

STM32F407_OTG 标准函数库使用说明

版本：1.1

前言

编写说明

根据 ST 提供的 STM32F407 OTG 应用例程，做一个详细的配置讲解。

目录

目录

1 STM32F407 OTG 模块讲解.....	1
1.1 OTG FS 模块讲解	1
1.1.1 引脚讲解.....	2
1.2 OTG HS 模块讲解.....	2
1.2.1 OTG FS PHY 引脚讲解	3
1.2.2 ULPI 接口讲解	3
1.3 STM32F407 OTG 模块总结.....	4
2 OTG 设备模式配置.....	5
2.1 OTG FS 设备模式配置	5
2.1.1 硬件电路修改.....	5
2.1.2 代码修改.....	5
2.2 OTG HS 设备模式配置.....	9
2.2.1 OTG HS 设备模式内部全速 PHY 配置	9
2.2.2 OTG HS 设备模式 ULPI 接口外挂 PHY 配置	13
3 OTG 主机模式配置.....	18
3.1 OTG HS 主机模式 ULPI 接口外挂 PHY 配置	18
3.1.1 ULPI 高速模式代码修改	18
3.1.2 ULPI 全速模式代码修改	18

1 STM32F407 OTG 模块讲解

从 STM32F40x 的数据手册可知，STM32F407 有两个 OTG 模块，分别是 USB OTG FS 和 USB OTG HS。

Table 2. STM32F405xx and STM32F407xx: features and peripheral counts (continued)

Peripherals	STM32F405RG	STM32F405OG	STM32F405VG	STM32F405ZG	STM32F405OE	STM32F407Vx	STM32F407Zx	STM32F407Ix
Communication interfaces	3/2 (full duplex) ⁽²⁾							
	3							
	4/2							
	Yes							
	Yes							
	2							
	Yes							
Camera interface	No				Yes			
GPIOs	51	72	82	114	72	82	114	140
12-bit ADC	3							
Number of channels	16	13	16	24	13	16	24	24
12-bit DAC	Yes							
Number of channels	2							
Maximum CPU frequency	168 MHz							
Operating voltage	1.8 to 3.6 V ⁽³⁾							
Operating temperatures	Ambient temperatures: -40 to +85 °C / -40 to +105 °C							
	Junction temperature: -40 to +125 °C							
Package	LQFP64	WLCSP90	LQFP100	LQFP144	WLCSP90	LQFP100	LQFP144	UFBGA176 LQFP176

1. For the LQFP100 and WLCSP90 packages, only FSMC Bank1 or Bank2 are available. Bank1 can only support a multiplexed NOR/PSRAM memory using the NE1 Chip Select. Bank2 can only support a 16- or 8-bit NAND Flash memory using the NCE2 Chip Select. The interrupt line cannot be used since Port G is not available in this package.

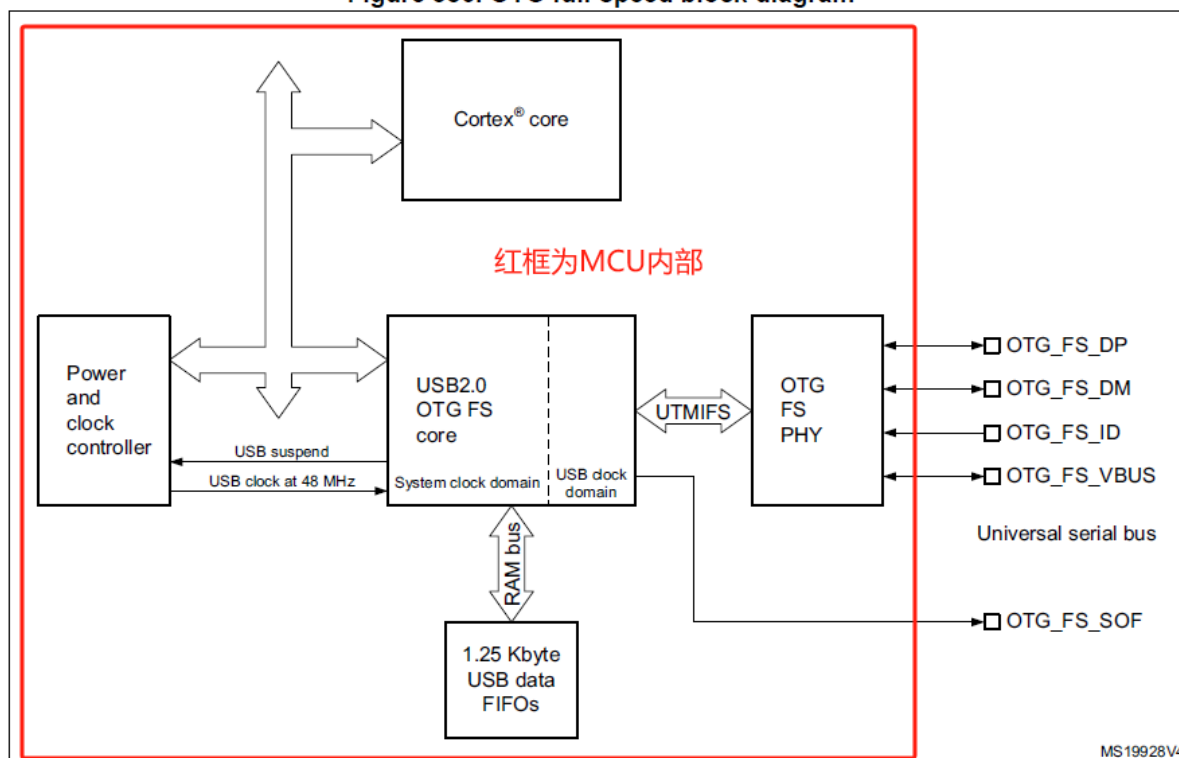
2. The SPI2 and SPI3 interfaces give the flexibility to work in an exclusive way in either the SPI mode or the I²S audio mode.

3. V_{DDP}/V_{DDA} minimum value of 1.7 V is obtained when the device operates in reduced temperature range, and with the use of an external power supply supervisor (refer to [Section : Internal reset OFF](#)).

1.1 OTG FS 模块讲解

从功能框图可知，USB FS 模块分为 OTG 内核部分和 OTG PHY 部分，内核部分主要是数字逻辑实现；PHY 部分主要是模拟实现，比如把特定的差分信号转换为特定的数字信号。

Figure 386. OTG full-speed block diagram



1.1.1 引脚讲解

从上图可知，开放给嵌入式软件工程师的接口为 OTG_FS_DP、OTG_FS_DM、OTG_FS_ID、OTG_FS_VBUS 和 OTG_FS_SOF 引脚，所以我们只需要关注这几条信号线即可（内部的 UTMIFS 接口可以不关注），引脚对应的功能如下：

Pin 脚	功能	备注
PA8	OTG_FS_SOF	SOF 输出引脚，对于全速/低速设备，输出的频率为 1KHz
PA9	OTG_FS_VBUS	VBUS 引脚，通过寄存器选择外部 VBUS 供电还是内部 VBUS 供电
PA10	OTG_FS_ID	ID 引脚，拉高/浮空为设备模式，拉低为主机模式
PA11	OTG_FS_DM	差分信号负端
PA12	OTG_FS_DP	差分信号正端

