QM10XD LCD USER GUIDE

发布日期 [发布日期]

修订历史

Revision	Date	Author	Description

目录

1. 简介

1.1. 概述

本文档提供了 LCD 调试的概要规范。LCD 调试工程师可以参考本文档进行详细的设计和实现。

1.2. LCD 接口功能描述

显示接口支持 MIPI、LCM(RGB/MCU)。

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。

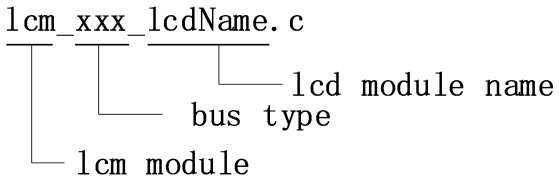
2.1. lcd 驱动目录信息

LCD 驱动文件目录结构如下：

```
media/driver/
├── lcm_module                                  //LCD 模组驱动
│   ├── include
│   ├── lcm_mcu_st7796s_16bits.c                //LCD MCU 屏文件
│   ├── lcm_mcu_st7796s.c
│   ├── lcm_mipi_ek79007.c                      //LCD MIPI 屏文件
│   ├── lcm_mipi_qfh24004.c
│   ├── lcm_mipi_sample_1280.c
│   ├── lcm_mipi_sample_1920.c
│   ├── lcm_mipi_st7701s.c
│   ├── lcm_rgb_at070tn94_18bits.c              //LCD RGB 屏文件
│   ├── lcm_rgb_at070tn94.c
│   ├── lcm_rgb_sample_640.c
│   ├── lcm_spi_st7789v3a.c
│   ├── Makefile
│   ├── mol_lcm_module.c
│   └── mol_lcm_table.c                         //LCD 屏模组注册表文件
├── osal
```

```
|       ├── Module.symvers                                  //osal 符号表文件
```

2.2. lcd 模组驱动文命名规则



图表 2-1 lcd 模组驱动命名规则

- 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. lcd 模组编译选项

lcm_module/makefile 文件中添加 lcd 模组驱动编译选项。

参考代码：

```
CSRCS := \
    mol_lcm_module.c \
    mol_lcm_table.c \
    lcm_mipi_ek79007.c \          //新增编译文件
```

2.4. lcd 模组注册

lcm_module/mol_lcm_table.c 文件中注册 lcd 模组。lcm_main_tab 绑定 VOU DHD0 主屏；lcm_sub_tab 绑定 VOU DHD1 副屏。

参考代码：

```
/*mipi lcm*/
extern lcm_cfg_t s_lcm_mipi_ek79007_info;//新增编译文件

/**------------------------------------------------------------------------------*
 **                  Local Variables                     *
 **------------------------------------------------------------------------------*/

/**------------------------------------------------------------------------------*
 **                  Constant Variables                      *
 **------------------------------------------------------------------------------*/
lcm_cfg_t *lcm_main_tab[] = {
    /*mipi lcm*/
    &s_lcm_mipi_ek79007_info,      //新增编译文件
    /*end lcm*/
    0,
};

lcm_cfg_t *lcm_sub_tab[] = {
    /*mipi lcm*/
    /*end lcm*/
    0,
};
```

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

参考代码：

```
const lcm_cfg_t s_lcm_mipi_ek79007_info =
{
    .verder_name = VENDER_NAME,                         //LCD 供应商名称(暂未使用)
    .name = "ek79007_dsi_wsvga_vdo",                    //LCD 屏名称
    .width = LCM_EK79007_WIDTH,                         //LCD 屏宽
    .height = LCM_EK79007_HEIGHT,                       //LCD 屏高
    .type = LCM_TYPE_MIPI, /*mcu, rgb, mipi*/           //LCD 屏类型
    .fps = 50,                                          //LCD 屏输出帧率
    .direction = LCM_DIRECT_NORMAL,
    .dp_sync = &_s_lcm_mipi_ek79007_dp_info,            //LCD 屏显示同步信息
    .infor = {
    .mipi = &_s_lcm_mipi_ek79007_mipi_info,             //LCD 总线信息
    },
    .fun = &_s_lcm_mipi_ek79007_ctrl,                   //LCD 控制函数信息
};
```

