文档修订说明

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订日期** | **修订说明** | **修订人** |
| V1.0.0 | 2019/10/31 | 创建文档 | 王海 |
|  |  |  |  |

目录

[1. LCD的工作原理 2](#_Toc16688951)

[1.1. 什么是LCM 2](#_Toc16688952)

[1.2. 工作原理 2](#_Toc16688953)

[1.2.1. 背光源 2](#_Toc16688954)

[1.2.2. LCD的重要参数 2](#_Toc16688955)

[2. LCD的硬件部分 3](#_Toc16688956)

[2.1. 背光电路 3](#_Toc16688957)

[2.1.1. 管脚说明 3](#_Toc16688958)

[2.2. LCD控制IC 4](#_Toc16688959)

[3. LCD的软件部分 5](#_Toc16688960)

[3.1. Bootloader(LK)部分 5](#_Toc16688961)

[3.1.1. 和LCD相关的文件 5](#_Toc16688962)

[3.1.2. 新建一个LCD IC的参数配置文件 5](#_Toc16688963)

[3.1.3. LK里的点屏流程图 9](#_Toc16688964)

[3.2. Kernel部分 9](#_Toc16688965)

[4. 高通平台LCD MIPI数据格式 10](#_Toc16688966)

[4.1. Bootloader里 10](#_Toc16688967)

[4.1.1. LCD初始化序列 10](#_Toc16688968)

[4.1.2. 读取寄存器序列 11](#_Toc16688969)

[4.1.3. 设置返回包长度 11](#_Toc16688970)

[4.2. Kernel里 13](#_Toc16688971)

[5. 常用的MIPI协议命令类型 14](#_Toc16688972)

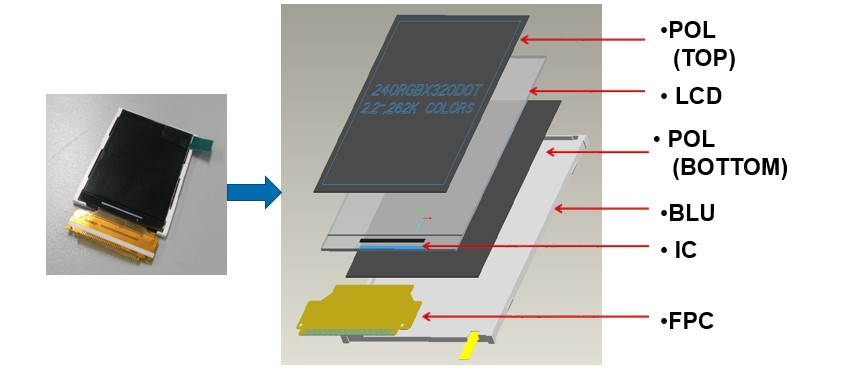
[6. 常见问题 15](#_Toc16688973)

[7. 调试一款新的LCD需要的资料 15](#_Toc16688974)

# LCD的工作原理

## 什么是LCM

LCM（LCD Module）即LCD显示模组，是指将液晶**显示器件**、连接件、**控制IC**与驱动等外围电路、PCB电路板、**背光源**，结构件等 装配在一起的组件。



