QM10XD LCD DEBUG USER GUIDE

发布日期 2024/9/24

Revision History

Revision	Date	Author	Description

Table of Contents

1. Introduction

1.1. Request & Purpose

本文着重描述 LCM 驱动结构和 LCD 适配 SOP，涉及 PWM 和 GPIO 相关内容可参考对应模块 SOP 文档。

1.2. Definitions &Abbreviations

Name	Description

1.3. Reference

[1]
[2] …

2. LCD 注册

主要说明了 LCD 驱动文件添加、命名规则、编译选项添加、注册信息添加。

2.1. 添加 LCD 驱动文件

在 media 目录下添加提供的 driver 目录文件(内含 osal、lcm_module 目录)，在 lcm_module 下直接执行 make 命令将会生成 lcm_module.ko。

在..\media\driver\lcm_module 路径下添加 lcm 模组驱动文件。

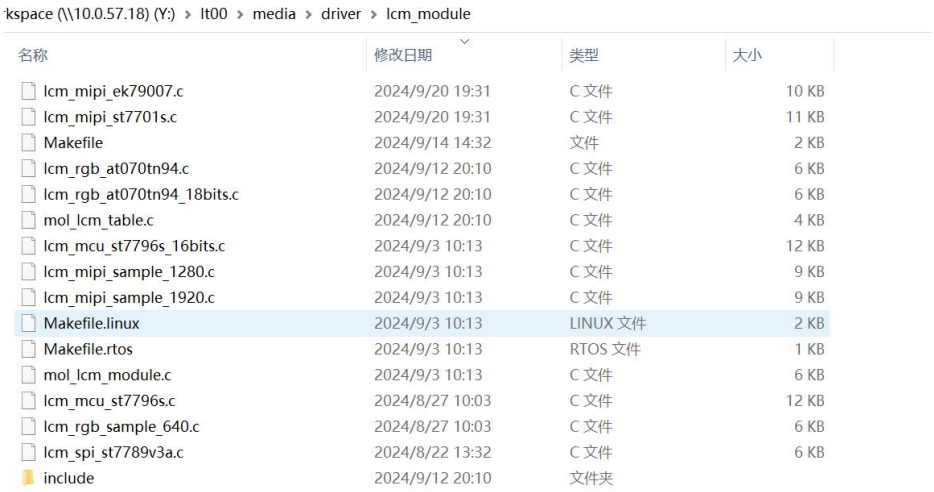


figure: 2- 1 添加 lcm 模组驱动文件

2.2. lcd 模组驱动文命名规则

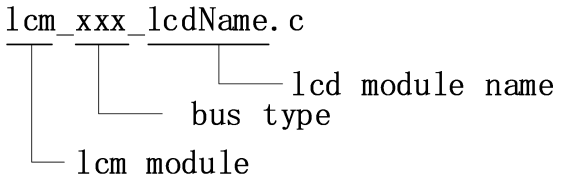


figure: 2- 2 lcd 模组驱动文命名规则

lcm: lcm 模块名 (不改变固定为 lcm)
xxx:lcd 总线类型（mipi/rgb/mcu/…）
lcdName:lcd 模组名（如:ek79007）

2.3. 添加 lcd 模组编译选项

在..\media\driver\lcm_module 路径下 Make 文件添加 lcd 模组驱动编译选项。

Makefile

```
31
32  lcm-objs += module/mol_lcm_table.o \
33              module/lcm_mipi_ek79007.o \
34              module/lcm_rgb_nv3047egib.o \
35              module/lcm_mipi_jd9365da_h3.o \
36              module/lcm_mipi_jd9365da_h3_640x480.o
37
38
```

figure: 2- 3 添加 lcd 模组驱动编译选项

2.4. lcd 模组注册

lcd 模组信息注册到 lcd 系统中,vou 或应用通过 LMC MIP 接口函数进行操作。
在..\media\driver\lcm_module 文件中注册 lcd 模组。其中 main_tab 绑定 DHD0 主屏，sub_tab 绑定 DHD1 副屏。

mol_lcm_table.c (vou\lcm\module)

```
/*mipi lcm*/
extern lcm_cfg_t lcm_mipi_ek79007_info;
extern lcm_cfg_t _lcm_mipi_jd9365da_info;
extern lcm_cfg_t lcm_mipi_jd9165da_info;
extern lcm_cfg_t _lcm_mipi_jd9365da_mini_info;

/*rgb lcm*/
extern lcm_cfg_t lcm_rgb_nv3047egib_info;

lcm_cfg_t *lcm_main_tab[] = {
    //&lcm_mipi_jd9165da_info,
    &lcm_mipi_ek79007_info,
    //&_lcm_mipi_jd9365da_info,
    &_lcm_mipi_jd9365da_mini_info,
    &lcm_rgb_nv3047egib_info,
    0,
};

lcm_cfg_t *lcm_sub_tab[] = {
    //&lcm_mipi_jd9165da_info,
    &lcm_mipi_ek79007_info,
    //&_lcm_mipi_jd9365da_info,
    &_lcm_mipi_jd9365da_mini_info,
    &lcm_rgb_nv3047egib_info,
    0,
};
```

figure: 2- 4 注册 lcd 模组

2.5. lcd 模组注册信息

lcd 模组注册信息包含以下信息：

- lcd 基本信息
  lcd name、lcd resolution width/height、总线类型、刷新帧率、刷新方向
- lcd 显示同步信息
  vou 输出同步信息
- lcd 总线信息
  mipi/mcu(i80)/rgb(sync)总线信息
- lcd 控制函数信息
  lcd 提供应用控制函数，应用可以定制控制流程

```
const lcm_cfg_t _lcm_mipi_ek79007_info =
{
    .name = "ek79007_dsi_wsvga_vdo",
    .width = LCM_EK79007_WIDTH,
    .height = LCM_EK79007_HEIGHT,
    .type = LCM_TYPE_MIPI, /*mcu, rgb, mipi*/
    .fps = 60,
    .direction = LCM_DIRECT_NORMAL,
    .dp_sync = &_lcm_mipi_ek79007_dp_info,
    .infor = {
        .mipi = &_lcm_mipi_ek79007_mipi_info,
    },
    .fun = &lcm_mipi_ek79007_ctrl,
};
```

```
typedef struct {
    char name[256];
    uint32_t width;
    uint32_t height;
    lcm_type_e type; /*mcu, rgb,
    uint32_t fps;
    lcm_direction direction;
    display_info *dp_sync;
    union {
        lcm_mcu_info *mcu;
        lcm_rgb_info *rgb;
        lcm_mipi_info *mipi;
    }infor;
    lcm_module_fun *fun;
}lcm_cfg_t;
```

lcd基本信息

lcd 显示同步信息

lcd总线信息

lcd控制函数信息

figure: 2- 5 lcd 模组注册信息

2.6. lcd 基本信息

.name: lcd name
.width: lcd resolution width
.height: lcd resolution height
.type: lcd 总线类型 mipi/mcu(i80)/rgb(sync)
.fps: lcd 刷新帧率
.direction: lcd 刷新方向