2.6. lcd 显示同步信息

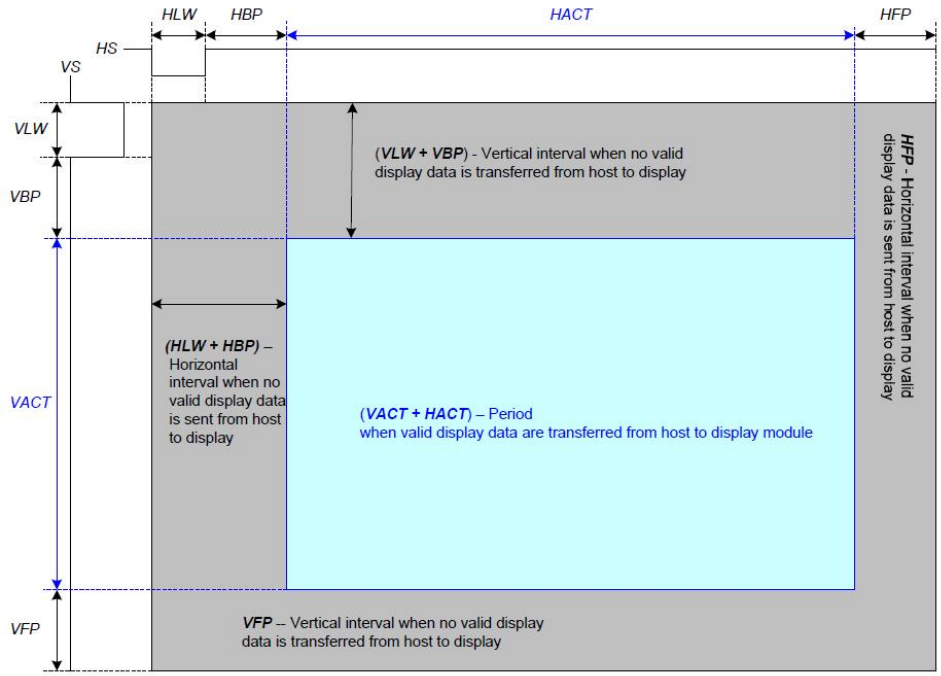
lcd 同步信号输出方式如（图表 2-2 同步信号）示，具体描述请参考 DSI/DPHY 协议文档。同步信息配置参数可以参考 LCD 模组文档或 LCD driver IC datasheet 如（图表 2-3 同步信号配置表）示，也可以请 lcd 供应商 FAE 提供。

以下是常规专有名词解释：

- 水平同步低电平宽度：（HLW, Horizon sync low width）
- 水平消隐后肩：（HBP, Horizon back porch）

- 水平有效显示周期：（HACTIVE, Horizon display period）
- 水平消隐前肩：（HFP, Horizon front porch）

- 垂直同步低电平宽度：（VLW, Vertical sync low width）
- 垂直消隐后肩：（VBP, Vertical back porch）
- 垂直有效显示周期：（VAVTIVE, Vertical display period）
- 垂直消隐前肩：（VFP, Vertical front porch）



图表 2- 2 同步信号

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

图表 2- 3 同步信号配置表

参考代码：

```
static lcm_display_sync_info_t _s_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,              //veitical display period
    .ver_front_porch = 12,                              //vertical front porch
};
```

2.7. lcd 接口注册

lcd 接口注册函数如下：

```
lcm_module_fun _s_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,
};
```

说明：

- poweron：LCD 上电及芯片内部 DSI、DPHY 驱动初始化
- poweroff：LCD 下电及芯片内部 DSI、DPHY 驱动去初始化
- identify： LCD 认证
- init： LCD 初始化屏参
- deinit： LCD 下电及芯片 DSI、DPHY 驱动去初始化
- suspend：LCD 休眠（暂未使用）
- resume： LCD 唤醒（暂未使用）

