命令功能： 获取当前节能模式状态

命令格式： 0x56+序列号+0x3F+0x01+命令类型（1 字节）

命令类型：表示命令类型；

0：节能模式控制

1：节能模式属性配置

回复格式：

如果运行正确，则回复：0x76+序列号+0x3F+0x00+数据长度+控制项

控制项：

- 当命令为节能模式控制时：

  第 1 个字节用于表示当前通过串口还是通过 GPIO 来进行节能模式的控制；

  0：表示通过 GPIO

  1：表示通过串口。

  第 2 个字节用于表示当前 Power Save 状态；

  0：未处于节能模式

  1：处于节能模式

- 当命令为节能模式属性配置：

  第 1 个字节表示当前的节能模式控制；

  Bit[1:0]：节能方式选择；

  2b’00：关闭 FBUF 后面相关模块和输出；

  2b’01：关闭 JPG 以及输出的相关模块；

  Bit2：控制节能模式是否与 Motion 联动，如果联动，则在没有 Motion 后指定时间进入节能模式，并且在监测到 Motion 的时候退出节能模式；

0：节能模式跟 Motion 不联动；

1：节能模式跟 Motion 联动；

如果要设置节能模式与 Motion 联动，则节能模式控制不能设置为 GPIO 口控制，否则将以 GPIO 的控制为准。

第 2，3 字节表示在节能模式跟 Motion 联动时，从没有 Motion 到进入节能模式的时间间隔，单位为 10 毫秒。

如果参数失败，则回复：0x76+序列号+0x3F+0x03+0x00

举例：

0x56+0x00+0x3F+0x01+0x00　　获取节能模式控制方式和节能模式状态

0x56+0x00+0x3F+0x01+0x01　　获取节能模式属性配置

说明：

2.3.2.16. AE_CTRL

命令功能： AE 属性控制

命令格式： 0x56+序列号+0x40+0x03+AE 属性（1 字节）+控制模式（1 字节）+模式设置（1 字节）

AE 属性：要进行控制的 AE 属性；

0：进行 50Hz/60Hz 设置

1：进行自动切换/强制户内设置

2：进行背光补偿设置

控制模式：表示通过串口还是通过 GPIO 来进行该项 AE 属性的控制；

0：GPIO 控制

1：串口控制

模式设置：表示对要设置的 AE 属性的配置；

- 当进行 50Hz/60Hz 设置时：表示要设置为 50Hz 还是 60Hz；
  - 0：50Hz
  - 1：60Hz
- 当进行自动切换/强制户内设置时：表示要设置的切换方式；
  - 0：自动户内户外切换
  - 1：强制户内模式
- 当进行背光补偿设置时：表示是否要打开背光补偿；
  - 0：关闭背光补偿
  - 1：打开背光补偿