2.7. lcd 显示同步信息

VOU 输出图像数按同步方式输出。

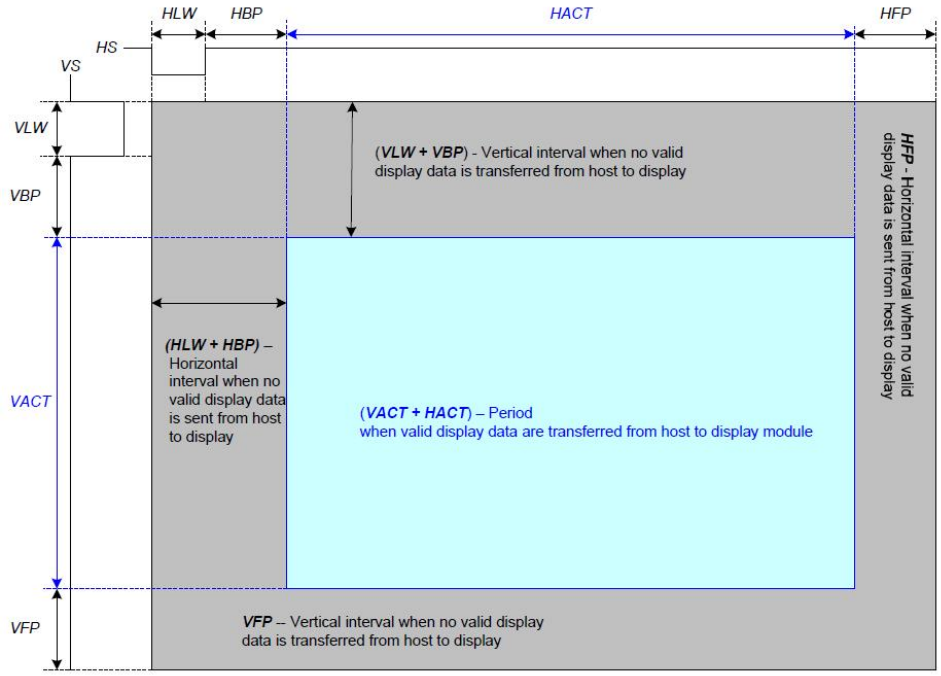


figure: 2- 6 同步信号

HLW: Horizon sync low width
HBP: Horizon back porch
HACTIVE： Horizon display period
HFP: Horizon front porch

VLW: Vertical sync low width
VBP: Vertical back porch
VAVTIVE: Vertical display period
VFP: Vertical front porch

```
typedef struct {
    int32_t hor_sync_active;
    int32_t hor_back_porch;
    int32_t hor_active_pixel;
    int32_t hor_front_porch;

    int32_t ver_sync_active;
    int32_t ver_back_porch;
    int32_t ver_active_line;
    int32_t ver_front_porch;
}display_info;
```

```
static display_info _lcm_mipi_ek79007_dp_info =
{
    .hor_sync_active = 40,        // horizon sync
    .hor_back_porch = 120,        // horizon back porch
    .hor_active_pixel = LCM_EK79007_WIDTH,   // horizon display period
    .hor_front_porch = 160,       // horizon front porch

    .ver_sync_active = 5,         // vertical sync
    .ver_back_porch = 18,         // vertical back porch
    .ver_active_line = LCM_EK79007_HEIGHT,   // vertical display period
    .ver_front_porch = 12,        // vertical front porch
};
```

figure: 2- 7 同步信息配置参数

同步信息配置 o 参数可以参考 LCD 模组文档或 LCD driver IC datasheet. 也可以请 lcd 供应商 FAE 提供。

Parameter	Symbols	Condition	Min.	Typ.	Max.	Units
Frame Rate	FR		54		66	fps
Horizontal Low Pulse width	HLW		1		-	DOTCLK
Horizontal Back Porch	HBP		2		126	DOTCLK
Horizontal Address	HACT			480		DOTCLK
Horizontal Front Porch	HFP		2		-	DOTCLK
Vertical Low Pulse width	VLW		1		126	Line
Vertical Back Porch	VBP		1		126	Line
Vertical Address	VACT				864	Line
Vertical Front Porch	VFP		1		255	Line
Data Clock	DCLK		16.6		41.7	MHz

figure: 2- 8 同步信息配置表

2.8. lcd ioctrl 函数注册

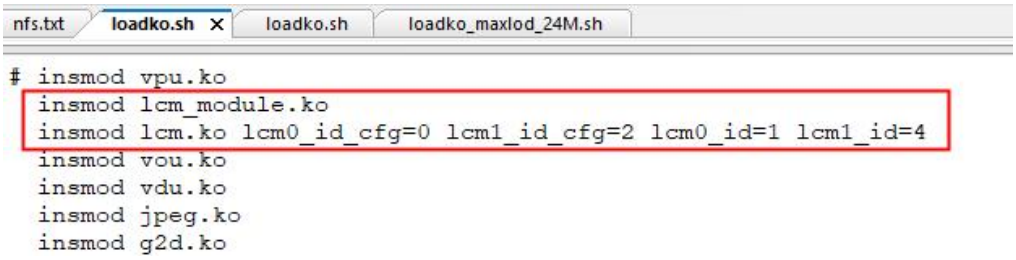
```
lcm_module_fun lcm_mipi_ek79007_ctrl =
{
    .poweron = _lcm_mipi_ek79007_PowerOn,
    .poweroff = _lcm_mipi_ek79007_PowerOff,
    .identify = _lcm_mipi_ek79007_identify,
    .init = _lcm_mipi_ek79007_init,
    .suspend = _lcm_mipi_ek79007_suspend,
    .resume = _lcm_mipi_ek79007_resume,
    .deinit = _lcm_mipi_ek79007_deinit,
};
```

figure: 2- 9 lcd ioctrl 函数注册

.poweron：lcd 上电函数注册
.poweroff：lcd 下电函数注册
.identify：lcd 侦测认证函数注册
.init：lcd 初始化函数注册
.suspend：lcd 进入睡眠函数注册
.resume：lcd 退出睡眠函数注册
.deinit：lcd 反初始化函数注册

2.9. lcm ko 加载

- QM10XD LCM 系统需要加载两个 KO (lcm_module.ko 和 lcm.ko)
- lcm_module.ko 包含所有 lcm module 配置参数，该 ko 是以源码开放给客户，由客户自行调试 lcm
- lcm.ko 包含 MIPI、RGB、MCU 等驱动各控制流程