2.7.1. Power on 函数

功能描述：该函数主要负责 LCD 上电时序初始化、复位以及芯片内部 DSI、DPHY 等模块驱动的初始化。

参考代码：

```
static int32_t _lcm_mipi_ek79007_PowerOn(void * param, lcm_ops *ops)
{
    int32_t rtn = 0;

    #if LCM_EK79007_DEBUG
```

```
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "start");
    #endif

       /*  1、LCD 上电时序  */
              //TODO

       /*  2、LCD 屏复位  */
              //TODO

       /*  3、PLL/DSI/DPHY 初始化  */
    ops->ctrl_fun.mipi.bus_init((void *)param, (void *)&s_lcm_mipi_ek79007_info);

    #if LCM_EK79007_DEBUG
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "end");
    #endif

    return rtn;
}
```

2.7.2. Power off 函数

功能描述：该函数主要负责 LCD 下电、复位以及 DSI、DPHY 驱动的去初始化。

参考代码：

```
static int32_t _lcm_mipi_ek79007_PowerOff(void * param, lcm_ops *ops)
{
    int32_t rtn = 0;

    #if LCM_EK79007_DEBUG
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "start");
    #endif

       /*  1、PLL/DSI/DPHY 去初始化  */
    ops->ctrl_fun.mipi.bus_deinit((void *)param, (void *)&s_lcm_mipi_ek79007_info);

       /*  2、LCD 屏复位  */
              //TODO

       /*  3、LCD 下电时序  */
              //TODO
    #if LCM_EK79007_DEBUG
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "end");
    #endif
```

```
    return rtn;
}
```

2.7.3. Identify 函数

功能描述：该函数主要负责 MIPI LCD 读取认证屏 ID，如无需认证屏 ID，做空即可。

参考代码：

```
static int32_t _lcm_mipi_ek79007_identify(void * param, lcm_ops *ops)
{
#define MIPI_EK79007_ID_ADDR (0x81)
#define MIPI_EK79007_ID (0x88)

    int32_t rtn = SUCCESS;
    lcm_handle_t *lcmHandle = (lcm_handle_t *)param;
    uint8_t read_buf = 0;
    uint8_t read_cnt = 4;

    #if LCM_EK79007_DEBUG
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "start");
    #endif

    CHECK_PTR(param, NULL, "inParam ptr is NULL ~_~!\n");
    CHECK_PTR(ops, NULL, "ops ptr is NULL ~_~!\n");

    //goto EXIT;

    while(read_cnt--) {
       //读接口
        ops->ctrl_fun.mipi.dcs_read(lcmHandle->lcm_id, DSI_DI_DCS_READ_0_PARAM,
MIPI_EK79007_ID_ADDR, 1, &read_buf);

        if(MIPI_EK79007_ID == read_buf) {
            goto EXIT;
        }

        rtn = FAIL_ID_NO_MATCH;

        ops->delayMs(2);
    }

    printf("ek79007_4lane id : 0x%x:0x%x", MIPI_EK79007_ID, read_buf);

    EXIT:

    #if LCM_EK79007_DEBUG
```

```
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "end");
    #endif

    return rtn;
}
```

2.7.4. Init 函数

功能描述：该函数主要负责 LCD 初始化屏参，如该屏没有屏参，做空即可。

参考代码：

```
static int32_t _lcm_mipi_ek79007_init(void * param, lcm_ops *ops)
{
    int32_t rtn = 0;
    lcm_handle_t *lcmHandle = (lcm_handle_t *)param;

    #if LCM_EK79007_DEBUG
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "start");
    #endif
       //写接口
    rtn = ops->ctrl_fun.mipi.push_tab(lcmHandle->lcm_id, s_lcm_mipi_ek79007_init_setting,
ARRAY_SIZE(s_lcm_mipi_ek79007_init_setting));
    rtn = ops->ctrl_fun.mipi.push_tab(lcmHandle->lcm_id, lcm_sleep_out_setting,
ARRAY_SIZE(lcm_sleep_out_setting));

    #if LCM_EK79007_DEBUG
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "end");
    #endif

    return rtn;
}
```

2.7.5. Deinit 函数

功能描述：该函数主要负责 LCD 去初始化（包含 LCD 下电、DSI/DPHY/PLL 等模块复位下电）

参考代码：

```
static int32_t _lcm_mipi_ek79007_deinit(void * param, lcm_ops *ops)
{
    int32_t rtn = 0;

    #if LCM_EK79007_DEBUG
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "start");
    #endif
```

```
    rtn = _lcm_mipi_ek79007_PowerOff(param, ops);

    #if LCM_EK79007_DEBUG
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "end");
    #endif

    return rtn;
}
```

