

屏幕对接

使用指南

文档版本 V1.5

发布日期 2025-12-11

前言

概述

本文档介绍 XM7606V13/XM7206V11A 芯片解决方案上开发调试 RGB LCD、MIPI、LVDS、MCU LCD、BT656、BT1120 屏，以帮助客户有序快速开发相关业务。

各芯片支持的接口类型，可参考《MPP 媒体处理软件开发参考》文档“视频输出”章关于接口时序类型的说明。

注意：关于屏幕的称呼差异较大，本文中 RGB LCD 屏指的是以 RGB 为接口的 LCD 屏幕，一般指 RGB 屏。MIPI LCD 指的是以 MIPI DSI 为接口的 LCD 屏，一般指 MIPI 屏。LVDS LCD 指的是以 LVDS 为接口的 LCD 屏，一般指 LVDS 屏。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示如不可避免则将会导致死亡或严重伤害的具有高等级风险的危害。
 警告	表示如不可避免则可能导致死亡或严重伤害的具有中等级风险的危害。
 注意	表示如不可避免则可能导致轻微或中度伤害的具有低等级风险的危害。
 须知	用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
 说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2024-12-21	V1.0	DI 版本发布。
2025-06-05	V1.1	添加 MCU LCD 屏对接说明。
2025-08-20	V1.2	修改文档目录架构。
2025-11-04	V1.3	1.3.2 更新 dts 节点匹配说明。
2025-12-05	V1.4	1.6.2 添加 MCU 点屏实例说明。

修订日期	版本	修订说明
2025-12-11	V1.5	1.5 添加三种异常情况处理。 1.3.2 更新关于 mcu 屏参 dtsi 命令结构的描述。

目 录

1 屏幕对接指南	1
1.1 屏幕类型与接口	1
1.1.1 RGB LCD 屏幕接口	1
1.1.1.1 RGB LCD 屏幕连线	1
1.1.1.2 RGB LCD 屏参数数据接口使能确认	1
1.1.1.3 RGB LCD 屏幕参数控制命令传输接口	3
1.1.1.4 RGB LCD 数据传输接口	3
1.1.2 MIPI 屏幕接口	4
1.1.2.1 MIPI 屏幕连线	4
1.1.2.2 MIPI 屏幕数据传输接口	5
1.1.3 LVDS 屏幕接口	5
1.1.3.1 LVDS 屏幕连线	5
1.1.3.2 LVDS 屏幕控制接口	6
1.1.3.3 LVDS 屏幕数据传输接口	6
1.1.4 BT656/1120 屏幕接口	6
1.1.4.1 BT1120 屏幕连线	6
1.1.5 MCU LCD 屏幕接口	7
1.1.5.1 屏幕连线	7
1.2 屏幕框架说明	8
1.2.1 RGB LCD 屏幕框架	8
1.2.2 MIPI 屏幕框架	9

1.2.3 LVDS 屏幕框架	11
1.2.4 BT656/1120 屏幕框架	12
1.2.5 MCU 屏幕框架	13
1.3 配置流程	14
1.3.1 硬件准备与基础配置	14
1.3.1.1 硬件连线确认	14
1.3.1.2 管脚复用和驱动能力配置	14
1.3.2 屏幕参数配置	14
1.3.2.1 复位配置	14
1.3.2.2 屏幕控制参数配置	14
1.3.2.3 VO 参数与时序配置	18
1.3.2.4 配置背光	24
1.4 验证与调试	25
1.4.1 管脚复用与驱动能力调试	25
1.4.2 复位操作验证	25
1.4.3 背光控制调试	26
1.4.4 控制屏幕参数通路验证	26
1.4.5 VO 输出时钟	27
1.4.6 VO Colorbar 调试	27
1.5 异常处理 FAQ	28
1.5.1 屏幕全黑/无反应	28
1.5.2 显示位置错误	29
1.5.3 显示裂屏	29
1.5.4 显示大小错误	30
1.5.5 显示颜色异常	30
1.5.5.1 画面完全混乱显示效果无法辨认出目标的轮廓	30

1.5.5.2 出现颜色发生变化.....	30
1.5.5.3 显示线条有色差	31
1.5.5.4 屏幕显示效果不佳.....	32
1.5.6 屏幕画面转换时出现残影	32
1.5.7 显示帧率低/画面卡顿	33
1.5.8 显示画面较暗，但背光正常.....	33
1.5.9 MIPI 屏存在画面输出，但出现偏移/翻转现象.....	33
1.5.10 屏幕显示花屏	34
1.5.11 RGB_LCD 显示花屏.....	35
1.5.12 RGB_LCD 显示发绿波纹	36
1.5.13 LT9611 MIPI 转 HDMI 画面倾斜且滚动式播放.....	37
1.5.14 MIPI 转 HDMI 画面细微偏移且偶然性黑屏	37
1.5.15 MIPI 输出画面后熄屏	38
1.6 点屏实例.....	38
1.6.1 点亮 MIPI 屏幕实例	38
1.6.1.1 PWM 点背光示例.....	39
1.6.1.2 VO 时序配置	41
1.6.2 点亮 MCU 4LINE 屏幕实例	41
1.6.2.1 DTS 配置	41
1.6.2.2 硬件改版、GPIO 复用及屏复位	42
1.6.2.3 VO 时序配置	42

插图目录

图 1-1 RGB 屏幕接口示意图.....	1
图 1-2 LINUX 端 SPI/I2C 控制器部署示意图	2
图 1-3 MIPI 屏幕接口示意图	5
图 1-4 LVDS 屏幕接口连线	6
图 1-5 BT1120 屏幕连线	7
图 1-6 MCU LCD TYPB 类型连线	7
图 1-7 MCU LCD TYPC 类型连线	8
图 1-8 芯片对接 RGB LCD 屏幕基本框架图	8
图 1-9 RGB LCD 屏配置流程图	9
图 1-10 芯片对接 MIPI 屏幕基本框架图	10
图 1-11 MIPI 屏配置流程图	10
图 1-12 芯片对接 LVDS 屏幕基本框架图	11
图 1-13 LVDS 屏配置流程图	11
图 1-14 对接 BT2HDMI 转接板基本框架图	12
图 1-15 BT2HDMI 配置流程图	12
图 1-16 对接 MCU 基本框架图	13
图 1-17 MCU 配置流程图	13
图 1-18 屏的初始化数据示意图	15
图 1-19 dts mipi 屏参数据结构_normal	15
图 1-20 dts mipi 屏参数据结构_packet	16

图 1-21 dts mcu 屏参数数据结构	17
图 1-22 某 RGB 屏时序配置示意图.....	21
图 1-23 VO User 时序下的 LCD 像素区域示意图	21
图 1-24 GPIO 方向控制寄存器示意图.....	26
图 1-25 水平 colorbar 样式.....	28
图 1-26 翻转及偏移现象.....	34
图 1-27 花屏现象	35
图 1-28 花屏现象	36
图 1-29 花屏现象	36
图 1-30 滚动画面	37
图 1-31 PWM 点背光流程.....	39
图 1-32 J310BP_RB_A IC 原理图	39
图 1-33 PIN_OUT 管脚寄存器.....	40

表格目录

表 1-1 xmedia_vo_user_sync_timing 结构体赋值示意表 21

1 屏幕对接指南

1.1 屏幕类型与接口

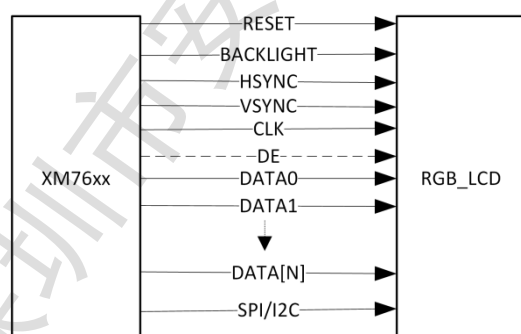
1.1.1 RGB LCD 屏幕接口

1.1.1.1 RGB LCD 屏幕连线

板端与屏幕一般需要四种连线，各执行不同的功能。如图 1-1 所示。

- 控制命令传输接口(对应图中 SPI OR I2C)
- 图像数据传输接口（对应图中 HSYNC VSYNC CLK DATA0~DATAN DE）
- 背光控制（对应图中 BACKLIGHT）
- 硬件复位接口（对应图中 RESET）

图1-1 RGB 屏幕接口示意图



1.1.1.2 RGB LCD 屏参数据接口使能确认

对接 RGB LCD 屏幕时，需要确认控制线 I2C 或者 DBI 的设备节点是否存在，时钟是否打开。

步骤 1 查询 I2C/SPI 设备节点。

查询 LINUX 端部署情况，可以在串口 ls /dev，如图 1-2 所示。

图1-2 LINUX 端 SPI/I2C 控制器部署示意图

userproc	ram7	tty50
i2c-2	ram8	tty51
i2c-3	ram9	tty52
i2c-7	random	tty53
ipcm	root	tty54
kmem	rtc0	tty55

以上图为例，在 LINUX 端部署了 I2C-2，I2C-3，I2C-7，并没有部署 SPI 设备，所以在 LINUX 端使用 SPI 设备或者没有部署的 I2C 设备就会有异常。

须知

当两端同时部署和使用相同的 I2C/SPI 控制器的时候，就会发生不可预知的异常。所以建议两端分别部署 SPI/I2C 控制器。

步骤 2 配置 I2C/SPI 设备。

当 SPI/I2C 控制器部署关系，不符合业务需求。如屏幕驱动部署在 SPI2，但是 LINUX 端并没有部署 SPI2 的时候，就需要更改控制器的部署关系。

以在 LINUX 端增加 SPI0 的部署为例：

首先，打开 sdk/source/kernel/linux-5.10.y/lotus/machine/xmfalcon/dts/xmfalcon.dtsi 文件，在 aliases 结构体中，增加 spi0=&spi_bus0;

增加 spi0 如下：

```
#include <dt-bindings/clock/xmfalcon-clock.h>
/ {
    aliases {
        spi0 = &spi_bus0;
        serial0 = &uart0;
        serial1 = &uart1;
        serial2 = &uart2;
        i2c0 = &i2c_bus0;
        i2c1 = &i2c_bus1;
        i2c2 = &i2c_bus2;
        spi1 = &spi_bus1;
    };
};
```

