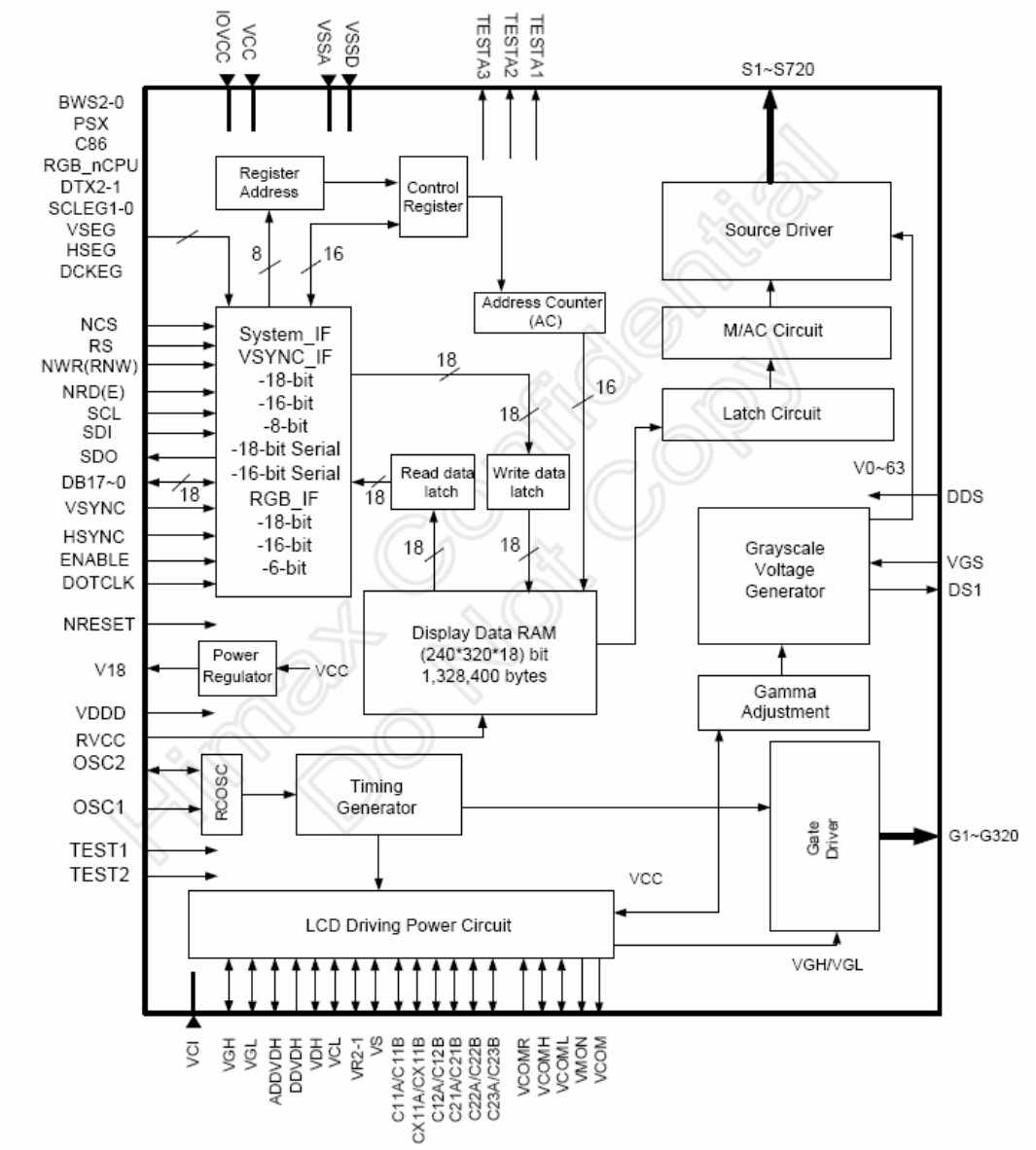
LCD Porting and LCD Interface

Date 2007.10.11

Agenda

- LCD与Baseband的连接
- LCD porting
- LCD interface

LCD block diagram（1/2）



LCD pin description

- 1. input parts
- 2. output parts
- 3. input/output parts
- 4. power parts
- 5. test pins and others
- 液晶显示的原理是液晶在不同电压的作用下会呈现出不同的光特性。
- LCD真彩显示使用TFT型LCD，主动点阵显示，需要采用源极驱动器（source driver）和栅极驱动器（gate driver）去控制LCD场效应晶体管FET的源极与栅极。源极驱动器接收显示数据驱动LCD列显示，也称为数据驱动器（data driver）栅极驱动器控制逐行扫描。

Baseband的LCD接法（1/5）

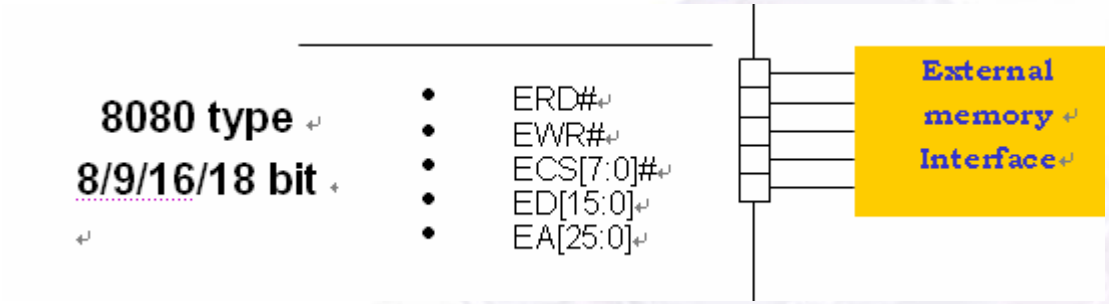
- BB 的LCM接法主要有NFI，EMI 和serial 三种接法：
- 1. 并行接法
- a. 专用NFI(nand flash interface)，支持8080总线接口。

Bus Type \ Signal Type	CS#	Reset#	RS#	RD#	WR#	NLD[17:0]
NFI	LPCE0# or LPCE1#	LRST#	LPA0	LRD#	LWR#	NLD[7:0] or NLD[8:0] or NLD[15:0] or NLD[17:0]

- **8 bit: NLD0~NLD7**
- **9 bit: NLD0~NLD8**
- **16 bit: NLD0~NLD15**
- **18 bit: NLD0~NLD17**

Baseband的LCD接法（2/5）

- b. 复用EMI，支持8080总线接口。



Bus Type	ECS7#	EA25	ERD#	EWR#	ED[15:0]
8080 series	CS#	A0	RD#	WR#	D[15:0]