2.9.1. lcm ko 加载参数

lcm0_id_cfg：main lcm 认证流程模式
lcm1_id_cfg：sub lcm 认证流程模式

- 0：正常流程模式，认证时先获 flash 中保存的 Id 信息进行认证，如果认证不到，进行全 table 中的 lcm 进行顺序认证
- 1：指定 ID 模式，认证时先将指定 lcm id 写入 flash。接下来认证过程同正常流程模式
- 2：关闭认证，关闭 main_lcm/sub_lcm 认证

lcm0_id：sub lcm 认证指定 ID
lcm1_id：sub lcm 认证指定 ID

2.9.2. lcm ko 指定 ID 模式

- **lcmx_id**

```
lcm_cfg_t *lcm_main_tab[] = {
    /*mipi lcm*/
    &s_lcm_mipi_ek79007_info,  <-- 0
    &s_lcm_mipi_st7701s_info,  <-- 1

    /*rgb lcm*/
#if 1
    /*mcu lcm*/
    &s_lcm_mcu_st7796s_info,   <-- 2
    &s_lcm_mcu16_st7796s_info, <-- 3

    /* ------ non id read ------*/
    /*rgb lcm*/
    &s_lcm_rgb_at070tn94_info,  <-- 4
    &s_lcm_rgb18_at070tn94_info, <-- 5

    // only for maxloading
    &s_lcm_mipi_sample_1920_info, <-- 6
    &s_lcm_mipi_sample_1280_info, <-- 7
#endif
    /*end lcm*/
    0,
};
```

```
lcm_cfg_t *lcm_main_tab[] = {
    /*mipi lcm*/
    &s_lcm_mipi_ek79007_info,
    &s_lcm_mipi_st7701s_info,

    /*rgb lcm*/
#if 1
    /*mcu lcm*/
    &s_lcm_mcu_st7796s_info,
    &s_lcm_mcu16_st7796s_info,

    /* ------ non id read ------*/
    /*rgb lcm*/
    &s_lcm_rgb_at070tn94_info,
    &s_lcm_rgb18_at070tn94_info,

    // only for maxloading
    &s_lcm_mipi_sample_1920_info,
    &s_lcm_mipi_sample_1280_info,
#endif
    /*end lcm*/
    0,
};
```

lcm_main_tab 中的 lcm id 由 lcm0_id_cfg 控制指定
lcm0_id：lcm_main_tab 中的 lcm id, 0/1/2/3

```
lcm_cfg_t *lcm_sub_tab[] = {
#if 1
    /*mipi lcm*/
    &s_lcm_mipi_st7701s_info,

    /*rgb lcm*/

    /*mcu lcm*/
    &s_lcm_mcu_st7796s_info,
    &s_lcm_mcu16_st7796s_info,

    /* ------ non id read ------*/
    /*rgb lcm*/
    &s_lcm_rgb_at070tn94_info,
    &s_lcm_rgb18_at070tn94_info,

    // only for maxloading
    &s_lcm_rgb_sample_640_info,
    &s_lcm_mipi_sample_1280_info,
    /*end lcm*/
#endif
    0,
};
```

lcm_sub_tab 中的 lcm id 由 lcm_id_cfg 控制指定
lcm1_id：lcm_sub_tab 中的 lcm id, 0/1/2/3

3. lcd 总线信息

LCD 总线有 MIPI/MCU/RGB(同步)模式。

4. MIPI LCD DEBUG

pixel_clk = total pixel = H-total * V-total * fps
bitclk= Total pixel * 24 / lane number
byteclk = bitclk/8
pclk(dsi pixel clock) = (byteclk * lane number)/bpp(byte) = total pixel
UI = 1/bitclk

4.1. LCD display flow

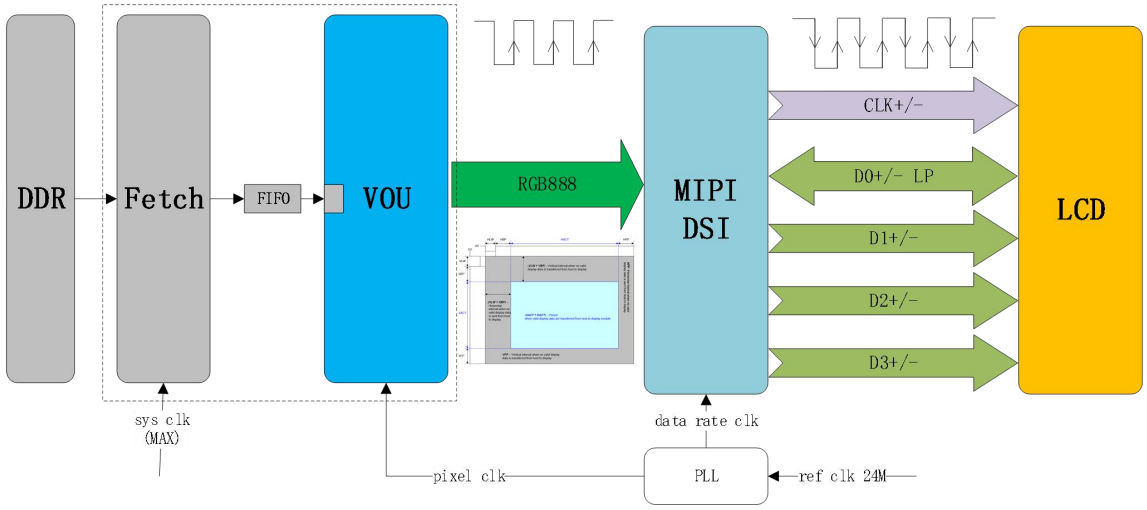


figure: 2- 10 mipi lcd display flow

4.2. DSI pin mux config

- MIPI0 1C4D

```
{0x102000BC, 0x00}, //DSI_DP3
{0x102000C0, 0x00}, //DSI_DN3
{0x102000C4, 0x00}, //DSI_DP2
{0x102000C8, 0x00}, //DSI_DN2
{0x102000CC, 0x00}, //DSI_CLKP1
{0x102000D0, 0x00}, //DSI_CLKN1
{0x102000D4, 0x00}, //DSI_CLKN0
{0x102000D8, 0x00}, //DSI_CLKP0
{0x102000DC, 0x00}, //DSI_DN1
{0x102000E0, 0x00}, //DSI_DP1
```

```
{0x102000E4, 0x00}, //DSI_DN0
{0x102000E8, 0x00}, //DSI_DP0
```

- MIPI0 1C2D

```
{0x102000D4, 0x00}, //DSI_CLKN0
{0x102000D8, 0x00}, //DSI_CLKP0
{0x102000DC, 0x00}, //DSI_DN1
{0x102000E0, 0x00}, //DSI_DP1
{0x102000E4, 0x00}, //DSI_DN0
{0x102000E8, 0x00}, //DSI_DP0
```

- MIPI1 1C2D

```
{0x102000BC, 0x00}, //DSI_DP3
{0x102000C0, 0x00}, //DSI_DN3
{0x102000C4, 0x00}, //DSI_DP2
{0x102000C8, 0x00}, //DSI_DN2
{0x102000CC, 0x00}, //DSI_CLKP1
{0x102000D0, 0x00}, //DSI_CLKN1
```