其次，在本文件下方 spi_bus0 的定义中，将状态从 disabled 改成 okay。

增加 SPI0 定义如下:

```
spi_bus0: spi@12070000 {
    compatible = "arm,pl022", "arm,primecell";
    arm,primecell-periphid = <0x00800022>;
    reg = <0x12070000 0x1000>;
    interrupts = <0 30 4>;
    clocks = <&clock XMFALCON_SPI0_CLK>;
    clock-names = "apb_pclk";
    #address-cells = <1>;
    #size-cells = <0>;

#ifdef CONFIG_EDMAC
    dmas = <&edmac 13 13>, <&edmac 12 12>;
    dma-names = "tx","rx";
#endif

    status = "okay";
};
```

最后重新编译和烧写内核。

步骤 3 配置 I2C/SPI 管脚复用和时钟门控。

- 首先, 先配置 I2C/SPI 的管脚复用状态, 将对应的 PIN 脚复用成 I2C/SPI, 参考《PIN_OUT.xlsx》进行修改。
- 其次, 打开 I2C/SPI 的时钟, 参考《MPP 媒体处理软件开发参考》的“系统”时钟章节, 找到对应的 I2C/SPI 时钟门控所在的寄存器和位, 配置对应的使能。

---结束

1.1.1.3 RGB LCD 屏幕参数控制命令传输接口

屏参数数据传输接口可以用来更改或者获取屏幕的寄存器配置。阅读 RGB LCD 屏规格手册, 可以知道控制命令传输的接口类型, 一般为 SPI 或者 I2C, 部分无需额外配置的屏幕则可能不配备此类接口。

1.1.1.4 RGB LCD 数据传输接口

板端利用该接口向 LCD 屏发送图像数据, 俗称 RGB 接口, 接口有 HSYNC VSYNC CLK DATA0~DATAN 等信号构成, 而 DE 信号是否需要则取决于屏幕。一般 RGB 接口又被根据 RGB 信号是否经由 DATA 线分时复用发送, 分为串行 RGB 和并行 RGB, 串行 RGB 即 R、G、B 分量在不同时刻传输, 并行 RGB 的 R、G、B 颜色分量在同一时刻传输。时钟线 HSYNC、VSYNC、CLK 用来保证 RGB 数据按正确时序由主芯片端向 LCD 传输。

复位线 RESET，用来进行屏幕复位。

数据线，用来传输 RGB 数据。传输模式支持串行 6bit、串行 8bit、并行 16bit RGB565、并行 18bit RGB666、并行 24bit RGB888 这五种。

须知

有关 DE 模式说明：

- DE 模式是通过多接一根到芯片输出到 RGB LCD 上的信号线来准确通知 RGB LCD 屏行有效区的起始和结束点，即图 1-1 中的 DE 线。
- 因此当选择 RGB LCD 屏 DE 模式时，并确保硬件的 DE 管脚正确连接到 RGB LCD 屏上，水平消隐区对屏来说是不重要的。
- SDK 包默认配置为 DE 模式，故当 RGB LCD 屏选择 DE 模式，芯片这边无需做相关配置；若 RGB LCD 屏选择非 DE 模式，也无需再芯片做开关 DE 模式配置，因为 RGB LCD 屏已忽略 DE 信号。

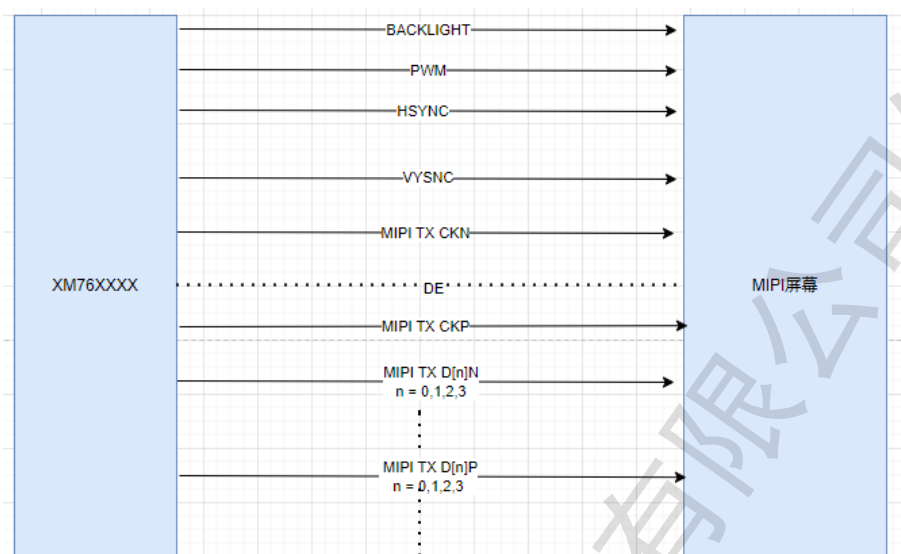
1.1.2 MIPI 屏幕接口

1.1.2.1 MIPI 屏幕连线

板端与 MIPI 屏幕一般需要三种连线，各执行不同的功能，如图 1-3。

- 控制命令和图像数据传输接口（对应图中 HSYNC VSYNC CLK MIPI_TX_D[0]P~MIPI_TX_D[n]P MIPI_TX_D[0]N~ MIPI_TX_D[n]N DE）
- 背光控制（对应图中 BACKLIGHT，PWM）
- Mipi phy 复位接口（对应图中 RESET）

图1-3 MIPI 屏幕接口示意图



1.1.2.2 MIPI 屏幕数据传输接口

我们现在主要使用 MIPI DSI 接口，兼容 DSI 接口的外围设备至少支持两种基本操作模式，command 模式与 video 模式中的一种。在 video mode 传输数据时，DSI 支持这三种包数据或者包顺序的传输方式：Non-burst Mode Sync pulses，Non-burst Mode Sync event 与 Burst mode。

为了传输不同的带宽数据，所以数据 lane 有 1-4 条，板端利用该接口向 mipi 屏传送图像数据。DSI 接口会将并行数据图像转换成数据包形式，同时加上包协议信息，包头传给 phy，然后 phy 再将数据串行化，通过 lane 发送给外围设备。

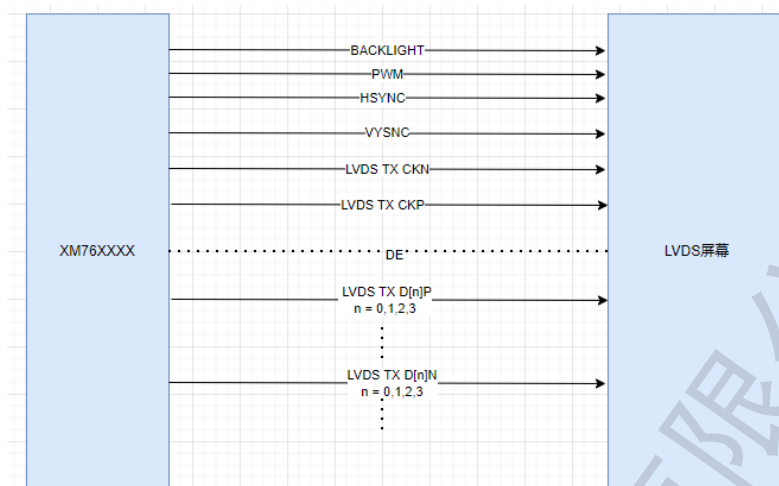
1.1.3 LVDS 屏幕接口

1.1.3.1 LVDS 屏幕连线

板端与 LVDS 屏幕一般需要两种连线，各执行不同的功能。如图 1-4 所示。

- 图像数据传输接口（对应图中 HSYNC VSYNC CLK LVDS_TX_D[0]P~LVDS_TX_D[n]P LVDS_TX_D[0]N~LVDS_TX_D[n]N DE）
- 背光控制（对应图中 BACKLIGHT，PWM）

图1-4 LVDS 屏幕接口连线



1.1.3.2 LVDS 屏幕控制接口

对接 lvds 屏幕需要确认 lvds phy 配置是否正确。

1.1.3.3 LVDS 屏幕数据传输接口

Lvds 采用串行方式传输数据。Lvds 信号有 VESA 标准和 JEIDA 两种标准，两种标准主要是数据包的封装标准顺序不一样，但是 DE 位置一定一样的。我们目前接的是五通道 LVDS，可以支持 6bit 和 8bit 输出模式。