2.7.6. Suspend 函数

功能描述：该函数主要负责 LCD 休眠功能，目前已由 Deinit 函数代替，做空即可。

参考代码：

```
static int32_t _lcm_mipi_ek79007_suspend(void * param, lcm_ops *ops)
{
    int32_t rtn = 0;

    #if LCM_EK79007_DEBUG
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "start");
    #endif

    #if LCM_EK79007_DEBUG
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "end");
    #endif

    return rtn;
}
```

2.7.7. Resume 函数

功能描述：该函数主要负责 LCD 唤醒功能，目前已由 Init 函数代替，做空即可。

参考代码：

```
static int32_t _lcm_mipi_ek79007_resume(void * param, lcm_ops *ops)
{
    int32_t rtn = 0;

    #if LCM_EK79007_DEBUG
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "start");
    #endif

    #if LCM_EK79007_DEBUG
```

```
    ops->debug_fun(LCM_EK79007_DEBUG, __FUNCTION__, __LINE__, "end");
    #endif

    return rtn;
}
```

2.8. lcd 驱动模块加载

2.8.1. lcd 驱动加载参数信息

lcd 驱动加载模块通过命令方式认证指定屏。主要功能为：

- 加载 lcmx_id_cfg 参数选择认证模式
- 加载 lcdx_id 参数选择认证指定屏（仅当 lcmx_id_cfg=1 时）
- 运行 SDK 应用之前执行 set_lcm 命令可以选屏。

参考示例如下：

```
set_lcm lcm0_id_cfg=1 lcm1_id_cfg=1 lcm0_id=0 lcm1_id=4 //lcm0、lcm1 指定 id 选屏，且分别指定的 lcm id 为 0 和 4
```

参数信息具体说明如下：

- 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，根据全 table 中的 lcm 进行顺序认证
    - 1：指定 ID 模式，通过传入的 ID 参数进行认证，如果认证失败，接下来认证过程同正常流程模式
    - 2：关闭认证，main_lcm 或 sub_lcm 中将不会初始化任何 lcd 驱动模块
- lcm0_id：main lcm 认证指定 ID
- lcm1_id：sub lcm 认证指定 ID

参考代码：（指定 ID 模式）

- Main tab

```
lcm_cfg_t *lcm_main_tab[] = {
    /*mipi lcm*/
    &s_lcm_mipi_ek79007_info,      //lcm0_id = 0
    /*end lcm*/
    0,
};
```

lcm_main_tab 中的 lcm id 由 lcm0_id_cfg 控制指定
lcm0_id：lcm_main_tab 中数组下标(0,1,2…)

- Sub tab

```
lcm_cfg_t *lcm_sub_tab[] = {
    /*mipi lcm*/
    &s_lcm_mipi_ek79007_info,      //lcm1_id = 0
    /*end lcm*/
    0,
};
```

lcm_sub_tab 中的 lcm id 由 lcm_id_cfg 控制指定
lcm1_id：lcm_sub_tab 中数组下标(0,1,2…)