4.3. lcd mipi 总线信息

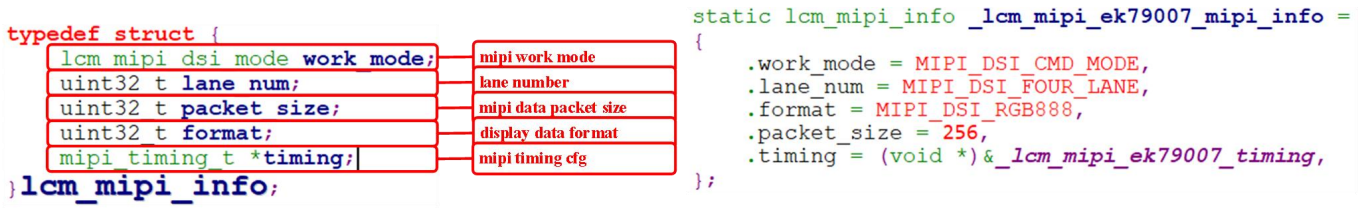


figure: 2- 11 lcd mipi 信息

work_mode: command_mode or video_mode, 默认设定 cmd mode
lane_num: mipi data lane number
packet_size: mipi data packet size
format: mipi display data format
timing: the pointer of mipi timing

4.4. lcd mipi 总线 timing 配置

lcd mipi 总线 timing 配置有三部分，clk lane、data lane、lp read, 默认驱动程序根据协议自动计算 timing 配置参数，如果 lcd 供应商 FAE 提供指定配置参数可以通过 manual 方式设定。

```
typedef struct {
    dsi_phy_clane_timing_t clane;
    dsi_phy_dlane_timing_t dlane;
    dsi_phy_lane0_read_t read;
    dsi_phy_pll_div_t pll;
}dsi_timing_t;
```

clk lane cfg param
dara lane cfg param
lp read cfg param
pll clk cfg param

figure: 2- 12 mipi 总线时序配置

lcd mipi clk timing 配置

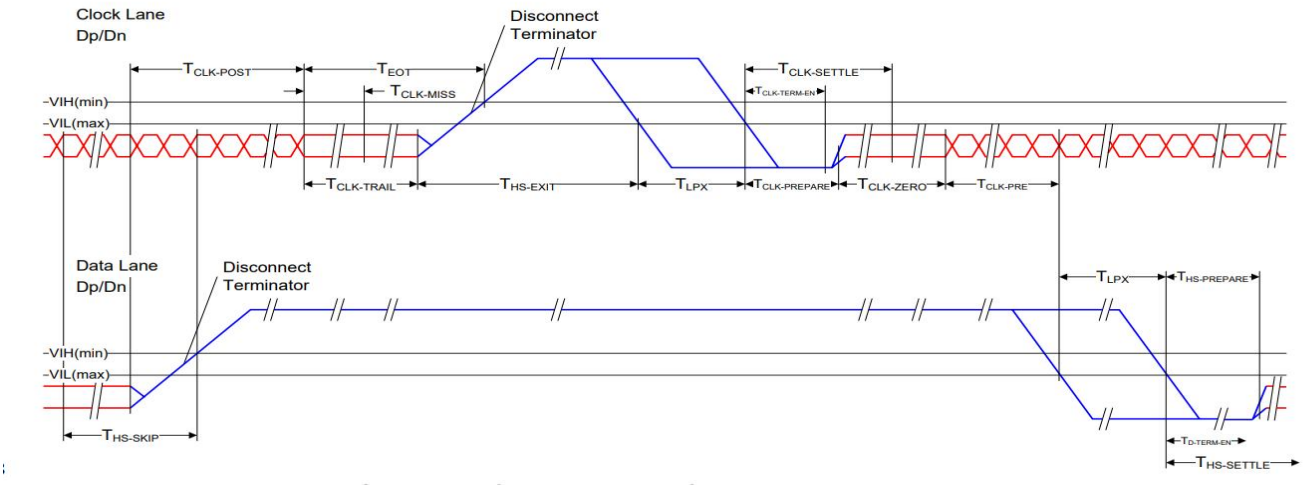


figure: 2- 13 mipi clk 协议

```
typedef struct {
    uint32_t mode;
    uint32_t lp11;
    uint32_t lp01;
    uint32_t zero;
    uint32_t prepare;
    uint32_t pre;
    uint32_t trail;
    uint32_t exit;
    uint32_t post;
}dsi_phy_clane_timing_t;
```

```
static dsi_timing_t _lcm_mipi_ek79007_timing =
{
    /*clk lane timing cfg*/
    .clane.mode = 0,
    .clane.lp11 = 0,
    .clane.lp01 = 0,
    .clane.zero = 0,
    .clane.prepare = 0,
    .clane.trail = 0,
    .clane.exit = 0,
};
```

figure: 2- 14 mipi clk timing cfg

mode: 0:auto calc mode 1:manul cfg mode

lcd mipi data timing 配置

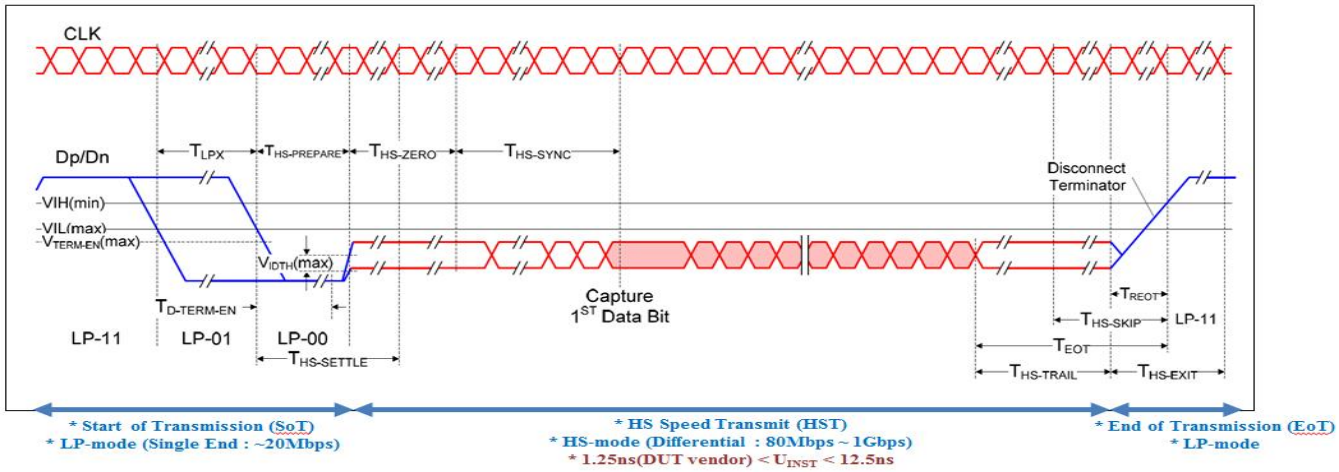


figure: 2- 15 mipi data 协议

```
typedef struct {
    uint32_t mode;
    uint32_t lp11;
    uint32_t lp01;
    uint32_t zero;
    uint32_t prepare;
    uint32_t trail;
    uint32_t exit;
}dsi_phy_dlane_timing_t;
```

```
static dsi_timing_t _lcm_mipi_ek79007_timing =
{
    /*data lane timing cfg*/
    .dlane.mode = 0,
    .dlane.lp11 = 0,
    .dlane.lp01 = 0,
    .dlane.zero = 0,
    .dlane.prepare = 0,
    .dlane.trail = 0,
    .dlane.exit = 0,
};
```

