MT8183 LCM Porting Guide

LCM Porting Guide

Customer Support

MT8183

Doc No: CS6000-D12C-PPG-V1.0EN
Version: V1.0
Release date: 2018-07-09
Classification: customer

1 Introduction

LCM Porting Guide For MT8183 and O, about lcm porting, there is lcm_drv.c architecture :

<kernel>/driver/misc/mediatek/video/common
mtkfb.c, mtkfb_dummy.c
<kernel>/driver/misc/mediatek/video/include
disp_session.h, disp_svp.h, mtkfb.h, mtkfb_info.h, mtkfb_vsync.h
<kernel>/driver/misc/mediatek/video/<platform>/dispsys
ddp_ovl.c/h, ddp_rdma.c/h, ddp_reg.h, ddp_color.c/h, ddp_color_format.c, ddp_debug.c/h, ddp_dpi.c/h, ddp_drv.c/h, ddp_dsi.c/h, ddp_dump.c/h, ddp_hal.h, ddp_info,c/h, ddp_irq.c/h, ddp_irq.h, ddp_log.h, ddp_manager.c/h, ddp_matrix_para.h, ddp_met.c/h, ddp_mmp.c/h, ddp_path.c/h, ddp_wdma.c/h, disp_event.h, display_recorder.c/h
<kernel>/driver/misc/mediatek/video/<platform>/videox
disp_drv_platform.h, debug.c/h, disp_assert_layer.c, disp_assert_layer_priv.h, disp_drv_ddp.h, disp_drv_log.h, disp_dts_gpio.c/h, disp_recovery.c/h, disp_helper.c/h, disp_lcm.c/h, disp_utils.c/h, fbconfig_kdebug_k2.c/h, font_8x16.c, mtk_disp_mgr.c/h, mtkfb_console.c/h, mtkfb_fence.c/h, mtk_mira.c/h, mtk_ovl.c/h, primary_display.c/h
<kernel>/driver/misc/mediatek/lcm
lcm_drv.c

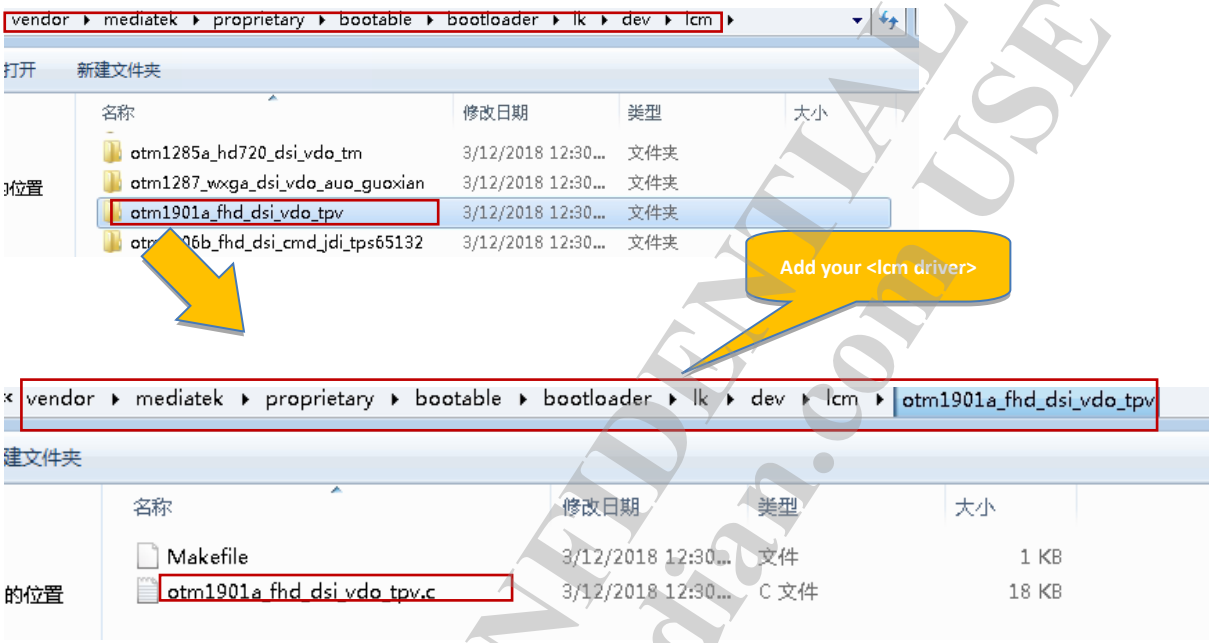
O display driver location as below,

1. Lcm driver porting for LK part,
   Step 1: Add your <lcm driver>

Add your <lcm driver> into the following path:

alps\vendor\mediatek\proprietary\bootable\bootloader\lk\dev\lcm

Take <otm1901a_fhd_dsi_vdo_tpv> for example:



Step 2: Add your <lcm config> in <project> makefile

Add your <lcm confing> in <project>.mk

alps\vendor\mediatek\proprietary\bootable\bootloader\lk\project\<project>.mk

Take < otm1901a_fhd_dsi_vdo_tpv > for example:

1). If the case is single LCM, add your <lcm> in CUSTOM_LK_LCM, as follow,

```
14 MTK_MT6370_PMU_BLED_SUPPORT := yes
15 MTK_LCM_PHYSICAL_ROTATION = 0
16 CUSTOM_LK_LCM="otm1901a_fhd_dsi_vdo_tpv"
```

2). If the case is multiple LCMs, add your <lcms> in CUSTOM_LK_LCM, and simply separated by space key,as follow,

```
MTK_MT6370_PMU_BLED_SUPPORT := yes
MTK_LCM_PHYSICAL_ROTATION = 0
CUSTOM_LK_LCM="otm1901a_fhd_dsi_vdo_tpv otm1285a_hd720_dsi_vdo_tm"
```

Step 3: Add your <lcm main structure> into lcm list

Add your <lcm main structure> into lcm list in

alps\vendor\mediatek\proprietary\bootable\bootloader\lk\dev\lcm\mt65xx_lcm_list.c

Take < otm1901a_fhd_dsi_vdo_tpv > for example:

Add your <lcm main structure> into lcm list, as follow,

```
extern LCM_DRIVER jd9365_hd720_dsi_lcm_drv;
extern LCM_DRIVER otm1901a_fhd_dsi_vdo_tpv_lcm_drv;
```

```
#if defined(OTM1901A_FHD_DSI_VDO_TPV)
  &otm1901a_fhd_dsi_vdo_tpv_lcm_drv,
#endif
};
```

Step 4: Switch logo if LCM resolution is different.

Modify define marco of BOOT_LOGO in

alps\vendor\mediatek\proprietary\bootable\bootloader\lk\project\<project>.mk

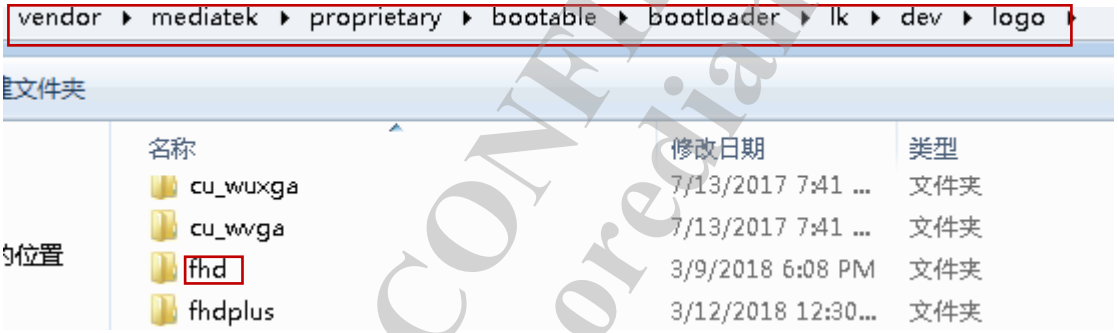
Take < otm1901a_fhd_dsi_vdo_tpv > for example:

Select the matching macro based on the resolution of the LCM,

```
MTK_SEC_FASTBOOT_UNLOCK_SUPPORT = yes
BOOT_LOGO := fhd
DEBUG := 2
#DEFINES += WITH_DEBUG_DCC=1
```

If the BOOT_LOGO does not match, it does not affect the boot, but logo will display the exception.

Logo location as follow,



Step 5: Rebuild lk

Rebuild lk and re-download lk.bin.

