AN1810
应用笔记

UM324xF 移植 CheeryUSB 设备栈

版本：V1.0

广芯微电子（广州）股份有限公司

http://www.unicmicro.com/

条款协议

1. 本文档中所记载的关于电路、软件和其他相关信息仅用于说明半导体产品的操作和应用实例。用户如在设备设计中应用本文档中的电路、软件和相关信息，请自行负责。对于用户或第三方因使用上述电路、软件或信息而遭受的任何损失，广芯微电子不承担任何责任。
2. 在准备本文档所记载的信息的过程中，广芯微电子已尽量做到合理注意，但是，广芯微电子并不保证这些信息都是准确无误的。用户因本文档中所记载的信息的错误或遗漏而遭受的任何损失，广芯微电子不承担任何责任。
3. 对于因使用本文档中的广芯微电子产品或技术信息而造成的侵权行为或因此而侵犯第三方的专利、版权或其他知识产权的行为，广芯微电子不承担任何责任。本文档所记载的内容不应视为对广芯微电子或其他人所有的专利、版权或其他知识产权作出任何明示、默示或其它方式的许可及授权。
4. 使用本文档中记载的广芯微电子产品时，应在广芯微电子指定的范围内，特别是在最大额定值、电源工作电压范围、热辐射特性、安装条件以及其他产品特性的范围内使用。对于在上述指定范围之外使用广芯微电子产品而产生的故障或损失，广芯微电子不承担任何责任。
5. 虽然广芯微电子一直致力于提高广芯微电子产品的质量和可靠性，但是，半导体产品有其自身的具体特性，如一定的故障发生率以及在某些使用条件下会发生故障等。此外，广芯微电子产品均未进行防辐射设计。所以请采取安全保护措施，以避免当广芯微电子产品在发生故障而造成火灾时导致人身事故、伤害或损害的事故。例如进行软硬件安全设计（包括但不限于冗余设计、防火控制以及故障预防等）、适当的老化处理或其他适当的措施等。

目录

1 摘要

本篇应用笔记主要介绍UM324xF移植CheeryUSB设备栈实现打印机类的通信。

本篇应用笔记主要包括：

- CheeryUSB简介
- CheeryUSB下载
- CheeryUSB设备栈移植
- 参考样例
- 鸣谢

注：具体功能及寄存器的操作等相关事项请以用户手册为准。

2 CheeryUSB 简介

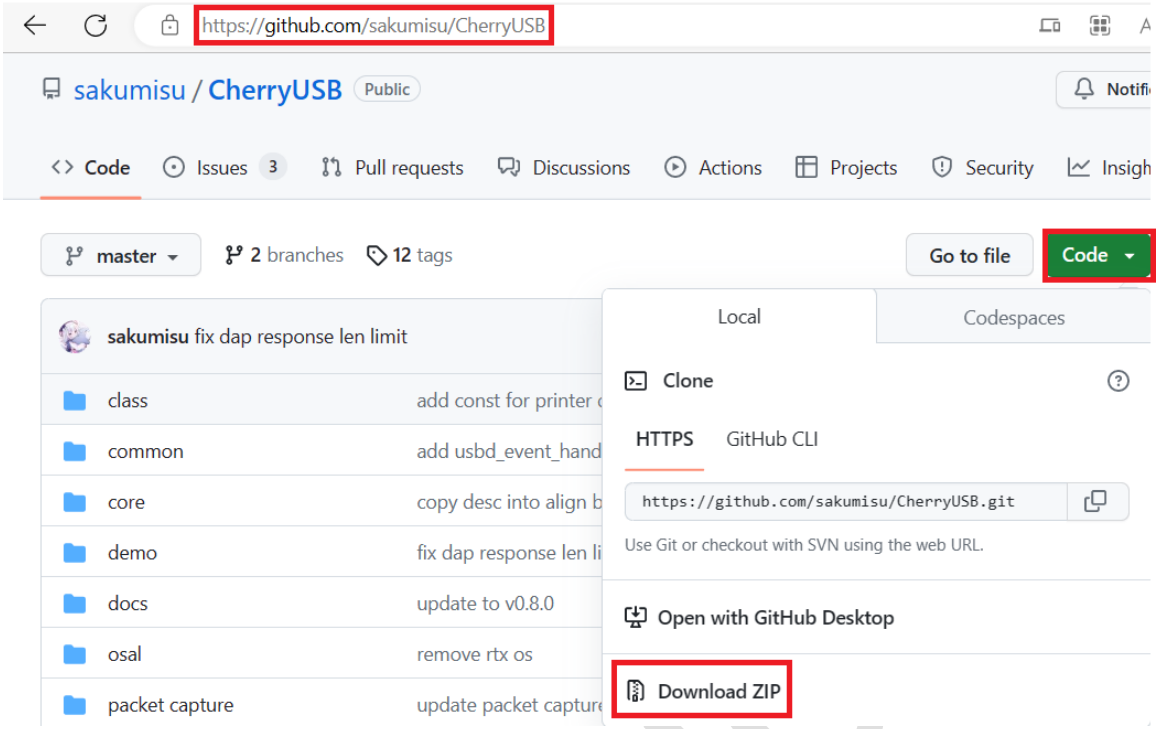
CherryUSB是一个小而美的、可移植性高的、用于嵌入式系统的USB主从协议栈。同时CherryUSB具有以下优点：

- 代码精简，并且内存占用极小，而且还可进一步的裁剪
- 全面的class驱动，并且主从class驱动全部模板化，方便用户增加新的class驱动以及学习的时候查找规律
- 可供用户使用的API非常少，并且分类清晰。从机：初始化+注册、命令回调类、数据收发类；主机：初始化+查找类、数据收发类
- 树状化编程，代码层层递进，方便用户理清函数调用关系、枚举和class驱动加载过程
- 标准化的porting接口，相同ip无需重写驱动，并且porting驱动也进行了模板化，方便用户新增porting CheeryUSB移植
- 主从收发接口的使用等价于UART TX/RX DMA的使用，长度也没有限制
- 能够达到USB硬件理论带宽