Table 14 Configuration for LCD Parallel Interface

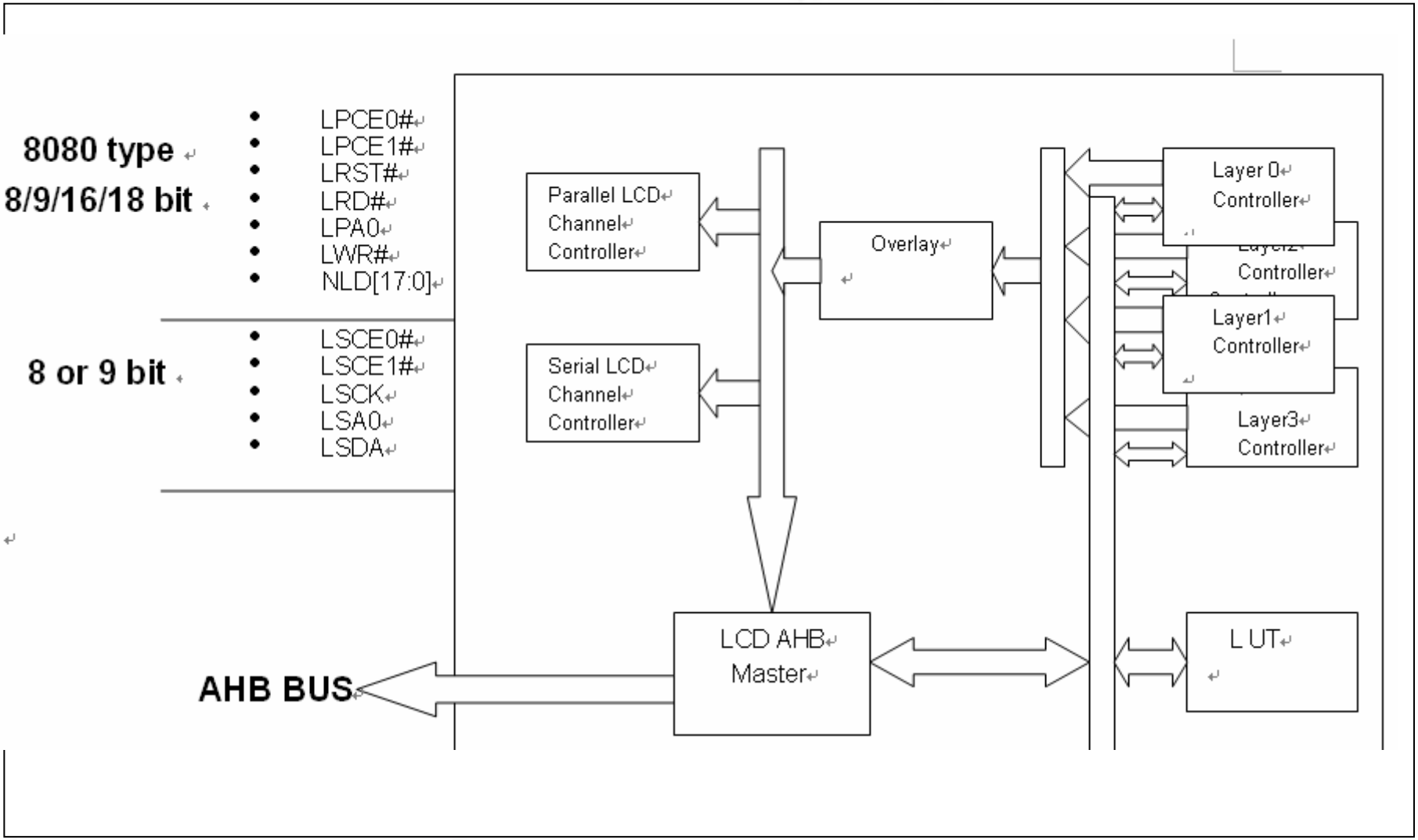
Baseband的LCD接法（3/5）

- 2. 串行接法
- 串行接口完成并行数据到串行数据的转换过程，支持8bit和9bit的传输方式：

- a．8bit串口需要 以下4根线来完成数据和命令的传输：LSCE#, LSDA,LSCK 和LSA0

- b．9bit串口需要以下3根线来完成数据和命令的传输： LSCE#, LSDA和LSCK

Baseband的LCD接法（4/5）



Baseband的LCD接法（5/5）

- 虽然这三种接口方式都可以选用，但从性能的角度考虑推荐使用NFI，它有如下优点：
- 有专用LCD controller 实现硬件加速
- 不用和memory复用EMI总线，从而Flash/SRAM 和LCD 都会有比较好的表现。

- LCD与Baseband的连接
- LCD porting
- LCD interface

前期准备

- 首先期望能有如下文档的源代码：
- Lcd_if.c
- lcd.c
- lcd_hw.h
- lcd_sw.h
- lcd_sw_inc.h
- lcd_sw_rnd.h
- custom_hw_default.c

custom_hw_default.c

- Open \custom\drv\misc_drv\[project]\custom_hw_default.c

Contrast level

```
/* Main LCD contrast level 1 ~ 15 */
126, 127, 128, 129, 130, 131, 132, 133, 134, 135,
136, 137, 138, 139, 140,
/* Main LCD Bias Param (Reserved) */
0, 0, 0, 0, 0,
/* Main LCD Linerate Param (Reserved) */
0, 0, 0, 0, 0,
/* Main LCD Temperature Param (Reserved) */
0, 0, 0, 0, 0,

/* Sub LCD contrast level 1 ~ 15 */
20, 22, 24, 26, 28, 30, 32, 34, 36, 38,
40, 42, 44, 46, 48,
/* Sub LCD Bias Param (Reserved) */
0, 0, 0, 0, 0,
/* Sub LCD Linerate Param (Reserved) */
0, 0, 0, 0, 0,
/* Sub LCD Temperature Param (Reserved) */
0, 0, 0, 0, 0,
```

Lcd_if.h(1/3)

```
typedef struct
{
  void (* Init)(kal_uint32 background, void **buf_addr);
  void (* PWRON)(kal_bool on);
  void (* BrightLevel)(kal_uint8 level);
  void (* SCREENON)(kal_bool on);
  void (* BlockWrite)(kal_uint16 startx,kal_uint16 starty,kal_uint16 endx,kal_uint16 endy);
  void (* GetSize)(kal_uint16 *out_LCD_width,kal_uint16 *out_LCD_height);
  void (* SleepIn)(void);
  void (* SleepOut)(void);
     void (* PartialOn) (kal_uint16 start_page,kal_uint16 end_page);
     void (* PartialOff) (void);
     kal_uint8 (*partial_display_align) (void);
  /*Engineering mode*/
  kal_uint8 (* get_param_number)(lcd_func_type type);
  void (* set_bias)(kal_uint8 *bias);
  void (* set_contrast)(kal_uint8 *contrast);
  void (* set_linerate)(kal_uint8 *linerate);
  void (* set_temp_compensate)(kal_uint8 *compensate);
} ? end {anonLCD_Funcs} ? LCD_Funcs;
```

Lcd_if.h(2/3)

- This function is to initialize MainLCD and SubLCD pointer of LCM. This function must be implemented for all LCMs. And all of the function pointer for LCD_Funs structure has to be assigned and implemented even it is as empty function.

Members	Description
Normal Mode	
Init	LCD initialize function
PWRON	LCD power on/off function
BrightLevel	Adjust LCD brightness
SCREENON	LCD screen on
BlockWrite	Block data write to LCD driver IC
GetSize	Obtain the size of the LCD panel
SleepIn	LCD enter sleep mode
SleepOut	LCD exit sleep mode
PartialOn	Start LCD partial display function
PartialOff	Stop LCD partial display function
partial_display_align	Return the partial display align line number
Engineering Mode	
get_param_number	Obtain the parameter number of the function
set_bias	Configure LCD bais
set_contrast	Configure LCD constrast
set_linerate	Configure LCD frame-refresh rate
set_temp_compensate	Configure LCD temperature compensation parameter
esd_check	Check the register of LCD driver IC to judge does the ESD corrupt it or not
set_rotate	Set LCM rotate function

Lcd_if.h(3/3)

- 在lcd_if.h中还定义了：
- LCD interface中寄存器地址映射
- 为操作寄存器中的位的bit mapping
- Macros for registers
- definition of LCM data outptu format
- MAIN_LCD_CMD_ADDR, MAIN_LCD_DATA_ADDR
- SUB_LCD_CMD_ADDR, SUB_LCD_DATA_ADDR

LCD driver porting procedure

- Step1:HW & build environment confirmation
- 确定硬件没有问题。
- 准备好Code Base。最好与新屏类似的，比如新屏是双彩的屏，如果Code Base本就是双彩的，可能会免去不少麻烦。
- Down入一个类似的Bin后，期望能有背光亮起来。否则先调背光。

Step 2: Study LCD spec & vender's init code

当确认（或基本确认）外部电路没有问题后，接下来一般可以先看LCD spec，同时对照着spec看供应商提供的init 程序，对当前芯片的操作模式有所了解。参照spec进行正确的连接。