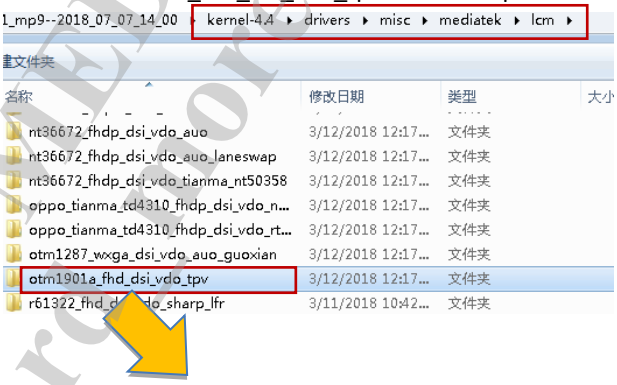
2. Lcm driver porting for Kernel part,

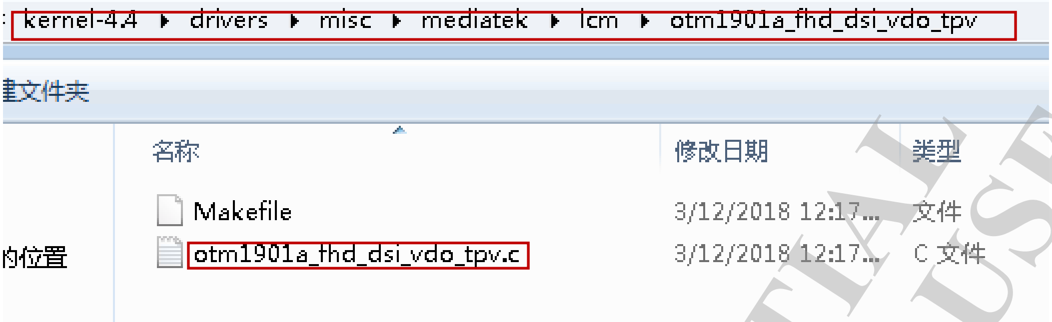
Step 1: Add your <lcm driver>

Add your <lcm driver> into the following path:

alps\<kernel>\drivers\misc\mediatek\lcm

Take < otm1901a_fhd_dsi_vdo_tpv > for example:



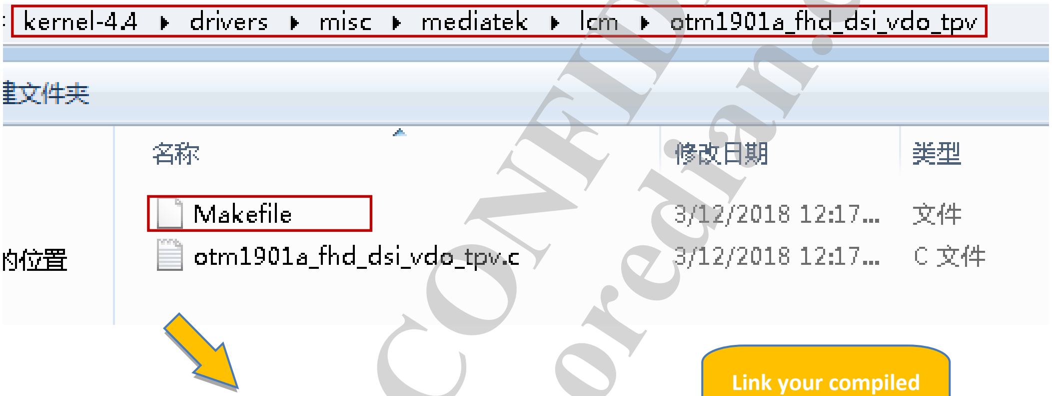


Step 2: Link your <lcm object>

Link your compiled <lcm object> in

alps\<kernel>\drivers\misc\mediatek\lcm\<lcm>\Makefile

Take < otm1901a_fhd_dsi_vdo_tpv > for example:



Link your compiled object

```

obj-y += otm1901a_fhd_dsi_vdo_tpv.o

ccflags-$(CONFIG_MTK_LCM) += -I$(srctree)/drivers/misc/mediatek/lcm/inc