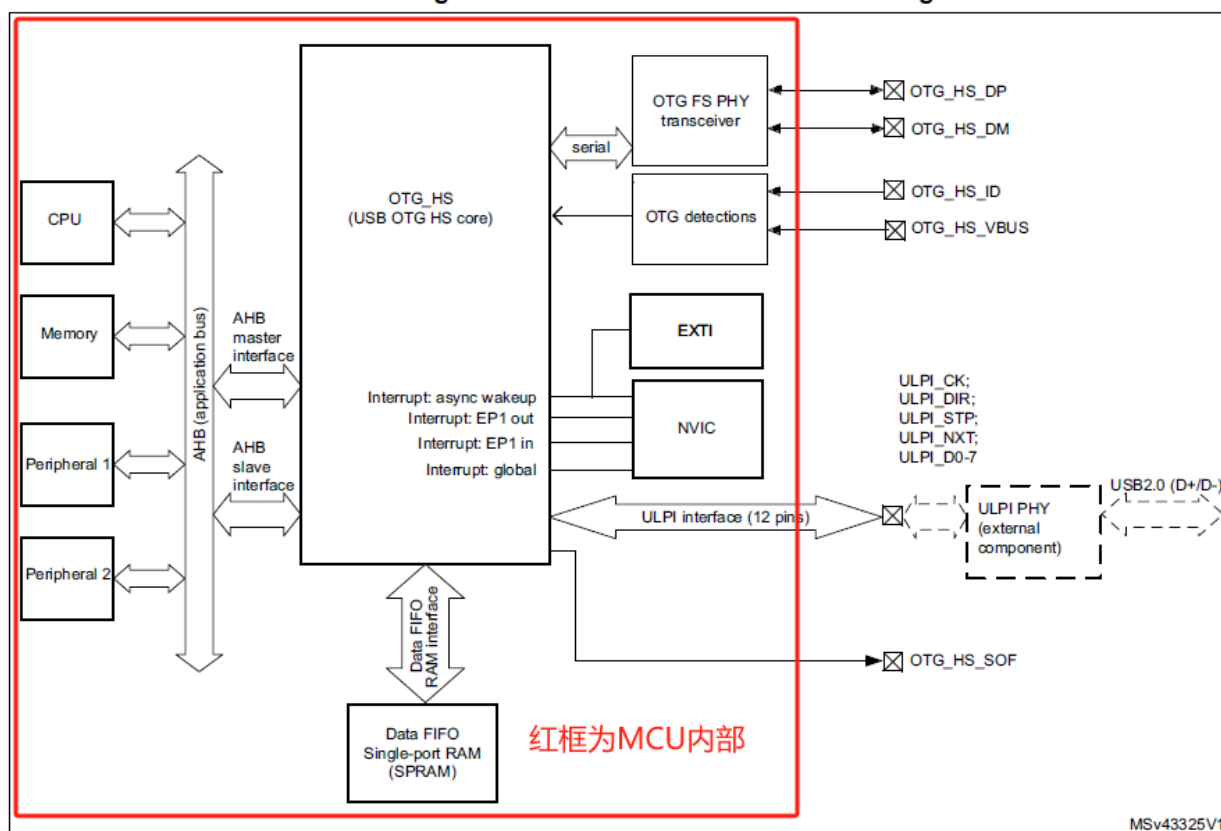
1.2 OTG HS 模块讲解

此框图主要描述了 OTG 的三大部分：

- OTG HS 内核部分
- OTG FS PHY 部分
- ULPI 接口部分

OTG HS 内核部分不是我们这里关注的要点，主要关注 OTG FS PHY 部分和 ULPI 接口部分。

Figure 410. USB OTG interface block diagram



1. The USB DMA cannot directly address the internal Flash memory.

1.2.1 OTG FS PHY 引脚讲解

OTG HS 模块中的 OTG FS PHY 可以理解为 OTG FS 模块中的 PHY, 他俩的 PHY 是同一类 PHY, 都是通过 UTMI 接口进行通信的, 因为在芯片内部, 所以我们只关心 PHY 模块引出的引脚, OTG_HS_DP、OTG_HS_DM、OTG_HS_ID、OTG_HS_VBUS、OTG_HS_SOF。

Pin 脚	功能	备注
PA4	OTG_HS_SOF	SOF 输出引脚, 对于全速/低速设备, 输出的频率为 1KHz
PB13	OTG_HS_VBUS	VBUS 引脚, 通过寄存器选择外部 VBUS 供电还是内部 VBUS 供电
PB12	OTG_HS_ID	ID 引脚, 拉高/浮空为设备模式, 拉低为主机模式
PB14	OTG_HS_DM	差分信号负端
PB15	OTG_HS_DP	差分信号正端

1.2.2 ULPI 接口讲解

此接口需要外挂一个 USB 的 PHY 芯片, 这个芯片我们一般使用 USB3318 和 USB3300 这两款芯片, 都是支持高速 USB 的, 通过我们配置 STM32F407 的 OTG HS 寄存器, 进一步在 ULPI 总线上下发命令来配置这类型的 PHY 芯片, 他们支持的协议都是一样的, 所以使用相同的 STM32F407 OTG 程序, 能够完全驱动这两款 PHY 芯片, 进而实现 USB 的通信。

Pin 脚	功能	备注
PA5	ULPI_CLK	MCU 时钟输入，固定 60M
PC0	ULPI_STP	MCU 在一个时钟周期内触发 STP，以停止当前总线上的数据流。如果 MCU 正在向 PHY 发送数据，STP 表示数据的最后一个字节在前一个周期中位于总线上。
PC3	ULPI_NXT	限制数据。当 MCU 向 PHY 发送数据时，NXT 指示当前字节何时已被 PHY 接受。MCU 在下一个时钟周期中将下一个字节放置在数据总线上。
PC2	ULPI_DIR	控制数据总线的方向。当 PHY 有数据要传输到链路时，它将 DIR 驱动到高电平以获得总线的所有权。当 PHY 没有要传输的数据时，它将 DIR 驱动为低电平，并监视总线以获取来自链路的命令。
PA3	ULPI_D0	双向数据总线
PB0	ULPI_D1	双向数据总线
PB1	ULPI_D2	双向数据总线
PB10	ULPI_D3	双向数据总线
PB11	ULPI_D4	双向数据总线
PB12	ULPI_D5	双向数据总线
PB13	ULPI_D6	双向数据总线
PB5	ULPI_D7	双向数据总线

其实以上 MCU 的引脚定义我们也不必太多关注，只需要把 PHY 芯片的对应引脚接入 MCU 对应的引脚即可。

1.3 STM32F407 OTG 模块总结

STM32F407 有两个 OTG 模块，分别是 OTG FS 和 OTG HS。

- OTG FS 内置全速 PHY 芯片，通过引脚 PA11 和 PA12 进行通信
- OTG HS 分为两个部分，一部分是内置全速 PHY 芯片，通过引脚 PB14 和 PB15 进行通信；另一部分是通过 ULPI 接口外挂一个高速 PHY 芯片（如 USB3318 或 USB3300），通过高速芯片进行 USB 的通信