一般软件上需要控制的有：背光源、控制IC；

## 工作原理

屏工作需要要**背光源**和**玻璃**(LCD显示)两个部分：

通过对液晶两端LCD电极的加电控制，使液晶在LCD内部做不断旋转，达到光线的通过或阻隔，从而实现显示白色画面（光线透过）和黑色画面（光线阻隔）。

### 背光源

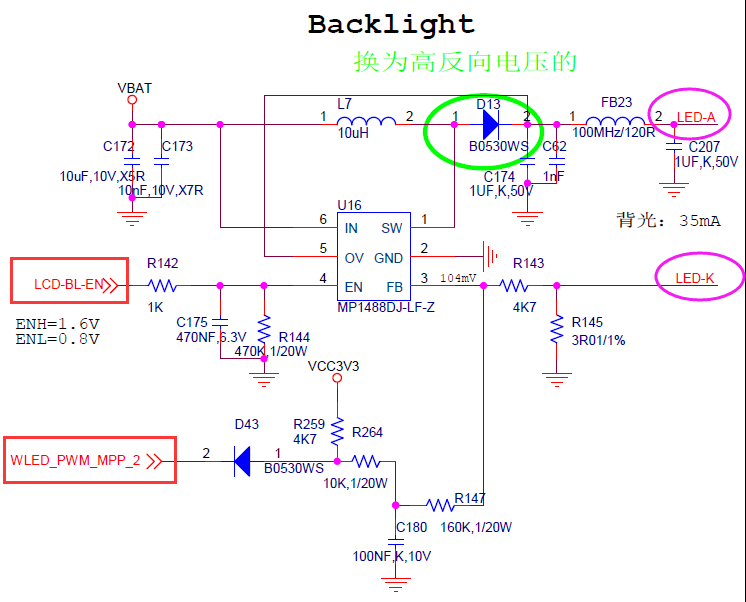
普通的LCD本身**并不发光**，要想让其显示画面，就必须使用白光背光源，常见的白光背光源一般由数个白色LED灯组成。

### LCD的重要参数

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **名称** | **功能** | **备注** |
| 1 | **分辨率** | LCD显示的像素，位置固定，一定尺寸的液晶屏，分辨率越高，显示图形越清晰；例如：1280\*720 |  |
| 2 | 尺寸 | LCD包含的显示区域，外观长宽厚度等，例如：5.0英寸 |  |
| 3 | 信号接口 | 常见的有RGB、SPI、LVDS、MIPI-CSI等 |  |
| 4 | 工作电压 | 例如：模拟电压2.8V，逻辑电压1.8V |  |
| 5 | 工作温度 | 例如：-20/70℃ |  |
| 6 | 视角 | 全视角，窄视角(确认几点钟方向) |  |
| 7 | 响应时间 | 如：typ 30ms, max 40ms; |  |
| 8 | 帧率 | 一般设置为56~60 |  |

# LCD的硬件部分

## 背光电路



### 管脚说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **管脚编号** | **管脚名称** | **功能** | **备注** |  |
| 1 | LCD-BL-EN | 控制背光的打开和关闭 |  |  |
| 2 | LCD-BL-PWM | 控制背光的亮度调节 |  |  |
| 3 | LED-BL-A | 接LED的阳极 |  |  |
| 4 | LED-BL-K | 接LED的阴极 |  |  |

背光开关规则：

关屏：先关背光，后清framebuffer

亮屏：先填framebuffer，后开背光

## LCD控制IC

一般LCM都会通过一颗LCD IC来控制图像在玻璃上的显示，和LCD相关的管脚如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **名称** | **功能** | **备注** |
| 1 | **IOVCC** | I/O Power supply ( to the digital circuit), 1.8/2.8V |  |
| 2 | **VCC** | Analog Power supply (to the analog circuit).2.8V |  |
| 3 | **RESET** | Reset input signal. Initialize the IC with a low input. Must be reset after power on . | 初始化IC |
| 4 | **LCD\_ID** | Output a voltage for ID identification .Connect a 100K ohm resistance to IOVCC. | 一般用于多个LCM的兼容 |
| 5 | **CLK** | 时钟信号，常见的例如：MIPI CLK |  |
| 6 | **DATA** | 数据信号/控制信号，常见的例如：MIPI DATA | 480P一般使用2 lane mipi，720p一般使用 4 lane mipi |

# LCD的软件部分

在Android平台上，点屏分为两部分：

1. 设备上电后亮屏，使用的是Bootloader里的屏参
2. 设备休眠后唤醒，使用的是Kernel里的参数

因此在Android平台上，调试LCD，首先要调试Bootloader部分，然后调试Kernel部分，具体调试过程如下，以下是在人脸项目上，调试一款LCM【IC：天钰的JD9365AA，玻璃：HSD】的详细过程。

## Bootloader(LK)部分

### 和LCD相关的文件

dev/gcdb/display/include/panel\_jd9365aa\_800p\_video.h

|  |  |  |
| --- | --- | --- |
| **编号** | **文件名** | **功能** |
| 1 | dev/gcdb/display/include/panel\_jd9365aa\_800p\_video.h | LCD IC的参数配置文件 |
| 2 | dev/gcdb/display/include/panel.h | 定义新屏的枚举 |
| 3 | target/msm8953/oem\_panel.c | 加载新屏的屏参 |
| 4 | platform/msm\_shared/mipi\_dsi.c | 实现新屏的兼容方式 |
| 5 | target/msm8953/target\_display.c |