```

Step 3: Add your <lcm main structure> into lcm list

Add your <lcm main structure> into lcm list in

alps\<kernel>\drivers\misc\mediatek\lcm\mt65xx_lcm_list.c

Take < otm1901a_fhd_dsi_vdo_tpv > for example:

```
#if defined(OTM1901A_FHD_DSI_VDO_TPV)
	&otm1901a_fhd_dsi_vdo_tpv_lcm_drv,
#endif

#if defined(ST7789H2_DBI_C_3WIRE)
	&st7789h2_dbi_c_3wire_lcm_drv,
#endif
```

alps\kernel-4.4\drivers\misc\mediatek\lcm\mt65xx_lcm_list.h

```
extern LCM_DRIVER jd9365_hd720_dsi_lcm_drv;
extern LCM_DRIVER otm1901a_fhd_dsi_vdo_tpv_lcm_drv;
extern LCM_DRIVER st7789h2_dbi_c_3wire_lcm_drv;
```

Step 4: Add your <lcm config> in <project> defconfig, and modify LCM width and height

alps\<kernel>\arch\<arm64>\configs\<project>_defconfig

alps\<kernel>\arch\<arm64>\configs\<project>_debug_defconfig

Take < otm1901a_fhd_dsi_vdo_tpv > for example:

1). If the case is single LCM, add your <lcm> in CUSTOM_LK_LCM

```
256 CONFIG_MTK_LCM=y
257 CONFIG_CUSTOM_KERNEL_LCM="otm1901a_fhd_dsi_vdo_tpv"
258 CONFIG_MTK_LENS=y
```

2). If the case is multiple LCMs, add your <lcms> in CUSTOM_LK_LCM, and simply separated by space key

```
256 CONFIG_MTK_LCM=y
257 CONFIG_CUSTOM_KERNEL_LCM="otm1901a_fhd_dsi_vdo_tpv otm1901a_fhd_dsi_vdo_tpv"
258 CONFIG_MTK_LENS=y
```

Modify the LCM width according to the new resolution,

```
CONFIG_MTK_LCM_PHYSICAL_ROTATION="0"
CONFIG_LCM_HEIGHT="1920"
CONFIG_LCM_WIDTH="1080"
CONFIG_MTK_AAL_SUPPORT=y
```

Step 6: Rebuild kernel and bootimage

Return to alps folder in console.

Rebuild kernel and bootimage, and re-download boot.img

3. Lcm driver porting for Device part,

Step 1: Switch logo modify LCM width and height if LCM resolution is different,

alps \device\<mediatekprojects>\<project>\ProjectConfig.mk

Take < otm1901a_fhd_dsi_vdo_tpv > for example:

```
BOOT_LOGO = fhd
BUILD_KERNEL = yes
BUILD_LK = yes
```

Need to the same as LK part config

```
KBUILD_OUTPUT_SUPPORT = yes
LCM_FAKE_HEIGHT = 0
LCM_FAKE_WIDTH = 0
LCM_HEIGHT = 1920
LCM_WIDTH = 1080
LINUX_KERNEL_VERSION = kernel-4.4
```

According to the new LCM resolution

4. Implement Lcm_drv.c configuration, take "<otm1901a_fhd_dsi_vdo_tpv.c" for example:

Step1: implement the lcm_drv structure,

```
690
691 LCM_DRIVER otm1901a_fhd_dsi_vdo_tpv_lcm_drv = {
692     .name = "otm1901a_fhd_dsi_vdo_tpv",
693     .set_util_funcs = lcm_set_util_funcs,
694     .get_params = lcm_get_params,
695     .init = lcm_init_lcm,
696     .suspend = lcm_suspend,
697     .resume = lcm_resume,
698 /*  .compare_id = lcm_compare_id, */
699     .init_power = lcm_init_power,
700     .resume_power = lcm_resume_power,
701     .suspend_power = lcm_suspend_power,
702 /*  .esd_check = lcm_esd_check, */
703     .ata_check = lcm_ata_check,
704     .update = lcm_update,
705 };
706
```