3 CheeryUSB 下载

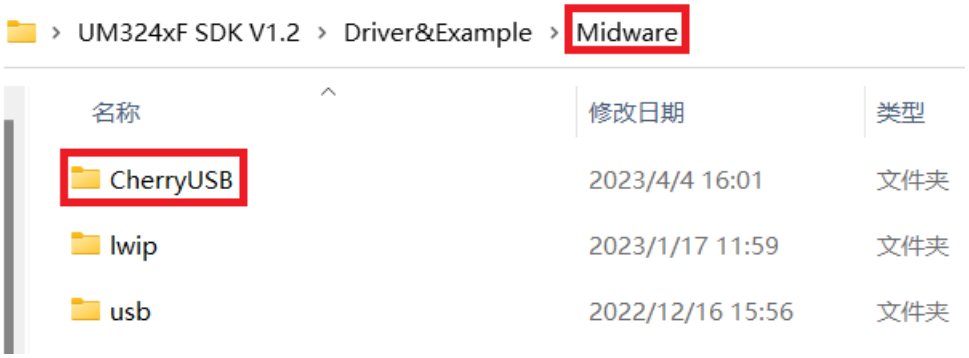
从如下网址获取 CheeryUSB 源码（编写本应用时，下载此源码的时间为 2023-04-04，下载的压缩包为“CherryUSB-master .zip”）

https://github.com/sakumisu/CherryUSB

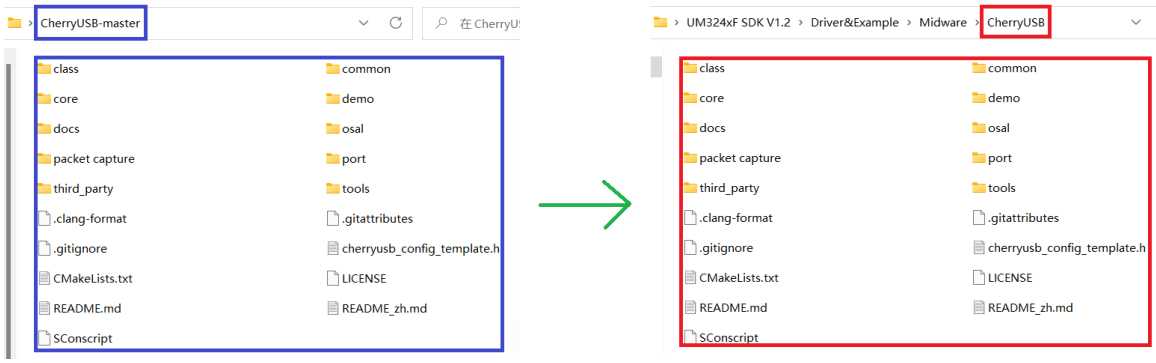


4 CheeryUSB 设备栈移植

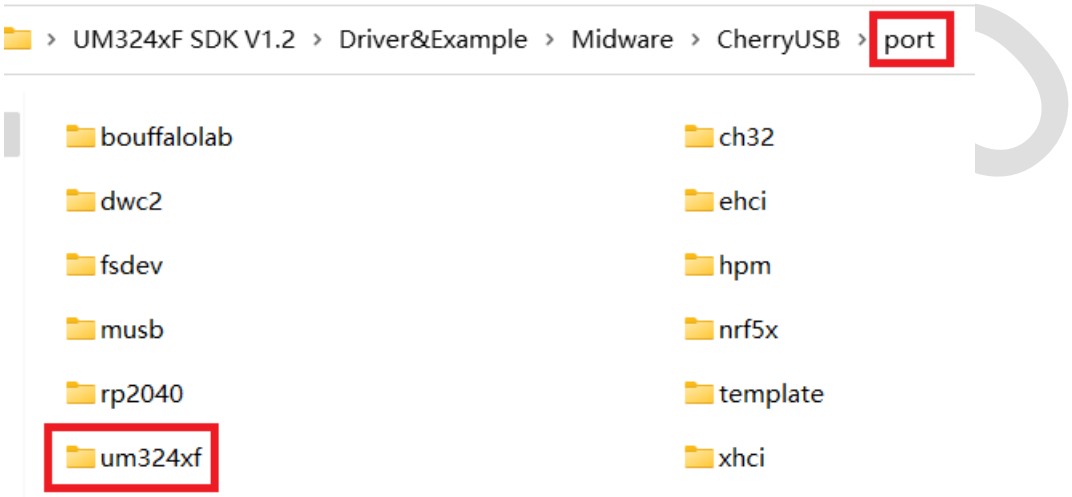
1. 解压“UM324xF SDK V1.2.rar”和上面下载好的“CherryUSB-master.zip”。
2. 在解压开的“UM324xF SDK”的“Midware”目录下新建“CheeryUSB”文件夹。