figure: 2- 16 mipi data timing cfg

mode: 0:auto calc mode 1:manul cfg mode

lcd mipi lp read timing 配置

```
typedef struct {
    uint32_t mode;
    uint32_t taget;
    uint32_t tasure;
    uint32_t tago;
}dsi_phy_lane0_read_t;
```

```
static dsi_timing_t _lcm_mipi_ek79007_timing =
{
    /*data lane0 read timing cfg*/
    .read.mode = 0,
    .read.tago = 0,
    .read.tasure = 0,
    .read.taget = 0,
};
```

figure: 2- 17 mipi lp read cfg

mode: 0:auto calc mode 1:manul cfg mode

lcd mipi pll clk 配置

```
typedef struct {
    uint32_t mode;
    uint32_t rate;
    uint32_t ref_clk_div;
    uint32_t pll_div;
    uint32_t pixel_clk_div;
}dsi_phy_pll_div_t;
```

```
static dsi_timing_t _lcm_mipi_ek79007_timing =
{
    /*pll clk div cfg*/
    .pll.mode = 0,
    .pll.rate = 0,
    .pll.ref_clk_div = 0,
    .pll.pll_div = 0,
    .pll.pixel_clk_div = 0,
};
```

figure: 2- 18 mipi pll clk cfg

mode: 0:auto calc mode 1:manul cfg mode

lcd mipi 初始化配置参数

```
static mipi_lcm_setting_tab s_lcm_mipi_ek79007_init_setting[] = {
    {DSI_DI_DCS_WRITE_1_PARAM, 2, {0x80, 0x47}},
    {DSI_DI_DCS_WRITE_1_PARAM, 2, {0x81, 0x40}},
    {DSI_DI_DCS_WRITE_1_PARAM, 2, {0x82, 0x04}},
    {DSI_DI_DCS_WRITE_1_PARAM, 2, {0x83, 0x77}},
    {DSI_DI_DCS_WRITE_1_PARAM, 2, {0x84, 0x0f}},
    {DSI_DI_DCS_WRITE_1_PARAM, 2, {0x85, 0x70}},
    {DSI_DI_DCS_WRITE_1_PARAM, 2, {0x86, 0x70}},
    {DSI_DI_DELAY,             20,            },
    {DSI_DI_END_OF_TABLE, 0x00, {0}}
};
```

1）mipi lcd 配置参数。

param1：写模式：generic 0/1/2/long param, dcs 0/1/2/long param 8 种模式

param2：配置参数数量（1 中大括号中第二个大括号参数数量）

param3-n：配置参数，generic/dcs long 模式 param3-4 是参数数量

2）delay 命令

param1：delay 标志

param2：delay 时间，单位 ms

param3-n：无效

3）mipi lcd 配置参数数组结束标志

param1：结束标志

param2：无效

param3-n：无效

5. RGB LCD DEBUG

5.1. RGB(TTL) pin mux config

```
{0x10200064, 0x00}, //LCD_DE
{0x10200068, 0x00}, //LCD_HSYNC
{0x1020006C, 0x00}, //LCD_VSYNC
{0x10200070, 0x00}, //LCD_CLK
{0x10200074, 0x00}, //LCD_D0
{0x10200078, 0x00}, //LCD_D1
{0x1020007C, 0x00}, //LCD_D2
{0x10200080, 0x00}, //LCD_D4
{0x10200084, 0x00}, //LCD_D3
{0x10200088, 0x00}, //LCD_D5
{0x1020008C, 0x00}, //LCD_D7
{0x10200090, 0x00}, //LCD_D6
{0x10200094, 0x00}, //LCD_D9
{0x10200098, 0x00}, //LCD_D8
{0x1020009C, 0x00}, //LCD_D10
{0x102000A0, 0x00}, //LCD_D11
{0x102000A4, 0x00}, //LCD_D12
{0x102000A8, 0x00}, //LCD_D13
{0x102000AC, 0x00}, //LCD_D14
{0x102000B0, 0x00}, //LCD_D15
{0x102000B4, 0x00}, //LCD_D16
{0x102000B8, 0x00}, //LCD_D17
/* 复用 DSI1 PIN */
{0x102000BC, 0x02}, //LCD_D21
{0x102000C0, 0x02}, //LCD_D20
{0x102000C4, 0x02}, //LCD_D23
{0x102000C8, 0x02}, //LCD_D22
{0x102000CC, 0x02}, //LCD_D19
{0x102000D0, 0x02}, //LCD_D18
```

如果使用的是 dhd1 的 lcm 接口，需要配置 0x24000E0C[6:5]=0x3
如果使用的是 dhd0 的 lcm 接口，需要配置 0x24000E0C[6:4]=0x1

0x26000040[2:0] dhd0 1: dsi mode 4: lcm mode(mcu/rgb)

0x26000040[6:4] dhd1 1: dsi mode 4: lcm mode(mcu/rgb)

Dhd0 base addr:0x26001000
Dhd0 base addr:0x26002000

0x260010B4[7] dhd0 0: rgb mode 1: mcu mode
0x260020B4[7] dhd1 0: rgb mode 1: mcu mode

0x26000040:display format
0x2600108c:
0x26001074:front_proch
0x26001078:sync_pulse
0x2600107c:back_proch
0x260010b4:display_cfg
0x2800006c:hdmi_clk
0x260010bc:lcm_cgf
0x260010e0:lcm_stage