The name need to the same in LK and kernel lcm_drv

Only for CMD mode

Step2: Fill the LCM parameters

Configure the basic information according to the HW connection, LCM type, DSI mode , LCM size and PLL in lcm_get_params function,

```
72  #define LCM_DSI_CMD_MODE                                    0
73  #define FRAME_WIDTH                                     (1080)
74  #define FRAME_HEIGHT                                    (1920)
75  #define GPIO_OUT_ONE  1
76  #define GPIO_OUT_ZERO 0
```

```
417  static void lcm_get_params(LCM_PARAMS *params)
418  {
419      memset(params, 0, sizeof(LCM_PARAMS));
420
421      params->type = LCM_TYPE_DSI;
422
423      params->width = FRAME_WIDTH;
424      params->height = FRAME_HEIGHT;
425
426  #if (LCM_DSI_CMD_MODE)
427      params->dsi.mode = CMD_MODE;
428      params->dsi.switch_mode = SYNC_PULSE_VDO_MODE;
429  #else
430      params->dsi.mode = SYNC_PULSE_VDO_MODE;
431      params->dsi.switch_mode = CMD_MODE;
432  #endif
433      params->dsi.switch_mode_enable = 0;
434
435      /* DSI */
436      /* Command mode setting */
437      params->dsi.LANE_NUM = LCM_FOUR_LANE;
438      /* The following defined the fomat for data coming from LCD engine. */
439      params->dsi.data_format.color_order = LCM_COLOR_ORDER_RGB;
440      params->dsi.data_format.trans_seq = LCM_DSI_TRANS_SEQ_MSB_FIRST;
441      params->dsi.data_format.padding = LCM_DSI_PADDING_ON_LSB;
442      params->dsi.data_format.format = LCM_DSI_FORMAT_RGB888;
443
444      /* Highly depends on LCD driver capability. */
445      params->dsi.packet_size = 256;
446      /* video mode timing */
447
448      params->dsi.PS = LCM_PACKED_PS_24BIT_RGB888;
449
450      params->dsi.vertical_sync_active = 2;
451      params->dsi.vertical_backporch = 8;
452      params->dsi.vertical_frontporch = 20;
453      params->dsi.vertical_frontporch_for_low_power = 620;
454      params->dsi.vertical_active_line = FRAME_HEIGHT;
455
456      params->dsi.horizontal_sync_active = 10;
457      params->dsi.horizontal_backporch = 20;
458      params->dsi.horizontal_frontporch = 40;
459      params->dsi.horizontal_active_pixel = FRAME_WIDTH;
```

How many lanes in LCM DSI interface

Only for VDO mode timing setting

MIPI Clock Spread Spectrum:

```
params->dsi.ssc_disable = 0;
params->dsi.ssc_range = 4;
```

Ssc_disable: 0 → enable ssc, 1→ disable ssc

Ssc_range: 1~8, default value:0

PLL CLK:

```
#if (LCM_DSI_CMD_MODE)
    params->dsi.PLL_CLOCK = 420;    /* this value must be in MTK suggested table */
#else
    params->dsi.PLL_CLOCK = 440;    /* this value must be in MTK suggested table */
#endif
```

```
1、in VDO mode data_rate calculation formula:
Data rate= (Height+VSA+VBP+VFP)*(Width+HSA+HBP+HFP)* total_bit_per_pixel*frame_per_second/total_lane_num

2、in CMD mode data_rate calculation formula:
Data rate= width*height*1.2* total_bit_per_pixel*frame_per_second/total_lane_num

Definitions:
Data_rate : the value is 2 times PLLCLK, as Data rate= 2*PLL_CLOCK
Width, Height  : LCM resolution
VSA VBP VFP : DSI vdo mode vertical porch timing
HSA HBP HFP : DSI vdo mode horizontal porch timing
total_bit_per_pixel : how many bits per pixel, for example:RGB888,it is 24bit;
frame_per_second : how many frames per second, Usually 60 FPS
total_lane_num : how many pairs of data lanes.
```

Mipi clk mode setting:

a. non-continous clk mode:

```
params->dsi.cont_clock = 0;
params->dsi.clk_lp_per_line_enable = 1;
```

b. continuous clk mode:

```
params->dsi.cont_clock = 1;
/* params->dsi.clk_lp_per_line_enable = 1; */
```

ESD check configuration:

```
params->dsi.esd_check_enable = 1;
params->dsi.customization_esd_check_enable = 0;
params->dsi.lcm_esd_check_table[0].cmd = 0x53;
params->dsi.lcm_esd_check_table[0].count = 1;
params->dsi.lcm_esd_check_table[0].para_list[0] = 0x24;

esd_check_enable: 1-->enable, 0-->disable;
customization_esd_check_enable: 0--> EX TE check, 1--> read lcm register

If do ESD by reading LCM IC register
Cmd:  the register you will read
Count:  how many parameters will be read back
Para_list:  the right value should been read back
If the read-back value unequal to the para_list , display system will do recovery
```