3. 把解压开的“CheeryUS-master”目录下的全部文件及文件夹 copy 到上面新建的“CheeryUSB”目录下。

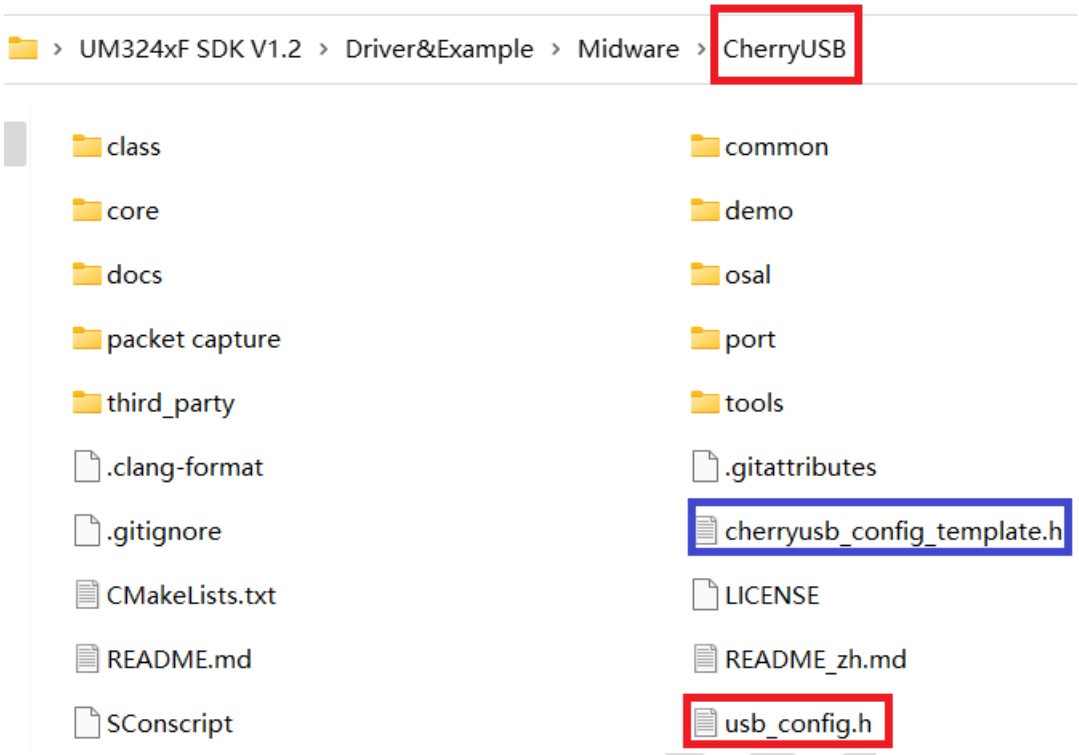


4. 在“port”目录下新建“um324xf”的文件夹。

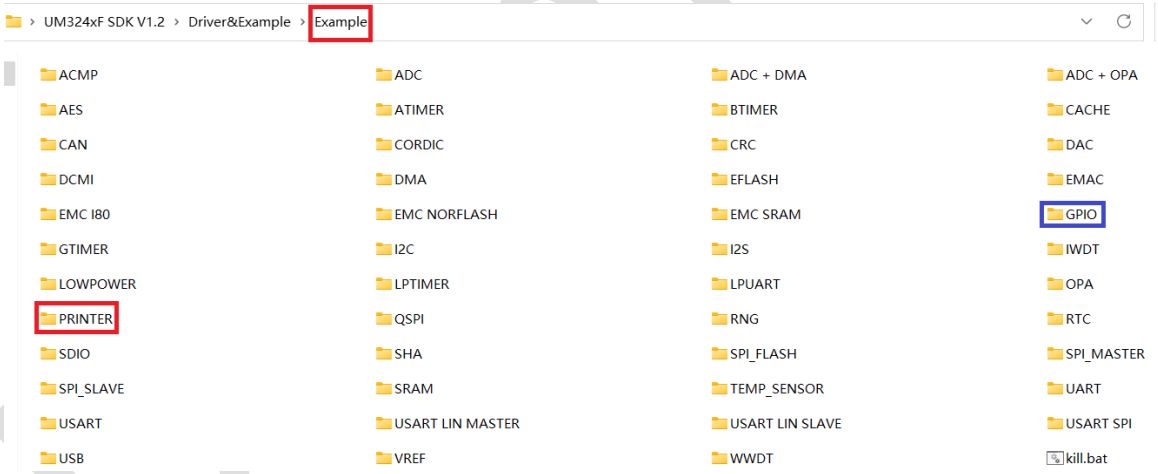


5. 从“template”目录下 copy “usb_dc.c”文件到上面新建的“um324xf”目录下，并且重新命名为“usb_dc_um324xf.c”，此文件是后面移植的重点。

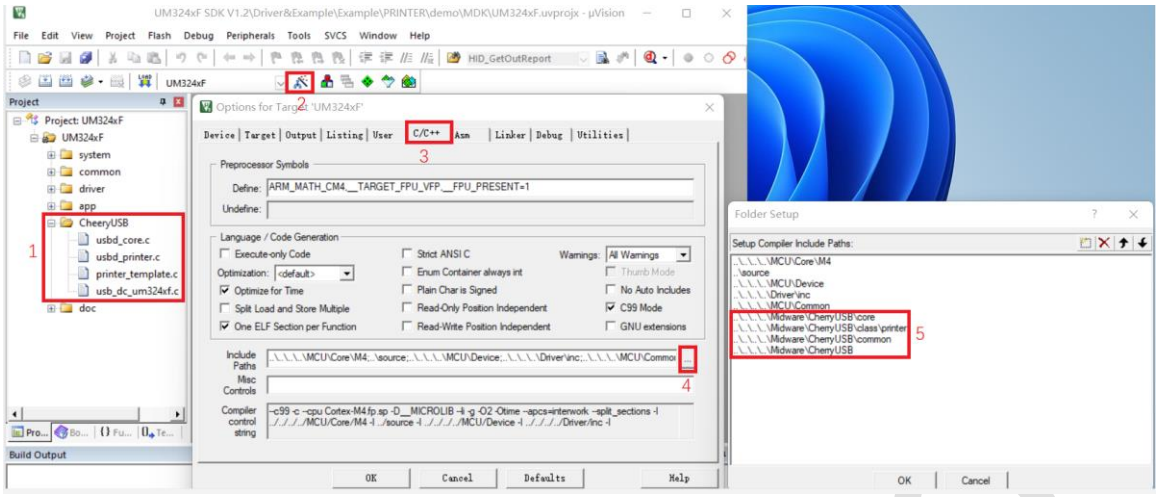
6. 在“CheeryUSB”目录下 copy 一份“cheeryusb_config_template.h”并重新命名为“usb_config.h”。



7. 为方便起见，在“Example”目录下 copy 一份 GPIO 的 demo，并重新命名为“PRINTER”（此应用笔记实例为 USB 打印机）



8. 打开“PRINTER”的工程（此应用笔记移植基于 MDK），新建“CheeryUSB”的 Group，并加入如下 CheeryUSB 设备栈及 printer 要用到的文件，以及增加头文件路径。



CheeryUSB 组下的文件存放目录如下：

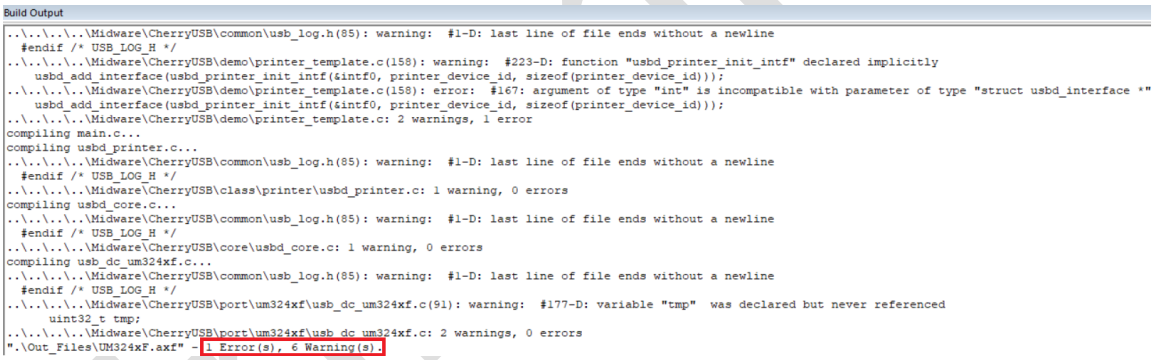
①UM324xF SDK V1.2\Driver&Example\Midware\CherryUSB\core

