DISP 设计方案

版本记录

Rev	Date	Author(s)	Remark
0.0	2021/11/15	刘翔	Display 初版完成
0.1	2022/3/8	刘翔	流片版本初稿完成
0.2	2022/4/14	刘翔	流片版本的修正版本，更新部分红字突出。

目录

图

表

1. DISPLAY 简介

Disp模块主要用于 chip 接外屏显示图像，模块的协议引脚与 io 口连接，从而直接操纵外屏。

模块可实现功能：

1.rgb传输协议，master function将储存在psram或share memory中图像数据，添加相应控制信号，传出至io到外屏显示。

2.intel 8080传输协议，添加120ms的reset初始化功能，支持i8080协议的cmd+param的输出格式。

3.yuv到rgb565的格式转换。可直接存储yuv数据在存储区，display模块可转换成rgb565直接输出。

两种模式不能同时开启，每次仅能使用一种功能。

2. DISPLAY 功能说明

Display 支持两种协议。RGB 协议和 Intel i8080 协议。

2.1.RGB 协议

RGB 接口又称 DPI（Display Pixel Iterface）接口，是并行接口，采用普通的同步，时钟和 信号线来传输数据。

其接口的数据线和控制线分离，因为屏幕内部没有 GRAM，所以协议数据速度快，成本 低。可以直接刷屏，通常用于大屏的驱动。

2.1.1. RGB 协议信号

RGB 协议其数据类型有 RGB565,RGB88,RGB666 等，颜色分量为红、绿、蓝三底色，通 过对三个颜色通道的变化，颜色的相互叠加得到各式各样的颜色。本模块采用的是 RGB565 数 据类型，其红色分量 R 占 5 比特，绿色分量 G 占 6 比特，蓝色分量 B 占 5 比特。其颜色的格 式图如图 1 所示：

图 1.rgb565 颜色格式

协议分为数据线和使能信号。如图 2 所示。

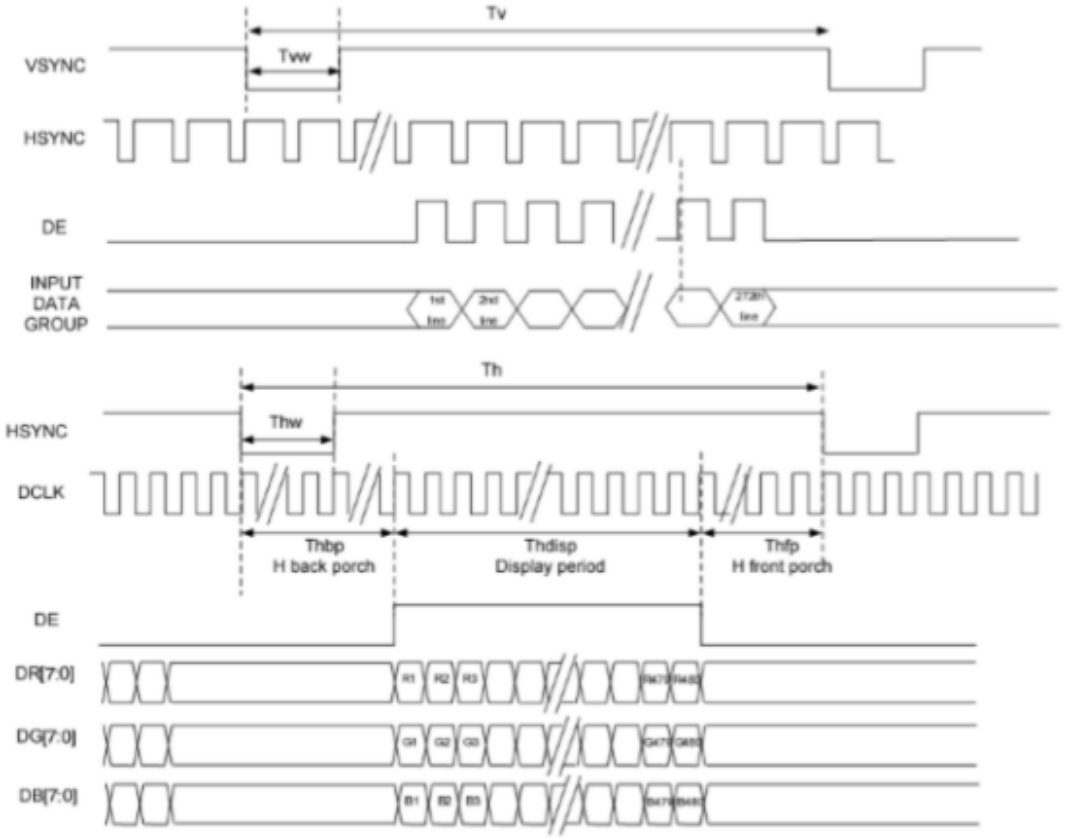


图 2.rgb 协议信号

数据线 R_data：5bit，G_data：6bit，B_data：5bit，共同组成数据线 Data：16bit。每 16bit 发送完成，一个像素数据完成。

