骁客 X91 GPU 核心处理模块核心板

用户手册

目录

1. X91 GPU 核心处理模块简介

1.1 功能概述

X91 GPU 核心处理模块提供了两组独立的 RGB888 输出信号，能够产生纯色，灰阶，彩条，Crosstalk，棋盘格等 Pattern，也支持读取 TF/SD 卡的 24-bit BMP 图片并输出到 RGB 接口上。配合 RGB 转 MIPI/LVDS/EDP 等桥接芯片，可驱动不同接口类型的显示面板，被广泛应用于 LCD/AMOLED 模组测试行业。

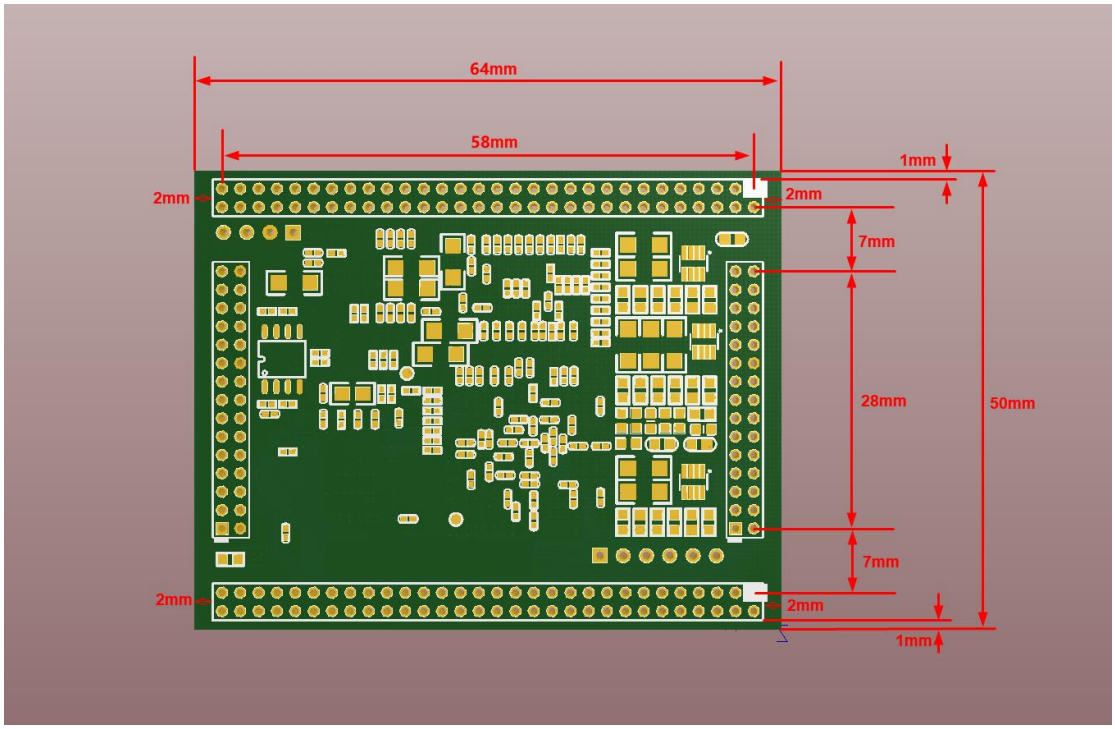
特性	描述
输出接口	RGB888：x2 组 PCLK 频率：0~200MHz
图片存储	SD 卡，支持 FAT32 文件系统 容量：<=128GB
显存	DDR3 Memory 容量：128M*16bit 性能：1333Mbps (带宽：21.328Gbps)
二次开发	MCU：STM32F103VCT6 参考代码提供以下功能： 1. 配置 RGB 输出时序 2. 配置 RGB 显示画面（显示图片，Patterns，画点，显示点阵字符） 3. GPIO 函数（GPU 的 GPIO 输入输出控制，见信号定义表）
上位机通信接口	USB 接口（USB 转 TTL 串口）
尺寸	64*50mm

1.2 尺寸

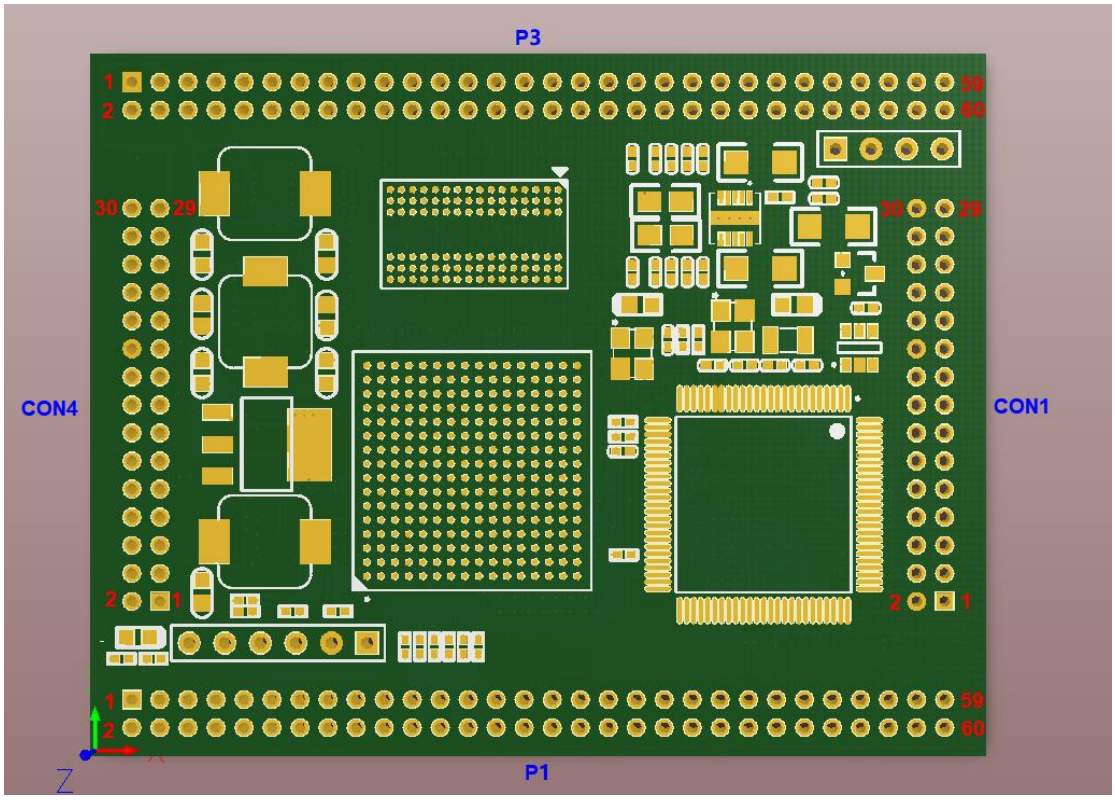
上下两排连接器：30*2，2.00mm Pitch 插针
左右两排连接器：15*2，2.00mm Pitch 插针