Step3: Implement LCM init function

According the init process specified in LCM datasheet, pull down/up the reset pin, delay and set LCM init register

```
527  static void lcm_resume_power(void)
528  {
529      lcm_set_gpio_output(GPIO_LCD_PWR_EN, GPIO_OUT_ONE);
530      MDELAY(20);
531
532      lcm_set_gpio_output(GPIO_LCD_PWR2_EN, GPIO_OUT_ONE);
533      MDELAY(20);
534
535      SET_RESET_PIN(1);
536      MDELAY(20);
537  }
538
539  static void lcm_init_lcm(void)
540  {
541      push_table(init_setting, sizeof(init_setting) / sizeof(struct LCM_setting_table), 1);
542  }
```

Step4: Implement LCM update function (only for CMD mode)

(Push_table 和 dsi_set_cmdq Please refer to the next chapter command queue)

```
554  static void lcm_update(unsigned int x, unsigned int y, unsigned int width, unsigned int height)
555  {
556      unsigned int x0 = x;
557      unsigned int y0 = y;
558      unsigned int x1 = x0 + width - 1;
559      unsigned int y1 = y0 + height - 1;
560
561      unsigned char x0_MSB = ((x0 >> 8) & 0xFF);
562      unsigned char x0_LSB = (x0 & 0xFF);
563      unsigned char x1_MSB = ((x1 >> 8) & 0xFF);
564      unsigned char x1_LSB = (x1 & 0xFF);
565      unsigned char y0_MSB = ((y0 >> 8) & 0xFF);
566      unsigned char y0_LSB = (y0 & 0xFF);
567      unsigned char y1_MSB = ((y1 >> 8) & 0xFF);
568      unsigned char y1_LSB = (y1 & 0xFF);
569
570      unsigned int data_array[16];
571
572      data_array[0] = 0x00053902;
573      data_array[1] = (x1_MSB << 24) | (x0_LSB << 16) | (x0_MSB << 8) | 0x2a;
574      data_array[2] = (x1_LSB);
575      dsi_set_cmdq(data_array, 3, 1);
576
577      data_array[0] = 0x00053902;
578      data_array[1] = (y1_MSB << 24) | (y0_LSB << 16) | (y0_MSB << 8) | 0x2b;
579      data_array[2] = (y1_LSB);
580      dsi_set_cmdq(data_array, 3, 1);
581
582      data_array[0] = 0x002c3909;
583      dsi_set_cmdq(data_array, 1, 0);
584  }
```

Step5: Implement LCM suspend/resume functions

```
544  static void lcm_suspend(void)
545  {
546      push_table(lcm_suspend_setting, sizeof(lcm_suspend_setting) / sizeof(struct LCM_setting_table), 1);
547  }
548
549  static void lcm_resume(void)
550  {
551      lcm_init_lcm();
552  }
```

Step6: Fill in the initialization parameters

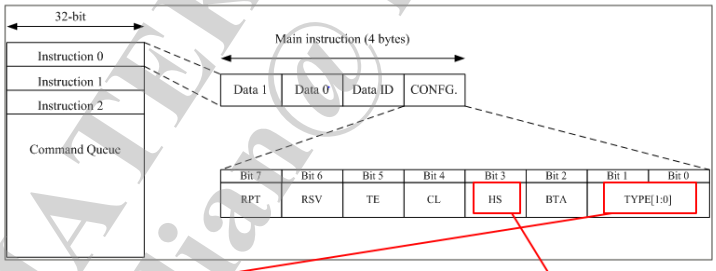
Get the initial code from LCM FAE

```
static struct LCM_setting_table init_setting[] = {
    {0x00, 1, {0x00} },
    {0xFF, 4, {0x19, 0x01, 0x01, 0x00} },
    {0x00, 1, {0x80} },
    {0xFF, 2, {0x19, 0x01} },
    {0x00, 1, {0x00} },
    {0x1C, 1, {0x33} },
    {0x00, 1, {0xA0} },
    {0xC1, 1, {0xE8} },
    {0x00, 1, {0xA7} },
    {0xC1, 1, {0x00} },
    {0x00, 1, {0x90} },
    {0xC0, 6, {0x00, 0x2F, 0x00, 0x00, 0x00, 0x01} },
    {0x00, 1, {0xC0} },
    {0xC0, 6, {0x00, 0x2F, 0x00, 0x00, 0x00, 0x01} },
    {0x00, 1, {0x9A} },
    {0xC0, 1, {0x1E} },
    {0x00, 1, {0xAC} },
    {0xC0, 1, {0x06} },
    {0x00, 1, {0xDC} },
```