信号线有 DCLK，DISP， HSYNC， VSYNC， DE 组成。具体描述如表 1 所示。

表 1，rgb565 信号

信号名	IO	描述
DCLK	output	输出给外屏的驱动时钟。基于当前使用的外屏，典型值为 9mhz。
DISP	output	屏幕开关使能。置 1 打开屏幕，置 0 关闭屏幕。
HSYNC	output	行同步信号。HSYNC 会在每行数据传输前置 1，在每行数据传输 结束置 0。
VSYNC	output	列同步信号，在一帧数据传输期间，VSYNC 会一直置 1，一帧传 输完成后。
DE	output	data_enable 信号，代表数据有效。其与 DATA 信号强相关，只要数据输出，此信号必须置高。
R_DATA	output	颜色数据红色分量
G_DAT	output	颜色数据绿色分量
B_DATA	output	颜色数据蓝色分量

协议运行顺序如下：

1. 输出 dclk 时钟，1 帧图像开始传输，图像分解成 m 行 n 列传输。

2. hysnc 每拉高一个 cycle，代表一行传输完成，单行的持续时钟数

h_time = h_back_porch+x-pixel+h_front_porch。

hsync 等待 thw 个时钟周期开始拉高（thw 典型值一般为 2），之后持续拉高 h_time – thw 个时钟周期。

3.vsync 每拉高一个 cycle，代表一帧画面传输完成。其 base cycle time 为 single hsync time，单帧的持续 cycle 数

v_time = v_back_porch + y_pixel + v_back_porch；

vsync 等到 tvw 个 cycle time 开始拉高，代表开始有图像数据传入，之后持续拉高 v_time – tvw 个 cycle time。

3.data_en 为指示数据传出信号，与 rgb_data 强相关，图像数据传出时，dat_on 持续置 1，无数据时拉低。

4.rgb_data 为图像数据，会在 hsync 处在 x_pixel 时间段内，vsync 处在 y_pixel 时间段内输出。

2.1.2. Rgb 功能实现

Rgb 模块的功能实现主要由以下几个模块，如图 3 所示：

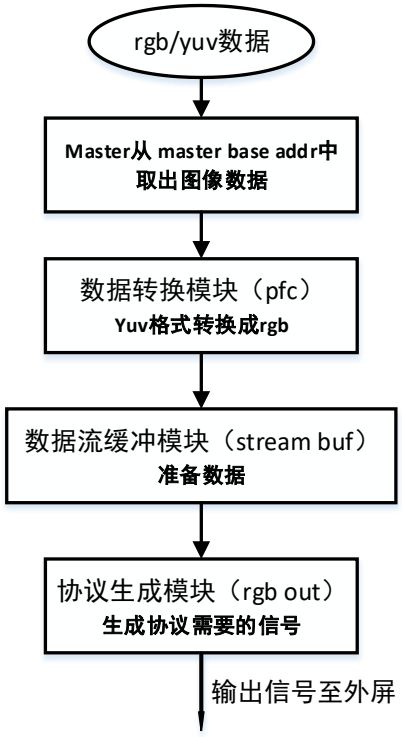


图 3.rgb 模块功能实现

模块运行输出如下：

1.display 内部的 ahb master 模块通过总线将存储区的图像取出。

2.经过格式转换模块，将 yuv 数据转换成 rgb565 数据。

3.rgb565 数据进入 stream buf 模块，缓冲总线和 display 的数据流速度。

4.最后过协议生成模块，从 stream buf 中取出图像数据，生成与数据相匹配的协议信号。

2.2.I8080 协议

I8080 协议是 intel 提出的 8080 总线标准，为串行协议。其控制简单方便，无需时钟和同 步信号，但屏幕内部通常需要 GRAM 保存每个 pixel 的图像数据，故很难做到大屏（3.8 以上）。

但其优点是若无图像数据继续传入，只要不断电，屏幕会一直保持最后一帧的图像， 不像 rgb 协议无图像传入会逐渐恢复白屏。

2.2.1. I8080 协议信号

I8080 总共有 5 根信号线，RESET、 CS、 RS、 WR、 RD 和 DB，其中 DB 为 8 位并行数据线。

其信号描述如表 2 所示。

表 2.i8080 协议信号

信号名	IO	描述
RESET	output	屏幕的 reset 信号，在启动和初始化时需要置低。
CS	output	片选信号，传输中置低。
RS	output	屏幕数据/命令指示信号。高为数据输出，低为令输出。
WR	output	读使能，数据在上升保持稳定。
RD	output	读使能 ，此功能不需要。
DB[7:0]	output	数据输出。

I8080 协议其数据传输是通过 cmd+parameter 的格式进行的。通过 cmd + param 的这种方式，实现对屏幕的初始化，模块功能的配置，图像数据传输。

RS 信号指示当前数据是 command 还是 parameter。写数据功能时，DB 线需要 在 WR 的上升沿保持稳定，CS 信号需要在传输期间保持置低。RESET 在传输期间保持为高电平，所有的信号默认为高电平。