1.1.4 BT656/1120 屏幕接口

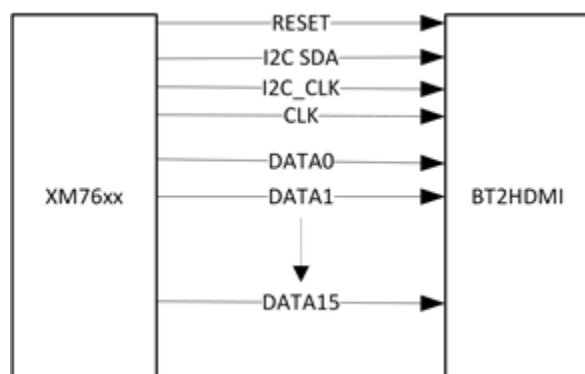
1.1.4.1 BT1120 屏幕连线

以 BT1120 接 BT1120 转 HDMI 转接板（简称 BT2HDMI）为例做说明。

板端与 BT2HDMI 连接，一般需要用到的 RESET/I2C/BT1120 三组线。

- RESET，用于对 BT2HDMI 复位
- I2C 用于和 BT2HDMI 通信
- BT1120 用于图像数据传输

图1-5 BT1120 屏幕连线



1.1.5 MCU LCD 屏幕接口

1.1.5.1 屏幕连线

MCU LCD TYPB 类型连线，如图 1-6：

DBI_CS 片选信号，低有效。

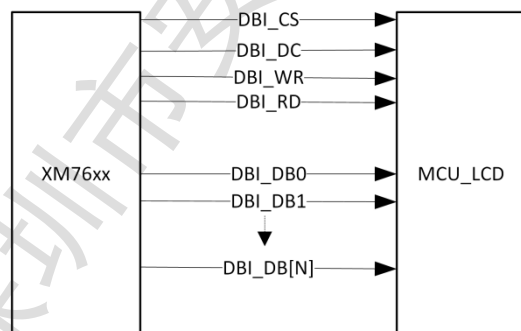
DBI_DC 数据/命令，高：数据，低：命令。

DBI_WR 主控读信号。

DBI_RD 主控写信号。

DBI_DB0 – DBI_DB17 数据信号。

图1-6 MCU LCD TYPB 类型连线



MCU LCD TYPEC 类型连线，如图 1-7：

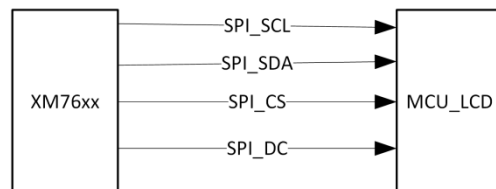
SPI_SCL 时钟线。

SPI_SDA 数据线。

SPI_CS 片选信号，低有效。

SPI_DC 数据/命令，高：数据，低：命令。

图1-7 MCU LCD TYPIC 类型连线



1.2 屏幕框架说明

在这一章中将对屏幕对接时在软件方面需要进行的配置进行说明。

1.2.1 RGB LCD 屏幕框架

图 1-8 介绍了芯片对接 RGB LCD 屏幕的基本框架，客户主要需要操作的部分为用户程序配置层和加载屏幕驱动两块。

图1-8 芯片对接 RGB LCD 屏幕基本框架图

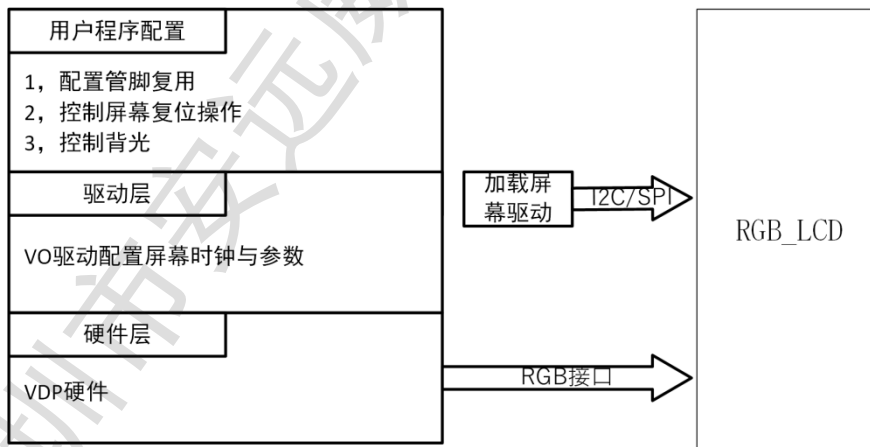
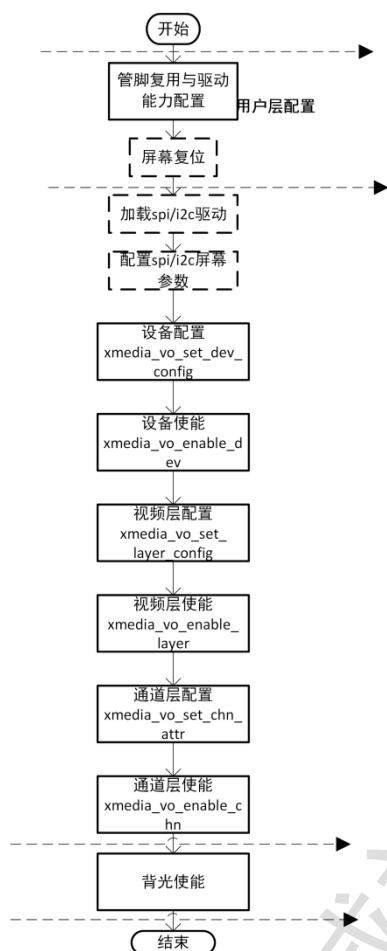


图 1-9 为 RGB LCD 屏配置流程图，提供各操作流程进行的先后次序。

图1-9 RGB LCD 屏配置流程图



须知

“spi/i2c 屏参初始化” 此步骤在某些屏上可能需要调整到 xmedia_vo_enable_dev 之后。

1.2.2 MIPI 屏幕框架

图 1-10 介绍了芯片对接 MIPI 屏幕的基本框架，客户主要需要操作的部分为用户程序配置层的内容和驱动层两部分。

图1-10 芯片对接 MIPI 屏幕基本框架图

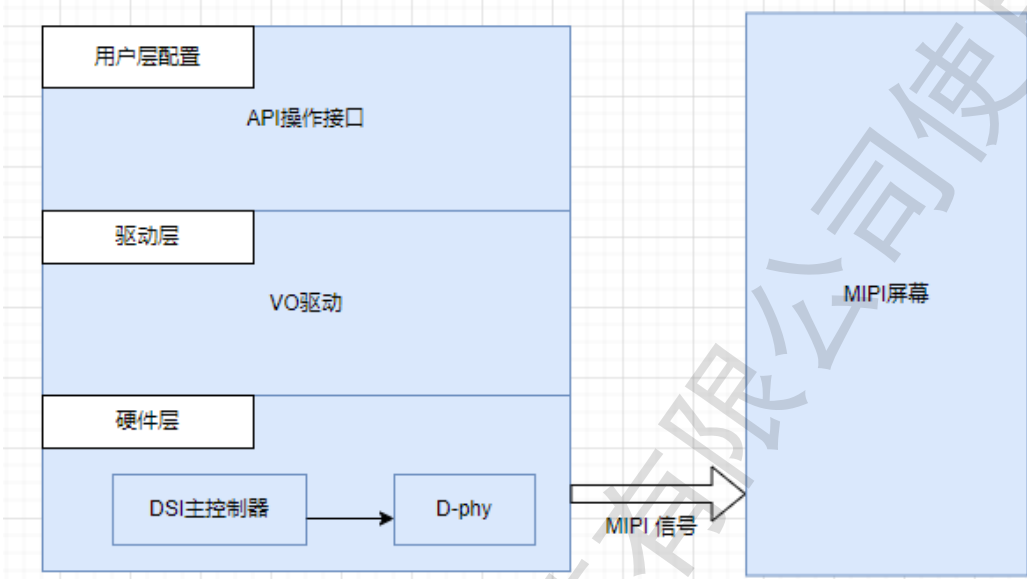
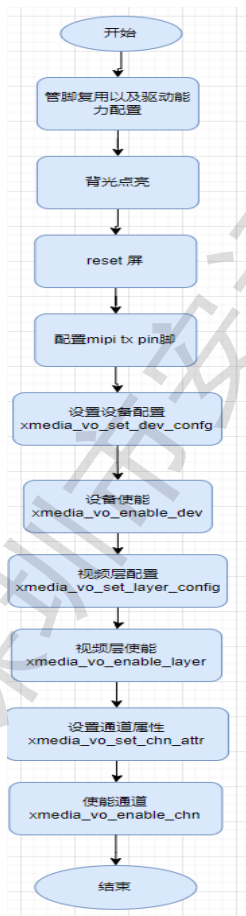


图 1-11 为 MIPI 屏配置流程图，提供各操作流程进行的先后次序。

图1-11 MIPI 屏配置流程图



1.2.3 LVDS 屏幕框架

图 1-12 介绍了芯片对接 LVDS 屏幕的基本框架，客户主要需要操作的部分为用户程序配置层的内容和驱动层两部分。

图1-12 芯片对接 LVDS 屏幕基本框架图

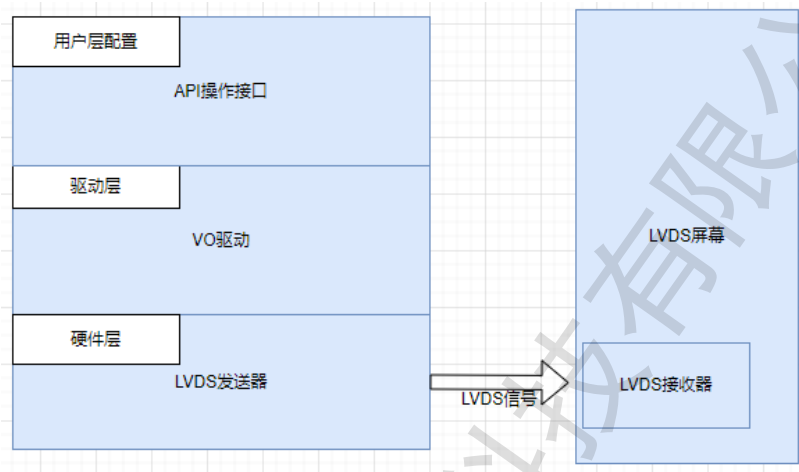
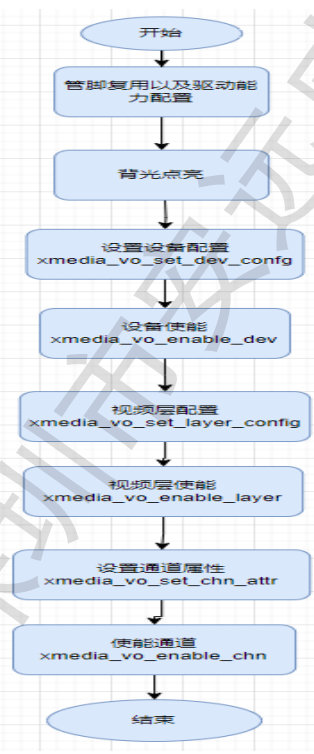


图 1-13 为 LVDS 配置流程图，提供各操作流程进行的先后次序。

图1-13 LVDS 屏配置流程图



1.2.4 BT656/1120 屏幕框架

图1-14 对接 BT2HDMI 转接板基本框架图

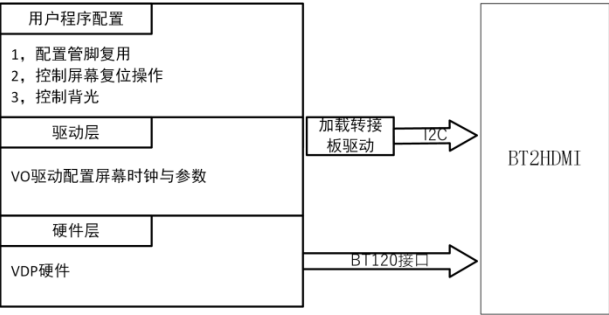
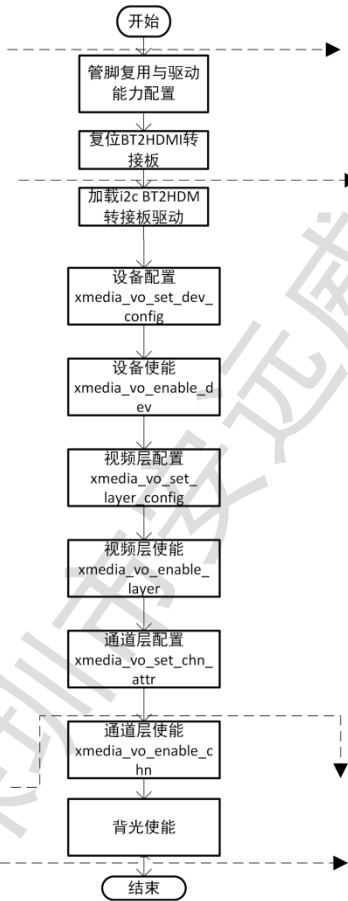