②UM324xF SDK V1.2\Driver&Example\Midware\CherryUSB\class\printer

③UM324xF SDK V1.2\Driver&Example\Midware\CherryUSB\demo

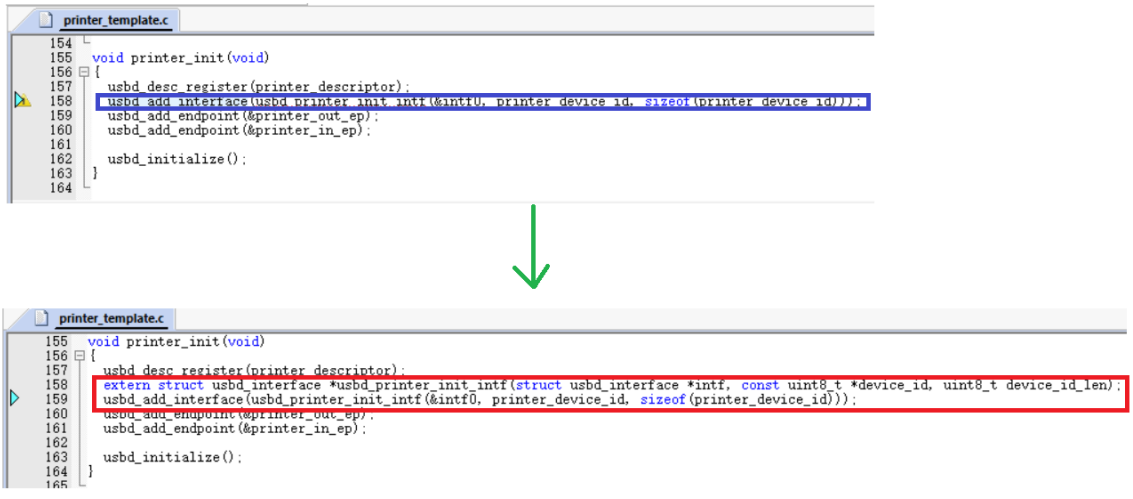
④UM324xF SDK V1.2\Driver&Example\Midware\CherryUSB\port\um324xf

9. 工程的基本的文件及配置已准备好，此时编译会有 1 个错误和 6 个警告。



```
Build Output
..\..\..\..\Midware\CherryUSB\common\usb_log.h(85): warning:  #1-D: last line of file ends without a newline
  #endif /* USB_LOG_H */
..\..\..\..\Midware\CherryUSB\demo\printer_template.c(158): warning:  #223-D: function "usbd_printer_init_intf" declared implicitly
    usbd_add_interface(usbd_printer_init_intf(&intf0, printer_device_id, sizeof(printer_device_id)));
..\..\..\..\Midware\CherryUSB\demo\printer_template.c(158): error:  #167: argument of type "int" is incompatible with parameter of type "struct usbd_interface *"
    usbd_add_interface(usbd_printer_init_intf(&intf0, printer_device_id, sizeof(printer_device_id)));
..\..\..\..\Midware\CherryUSB\demo\printer_template.c: 2 warnings, 1 error
compiling main.c...
compiling usbd_printer.c...
..\..\..\..\Midware\CherryUSB\common\usb_log.h(85): warning:  #1-D: last line of file ends without a newline
  #endif /* USB_LOG_H */
..\..\..\..\Midware\CherryUSB\class\printer\usbd_printer.c: 1 warning, 0 errors
compiling usbd_core.c...
..\..\..\..\Midware\CherryUSB\common\usb_log.h(85): warning:  #1-D: last line of file ends without a newline
  #endif /* USB_LOG_H */
..\..\..\..\Midware\CherryUSB\core\usbd_core.c: 1 warning, 0 errors
compiling usb_dc_um324xf.c...
..\..\..\..\Midware\CherryUSB\common\usb_log.h(85): warning:  #1-D: last line of file ends without a newline
  #endif /* USB_LOG_H */
..\..\..\..\Midware\CherryUSB\port\um324xf\usb_dc_um324xf.c(91): warning:  #177-D: variable "tmp"  was declared but never referenced
      uint32_t tmp;
..\..\..\..\Midware\CherryUSB\port\um324xf\usb_dc_um324xf.c: 2 warnings, 0 errors
".\Out_Files\UM324xF.axf" - 1 Error(s), 6 Warning(s).
```

10. 先解决错误，后解决警告，定位到错误处，声明下“usbd_printer_init_intf”函数错误解决。



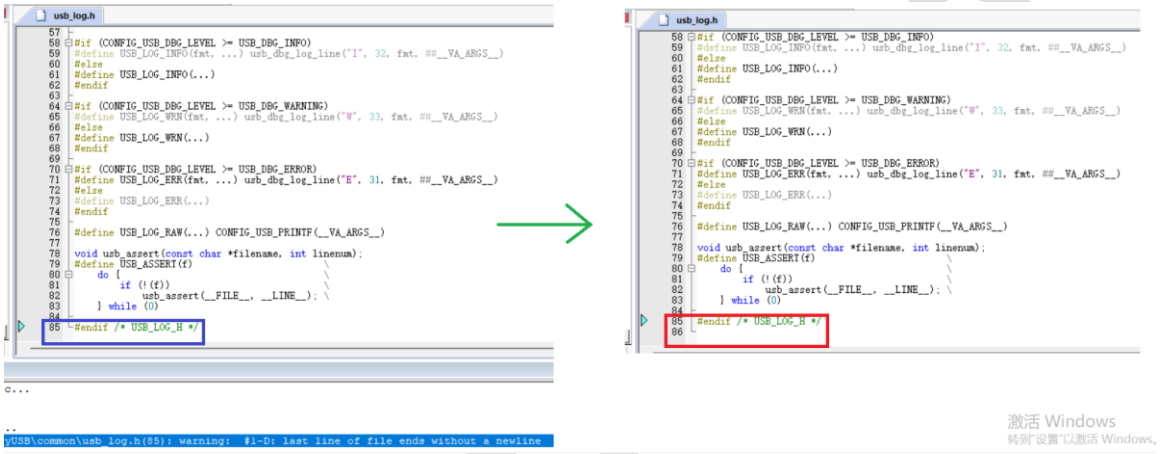
```
printer_template.c
154
155  void printer_init(void)
156  {
157    usbd_desc_register(printer_descriptor);
158    usbd_add_interface(usbd_printer_init_intf(&intf0, printer_device_id, sizeof(printer_device_id)));
159    usbd_add_endpoint(&printer_out_ep);
160    usbd_add_endpoint(&printer_in_ep);
161
162    usbd_initialize();
163  }
164
```

```
printer_template.c
155  void printer_init(void)
156  {
157    usbd_desc_register(printer_descriptor);
158    extern struct usbd_interface *usbd_printer_init_intf(struct usbd_interface *intf, const uint8_t *device_id, uint8_t device_id_len);
159    usbd_add_interface(usbd_printer_init_intf(&intf0, printer_device_id, sizeof(printer_device_id)));
160    usbd_add_endpoint(&printer_out_ep);
161    usbd_add_endpoint(&printer_in_ep);
162
163    usbd_initialize();
164  }
165
```