I8080 其进行 cmd + param 的传输时序如图 4 所示：

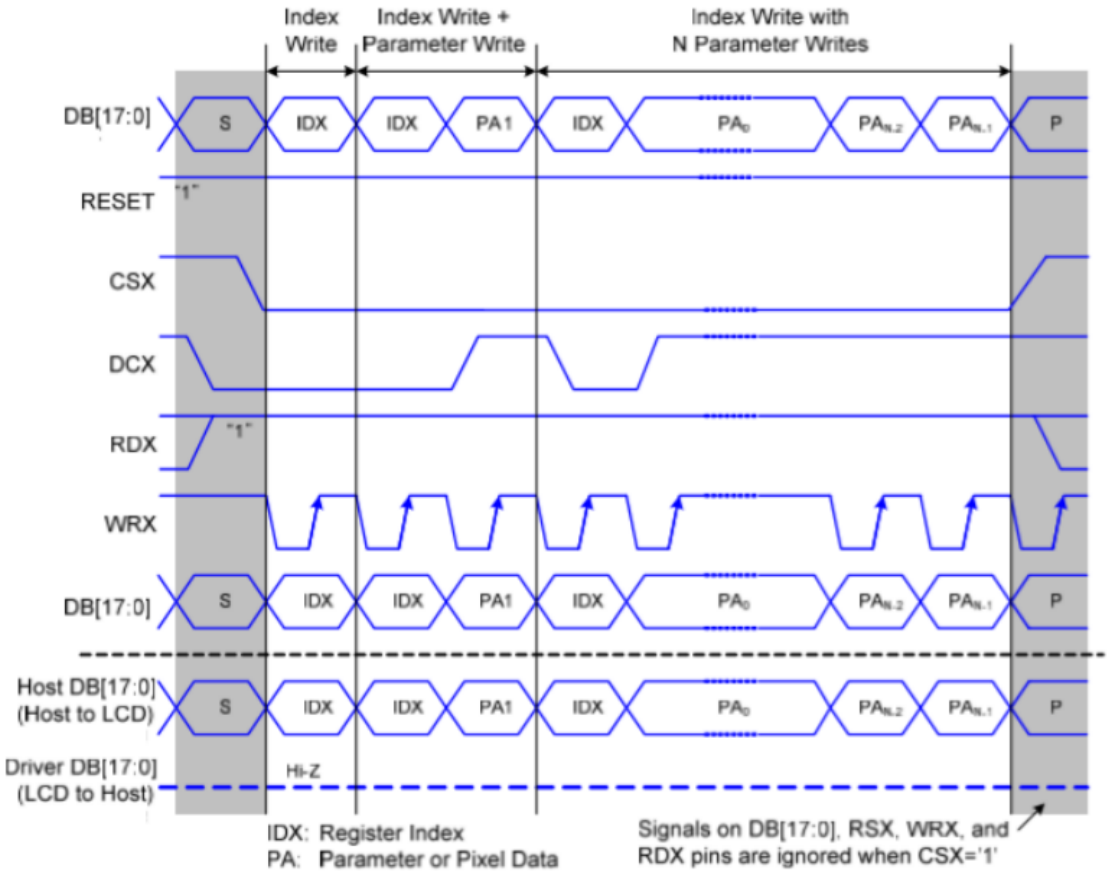


图 4.i8080 传输时序

Reset 在传输时置 1，发送 db 数据时，csx 置低，wrx 拉低。

如果是 command 数据，dcx(rs)置低，如果是 data 数据，dcx（rs）置高，在数据传输期 间，所有信号信号保持稳定。

wrx 信号在数据 hold 的中间时刻置 1，数据在 wrx 上升沿写入屏幕。如图 5 所示：

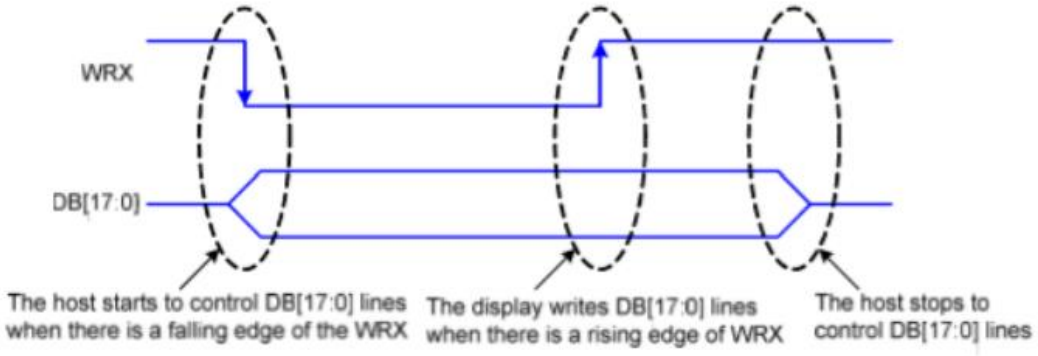


图 5.db 在 wrx 的边沿写入

2.2.2. I8080 功能实现

I8080 主要由以下几个模块实现，如图 6 所示。

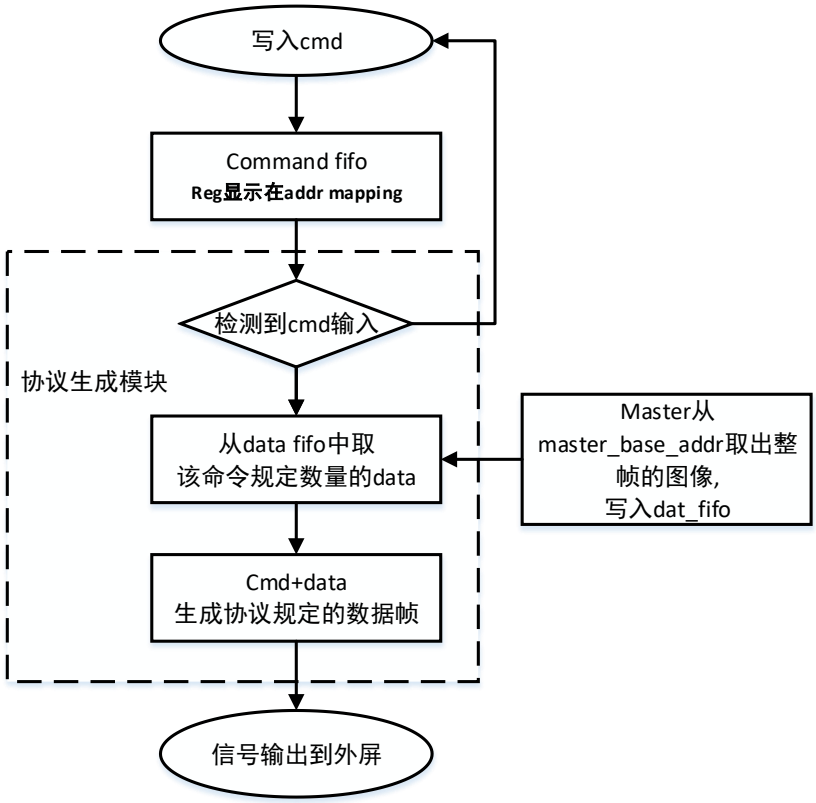


图 6.8080 模块实现过程

屏幕初始化：

设置当前 cmd 需要的 param 的个数。

在 cmdfifo 中将 cmd 写入，cmd buf 会缓存 cmd 命令，等待协议生成模块使用。

在 datfifo 中将 param 写入，datbuf 会缓存 dat 的 param，等待协议生成模块使用。

检测到 cmdfifo 中有 command，协议生成模块取出 cmd ，从 dat fifo 取出相应个数的 param，生成相应的协议时序，传输完成信号置 1。输出到外屏。

传输图像：

在 cmd fifo 中将命令写入 cmd fifo，cmd buf 会缓存 cmd 命令，等待协议生成模块使用。

Data_fifo：master 会将 base_addr 地址上的图像数据直接搬运到 dat_fifo 中，dat_buf 会缓存 dat 数据，等到协议生成模块使 用。

协议生成模块：检测到 cmd_buf 有数据后，根据 cmdbuf 的 command，确认 data_buf 的数 据长度。并将此 command 和其对应数据长度的 data 按照 i8080 协议，生成相对应的时序，直 接输出。此数据流完成后，再次取 cmdbuf 的 command，根据 cmd 确认其 databuf 中的数据长 度，取出相应

长度数据，生成相应时序，如此往复输出。

3. 寄存器与软件编写

3.1.寄存器列表

Disp 的寄存器列表如表 3 所示。红色的为较 v0 版本新加的寄存器。

表 3.disp address mapping

	映射基址	0x48060000			
	分序号	**寄存器地址 [bit]**	**控制信号名**	**默认值**	**支持操作**
1	display_int	0x0[31:29]	reserved	0x0	r/w
		0x0[28]	disconti_mode	0x0	r/w
		0x0[27:13]	reserved	0x0	r/w
		0x0[12]	Soft_reset	0x0	w
		0x0[11: 8]	reserved	0x0	r/w
		0x0[7]	i8080_eof	0x0	r/w
		0x0[6]	i8080_sof	0x0	r/w
		0x0[5]	rgb_eof	0x0	r/w
		0x0[4]	rgb_sof	0x0	r/w
		0x0[3: 2]	i8080_int_en	0x0	r/w
		0x0[1: 0]	rgb_int_en	0x0	r/w
2	rgb_cfg	0x1[31:27]	reserved	0x0	r/w