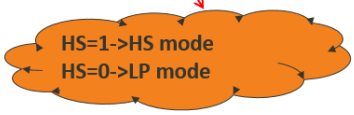
5. COMMAND QUEUE

DSI Command Queue(1/2)

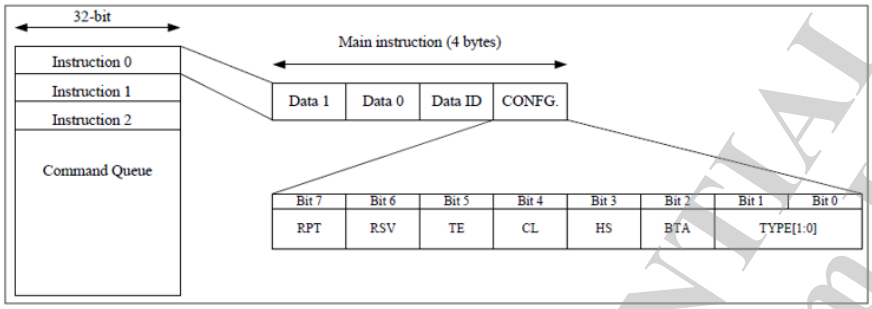
- Two dedicated command queues with 32-bit wide and 32-entry depth for each.



- Type[1:0]
  - 2'b00: (**short packet**) Read/write command
  - 2'b01: (**long packet**) Frame buffer write command (from LCD)
  - 2'b10: (**long packet**) Generic long packet write command (from command queue)
  - 2'b11: (**short packet**) Frame buffer read command



DSI Command Queue(2/2)



Byte 3	Byte 2	Byte 1	Byte 0
Data 1	Data 0	Data ID	CONFG.

Fig. 5-8: Type-0 instruction format

Byte 3	Byte 2	Byte 1	Byte 0
Mem start 1 (optional)	Mem start 0	Data ID	CONFG.

Fig. 5-9: Type-1 instruction format

Byte 3	Byte 2	Byte 1	Byte 0
WC 1	WC 0	Data ID	CONFG.
Data 3	Data 2	Data 1	Data 0
		Data WC-1	Data WC-2

Fig. 5-10: Type-2 instruction format

Byte 3	Byte 2	Byte 1	Byte 0
Men start 1 (optional)	Mem start 0	Data ID	CONFG.

Fig. 5-11: Type-3 instruction format

dsi_set_cmdq(*pdata,queue_size,force_update)

Type 0 & Type 2

```
static void init_lcm_registers(void)
{
    unsigned int data_array[16];

    data_array[0]=0x00043902;
    data_array[1]=0x6983FFB9;
    dsi_set_cmdq(&data_array, 2, 1);


    data_array[0] = 0x073A1500;
    dsi_set_cmdq(&data_array, 1, 1);


    data_array[0] = 0x00110500; // Sleep Out
    dsi_set_cmdq(&data_array, 1, 1);
    MDELAY(100);

    data_array[0] = 0x00290500; // Display On
    dsi_set_cmdq(&data_array, 1, 1);
    MDELAY(10);
```

1.data_array[0]=0x00043902
0x02→type 2 Generic Long Write
0x39→DI=0x39,DCS long write command(long packet)
0x0004→WC=4,4Byte data
2.data_array[1]=0x6983FFB9写LCM Register
→command: 0xB9
params:0xFF, 0X83, 0X69

1.data_array[0]=0x073A1500
0x00→type 0 Short Packet Write
0x15→DI=0x15,DCS short write, 1 params
0x05→DI=0X05,DCS short write, no params
0x073A→Data0,Data1 = 0x3A,0X07
→DCS command=0x3A写LCM Register
params=0x07

Byte 3	Byte 2	Byte 1	Byte 0
WC 1	WC 0	Data ID	CONFG.
Data 3	Data 2	Data 1	Data 0
		Data WC-1	Data WC-2

Fig. 5-10: Type-2 instruction format

Byte 3	Byte 2	Byte 1	Byte 0
Data 1	Data 0	Data ID	CONFG.