2 OTG 设备模式配置

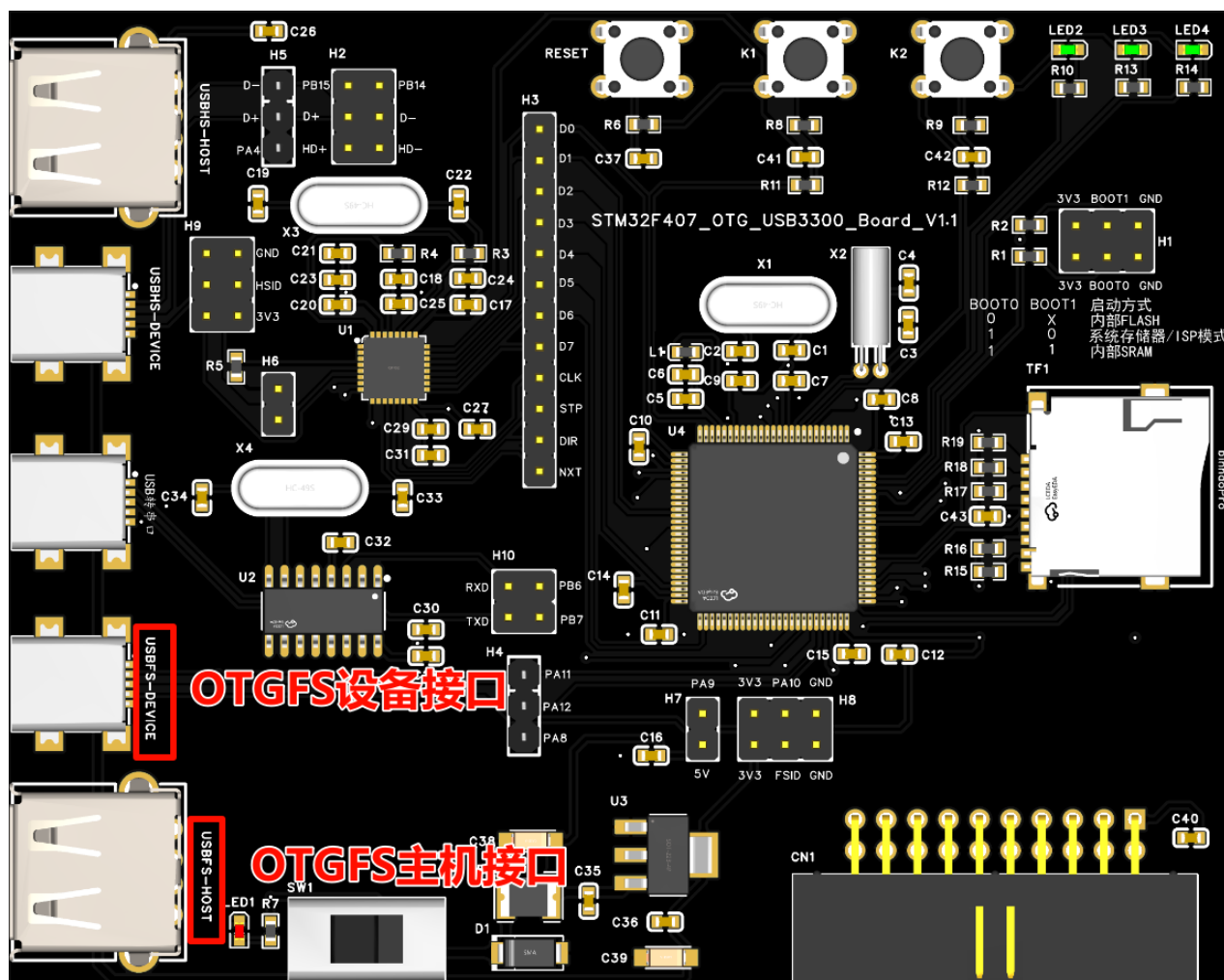
这里以《VCP_Loopback》代码进行讲解，其它代码类似。

2.1 OTG FS 设备模式配置

2.1.1 硬件电路修改

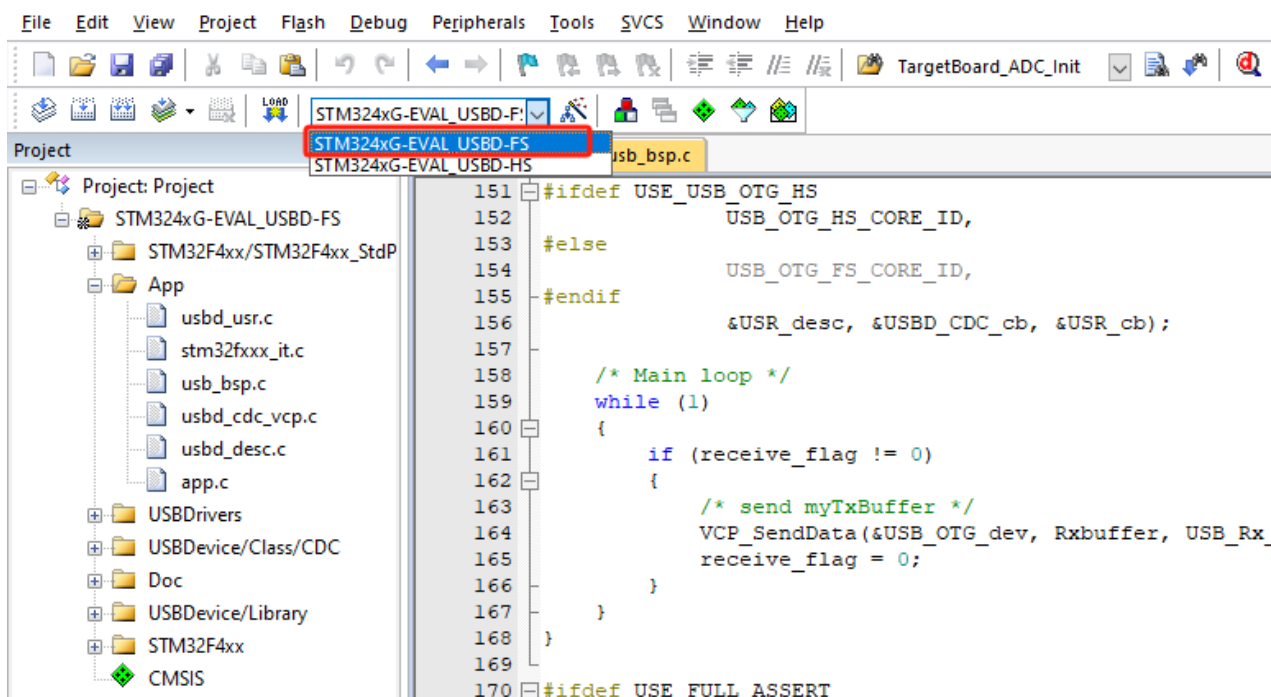
硬件电路无需修改，这里以 STM32F407_OTG_USB3300_Board 进行接线讲解，其他板子类似。

OTG FS 主机模式的时候，需要使用 USB 线接 USBFS-HOST 端子处，OTG FS 设备模式的时候，需要使用 USB Mini 线接 USBFS-DEVICE 端子处。



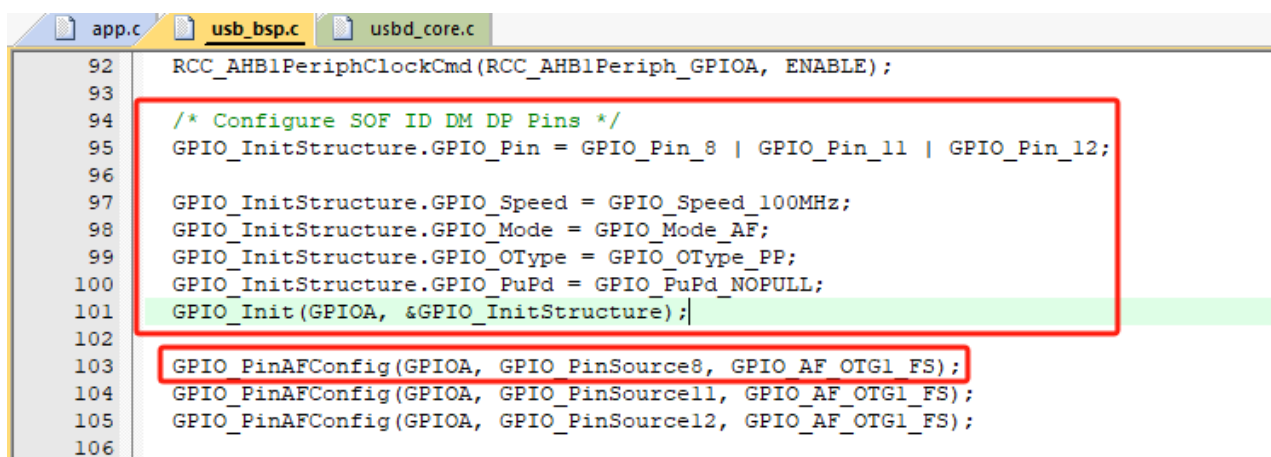
2.1.2 代码修改

1. 打开《VCP_Loopback》工程，选择《STM324xG-EVAL_USBD-FS》，修改之后编译下载，就能在任务管理器查看到相应的设备。



2.1.2.1 SOF 输出代码修改

想要 SOF 输出到 pin 脚，需要配置 PA8 为 AF 功能，代码如下：



注意：这里不需要设置 OTG_FS_GCCFG 寄存器的 SOFOUTEN 位，其实设置了也没事，测试只要把 PA8 配置为 AF 功能，PA8 会自动输出 1KHz 的频率。

2.1.2.2 VBUS 引脚供电代码修改

通过宏定义《#define VBUS_SENSING_ENABLED》确定 VBUS（PA9）引脚是否需要外接电源

```

usb_conf.h
152
153 /****** USB OTG FS CONFIGURATION *****/
154 #ifndef USB_OTG_FS_CORE
155 #define RX_FIFO_FS_SIZE          128
156 #define TX0_FIFO_FS_SIZE        64
157 #define TX1_FIFO_FS_SIZE        128
158 #define TX2_FIFO_FS_SIZE        0
159 #define TX3_FIFO_FS_SIZE        0
160
161 /* #define USB_OTG_FS_LOW_PWR_MGMT_SUPPORT */
162 /* #define USB_OTG_FS_SOF_OUTPUT_ENABLED */
163 #endif
164
165 /****** USB OTG MISC CONFIGURATION *****/
166 #define VBUS_SENSING_ENABLED
167
168 /****** USB OTG MODE CONFIGURATION *****/
169 /* #define USE_HOST_MODE */

```

如果定义了《VBUS_SENSING_ENABLED》这个宏，那么 VBUS（PA9）引脚需要外部供电（5V），且必须保证 PA9 配置为输入模式。

```

app.c  usb_bsp.c
94 /* Configure SOF ID DM DP Pins */
95 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8 | GPIO_Pin_11 | GPIO_Pin_12;
96
97 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
98 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
99 GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
100 GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
101 GPIO_Init(GPIOA, &GPIO_InitStructure);
102
103 GPIO_PinAFConfig(GPIOA, GPIO_PinSource8, GPIO_AF_OTG1_FS);
104 GPIO_PinAFConfig(GPIOA, GPIO_PinSource11, GPIO_AF_OTG1_FS);
105 GPIO_PinAFConfig(GPIOA, GPIO_PinSource12, GPIO_AF_OTG1_FS);
106
107 /* Configure VBUS Pin */
108 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
109 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
110 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
111 GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
112 GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
113 GPIO_Init(GPIOA, &GPIO_InitStructure);
114
115

```

如果没有定义《VBUS_SENSING_ENABLED》这个宏，那么 VBUS（PA9）引脚不需要外部供电，此时 PA9 可用作其它 GPIO 引脚。

2.1.2.3 ID 引脚有效代码修改

程序默认配置强制设置设备模式，此时 ID 引脚无效。

```

usb_conf.h  usb_bsp.c  app.c  usbd_core.c  usb_dcd.c
131
132  /*Init the Core (common init.) */
133  USB_OTG_CoreInit(pdev);
134
135  #else
136
137  /*Init the Core (common init.) */
138  USB_OTG_CoreInit(pdev);
139
140  /* Force Device Mode*/
141  USB_OTG_SetCurrentMode(pdev, DEVICE_MODE);
142
143  #endif
144
145  /* Init Device */
146  USB_OTG_CoreInitDev(pdev);
147
148  /* Enable USB Global interrupt */
149  USB_OTG_EnableGlobalInt(pdev);
150  }
151
152
153  /**

```

如果想要 ID 引脚有效，需要屏蔽以下代码，并配置 PA10 为 AF 功能，当 ID 引脚拉高/浮空的时候，设备管理器能够识别到设备，当 ID 引脚拉低的时候，设备管理器不能识别到设备。

```

app.c  startup_stm32f40xx.s  system_stm32f4xx.c  stm32f4xx.h  usb_conf.h  usb_bsp.c  usbd_core.c  usb_dcd.c  stm32f4xx_gpio.h
134
135  #else
136
137  /*Init the Core (common init.) */
138  USB_OTG_CoreInit(pdev);
139
140  /* Force Device Mode*/
141  //USB_OTG_SetCurrentMode(pdev, DEVICE_MODE);
142
143  #endif
144
145  /* Init Device */
146  USB_OTG_CoreInitDev(pdev);
147
148  /* Enable USB Global interrupt */
149  USB_OTG_EnableGlobalInt(pdev);
150  }
151
152
153  /**
154  * @brief Configure an EP

```

```

app.c  usb_bsp.c
106
107  /* Configure VBUS Pin */
108  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
109  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
110  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
111  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
112  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
113  GPIO_Init(GPIOA, &GPIO_InitStructure);
114
115  /* Configure ID pin */
116  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
117  // GPIO_InitStructure.GPIO_OType = GPIO_OType_OD;
118  // GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
119  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
120  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
121  GPIO_Init(GPIOA, &GPIO_InitStructure);
122  GPIO_PinAFConfig(GPIOA, GPIO_PinSource10, GPIO_AF_OTG1_FS);
123
124  RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);
125  RCC_AHB2PeriphClockCmd(RCC_AHB2Periph_OTG_FS, ENABLE);
126  #else