图 1：Top 层视图

Bottom 层视图



Top 层视图



1.3 信号定义

颜色标识说明：

	电源
	GND
	GPU IO 扩展（可以通过 STM32 函数访问）
	STM32 IO
	NC（不连接）

P3 接口：

引脚	信号	引脚	信号
1	5V（核心板电源输入）	2	GND
3	**LCD_R[7]/DB[23] _A**	**4**	**LCD_R[6]/DB[22] _A**
5	**LCD_R[5]/DB[21] _A**	**6**	**LCD_R[4]/DB[20] _A**
7	**LCD_R[3]/DB[19] _A**	**8**	**LCD_R[2]/DB[18] _A**
9	**LCD_G[7]/DB[15] _A**	**10**	**LCD_G[6]/DB[14] _A**
11	**LCD_G[5]/DB[13] _A**	**12**	**LCD_G[4]/DB[12] _A**
13	**LCD_G[3]/DB[11] _A**	**14**	**LCD_G[2]/DB[10] _A**
15	**LCD_B[7]/DB[07] _A**	**16**	**LCD_B[6]/DB[06] _A**
17	**LCD_B[5]/DB[05] _A**	**18**	**LCD_B[4]/DB[04] _A**
19	**LCD_B[3]/DB[03] _A**	**20**	**LCD_B[2]/DB[02] _A**
21	**LCD_DE_A**	22	NC
23	**LCD_VS_A**	**24**	**LCD_HS_A**
25	**LCD_PCLK_A**	26	MCU_PB1
27	NC	28	NC
29	NC	30	MCU_PB2（不能连接上拉电阻）
31	MCU_PA11	32	NC
33	MCU_PA12	34	NC
35	MCU_PC4	36	MCU_PC3
37	MCU_PC2	38	MCU_PC1
39	MCU_PC0	40	MCU_PC5
41	NC	42	NC
43	MCU_PB15	44	MCU_PB13
45	MCU_PB12	46	MCU_PE6
47	MCU_PB14	48	NC
49	NC	50	NC
51	NC	52	MCU_PB0
53	MCU_PA8	54	MCU_PC13
55	MCU_PC6	56	MCU_PC14
57	MCU_PC7	58	MCU_PC15（上拉 10K 电阻）
59	MCU_PB6	60	MCU_PB7

P1 接口：

引脚	信号	引脚	信号
1	MCU_Boot0（保持悬空）	2	NC
3	MCU_NRST（保持悬空）	4	MCU_RTC_BAT
5	NC	6	NC
7	MCU_PA9（GPU/STM32 升级接口）	8	MCU_PA10（GPU/STM32 升级接口）
9	MCU_PB10	10	MCU_PB11
11	NC	12	GPIO_P1_Pin12
13	GPIO_P1_Pin13	14	GPIO_P1_Pin14
15	MCU_PB15	16	MCU_PB14
17	MCU_PB13	18	MCU_PD3
19	GPIO_P1_Pin19	20	GPIO_P1_Pin20（不能连接下拉电阻或者 GND）
21	SDCard_CLK0_ MCU_PC12 连接 SD/TF 卡	22	SDCard_CMD0_ MCU_PD2 连接 SD/TF 卡
23	MCU_PD6	24	MCU_PD3
25	SDCard_DATA0_ MCU_PC8 连接 SD/TF 卡	26	SDCard_DATA1_ MCU_PC9 连接 SD/TF 卡
27	SDCard_DATA2_ MCU_PC10 连接 SD/TF 卡	28	SDCard_DATA3_ MCU_PC11 连接 SD/TF 卡
29	MCU_PB8	30	MCU_PB14
31	MCU_PB13	32	MCU_PB15
33	MCU_PB9	34	GPIO_P1_Pin34
35	GPIO_P1_Pin35	36	GPIO_P1_Pin36
37	GPIO_P1_Pin37	38	GPIO_P1_Pin38
39	GPIO_P1_Pin39	40	GPIO_P1_Pin40
41	GPIO_P1_Pin41	42	GPIO_P1_Pin42
43	GPIO_P1_Pin43	44	GPIO_P1_Pin44
45	GPIO_P1_Pin45	46	GPIO_P1_Pin46
47	GPIO_P1_Pin47	48	GPIO_P1_Pin48
49	GPIO_P1_Pin49	50	GPIO_P1_Pin50
51	GPIO_P1_Pin51	52	GPIO_P1_Pin52
53	GPIO_P1_Pin53	54	GPIO_P1_Pin54
55	GPIO_P1_Pin55	56	GPIO_P1_Pin56
57	GPIO_P1_Pin57	58	GPIO_P1_Pin58
59	**LCD_DE_B**	**60**	**LCD_PCLK_B**