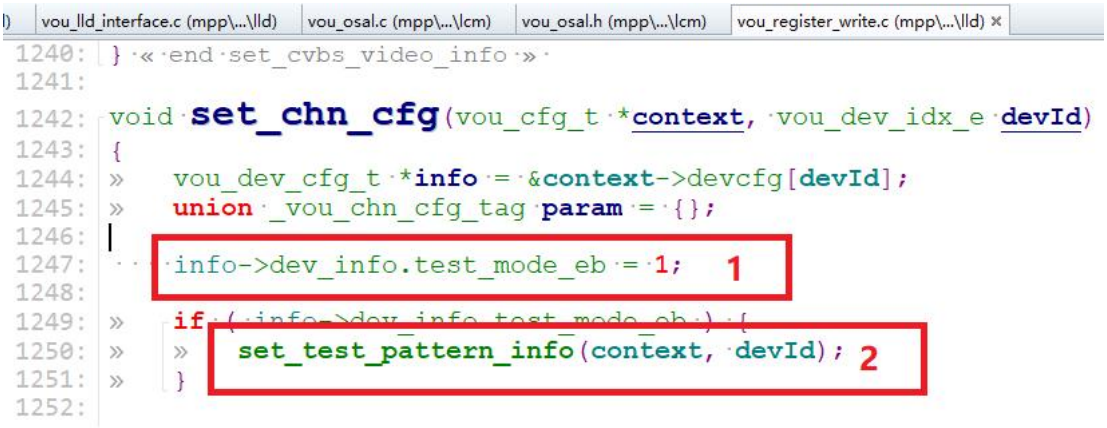
6. MCU LCD DEBUG

7. Test pattern

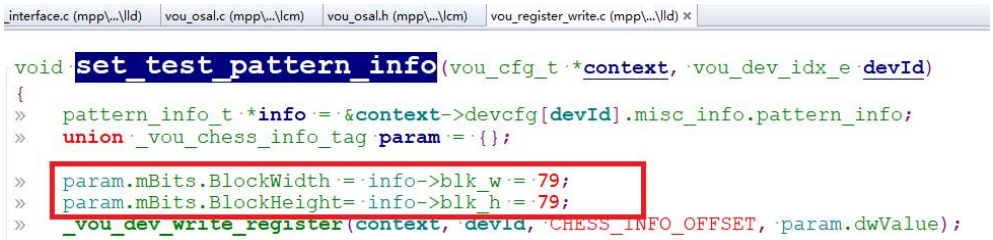
在调试过程中如果不需要输入源，仅验证 VOU 到 DSI 或 DSI 直接输出到 LCD。

7.1. Vou test pattern

1.) 进入 vou/lld/vou_interface.c 文件
2.) 进入_vou_dev_kickoff 函数
3.) 进入 set_chn_cfg 函数



1.)将 set_chn_cfg 函数 test_mode_eb 设置为 1
2.)在 set_test_pattern_info 函数中设置 test_pattern 大小



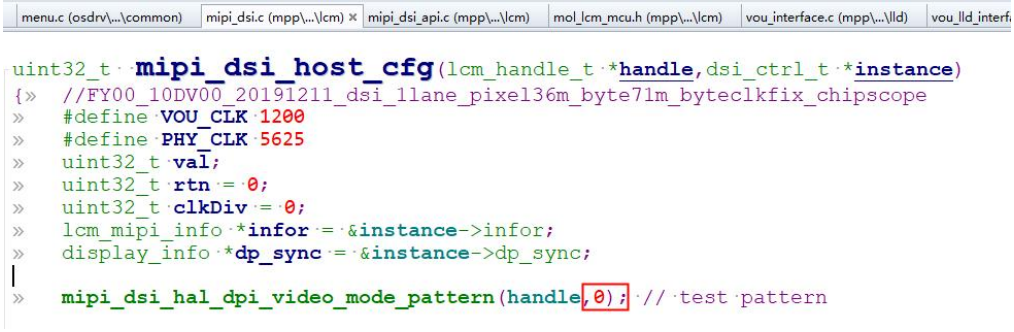
将 block_w/block_h 设置为合适尺寸，实际设定值为 宽-1/高-1

dhd0 colorbar 彩条模式
devmem 0x26001000 32 0x103
devmem 0x26001008 32 0xffffffff

dhd1 colorbar 彩条模式
devmem 0x26002000 32 0x103
devmem 0x26002008 32 0xffffffff

7.2. DSI test pattern

将 mpip_dsi_host_cfg 函数中将 mipi_dsi_dpi_vide_mode_pattern 函数中的参数设置为 1.



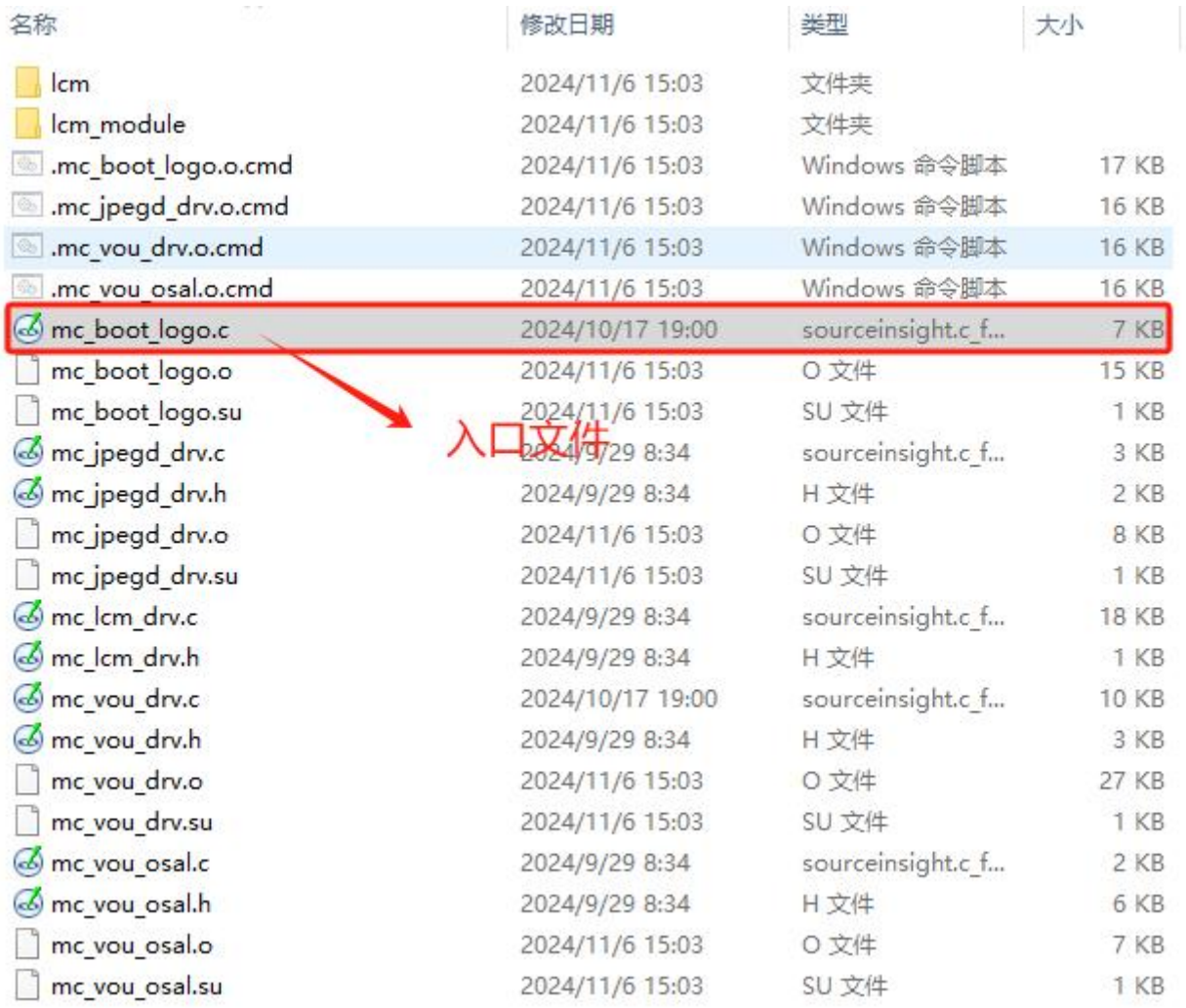
host0 colorbar
devmem 0x25c00038 32 0x10202

host1 colorbar
devmem 0x25700038 32 0x10202

8. Uboot LCD DEBUG

产品开机时，kernal 开启还需要一段时间。可以在 uboot 先显示一张开机 logo，给用户更好的体验。

8.1. 开机 logo 实现文件



bsp\uboot\uboot\common\mc

目前 showlogo 代码与 kernel 代码基本保持一致，添加新屏仍需和 kernel 一致，放入 lcm_module 下的 table 中。选择屏也如 kernel 下一致，支持 lcm0_id_cfg、lcm0_id、lcm1_id_cfg、lcm1_id。

