MIPI-2-RGB

|  |
| --- |
| **目录**   [[隐藏](javascript:toggleToc())]   * [1 MIPI转RGB](http://192.168.0.138/wiki/index.php/MIPI-2-RGB#MIPI.E8.BD.ACRGB)   + [1.1 简介](http://192.168.0.138/wiki/index.php/MIPI-2-RGB#.E7.AE.80.E4.BB.8B)     - [1.1.1 RGB 简介](http://192.168.0.138/wiki/index.php/MIPI-2-RGB#RGB_.E7.AE.80.E4.BB.8B)     - [1.1.2 ICN6211 简介](http://192.168.0.138/wiki/index.php/MIPI-2-RGB#ICN6211_.E7.AE.80.E4.BB.8B)   + [1.2 添加 MIPI 转 RGB 支持](http://192.168.0.138/wiki/index.php/MIPI-2-RGB#.E6.B7.BB.E5.8A.A0_MIPI_.E8.BD.AC_RGB_.E6.94.AF.E6.8C.81)     - [1.2.1 ICN6211 参数图](http://192.168.0.138/wiki/index.php/MIPI-2-RGB#ICN6211_.E5.8F.82.E6.95.B0.E5.9B.BE)     - [1.2.2 LK](http://192.168.0.138/wiki/index.php/MIPI-2-RGB#LK)     - [1.2.3 kernel](http://192.168.0.138/wiki/index.php/MIPI-2-RGB#kernel)   + [1.3 修改点](http://192.168.0.138/wiki/index.php/MIPI-2-RGB#.E4.BF.AE.E6.94.B9.E7.82.B9)   + [1.4 ICN6221文档](http://192.168.0.138/wiki/index.php/MIPI-2-RGB#ICN6221.E6.96.87.E6.A1.A3) |

[[编辑](http://192.168.0.138/wiki/index.php?title=MIPI-2-RGB&action=edit&section=1)]MIPI转RGB

[[编辑](http://192.168.0.138/wiki/index.php?title=MIPI-2-RGB&action=edit&section=2)]简介

屏的接口种类非常多，常见的包括 RGB、HDMI、VGA、LVDS、EDP、MIPI等接口。 其中，在 Android 移动设备上，大多采用的是MIPI接口。 某些时候，由于某种需求，需要将 Android 设备上的 MIPI 数据显示到其他接口的屏上，此时，则需要利用相关转换芯片将 MIPI 接口的数据转换成其他接口的数据。 在 msm8916 上有过这样的需求：将 MIPI 数据转换成 RGB 进行输出，当时采用的转换芯片是 ICN6211。以下记录之。

[[编辑](http://192.168.0.138/wiki/index.php?title=MIPI-2-RGB&action=edit&section=3)]**RGB 简介**

R(红)、G(绿)、B(蓝)是自然界三原色，三原色按照不同比例进行相加，混合成不同的颜色。 RGB其取值范围分别为0~255，其值越大，则越亮，当RGB都取值为255时，则为白色。相反全为0时，则为黑色。 而RGB接口就是分三原色进行输入的图形视频接口，也叫作色差分量接口。