CON4 接口：

引脚	信号	引脚	信号
1	MCU_PB9	2	MCU_PB8
3	MCU_PA15	4	MCU_PE3
5	MCU_PB3	6	MCU_PE1
7	GPIO_CON4_Pin7	8	GPIO_CON4_Pin8
9	GPIO_CON4_Pin9	10	GPIO_CON4_Pin10
11	GPIO_CON4_Pin11	12	GPIO_CON4_Pin12
13	MCU_PE2	14	MCU_PE0
15	MCU_PE5	16	MCU_PE4
17	NC	18	NC
19	NC	20	GND
21	GPIO_CON4_Pin21	22	MCU_PB5
23	GPIO_CON4_Pin23	24	MCU_PB4
25	**LCD_B[0]/DB[00] _A**	**26**	**LCD_B[1]/DB[01] _A**
27	**LCD_G[0]/DB[08] _A**	**28**	**LCD_G[1]/DB[09] _A**
29	**LCD_R[0]/DB[16] _A**	**30**	**LCD_R[1]/DB[17] _A**

CON1 接口：

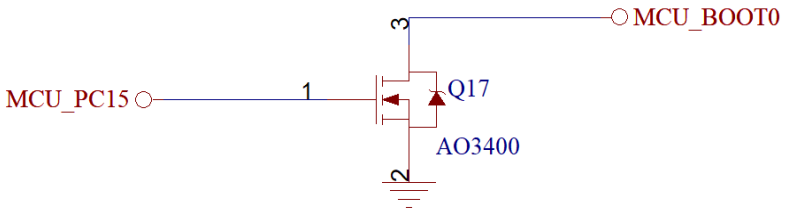
引脚	信号	引脚	信号
1	NC（保持悬空）	2	GND
3	**LCD_B[0]/DB[00] _B**	**4**	**LCD_B[1]/DB[01] _B**
5	**LCD_B[2]/DB[02] _B**	**6**	**LCD_B[3]/DB[03] _B**
7	**LCD_B[4]/DB[04] _B**	**8**	**LCD_B[5]/DB[05] _B**
9	**LCD_B[6]/DB[06] _B**	**10**	**LCD_B[7]/DB[07] _B**
11	**LCD_G[0]/DB[08] _B**	**12**	**LCD_G[1]/DB[09] _B**
13	**LCD_G[2]/DB[10] _B**	**14**	**LCD_G[3]/DB[11] _B**
15	**LCD_G[4]/DB[12] _B**	**16**	**LCD_G[5]/DB[13] _B**
17	**LCD_G[6]/DB[14] _B**	**18**	**LCD_G[7]/DB[15] _B**
19	**LCD_R[0]/DB[16] _B**	20	GND
21	**LCD_R[2]/DB[18] _B**	**22**	**LCD_R[1]/DB[17] _B**
23	**LCD_R[4]/DB[20] _B**	**24**	**LCD_R[3]/DB[19] _B**
25	**LCD_R[6]/DB[22] _B**	**26**	**LCD_R[5]/DB[21] _B**
27	**LCD_HS_B**	**28**	**LCD_R[7]/DB[23] _B**
29	**LCD_VS_B**	30	GND

备注：

1） P1/P3/CON1/CON4 存在复用的引脚：PB8，PB9，PB13，PB14，PB15，PD3

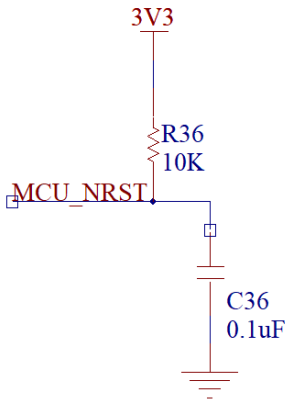
2） STM32 Boot1 对应 PB2 在外部不能有上拉（在核心板上连接了 10K 下拉电阻）；