### 新建一个LCD IC的参数配置文件

dev/gcdb/display/include/panel\_jd9365aa\_800p\_video.h

在此文件里，我们主要修改的部分如下：

#### 填充panel\_config数据结构

static struct **panel\_config** jd9365aa\_800p\_video\_panel\_data = {

**"qcom,mdss\_dsi\_hsd\_jd9365aa\_800p\_video", /\*** panel\_**node\_id \*/**

"dsi:0:", /\* panel\_controller \*/

"qcom,mdss-dsi-panel", /\* panel\_compatible \*/

10, /\* panel\_interface \*/

0, /\* **panel\_type** \*/

"DISPLAY\_1", /\* panel\_destination \*/

0, /\* panel\_orientation \*/

0, /\* panel\_clockrate \*/

**60, /\*** panel\_**framerate \*/**

0, /\* panel\_channelid \*/

0, /\* dsi\_virtualchannel\_id \*/

0, /\* panel\_broadcast\_mode \*/

0, /\* panel\_lp11\_init \*/

0, /\* panel\_init\_delay \*/

0, /\* dsi\_stream \*/

0, /\* interleave\_mode \*/

0, /\* panel\_bitclock\_freq \*/

0, /\* panel\_operating\_mode \*/

0, /\* panel\_with\_enable\_gpio \*/

0, /\* mode\_gpio\_state \*/

NULL //"qcom,mdss\_dsi\_lt8912\_1080p"/\* **slave\_panel\_node\_id** \*/

};

#### 填充panel\_resolution数据结构

static struct **panel\_resolution** jd9365aa\_800p\_video\_panel\_res = {

800, /\* panel\_width \*/

1280, /\* panel\_height \*/

32, /\* hfront\_porch \*/

20, /\* hback\_porch \*/

36, /\* hpulse\_width \*/

0, /\* hsync\_skew \*/

8, /\* vfront\_porch \*/

4, /\* vback\_porch \*/

4, /\* vpulse\_width \*/

0, /\* hleft\_border \*/

0, /\* hright\_border \*/

0, /\* vtop\_border \*/

0, /\* vbottom\_border \*/

0, /\* hactive\_res \*/

0, /\* uint16\_t vactive\_res \*/

0, /\* invert\_data\_polarity \*/

0, /\* invert\_vsync\_polarity \*/

0 /\* invert\_hsync\_polarity \*/

};

#### 填充初始化序列mipi\_dsi\_cmd数据结构

static char jd9365aa\_800p\_video\_on\_cmd0[] = {

0x02,0x00,0x39,0xC0,

0xE0,0x00,0xFF,0XFF,

};

static char jd9365aa\_800p\_video\_on\_cmd1[] = {

0x02,0x00,0x39,0xC0,

0xE1,0x93,0xFF,0XFF,

};

static char jd9365aa\_800p\_video\_on\_cmd2[] = {

0x02,0x00,0x39,0xC0,

0xE2,0x65,0xFF,0XFF,

};

…………..

static char jd9365aa\_800p\_video\_on\_cmd210[] = {

0x11, 0x00, 0x05, 0x80

};

static char jd9365aa\_800p\_video\_on\_cmd211[] = {

0x29, 0x00, 0x05, 0x80

};

static struct **mipi\_dsi\_cmd** jd9365aa\_800p\_video\_on\_command[] = {

MIPI\_DSI\_CMD\_WAIT{jd9365aa\_800p\_video\_on\_cmd0,0x00},

MIPI\_DSI\_CMD\_WAIT{jd9365aa\_800p\_video\_on\_cmd1,0x00},

MIPI\_DSI\_CMD\_WAIT{jd9365aa\_800p\_video\_on\_cmd2,0x00},

…….