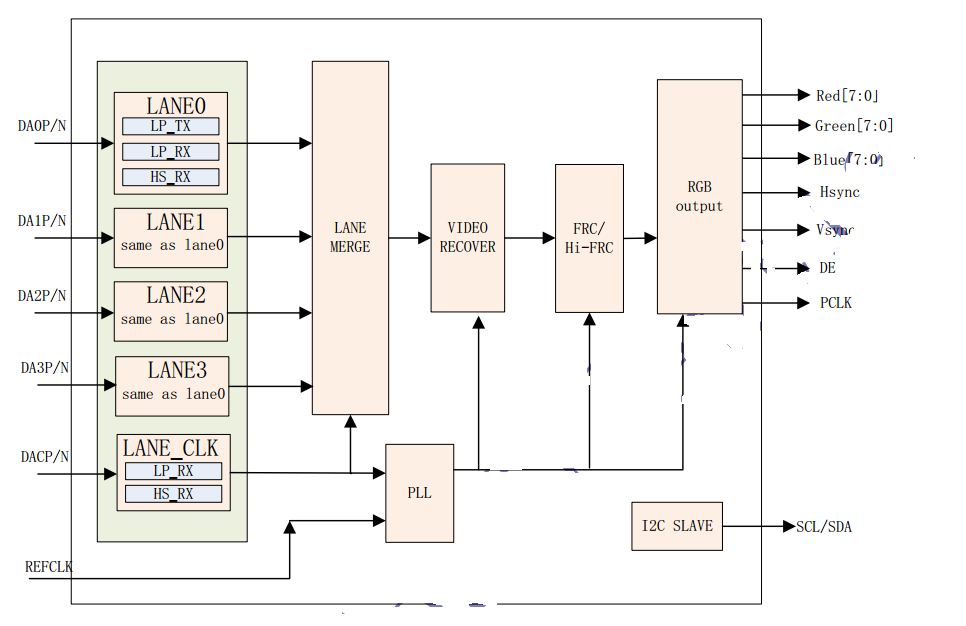
RGB 接口中包括三大类信号：数据信号、时钟信号和控制信号。

* 数据信号，通过数据线传输，一般分为6 bit 或者 8 bit传输(R0~R5/G0~G5/B0~B5 或 R0~R7/G0~G7/B0~B7)，也即是数据信号线分为18根或者24根
* 时钟信号，特指像素时钟信号，传输数据和对数据信号进行读取的基准。
* 控制信号，包括数据使能信号DE(或有效显示数据选通信号)、行同步信号HS、帧同步信号VS

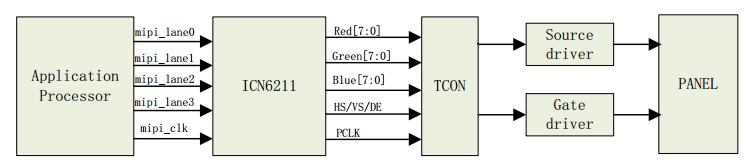
[[编辑](http://192.168.0.138/wiki/index.php?title=MIPI-2-RGB&action=edit&section=4)]**ICN6211 简介**

ICN6211 是一款 MIPI 转 RGB 的芯片，下面是它的简要特性:

* 其最高支持 MIPI DSI 的4 Lane 输入；当4lane输入时，其支持的最大带宽为1GBps
* 其能对 MIPI DSI的 RGB565-16bpp、RGB666-18bpp 以及 RGB888-24bpp 的数据包进行解码转换
* 其能输出 18 bpp 或者 24 bpp 像素的 RGB 图形数据，像素时钟的范围在 5MHz ~ 154MHz
* 其支持的最大分辨率为 FHD(1920x1080p) 以及 WUXGA(1920x1200p)
* 支持 I2C 接口

[](http://192.168.0.138/wiki/index.php/%E6%96%87%E4%BB%B6:Icn6211_functional.png)

icn6211 Functional Block Diagram

[](http://192.168.0.138/wiki/index.php/%E6%96%87%E4%BB%B6:Icn6211_app.png)