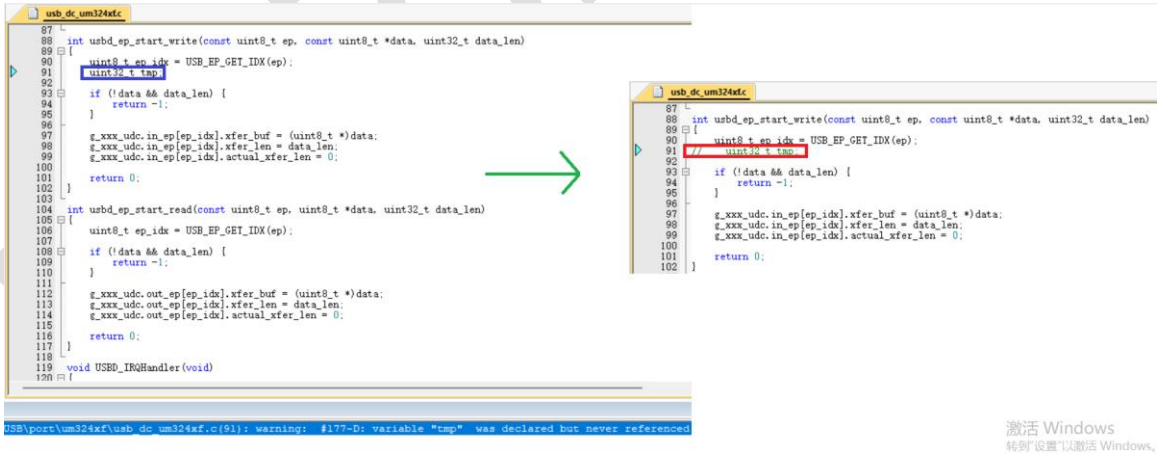
11. 上面的错误解决后，再编译，还有 5 个警告，为文件最后一行没有空白行和定义的变量没有引用。

```
Build Output
..\..\..\..\Midware\CherryUSB\common\usb_log.h(85): warning:  #1-D: last line of file ends without a newline
  #endif /* USB_LOG_H */
..\..\..\..\Midware\CherryUSB\class\printer\usbd_printer.c: 1 warning, 0 errors
compiling printer_template.c...
..\..\..\..\Midware\CherryUSB\common\usb_log.h(85): warning:  #1-D: last line of file ends without a newline
  #endif /* USB_LOG_H */
..\..\..\..\Midware\CherryUSB\demo\printer_template.c: 1 warning, 0 errors
compiling usb_dc_um324xf.c...
..\..\..\..\Midware\CherryUSB\common\usb_log.h(85): warning:  #1-D: last line of file ends without a newline
  #endif /* USB_LOG_H */
..\..\..\..\Midware\CherryUSB\port\um324xf\usb_dc_um324xf.c(91): warning:  #177-D: variable "tmp"  was declared but never referenced
      uint32_t tmp;
..\..\..\..\Midware\CherryUSB\port\um324xf\usb_dc_um324xf.c: 2 warnings, 0 errors
compiling usbd_core.c...
..\..\..\..\Midware\CherryUSB\common\usb_log.h(85): warning:  #1-D: last line of file ends without a newline
  #endif /* USB_LOG_H */
..\..\..\..\Midware\CherryUSB\core\usbd_core.c: 1 warning, 0 errors
linking...
Program Size: Code=4652 RO-data=572 RW-data=44 ZI-data=10332
FromELF: creating hex file...
".\Out_Files\UM324xF.axf" - 0 Error(s), 5 Warning(s).
```