		0x1[26]	lcd_display_on	0x0	r/w
		0x1[25]	rgb_on	0x0	r/w
		0x1[24]	rgb_disp_on	0x0	r/w
		0x1[23]	reserved	0x0	r/w
		0x1[22:12]	y_pixel	d'272	r/w
		0x1[11]	dclk_rev	0x0	r/w
		0x1[10: 0]	x_pixel	d'480	r/w
3	No use	0x2[31: 0]	NC	NC	NC
4	Sync_config	0x3[31]	reserved	0x0	r/w
		0x3[30:28]	yuv_sel	0x0	r/w
		0x3[27]	reserved	0x0	r/w
		0x3[26:20]	vsync_front_porch	d'8	r/w
		0x3[19:15]	vsync_back_porch	d'8	r/w
		0x3[14: 8]	hsync_front_porch	d'5	r/w

		0x3[7: 0]	hsync_back_porch	d'40	r/w
5	i8080_config	0x4[31:21]	reserved	0x0	r/w
		0x4[20:12]	i8080_1ms_count	0x0	r/w
		0x4[11:10]	reserved	0x0	r/w
		0x4[9: 8]	tik_cnt	0x0	r/w
		0x4[7: 6]	reserved	0x0	r/w
		0x4[5]	reset_sleep_in	0x0	r/w
		0x4[4: 3]	reserved	0x0	r/w
		0x4[2]	i8080_fifo_mode	0x0	r/w
		0x4[1]	i8080_dat_on	0x0	r/w
		0x4[0]	i8080_disp_en	0x0	r/w
6	8080_cmd_fifo	0x5[31:16]	reserved	0x0	r/w
		0x5[15: 0]	i8080_cmd_fifo	0x0	w
7	8080_dat_fifo	0x6[31: 0]	i8080_dat_fifo	0x0	w
8	8080thrd	0x7[31:24]	cmd_rd_thrd	0x0	r/w
		0x7[15: 8]	cmd_wr_thrd	0x0	r/w
9	status_rd	0x8[10: 0]	rgb_ver_cnt	0x0	r
		0x8[11]	cmd_cfg_done	0x0	r
		0x8[22:12]	i8080_ver_cnt	0x0	r
		0x8[23]	Disp_fifo_empty	0x0	r
		0x8[24]	Disp_fifo_near_full	0x0	r
		0x8[31:25]	reserved	0x0	r/w
10	rgb_sync_low	0x9[5: 0]	hsync_back_low	0x0	r/w
		0x9[13: 8]	vsync_back_low	0x0	r/w
		0x9[16]	pfc_pixel_reve	0x0	r/w
11	rgb_clum_offset	0xa[10: 0]	partial_offset_clum_l	0x0	r/w
		0xa[26:16]	partial_offset_clum_r	0x0	r/w
		0xa[28]	partial_area_ena	0x0	r/w
12	rgb_line_offset	0xb[10: 0]	partial_offset_line_l	0x0	r/w
		0xb[26:16]	partial_offset_line_r	0x0	r/w