```

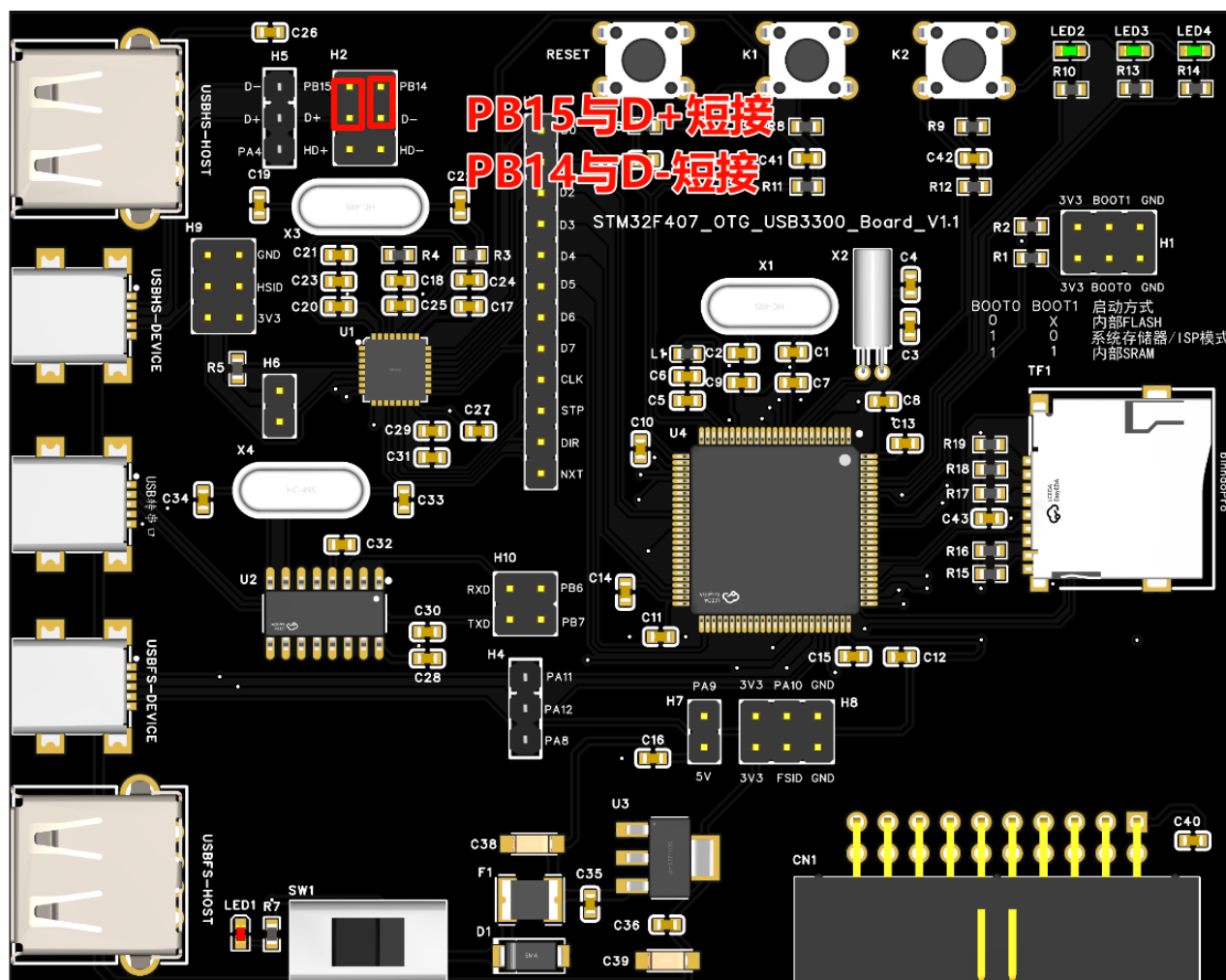
2.2 OTG HS 设备模式配置

OTG HS 模式又分为内部全速 PHY 和 ULPI 接口外挂 PHY。

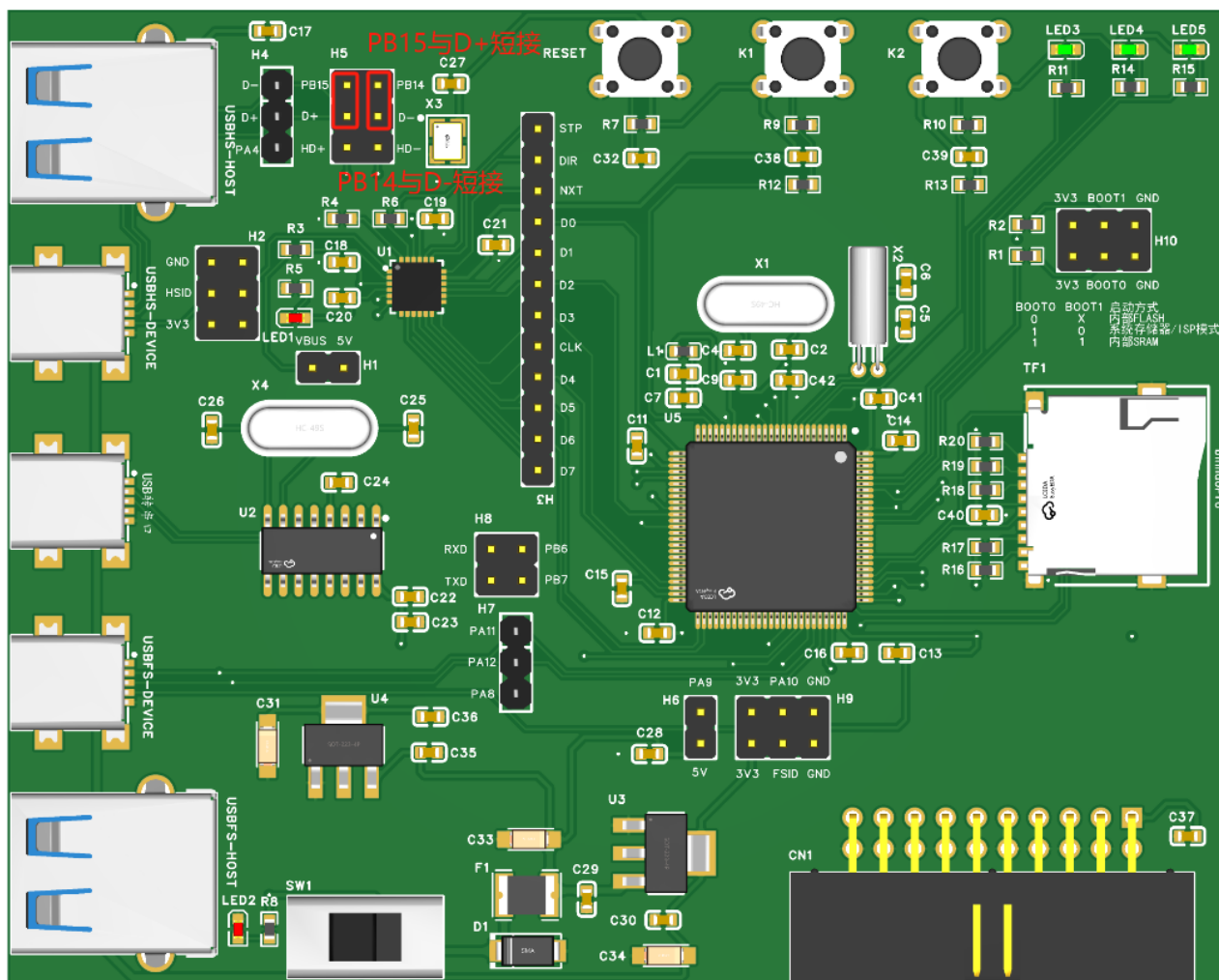
2.2.1 OTG HS 设备模式内部全速 PHY 配置

2.2.1.1 硬件电路修改

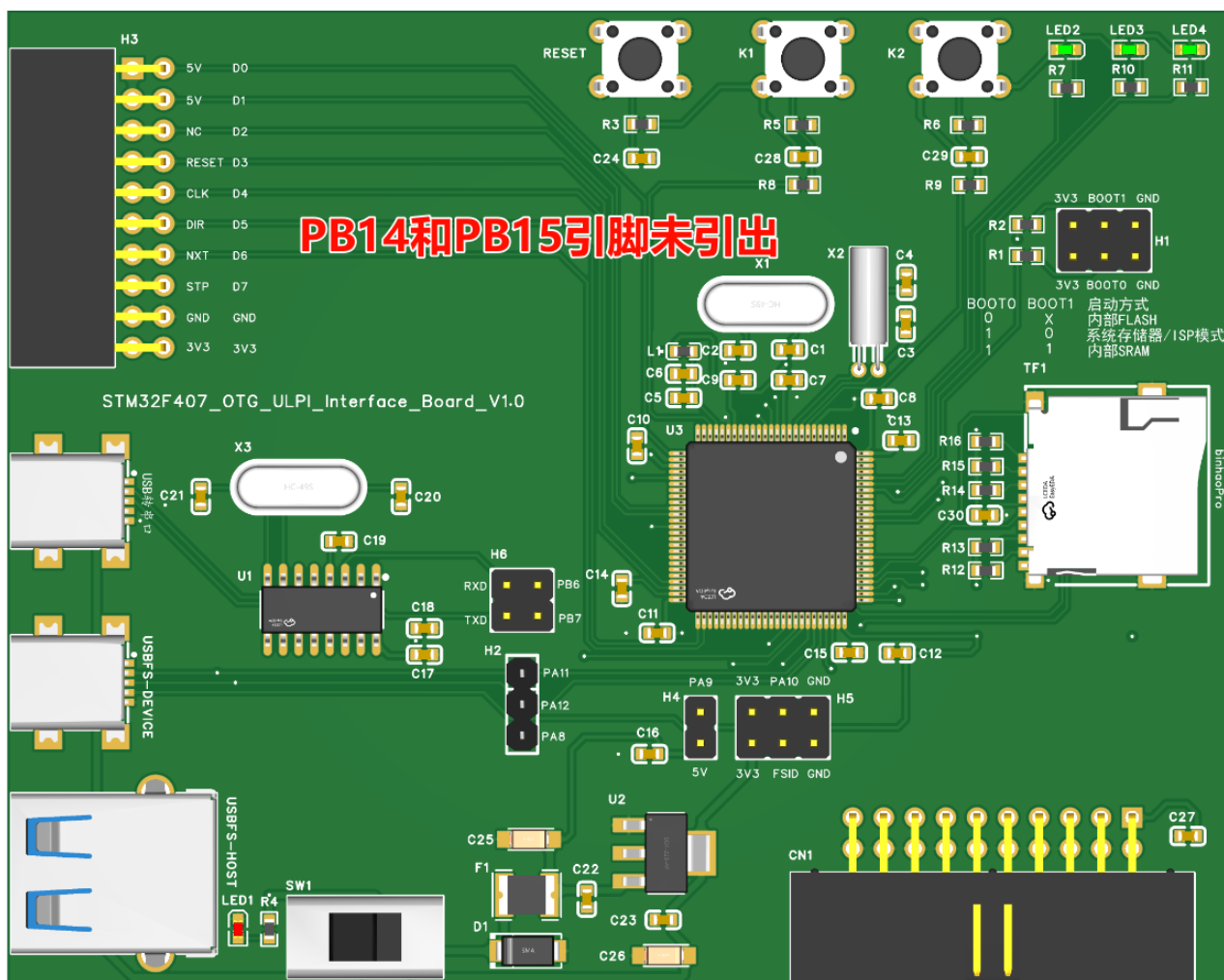
对于 STM32F407_OTG_USB3300_Board，在 H2 端子处，PB15 与 D+短接，PB14 与 D-短接，主机模式使用 USB 线接 USBHS-HOST 端子处，设备模式使用 USB Mini 线接 USBHS-DEVICE 端子处