3） STM32 Boot0 及相关的 PC15：PC15 需要外接上拉，以保证正常工作时 Boot0 为低电平；



4） P1.Pin20 不能接下拉电阻或者 GND（核心板连接了上拉电阻），这个引脚在上电时，需要保持高电平；

5） MCU_NRST：核心板连接了 RC 复位电路

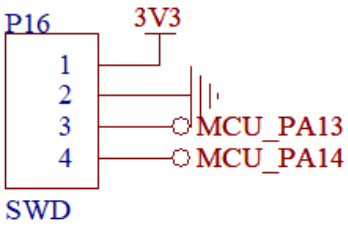


6） 时钟：STM32 的 Pin12（OSC_IN），Pin13（OSC_OUT）连接了 16MHz 晶振

7） STM32 在线调试接口：

MCU_PA13：SWDIO

MCU_PA14：SWCLK



2. STM32 函数说明

2.1 概述

提供了以下基本功能：

1） 配置 RGB 输出时序

2） 显示图片

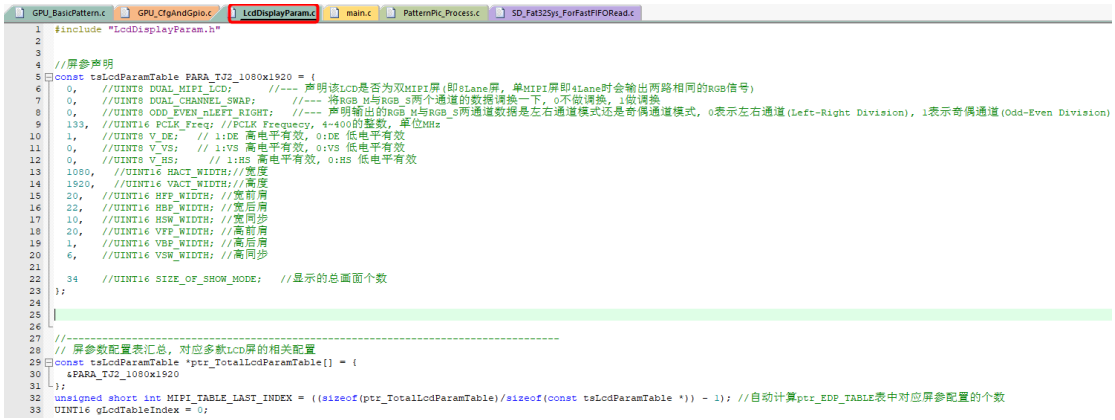
3） 显示 Pattern

4） GPIO 输入输出函数

2.2 功能介绍

2.2.1 修改 RGB 输出时序

修改 LcdDisplayParam.c



用户可以按照以上格式增加多款屏参，然后调用 void SetLcdParamTableIndex(UINT16 Val)函数选择实际使用哪种屏参。

2.2.2 显示内容设置

修改 PatternPic_Process.c

GPU_BasicPattern.c | GPU_CfgAndGpio.c | LcdDisplayParam.c | main.c | PatternPic_Process.c | SD_Fat32Sys_ForFastFIFORead.c

```c
#include "GPU_BasicPattern.h"
#include "SD_Fat32Sys_ForFastFIFORead.h"


void Lcd_ShowMode(UINT8 ShowId) {
  switch(ShowId) {
    case 1:   Lcd_ShowPureColor(0xff, 0x00, 0x00);              break; //(R,G,B)红
    case 2:   Lcd_ShowPureColor(0x00, 0xff, 0x00);              break; //(R,G,B)绿
    case 3:   Lcd_ShowPureColor(0x00, 0x00, 0xff);              break; //(R,G,B)蓝
    case 4:   Lcd_ShowPureColor(0x00, 0x00, 0x00);              break; //(R,G,B)黑
    case 5:   Lcd_ShowPureColor(0xff, 0xff, 0xff);              break; //(R,G,B)白
    case 6:   Lcd_ShowPureColor(0x80, 0x80, 0x80);              break; //(R,G,B)灰, 128Gray
    case 7:   op_PatternColorOutline(255,255,255);              break; //边框, 黑底白框
    case 8:   op_PatternGrayScaleH(0, 255);                     break; //Gray_H Level 255
    case 9:   op_PatternGrayScaleH(1, 255);                     break; //Gray_H Level 255 INV
    case 10:  op_PatternGrayScaleH(0, 127);                     break; //Gray_H Level 127
    case 11:  op_PatternGrayScaleH(1, 127);                     break; //Gray_H Level 127 INV
    case 12:  op_PatternGrayScaleV(0, 255);                     break; //Gray_V Level 255
    case 13:  op_PatternGrayScaleV(1, 255);                     break; //Gray_V Level 255 INV
    case 14:  op_PatternGrayScaleV(0, 127);                     break; //Gray_V Level 127
    case 15:  op_PatternGrayScaleV(1, 127);                     break; //Gray_V Level 127 INV
    case 16:    op_Pattern8ColorBarH_BBRGWCYM();                  break; //Color Table_H
    case 17:    op_Pattern8ColorBarV_BBRGWCYM();                  break; //Color Table_V
    case 18:    op_PatternCrosstalk_BlackWhite(0);                break; //Cross Talk WG
    case 19:    op_PatternCrosstalk_BlackWhite(1);                break; //Cross Talk BG
    case 20:    op_PatternChecker_BlackWhite(0, 9, 13);           break; //9*13_WB
    case 21:    op_PatternChecker_BlackWhite(1, 9, 13);           break; //9*13_BW
    case 22:    op_PatternFlickerOneDot(127,0,127, 0,127,0);      break; //Flicker 1dot 127
    case 23:    op_PatternFlickerColumnInversion(127,0,127, 0,127,0);break; //Flicker col 127
    case 24:    op_Pattern4ColorBarGrayScaleV(0, 255);            break; //Color Level_V 255
    case 25:    op_Pattern4ColorBarGrayScaleV(1, 255);            break; //Color Level_V 255 INV
    case 26:    op_Pattern4ColorBarGrayScaleH(0, 255);            break; //Color Level_H 255
    case 27:    op_Pattern4ColorBarGrayScaleH(1, 255);            break; //Color Level_H 255 INV
    case 28:    DisplayBMP(1);                                    break; //PIC 1
    case 29:    DisplayBMP(2);                                    break; //PIC 2
    case 30:    DisplayBMP(3);                                    break; //PIC 3
    case 31:    DisplayBMP(4);                                    break; //PIC 4
    case 32:    DisplayBMP(5);                                    break; //PIC 5
    case 33:    DisplayBMP(6);                                    break; //PIC 6
    case 34:    DisplayBMP(7);                                    break; //PIC 7
    default: break;
  }
}
```

main.c 调用 Lcd_ShowMode()函数：

```c
int main (void) {

    /*!< At this stage the microcontroller clock setting is already configured,
       this is done through SystemInit() function which is called from startup
       file (startup_stm32f10x_xx.s) before to branch to application main.
       To reconfigure the default setting of SystemInit() function, refer to
       system_stm32f10x.c file
     */

  //初始化，包含使用到的GPIO的配置，定时器配置，串口配置，中断向量配置
    Init_Config();


  //配置GPU相关寄存器，初始化SD卡
  Init_CoreSystem();


  //显示画面1
  Lcd_ShowMode(1);
  while(1) {
    //User Logic, eg.:Key-Input process


    //GPU SPI配置处理，通过STM32配合实现对GPU固件的配置
    GPU_SPI_Config_Process();
    if(EBindingPinTrigger) {
      SPI_Flash_EncryptBinding_Process();
      EBindingPinTrigger = 0;
    }
  }
}
```