**/\* 开显示 \*/**

MIPI\_DSI\_CMD\_WAIT{jd9365aa\_800p\_video\_on\_cmd210,0x78},

MIPI\_DSI\_CMD\_WAIT{jd9365aa\_800p\_video\_on\_cmd211,0x05},

};

#define JD9365AA\_800P\_VIDEO\_ON\_COMMAND\_CNT \

(ARRAY\_SIZE(jd9365aa\_800p\_video\_on\_command))

这一部分和**LCD IC、玻璃** 直接相关，需要模组厂提供，最好让模组厂提供高通平台格式的，否则要自己转换，转换方法参考**<高通平台LCD MIPI数据格式**> 章节

#### Mipi接口的配置lane\_configuration

static struct lane\_configuration jd9365aa\_800p\_video\_lane\_config = {

**4**, /\* dsi\_lanes \*/

0,

1, /\* lane0\_state \*/

1, /\* lane1\_state \*/

1, /\* lane2\_state \*/

1, /\* lane3\_state \*/

0

};

一般480p，使用2 lane date

#### TIMMING的设置

这个用高通提供的excel来计算，不同的平台可能使用方法不一样，具体的需要和方案商确认

#### Reset时序设置，填充panel\_reset\_sequence

static struct **panel\_reset\_sequence** jd9366\_800p\_video\_reset\_seq = {

{1, 0, 1, }, {20, 2, 20, }, 2

};

#### 定义LCD兼容ID

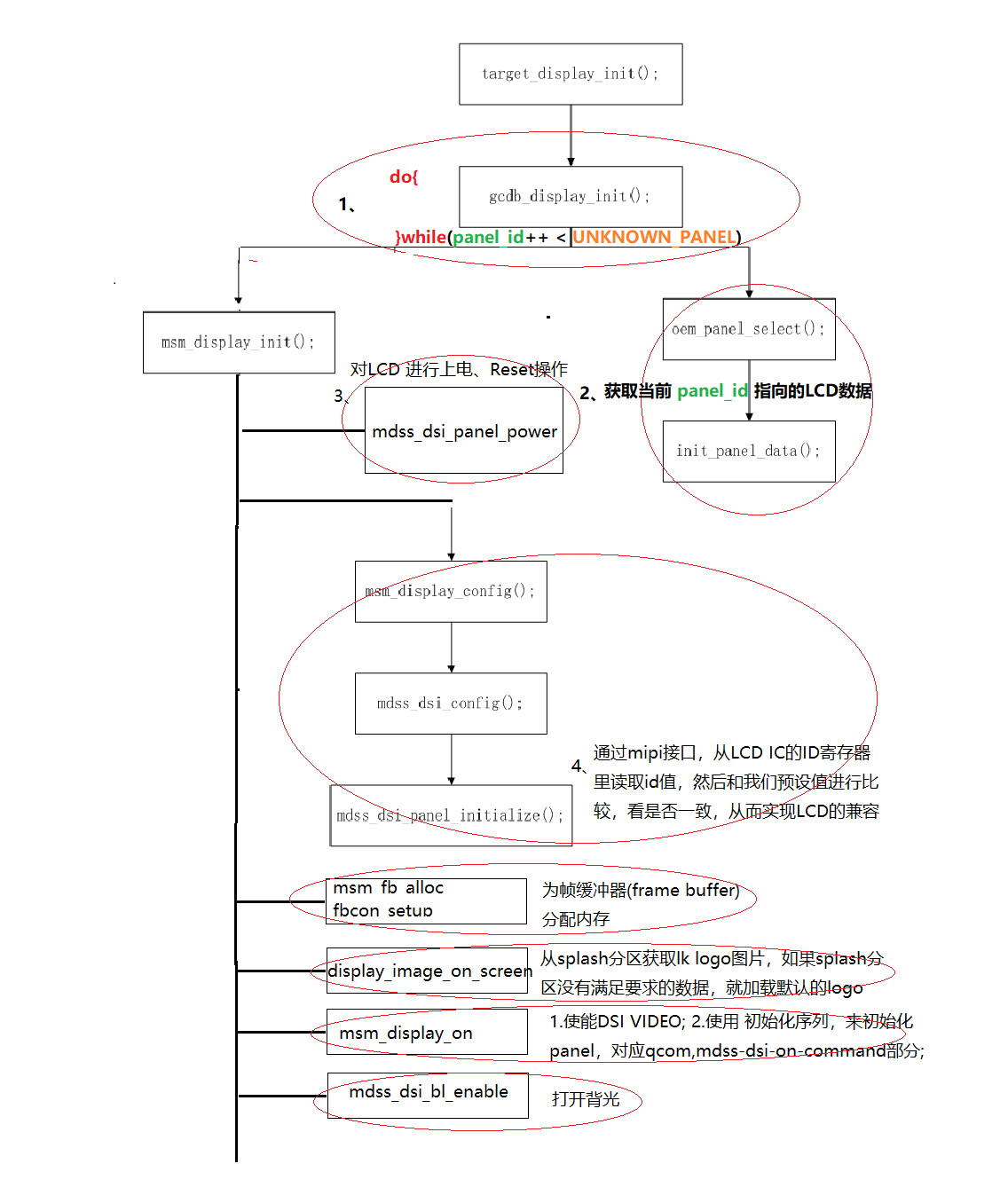
在整机成本里，LCM占了一个大头，一般为了实现降本，我们常常在一个平台上需要支持多个LCM，因此需要实现LCD的兼容