注：uboot 下不支持动态计算屏 pll、timing 等参数如：

```
typedef struct {
    uint32_t mode;
    uint32_t lp11;
    uint32_t lp01;
    uint32_t zero;
    uint32_t prepare;
    uint32_t pre;
    uint32_t trail;
    uint32_t exit;
    uint32_t post;
}dsi_phy_clane_timing_t;
```

```
static dsi_timing_t _lcm_mipi_ek79007_timing =
{
    /*clk lane timing cfg*/
    .clane.mode = 0,
    .clane.lp11 = 0,
    .clane.lp01 = 0,
    .clane.zero = 0,
    .clane.prepare = 0,
    .clane.trail = 0,
    .clane.exit = 0,
};
```

则需要把相应 mode 置为 1，需配置的参数从 kernel 读出填入即可。

Kernel 下读取配置参数方式，cat /proc/driver/lcm，将会列举出 phy0、phy1 的配置参数：

```
/nfs/lt00/ko # cat /proc/driver/lcm
>>>-----------------------dsi phy info-----------------------<<<

----------dsi txPhy0 param---
dsiPhyContext->phyworkMode[0]  : 1

----------pll-----------------
frameInfo->fromat          : 5
frameInfo->fps             : 60
frameInfo->lane_num        : 1
dsiPhyContext->laneclk[0]: 169660800
core->pll_freq_doubler_en : 0
core->pll_div_s            : 1
core->pll_n                : 0
core->pll_kint             : 4643933
core->pll_nint             : 56
core->pixelclk_div         : 0
core->pll_pdiv             : 2
dsiPhyContext->dhd_div     : 2

--------clane-----------------
clane->rst2enlptx          : 500
clane->inittime            : 6
clane->clkprepare          : 2
clane->clkzero             : 13
clane->clkpre              : 3
clane->clkpost             : 10
clane->clktrail            : 3
clane->hsexit              : 6
```

```
--------clane-----------------
clane->rst2enlptx          : 500
clane->inittime            : 6
clane->clkprepare          : 2
clane->clkzero             : 13
clane->clkpre              : 3
clane->clkpost             : 10
clane->clktrail            : 3
clane->hsexit              : 6
clane->pulldwn_mode        : 0
clane->wakeup              : 256

--------dlane-----------------
dlane->rst2enlptx[0]          : 108
dlane->inittime[0]            : 6
dlane->hsprepare[0]           : 3
dlane->hszero[0]              : 4
dlane->hstrail[0]             : 4
dlane->hsexit[0]              : 6
dlane->wakeup[0]              : 54540
dlane->tago[0]                : 27
dlane->tasure[0]              : 10
dlane->taget[0]               : 34
dlane->rst2enlptx[1]          : 108
dlane->inittime[1]            : 6
dlane->hsprepare[1]           : 3
dlane->hszero[1]              : 4
dlane->hstrail[1]             : 4
dlane->hsexit[1]              : 6
dlane->wakeup[1]              : 54540
```

```
--------dlane-----------------
dlane->rst2enlptx[0]          : 108
dlane->inittime[0]            : 6
dlane->hsprepare[0]           : 3
dlane->hszero[0]              : 4
dlane->hstrail[0]             : 4
dlane->hsexit[0]              : 6
dlane->wakeup[0]              : 54540
dlane->tago[0]                : 27
dlane->tasure[0]              : 10
dlane->taget[0]               : 34
dlane->rst2enlptx[1]          : 108
dlane->inittime[1]            : 6
dlane->hsprepare[1]           : 3
dlane->hszero[1]              : 4
dlane->hstrail[1]             : 4
dlane->hsexit[1]              : 6
dlane->wakeup[1]              : 54540
dlane->tago[1]                : 27
dlane->tasure[1]              : 10
dlane->taget[1]               : 34
com->lpx lp01                   : 6
```

读出来的参数直接写入即可。

```
    .read.tago = 0x1b,
    .read.tasure = 0x0b,
    .read.taget = 0x22,
    /*clk cfg*/
    .clk = {
        .vou.mode = 1,
        .vou.sel = 0x3,
        .vou.div = 0,

        .mif.mode = 1,
        .mif.pll_doubler = 0,
        .mif.pll_div_s = 1,
        .mif.pll_n = 0,
        .mif.pll_kint = 5637144,
        .mif.pll_nint = 42,
        .mif.pixelclk_div = 12,
        .mif.pll_pdiv = 2,
        .mif.dhd_div = 1,
    }
};
```

Vou.mode 该参数无论 uboot 或者 kernel 下常为 1，且配置参数全部保持一致。

8.2. 配置

pin max gpio

dst pwm

9. 调试工具使用

0x25D00000~0x25D00080 // dsi phy
0x25700000~0x25700100 // dsi host1
0x25C00000~0x25C00100 // dsi host0

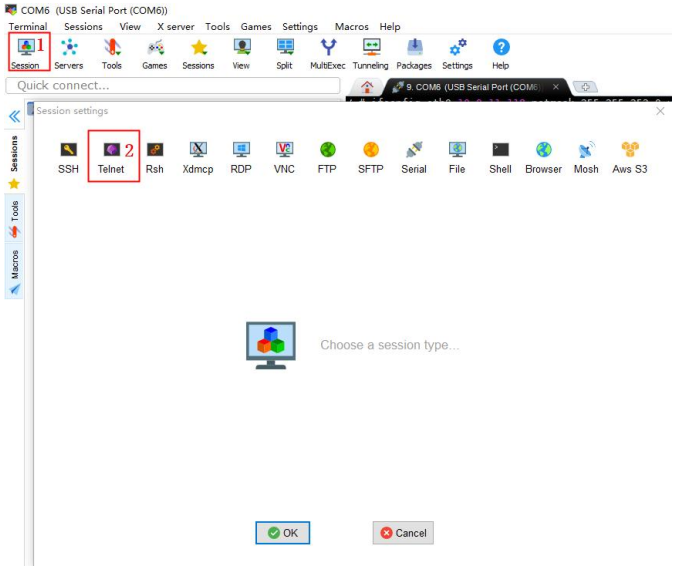
0x26000000~0x260000FF // vou global
0x26001000~0x26001DFF // vou dhd0
0x26002000~0x26002DFF // vou dhd1