图 1-15 为 BT2HDMI 配置流程图，提供各操作流程进行的先后次序。

图1-15 BT2HDMI 配置流程图



1.2.5 MCU 屏幕框架

图1-16 对接 MCU 基本框架图

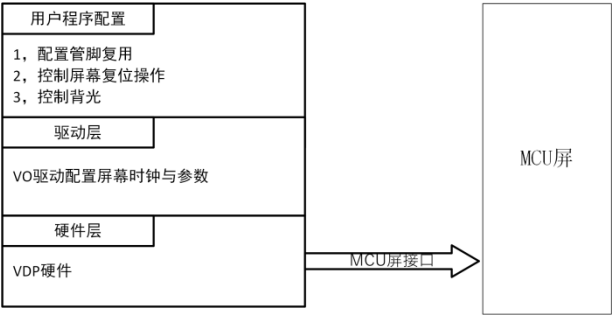
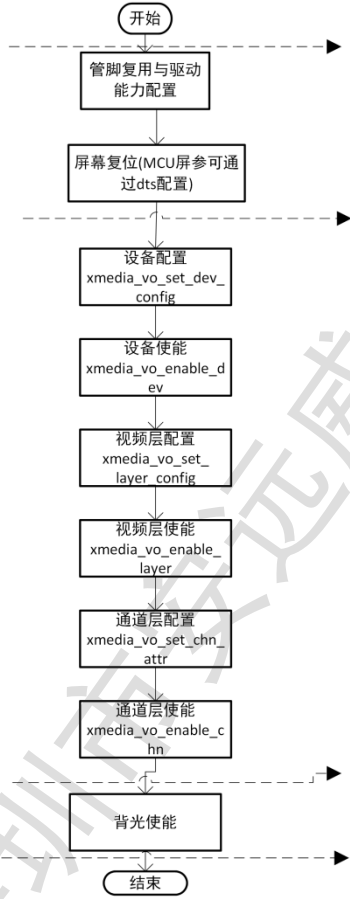


图1-17 MCU 配置流程图



1.3 配置流程

1.3.1 硬件准备与基础配置

1.3.1.1 硬件连线确认

- 确认硬件设计按照《CB3200XX 单板用户指南》的要求，且连线无异常。
- 确认硬件设计按照《PIN_OUT.xlsx》的要求，配置正确的管脚，否则会导致错误管脚无数据输出，显示异常。

1.3.1.2 管脚复用和驱动能力配置

对于所有的屏，都可通过硬件原理图找到显示屏与主芯片的 I2C 或者 SPI 接口管脚，以及其数据交互管脚；然后通过查找主芯片的管脚信息文档《PIN_OUT.xlsx》，使用配置寄存器的形式，配置管脚复用和驱动能力。

须知

管脚驱动能力配置取决于管脚对接外设，如果配的过大可能会导致时序波形有较大过冲，从而导致管脚受损，配的太小可能导致波形无法达到时序要求，因此，建议将驱动能力从小到大调试到符合其管脚对应时序要求。

1.3.2 屏幕参数配置

1.3.2.1 复位配置

由于屏幕一般复位管脚用的是 GPIO 口。所以需要对 GPIO 口进行配置，同时进行屏的复位操作。查询硬件连接图，获取复位的 GPIO 管脚。配置复位所用的 GPIO 方向为输出（输出输入，是相对于主芯片来说的）。

屏的复位操作方法需要参考器件的说明书，若无复位操作，屏幕可能会无法点亮。

1.3.2.2 屏幕控制参数配置

spl 数据传输

屏幕一般会有初始化的过程，比如 RGB LCD 屏一般通过 I2C 或者 DBI 发送数据或者命令来进行屏幕初始化，初始化数据由屏厂提供，本质就是向屏幕发送配置信息的过程。

以 ST7789 RGB LCD 屏为例，图 1-18 为某款 RGB LCD 屏厂提供的初始化数据的一部分，初始化数据可体现以下信息：

图1-18 屏的初始化数据示意图

```
.....ssp_write_cmd(0x11);
.....msleep(150);
.....//-----ST7789v·display·setting-----
.....ssp_write_cmd(0x36);
.....ssp_write_dat(0x00,·1);

.....//-----ST7789S·Frame·rate·setting-----
.....ssp_write_cmd(0xb2);
.....ssp_write_dat(0x00,·1);
.....ssp_write_dat(0x00,·1);
.....ssp_write_dat(0x00,·1);
.....ssp_write_dat(0x33,·1);
.....ssp_write_dat(0x33,·1);
```

屏的初始化数据一般包括像素格式、数据刷新方向、Gamma 配置等，初始化数据中每一个指令具体含义，请在屏幕驱动 IC 的说明书中查找。

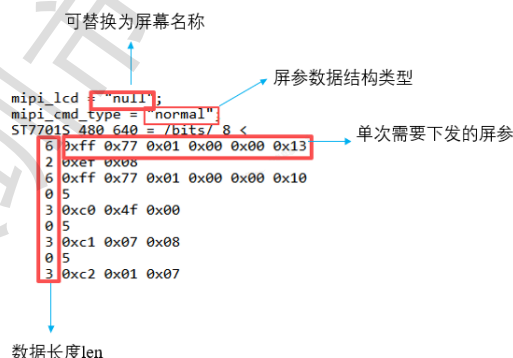
dts 节点匹配

1) 如果需要额外下发 mipi 屏幕参数，可以采用 dts 节点匹配的方式，即 vo 驱动匹配 dts 节点获取屏幕参数，通过 MIPI DSI 下发到 mipi 屏；

以 XM7606V13 为例，用户可在 sdk\source\kernel\linux-5.10.y\lotus\machine\xmfalcon\dts\xmfalcon.dtsi（注意：具体 dtsi 文件名称及路径和芯片型号有所差异）文件下进行配置，根据屏参下发需求，mipi_cmd_type 字段可以选择 “normal” 、 “packet” 两种 dts 屏参数据结构类型；

dts 节点 normal 数据结构如图 1-19 所示：

图1-19 dts mipi 屏参数据结构_normal



mipi_lcd 可以赋值为对应的屏幕名称，如 mipi_lcd 字段替换为 ST7701S_480_640，驱动即可获取对应的参数信息。

每行的第一个元素是单次下发屏参的数据长度，当数据长度为 0 时，后面的数字为睡眠的时间，单位是 ms；当数据长度不为 0 时，后面则是单次需要下发的屏幕参数。
normal 数据结构默认采用 DCS 数据包下发。

dto 节点 packet 数据结构如图 1-20 所示：

图1-20 dto mipi 屏参数据结构_packet

```
mipi_lcd = "ST7701S 480 640";
mipi_cmd_type = "packet";
ST7701S 480 640 = /bits/ 8 <
0x39 6 0xff 0x77 0x01 0x00 0x00 0x13
0x15 2 0xef 0x08
0x39 6 0xff 0x77 0x01 0x00 0x00 0x10
0 0 5
0x39 3 0xc0 0x4f 0x00
0 0 5
0x39 3 0xc1 0x07 0x08
0 0 5
0x39 3 0xc2 0x01 0x07
```

与 normal 数据结构不同点在于，每行的第一列元素可以指定 mipi 下发的数据包格式：

- 0x13: Generic short write, 通用短数据包
- 0x15: DCS short write, DCS 短数据包
- 0x29: Generic long write, 通用长数据包
- 0x39: DCS long write, DCS 长数据包

其余配置则与 normal 数据结构一致，用户可根据屏参需求，自行选择合适的数据结构去适配；

2) 如果需要额外下发 MCU 屏幕参数，可以采用 dto 节点匹配的方式，即 vo 驱动匹配 dto 节点获取屏幕参数，通过 MCU LCD 控制器下发到 MCU 屏；

以 XM7206V11A 为例，用户可在 sdk\source\kernel\linux-

5.10.y\lotus\machine\xmorca\dto\xmorca.dto (注意：具体 dto 文件名称及路径和芯片型号有所差异) 文件下进行配置，dto 节点数据结构如图 1-21 所示：

图1-21 dts mcu 屏参数据结构

```
vo: vo@11280000 {
    compatible = "lotus,vo";
    reg = <0x11280000 0x40000>;
    reg-names = "vo";
    interrupts = <0 52 4>;
    interrupt-names = "vo";
    mcu_lcd = "ILI9342C_KD26_320_240_4LINE";
    ILI9342C_KD26_320_240_18BIT = /bits/ 8 <
        5 0xc8 0x03 0xff 0x93 0x42
        3 0x36 0x01 0x08
        3 0x3a 0x01 0x66
        4 0xc0 0x02 0x0c 0x06
        3 0xc1 0x01 0x20
        3 0xc5 0x01 0xe7
        4 0xb1 0x02 0x00 0x1b
        3 0xb4 0x01 0x02
        4 0xb6 0x02 0x0a 0xe0
        17 0xe0 0x0f 0x00 0x04 0x09 0x05 0x13 0x09 0x35 0x69 0x46 0x04 0x0c 0x09 0x15 0x16 0x0f
        17 0xe1 0x0f 0x00 0x1f 0x20 0x03 0x0f 0x05 0x38 0x55 0x4c 0x04 0x0e 0x0b 0x37 0x37 0x0f
        6 0x2a 0x04 0x00 0x00 0x01 0x3f
        6 0x2b 0x04 0x00 0x00 0x00 0xef
        1 0x11
        0 200
        1 0x29
    >;
    ILI9342C_KD26_320_240_4LINE = /bits/ 8 <
        5 0xc8 0x03 0xff 0x93 0x42
        3 0x36 0x01 0x08
        3 0x3a 0x01 0x55
        4 0xc0 0x02 0x0c 0x06
        3 0xc1 0x01 0x20
        3 0xc5 0x01 0xe7
        4 0xb1 0x02 0x00 0x1b
        3 0xb4 0x01 0x02
        4 0xb6 0x02 0x0a 0xe0
        17 0xe0 0x0f 0x00 0x04 0x09 0x05 0x13 0x09 0x35 0x69 0x46 0x04 0x0c 0x09 0x15 0x16 0x0f
        17 0xe1 0x0f 0x00 0x1f 0x20 0x03 0x0f 0x05 0x38 0x55 0x4c 0x04 0x0e 0x0b 0x37 0x37 0x0f
        6 0x2a 0x04 0x00 0x00 0x01 0x3f
        6 0x2b 0x04 0x00 0x00 0x00 0xef
        1 0x11
        0 200
        1 0x29
    >;
};
```