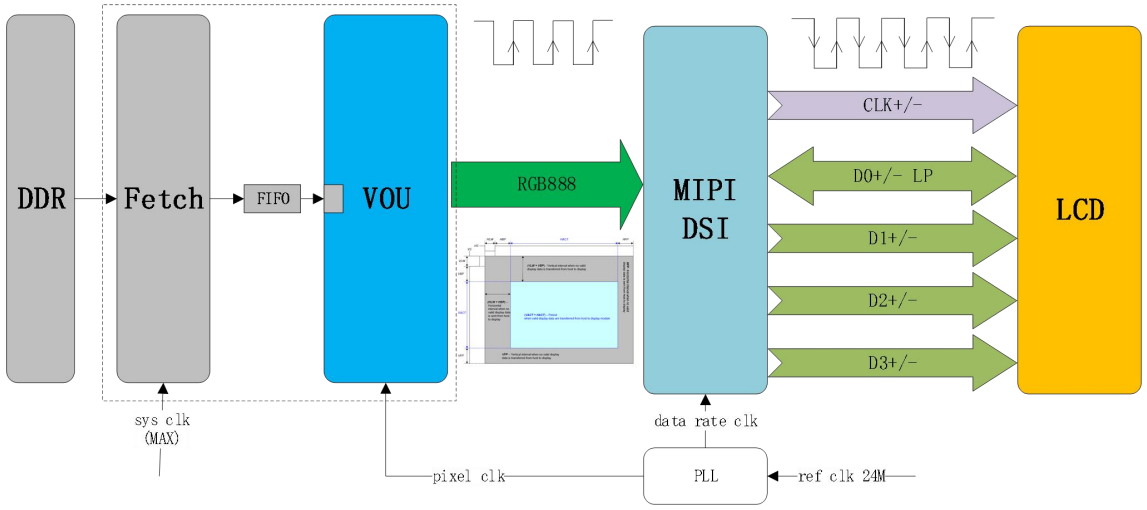
注：**通常情况下使用指定 ID 模式**

3. MIPI LCD GUIDE

MIPI 屏仅以（ek79007）为示例，如下所有结构体参数需按照实际屏配置。

3.1. lcd display flow

lcd 硬件模块显示流程如（图表 3-1display flow）所示：



图表 3-1display flow

3.2. lcd mipi pin mux config

MIPI LCD 芯片管脚配置请参考 pinmux 文档。

3.3. lcd mipi 总线信息

lcd mipi 总线包含以下几个信息：

- 工作模式：（work_mode, command_mode or video_mode）
- 通道数量：（lane_num, mipi data lane number）
- 数据包大小：（packet_size, mipi data packet size）
- 数据格式：（ mipi display data format）
- 时序：（the pointer of mipi timing）

参考代码：

```
static lcm_mipi_info _s_lcm_mipi_ek79007_mipi_info =
{
    .work_mode = DSI_VIDEO_MODE,                        //默认 video mode
    .lane_num = DSI_FOUR_LANE,                          //lane number
    .format = DSI_RGB888,                               //数据格式
    .packet_size = 256,                                 //默认配置 256
    .timing = (void *)&_s_lcm_mipi_ek79007_timing,      //dsi 时序
};
```

3.4. lcd mipi 总线 timing 配置

lcd mipi 总线时序（timing）配置有三部分，时钟通道（clk lane）、数据通道（data lane）、低速读（lp read）, 默认驱动程序根据协议自动计算时序（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。clk 配置默认驱动程序根据时钟自动计算或者通过 manual 方式设定。

mipi 总线 timing 信息包含以下信息

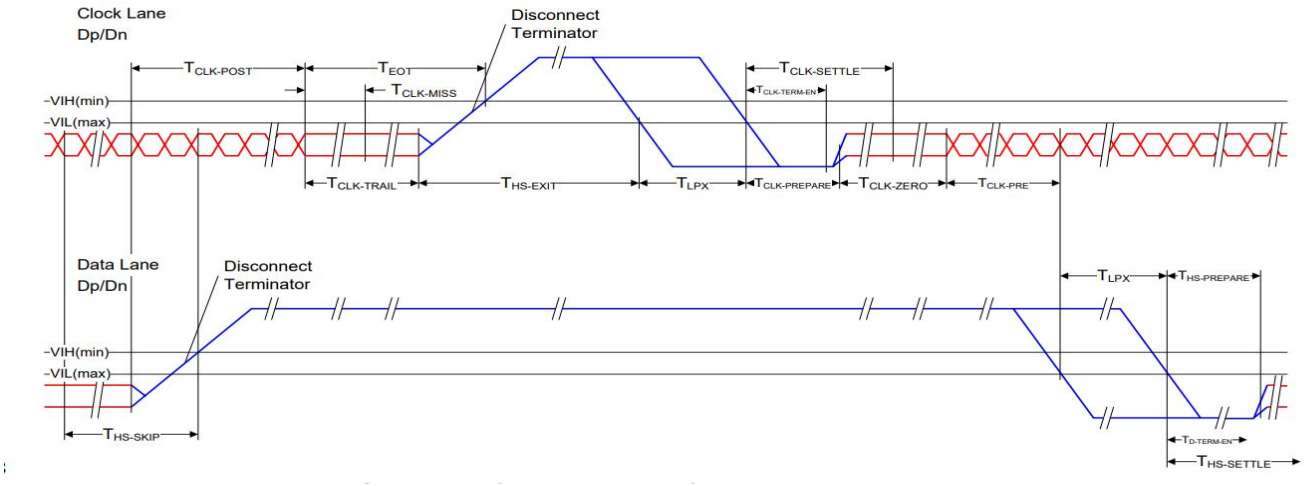
- dsi_phy_clane_timing_t：时钟通道时序参数
- dsi_phy_dlane_timing_t：数据通道时序参数
- dsi_phy_lane0_read_t：低速 lane0 通道读时序参数
- dsi_clk_cfg_t：dsi 时钟参数