LCD的常用兼容方法如下：

1、通过ADC识别：在硬件上，为每一个LCM定义一个固定的电压，上电时，软件通过ADC来识别当前的LCM

2、通过软件实现，一般每一个IC都有可定义的OPT寄存器，我们要提前为每一个模组厂分配一个ID，这样当软件上电之后，我们可以通过MIPI接口是判断当前的LCM是不是我们所支持的

### LK里的点屏流程图



## Kernel部分

Kernel里需要填充的部分和LK里的基本上是一致的，因为kernel里是填充到了dts里，所以格式和LK有区别，具体的参考**<高通平台LCD MIPI数据格式**>

注意：

kernel里需要填充**关屏**序列

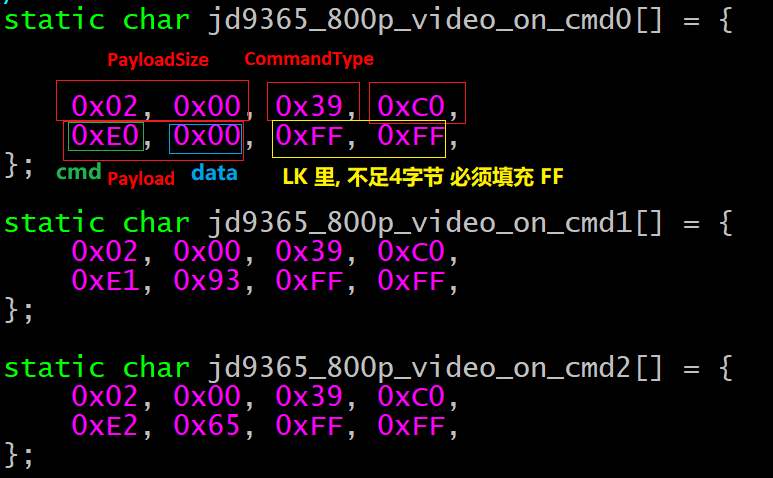
# 高通平台LCD MIPI数据格式

一般每一个SOC厂商都定义了自己的MIPI接口LCD的初始化序列，例如：QCOM、MTK都有自己的格式，并且每一个LCD IC的初始化序列条数和长度都可能不同，例如：奕力的IC，一般都有200多条。

## Bootloader里

### LCD初始化序列

数据结构如下：



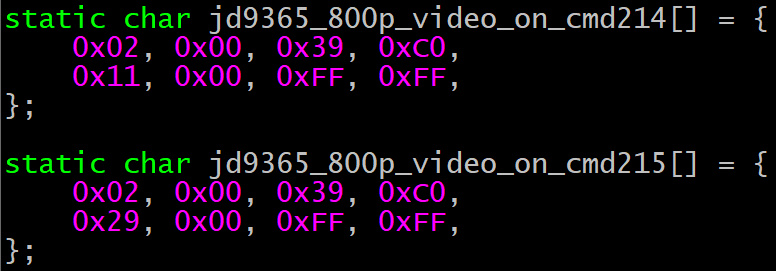
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **第几字节** | **功能** | **描述** | **字节长度** | **备注** |
| 1 | **PayloadSize** | 有效荷载大小 | **2** | 填写**实际**的payload大小 |
| 2 | **CommandType** | MIPI协议的命令类型 | 1 | 常用的命令类型：0x39 |
| 3 | 固定为0xC0 | 当CommandType为0x39时，这里固定为0xC0 | 1 |  |
| 4 | **Payload** | 实际的命令 | **不固定** | 一般由 **地址+值** 构成，就是所谓的初始化序列，必须4字节对齐，使用0xFF填充 |