该控制项只在控制模式为通过串口控制时有效，通过 GPIO 控制时，则是根据控制节能模式的 GPIO 的值来设置。

回复格式：

如果运行正确，则回复：0x76+序列号+0x40+0x00+0x01

如果参数设置错误，则回复：0x76+序列号+0x40+0x03+0x00

举例：

0x56+0x00+0x40+0x03+0x00+0x01+0x00　　设置为串口控制 AE 50Hz/60Hz 切换，并设置为

50Hz

0x56+0x00+0x40+0x03+0x00+0x01+0x01　　设置为串口控制 AE 50Hz/60Hz 切换，并设置为 60Hz

0x56+0x00+0x40+0x03+0x00+0x00+0x00　　设置为 GPIO 控制 AE 50Hz/60Hz 切换

0x56+0x00+0x40+0x03+0x01+0x01+0x00　　设置为串口控制户内户外切换，并设置为自动户内户外切换

0x56+0x00+0x40+0x03+0x01+0x01+0x01　　设置为串口控制户内户外切换，并设置为强制户内

0x56+0x00+0x40+0x03+0x01+0x00+0x00　　设置为 GPIO 控制户内户外切换

0x56+0x00+0x40+0x03+0x02+0x01+0x00　　设置为串口控制背光补偿，并设置为打开背光补偿

0x56+0x00+0x40+0x03+0x02+0x01+0x01　　设置为串口控制背光补偿，并设置为关闭背光补偿

0x56+0x00+0x40+0x03+0x02+0x00+0x00　　设置为 GPIO 控制背光补偿

说明：

2.3.2.17. AE_STATUS

命令功能： 获取指定的 AE 属性的配置

命令格式： 0x56+序列号+0x41+0x01+AE 属性（1 字节）

AE 属性：要进行控制的 AE 属性；

0：进行 50Hz/60Hz 设置

1：进行自动切换/强制户内设置

2：进行背光补偿设置

回复格式：

如果运行正确，则回复：0x76+序列号+0x41+0x00+0x03+ AE 属性（1 字节）+控制模式（1 字节）+模式设置（1 字节）

AE 属性：当前要进行控制的 AE 属性；

0：进行 50Hz/60Hz 设置

1：进行自动切换/强制户内设置

2：进行背光补偿设置

控制模式：表示当前是通过串口还是通过 GPIO 来进行该项 AE 属性的控制；

0：GPIO 控制

1：串口控制

模式设置：表示对要设置的 AE 属性的当前的配置；

- 当进行 50Hz/60Hz 设置时：表示当前为 50Hz 还是 60Hz；
  - 0：50Hz
  - 1：60Hz
- 当进行自动切换/强制户内设置时：表示当前的切换方式；
  - 0：自动户内户外切换
  - 1：强制户内模式

- 当进行背光补偿设置时：表示当前背光补偿是否打开；
  0：关闭背光补偿
  1：打开背光补偿

如果参数错误，则回复：0x76+序列号+0x41+0x03+0x00

举例：

0x56+0x00+0x41+0x01+0x00	获取 AE 50Hz/60Hz 控制方式和当前的状态
0x56+0x00+0x41+0x01+0x01	获取户内户外切换控制方式和当前的状态
0x56+0x00+0x41+0x01+0x02	获取背光补偿控制方式和当前的状态

说明：

2.3.2.18. MOTION_CTRL

命令功能： Motion 控制

命令格式： 0x56+序列号+0x42+数据长度+Motion 控制属性+控制项

Motion 控制属性：Motion 控制命令；
0：Motion 控制方法及使能控制
1：设置报警输出属性
2：报警输出使能控制
3：报警输出控制

控制项：表示对要设置的 Motion 控制属性的配置；

- 表示 Motion 控制方法及使能控制
  第一个字节表示 GPIO 控制还是串口控制；
  0：GPIO 控制
  1：串口控制
  第二个字节表示如果为串口控制的话，要设置的 Motion 使能状态；
  0：禁止 Motion 检测
  1：打开 Motion 检测
- 表示报警输出属性设置
  第一个字节表示报警方式：
  Bit0：报警方式；
  0：报警指定时间后，停止报警；
  1：一直报警；
  Bit1：报警电平
  0：平时输出低电平输出，报警时输出高电平；
  1：平时输出高电平输出，报警时输出低电平；
  第二三字节表示报警时间：数据存放方式为高位在前，低位在后，单位为 10ms
- 表示报警输出使能控制
  第一个字节表示打开还是关闭报警输出功能；
  0：禁止报警输出
  1：使能报警输出
- 表示报警输出控制