注意：

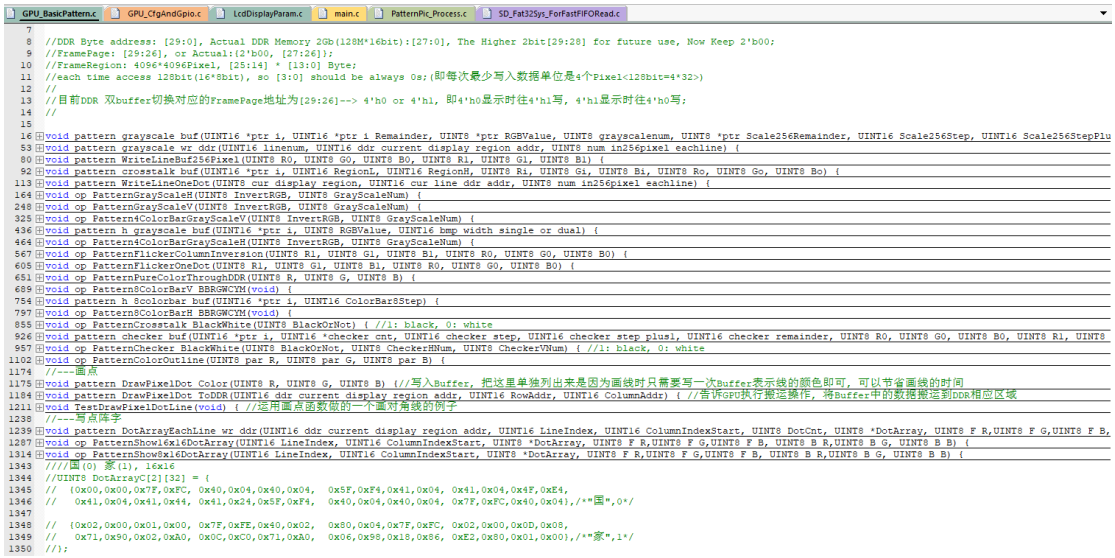
1） Pattern 函数种类可以参考 GPU_Basic_Pattern.c

2） 关于图片显示函数的说明：

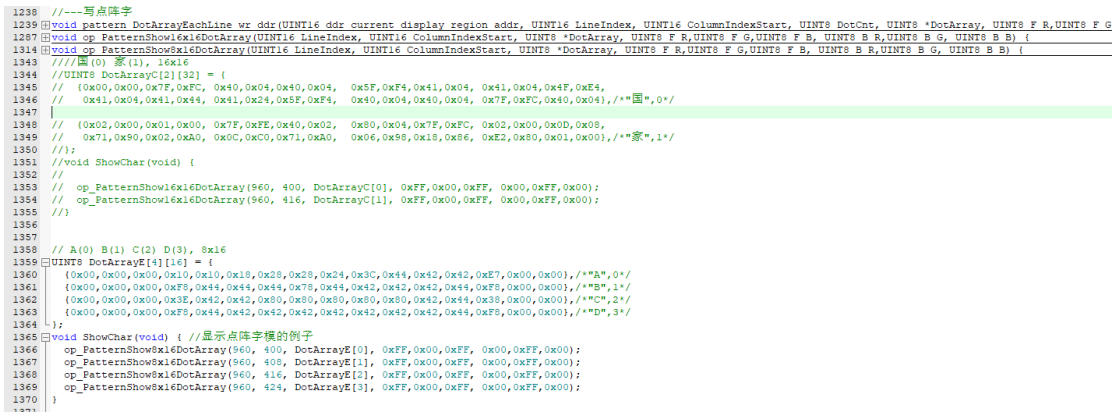
a) DisplayBMP(1); //显示 SD 卡根目录下的 F01.BMP，当前参考代码支持 F01~F99；