对于 STM32F407_OTG_USB3318_Board, 在 H5 端子处, PB15 与 D+短接, PB14 与 D-短接, 主机模式使用 USB 线接 USBHS-HOST 端子处, 设备模式使用 USB Mini 线接 USBHS-DEVICE 端子处



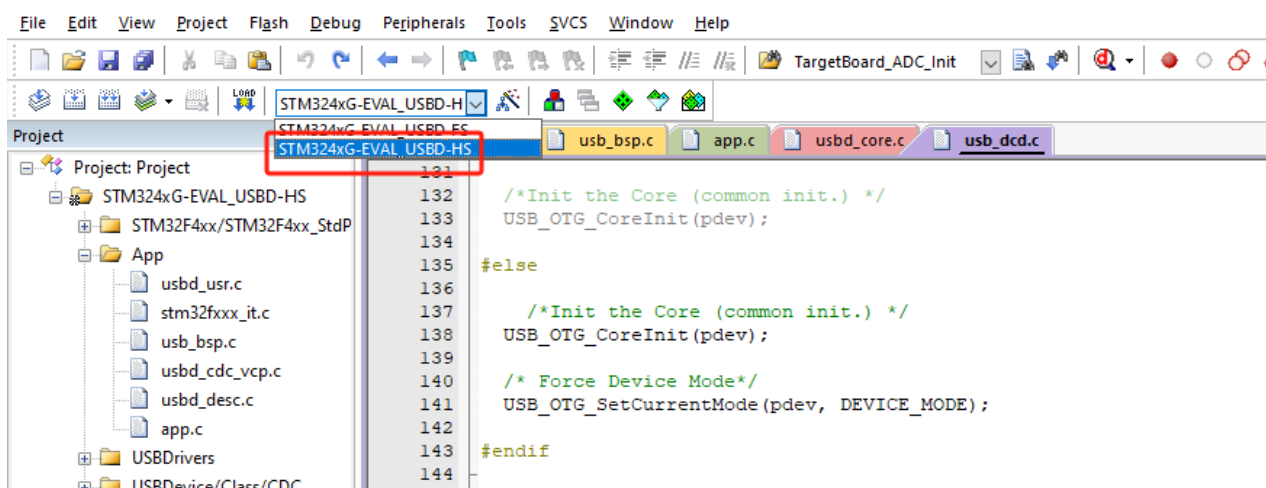
STM32F407 OTG ULPI Interface Board 此接口未开放



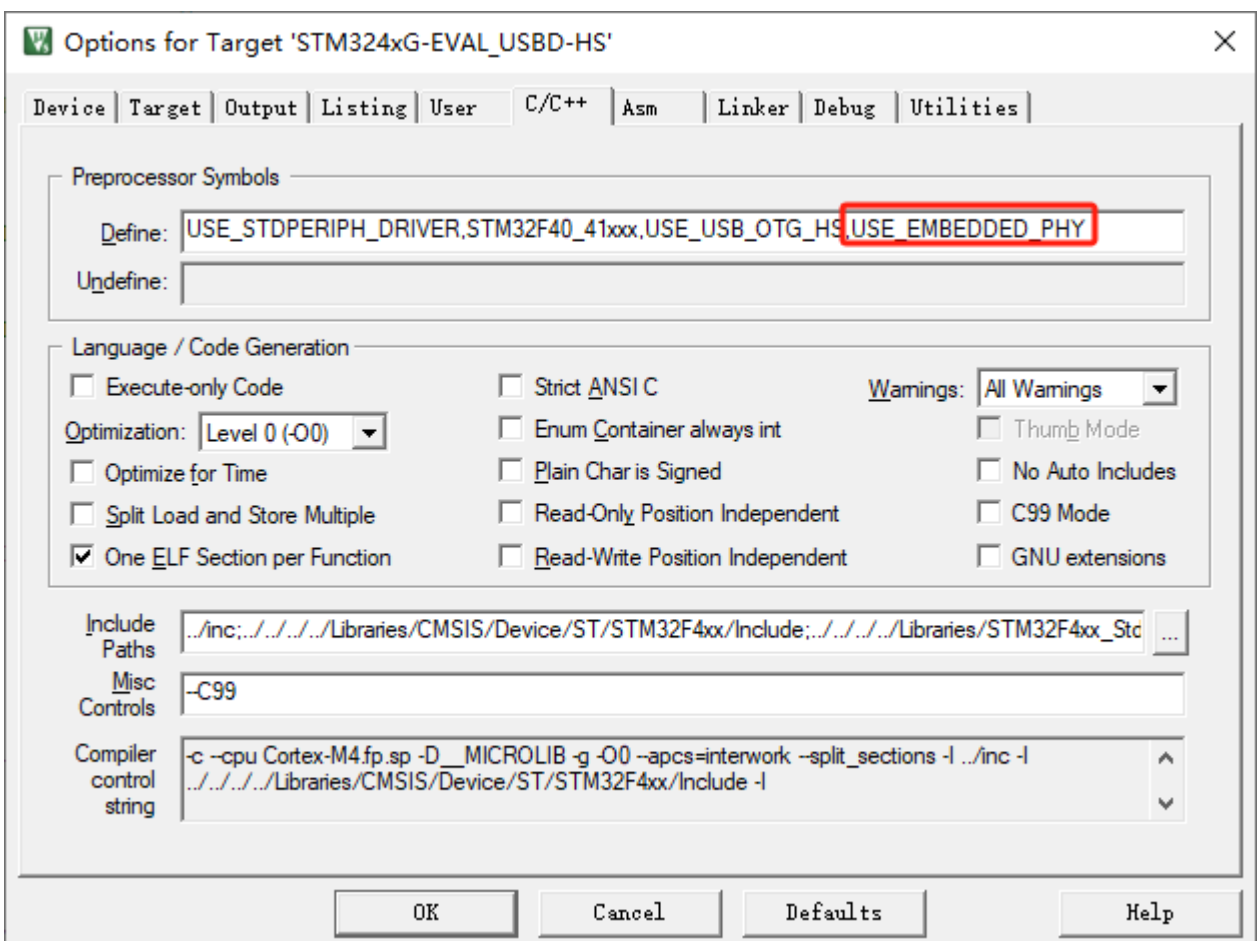
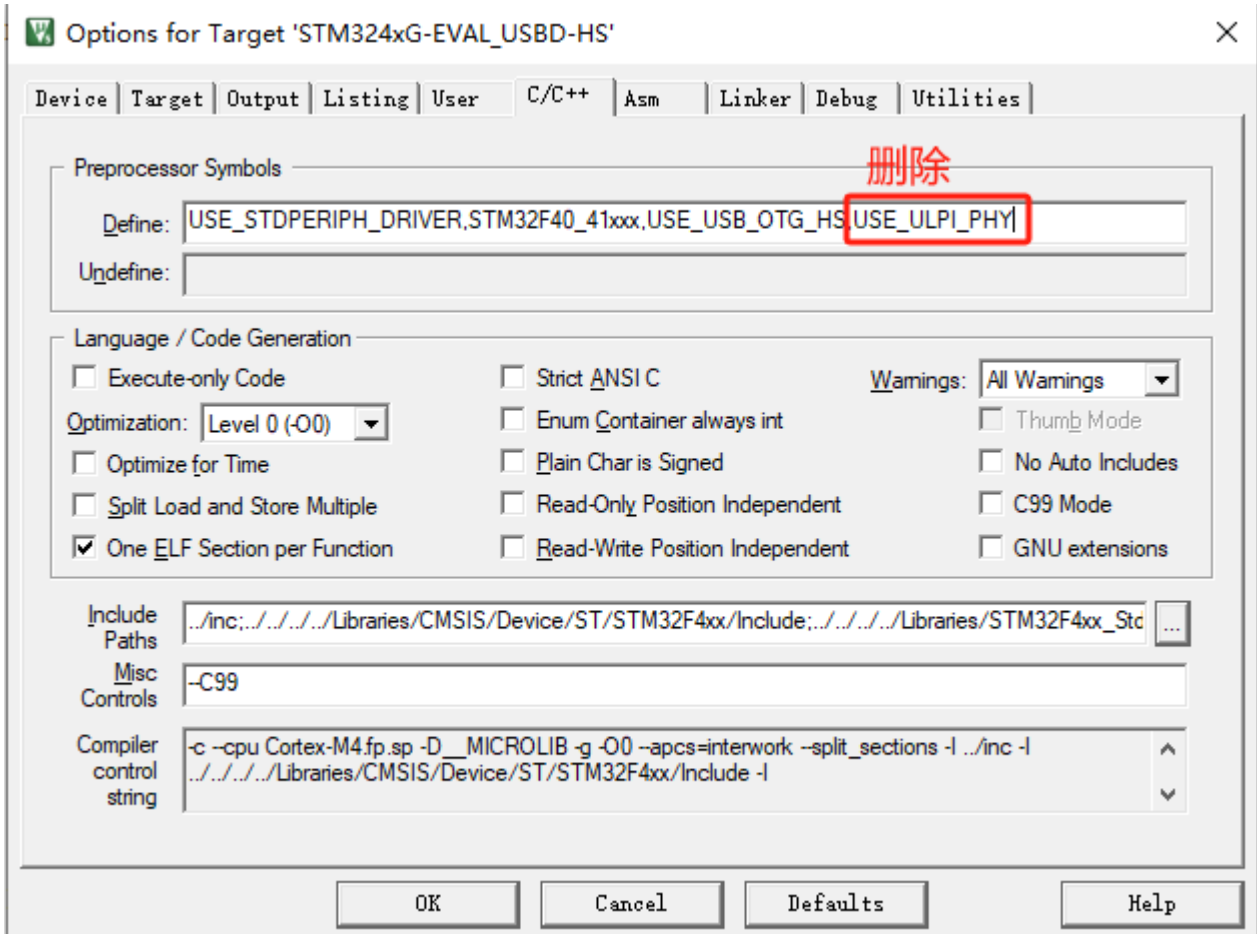
2.2.1.2 代码修改