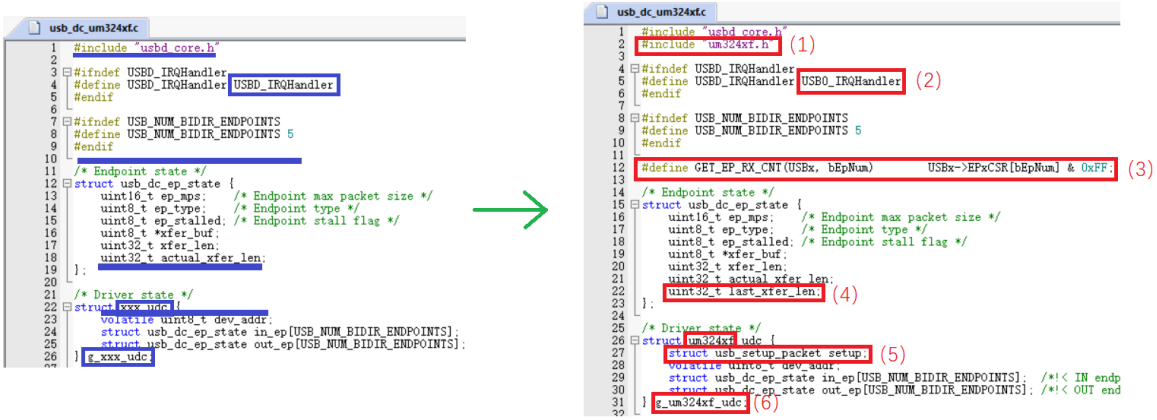
12. 定位到最后一行没空白行的文件，加上空白行，即可解决。



13. 定位到变量没引用的文件，把此变量屏蔽，即可解决。



14. 此步骤开始是移植的重点，主要是 UM324xF 的 usb 底层的实现，即“usb_dc_um324xf.c”文件的具体实现。



A. 包含 um324xf 的头文件进来。

B. 启动文件中 USB 中断入口名称为 USB0_IRQHandler，所以这里改为一样。

C. 获取端点接收的字节数。

D. 用来记录上一包数据传输的字节数。

E. 存放 setup 数据包。

F. 改下名称，尽量保持 CheeryUSB 的风格。

15. “usb_dc_low_level_init”函数实现，主要是使能 USB 时钟、打开相应的中断、使能 DP 的上拉电阻。

usb_dc_um324xf.c

```
__WEAK void usb_dc_low_level_init(void)
{
  RCM_HANDLE->RCMPR = 0xA5A55A5A;          //RCM寄存器写使能打开

  RCM_HANDLE->AHB1CKENR |= (1<<0);         //GPIOA控制器时钟使能
  RCM_HANDLE->AHB1RSTR &= (~(1<<0));       //GPIOA控制器不复位

  RCM_HANDLE->AHBOCKENR |= (1<<0);         //USB控制器时钟使能
  RCM_HANDLE->AHBORSTR &= (~(1<<0));       //USB控制器不复位

  RCM_HANDLE->RCMPR = 0xFFFFFFFF;          //RCM寄存器写使能关闭

  GPIOA_HANDLE->MODE |= (3<<(11*2));       //PA11-DM模拟模式
  GPIOA_HANDLE->MODE |= (3<<(12*2));       //PA12-DP模拟模式

  USB_HANDLE->WORKINGMODE |= (1<<2);       //复位USB
  USB_HANDLE->WORKINGMODE &= (~(1<<2));    //不复位USB
  USB_HANDLE->INTEN = 0;                   //USB中断禁止

  USB_HANDLE->WORKINGMODE |= (1<<0);       //全速模式

  USB_HANDLE->WORKINGMODE |= (1<<3);       //复位状态机使能
  USB_HANDLE->WORKINGMODE |= (1<<23);
  USB_HANDLE->WORKINGMODE |= (1<<24);

  USB_HANDLE->WORKINGMODE |= (1<<25);      (1)

  //Host Reset中断使能
  //Ep0 Ack中断使能
  //Ep1 Ack中断使能
  //Ep2 Ack中断使能
  //Ep3 Ack中断使能
  //Ep4 Ack中断使能
  USB_HANDLE->INTEN = (1<<0) | (1<<8) | (1<<11) | (1<<14) | (1<<17) | (1<<20) | (1<<25);   (2)

  USB_HANDLE->INTCLR = 0xFFFFFFFF;         //清除中断标志位
  NVIC_ClearPendingIRQ(USB0_IRQn);
  NVIC_EnableIRQ(USB0_IRQn);               //使能USB总中断

  USB_HANDLE->WORKINGMODE |= (1<<6)|(1<<4); //打开DP上拉电阻
}
```

A. 打开 EP0 OUT 的 0 长度数据包中断功能。

B. 此 demo 的 SETUP、IN、OUT 都利用 ACK 中断来处理，um324xf 没有发送完成中断，但 Host 如果回应了 ACK，那说明正确传输完成了。此实例 printer 只用了端点 0（控制传输）、端点 1（BLUK IN），端点 2（BLUK OUT）。

16. “usb_dc_low_level_deinit”函数实现，类似“usb_dc_low_level_init”的反向操作。

usb_dc_um324xf.c

```
75  __WEAK void usb_dc_low_level_deinit(void)
76  {
77    USB_HANDLE->WORKINGMODE |= (1<<2);      //复位USB
78    USB_HANDLE->INTEN = 0;                  //USB中断禁止
79    USB_HANDLE->INTCLR = 0xFFFFFFFF;        //清除中断标志位
80
81    RCM_HANDLE->RCMPR = 0xA5A55A5A;         //RCM寄存器写使能打开
82    RCM_HANDLE->AHBOCKENR &= (~(1<<0));     //USB控制器时钟禁止
83    RCM_HANDLE->AHBORSTR |= (1<<0);         //USB控制器复位
84    RCM_HANDLE->RCMPR = 0xFFFFFFFF;         //RCM寄存器写使能关闭
85
86    NVIC_DisableIRQ(USB0_IRQn);             //关闭USB总中断
87
88    USB_HANDLE->WORKINGMODE &= (~((1<<6)|(1<<4))); //关闭DP上拉电阻
89  }
```

17. “usb_dc_low_level_init”函数由“usb_dc_init”函数调用，“usb_dc_low_level_deinit”由“usb_dc_deinit”函数调用。

usb_dc_um324xf.c

```
90
91  int usb_dc_init(void)
92  {
93      usb_dc_low_level_init();
94      return 0;
95  }
96
97  int usb_dc_deinit(void)
98  {
99      usb_dc_low_level_deinit();
100     return 0;
101 }
102
103 int usbd_set_address(const uint8_t addr)
104 {
105     g_um324xf_udc.dev_addr = addr;
106     return 0;
107 }
```

(1)

注：um324xf 的设置地址由硬件自动完成，不用用户去设置 USB 地址的寄存器。

18. 增加“usbd_get_port_speed”函数。

usb_dc_um324xf.c

```
108
109 uint8_t usbd_get_port_speed(const uint8_t port)
110 {
111   return USB_SPEED_FULL;
112 }
```

19. 实现端点打开的函数。

usb_dc_um324xf.c

```
114  int usbd_ep_open(const struct usbd_endpoint_cfg *ep_cfg)
115  {
116      uint8_t ep_idx = USB_EP_GET_IDX(ep_cfg->ep_addr);
117
118      if (ep_idx > (USB_NUM_BIDIR_ENDPOINTS - 1))
119      {
120        USB_LOG_ERR("Ep addr %d overflow\r\n", ep_cfg->ep_addr);
121        return -1;
122      }
123
124      if (USB_EP_DIR_IS_OUT(ep_cfg->ep_addr))
125      {
126        g_um324xf_udc.out_ep[ep_idx].ep_mps = ep_cfg->ep_mps;
127        g_um324xf_udc.out_ep[ep_idx].ep_type = ep_cfg->ep_type;
128      }
129      else
130      {
131        g_um324xf_udc.in_ep[ep_idx].ep_mps = ep_cfg->ep_mps;
132        g_um324xf_udc.in_ep[ep_idx].ep_type = ep_cfg->ep_type;
133      }
134
135      USB_HANDLE->EPxCSR[ep_idx] |= (1<<8);        //端点使能
136
137      return 0;
138  }
```

20. 实现端点关闭的函数。

usb_dc_um324xf.c

```
139
140  int usbd_ep_close(const uint8_t ep)
141  {
142      uint8_t ep_idx = USB_EP_GET_IDX(ep);
143
144      USB_HANDLE->EPxCSR[ep_idx] &= (~(1<<8));         //端点关闭
145
146      return 0;
147  }
```

21. 实现发送和不发送 STALL 的函数。

usb_dc_um324xf.c

```c
int usbd_ep_set_stall(const uint8_t ep)
{
    uint8_t ep_idx = USB_EP_GET_IDX(ep);

    if (ep_idx > (USB_NUM_BIDIR_ENDPOINTS - 1))
    {
      USB_LOG_ERR("Ep addr %d overflow\r\n", ep);
      return -1;
    }

    USB_HANDLE->EPxCSR[ep_idx] |= (1<<12);      //SET STALL

    return 0;
}

int usbd_ep_clear_stall(const uint8_t ep)
{
    uint8_t ep_idx = USB_EP_GET_IDX(ep);

    if (ep_idx > (USB_NUM_BIDIR_ENDPOINTS - 1))
    {
      USB_LOG_ERR("Ep addr %d overflow\r\n", ep);
      return -1;
    }

    USB_HANDLE->EPxCSR[ep_idx] &= (~(1<<12));   //CLEAR STALL

    return 0;
}
```