b) BMP 图片要使用 24-bit 格式，也就是 RGB888 格式；

c) 第一次使用 SD 卡前，先格式化为 FAT32 格式，然后再增加图片，图片的命名使用 F01.BMP ~ F99.BMP；

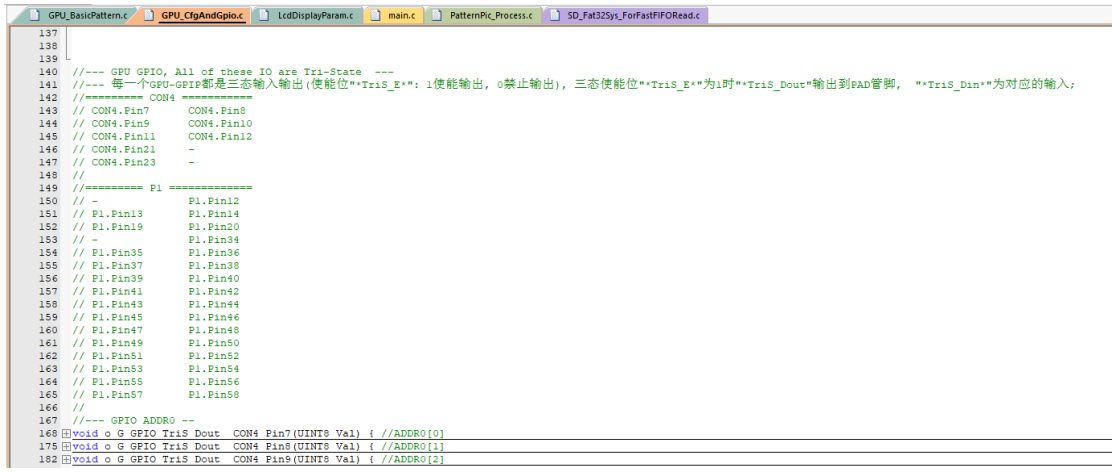


显示点阵函数：



2.2.3 GPU GPIO 输入输出

调用 GPU_CfgAndGpio.c 中描述的函数：



例如：

设置 CON4 连接器的 Pin7 输出为“1”

```
o_G_GPIO_TriS_Dout__CON4_Pin7(1);        //设置输出为高电平
o_G_GPIO_TriS_E__CON4_Pin7(1);           //打开输出使能
```

设置 CON4 连接器的 Pin7 输出为”0”

```
o_G_GPIO_TriS_Dout__CON4_Pin7(0);        //设置输出为高电平
o_G_GPIO_TriS_E__CON4_Pin7(1);           //打开输出使能
```

设置 CON4 连接器的 Pin7 输出为三态

```
o_G_GPIO_TriS_E__CON4_Pin7(0);           //关闭输出使能
```

读取 CON4 连接器的 Pin7 的电平值

```
UINT8 Value;
Value = i_G_GPIO_TriS_Din__CON4_Pin7();  //读取 IO 电平
```