以 ILI9342C_KD26_320_240_4LINE 第一条指令为例，VO 驱动默认是不加载 MCU 屏参，需要将 mcu_lcd= "null" 修改为 mcu_lcd = "ILI9342C_KD26_320_240_4LINE"。

做命令结构说明：

命令 "5 0xc8 0x03 0xff 0x93 0x42" 。

5：表示整个命令长度即 "0xc8 0x03 0xff 0x93 0x42" 个数为 5。

0xc8：表示 mcu 屏参指令。

0x03：表示 0xc8 指令所跟的数据长度即 "0xff 0x93 0x42" 个数为 3。

0xff 0x93 0x42：表示会 0xc8 指令所跟的数据。

命令 "0 200"

0：表示延时。

200：延时 200ms。

1.3.2.3 VO 参数与时序配置

User 时序配置

当对接一款屏幕时，请使用 User 时序接口的方式。相关接口的使用方法在《MPP 媒体处理软件 开发参考》中“视频输出”章节中可以查到。配置设备参数 `xmedia_vo_dev_config`，结构体如下所示。

```
typedef struct {
    xmedia_intf_mipi_dsi_data_type data_type;
    xmedia_bool lane_pn_swap[XMEDIA_VO_MIPI_LANE_MAX];
    xmedia_vo_mipi_lane lane_sel[XMEDIA_VO_MIPI_LANE_MAX];
    xmedia_s32 mipi_htotal_adjust;
    xmedia_s32 mipi_vsync_adjust;
    xmedia_vo_mipi_lane_num mipi_lane_num;
} xmedia_vo_mipi_attr;
typedef struct {
    xmedia_intf_lvds_data_type data_type;
    xmedia_intf_lvds_format format;
    xmedia_vo_lvds_lane lane_sel[XMEDIA_VO_LVDS_LANE_MAX];
    xmedia_bool lane_pn_swap[XMEDIA_VO_LVDS_LANE_MAX];
} xmedia_vo_lvds_attr;
typedef struct {
    xmedia_intf_type intf_type;
    union {
        xmedia_vo_bt656_attr bt656_attr;
        xmedia_vo_bt1120_attr bt1120_attr;
        xmedia_vo_lcd_attr lcd_attr;
        xmedia_vo_mipi_attr mipi_attr;
        xmedia_vo_lvds_attr lvds_attr;
    };
} xmedia_vo_intf_config;
typedef struct {
    xmedia_u32 bg_color; //设备的背景色，RGB格式
    xmedia_vo_intf_config intf_config; //输出接口属性
    xmedia_vo_intf_sync intf_sync; //输出接口的时序
    xmedia_vo_user_sync_config user_sync_config; //用户同步属性
} xmedia_vo_dev_config;
```

- 确认用户时序接口类型

接口配置结构体 `xmedia_vo_intf_config` 中 `intf_type` 用来设定屏幕类型，比如：
`XMEDIA_INTF_TYPE_LCD`。则需要配置 `xmedia_vo_lcd_attr` 属性。

```
.lcd_attr = {  
    .data_type = XMEDIA_INTF_LCD_DATA_TYPE_RGB888_24BIT,  
    .data_mode = XMEDIA_VO_LCD_DATA_MODE_PARA,  
    .data_seq = XMEDIA_INTF_LCD_DATA_SEQ_RGB,  
},
```

- 确认用户时序类型

`xmedia_vo_intf_sync` 的值表示时序类型，选择
`XMEDIA_VO_INTF_SYNC_USER`。`xmedia_vo_intf_sync` 相关说明在《MPP 媒体处理软件 开发参考》中“视频输出”章节中可以查到。

- 填充用户时序信息

一般来说，如图 1-22 所示，某典型 RGB 屏的 SPEC 上有相关的信息，而对于
VO 来说，User 时序对应的 LCD 像素区域示意图则如图 1-23 所示，用户时序结构体的数据类型为 `xmedia_vo_user_sync_config`。

- 用户自定义帧率 `frame_rate`。（如果时序类型为
`XMEDIA_VO_INTF_SYNC_USER`，则需要配置自定义帧率）。
- 用户自定义时钟翻转 `clk_reverse_en`（时钟是否反向由对接的屏幕决定，调试时可以结合示波器观察 `clk`，并与 RGB LCD 屏幕说明书比对）。
- 用户自定义同步时序结构体 `xmedia_vo_user_sync_timing`。（如果时序类型为
`XMEDIA_VO_INTF_SYNC_USER`，则需要配置自定义同步时序）。

```
//定义用户同步时序  
typedef struct {  
    xmedia_u16 vact; //垂直有效区  
    xmedia_u16 vbb; //垂直消隐后肩  
    xmedia_u16 vfb; //垂直消隐前肩  
  
    xmedia_u16 hact; //水平有效区  
    xmedia_u16 hbb; //水平消隐后肩  
    xmedia_u16 hfb; //水平消隐前肩  
    xmedia_u16 hmid; //底场同步水平有效区域
```

```
xmedia_u16 bvact; //底场垂直有效区

xmedia_u16 bvbb; //底场垂直消隐后肩

xmedia_u16 bvfb; //底场垂直消隐前肩


xmedia_u16 hpw; //水平脉冲宽度

xmedia_u16 vpw; //垂直脉冲宽度


xmedia_bool idv; //数据有效信号的极性

xmedia_bool ihs; //水平有效信号的极性

xmedia_bool ivs; //垂直有效信号的极性


xmedia_bool synm; //同步模式

xmedia_bool iop; //隔行或者逐行显示(0表示隔行, 1表示逐行)

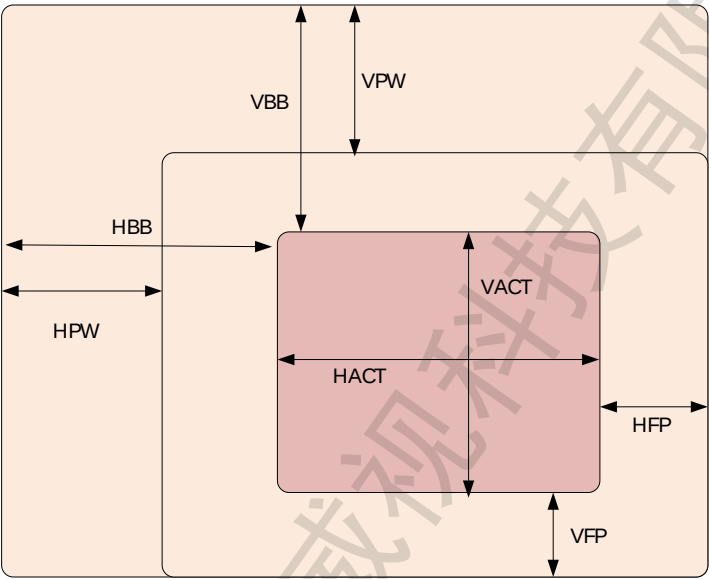
xmedia_u8 intfb; //隔行输出的位宽

} xmedia_vo_user_sync_timing;
```

图1-22 某 RGB 屏时序配置示意图

Parameter	Symbol	Min.	Typ.	Max.	Unit
Horizontal Sync.Width	hpw	2	10	hpn+hbp = hbb	Clock
Horizontal Sync.Back Porch	hbp	4	10		Clock
Horizontal Sync.Front Porch	hfp	2	39		Clock
Vertical Sync.Width	vsa	1	4	vsa + vbp = vbb	Line
Vertical Sync.Back Porch	vbp	1	4		Line
Vertical Sync.Front Porch	vfp	1	8		Line

图1-23 VO User 时序下的 LCD 像素区域示意图



- 图 1-22 中 hpn+hbp 对应图 1-23 中的 HBB
- 图 1-22 中 hfp 对应图 1-23 中的 HFP
- 图 1-22 中 vsa+vbp 对应图 1-23 中的 VBB
- 图 1-22 中 vfp 对应图 1-23 中的 VFP

表1-1 xmedia_vo_user_sync_timing 结构体赋值示意表

成员名称	描述
Synm	同步模式。参数无意义配 0。
lop	0 为隔行时序，1 为逐行时序。
intfb	输出接口位宽。参数无意义配 0。

成员名称	描述
vact	垂直有效区，单位为行。根据屏幕手册可知。
vbb	垂直消隐后肩，单位为行。根据屏幕手册可知 (VSA+VBP)。
vfb	垂直消隐前肩，单位为行。根据屏幕手册可知。
hact	水平有效区，单位为像素。根据屏幕手册可知。
hbb	水平消隐后肩，单位为像素。根据屏幕手册可知 (HSA+HBP) 。
hfb	水平消隐前肩，单位为像素。根据屏幕手册可知。
hmid	底场垂直同步有效像素值。逐行时序时无意义，置为 1。
bvact	底场垂直有效区，逐行时有效，单位为行。逐行时序时无意义，置为 1。
bvbb	底场垂直消隐后肩，逐行时有效，单位为行。逐行时序时无意义，置为 1。
bvfb	底场垂直消隐前肩，逐行时有效，单位为行。逐行时序时无意义，置为 1。
hpw	水平同步信号的宽度，单位为像素。屏幕手册上没特殊说明的话，可以自己设置，加起来消隐区满足屏幕手册要求即可。手册上有说明的就按手册配置。
vpw	垂直同步信号的宽度，单位为行。屏幕手册上没特殊说明的话，可以自己设置，加起来消隐区满足屏幕手册要求即可。手册上有说明的就按手册配置。
idv	数据有效信号的极性。0 为高有效，1 为低有效。
ihs	水平有效信号的极性，0 为高有效，1 为低有效。
ivs	垂直有效信号的极性，0 为高有效，1 为低有效。