- **Step3:Define the color & panel option in [project name]_GPRS.mak**
- **COM_DEFS_FOR_XXXX_LCM =**
  - COLOR_LCD ← MAIN LCD
  - COLOR_SUBLCD ← SUB LCD
  - DUAL_LCD ← TWO LCDS

```
# ********************************************************************************
COM_DEFS_FOR_T6219_MT6129D          = MT6129D_RF T6219_MT6129D
COM_DEFS_FOR_T6219_LCM       = COLOR_LCD T6219_LCM TFT_MAINLCD
                               #COLOR_LCD COLOR_SUBLCD DUAL_LCD

# ********************************************************************************

# ********************************************************************************
COM_DEFS_FOR_T6219_MT6129D          = MT6129D_RF T6219_MT6129D
COM_DEFS_FOR_T6219_LCM       = COLOR_LCD T6219_LCM TFT_MAINLCD|COLOR_SUBLCD DUAL_LCD
```

Step 4: Declare DUAL_LCD option or not

- 在**.mak里定义。
- 如果不是双屏，DUAL_LCD要拿掉，否则必须予以定义

Step5: Turn on the LCD_CMD_DMA_MODE option or not in LCD_SW.H

#define LCD_CMD_DMA_MODE

The LCD_CMD_DMA_MODE is the compiler option for LCD module command. If it is defined, the LCM command will be sent by LCD interface DMA. If the LCM with slower response time, the compiler option should not be defined to make sure the LCM can receive commands correctly.

Step 6: 在lcd_sw.h中确定Ctrl和Data的地址线，以及Command和Data的读写方式

These definitions declare the main/sub LCD command and data address port for hardware LCD interface to send command and data. These addresses should be defined according to the hardware configuration.

```
#define LCD_HD66773R_CTRL_ADDR    LCD_PARALLEL0_A0_LOW_ADDR
#define LCD_HD66773R_DATA_ADDR    LCD_PARALLEL0_A0_HIGH_ADDR

#define MAIN_LCD_OUTPUT_FORMAT    LCM_8BIT_16_BPP_RGB565_1

#define LCD_SSD1788_CTRL_ADDR     LCD_PARALLEL1_A0_LOW_ADDR
#define LCD_SSD1788_DATA_ADDR     LCD_PARALLEL1_A0_HIGH_ADDR

#define SUB_LCD_OUTPUT_FORMAT     LCM_8BIT_12_BPP_RGB444_1
```

Step 7: Define main/sub LCD output format

#define MAIN_LCD_OUTPUT_FORMAT LCM_8BIT_16_BPP_RGB565_1

#define SUB_LCD_OUTPUT_FORMAT LCM_8BIT_12_BPP_RGB444_1

LCD +0050h Region of Interest Window Control Register LCD_WROICON

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EN0	EN1	EN2	EN3							PERIOD					
Type	R/W	R/W	R/W	R/W							R/W					
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ENC	W2M	COMM_SEL	COMMAND					FORMAT							
Type	R/W	R/W	R/W	R/W					R/W							

```
/* definition of LCM data output format */
#define LCM_8BIT_8_BPP_RGB332_1      0x00    /* RRRGGBB */
#define LCM_8BIT_8_BPP_RGB332_2      0x01    /* BBGGGRR */
#define LCM_8BIT_12_BPP_RGB444_1     0x08    /* RRRRGGGG, BBBBRRRR, GGGGBBBB */
#define LCM_8BIT_12_BPP_RGB444_2     0x0B    /* GGGGRRRR, RRRRBBBB, BBBBGGGG */
#define LCM_8BIT_16_BPP_RGB565_1     0x10    /* RRRRRGGG, GGGBBBBB */
#define LCM_8BIT_16_BPP_RGB565_2     0x12    /* GGGBBBBB, RRRRRGGG */
#define LCM_8BIT_16_BPP_BGR565_1     0x11    /* BBBBBGGG, GGGRRRRR */
#define LCM_8BIT_16_BPP_BGR565_2     0x13    /* GGGRRRRR, BBBBBGGG */
#define LCM_8BIT_18_BPP_RGB666_1     0x18    /* RRRRRRXX, GGGGGGXX, BBBBBBXX */
#define LCM_8BIT_18_BPP_RGB666_2     0x1C    /* XXRRRRRR, XXGGGGGG, XXBBBBBB */
#define LCM_8BIT_24_BPP_RGB888_1     0x20    /* RRRRRRRR, GGGGGGGG, BBBBBBBB */
```

- Step8:**Implement**

  LCD_CtrlWrite_HD66773R(_data)

  LCD_DataWrite_HD66773R(_data) in lcd_sw.h

```
#define LCD_SEND_DMA_CMD(n) \
{ \
   while (LCD_IS_RUNNING) {};\
   DISABLE_ALL_LCD_LAYER_WINDOW;\
   DISABLE_LCD_TRANSFER_COMPLETE_INT;\
   SET_LCD_ROI_CTRL_NUMBER_OF_CMD(n);\
   ENABLE_LCD_ROI_CTRL_CMD_FIRST;\
   SET_LCD_ROI_WINDOW_SIZE(0,0);\
   START_LCD_TRANSFER;\
   while (LCD_IS_RUNNING) {};\
}
```

```
#ifdef LCD_16BIT_MODE
   #define LCD_CtrlWrite_HX8306A(_data) \
   {\
      SET_LCD_CMD_PARAMETER(0, LCD_CMD, _data);\
      LCD_SEND_DMA_CMD(1);\
   }

   #define LCD_DataWrite_HX8306A(_data) \
   {\
      SET_LCD_CMD_PARAMETER(0, LCD_DATA, _data);\
      LCD_SEND_DMA_CMD(1);\
   }
```

```
#define LCD_CtrlWrite_HX8306A_ESD(_data)  \
{\
   *(volatile kal_uint32 *) LCD_HX8306A_CTRL_ADDR = _data;\
   LCD_delay_HX8306A();\
}

#define LCD_DataWrite_HX8306A_ESD(_data) \
{\
   *(volatile kal_uint32 *) LCD_HX8306A_DATA_ADDR = _data;\
   LCD_delay_HX8306A();\
}
```

- while (LCD_IS_RUNNING);
- #define LCD_IS_RUNNING (REG_LCD_STA & LCD_STATUS_RUN_BIT)
- #define REG_LCD_STA *((volatile unsigned short *) (LCD_base+0x0000))
- #define LCD_STATUS_RUN_BIT 0x0001

LCD +0000h LCD Interface Status Register LCD_STA

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														CMD_CPEND	DATA_PEND	RUN
Type														R	R	R
Reset														0	0	0

RUN LCD Interface Running Status

- /* disable lcd frame transfer complete interrupt control */
- DISABLE_LCD_TRANSFER_COMPLETE_INT;\
- DISABLE_ALL_LCD_LAYER_WINDOW;\
- SET_LCD_ROI_CTRL_NUMBER_OF_CMD(n);\
- ENABLE_LCD_ROI_CTRL_CMD_FIRST;\
- #define DISABLE_ALL_LCD_LAYER_WINDOW REG_LCD_ROI_CTRL &= ~LCD_ROI_CTRL_LAYER_MASK;
- #define SET_LCD_ROI_CTRL_NUMBER_OF_CMD(n) REG_LCD_ROI_CTRL &= ~LCD_ROI_CTRL_CMD_NUMBER_MASK;\
  REG_LCD_ROI_CTRL |= ((n-1)<<7);
- #define ENABLE_LCD_ROI_CTRL_CMD_FIRST REG_LCD_ROI_CTRL |= LCD_ROI_CTRL_CMD_ENABLE_BIT;
- #define LCD_ROI_CTRL_LAYER_MASK 0xF0000000
- #define LCD_ROI_CTRL_CMD_NUMBER_MASK 0x00001F00
- #define LCD_ROI_CTRL_CMD_ENABLE_BIT 0x00008000

LCD +0050h Region of Interest Window Control Register LCD_WROICON

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EN0	EN1	EN2	EN3			PERIOD									
Type	R/W	R/W	R/W	R/W			R/W									
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ENC	W2M	COMM_SEL	COMMAND					FORMAT							
Type	R/W	R/W	R/W	R/W					R/W							

COMMAND Number of Commands to be sent to LCD module. Maximum is 31.