3. 固件升级

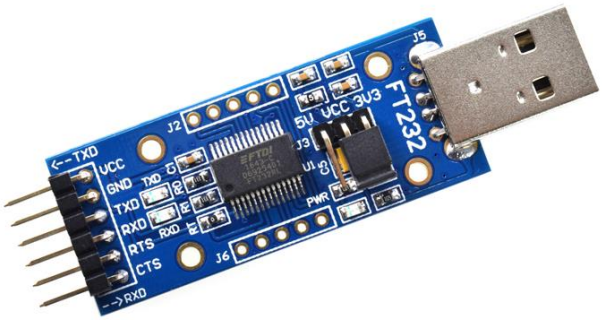
3.1 STM32 固件升级

3.1.1 准备材料

1） 核心板

2） 5V 电源

3） USB→UART（TTL 电平）串口工具，以下简称串口工具，见下图



4） STMicroelectronics flash loader 软件；

下载地址：https://pan.baidu.com/s/1FmbYAJEDwBcNyFl3ff0bpw 密码：1iyq

3.1.2 固件升级步骤

1） 核心板升级接口连接串口工具

P3 连接器 Pin58（Boot）连接 GND（让 STM32 进入固件升级模式）

P1 连接器 Pin7 连接串口工具 RX

P1 连接器 Pin8 连接串口工具 TX

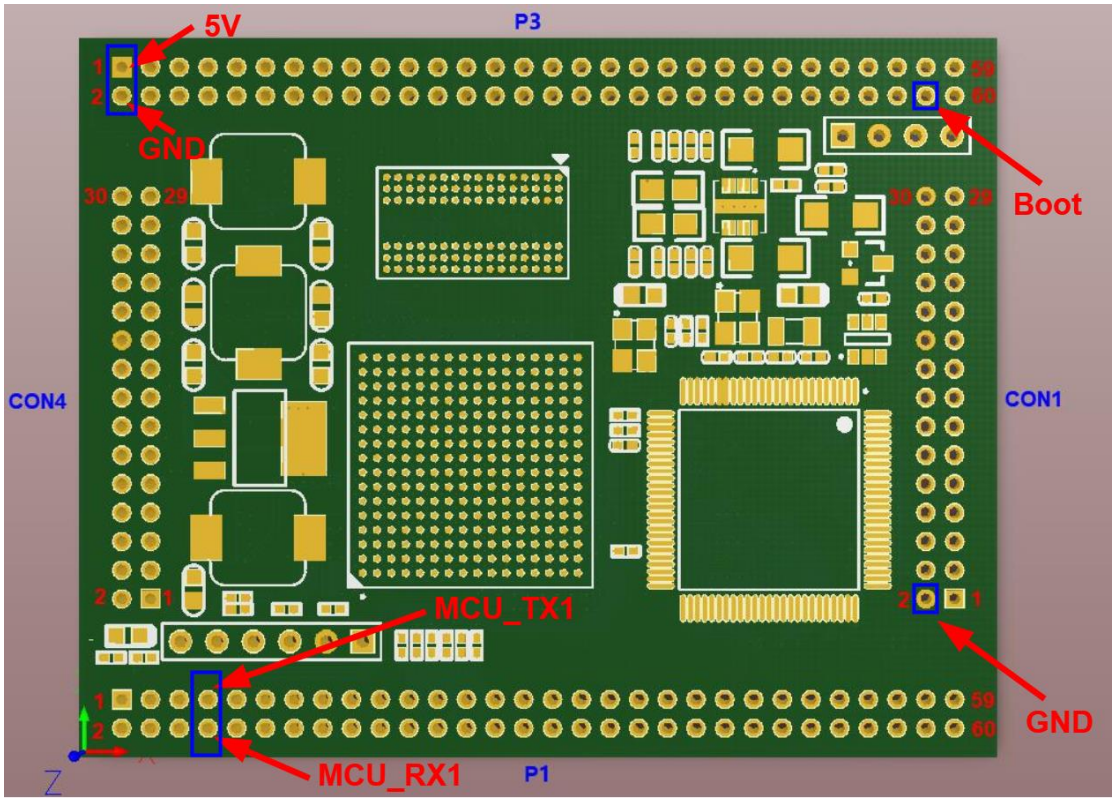
CON1 连接器 Pin2 连接串口工具 GND

2） 核心板供电

P3 连接器 Pin1 连接 5V

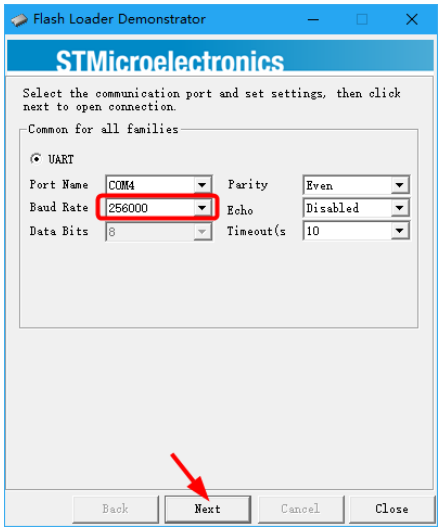
P3 连接器 Pin2 连接 GND

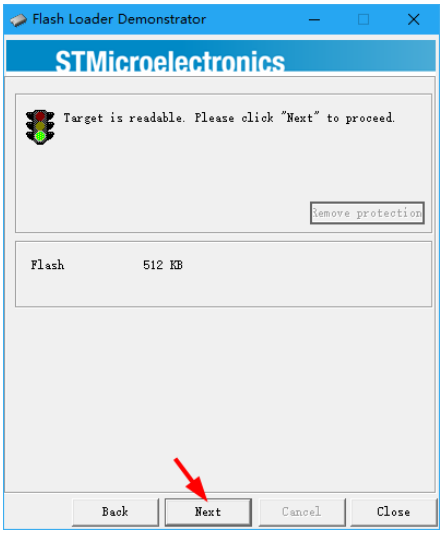
注意：只有先按照步骤 1 连接好，再上电 5V，才可以让 STM32 进入固件升级模式