22. 实现发送数据的函数。

usb_dc_um324xf.c

```c
int usbd_ep_start_write(const uint8_t ep, const uint8_t *data, uint32_t data_len)
{
    uint8_t *src, *dst;
    uint8_t ep_idx = USB_EP_GET_IDX(ep);

    if (!data && data_len)
    {
      return -1;
    }

    g_um324xf_udc.in_ep[ep_idx].xfer_buf = (uint8_t *)data;
    g_um324xf_udc.in_ep[ep_idx].xfer_len = data_len;
    g_um324xf_udc.in_ep[ep_idx].actual_xfer_len = 0;

    data_len = MIN(data_len, g_um324xf_udc.in_ep[ep_idx].ep_mps);
    g_um324xf_udc.in_ep[ep_idx].last_xfer_len = data_len;

    src = (uint8_t *)data;
    dst = (uint8_t *)(&(USB_HANDLE->EPxMEM[ep_idx][0]));
    for (; data_len; data_len--)
    {
      *dst++ = *src++;
    }

    USB_HANDLE->EPxCSR[ep_idx] &= (~(1<<12));
    USB_HANDLE->EPxSENDBN[ep_idx]= g_um324xf_udc.in_ep[ep_idx].last_xfer_len;
    USB_HANDLE->EPxCSR[ep_idx] |= (1<<10);

    return 0;
}
```

23. 实现接收数据的函数。

usb_dc_um324xf.c

```c
int usbd_ep_start_read(const uint8_t ep, uint8_t *data, uint32_t data_len)
{
    uint8_t ep_idx = USB_EP_GET_IDX(ep);

    if (!data && data_len) {
        return -1;
    }

    g_um324xf_udc.out_ep[ep_idx].xfer_buf = (uint8_t *)data;
    g_um324xf_udc.out_ep[ep_idx].xfer_len = data_len;
    g_um324xf_udc.out_ep[ep_idx].actual_xfer_len = 0;

    USB_HANDLE->EPxCSR[ep_idx] &= (~(1<<12));
    USB_HANDLE->EPxCSR[ep_idx] |= (1<<11); //set rx ready

    return 0;
}
```

24. 实现 USB 中断处理的函数，此实例 printer 只用了端点 0（控制传输）、端点 1（BLUK IN），端点 2（BLUK OUT）。

usb_dc_um324xf.c

```c
void USBD_IRQHandler(void)
{
  uint32_t wIstr = USB_HANDLE->INTSTATRAW;

  //Reset
  if(wIstr & (1<<0))
  {
    USB_HANDLE->INTCLR = (1<<0);

    memset(&g_um324xf_udc, 0, sizeof(struct um324xf_udc));
    usbd_event_reset_handler();

    USB_HANDLE->WORKINGMODE |= (1<<25);
  }

  //EP0
  EP0_Handler(wIstr);

  //EP1
  EP1_Handler(wIstr);

  //EP2
  EP2_Handler(wIstr);

  //EP3
  EP3_Handler(wIstr);

  //EP4
  EP4_Handler(wIstr);
}
```

25. 实现 EP0 的处理。

usb_dc_um324xf.c

```
232
233 void EP0_Handler(uint32_t wIstr)
234 {
235   uint8_t *src, *dst;
236   uint8_t read_count, write_count, size;
237
238   if (wIstr & (1<<8))
239   {
240     //Setup
241     if (wIstr & (1<<5))
242     {
243       USB_HANDLE->INTCLR = (1<<5) | (1<<7) | (1<<8);
244
245       g_um324xf_udc.setup.bmRequestType = (USB_HANDLE->SETUP03DATA >> 0);
246       g_um324xf_udc.setup.bRequest = (USB_HANDLE->SETUP03DATA >> 8);
247       g_um324xf_udc.setup.wValue = (USB_HANDLE->SETUP03DATA >> 16);
248       g_um324xf_udc.setup.wIndex = (USB_HANDLE->SETUP47DATA >> 0);
249       g_um324xf_udc.setup.wLength = (USB_HANDLE->SETUP47DATA >> 16);
250
251       usbd_event_ep0_setup_complete_handler((uint8_t *)&g_um324xf_udc.setup);
252     }
253     //OUT
254     else if (wIstr & (1<<7))
255     {
256       USB_HANDLE->INTCLR = (1<<7) | (1<<8);
257
258       read_count = GET_EP_RX_CNT(USB_HANDLE, 0);
259       size = read_count;
260
261       src = (uint8_t *)(&(USB_HANDLE->EPxMEM[0][0]));
262       dst = g_um324xf_udc.out_ep[0].xfer_buf;
263       for (; size; size--)
264       {
265         *dst++ = *src++;
266       }
267
268       g_um324xf_udc.out_ep[0].xfer_buf += read_count;
269       g_um324xf_udc.out_ep[0].xfer_len -= read_count;
270       g_um324xf_udc.out_ep[0].actual_xfer_len += read_count;
271
272       usbd_event_ep_out_complete_handler(0, g_um324xf_udc.out_ep[0].actual_xfer_len);
273
274       if (read_count == 0)
275       {
276         /* Out status, start reading setup */
277         usbd_ep_start_read(0x00, NULL, 0);
278       }
279     }
```

```
280     //IN
281     else
282     {
283       USB_HANDLE->INTCLR = (1<<8);
284
285       write_count = g_um324xf_udc.in_ep[0].last_xfer_len;
286
287       g_um324xf_udc.in_ep[0].xfer_buf += write_count;
288       g_um324xf_udc.in_ep[0].xfer_len -= write_count;
289       g_um324xf_udc.in_ep[0].actual_xfer_len += write_count;
290
291       usbd_event_ep_in_complete_handler(0x80, g_um324xf_udc.in_ep[0].actual_xfer_len);
292
293       if (g_um324xf_udc.setup.wLength == 0)
294       {
295         /* In status, start reading setup */
296         usbd_ep_start_read(0x00, NULL, 0);
297       }
298       else if (g_um324xf_udc.setup.wLength && ((g_um324xf_udc.setup.bmRequestType & USB_REQUEST_DIR_MASK) == USB_REQUEST_DIR_OUT))
299       {
300         /* In status, start reading setup */
301         usbd_ep_start_read(0x00, NULL, 0);
302       }
303
304       if ((g_um324xf_udc.dev_addr > 0U) && (write_count == 0U))
305       {
306         g_um324xf_udc.dev_addr = 0U;
307       }
308     }
309   }
310 }
```