COMM_SEL Command Queue Selection, 0 for LCD_COMD0 , 1 for LCD_COMD1.

W2M Enable Data Address Increasing After Each Data Transfer

ENC Command Transfer Enable Control

PERIOD Waiting period between two consecutive transfers, effective for both data and command.

ENn Layer Window Enable Control

- DMA model CtrlWrite:

LCD +C400h~C47C LCD Interface Command/Parameter 0 Registers LCD_COMD0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name														C0	COMM[17:16]	
Type														R/W	R/W	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	COMM[15:0]															
Type	R/W															

COMM Command Data and Parameter Data for LCD Module

C0 Write to ROI Command Address if C0 = 1, otherwise write to ROI Data Address

Non DMA:

LCD_HX8306A_CTRL_ADDR

LCD_HX8306A_DATA_ADDR

把LPA0拉高，拉低。

- **Step9: Declare LCD_Funcs variable for MainLCD, SubLCD**

```
LCD_Funcs LCD_func_HD66773R = {
  LCD_Init_HD66773R,
  LCD_PWRON_HD66773R,
  LCD_SetContrast_HD66773R,
  LCD_ON_HD66773R,
  LCD_BlockWrite_HD66773R,
  LCD_Size_HD66773R,
  LCD_EnterSleep_HD66773R,
  LCD_ExitSleep_HD66773R,
  LCD_Partial_On_HD66773R,
  LCD_Partial_Off_HD66773R,
  LCD_Partial_line_HD66773R,
  /*Engineering mode*/
  LCD_GetParm_HD66773R,
  LCD_SetBias_HD66773R,
  LCD_Contrast_HD66773R,
  LCD_LineRate_HD66773R,
  LCD_Temp_Compensate_HD66773R
};

#ifdef DUAL_LCD
LCD_Funcs LCD_func_SSD1788 = {
  LCD_Init_SSD1788,
  LCD_PWRON_SSD1788,
  LCD_SetContrast_SSD1788,
  LCD_ON_SSD1788,
  LCD_BlockWrite_SSD1788,
  LCD_Size_SSD1788,
  LCD_EnterSleep_SSD1788,
```

Step10: Configure in LCD_FunConfig() in LCD.C

```
void LCD_FunConfig(void)
{
  MainLCD = &LCD_func_HD66773R;
  #ifdef DUAL_LCD
    SubLCD = &LCD_func_SSD1788;
  #endif
}
```

```
void (* Init)(kal_uint32 background, void **but
void (* PWRON)(kal_bool on);
void (* BrightLevel)(kal_uint8 level);
void (* SCREENON)(kal_bool on);
void (* BlockWrite)(kal_uint16 startx,kal_uint1
void (* GetSize)(kal_uint16 *out_LCD_width,ka
void (* SleepIn)(void);
void (* SleepOut)(void);
   void (* PartialOn) (kal_uint16 start_page,ka
   void (* PartialOff) (void);
   kal_uint8 (*partial_display_align) (void)
/*Engineering mode*/
kal_uint8 (* get_param_number)(lcd_func_
void (* set_bias)(kal_uint8 *bias);
void (* set_contrast)(kal_uint8 *contrast);
void (* set_linerate)(kal_uint8 *linerate);
void (* set_temp_compensate)(kal_uint8 *c
```

```
LCD_Funcs LCD_func_HD66773R = {
  LCD_Init_HD66773R,
  LCD_PWRON_HD66773R,
  LCD_SetContrast_HD66773R,
  LCD_ON_HD66773R,
  LCD_BlockWrite_HD66773R,
  LCD_Size_HD66773R,
  LCD_EnterSleep_HD66773R,
  LCD_ExitSleep_HD66773R,
  LCD_Partial_On_HD66773R,
  LCD_Partial_Off_HD66773R,
  LCD_Partial_line_HD66773R,
  /*Engineering mode*/
  LCD_GetParm_HD66773R,
  LCD_SetBias_HD66773R,
  LCD_Contrast_HD66773R,
  LCD_LineRate_HD66773R,
  LCD_Temp_Compensate_HD66773R
};
```

- Step11: Configure void init_lcd_interface(void)
- This function is used to configure the read/write timing of LCM. They should include the following items.
  - CS to WR setup time by use the macro SET_LCD_PARALLEL_CE2WR_SETUP_TIME(n). n<=3
  - CS to WR hold time by use the macro SET_LCD_PARALLEL_CE2WR_HOLD_TIME(n), n<=3
  - CS to RD setup time by use the macro SET_LCD_PARALLEL_CE2RD_SETUP_TIME(n), n<=3
  - Data write wait state period by use the macro SET_LCD_PARALLEL_WRITE_WAIT_STATE(n), n<=31
  - Data read latency period by use the macro SET_LCD_PARALLEL_READ_LATENCY_TIME(n), n<=31
  - Data read/write interval between two data read/write can be set by use macro SET_LCD_ROI_CTRL_CMD_LATENCY(n), n<=1023

- SET_LCD_PARALLEL_18BIT_DATA_BUS;

```
#if (defined(MT6219)||defined(MT6226)||defined(MT6226D)|
    SET_LCD_PARALLEL_CE2WR_SETUP_TIME((kal_uint32)1);
    SET_LCD_PARALLEL_CE2WR_HOLD_TIME(1);
    SET_LCD_PARALLEL_CE2RD_SETUP_TIME(3);
    SET_LCD_PARALLEL_WRITE_WAIT_STATE(2);
    SET_LCD_PARALLEL_READ_LATENCY_TIME(31);
    SET_LCD_ROI_CTRL_CMD_LATENCY(2);
    //SET_LCD_PARALLEL_CLOCK_52M

SET_GAMMA_TABLE(LCD_GAMMA_TABLE0, i, i);
SET_GAMMA_TABLE(LCD_GAMMA_TABLE0, 62, 61);
SET_GAMMA_TABLE(LCD_GAMMA_TABLE0, 63, 61);
SET_LCD_PARALLEL_GAMMA_R_TABLE(LCD_PARALLEL_GAMMA_TABLE0);
SET_LCD_PARALLEL_GAMMA_G_TABLE(LCD_PARALLEL_GAMMA_TABLE0);
SET_LCD_PARALLEL_GAMMA_B_TABLE(LCD_PARALLEL_GAMMA_TABLE0);
```

LCD +0018h LCD Parallel Interface Configuration Register 0 LCD_PCNF0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	C2WS		C2WH		C2RS				GAMMA_ID_R		GAMMA_ID_G		GAMMA_ID_B		DW	
Type	R/W		R/W		R/W				R/W		R/W		R/W		R/W	
	0		0		0				0		0		0		0	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	26M	13M			WST							RLT				
Type	R/W	R/W			R/W							R/W				
Reset	0	0			0							0				

LCD +0050h Region of Interest Window Control Register LCD_WROICON

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	EN0	EN1	EN2	EN3							PERIOD					
Type	R/W	R/W	R/W	R/W							R/W					
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ENC	W2M	COMM_SEL			COMMAND						FORMAT				
Type	R/W	R/W	R/W			R/W						R/W				

RLT Read Latency Time
WST Write Wait State Time
13M Enable 13MHz clock gating
26M Enable 26MHz clock gating
DW Data width of the parallel interface
- **00** 8-bit.
- **01** 9-bit
- **10** 16-bit
- **11** 18-bit

GAMMA_ID _R Gamma Correction LUT ID for Red Component
- **00** table 0
- **01** table 1
- **10** table 2
- **11** no table selected

GAMMA_ID_G Gamma correction LUT ID for Green Component
- **00** table 0
- **01** table 1
- **10** table 2
- **11** no table selected

GAMMA_ID_B Gamma correction LUT ID for Blue Component
- **00** table 0
- **01** table 1
- **10** table 2
- **11** no table selected