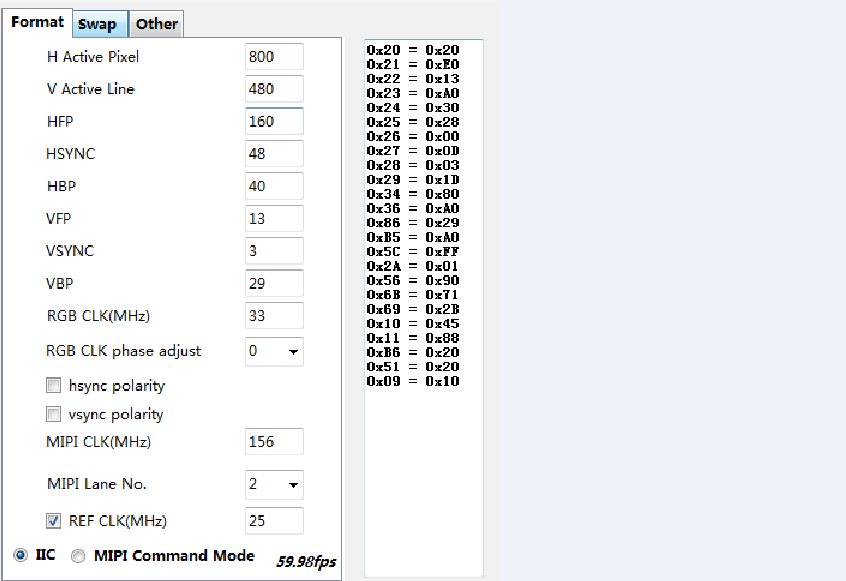
System Application Diagram

屏的显示分为两部分，一部分是开机显示厂商 logo，一部分是系统启动完毕后正常的显示。下面就这两部分的开发细节进行介绍。

[[编辑](http://192.168.0.138/wiki/index.php?title=MIPI-2-RGB&action=edit&section=5)]添加 MIPI 转 RGB 支持

针对 MIPI 接口的屏，屏的配置参数通过 DATA0-、DATA0+ 这对数据线下发到屏端。 如果初始化正确，则切换到 HS 模式下，直接把数据从DATA0~DATA3刷出去即可。 那么针对 ICN6211-MIPI-RGB 的屏，又是怎么操作的呢？其实也非常简单，在给定 ICN6211 正确的电压后，将屏的配置参数通过 ICN6211 的I2C下发到屏端即可。 后续数据的传输则不再需要操心。

[[编辑](http://192.168.0.138/wiki/index.php?title=MIPI-2-RGB&action=edit&section=6)]**ICN6211 参数图**

[](http://192.168.0.138/wiki/index.php/%E6%96%87%E4%BB%B6:Icn6211_para.png)

[[编辑](http://192.168.0.138/wiki/index.php?title=MIPI-2-RGB&action=edit&section=7)]**LK**

在 **mdss\_dsi\_panel\_initialize()** 函数中，会下发屏的配置参数。 它原本是调用 MIPI 接口函数 **mipi\_dsi\_cmds\_tx()** 去下发的参数，在这里改成或者添加 I2C 的传输接口即可。

int mdss\_dsi\_panel\_initialize(struct mipi\_dsi\_panel\_config \*pinfo, uint32\_t

broadcast)

{

if (pinfo->i2c\_cmds\_table) {

mipi\_i2c\_cmds\_tx(pinfo->i2c\_cmds\_table);

}

}

static struct qup\_i2c\_dev \*i2c\_dev;

static int qrd\_lcd\_i2c\_write(int addr, int reg, int val)

{

int ret = 0;

uint8\_t data\_buf[] = { reg, val };

/\* Create a i2c\_msg buffer, that is used to put the controller into write

mode and then to write some data. \*/

struct i2c\_msg msg\_buf[] = { {addr,

I2C\_M\_WR, 2, data\_buf}

};

ret = qup\_i2c\_xfer(i2c\_dev, msg\_buf, 1);

if(ret < 0) {

dprintf(CRITICAL, "qup\_i2c\_xfer error %d\n", ret);

return ret;

}

return 0;

}

/\*write i2c command\*/

int mipi\_i2c\_cmds\_tx(struct i2c\_on\_command\_table \*i2c\_cmds\_table)

{

int i = 0;

i2c\_dev = qup\_blsp\_i2c\_init(BLSP\_ID\_1, QUP\_ID\_1, 100000, 19200000);

if(!i2c\_dev) {

dprintf(CRITICAL, "qup\_blsp\_i2c\_init failed \n");

ASSERT(0);

}

for(i = 0; i < i2c\_cmds\_table->cmds\_cnt; i++)

{

qrd\_lcd\_i2c\_write(i2c\_cmds\_table->slave\_addr, i2c\_cmds\_table->cmds\_list[i].reg, i2c\_cmds\_table->cmds\_list[i].val);