Uboot
read register
md.l 0x28200100
write register
mw.l 0x34500000 b0 1

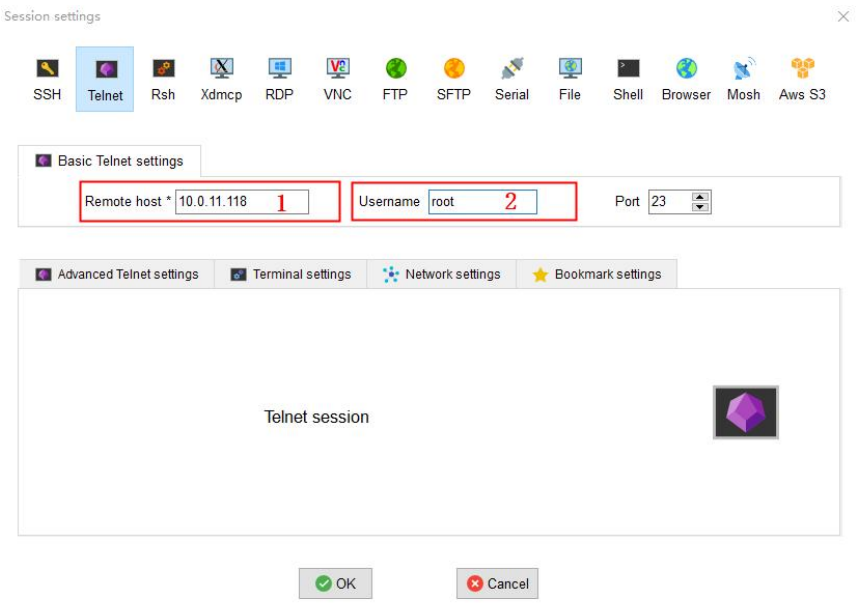
kernel
read register
devmem 0x1b070004
write register
devmem 0x1b070004 32 0x80

Telnetd 使用
系统进入 linux kernel laod ko 应用程序运行起来后（串口会背应用程序占用），这时使用 telnetd 进行寄存器查看

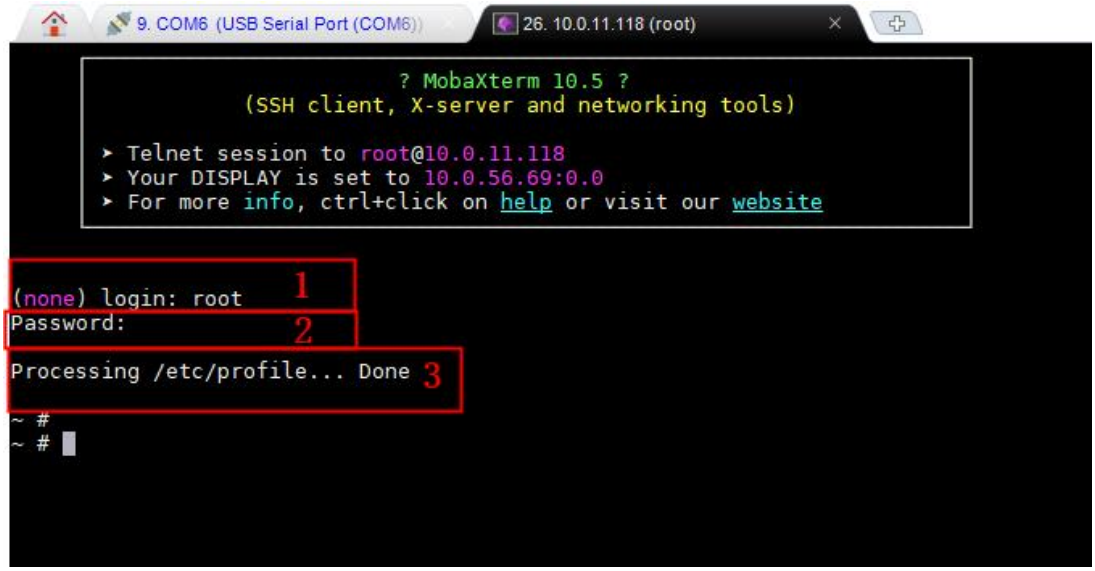
双击打开 MobaXterm 工具进入 telnetd 配置



1.）打开 Sessiont 选项。
2.）选择 Telnetd，进行 Telnetd 配置。



1.）填写板子 IP 地址（nfs 时板子的 IP 10.0.11.118）。
2.）填写 root。



1.）键入 root，点击回车进入 Password。
2.）点击回车，进行 mount。
3.）返回 Processing/etc/profile… Done，mount 成功。

如果 telnetd 不能成功连接：
1.）在 kernal 下 ping 10.0.56.137(ubuntu IP addr)
2.）出现 lonin:

telnetd 大量读串口显示
echo -r 0x34000000 -l 1000 > /sys/devices/platform/sysreg/lookat

telnetd 写寄存器串口显示
echo -s 0x34000000 -l 1 0xff > /sys/devices/platform/sysreg/lookat

echo -r 0x34300000 -l 64 > /sys/devices/platform/sysreg/lookat
echo -r 0x34500000 -l 31 > /sys/devices/platform/sysreg/lookat
echo -r 0x34100000 -l 64 > /sys/devices/platform/sysreg/lookat
echo -r 0x34101000 -l 64 > /sys/devices/platform/sysreg/lookat

10. 客户调屏支持流程

客户调屏支持流程				
排查步骤	**调试/排查事项**	**各方主导人**	**调试结果/状态**	**备注/重点排查点**
1	联系屏厂获取：驱动 IC/模组 SPEC、参考代码	客户（屏厂支持）	资料	
2	核对原理图，确认 LCD 引脚定义、上电、复位、背光	客户	基本硬测报告	确认管脚连接正确、供电电压和信号电平匹配，避免引脚复用冲突
3	参考 SDK 中同接口驱动/LCD 移植文档，集成屏厂参数和平台参数配置，完成驱动移植	客户（主控方提供 SDK 支持）	屏驱动	对比参考驱动时序
4	上板实测，确认屏参配置成功，如读 ID/量信号，量测供电/复位/时钟/数据线波形，核对软件参数	客户	信号波形、与 SPEC 匹配	供电查电源电路，信号波形检查，信号/参数对照模组 SPEC
5	开启 BIST 或自测模式，验证屏体自身是否正常	客户（屏厂支持）	自测显示	建议 2 片屏调试，避免单屏问题；
6	问题仍未解决，Redmine 提单，同步提供：排查记录、屏厂资料、驱动代码，拉通三方沟通，确定接下来调试方案	客户+屏厂+主控三方（客户牵头三方沟通）	redmine 提单和三方会议纪要	客户三方沟通前：提供排查记录/波形图，明确待确认疑点、驱动代码标注修改点；
7	如远程支持无法解决，与 QM 协商寄 LCD 调试套件或来现场调试	客户+主控方（QM 对接）	协商支持方式	提供：资料需一次性补齐，2 套待调试屏硬件（客户板、屏连接、供电、调试线、板子使用说明等都齐全），配套开发

				调试环境（调屏 patch）
8	QM 根据收到板卡和工单排期情况，安排调试/交付	主控方（QM 对接）	屏点亮	交付亮屏 patch