C2RS Chip Select (LPCE#) to Read Strobe (LRD#) Setup Time
C2WH Chip Select (LPCE#) to Write Strobe (LWR#) Hold Time
C2WS Chip Select (LPCE#) to Write Strobe (LWR#) Setup Time

- Step12:**Implement**

  LCD_Init_HD66773R()

  LCD_BlockWrite_HD66773R()

First, then you can see something on the LCD

BlockWrite is to move data from frame buffer to LCD module. For MT6205B, the frame buffer address should be get by calling get_lcd_frame_buffer_address() function that declares in lcd_if.c. For MT6218B and MT6219, if color LCM is used, the frame buffer address has been assigned in the LCD interface, LCM driver does not have to know the address of frame buffer. If mono LCM is provided in MT6218B/MT6219, LCD driver also need to get the frame buffer address by calling get_lcd_frame_buffer_address() function

- LCD与Baseband的连接
- LCD porting
- LCD interface

LCD controller(1/2)

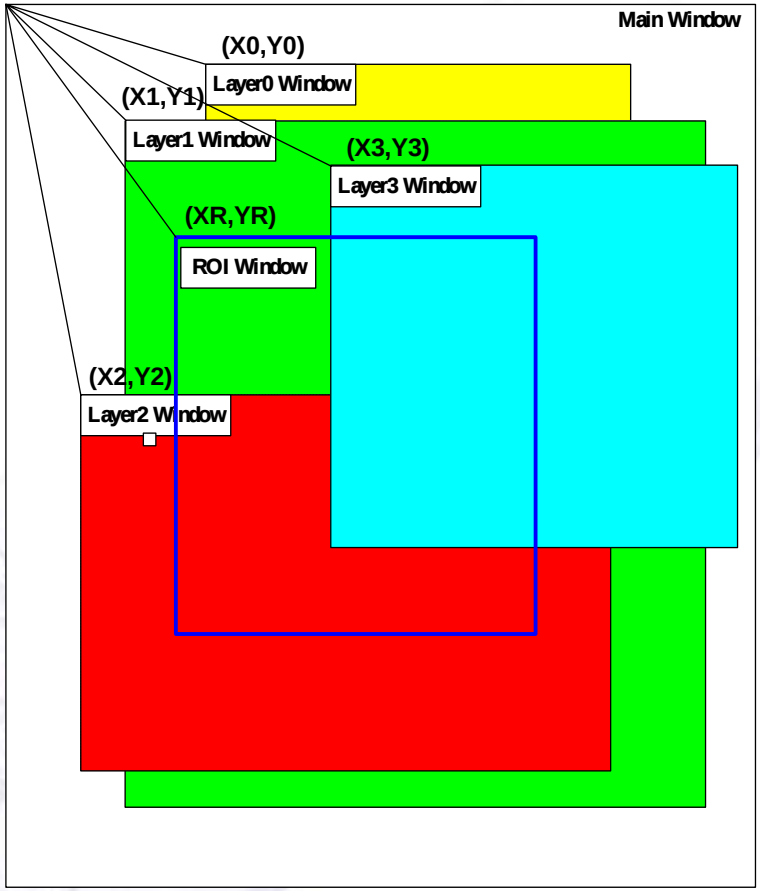
- 显示屏所显示的一幅完整画面就是一个帧(Frame)，其整个显示区域，在系统内会有一段存储空间与之对应，通过改变该存储空间的内容，从而改变显示屏的内容，该存储空间被称为Frame Buffer。显示屏上的每一点都必然与Frame Buffer里的某一位置对应。而计算机显示的颜色是通过RGB值来表示的，因此如果要在屏幕某一点显示某种颜色，则必须给出相应的RGB值。Frame Buffer就是用来存放整个显示的编码和像点值的外部存储器区域。帧缓冲器的每一个字节对应着LCD中的一个像素。
- Frame Buffer和LCD显示屏之间的数据传输很频繁，完全由CPU通过程序直接驱动显然不合适。因此，为减轻CPU的负担，在Frame Buffer与显示屏之间还需要一个中间件，该中间件负责从Frame Buffer里提取数据，进行处理，并传输到显示屏上。

LCD controller(2/2)

- LCD控制器的功能是显示驱动信号，进而驱动LCD显示器。在驱动LCD设计的过程中首要的是配置LCD控制器。在配置LCD控制器中最重要的一步则是帧缓冲区的指定。用户所要显示的内容皆是从缓冲区中读出，从而显示到屏幕上。帧缓冲区的大小由屏幕的分辨率和显示色彩数决定。
- LCD控制器用来传输图像数据并产生相应的控制信号,MT6226's lcd controller's features:

- Up to 320 x 240 resolution
- The internal frame buffer supports 8bpp indexed color and RGB 565 format.
- Supports 8-bpp (RGB332), 12-bpp (RGB444), 16-bpp (RGB565), 18-bit (RGB666) and 24-bit (RGB888) LCD modules.
- 4 Layers Overlay with individual color depth, window size, vertical and horizontal offset, source key, alpha value and display rotation control(90°,180°, 270°, mirror and mirror then 90°, 180° and 270°)
- One Color Look-Up Tables

Lcd layer(1/2)



Layer0 Opacity : 70%

Layer1 Opacity : 70%

Layer2 Opacity : 40%

Layer3 Opacity : 80%

- the LCD interface supports multi-layers frame buffer output to LCM simultaneously after overlaying the data of the specified layers. With the ability to overlay multiple layers of data, the LCD module can apply an opacity effect.
- 可以通过lcd_if.c中的
- kal_uint32 get_lcd_frame_buffer_address(void)来取得每个frame buffer的地址。

```
kal_uint32 get_lcd_frame_buffer_address(void)
{
#if (defined(MT6217)||defined(MT6218B)||defined(MT6219)||defined(MT6225)||defined(MT6226)||define
    switch (REG_LCD_ROI_CTRL&LCD_LAYER_MASK)
    {
        case LCD_LAYER0_ENABLE:
            lcd_frame_buffer_address=REG_LCD_LAYER0_BUFF_ADDR;
        break;
        case LCD_LAYER1_ENABLE:
            lcd_frame_buffer_address=REG_LCD_LAYER1_BUFF_ADDR;
        break;
        case LCD_LAYER2_ENABLE:
            lcd_frame_buffer_address=REG_LCD_LAYER2_BUFF_ADDR;
        break;
        case LCD_LAYER3_ENABLE:
            lcd_frame_buffer_address=REG_LCD_LAYER3_BUFF_ADDR;
        break;
    #if defined(MT6228)||defined(MT6229)||defined(MT6268T)||defined(MT6230)
        case LCD_LAYER4_ENABLE:
            lcd_frame_buffer_address=REG_LCD_LAYER4_BUFF_ADDR;
        break;
        case LCD_LAYER5_ENABLE:
            lcd_frame_buffer_address=REG_LCD_LAYER5_BUFF_ADDR;
        break;
        #endif
    } ? end switch REG_LCD_ROI_CTRL&LCD_... ?
#endif /* MT6217, MT6218B, MT6219 */
    return lcd_frame_buffer_address;
} ? end get_lcd_frame_buffer_address ?  /* get_lcd_frame_buffer_address */
```

Lcd interface

- The LCD interface provides a standard API for the upper layer AP to develop its image display. The LCD interface supports multi-layer frame buffer output to LCM simultaneously after overlaying the data of the specified layers.