内部全速 PHY 不需要外挂 PHY 芯片，通过 PB14 和 PB15 引脚进行通信。

1. 打开《VCP Loopback》工程，选择《STM324xG-EVAL USB-D-HS》



2. 删除《USE_ULPI_PHY》宏定义，并添加《USE_EMBEDDED_PHY》宏定义。编译下载，就能在任务管理器查看到相应的设备。



2.2.1.2.1 SOF 输出代码修改

参考 2.1.2.1

2.2.1.2.2 VBUS 引脚供电代码修改

此板子不支持，原理类似 2.1.2.2

2.2.1.2.3 ID 引脚有效代码修改

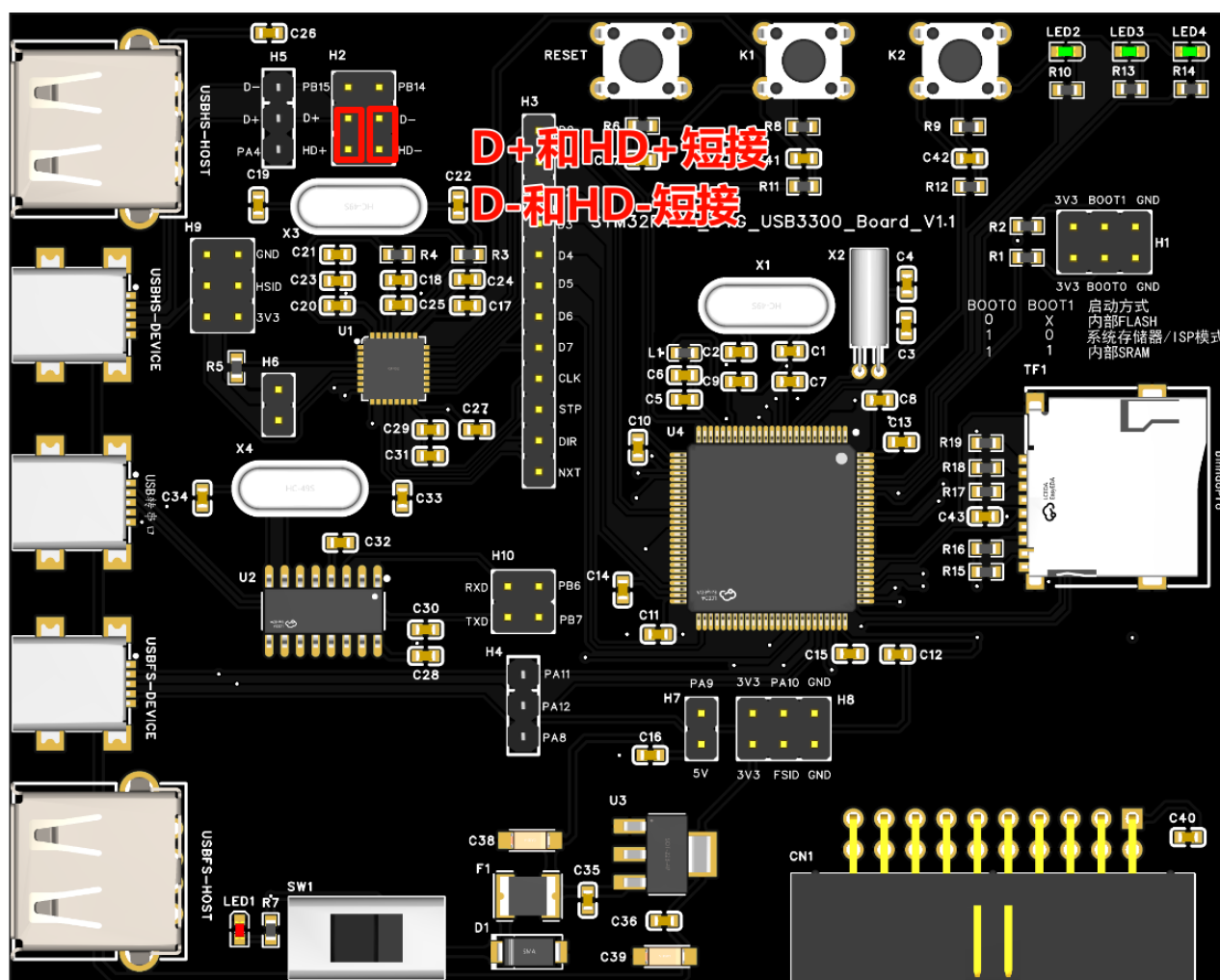
此板子不支持，原理类似 2.1.2.3

2.2.2 OTG HS 设备模式 ULPI 接口外挂 PHY 配置

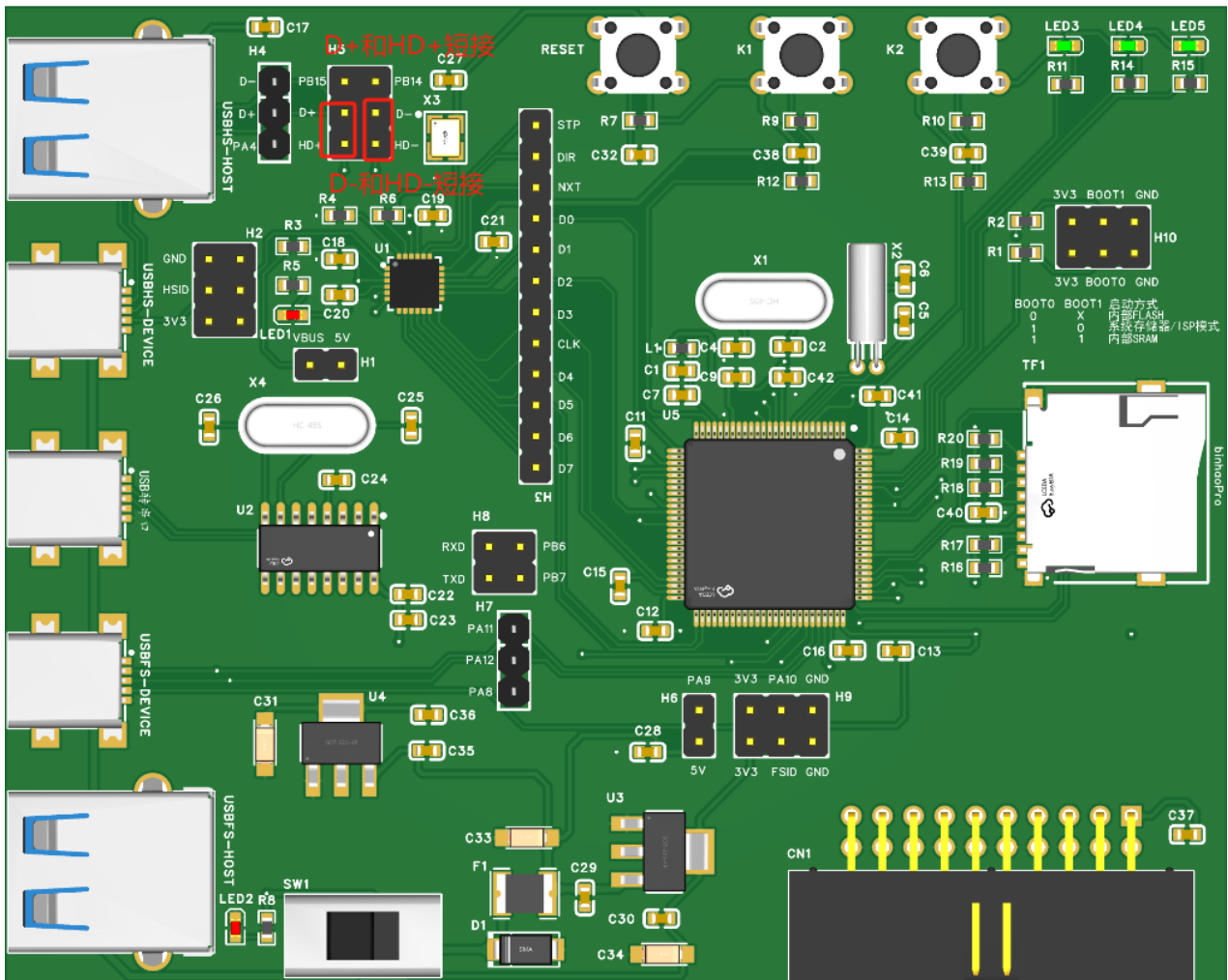
ULPI 接口外挂 PHY 需要外接 USB 芯片，通常使用 USB3318 或 USB3300 芯片，MCU 通过 ULPI 引脚与 PHY 芯片进行交互，通过 PHY 芯片的 D+和 D-引脚与其他 USB 主机通信。