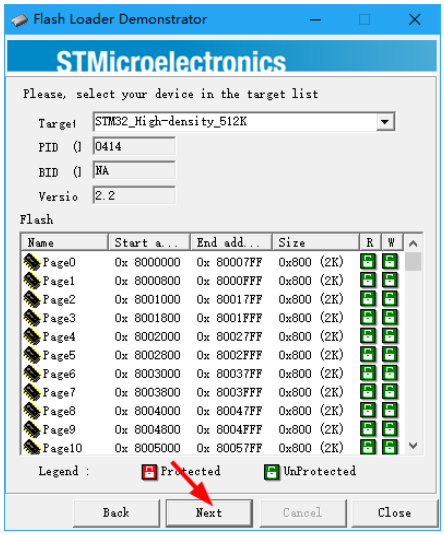


3） 启动 STMicroelectronics Flash Loader 软件：

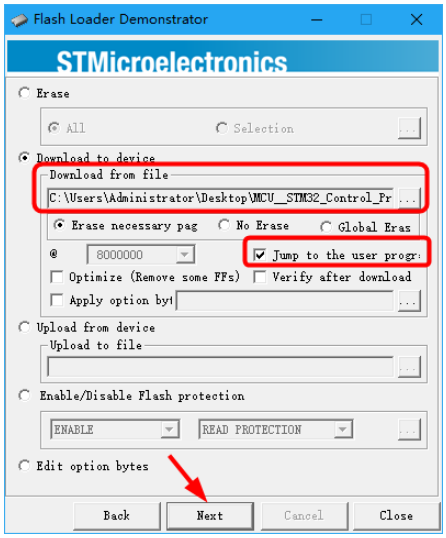
选择正确的 UART 端口（Port Name）

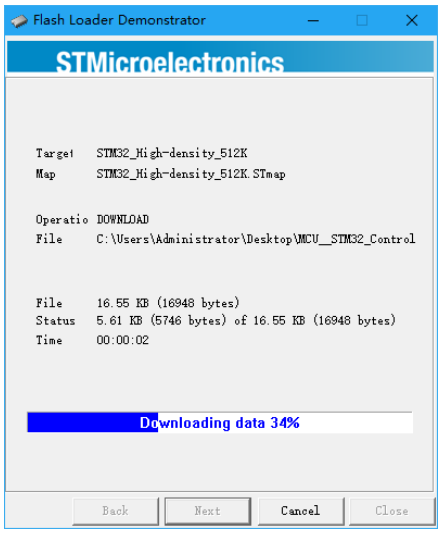






通过以下窗口选择固件。





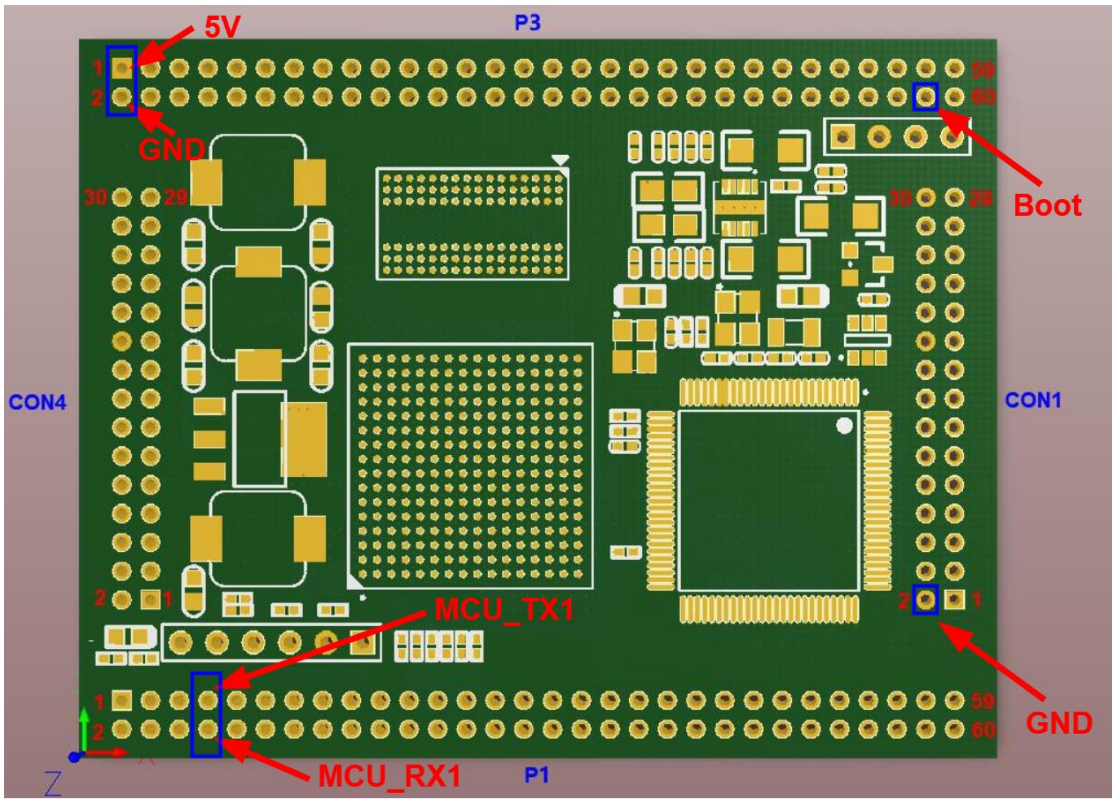
3.2 GPU 固件升级

3.2.1 准备材料

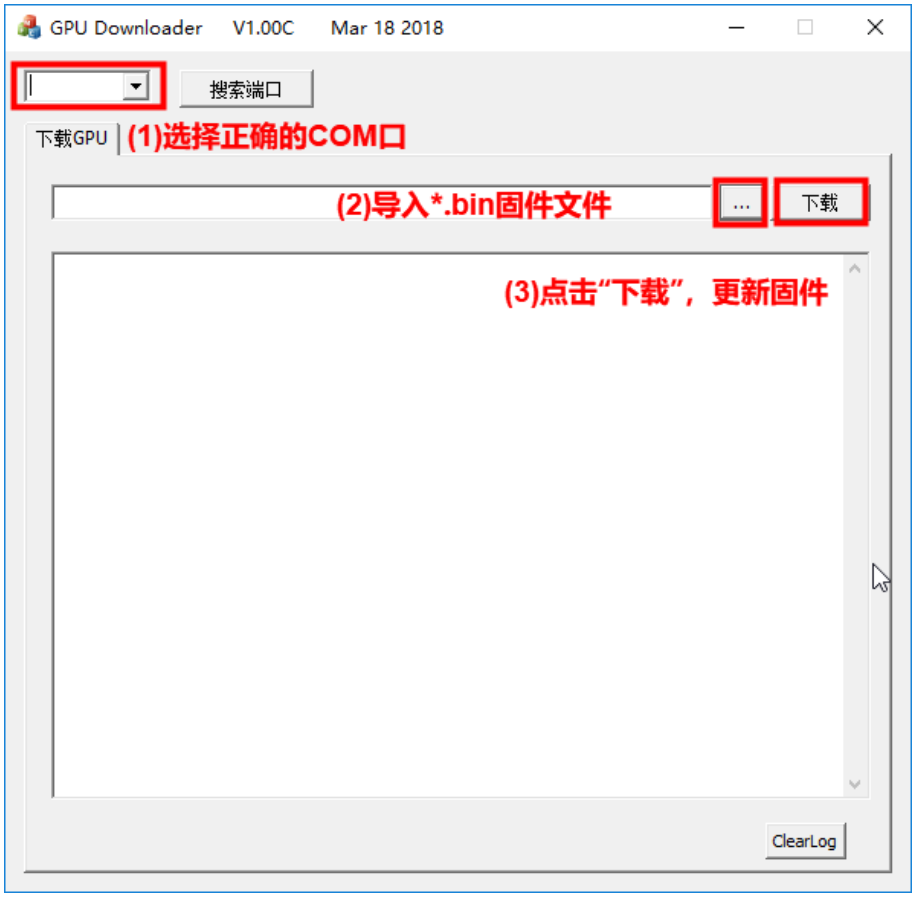
1） 核心板
2） 5V 电源
3） USB→UART（TTL 电平）串口工具，以下简称串口工具
4） GPU Downloader 软件
载地址：https://pan.baidu.com/s/1FmbYAJEDwBcNyFl3ff0bpw 密码：1iyq

3.2.2 固件升级步骤

1） 核心板升级接口连接串口工具
P3 连接器 Pin58（Boot）连接 3.3V
P1 连接器 Pin7 连接串口工具 RX
P1 连接器 Pin8 连接串口工具 TX
CON1 连接器 Pin2 连接串口工具 GND
2） 核心板供电
P3 连接器 Pin1 连接 5V
P3 连接器 Pin2 连接 GND



3） 启动 GPU Downloader 软件：



4. 文档版本

时间	版本	章节	更新记录
2018.03.10	1.0	All	初始版本
2018.03.18	1.1	1.3 章节	更新信号定义说明