13	dat_fifo_thrd	0xc[3: 0]	i8080_cmd_para_count	0x0	r/w
		0xc[12: 4]	dat_wr_thrd	0x0	r/w
		0xc[24:16]	dat_rd_thrd	0x0	r/w
14	mater_rd_base_addr	0xd[31: 0]	master-rd_base_adr	0x0	r/w

(1) reg0：display int 寄存器

Reg0 寄存器主要是配置和查询中断用的。总共有 4 个中断，两种类型，sof 和 eof 中断。

Sof 中断，在一帧开始时触发，开始数据传输时，中断触发，写 1 清 0。Eof 中断，一帧数据写 完后中断触发，写 1 清 0。所有的中断开启后每一帧都会触发。具体如下：

Reg0[1:0] rgb_int_en:

Reg0[0],rgb_sof 中断使能，1 为 enable；

Reg0[1],rgb_eof 中断使能，1 为 enable.

Reg0[3:2] i8080_int_en:

Reg0[2], i8080_sof 中断使能，1 位 enable；

Reg0[3],i8080_eof 中断使能，1 为 enable。

Reg0[4] rgb_sof:

rgb_sof 中断使能，rgb 模块的 start of frame 中断位，写 1 清 0。

Reg0[5] rgb_eof:

rgb_eof 中断使能，rgb 模块的 end of frame 中断位，写 1 清 0。

Reg0[6] i8080_sof:

I8080_sof 中断使能，8080 模块的 start of frame 中断位，写 1 清 0。

Reg0[7] i8080_eof:

I8080_eof 中断使能，8080 模块的 end of frame 中断位，写 1 清 0。

Reg0[12]soft_reset：

Disp 模块的软复位，在模块功能异常的时候，关闭模块使能，将此位置 1 即可复位。

Reg0[28] disconti_mode:

disconti_mode:此功能是 防止因模块速度过快导致数据读出不及时。 默认置 1，需要设置。改变了寄存器位置。

(2) reg1：status 寄存器。

Reg1 寄存器主要是一些 rgb 和 8080 都需要配置的寄存器，以及系统初始化的寄存器，外 加一些 rgb 的功能开关。具体如下：

Reg1[10:0] x_pixel:

Rgb 和 8080 的行像素设置。按分辨率直接填写。

Reg1[11] dclk_rev rgb ：

数据输出沿选择，沿 dclk 的上升沿还是下降沿输出。0：posedge；1： negedge。

Reg1[22:12] y_pixel：

Rgb 和 8080 的列像素设置。按分辨率直接填写。

Reg1[23] str_fifo_clr：（已去掉不用）

Reg1[24] rgb_disp_on

Rgb module 使能，1：rgb enable。0：rgb disable。

Reg1[25] rgb_on

Rgb module IO 口选择，打开模块输出即为 rgb 输出。1：rgb output。0：i8080 output。

Reg1[26] lcd_display_on：

Rgb output signal. 直接输出端口，置 1 即为打开屏幕，在数据传输前要打开屏幕。

Reg1[31:27] rgb_clk_div：（已去掉不用）

(4) reg3：hsync&vsync config。

reg3[7:0] hsync_back_porch hsync：

在一行数据发送前需要提前时间，以 rgbclk 为基准，典型值 40’d。

reg3[14:8] hsync_front_porch：

hsync 在一行数据发送完成后需要 hold 住的时间，以 rgbclk 为基准，典型值 5。

reg3[19:15] vsync_back_porch：

vsync 在一帧数据发送之前需要提前的时间，以 hsync 为基准单位，典型值为 8。

reg3[26:20] vsync_front_porch：

vysnc 在一帧数据发送后需要 hold 住的时间，以 hsync 为基准单位，典型值为 8。

reg3[30:28] yuv_sel：

rgb 的输入选择，输入可以 rgb565 数据，也可以为 yuv 数据。

0：orginal rgb565data；1：orginal yuyv data；2：uyvy data；

3：yyuv data；4：uvyy data； 5：vuyy data。

(5) Reg4：i8080 config。

Reg4[0] i8080_disp_en：

模块使能信号，0：disable；1：enable。

Reg4[1] i8080_dat_on ：

I8080 模块初始化完成后，置 dat_on 为 1。

Reg4[2] i8080_fifo_mode：

I8080fifo 的写入模式。

0：在 i8080_disp_en 为低的时候，向 fifo 中写入数据无效。

1：任何时候向 fifo 中写入数据都有效。

Reg4[3] i8080_fifo_clr：（已去掉不用）

Reg4[4] i8080_cmdfifo_clr：（已去掉不用）

Reg4[5] reset_sleep_in I8080：

command 初始化前需要 reset 信号拉低 10ms，等待 120ms 稳定在初始化。

置 1 后将 reset 拉低等待 120ms。

Reg4[9:8] tik_cnt：

I8080 的数据持续时间，以 i8080clk 为基准。

0：8clk；1：6clk；2：4clk；3： 2clk。

Reg4[20:12] i8080_1ms_count：

进行 reset 初始化时的 1ms 计数器计数值。需要结合实际时钟判断。

实际 count = i8080_1ms_count * 1000。

(6) reg5：cmd_fifo。

reg5[15:0] cmd fifo：

command fifo。所有 command 需要写到这里。

(7) reg6：dat_fifo。

reg6[31:0] dat_fifo。（扩位到了 32bit）

Dat_fifo。发送完 command 后，需要向 dat fifo 写入所需 data。

(8) reg7：i8080 fifo thrd。

reg7[15:8] cmd_wr_thrd：

cmd fifo 的写门限，超过多少 fifo 将不接受 dma 传来的数据。典型值：96。

reg7[31:24] cmd_rd_thrd：

cmd fifo 的读门限，超过多少将无法读取 fifo 的数据。典型值：0。

(9)reg8：disp status。

Reg8[10:0] rgb_ver_cnt：

Rgb 模块的行计数器。在 rgb 运行中可以看到 rgb 的行进度。

Reg8[11] cmd_cfg_done：

I8080 模块的 cmd 屏幕初始化 config 完成标志位。

初始化屏幕期间，每发送一次 cmd+data，都需要读到此信号拉高，等待发送完成。

Reg8[22:12] i8080_ver_cnt：

I8080 模块的行计数器。在 I8080 运行中可以看到 I8080 的行进度。

Reg8[23] disp_fifo_empty：

disp 的数据 fifo 为空信号。读到 1 时，fifo 内没有数据。

Reg8[23] disp_fifo_near_full：

disp 的数据 fifo 快满信号。读到 1 时，fifo 内数据即将溢出。

(10)reg9：rgb_sync_low。

Reg9[5:0]hsync_back_low：

Hsync 在帧开始时拉低的时间，使用典型值，默认为 2。

Reg9[13:8]vsync_back_low：

Hsync 在帧开始时拉低的时间，使用典型值，默认为 2。

Reg9[16]：pfc_pixel_reve：

按 Pixel（16bit）进行翻转。防止写入数据高低位冲突。

(11)rega：disp_clum_offset

Rega[10:0]partial_offset_clum_l：

区域显示功能的行左侧偏移量。不能大于 partial_offset_clum_r。

Rega[26:16] partial_offset_clum_r：

区域显示功能的行右侧偏移量。不能小于 partial_offset_clum_l。

Rega[28] partial_area_ena：

区域显示功能使能，此功能的作用是将整幅图像的部分显示在屏幕上。

设置完成后，会将指定行列的数据按标准协议送出。

(12)regb: disp_line_offset

Regb[10:0] partial_offset_line_l：

区域显示功能的列上侧偏移量。不能大于 partial_offset_line_r。

Regb[26:16] partial_offset_line_r：

区域显示功能的列 xia 侧偏移量。不能小于 partial_offset_line_l。

(13)regc：dat_fifo_thrd

Regc[4:0] i8080_cmd_param_count：

I8080 模块进行 cmd+data 设置时，一个 cmd 命令后跟的 param 的个数。

在写 cmd fifo 和 dat fifo 前先设置 param 的个数。

Regc[16:8] dat_wr_thrd：

Dat fifo 将要写满的阈值，可使用默认值 128。Fifo 最大深度的 25%.