Fig. 5-8: Type-0 instruction format

dsi_set_cmdq_V2(cmd, count, *para_list, force_update)

```
static void lcm_resume(void)
{
 push_table(lcm_sleep_out_setting, sizeof(lcm_sleep_out_setting) / sizeof(struct LCM_setting_table), 1);
}
```

```
static struct LCM_setting_table lcm_sleep_out_setting[] = {
    // Sleep Out
    {0x11, 1, {0x00}},
    {REGFLAG_DELAY, 120, {}},

    // Display ON
    {0x29, 1, {0x00}},
    {REGFLAG_END_OF_TABLE, 0x00, {}}
};
```

```
struct LCM_setting_table {
    unsigned cmd;
    unsigned char count;
    unsigned char para_list[64];
};
```

```
void push_table(struct LCM_setting_table *table, unsigned int count, unsigned char force_update)
{
    unsigned int i;

    for(i = 0; i < count; i++) {

        unsigned cmd;
        cmd = table[i].cmd;

        switch (cmd) {

            case REGFLAG_DELAY :
                MDELAY(table[i].count);
                break;

            case REGFLAG_END_OF_TABLE :
                break;

            default:
                dsi_set_cmdq_V2(cmd, table[i].count, table[i].para_list, force_update);
        }
    }
} ? end push_table ?
```

dsi_set_cmdq_V2(cmd, count, *para_list, force_update)

```
static void lcm_init(void)
{
    SET_RESET_PIN(1);
    SET_RESET_PIN(0);
    MDELAY(1);
    SET_RESET_PIN(1);
    MDELAY(10);

    push_table(lcm_initialization_setting, sizeof(lcm_initialization_setting) / sizeof(struct LCM_setting_table), 1);
}
```

```
static struct LCM_setting_table lcm_initialization_setting[] = {

    /*
    Note :

    Data ID will depends on the following rule.

        count of parameters > 1     => Data ID = 0x39
        count of parameters = 1     => Data ID = 0x15
        count of parameters = 0     => Data ID = 0x05

    Structure Format :

    {DCS command, count of parameters, {parameter list}}
    {REGFLAG_DELAY, milliseconds of time, {}},

    ...

    Setting ending by predefined flag

    {REGFLAG_END_OF_TABLE, 0x00, {}}
    */
```

```
{0xB9,  3,  {0xFF, 0x83, 0x69}},
{REGFLAG_DELAY, 10, {}},

{0xB0,  2,  {0x01, 0x0B}},
{REGFLAG_DELAY, 10, {}},

{0xB2,  15, {0x00, 0x20, 0x05, 0x05,
             0x70, 0x00, 0xFF, 0x00,
             0x00, 0x00, 0x00, 0x03,
             0x03, 0x00, 0x01}},
{REGFLAG_DELAY, 10, {}},
```

6. GPIO Kernel Standard Usage

Display gpio DTS

Every item will represent a gpio mode

(alps\<kernel>\arch\<arm64>\boot\dts\<project>.dts)

```
27 &lcm {
28   compatible = "tpv,otm1901a";
29   gpio_lcd_rst = <&pio 45 0>;
30   gpio_lcd_pwr_en = <&pio 158 0>;
31   gpio_lcd_pwr2_en = <&pio 159 0>;
32   status = "okay";
33 };
34
```

If not 45/158/159 , please change it. Set correct compatible name

how to use in LCM Driver

```
static void lcm_suspend_power(void)
{
  SET_RESET_PIN(0);
  MDELAY(10);

  lcm_set_gpio_output(GPIO_LCD_PWR2_EN, GPIO_OUT_ZERO);
  MDELAY(20);

  lcm_set_gpio_output(GPIO_LCD_PWR_EN, GPIO_OUT_ZERO);
}

static void lcm_resume_power(void)
{
  lcm_set_gpio_output(GPIO_LCD_PWR_EN, GPIO_OUT_ONE);
  MDELAY(20);

  lcm_set_gpio_output(GPIO_LCD_PWR2_EN, GPIO_OUT_ONE);
  MDELAY(20);

  SET_RESET_PIN(1);
  MDELAY(20);
}
```

```
483 /* ------------------------------------------------------------------------- */
484 static void lcm_set_gpio_output(unsigned int GPIO, unsigned int output)
485 {
486 #ifdef BUILD_LK
487   mt_set_gpio_mode(GPIO, GPIO_MODE_00);
488   mt_set_gpio_dir(GPIO, GPIO_DIR_OUT);
489   mt_set_gpio_out(GPIO, output);
490 #else
491   gpio_set_value(GPIO, output);
492 #endif
493 }
```