}

}

当然， **pinfo->i2c\_cmds\_table** 这个table,按照 qcom 的屏处理流程，它是在 **msm8916/oem\_panel.c** 里面进行处理的，下面直接贴出处理代码片段：

case ICN6211\_QHD\_VIDEO\_PANEL:

panelstruct->paneldata = &icn6211\_qhd\_video\_panel\_data;

panelstruct->panelres = &icn6211\_qhd\_video\_panel\_res;

panelstruct->color = &icn6211\_qhd\_video\_color;

panelstruct->videopanel = &icn6211\_qhd\_video\_video\_panel;

panelstruct->commandpanel = &icn6211\_qhd\_video\_command\_panel;

panelstruct->state = &icn6211\_qhd\_video\_state;

panelstruct->laneconfig = &icn6211\_qhd\_video\_lane\_config;

panelstruct->paneltiminginfo

= &icn6211\_qhd\_video\_timing\_info;

panelstruct->panelresetseq

= &hx8389b\_qhd\_video\_reset\_seq;

pinfo->mipi.i2c\_cmds\_table = &icn6211\_i2c\_on\_command\_table;

pinfo->mipi.i2c\_cmds\_table->cmds\_cnt = ICN6211\_QHD\_VIDEO\_ON\_COMMAND;

pinfo->mipi.i2c\_cmds\_table->slave\_addr = ICN6211\_I2C\_SLAVE\_ADDR;

pinfo->mipi.i2c\_cmds\_table->blsp\_id = I2C\_BLSP\_ID;

pinfo->mipi.i2c\_cmds\_table->qup\_id = I2C\_QUP\_ID;

panelstruct->backlightinfo = &icn6211\_qhd\_video\_backlight;

memcpy(phy\_db->timing,

icn6211\_qhd\_video\_timings, TIMING\_SIZE);

break;

[[编辑](http://192.168.0.138/wiki/index.php?title=MIPI-2-RGB&action=edit&section=8)]**kernel**

在 msm8916 的 kernel 部分，对一个屏驱动的处理分为三步：

* 解析该屏的设备树文件(DTSI)
* 注册屏初始化命令传输接口
* 下发初始化命令，使能 Panel

下面就一步一步的来看怎么添加的:

1. 解析 I2C 命令

static int mdss\_panel\_parse\_dt(struct device\_node \*np,

struct mdss\_dsi\_ctrl\_pdata \*ctrl\_pdata)

{

//...

rc = of\_property\_read\_string(np,

"qcom,mdss-command-access", &data);

if (!rc && !strcmp(data, "i2c")) { /\*the property is self defined, it may not exist in some dts configure\*/

ctrl\_pdata->cmd\_access = CMD\_ACCESS\_I2C;

ctrl\_pdata->i2c\_handle = mdss\_get\_icn6211\_i2c\_client();

if (!ctrl\_pdata->i2c\_handle)

{

pr\_err("ctrl\_pdata->i2c\_handle is null\n");

//goto error; /\*modified by fangchengbing return error will lead to crash\*/

}

mdss\_i2c\_parse\_dcs\_cmds(np, &ctrl\_pdata->i2c\_on\_cmds, "qcom,mdss-i2c-on-command");

} else { /\*normal flow\*/

//....

}

}

/\*\*

\* @brief parse i2c command in dtsi file, format [%d %d] ==> [reg, val]

\* store them in struct i2c\_cmd\_list

\*

\* @param np

\* @param pcmds where i2c will store

\* @param cmd\_key flag contain i2c command

\*

\* @return

\*/

static int mdss\_i2c\_parse\_dcs\_cmds(struct device\_node \*np,

struct i2c\_cmd\_list \*pcmds, char \*cmd\_key)

{

const char \*data;

int blen = 0;

char \*buf;

int i = 0, j = 0;

data = of\_get\_property(np, cmd\_key, &blen);

if (!data) {

pr\_err("%s: failed, key=%s\n", \_\_func\_\_, cmd\_key);

return -ENOMEM;