Regc[28:20] dat_rd_thrd：

Dat fifo 即将读空的阈值，可使用默认值 384。Fifo 最大深度的 75%。

(14)regd：master_rd_base_addr

Reg[31:0] master_rd_base_addr：

display 模块从存储取数据的 base address。

可设置为 share mem 和 psram 还有 flash 内的地址。

3.2.软件编写

bk7256 的 disp 部分的编写顺序如下。其 rgb 的驱动编写流程如图 7 所示。

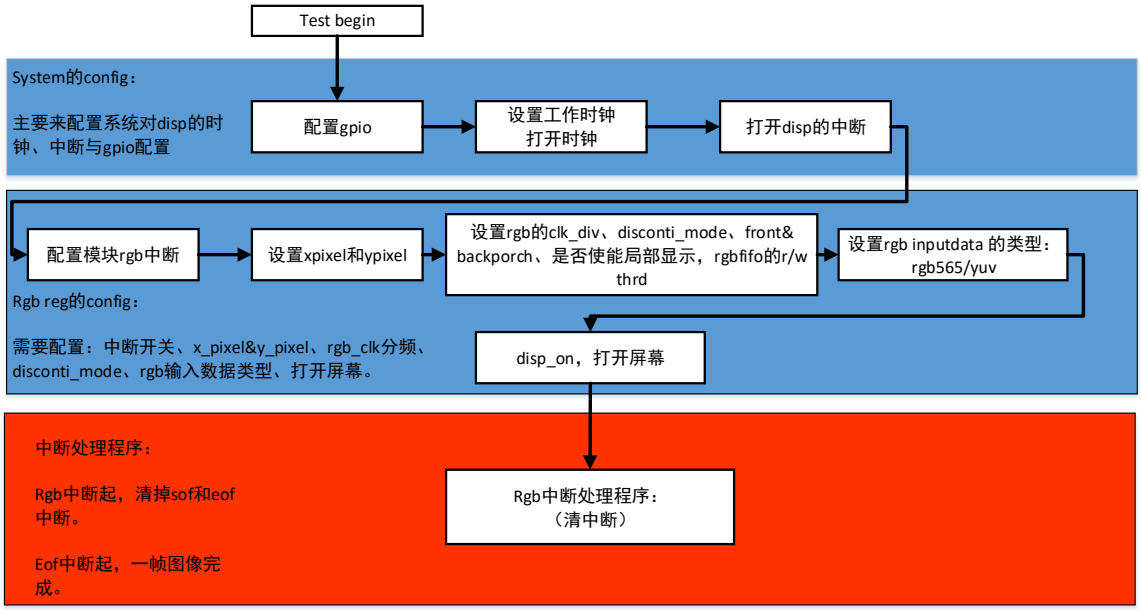


图 7.rgb 的驱动编写流程

具体的步骤：

1.对 system 的配置：

打开 gpio，配置 sys 的工作时钟，打开 display 模块的系统中断。

2.模块的配置：

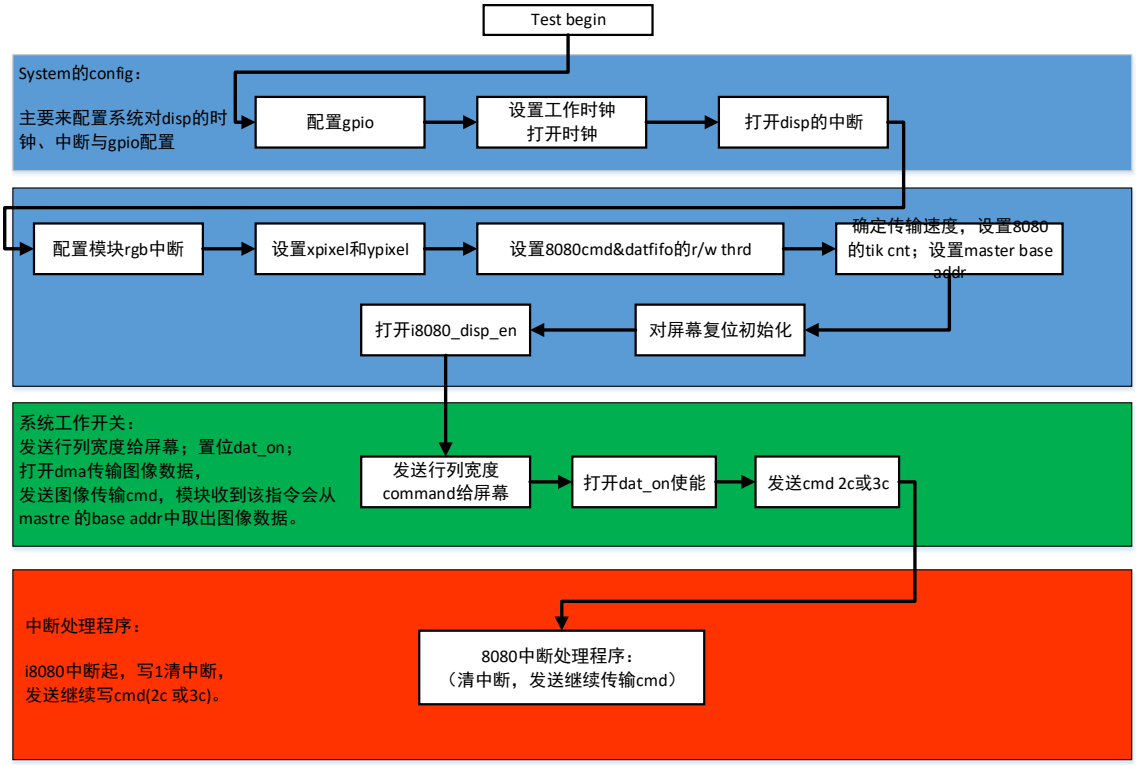
配置 sof eof 中断使能，设置行列像素，设置 discontimode，vsync 和 hsync 的前后等待数，yuv 模式选择，设置 display 模块取数据的 base addr。

3.打开 rgb ena 信号，开始传输一帧数据。

4.进入中断程序，sof eof 中断，进行相关的操作。

i8080 的驱动编写流程如图 8 所示。

图 8.i8080 驱动流程



具体的步骤：

1. System 配置：

系统的配置主要有打开 disp 的中断使能，设置 disp 的时钟，配置模块的 IO 口。

2. Reg 配置

Disp module 的配置主要有：中断开关、x_pixel&y_pixel、8080fifo 的 r/w 门限、设置 tik_cnt 确定传输速度，以及以及 master base addr。

3.进行屏幕的初始化：

具体的步骤是，向 i8080_cmd_param_cout 写入该指令的参数量，向 cmd fifo 写 cmd，向 dat fifo 写入所有的 param。在 rd_status 寄存器读取 cfg_done_d 寄存器直到为 1。

接着依次写入其他寄存器。

例如写入 cmd：e8， param: 40，8a，00，01。

1)其中其中 param 的参数为 4 个。

2)设置 reg4 的 i8080_cmd_param_count =4;

3)向 reg5 cmd fifo 写入 e8.