实际结构体如下：

```
typedef struct {
    dsi_phy_clane_timing_t clane;        //clk lane cfg param
```

```
    dsi_phy_dlane_timing_t dlane;        //data lane cfg param
    dsi_phy_lane0_read_t read;           //lp read cfg param
    dsi_clk_cfg_t clk;                   //dsi clk cfg param
}dsi_timing_t;
```

lcd mipi clk timing 配置

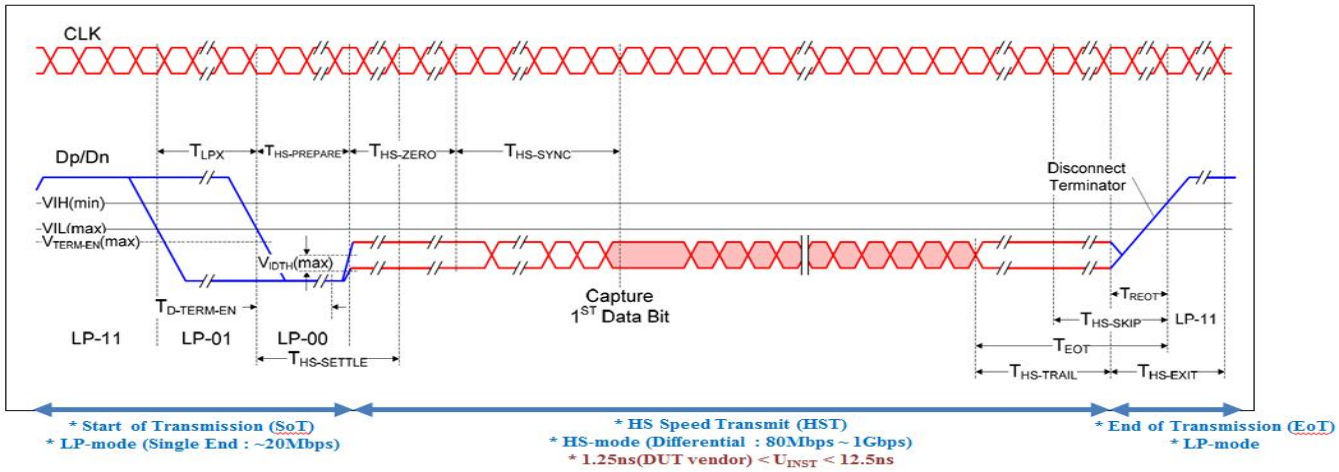


图表 3- 2 mipi clk 协议

参考代码：

```
static dsi_timing_t _s_lcm_mipi_ek79007_timing =
{
    /*clk lane timing cfg*/
    .clane.mode = 0,
    .clane.lp11 = 0,
    .clane.lp01 = 0,
    .clane.zero = 0x22,
    .clane.prepare = 0x1,
    .clane.pre = 0x05,
    .clane.post = 0x0b,
    .clane.trail = 0x04,
    .clane.inittime = 0x00,
    .clane.exit = 0x06,
       …
}
```

lcd mipi data timing 配置



图表 3- 3 mipi data 协议

参考代码：

```
static dsi_timing_t _s_lcm_mipi_ek79007_timing =
{
        …
    /*data lane timing cfg*/
    .dlane.mode = 0,
    .dlane.lp11 = 0,
    .dlane.lp01 = 6,
    .dlane.zero = 0x0f,
    .dlane.prepare = 0x01,
    .dlane.trail = 0x05,
    .dlane.inittime = 0x00,
    .dlane.exit = 0x06,
        …
}
```