第一个字节表示打开还是关闭报警输出；

0：停止报警输出

1：启动报警输出

回复格式：

如果运行正确，则回复：0x76+序列号+0x42+0x00+0x00

如果参数错误，则回复：0x76+序列号+0x42+0x03+0x00

举例：

0x56+0x00+0x42+0x03+0x00+0x01+0x01 设置为串口控制 Motion 检测的使能，并且打开 Motion 检测

0x56+0x00+0x42+0x03+0x00+0x01+0x00 设置为串口控制 Motion 检测的使能，并且关闭 Motion 检测

0x56+0x00+0x42+0x03+0x00+0x00+0x00 设置为 GPIO 控制 Motion 检测的使能

0x56+0x00+0x42+0x04+0x01+0x02+0x00+0x64 设置报警输出属性

0x56+0x00+0x42+0x02+0x02+0x01 设置为打开报警输出功能

0x56+0x00+0x42+0x02+0x02+0x00 设置为关闭报警输出功能

0x56+0x00+0x42+0x02+0x03+0x01 启动报警输出

0x56+0x00+0x42+0x02+0x03+0x00 停止报警输出

说明：

如要使用报警输出，请先为报警输出分配 GPIO 口。

2.3.2.19. MOTION_STATUS

命令功能： 获取指定的 Motion 属性的配置

命令格式： 0x56+序列号+0x43+0x00+ Motion 控制属性+控制项

Motion 控制属性：Motion 控制命令；

0：Motion 控制方法及使能控制

1：设置报警输出属性

2：报警输出使能控制

3：报警输出控制

控制项：表示对要设置的 Motion 控制属性的配置；

- 表示 Motion 控制方法及使能控制

  第一个字节表示 GPIO 控制还是串口控制。

  0：GPIO 控制

  1：串口控制

  第二个字节用于表示当前的状态

  Bit0 表示打开了 Motion 检测

  0：禁止 Motion 检测

  1：打开 Motion 检测

  Bit4 表示是否检测到了 Motion

  0：未检测到 Motion

  1：检测到了 Motion

- 报警输出属性设置

  第一个字节表示报警方式：

  Bit0：报警方式

  0：报警指定时间后，停止报警；

  1：一直报警；

  Bit1：报警电平

  0：平时输出低电平，报警时输出高电平；

  1：平时输出高电平，报警时输出低电平；

  第二三字节表示报警时间：高位在前，单位为 10ms

- 报警输出控制

  第一个字节表示打开还是关闭报警输出；

  0：禁止报警输出

  1：使能报警输出

- 报警输出控制

  第一个字节表示打开还是关闭报警输出；

  0：停止报警输出

  1：启动报警输出

回复格式：

如果运行正确，则回复：0x76+序列号+0x43+0x00+数据长度+ Motion 控制属性+控制项

如果运行失败，则回复：0x76+序列号+0x43+错误代码+0x00

举例：

0x56+0x00+0x43+0x01+0x00　　获取 Motion 当前的控制方式和当前 Motion 检测是否打开的状态

0x56+0x00+0x43+0x01+0x01　　获取当前的报警输出属性

0x56+0x00+0x43+0x01+0x02　　获取报警输出功能的使能状态

0x56+0x00+0x43+0x01+0x03　　获取当前是否启动了报警输出

说明：

2.3.2.20. TV_OUT_CTRL

命令功能： TV 输出控制

命令格式： 0x56+序列号+0x44 +0x01+控制项（1 个字节）

控制项：控制 TV 输出的打开与关闭

0：关闭 TV 输出

1：打开 TV 输出

回复格式：

如果运行正确，则回复：0x76+序列号+0x44+0x01+0x00

如果参数错误，则回复：0x76+序列号+0x44+0x03+0x00

举例：

0x56+0x00+0x44+0x01+0x01　　打开 TV 输出

0x56+0x00+0x44+0x01+0x00 关闭 TV 输出

说明：

2.3.2.21. OSD_ADD_CHAR

命令功能： 向 OSD 的字符通道（通道 1）添加 OSD 字符

命令格式： 0x56+序列号+0x45 +数据长度+字符个数（1 个字节）+起始地址（1 字节）+字符（n 字符）

字符个数：表示连续写入的字符的个数，最多 14 个；

起始地址：表示要显示的字符的起始地址；表示格式为：

Bit[4-0]：表示该字符在字符通道的某行中的列坐标；

Bit[6-5]：表示该字符所在的字符通道的行；

字符：要显示的字符，字符格式为 VC0703 的 OSD 字符格式；

回复格式：

如果运行正确，则回复：0x76+序列号+0x45+0x00+0x00

如果参数错误，则回复：0x76+序列号+0x45+0x03+0x00

举例：

0x56+0x00+0x45+0x07+0x06+0x24+0x1F+0x2C+0x30+0x2C+0x26+0x35+0x32

向字符通道的第 2 行第 4 列开始依次写入“Vimicro”7 个字符

说明：

在添加 OSD 字符前，需要先设置好 OSD 模块。

VC0703 共支持 80 个字符，下面的表格说明各个字符对应的索引值，例如字符 “0” 索引值为 0x00, “1” 索引值为 0x01, … “G” 索引值为 0x10, “H” 索引值为 0x11, …