2.2.2.1 硬件电路修改

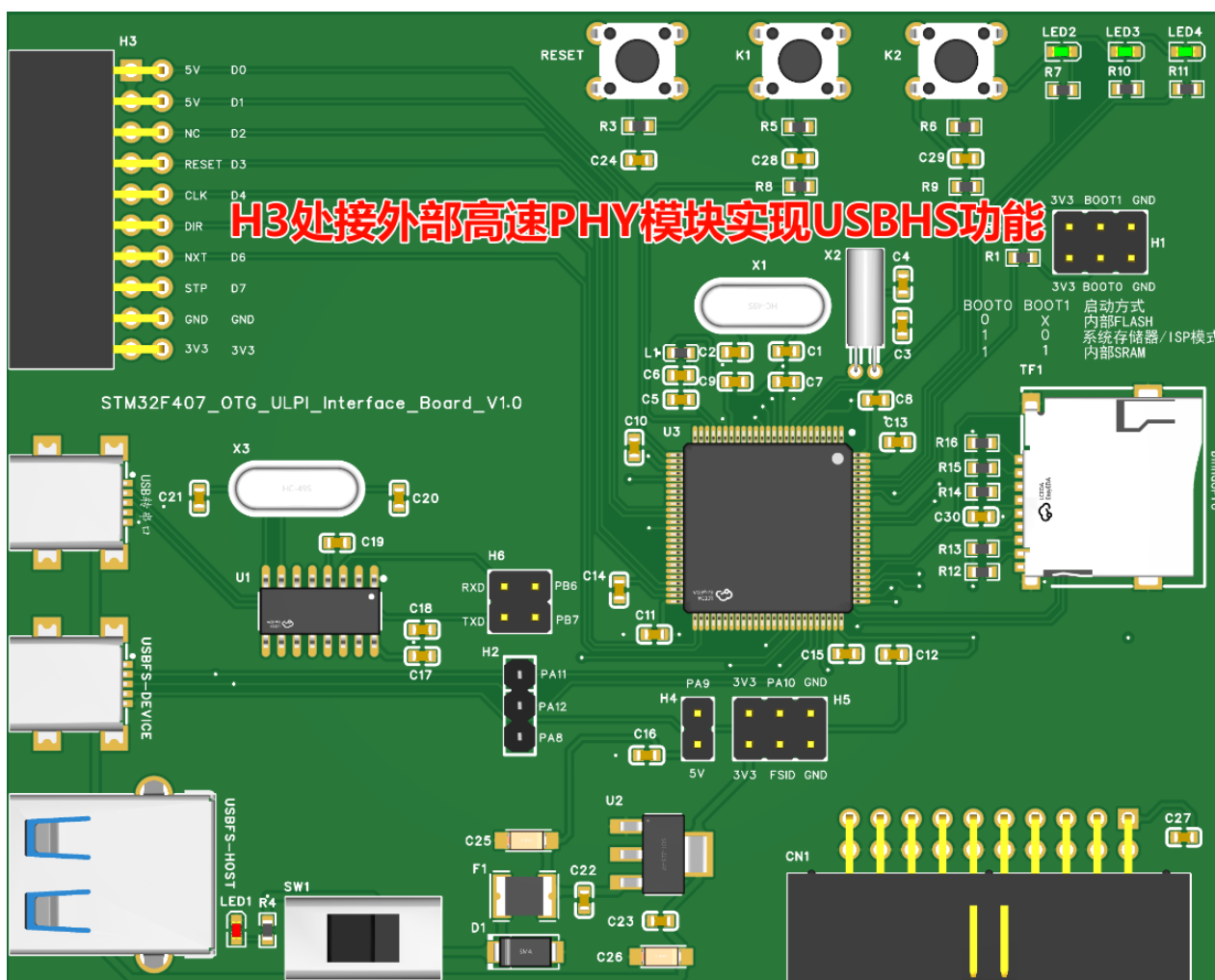
对于 STM32F407_OTG_USB3300_Board，在 H2 端子处，HD+与 D+短接，HD-与 D-短接，主机模式使用 USB 线接 USBHS-HOST 端子处，设备模式使用 USB Mini 线接 USBHS-DEVICE 端子处



对于 STM32F407_OTG_USB3318_Board, 在 H5 端子处, HD+与 D+短接, HD-与 D-短接, 主机模式使用 USB 线接 USBHS-HOST 端子处, 设备模式使用 USB Mini 线接 USBHS-DEVICE 端子处

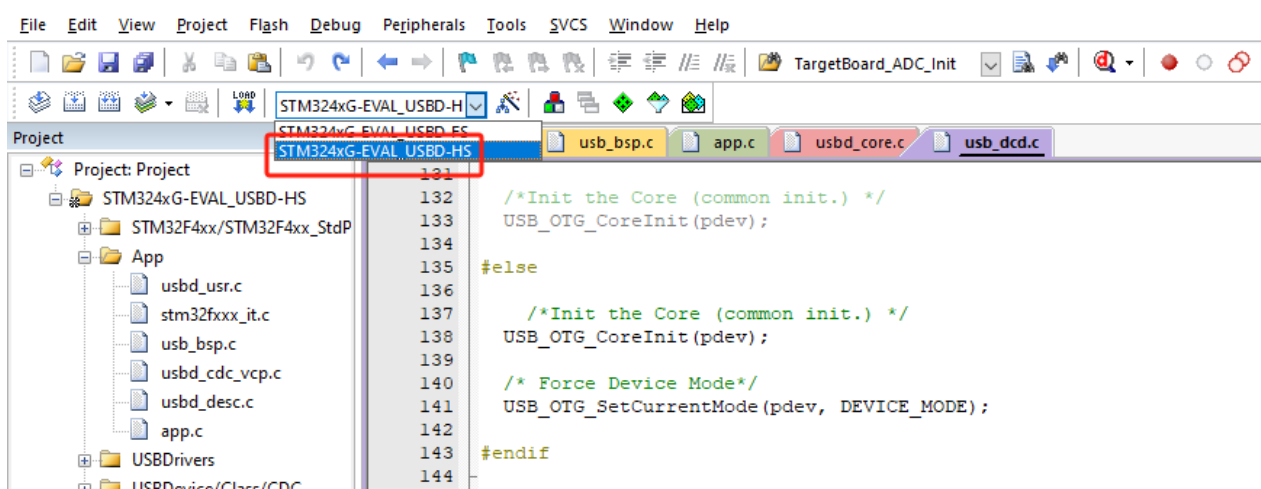


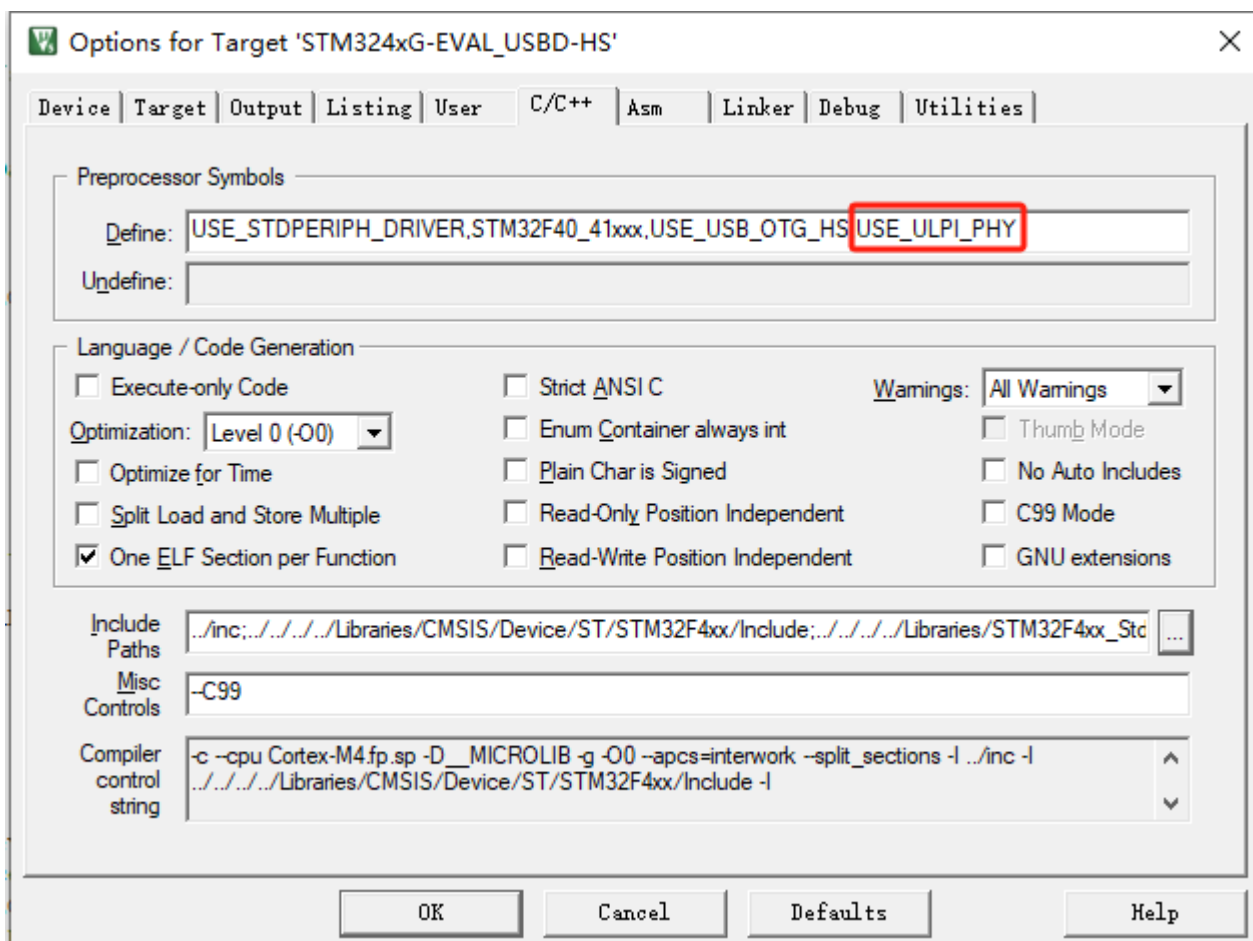
对于STM32F407_OTG_ULPI_Interface_Board,需要在H3段子处(ULPI接口)外接高速PHY模块,如USB3320、USB3300 等高速模块