}

if (blen < 0 && blen / 2 == 1)

pr\_err("%s format err\n", cmd\_key);

pr\_debug("mdss\_i2c\_parse\_dcs\_cmds blen = %d\n", blen);

buf = kzalloc(sizeof(char) \* blen, GFP\_KERNEL);

if (!buf)

return -ENOMEM;

memcpy(buf, data, blen);

pcmds->cmds = kzalloc(sizeof(struct i2c\_ctrl\_hdr) \* blen / 2, GFP\_KERNEL);

pcmds->cmds\_cnt = blen / 2;

for (i = 0; i < blen; i+=2)

{

j = i / 2;

pcmds->cmds[j].reg = buf[i];

pcmds->cmds[j].val = buf[i+1];

}

return 0;

}

2. 控制器初始化时，注册屏初始化命令表:

void mdss\_dsi\_ctrl\_init(struct mdss\_dsi\_ctrl\_pdata \*ctrl)

{

//...

if (ctrl->cmd\_access == CMD\_ACCESS\_DSI)

ctrl->cmdlist\_commit = mdss\_dsi\_cmdlist\_commit;

else if (ctrl->cmd\_access == CMD\_ACCESS\_I2C)

ctrl->cmdlist\_commit = mdss\_i2c\_cmdlist\_commit;

//...

}

/\*\*

\* @brief write i2c command to mipi@rgb chip

\*

\* @param ctrl include i2c\_client

\* @param from\_mdp not important

\*

\* @return

\*/

int mdss\_i2c\_cmdlist\_commit(struct mdss\_dsi\_ctrl\_pdata \*ctrl, int from\_mdp)

{

int rc = 0;

int i = 0;

struct i2c\_cmd\_list on\_cmds;

on\_cmds = ctrl->i2c\_on\_cmds;

mutex\_lock(&ctrl->cmd\_mutex);

if (ctrl->i2c\_handle) {

for (i = 0; i < on\_cmds.cmds\_cnt; i++)

{

rc = i2c\_smbus\_write\_byte\_data(ctrl->i2c\_handle, on\_cmds.cmds[i].reg, on\_cmds.cmds[i].val);

if (rc < 0) {

mutex\_unlock(&ctrl->cmd\_mutex);

return rc;

}

}

}

mutex\_unlock(&ctrl->cmd\_mutex);

return rc;

}

3. 下发屏参数命令

static int mdss\_dsi\_panel\_on(struct mdss\_panel\_data \*pdata)

{

//...

if (ctrl->cmd\_access == CMD\_ACCESS\_DSI) {

if (ctrl->on\_cmds.cmd\_cnt)

mdss\_dsi\_panel\_cmds\_send(ctrl, &ctrl->on\_cmds);

} else if (ctrl->cmd\_access == CMD\_ACCESS\_I2C) {

if (ctrl->i2c\_on\_cmds.cmds\_cnt)

ctrl->cmdlist\_commit(ctrl, 0);

}

//...

}

注意!

最重要的有一点不要忘记，平台需要利用 I2C 总线去下发屏初始化参数，根据 I2C 协议，需要给定 I2C 设备在总线上的地址：

在 LK 里直接配置上 0x2C 即可，在 kernel 里面，dtsi 中配置如下:

i2c@78b6000 {

icn6211\_mipi\_rgb@2C {

compatible = "qcom,icn6211\_mipi\_rgb";

reg = <0x2C>;

};

};

[[编辑](http://192.168.0.138/wiki/index.php?title=MIPI-2-RGB&action=edit&section=9)]修改点

其实 MIPI 转 RGB 的修改点不多，也不难。就一个地方：初始化命令通过 MIPI 的 DATA0 下发，转移到了 I2C 下发。

[msm8916-751 聚力 MIPI转RGB LK修改点](http://192.168.0.240/#/c/626/)

[msm8916-751 聚力 MIPI转RGB kernel修改点](http://192.168.0.240/#/c/509/)

[[编辑](http://192.168.0.138/wiki/index.php?title=MIPI-2-RGB&action=edit&section=10)]ICN6221文档

[文件:ICN6211 MIPI RGB specification V05.pdf](http://192.168.0.138/wiki/index.php/%E6%96%87%E4%BB%B6:ICN6211_MIPI_RGB_specification_V05.pdf)

by Mickey.SHi

2016/8/15