Data structure

```
typedef struct
{
  kal_bool   layer_update_queue;    /* lcd layer parameter queue is full or not */
  kal_bool   source_key_enable;     /* enable/disable source key for specified layer */
  kal_bool   color_palette_enable;  /* enable/disable color palette for specified layer */
  kal_bool   opacity_enable;        /* enable/disable opacity for specified layer */
  kal_uint8  source_color_format;   /* color format of the specified layer, for MT6228 and MT6229 only */
  kal_uint8  color_palette_select;  /* selection of color palette table */
  kal_uint8  opacity_value;         /* opacity value for specified layer */
  kal_uint8  rotate_value;          /* rotate select for specified layer */
  kal_uint16 x_offset;              /* x axis offset from main window for specified layer */
  kal_uint16 y_offset;              /* y axis offset from main widnow for specified layer */
  kal_uint16 row_number;            /* layer buffer height of specified layer */
  kal_uint16 column_number;         /* layer buffer width of specified layer */
  kal_uint32 source_key;            /* source key color in RGB565 format for specified layer */
  kal_uint32 frame_buffer_address;  /* frame buffer start address of specified layer */
} lcd_layer_struct;
```

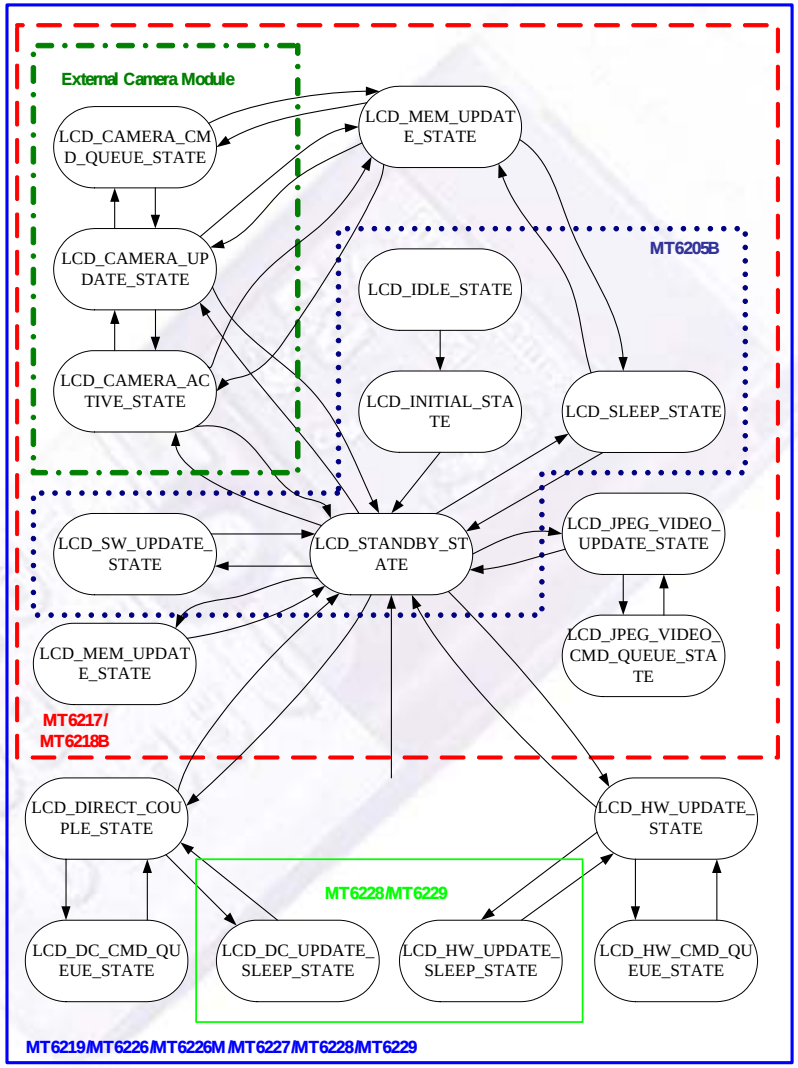
Data structure

```
typedef struct
{
  /// module ID that request frame buffer update
  /// LCD_UPDATE_MODULE_MMI, LCD_UPDATE_MODULE_MEDIA, LCD_UPDAET_MOD
  kal_uint8 module_id;
  /// which lcd will be updated (MAIN_LCD or SUB_LCD)
  kal_uint8 lcd_id;
  /// block caller or not
  kal_bool block_mode_flag;
  /// callback when lcd update is done
  void (* lcd_block_mode_cb)(void);
  //// frame buffer update mode (SW_TRIGGER, HW_TRIGGER or DIRECT_COUPLE)
  kal_uint8 fb_update_mode;
  /// the starting x coordinate of LCM to be updated
  kal_uint16 lcm_start_x;
  /// the starting y coordinate of LCM to be updated
  kal_uint16 lcm_start_y;
  /// the ending x coordinate of LCM to be updated
  kal_uint16 lcm_end_x;
  /// the ending y coordinate of LCM to be updated
  kal_uint16 lcm_end_y;
  /// the ROI window offset x from main window
  kal_uint16 roi_offset_x;
  /// the ROI window offset y from main window
  kal_uint16 roi_offset_y;
  /// the layers to be updated
  kal_uint32 update_layer;
  /// which layer will be applied by hw trigger or direct couple
  kal_uint32 hw_update_layer;
  /// rotate select for hardware update layer
  kal_uint8 rotate_value;

.....

} ? end {anonlcd_frame_update_struct} ?   lcd_frame_update_struct;
```

Data structure

```
typedef struct
{
  kal_uint8  fb_update_mode;     /* frame buffer update mode (SW_TRIGGER, HW_TRIGGER or DIRECT_COUPLE

  kal_uint8  block_mode;         /* block write out or not */
  kal_uint16 dest_block_width;   /* x pitch of block write operation */
  kal_uint32 dest_buffer_address; /* the start address of desination buffer for LCD memory write out */
  kal_uint32 dest_buffer_size;
  kal_uint16 roi_offset_x;       /* x offset of interest area from dest buffer */
  kal_uint16 roi_offset_y;       /* y offset of interest area from dest buffer */
  kal_uint16 roi_width;          /* dest image width */
  kal_uint16 roi_height;         /* dest image height */
  kal_uint32 update_layer;
  kal_uint32 hw_update_layer;
  /* for MT6228 and MT6229, MT6230 only */
  kal_uint8  hw_trigger_src;       /* LCD_HW_TRIGGER_IBW1 or LCD_HW_TRIGGER_IBW2 that will trigger LCD *
  kal_uint32 roi_background_color;  /* background color of memory otuput buffer */
  kal_uint8  memory_data_format;    /* output data format */
} lcd_frame_update_to_mem_struct;
```