标准时序配置

当对接一款屏幕时，可以使用标准时序配置。相关接口的使用方法在《MPP 媒体处理软件 开发参考》中“视频输出”章节中可以查到。配置设备参数 xmedia_vo_dev_config，结构体如下所示。

```
typedef struct {
    xmedia_intf_mipi_dsi_data_type data_type;
    xmedia_bool lane_pn_swap[XMEDIA_VO_MIPI_LANE_MAX];
    xmedia_vo_mipi_lane lane_sel[XMEDIA_VO_MIPI_LANE_MAX];
    xmedia_s32 mipi_htotal_adjust;
    xmedia_s32 mipi_vsync_adjust;
    xmedia_vo_mipi_lane_num mipi_lane_num;
} xmedia_vo_mipi_attr;
typedef struct {
    xmedia_intf_lvds_data_type data_type;
    xmedia_intf_lvds_format format;
    xmedia_vo_lvds_lane lane_sel[XMEDIA_VO_LVDS_LANE_MAX];
    xmedia_bool lane_pn_swap[XMEDIA_VO_LVDS_LANE_MAX];
} xmedia_vo_lvds_attr;
typedef struct {
    xmedia_intf_type intf_type;
    union {
        xmedia_vo_bt656_attr bt656_attr;
        xmedia_vo_bt1120_attr bt1120_attr;
        xmedia_vo_lcd_attr lcd_attr;
        xmedia_vo_mipi_attr mipi_attr;
        xmedia_vo_lvds_attr lvds_attr;
    };
} xmedia_vo_intf_config;
typedef struct {
    xmedia_u32 bg_color; //设备的背景色，RGB格式
    xmedia_vo_intf_config intf_config; //输出接口属性
    xmedia_vo_intf_sync intf_sync; //输出接口的时序
    xmedia_vo_user_sync_config user_sync_config; //用户同步属性
} xmedia_vo_dev_config;
```

- 确认用户时序接口类型

接口配置结构体 xmedia_vo_intf_config 中 intf_type 用来设定屏幕类型，比如：XMEDIA_INTF_TYPE_LCD。则需要配置 xmedia_vo_lcd_attr 属性。

```
.lcd_attr = {
    .data_type = XMEDIA_INTF_LCD_DATA_TYPE_RGB888_24BIT,
    .data_mode = XMEDIA_VO_LCD_DATA_MODE_PARA,
    .data_seq = XMEDIA_INTF_LCD_DATA_SEQ_RGB,
},
```

- 确认用户时序类型

xmedia_vo_intf_sync 的值表示时序类型，选择 XMEDIA_VO_INTF_SYNC_

1080P24。xmedia_vo_intf_sync 相关说明在《MPP 媒体处理软件 开发参考》中“视频输出”章节中可以查到。

标准时序不需要填充用户时序信息。

1.3.2.4 配置背光

一般是通过 PWM 动态调整输出电流，继而控制屏幕背光亮度；关于 PWM 的设置可以参考如下命令（以设置 pwm15 为例）或者使用电阻固定电流、无法动态调整，只能开关背光。

1. 先将 pwm15 设置为 pwm 输出模式

```
# echo 15 > /sys/class/pwm/pwmchip0/export
```

2. 设置 period,默认频率配置为 120KHZ，超过这个默认值有可能会损坏背光芯片

```
# echo 8333 > /sys/class/pwm/pwmchip0/pwm15/period
```

3. 设置默认占空比为 0，后续通过设置占空比调节背光亮度(duty_cycle 值 X = period * N%)

```
# echo 0 > /sys/class/pwm/pwmchip0/pwm15/duty_cycle
```

4. 使能 pwm

```
# echo 1 > /sys/class/pwm/pwmchip0/pwm15/enable
```

5. 关 pwm

```
# echo 0 > /sys/class/pwm/pwmchip0/pwm15/enable
```

须知

1. 默认频率配置为 120KHZ，超过这个默认值有可能会损坏背光芯片，这里最好不要改动。

2. 初始化以后，每次调节占空比的时候都需要执行：设置占空比，关 pwm 然后再开 pwm 的流程来使能生效，否则背光配置无效。
 3. 另外还可以使用电阻固定电流的方式控制背光、但是无法动态调整背光亮度的。
-

1.4 验证与调试

1.4.1 管脚复用与驱动能力调试

步骤 1 如果需要用到 SPI/I2C，需要确认控制接口 SPI/I2C 的接口管脚复用是否配置正常。

步骤 2 确认屏幕的数据接口(LCD_DATA、HSYNC、VSYNC、CLK 等)的接口管脚复用是否配置正常。驱动能力在调试初期可以先配置较低档位，待调亮屏幕后，再调优，以免损坏器件。

可以参照《PIN_OUT.xlsx》和硬件设计图，通过 xmmcl + 寄存器地址，来读出管脚复用状态。

如果状态不符合，请在代码的对应地方，修改不匹配的复用关系。

----结束

1.4.2 复位操作验证

如果需要复位的话，首先确认复位管脚能否正常置高置低，用万用表或者示波器观测均可。

1， 查询硬件设计图，找到复位管脚对应的 GPIO 口。

2， 确认对应的管脚的复用关系为 GPIO。参考《PIN_OUT.xlsx》文档，分别将该 GPIO 口拉高拉低，通过示波器测量引脚状态是否正确。

以拉高拉低 GPIO16 为例：

找到 GPIO16 的寄存器地址为 0x20000904，如图 1-24，再将该寄存器的第 4 位，第 5 位分别配成 1，对应的是上拉和下拉。

图1-24 GPIO 方向控制寄存器示意图

ioof_reg2	N21	Pin LSADC_CH2 IO Config Register.	0x120C0114	0x00001000	31:21	保留。
					20:19	保留。
					18	保留。
					17	管脚保持使能: 0x0: 关闭; 0x1: 打开。
					16	保留。
					15:14	管脚施密特控制: 0x00: ST0, ST1关闭; 0x01: ST0打开, ST1关闭; 0x10: ST0关闭, ST1打开; 0x11: ST0, ST1打开。
					13	保留。
					12	管脚输入使能: 0x0: 关闭; 0x1: 打开。
					11	保留。
					10	管脚电平转换速率控制: 0x0: 快速输出; 0x1: 慢速输出。
					9	管脚下拉控制: 0x0: 关闭; 0x1: 打开。
					8	管脚上拉控制: 0x0: 关闭; 0x1: 打开。
					7:4	管脚驱动能力选择: 0x0: 档位1; 0x1: 档位2; 0x2: 档位3; 0x3: 档位4; 其它: 保留。
					3:0	功能选择: 0x0: LSADC_CH2; 0x1: GPIO5_6; 0x2: I2C2_SDA; 其它: 保留。

3, 通过查询屏幕的说明书, 确认复位操作是否和屏幕要求一致。如先置高, 再置低, 再置高, 并确认各步骤间的延时要求。

1.4.3 背光控制调试

背光控制分两种:

- 一种可以动态调整背光亮度。

动态调整背光亮度的 LCD 屏, 多数情况是通过 PWM 调整占空比来实现亮度调整, 需实现 PWM 驱动。占空比设置命令 (以 pwm15 举例) 如下:

```
echo 0 > /sys/class/pwm/pwmchip0/pwm15/duty_cycle //设置 pwm15 的占空比为 0, 需要使能 pwm15 以后才能调整占空比控制亮度
```

- 一种只能通过硬件配置亮度 (即无法动态配置)。

调试阶段可先将背光调至最亮。

1.4.4 控制屏幕参数通路验证

可以通过以下几种方法确认控制命令发送是否成功:

- 一般屏幕的控制命令发送接口为 SPI 或者 I2C, 可以通过示波器观察命令发送时的波形, 并结合对应协议, 核对发送信息, 并与屏幕说明书指定的信息发送时序比对。
- 可以通过读取一些屏幕的寄存器, 看读取到的寄存器值是否符合预期, 以达到测试调试通路是否正常的目的。可以先将一个寄存器设置成一个特定的值, 再将这

个寄存器读出，看读出的值是否符合预期。需要注意的是有些寄存器是只读的，或者只写的，并不是所有的寄存器都同时支持读写功能，具体需要看屏幕驱动 IC 的说明书手册。

- 通过发送一些特定的命令，开启屏幕的自测功能，比如屏幕的 colorbar 厂测程序等。

1.4.5 VO 输出时钟

VO 的输出时钟，可以通过示波器测量 vo_clk 引脚的频率，通过示波器测量得到的频率应该与通过以下两种方法得的频率一致。

```
cat /proc/umap/vo_dev0
```

查看 crg_clk 数值看是否和示波器相符。

$$\text{crg_clk} = (\text{hact} + \text{hbb} + \text{hfb}) * (\text{vact} + \text{vbb} + \text{vfb}) * \text{framerate} * \text{n_pixle_cycle}$$

须知

n_pixcel_cycle: BT 串行 8bit (BT656)设置为 2, LCD 串行 6bit/8bit 设置为 3, BT1120 和并行 lcd 设置为 1, mcu 屏需根据实际屏规格确认

1.4.6 VO Colorbar 调试

colorbar 是一种重要的调试手段，通过 colorbar 的显示，能够看到特定的色条，有利于分析时序信息异常，颜色异常。

如果 VO 的配置上有问题时大多数都可以在 colorbar 上表现出来。如果 colorbar 上的颜色出现了偏色，则可能 VO 的 CSC 设置有问题，如果 colorbar 的图像显示出现错位，则可能存在时钟时序的问题。如果 colorbar 的显示完全不成型，则屏的设置可能存在问题。

colorbar 的调试手段可以很好的帮助定位问题，定位时需要根据具体现象具体分析。

```
echo pattern en=1 pos=1 mode=2 color=0xff0000 > /proc/umap/vo_dev0
```

图1-25 水平 colorbar 样式



1.5 异常处理 FAQ

1.5.1 屏幕全黑/无反应

【问题现象】

屏幕完全黑色，没有响应。

【问题分析】

有很多因素会导致屏幕无响应，需要逐个环节排查，例如器件损坏、连接异常、供电异常、管脚复用配置异常、reset 管脚状态异常、vo 时序配置参数错误等。

【解决方案】

步骤 1 硬件确认。

找硬件工程师确认屏幕供电、连接正常，且器件无损坏。

步骤 2 确认软件流程的正确性。

请比对 1.3.2.2 屏幕控制参数配置章节的配置屏幕参数，需要特别注意的是如果有 reset 管脚，reset 管脚操作有严格的时间要求，比如有些必须要 120ms 后才能接受控制命令等。