2.2.2.2 代码修改

1. 打开《VCP_Loopback》工程, 选择《STM324xG-EVAL_USBD-HS》





2.2.2.2.1 ULPI 高速模式代码修改

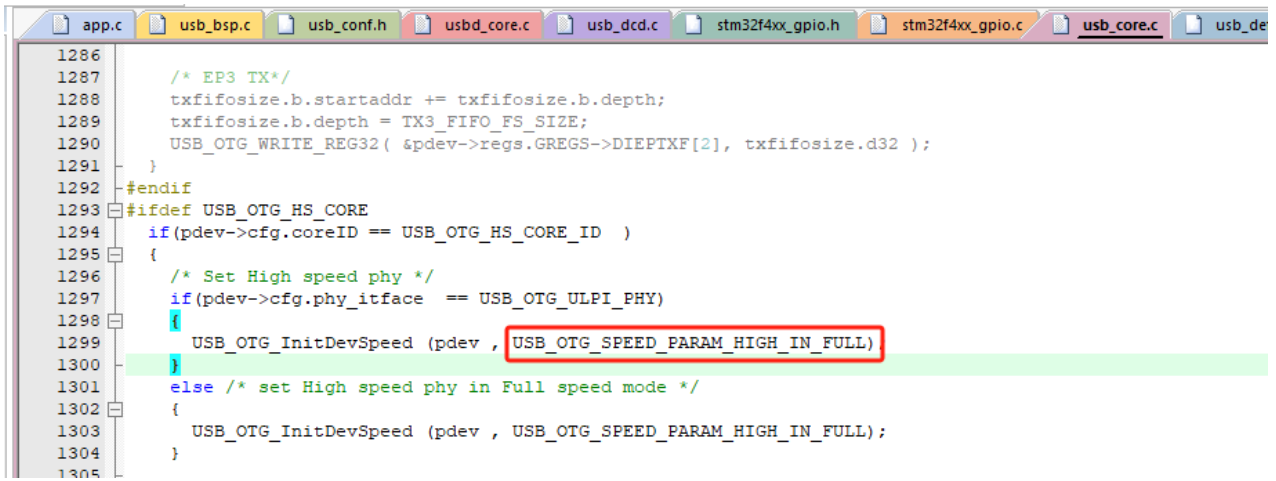
外挂的 USB3318 phy 芯片支持高速和全速模式,通过代码可以修改 USB 只运行在高速模式和全速模式下。

只需修改一下代码,把宏定义《USB_OTG_SPEED_PARAM_HIGH》传入以下函数,即可实现外挂 PHY 运行在高速模式。



2.2.2.2.2 ULPI 全速模式代码修改

只需修改一下代码,把宏定义《USB_OTG_SPEED_PARAM_HIGH_IN_FULL》传入以下函数,即可实现外挂 PHY 运行在全速模式。



```
1286
1287     /* EP3 TX*/
1288     txfifo.b.startaddr += txfifo.b.depth;
1289     txfifo.b.depth = TX3_FIFO_FS_SIZE;
1290     USB_OTG_WRITE_REG32( &pdev->regs.GREGS->DIEPTXF[2], txfifo.d32 );
1291 }
1292 #endif
1293 #ifdef USB_OTG_HS_CORE
1294 if(pdev->cfg.coreID == USB_OTG_HS_CORE_ID )
1295 {
1296     /* Set High speed phy */
1297     if(pdev->cfg.phy_iface == USB_OTG_ULPI_PHY)
1298     {
1299         USB_OTG_InitDevSpeed (pdev , USB_OTG_SPEED_PARAM_HIGH_IN_FULL);
1300     }
1301     else /* set High speed phy in Full speed mode */
1302     {
1303         USB_OTG_InitDevSpeed (pdev , USB_OTG_SPEED_PARAM_HIGH_IN_FULL);
1304     }
1305 }
```

3 OTG 主机模式配置

主机模式配置和设备模式相同，请参考设备模式配置，这里有一些特例，需要单独举例说明

3.1 OTG HS 主机模式 ULPI 接口外挂 PHY 配置

3.1.1 ULPI 高速模式代码修改

改为下面代码，即为高速模式



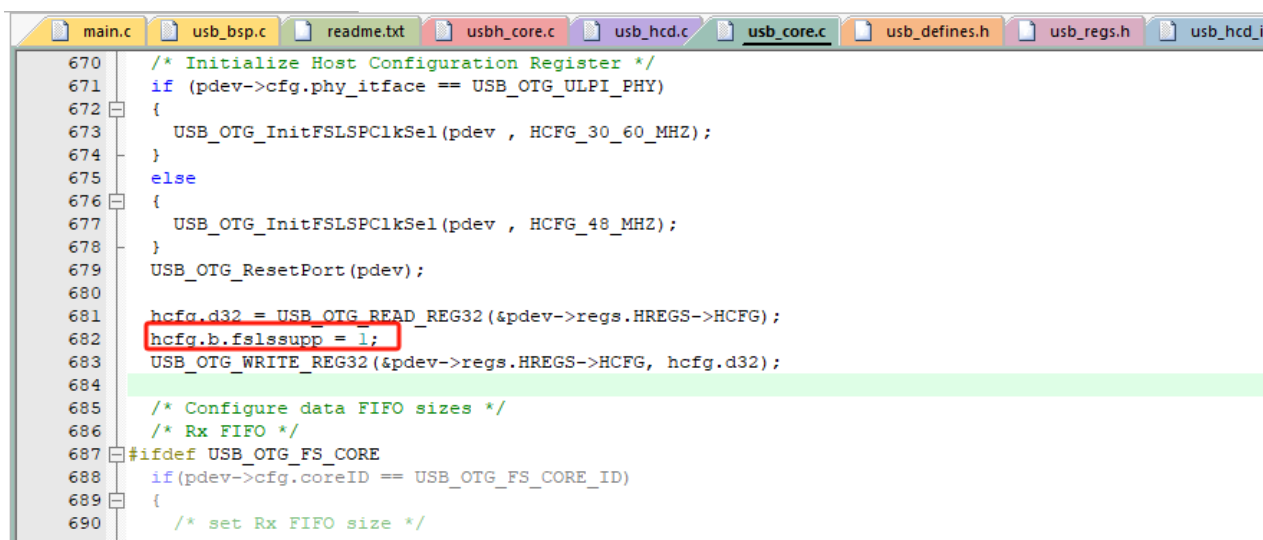
```

670  /* Initialize Host Configuration Register */
671  if (pdev->cfg.phy_iface == USB_OTG_ULPI_PHY)
672  {
673      USB_OTG_InitFSLSPClkSel(pdev , HCFG_30_60_MHZ);
674  }
675  else
676  {
677      USB_OTG_InitFSLSPClkSel(pdev , HCFG_48_MHZ);
678  }
679  USB_OTG_ResetPort(pdev);
680
681  hcfg.d32 = USB_OTG_READ_REG32(&pdev->regs.HREGS->HCFG);
682  hcfg.b.fslssupp = 0;
683  USB_OTG_WRITE_REG32(&pdev->regs.HREGS->HCFG, hcfg.d32);
684
685  /* Configure data FIFO sizes */
686  /* Rx FIFO */
687  #ifdef USB_OTG_FS_CORE
688  if(pdev->cfg.coreID == USB_OTG_FS_CORE_ID)
689  {
690      /* set Rx FIFO size */

```

3.1.2 ULPI 全速模式代码修改

改为下面代码，即为全速模式



```

670  /* Initialize Host Configuration Register */
671  if (pdev->cfg.phy_iface == USB_OTG_ULPI_PHY)
672  {
673      USB_OTG_InitFSLSPClkSel(pdev , HCFG_30_60_MHZ);
674  }
675  else
676  {
677      USB_OTG_InitFSLSPClkSel(pdev , HCFG_48_MHZ);
678  }
679  USB_OTG_ResetPort(pdev);
680
681  hcfg.d32 = USB_OTG_READ_REG32(&pdev->regs.HREGS->HCFG);
682  hcfg.b.fslssupp = 1;
683  USB_OTG_WRITE_REG32(&pdev->regs.HREGS->HCFG, hcfg.d32);
684
685  /* Configure data FIFO sizes */
686  /* Rx FIFO */
687  #ifdef USB_OTG_FS_CORE
688  if(pdev->cfg.coreID == USB_OTG_FS_CORE_ID)
689  {
690      /* set Rx FIFO size */

```