State machine of lcd interface

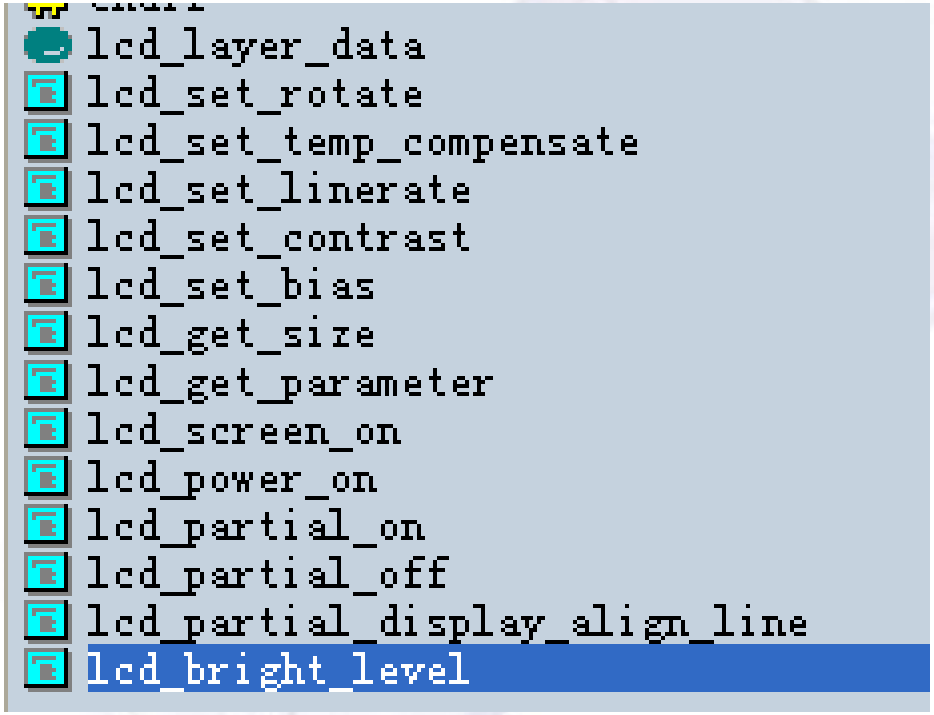


- **LCD_IDLE_STATE**: At the beginning, LCD operation stops at LCD_IDLE_STATE.
- **LCD_INITIAL_STATE**: The LCD is being initialized. When LCD initialization is complete, the state transits to LCD_STANDBY_STATE that is the temporary state of LCD state machines.
- **LCD_STANDBY_STATE**:
- **LCD_SW_UPDATE_STATE**: The LCD is busy with a software-triggered frame update process that is in progress. The state moves back to LCD_STANDBY_STATE when the data transfer is complete. The task that triggers the lcd software update is blocked until the data transfer is complete. After the frame is transferred complete, the state moves back to LCD_STANDBY_STATE.
- **LCD_SLEEP_STATE**: The LCD is in sleep mode, all backlights and LCD screen are turned off. The state moves back to LCD_STANDBY_STATE when lcd_sleep_out() function is called.

- **LCD_MEM_UPDATE_STATE**: This state is a temporary state. LCD operation returns to its source state after the data transfers completely.
- **LCD_CAMERA_ACTIVE_STATE**: In this state, the external camera module is activated and the LCD update procedure is performed by the external camera module. The state changes to LCD_CAMERA_UPDATE_STATE after the external camera starts image preview.
- **LCD_CAMERA_UPDATE_STATE**: In this state, the control of LCM updated transfers to the external camera driver.
- **LCD_CAMERA_CMD_QUEUE_STATE**
- **LCD_HW_UPDATE_STATE**
- **LCD_DC_UPDATE_STATE**
- **LCD_HW_CMD_QUEUE_STATE**
- **LCD_DC_CMD_QUEUE_STATE**
- **LCD_HW_UPDATE_SLEEP_STATE**
- **LCD_DC_UPDATE_SLEEP_STATE**
- **LCD_JPEG_VIDEO_UPDATE_STATE**
- **LCD_JPEG_VIDEO_CMD_QUEUE_STATE**

Lcd_lcm_if.c

- This function implements LCM interface adaptation layer



```
void lcd_set_bias(kal_uint8 lcd_id, kal_uint8 *bias)
{
    lcd_power_ctrl(KAL_TRUE);//lcd_power_up();
  switch (lcd_id)
  {
    case MAIN_LCD:
#if (defined(MT6217)||defined(MT6218B)||defined(MT6219)||defined(MT6
      DRV_WriteReg32(LCD_ROI_CMD_ADDR_REG,MAIN_LCD_CMD_ADDR);
      DRV_WriteReg32(LCD_ROI_DATA_ADDR_REG,MAIN_LCD_DATA_ADDR)
#endif /* MT6218B, MT6219 */
      MainLCD->set_bias(bias);
    break;
#ifdef DUAL_LCD
    case SUB_LCD:
     #if (defined(MT6217)||defined(MT6218B)||defined(MT6219)||defined
      DRV_WriteReg32(LCD_ROI_CMD_ADDR_REG,SUB_LCD_CMD_ADDR);
      DRV_WriteReg32(LCD_ROI_DATA_ADDR_REG,SUB_LCD_DATA_ADDR);
     #endif /* MT6218B, MT6219 */
      SubLCD->set_bias(bias);
    break;
#endif  /* DUAL_LCD */
    default:
      ASSERT(0);
    break;
  } ? end switch lcd_id ?
  lcd_power_ctrl(KAL_FALSE);//lcd_power_down();
} ? end lcd_set_bias ?   /* lcd_set_bias() */
```

LCD +0058h Region of Interest Window Command Start Address Register LCD_WROICADD

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ADDR															
Type	R/W															

ADDR ROI Window Command Address. Only writing to LCD modules is allowed.

LCD +005Ch Region of Interest Window Data Start Address Register LCD_WROIDADD

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	ADDR															
Type	R/W															

ADDR ROI Window Data Address Only writing to LCD modules is allowed.

lcd_sw_rnd.h

```
/* Color Transform */
#if (defined(MT6218B) || defined(MT6219)||defined(MT6217)||defined(MT6226)||defined(MT6227)||d
#define DRV_RGB_TO_HW(R,G,B)        ( (kal_uint16)(((B)&0xf8)>>3)|(((G)&0xfc)<<3)|(((R)&0xf8)<<
#define DRV_HW_TO_RGB_R(VALUE)      ( (kal_uint8)(((VALUE)&0xf800)>>8))
#define DRV_HW_TO_RGB_G(VALUE)      ( (kal_uint8)(((VALUE)&0x07e0)>>3))
#define DRV_HW_TO_RGB_B(VALUE)      ( (kal_uint8)(((VALUE)&0x001f)<<3))

#else /* Other than MT6218B */
#define DRV_RGB_TO_HW(R,G,B)        ( (((B)&0xf8)<<5)|(((R)&0xf8))|(((G)&0xe0)>>5)|(((G)&0x1c)
#define DRV_HW_TO_RGB_R(VALUE)      ( (kal_uint8)((VALUE)&0x00f8) )
#define DRV_HW_TO_RGB_G(VALUE)      ( (kal_uint8)((((VALUE)&0xe000)>>11)|(((VALUE)&0x0007)<<
#define DRV_HW_TO_RGB_B(VALUE)      ( (kal_uint8)(((VALUE)&0x1f00)>>5) )
#endif
```

- RGB565 16位
- 随便写一个16位数据的颜色数据 RGB = 10101101 10111001
- Blue:11001 green: 101 101 red:10101
- 我们得到了一个RGB值为21：45：25，就是这个颜色
  那么反过来，有了RGB的值我们该如何，因为RGB的有效位数都不足一个字节（8位），那我们舍弃掉低位数据，代码如下:

```
r = R & 0xF8;
g = G & 0xFC;
b = B & 0xF8;
high = r | (g<<5);
low = (g<<3) | (b>>3);
color= (high << 8) | low;
```

  这段代码在GUI程序中是有用的

LCD interface API

- /**
- * Configure the layer parameters for LCD multi-layer structure.
- * @param lcd_layer LCD_LAYER[0-5]
- * @param layer_data the parameters for lcd_layer
- * @return KAL_TRUE if successful
- */
- kal_bool config_lcd_layer_window(kal_uint8 lcd_layer,lcd_layer_struct *layer_data)

- /**
- * Configure the ROI window offset and size of LCD interface.
- * @param roi_x_offset ROI X offset
- * @param roi_y_offset ROI Y offset
- * @param roi_column ROI width
- * @param roi_row ROI height *
- * @return KAL_TRUE if sucessful
- */
- kal_bool config_lcd_roi_window(kal_uint16 roi_offset_x, kal_uint16 roi_offset_y, kal_uint16 roi_column, kal_uint16 roi_row)