lcd mipi lp read timing 配置

参考代码：

```
static dsi_timing_t _s_lcm_mipi_ek79007_timing =
{
        …
    .read.mode = 1,
    .read.tago = 0x1b,
    .read.tasure = 0x0b,
    .read.taget = 0x22,
        …
}
```


lcd mipi pll clk 配置

参考代码：

```
static dsi_timing_t _s_lcm_mipi_ek79007_timing =
{
       …
    /*clk cfg*/
    .clk = {
        .vou.mode = 1,              //可删除
        .vou.sel = 0x3,             //可删除
        .vou.div = 0,               //可删除

        .mif.mode = 0,
        .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,
    }
}
```

概念：

- **AUTO 模式:配置 clane.mode = 0；dlane.mode=0；read.mode=0；mif.mode=0；其余参数由驱动内部自动计算配置到硬件，所以用户直接配置 0 即可。**
- **MANUL 模式:配置 clane.mode = 1；dlane.mode=1；read.mode=1；mif.mode=1；配置该模式之前需要提前获取到结构体相关参数值，将其配置到结构体中。**

lcd mipi 初始化配置参数

注：所有初始化配置参数由屏厂提供。

mipi 下发参数数据包类型有 2 种方式：

- 短包：4 bytes，由 3 部分组成
- 长包：长包：6 ~ 65541 bytes，同样由 3 部分组成

以下为长短包组包详解：

- DSI_DI_DCS_WRITE_0_PARAM

param1：写模式短包

param2：配置参数数量（0x1）
param3：配置参数（实际屏参）

参考代码：

```
static dsi_setting_tab_t s_lcm_mipi_st7701s_init_code[] = {
    {DSI_DI_DCS_WRITE_0_PARAM, 0x01, {0x01}},
}
```

- DSI_DI_DCS_WRITE_1_PARAM

param1：写模式短包
param2：配置参数数量（0x2）
param3-n：配置参数（实际屏参）

参考代码：

```
static dsi_setting_tab_t s_lcm_mipi_st7701s_init_code[] = {
    {DSI_DI_DCS_WRITE_1_PARAM, 0x02, {0xEF,0x08}},
}
```

- DSI_DI_DCS_WRITE_LONG

param1：写模式长包
param2：配置参数数量（param3 第一个参数 param3[0] + 2）
param3-n： param3[0]: {param3[2]- param3[n]}的参数个数 n-2+1
param3[1]: 默认写 0x0
param3[2-n]: 实际屏参

参考代码：

```
static dsi_setting_tab_t s_lcm_mipi_st7701s_init_code[] = {
    {DSI_DI_DCS_WRITE_LONG,    0x08, {0x06,0x00, 0xFF,0x77,0x01,0x00,0x00,0x10}},
}
```

- DSI_DI_DELAY

param1：delay 标志
param2：delay 时间，单位 ms
param3-n：无效

- DSI_DI_DELAY

param1：结束标志
param2：无效
param3-n：无效

注：**实际请参考 DSI 协议文档，本文档不对协议做过多讲解。**

4. RGB LCD GUIDE

RGB 屏仅以 at070tn94 为示例，为示例，如下所有结构体参数需按照实际屏配置。

4.1. lcd rgb pin mux config

RGB LCD 芯片管脚配置请参考 pinmux 文档。

4.2. lcd rgb 总线信息

lcd rgb 总线信息包含以下几个信息：

- ctrl_bus_mode：默认配置 SERIAL_BUS_MAX
- video_bus_width：默认配置 0
- h_sync_pol：水平信号极性
- v_sync_pol：垂直信号极性
- de_pol：de 信号极性
- format：数据格式（LCM_RGB_888/ LCM_RGB_666/ LCM_RGB_565）
- if_type：接口类型（RGB_IF_18BITS/ RGB_IF_16BITS/ RGB_IF_24BITS）
- order：RGB 分量顺序（如 RGB_ORDER_RGB，R 在高位，B 在低位）
- dither：默认配置 0
- bitswap：RGB 内部 bit 位置 swap
- serial：默认配置为 NULL
- clk：rgb pll 时钟