4）向 reg6 dat fifo 写入 40，8a，00，01。

5）查询 reg8 的 cmf_cfg_done_d 为 1。

6）继续进行下一次的寄存器操作。

4.传输图像。

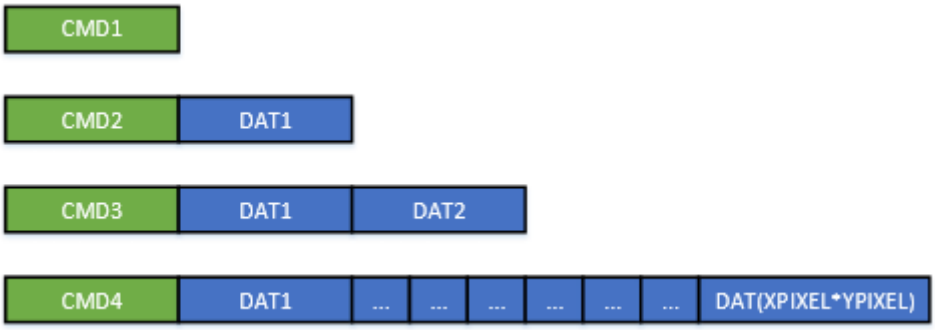
dat on 寄存器置 1，Cmd fifo 寄存器写入 2c，开始传输整帧图像。

5.中断处理程序。

处理中断，eof 中断后，发送 3c 继续传输。或者 dat_on 置 0，配置屏幕后，frame_on 置 1，发送 3c 传输下一次数据。

3.3.I8080 的数据传输格式

I8080 的数据传输格式是一个 cmd 后跟特定个数的 data。如图 9 所示。



其 cmd 后跟的 dat 个数，由 cmd 命令所控制。

如开始图像传输 cmd（例 CMD4），其后跟 屏幕像素个数个 dat（xpixel*y_pixel 个）。如打开屏幕 cmd(例 cmd1)，其后不需要跟 data，所以 data 个数为 0。又比如电压调节 cmd（例 cmd3），其后跟两个 data。后跟多少 数据由外接屏幕的内部寄存器来决定。

所以发送帧图像需要先发送开始传输 cmd，之后再发帧图像 data。

I8080_cmd_param_count 这个寄存器就是设置 cmd 后面跟的 data 的个数。

3.4.关于屏幕局部显示功能。

屏幕的局部显示主要是为了覆盖不规则分辨率的屏幕，以降低 sensor 产生非常用分辨率的 yuv

数据时，产生坏帧的风险。

在使用局部显示时，需要配置的寄存器有 x_pixel，y_pixel，partial_area_ena，x_partial_offset_l，x_partial_offset_r，y_partial_offset_l，y_partial_offset_r。

x_pixel，y_pixel：被剪切前的图像的原始分辨率，方便不同分辨率复用，保留了原始的分辨率设置。

x_pixely_pixel：局部显示的使能位。

x_partial_offdet_l：局部显示的目标区域的行的起始位置。

x_partial_offset_r：局部显示的目标区域的行的结束位置。

y_partial_offset_l：局部显示的目标区域的列的起始位置。

y_partial_offset_r：局部显示的目标区域的列的结束位置。

以从 640x480 分辨率剪裁出 480x272 分辨率为示例，各个参与寄存器的作用的简单示意图如图 10 所示：

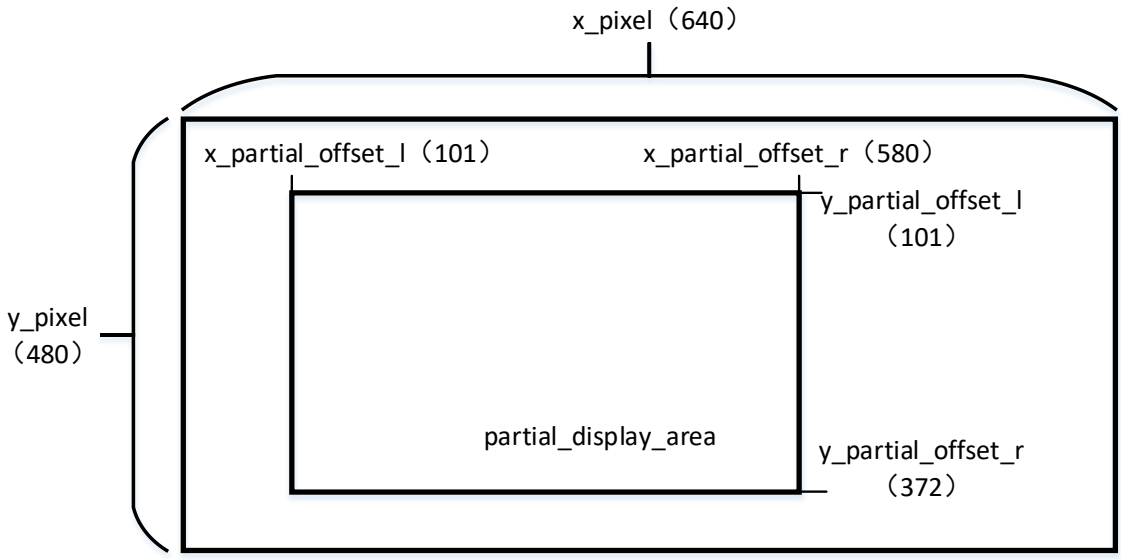


图 10.各寄存器的简单示意图