注意：

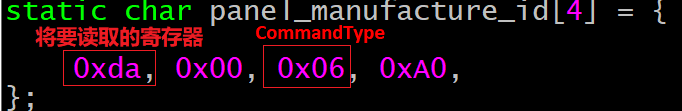
1、初始化序列，如果总长度不是4的整数倍，用0xFF来填充

2、参数“Payload”需要由屏厂提供，就是我们通常说的LCD 初始化参数

3、<OnCommand>最后一般都是如下两条指令，个别屏可能不需要

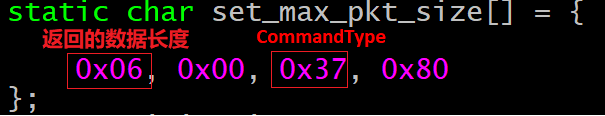


### 读取寄存器序列

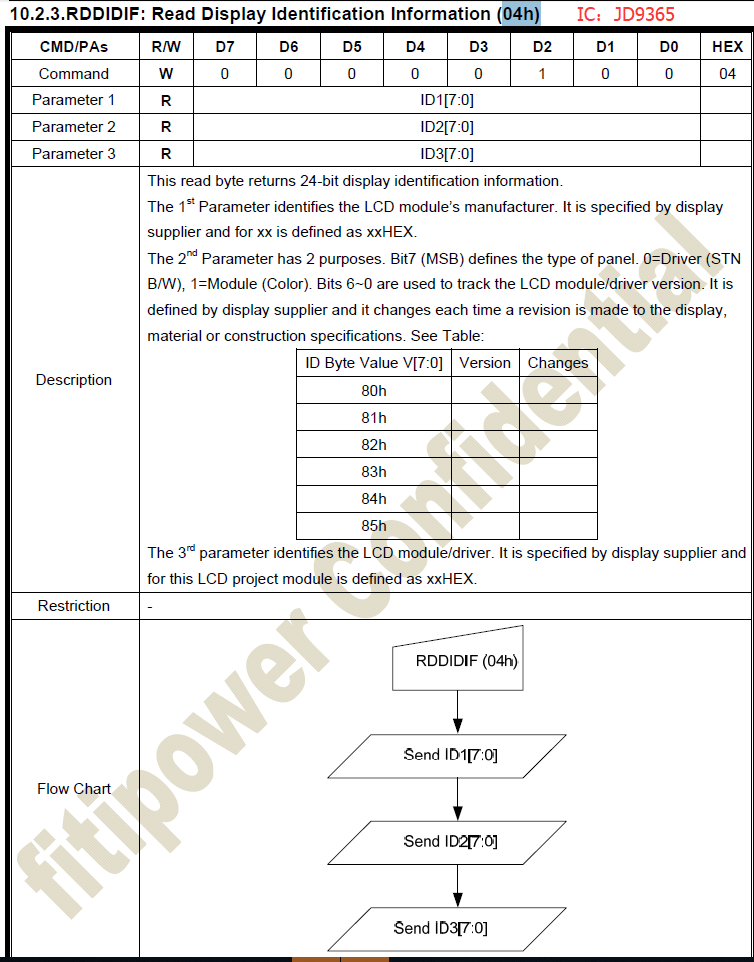


一般每一个LCD IC，都有几个OPT寄存器，例如DAH，DBH，DCH等，在LCM出厂之前就让模组厂把我们指定的值烧录到对应的OPT寄存器里，这样就可以识别不同的模组厂。

### 设置返回包长度

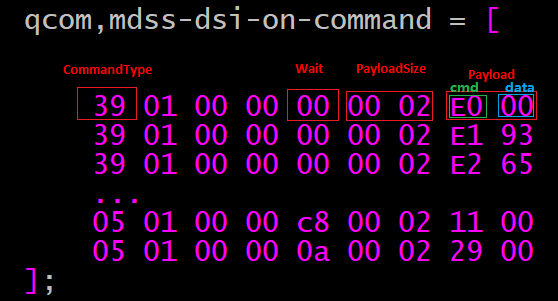


有时候我们读取LCD-ID的时候，按照datasheet，可能会返回多个字节，例如：当我们读取天钰JD9365 IC的0x04H的时候，这时候IC会返回3个数据，因此在读取数据之前，我们要通过MIPI接口设置返回的包大小