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**	**-**
_	**:**	**.**	**/**	*****	**(**	**)**	**[**	**]**	**@**	**!**	**+**	**|**	****	**#**	■					

2.3.2.22. GET_FLASH_SIZE

命令功能： 获取 SPI Flash 的大小

命令格式： 0x56+序列号+0x60+0x00

回复格式： 0x76+序列号+0x60+0x00+0x04+Flash 大小（4 字节）

返回的 Flash 大小单位为 K 字节

举例：

0x56+0x00+0x60+0x00 获取 SPI Flash 的大小

说明：

系统只能识别部分 SPI Flash 的容量，不能识别其容量的 SPI Flash，通过该命令获取的长度为 0。

2.3.2.23. ERASE_FLASH_SECTOR

命令功能： 擦除 SPI Flash 指定的 Sector

命令格式： 0x56+序列号+0x61+0x04+起始地址（4 字节）

起始地址：指要擦除的 Sector 的起始地址

回复格式： 0x76+序列号+0x61+0x00+0x00

举例：

0x56+0x00+0x61+0x04+0x00+0x01+0x00+0x00　　擦除 SPI Flash 中以 0x10000 地址开始的 Sector

说明：

2.3.2.24. ERASE_FLASH_ALL

命令功能： 擦除整个 SPI Flash

命令格式： 0x56+序列号+0x62 +0x00

回复格式： 0x76+序列号+0x62+0x00+0x00

举例：

0x56+0x00+0x62 +0x00　　擦除整个 SPI Flash

说明：

2.3.2.25. READ_LOGO

命令功能： 从控制信息读取 Logo 并显示

命令格式： 0x56+序列号+0x70+0x06 +数据大小（2 字节）+起始地址（4 字节）

数据大小：指 Logo 数据长度；

起始地址：Logo 数据在控制信息中的起始位置；

回复格式： 0x76+序列号+0x70+0x00+0x00

系统在将 Logo 从控制信息读取到 OSD 后进行命令回复。

举例：

0x56+0x00+0x70+0x06 +0x02+0x00+0x00+0x00+080+0x00

从控制信息的 0x800 地址开始读取 512 字节的 Logo 数据到 OSD 中，并显示.

说明：

2.3.2.26. SET_BITMAP

命令功能： 对 Bitmap 进行操作；从控制信息中读取 Bitmap 以及控制 Bitmap 的显示

命令格式： 0x56+序列号+0x71+数据长度+操作类型（1 字节）+数据大小（2 字节）+起始地址（4 字节）

操作类型：表示对 Bitmap 的操作类型

1：表示停止从 SPI Memory（SPI EEPROM 或 SPI Flash）中传送 Bitmap，也即停止了 Bitmap 显示；

2：表示启动从 SPI Memory（SPI EEPROM 或 SPI Flash）中传送 Bitmap，也即打开了 Bitmap 显示；

数据大小：要传送的 Bitmap 的长度，当操作类型为 1 时，没有该控制项；

起始地址：要传送的 Bitmap 数据在 SPI Memory 中的起始位置，只有操作类型为 2 的操作需要该控制项；

从 SPI Memory 中读取 Bitmap 并显示的命令为：

0x56+序列号+0x71+0x07+0x02+数据大小（2 字节）+起始地址（4 字节）

停止 Bitmap 显示的命令为：

0x56+序列号+0x71+0x01+0x01

回复格式：

如果运行正确，则回复：0x76+序列号+0x71+0x00+0x00

如果参数错误，则回复：0x76+序列号+0x71+0x03+0x00

举例：

- 0x56+0x00+0x71+0x01+0x01

  停止 Bitmap 显示

- 0x56+0x00+0x71+0x07+0x02+0x25+0x80+0x00+0x00+0x40+0x00

  从 SPI Memory 的地址 0x4000 中读取长度为 0x2580 字节的 Bitmap，并显示

说明：

如果启动了 Bitmap 显示，并且是读取 SPI Memory 的方式，则 SPI 通道就被独占了，不能够再进行 SPI Memory 的读写访问，如果要对 SPI Memory 进行读写操作，则必须先停止 Bitmap 显示。

3. Revision History

Version No.	Remarks	Release Date
0.9	Preliminary release	2007-11-5