参考代码：

```
static lcm_rgb_info s_lcm_rgb_at070tn94_rgb_info =
{
    .ctrl_bus_mode = SERIAL_BUS_MAX,
    .video_bus_width = 0,
    .h_sync_pol = 0,
    .v_sync_pol = 0,
    .de_pol = 0,
    .format = LCM_RGB_888,
    .if_type = RGB_IF_24BITS,
    .order = RGB_ORDER_RGB,
    .dither = 0,
    .bitswap = 0,
    .serial = NULL,
```

```
    .clk = (void *)&s_lcm_rgb_at070tn94_clk,
};
```

4.3. lcd rgb 总线时钟配置

rgb 总线时钟 clk 配置可以通过默认驱动程序根据时钟自动计算或者通过 manual 方式设定。

参考代码：

```
static rgb_clk_t s_lcm_rgb_at070tn94_clk =
{
    .mif.mode = 0,
    .mif.pll_doubler = 0,
    .mif.pll_div_s = 1,
    .mif.pll_n = 0,
    .mif.pll_kint = 0x2aaaaa,
    .mif.pll_nint = 0x55,
    .mif.pixelclk_div = 6,
    .mif.pll_pdiv = 3,
    .mif.dhd_div = 7,
};
```

：

- **AUTO 模式:配置 mif.mode=0；其余参数由驱动内部自动计算配置到硬件，所以用户直接配置 0 即可。**
- **MANUL 模式:配置 mif.mode=1；配置该模式之前需要提前获取到结构体相关参数值，将其配置到结构体中。**

5. MCU LCD GUIDE

MCU 屏以 st7796s 为示例，，如下所有结构体参数需按照实际屏配置。

5.1. lcd mcu 总线信息

mcu 总线信息包含以下信息：

- format：数据格式（LCM_RGB_666/ LCM_RGB_565）
- if_type：接口类型（RGB_IF_18BITS/ RGB_IF_16BITS/ RGB_IF_8BITS）
- trans：MCU_LSB_FIRST/ MCU_HSB_FIRST
- order：RGB 分量顺序（如 RGB_ORDER_RGB，R 在高位，B 在低位）
- dither：默认配置 0
- bitswap：RGB 内部 bit 位置 swap
- te：默认全部关闭
- timing：mcu 总线 timing

- clk：mcu pll 时钟

参考代码:

```
static lcm_mcu_info _s_lcm_mcu16_st7796s_info =
{
    .format = LCM_RGB_565,
    .if_type = MCU_IF_16BITS,
    .trans = MCU_HSB_FIRST,
    .order = RGB_ORDER_BGR,
    .dither = 0,
    .bitswap = 0,
    .rd_eb = 0,
    .te = {
        .eb = 0,
        .mode = 0,
        .edge = 0,
    },
    .timing = &_s_lcm_mcu16_st7796s_bus_timing,
    .clk = &_s_lcm_mcu16_st7796s_clk,
};
```

5.2. lcd mcu 总线 timing 配置

mcu 总线 timing 与 vou 显示模块相关，以示例 timing 参数为准

参考代码：

```
lcm_parallel_bus_timing _s_lcm_mcu16_st7796s_bus_timing =
{
    .csType = 0,
    .rdSetup = 1,
    .rdHold = 1,
    .rdDestory = 1,
    .wrSetup = 1,
    .wrHold = 1,
    .wrDestory = 1,
};
```

5.3. lcd mcu 总线时钟配置

mcu 总线时钟 clk 配置可以通过默认驱动程序根据时钟自动计算或者通过 manual 方式设定。

参考代码：

```
lcm_mcu_clk_t _s_lcm_mcu16_st7796s_clk =
{
    .mif.mode = 1,
    .mif.pll_doubler = 0,
    .mif.pll_div_s = 1,
    .mif.pll_n = 0,
    .mif.pll_kint = 0x10624,
    .mif.pll_nint = 0x2b,
    .mif.pixelclk_div = 14,
    .mif.pll_pdiv = 0,
    .mif.dhd_div = 1,
};
```

概念：

- AUTO 模式:配置 mif.mode=0；其余参数由驱动内部自动计算配置到硬件，所以用户直接配置 0 即可。
- MANUL 模式:配置 mif.mode=1；配置该模式之前需要提前获取到结构体相关参数值，将其配置到结构体中。