## Kernel里

数据结构如下：



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **第几字节** | **功能** | **描述** | **字节长度** | **备注** |
| 1 | **CommandType** | MIPI协议的命令类型 | 1 | 常用的命令类型：0x39/0x05/ |
| 2 | Last | 指定此命令数据包是否是单独的 | 1 | 一般设置为1 |
| 3 | VC | 用于发送此命令的虚拟通道 | 1 | 一般设置为0 |
| 4 | Ack | 是否需要panel应答 | 1 | 一般设置为0 |
| 5 | **Wait** | 发送下一条命令之前，等待的时间 | 1 | 单位微妙，无需等等设置为0 |
| 6 | **PayloadSize** | 有效荷载大小 | **2** | 填写**实际**的payload大小 |
| 7 | **Payload** | 实际的命令 | **不固定** | 一般由 **地址+值** 构成，就是所谓的初始化序列，在Kernel里不用对齐 |

注意：

1、通常我们只要修改“PayloadSize” and “Payload”.

2、有些lcd 会要求在发送command 前延时xx ms。可以修改”wait”参数，单位ms.

3、参数“Payload”需要由屏厂提供，就是我们通常说的LCD 初始化参数

4、<OnCommand>最后一般都是如下两条指令，个别屏可能不需要

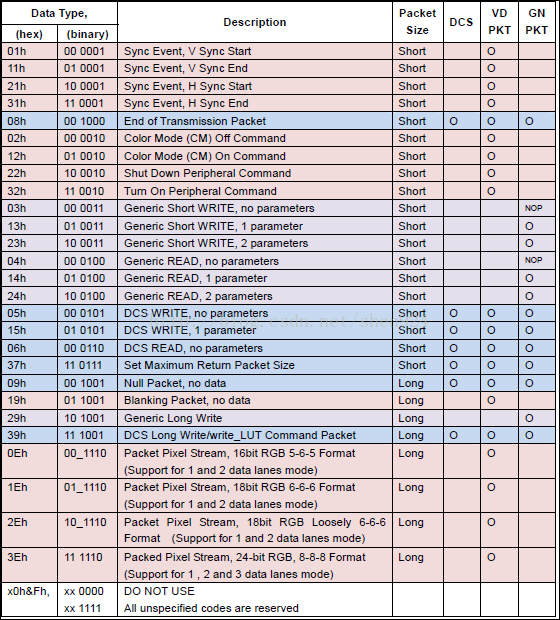
0x05, 0x01, 0x00, 0x00, 0x78, 0x00, 0x02, 0x11, 0x00,

0x05, 0x01, 0x00, 0x00, 0x05, 0x00, 0x02, 0x29, 0x00,

0x11: sleep out command of LCD.

0x29: display on command of LCD.

# 常用的MIPI协议命令类型



# 常见问题

1. 液晶激化：有时候调试LCD的时候，会把一家模组厂的初始化序列下载到另一家模组厂的IC里，发现屏幕花屏，但是后来把正确的初始化序列下载到IC里，依然有花屏，例如：奕力\ILI9881D这个IC，就会出现类似的问题，一般这个是可逆的，但是要断电等待大半天。
2. 背光时序的控制
3. LCD的兼容，OPT寄存器的读取
4. LK和Kernel的panle\_node\_id 不一致，导致花屏；

# 调试一款新的LCD需要的资料

|  |  |  |
| --- | --- | --- |
| **编号** | **资料** | **备注** |
| 1 | IC的Datasheet |  |
| 2 | 初始化序列 | 模组厂demo板子上的 和 当前平台的 |
| 3 | 前后肩信息，如下： | 需要模组厂提供，如下：  frame rate 60  lane config 4  pixel format BPP 3  Active Width 720  Active Height 1280  Hsync Pulse Width 8  Hori. Back Porch 48  Hori. Front Porch 52  Vsync Pulse Width 6  Vert. Back Porch 15  Vert. Front Porch 16 |
| 4 | BIST自测试指令 |  |
| 5 | 实现LCM的兼容，ID寄存器的地址，和对应的值 |  |
| 6 |  |  |
|  |  |  |