- Config lcd roi window

```
    }
    else
    {
      if ((sub_lcd_operation_state!=LCD_STANDBY_STATE)&&
         (sub_lcd_operation_state!=LCD_SW_UPDATE_STATE)&&
         (sub_lcd_operation_state!=LCD_FW_UPDATE_STATE)&&
         #if (!defined(MT6225))
         (sub_lcd_operation_state!=LCD_HW_UPDATE_STATE)&&
         (sub_lcd_operation_state!=LCD_DC_UPDATE_STATE)&&
         #endif
         //#if (defined(MT6228)||defined(MT6229)||defined(MT6230))
         #ifdef TV_OUT_SUPPORT
         (main_lcd_operation_state!=LCD_DC_UPDATE_SLEEP_STATE)&&
         (main_lcd_operation_state!=LCD_HW_UPDATE_SLEEP_STATE)&&
         #endif
          (sub_lcd_operation_state!=LCD_SLEEP_STATE))
        ASSERT(0);
    }
     #endif /* DUAL_LCD */
     #endif
     lcd_power_ctrl(KAL_TRUE);//lcd_power_up();

  SET_LCD_ROI_WINDOW_OFFSET(roi_offset_x,roi_offset_y);
  SET_LCD_ROI_WINDOW_SIZE(roi_column,roi_row);

     lcd_power_ctrl(KAL_FALSE);//lcd_power_down();
```

```
void config_lcd_layer_offset(kal_uint8 lcd_layer, kal_uint16 layer_offset_x, kal_uint16 layer_offset_y)
{
#if (defined(MT6217)||defined(MT6218B)||defined(MT6219)||defined(MT6225)||defined(MT6226)||defined(
    switch (lcd_layer)
    {
        case LCD_LAYER0:
            SET_LCD_LAYER0_WINDOW_OFFSET(layer_offset_x,layer_offset_y);
        break;
        case LCD_LAYER1:
            SET_LCD_LAYER1_WINDOW_OFFSET(layer_offset_x,layer_offset_y);
        break;
        case LCD_LAYER2:
            SET_LCD_LAYER2_WINDOW_OFFSET(layer_offset_x,layer_offset_y);
        break;
        case LCD_LAYER3:
            SET_LCD_LAYER3_WINDOW_OFFSET(layer_offset_x,layer_offset_y);
        break;
    #if (defined(MT6228)||defined(MT6229)||defined(MT6268T)||defined(MT6230))
        case LCD_LAYER4:
            SET_LCD_LAYER4_WINDOW_OFFSET(layer_offset_x,layer_offset_y);
        break;
        case LCD_LAYER5:
            SET_LCD_LAYER5_WINDOW_OFFSET(layer_offset_x,layer_offset_y);
        break;
    #endif
    } ? end switch lcd_layer ?
#endif
} ? end config_lcd_layer_offset ?  /* config_lcd_layer_offset() */
```

```
kal_uint8 get_lcd_hw_layer_rotate_value(kal_uint32 hw_layer)
{
    kal_uint8 rotate_value=0;

    lcd_power_ctrl(KAL_TRUE);//lcd_power_up();
  switch (hw_layer)
    {
    case LCD_LAYER0_ENABLE:
        rotate_value=GET_LCD_LAYER0_ROTATE;
    break;
    case LCD_LAYER1_ENABLE:
            rotate_value=GET_LCD_LAYER1_ROTATE;
    break;
    case LCD_LAYER2_ENABLE:
      rotate_value=GET_LCD_LAYER2_ROTATE;
    break;
    case LCD_LAYER3_ENABLE:
      rotate_value=GET_LCD_LAYER3_ROTATE;
    break;
     #if (defined(MT6228)||defined(MT6229)||defined(MT6268T)||defined(MT6230))
     case LCD_LAYER4_ENABLE:
      rotate_value=GET_LCD_LAYER4_ROTATE;
    break;
     case LCD_LAYER5_ENABLE:
      rotate_value=GET_LCD_LAYER5_ROTATE;
    break;
     #endif
     } ? end switch hw_layer ?
     lcd_power_ctrl(KAL_FALSE);//lcd_power_down();

     return rotate_value;
} ? end get_lcd_hw_layer_rotate_value ?
```

LCD interface API

- /**
- * Initialize the LCD interface and LCD module.
- * @param lcd_id MAIN_LCD or SUB_LCD
- * @param background_color RGB565 color that LCM will use to initialize its frame buffer
- * @return None.
- */
- void lcd_init(kal_uint8 lcd_id, kal_uint16 background_color)

```
  switch (lcd_id)
  {
    case MAIN_LCD:
      main_lcd_operation_state=LCD_INITIAL_STATE;
      DRV_WriteReg32(LCD_ROI_CMD_ADDR_REG,MAIN_LCD_CMD_ADDR);
      DRV_WriteReg32(LCD_ROI_DATA_ADDR_REG,MAIN_LCD_DATA_ADDR);
      SET_LCD_ROI_CTRL_OUTPUT_FORMAT(MAIN_LCD_OUTPUT_FORMAT);
      MainLCD->Init(background_color,0);
      main_lcd_operation_state=LCD_STANDBY_STATE;
    break;
#ifdef DUAL_LCD
    case SUB_LCD:
      sub_lcd_operation_state=LCD_INITIAL_STATE;
      DRV_WriteReg32(LCD_ROI_CMD_ADDR_REG,SUB_LCD_CMD_ADDR);
      DRV_WriteReg32(LCD_ROI_DATA_ADDR_REG,SUB_LCD_DATA_ADDR);
      SET_LCD_ROI_CTRL_OUTPUT_FORMAT(SUB_LCD_OUTPUT_FORMAT);
      SubLCD->Init(background_color,0);
      sub_lcd_operation_state=LCD_STANDBY_STATE;
    break;
#endif  /* DUAL_LCD */
    default:
      ASSERT(0);
    break;
  } ? end switch lcd_id ?
```

LCD interface API

- /**
- * Trigger LCD interface to update the display RAM of LCM in specified area
- * @param lcd_para the parameters for lcd frame buffer update
- * @return None.
- */
- void lcd_fb_update(lcd_frame_update_struct *lcd_para)

- /**
- * Trigger LCD interface to output the MMI screen to one buffer
- * @param lcd_para the parameters for lcd frame buffer update to memory
- * @return None.
- */
- void lcd_fb_update_to_memory(lcd_frame_update_to_mem_struct *lcd_para)

LCD interface API

```
/**
 * Set the color palette of LCD interface.
 *
 * @param color_palette_select   LCD_COLOR_PALETTE[0-1]
 * @param color_palette_addr_ptr the address ptr that points to the color with offset
 *                               (start_index) from the 0th color in color palette
 * @param start_index            the offset from the 0th color in color palette
 * @param number_of_color        number of colors in color palette will be set.
 *
 * @return None.
 */
void set_lcd_color_palette(kal_uint8 color_palette_select,kal_uint32 *color_palette_addr_ptr,
                           kal_uint8 start_index, kal_uint8 number_of_color)
```

```
void set_lcd_color_palette(kal_uint8 color_palette_select,kal_uint32 *color_palette_addr_ptr,
                           kal_uint8 start_index, kal_uint8 number_of_color)
{
#if (defined(MT6217)||defined(MT6218B)||defined(MT6219)||defined(MT6225)||defined(MT6226)||
    kal_uint16 i;

    lcd_power_ctrl(KAL_TRUE);//lcd_power_up();
    #if (defined(MT6217)||defined(MT6218B)||defined(MT6219))
    if (color_palette_select==LCD_COLOR_PALETTE0)
    {
        for (i=start_index;i<(start_index+number_of_color);i++)
        {
            SET_LUT0_COLOR(i,*(color_palette_addr_ptr+i));
        }
    }
    else if (color_palette_select==LCD_COLOR_PALETTE1)
    {
        for (i=start_index;i<(start_index+number_of_color);i++)
        {
            SET_LUT1_COLOR(i,*(color_palette_addr_ptr+i));
        }
    }
    #elif (defined(MT6226)||defined(MT6225)||defined(MT6226M)||defined(MT6227)||defined(MT6
    for (i=start_index;i<(start_index+number_of_color);i++)
    {
        SET_LUT0_COLOR(i,*(color_palette_addr_ptr+i));
    }
    #endif
    lcd_power_ctrl(KAL_FALSE);//lcd_power_down();
#endif /* MT6218B, MT6219 */
} ? end set_lcd_color_palette ?     /* set_lcd_color_palette() */
```

LCD Frame Buffer Update Procedure

- Initialize the external LCM by calling the lcd_init() function.
- Configure the LCD interface of MT6218B or MT6219. Include each layer's parameters by calling the config_lcd_layer_window() function and the ROI parameters by calling the config_lcd_roi_window() function.
- Prepare the contents of each layer to display on the LCD screen.
- Update a frame by calling the lcd_fb_update() function.

Thank you!