步骤 3 确认管脚复用与驱动能力配置正确，请参考 1.3.1.2 管脚复用和驱动能力配置章节。

步骤 4 确认控制屏幕参数传输通路是否正常，请参考 1.4.4 控制屏幕参数通路验证章节。

步骤 5 确认背光开启，确保背光打开，请参考 1.3.2.4 配置背光章节。

步骤 6 确认 vo 的输出时钟与输出序列是否正确。

请参考 1.3.2.3 VO 参数与时序配置和 1.4.5 VO 输出时钟来检查 vo 输出配置是否正确。

注意：mipi/lvds 需确认线序(lane_sel)是否配置正确、每条 lane 是否有信号传输，mipi 还需确认 P/N 信号(lane_pn_swap)是否翻转以及 lane 线数量 (lane_num) 是否配置正确；

----结束

1.5.2 显示位置错误

【问题现象】

图像只在屏幕上显示出一部分。

【问题分析】

这种情况的表象为实际显示的图像在屏幕上只显示一部分，显示图像起始点不是屏幕的初始点。

【解决方案】

一般是主芯片侧的行或者场的消隐区配置与屏幕的配置未匹配导致的，需要将两者保持一致，可以调整屏幕配置，也可以调整芯片侧配置。

1.5.3 显示裂屏

【问题现象】

显示图像起点位置移至屏幕中间。

【问题分析】

调整前后消隐区大小没有变化。可能是输出像素时钟与时序不匹配。

【解决方案】

通过 1.4.5 VO 输出时钟中的调试手段定位时钟是否正常，如果不对确定时钟有问题。

1.5.4 显示大小错误

【问题现象】

图像在屏幕上显示时出现实际值大小不匹配的情况。

【问题分析】

这种情况表象为实际显示图像的尺寸超过屏幕尺寸（即显示不全）或者小于屏幕尺寸。

【解决方案】

一般是主芯片侧的行或者场的有效区配置与屏幕的配置未匹配导致的，需要将两者保持一致，可以调整屏幕配置，也可以调整芯片侧配置。

1.5.5 显示颜色异常

这种情况表象为图像的颜色与预期发生明显的差别，有多种表象，举例如下。

1.5.5.1 画面完全混乱显示效果无法辨认出目标的轮廓

【问题现象】

图像画面完全混乱，看不出任何的图像效果。

【问题分析】

- 1，这种情况表现多为接口类型选择有误，比如主芯片侧选用了 BT656 接口发送，而屏幕端选择了 RGB 类型。
- 2，如果屏幕接口类型正确的话，还需要确认屏幕 HS，VS 极性，极性反了也有可能导致画面完全混乱。

【解决方案】

- 1，请核对设置的 VDP 接口类型与所选择的接口是否一致。
- 2，可以尝试修改 xmedia_vo_user_sync_timing 结构体的 ihs 和 ivs 成员。

1.5.5.2 出现颜色发生变化

【问题现象】

RGB LCD 显示图像出现红、蓝、绿出现错乱的情况。

【问题分析】

这种情况可能为 RGB 发送顺序未匹配。

【解决方案】

对于串行 RGB 来说，可以尝试以下几种解决方式：

- 请确认屏幕发送 RGB 的顺序和屏幕接受 RGB 顺序一致，有些屏幕支持 RGB 接收顺序的调整，这种情况的话，更改屏幕寄存器配置即可。
- clk 输出的极性如果和屏幕预期的不一致，会导致 rgb 顺序错乱，可以尝试修改 xmedia_vo_set_dev_config 接口中 xmedia_vo_user_sync_config 结构体的 clk_reverse_en 成员。

对于并行 RGB 来说

- 先确认硬件的线序是否正确，是否出现 RGB 排列错误。
- 其次看当前的软件配置，是否能保证主芯片侧 RGB 的排序和屏幕接受 RGB 的排序一致。

假如出现了不一致的情况，可以考虑调整主芯片侧或者屏幕侧的排序（有些屏幕支持调整）。对于主芯片侧 RGB 的排序配置调整方法，请参考《MPP 媒体处理软件开发参考》中的“视频输出”章节的 xmedia_vo_lcd_attr; xmedia_intf_lcd_data_seq data_seq; 来调整 RGB 顺序显示颜色异常

【问题现象】

图像显示的颜色不正常，颜色转换方式不对。

【问题分析】

这种情况有可能是视频层的 CSC 没有配置正确，LCD 或者 MIPI 屏幕需要转换 RGB 的 CSC 输出才行。

【解决方案】

调用 xmedia_vo_set_dev_csc 函数设置 CSC 属性。设置方法可参考《MPP 媒体处理软件开发参考》中的“视频输出”章节中相关接口的操作说明。

1.5.5.3 显示线条有色差

【问题现象】

显示图像的轮廓线处有色差。

【问题分析】

这种情况的表象为线条颜色发生异常，特别是图像的轮廓线，受相邻像素的影响，这种情况一般为时序有细微错误，导致数据传输的时候，该像素的信息和相邻像素的信息发生了传输错误，这种情况有多种可能。

【解决方案】

硬件干扰较大，请先用示波器看看是否是波形符合预期，如果不符合，可能是走线过长导致（一般飞线才可能出现这种情况），或者是管脚驱动能力配置不足。

视频输出的 clk 相位错误，未和屏幕匹配，需要将主芯片侧和屏幕侧调整到时序正确。

1.5.5.4 屏幕显示效果不佳

【问题现象】

图像显示时在亮度、色度、对比度等方面达不到想要的效果。

【问题分析】

屏幕显示效果不够满意。

【解决方案】

调节屏幕显示效果，需要从两个角度出发：

- 调整主芯片侧的图像输出效果。
- 调整屏幕侧的显示效果参数。

主芯片侧的显示效果可以通过调整 vo 的参数，可以调整亮度对比度等，屏幕侧一般可以调整 gamma 曲线，亮度、色度等，一般屏幕厂商会给出推荐配置，能满足大部分场景需求。

1.5.6 屏幕画面转换时出现残影

【问题现象】

屏幕显示的图像在变换的过程中出现上一帧画面的残留。

【问题分析】

残影一般由 vcom 电压配置不合理或者一些供电管脚电压异常导致。

【解决方案】

请查看屏幕手册说明或者屏幕厂商获取解决办法。

1.5.7 显示帧率低/画面卡顿

【问题现象】

显示的画面出现卡顿的情况。

【问题分析】

一般这种情况是数据时钟频率不够。

【解决方案】

请调整时钟频率。

1.5.8 显示画面较暗，但背光正常

【问题现象】

图像显示画面较暗，但是背光确实正常的。

【问题分析】

可能 RGB 屏部分 DATA 线无数据，硬件设计的时候，没有参考 XMxx 配置正确的管脚，导致屏幕与主芯片间部分管脚无数据；同时，可能存在 PWM 背光占空比设置较小导致显示画面较暗的情况。

【解决方案】

出现画面较暗的原因较多，遇到这种问题时，可以先排除上述问题分析的可能性。

1.5.9 MIPI 屏存在画面输出，但出现偏移/翻转现象

【问题现象】

当按照屏幕手册配置正确的时序后，图像显示画面依然没有占据全屏，出现偏移以及较小角度的翻转。

图1-26 翻转及偏移现象



【问题分析】

这种情况出现较少，但还是时序配置问题。

【解决方案】

参考 1.3.2.3 VO 参数与时序配置，对 `mipi_h toal_adjust`、`mipi_vsync_adjust` 值在[0,10] 的范围内进行微调，观察画面变化。

1.5.10 屏幕显示花屏

【问题现象】

图像显示画面出现花屏。

图1-27 花屏现象



【问题分析】

可能是 vo 时钟频率配置不对。

【解决方案】

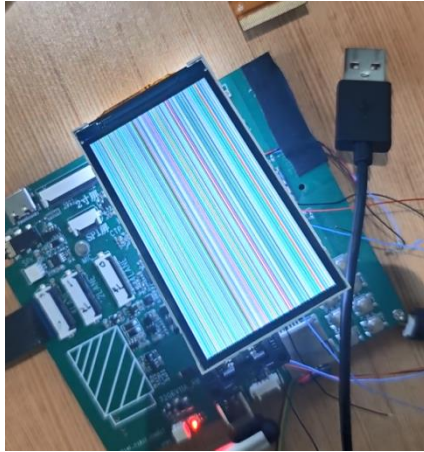
vo 时钟频率与 vo 时序密切相关，因此需要再次确认 vo 时序是否配置正确。

1.5.11 RGB_LCD 显示花屏

【问题现象】

图像显示乱码，完全看不出轮廓。

图1-28 花屏现象



【问题分析】

可能是相关信号极性不对导致。

【解决方案】

尝试调换 idv ihs ivs 相关极性。

1.5.12 RGB_LCD 显示发绿波纹

【问题现象】

显示图像有少许发绿的波纹。

图1-29 花屏现象



【问题分析】

可能是时钟极性不对导致。

【解决方案】

尝试调换 clk_reverse_en 极性。

1.5.13 LT9611 MIPI 转 HDMI 画面倾斜且滚动式播放

【问题现象】

显示图像倾斜且输出画面会左右滚动式循环播放，且伴随间歇性地黑屏。

图1-30 滚动画面



【问题分析】

主控 MIPI 输出稳定，但转换芯片 9611 到 HDMI 输出画面出现如上情况，是由于主控 MIPI 输出与转换芯片握手存在异常，导致 PCR 信号不稳定。

【解决方案】

1. 调整主控时钟余量，增加 mipi_phy_data 速率，达到转换芯片预期值；
2. 转换芯片厂商对 9611PCR 算法进行调试，若调试不理想，考虑更换转换芯片迭代版本，如 9611ex。

1.5.14 MIPI 转 HDMI 画面细微偏移且偶然性黑屏

【问题现象】

720P、1080P 画面均正常输出，4K 画面细微偏移且偶然性黑屏。

【问题分析】

转换芯片 log 出现 fifo err 打印，可能是 4K 场景 fifo 不足引起的。

【解决方案】

增大转换芯片 fifo。

1.5.15 MIPI 输出画面后熄屏

【问题现象】

MIPI 正常输出画面后，在背光正常的情况下，画面亮度逐渐变暗直至熄屏。

【问题分析】

可能是频偏引起的。

【解决方案】

参考 1.3.2.3 VO 参数与时序配置，对 mipi_h toal_adjust、mipi_vsync_adjust 值在[0,20] 的范围内进行微调，观察画面变化。

1.6 点屏实例

1.6.1 点亮 MIPI 屏幕实例

屏幕类型：SATOZ_SAT935_MIPI_800X1280_RGB8888_24BIT

单板型号：XM7606V13 系列 J310BP_RB_A 版型

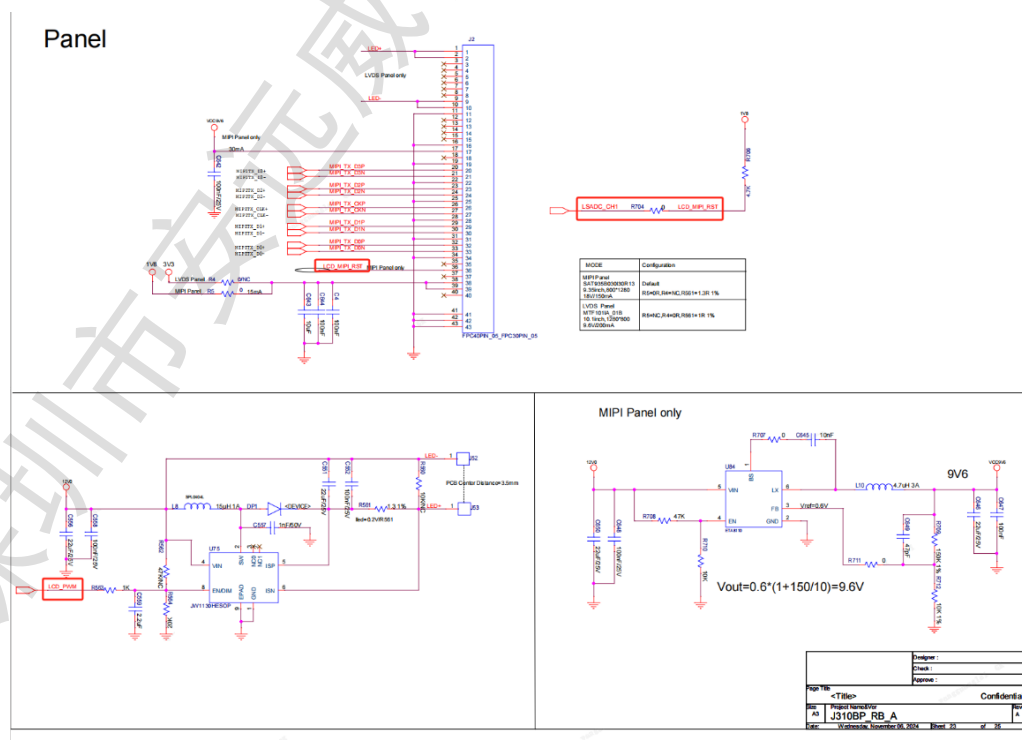
1.6.1.1 PWM 点背光示例

图1-31 PWM点背光流程



如图 1-31 所示，首先需要通过单板型号，找到对应的单板的 IC 原理图，BP_RB_A 单板 IC 原理图如下所示。

图1-32 J310BP RB AIC 原理图



从图 1-32 中的 configuration 中可以看到默认是支持 MIPI800x1280，如果显示不支持，需要借助硬件人员去进行相对应的改板，从而支持 MIPI 点背光；

从图 1-32 可知，需要配置 LCD_MIPI_RST 以及 LCD_PWM 两个管脚复用。如图 1-33 所示，通过《7606V13_PIN_OUT_MIPI.xlsx》，可以找到 LCD_MIPI_RST 以及 LCD_PWM 所对应的寄存器分别为 0x11980068、0x120c0110，再将其低四位位分别配置为 0x2、0x1，即可完成管脚复用：

- xmmm 0x11980068 0x1032
- xmmm 0x120c0110 0x1001

图1-33 PIN_OUT 管脚寄存器

iocfg_reg28	E20	Pin VART2_RTSN IO Config Register.	0x11980068	0x00001000	31:21	保留。
					20:19	保留。
					18	保留。
					17	保留。
					16	保留。
					15:14	管脚施密特控制： 0x0: ST关闭； 0x1: ST打开； 其他: 保留。
					13	保留。
					12	管脚输入使能： 0x0: 关闭； 0x1: 打开。
					11	保留。
					10	管脚电平转换速率控制： 0x0: 快沿输出； 0x1: 慢沿输出。
					9	管脚下拉控制： 0x0: 关闭； 0x1: 打开。
					8	管脚上拉控制： 0x0: 关闭； 0x1: 打开。
					7:4	管脚驱动能力选择： 0x0: 档位1； 0x1: 档位2； 0x2: 档位3； 0x3: 档位4； 0x4: 档位5； 0x5: 档位6； 0x6: 档位7； 0x7: 档位8； 其它: 保留。
					3:0	功能选择： 0x0: GPIO8_5； 0x1: VART2_RTSN； 0x2: PWM15； 0x3: VOUT120_DATA2； 0x7: I2S_BCLK_TX； 其它: 保留。
					31:21	保留。
iocfg_reg1	N22	Pin LSADC_CH1 IO Config Register.	0x120C0110	0x00001000	20:19	保留。
					18	保留。
					17	管脚保持使能： 0x0: 关闭； 0x1: 打开。
					16	保留。
					15:14	管脚施密特控制： 0x00: ST0, ST1关闭； 0x01: ST0打开, ST1关闭； 0x10: ST0关闭, ST1打开； 0x11: ST0, ST1打开。
					13	保留。
					12	管脚输入使能： 0x0: 关闭； 0x1: 打开。
					11	保留。
					10	管脚电平转换速率控制： 0x0: 快沿输出； 0x1: 慢沿输出。
					9	管脚下拉控制： 0x0: 关闭； 0x1: 打开。
					8	管脚上拉控制： 0x0: 关闭； 0x1: 打开。
					7:4	管脚驱动能力选择： 0x0: 档位1； 0x1: 档位2； 0x2: 档位3； 0x3: 档位4； 其它: 保留。
					3:0	功能选择： 0x0: LSADC_CH1； 0x1: GETIO_4； 0x2: UART5_TXD； 其它: 保留。
					31:21	保留。
					20:19	保留。

从图 1-33 可知，PWM 通道编号为 15，参考 1.3.2.4 配置背光章节进行 PWM 通道设置：

- echo 15 > /sys/class/pwm/pwmchip0/export
- echo 8333 > /sys/class/pwm/pwmchip0/pwm15/period
- echo 0 > /sys/class/pwm/pwmchip0/pwm15/duty_cycle
- echo 1 > /sys/class/pwm/pwmchip0/pwm15/enable

1.6.1.2 VO 时序配置

参考 1.3.2.3 VO 参数与时序配置章节进行 VO 时序配置：

```
vo_config->dev_config.intf_config.intf_type = XMEDIA_INTF_TYPE_MIPI_DSI;
vo_config->dev_config.intf_config.mipi_attr.data_type =
XMEDIA_INTF_MIPI_DSI_DATA_TYPE_RGB888_24BIT;
vo_config->dev_config.intf_config.mipi_attr.lane_sel[0] = XMEDIA_VO_MIPI_LANE_DATA0;
vo_config->dev_config.intf_config.mipi_attr.lane_sel[1] = XMEDIA_VO_MIPI_LANE_DATA1;
vo_config->dev_config.intf_config.mipi_attr.lane_sel[2] = XMEDIA_VO_MIPI_LANE_CLK;
vo_config->dev_config.intf_config.mipi_attr.lane_sel[3] = XMEDIA_VO_MIPI_LANE_DATA2;
vo_config->dev_config.intf_config.mipi_attr.lane_sel[4] = XMEDIA_VO_MIPI_LANE_DATA3;
vo_config->dev_config.intf_config.mipi_attr.lane_pn_swap[0] = 0;
vo_config->dev_config.intf_config.mipi_attr.lane_pn_swap[1] = 0;
vo_config->dev_config.intf_config.mipi_attr.lane_pn_swap[2] = 0;
vo_config->dev_config.intf_config.mipi_attr.lane_pn_swap[3] = 0;
vo_config->dev_config.intf_config.mipi_attr.lane_pn_swap[4] = 0;
vo_config->dev_config.intf_config.mipi_attr.mipi_htotal_adjust = 0;
vo_config->dev_config.intf_config.mipi_attr.mipi_vsync_adjust = 0;
vo_config->dev_config.intf_config.mipi_attr.mipi_lane_num = XMEDIA_VO_MIPI_LANE_NUM4;
vo_config->dev_csc.color_info.color_space = XMEDIA_VIDEO_COLOR_SPACE_RGB;
vo_config->dev_config.user_sync_config.clk_reverse_en = 0;
vo_config->dev_config.user_sync_config.frame_rate = 60;
vo_config->dev_config.intf_sync = XMEDIA_VO_INTF_SYNC_USER;
```

1.6.2 点亮 MCU 4LINE 屏幕实例

屏幕类型：MCU_KD026_320X240P6_4LINE_RGB565

单板型号：XM7206V11A 系列 7206V10_RB_A_VA 版型

1.6.2.1 DTS 配置

参考本文档“[dts 节点匹配](#)”章节，配置 MCU 屏 dts 节点名称。

1.6.2.2 硬件改版、GPIO 复用及屏复位

可参考 sdk 包 “sdk/sample/vo/readme.txt 文档” , “7206V10_RB_A_VA 版型版本” 章节说明。

1.6.2.3 VO 时序配置

参考 1.3.2.3 VO 参数与时序配置章节进行 VO 时序配置：

```
case MCU_KD026_320X240P6_4LINE_RGB565:
    vo_config->dev_config.intf_config.intf_type = XMEDIA_INTF_TYPE_MCU;
    vo_config->dev_config.intf_config.mcu_attr.te_mode_en = XMEDIA_FALSE;
    vo_config->dev_config.intf_config.mcu_attr.data_type =
XMEDIA_VO_MCU_DATA_TYPE_4LINE_RGB565;
    vo_config->dev_csc.color_info.color_space =
XMEDIA_VIDEO_COLOR_SPACE_RGB;
    vo_config->dev_config.user_sync_config.frame_rate = 6;
    vo_config->dev_config.intf_sync = XMEDIA_VO_INTF_SYNC_USER;
```