26. 实现 EP1 的处理。

usb_dc_um324xf.c

```
}

//PRINTER_BLUK_IN
void EP1_Handler(uint32_t wIstr)
{
  uint8_t *src, *dst;
  uint8_t write_count, size;

  if (wIstr & (1<<11))
  {
    USB_HANDLE->INTCLR = (1<<11);

    write_count = g_um324xf_udc.in_ep[1].last_xfer_len;

    g_um324xf_udc.in_ep[1].xfer_buf += write_count;
    g_um324xf_udc.in_ep[1].xfer_len -= write_count;
    g_um324xf_udc.in_ep[1].actual_xfer_len += write_count;

    if (g_um324xf_udc.in_ep[1].xfer_len == 0)
    {
      usbd_event_ep_in_complete_handler(0x81, g_um324xf_udc.in_ep[1].actual_xfer_len);
    }
    else
    {
      write_count = MIN(g_um324xf_udc.in_ep[1].xfer_len, g_um324xf_udc.in_ep[1].ep_mps);
      size = write_count;
      g_um324xf_udc.in_ep[1].last_xfer_len = write_count;
      src = (uint8_t *)g_um324xf_udc.in_ep[1].xfer_buf;
      dst = (uint8_t *)(&(USB_HANDLE->EPxMEM[1][0]));
      for (; size; size--)
      {
        *dst++ = *src++;
      }

      USB_HANDLE->EPxCSR[1] &= (~(1<<12));
      USB_HANDLE->EPxSENDBN[1]= g_um324xf_udc.in_ep[1].last_xfer_len;
      USB_HANDLE->EPxCSR[1] |= (1<<10);
    }
  }
}
```

27. 实现 EP2 的处理。

usb_dc_um324xf.c

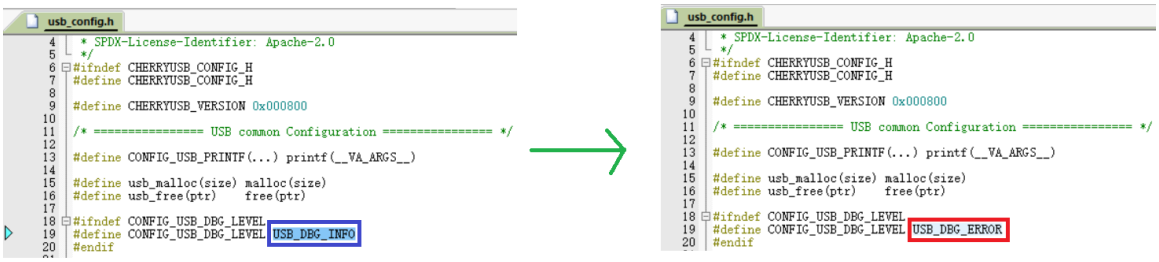
```
}

//PRINTER_BLUK_OUT
void EP2_Handler(uint32_t wIstr)
{
  uint8_t *src, *dst;
  uint8_t read_count, size;

  if (wIstr & (1<<14))
  {
    USB_HANDLE->INTCLR = (1<<13) | (1<<14);

    read_count = GET_EP_RX_CNT(USB_HANDLE, 2);
    size = read_count;

    src = (uint8_t *)(&(USB_HANDLE->EPxMEM[2][0]));
    dst = g_um324xf_udc.out_ep[2].xfer_buf;
    for (; size; size--)
    {
      *dst++ = *src++;
    }

    g_um324xf_udc.out_ep[2].xfer_buf += read_count;
    g_um324xf_udc.out_ep[2].xfer_len -= read_count;
    g_um324xf_udc.out_ep[2].actual_xfer_len += read_count;


    if ((read_count < g_um324xf_udc.out_ep[2].ep_mps) || (g_um324xf_udc.out_ep[2].xfer_len == 0))
    {
      usbd_event_ep_out_complete_handler(2, g_um324xf_udc.out_ep[2].actual_xfer_len);
    }
    else
    {
      USB_HANDLE->EPxCSR[2] |= (1<<11); //set rx ready
    }
  }
}
```

28. EP3、EP4 此实例没使用，空函数即可。

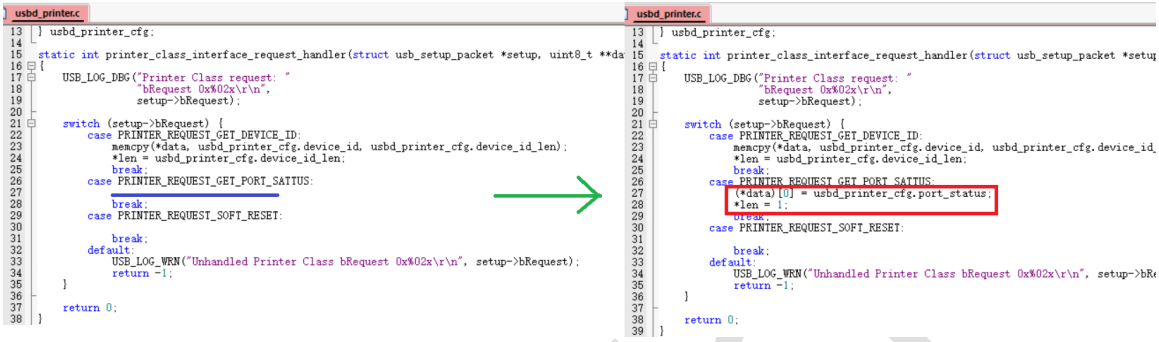
usb_dc_um324xf.c

```
}

void EP3_Handler(uint32_t wIstr)
{
}

void EP4_Handler(uint32_t wIstr)
{
}
```

29. 为避免打印信息影响到 USB 通信，把打印等级改为 USB_DBG_ERROR。



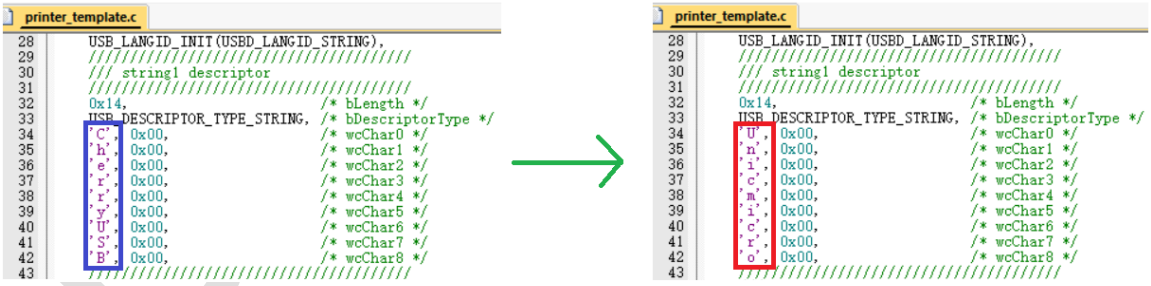
30. 增加返回打印机端口状态的处理。



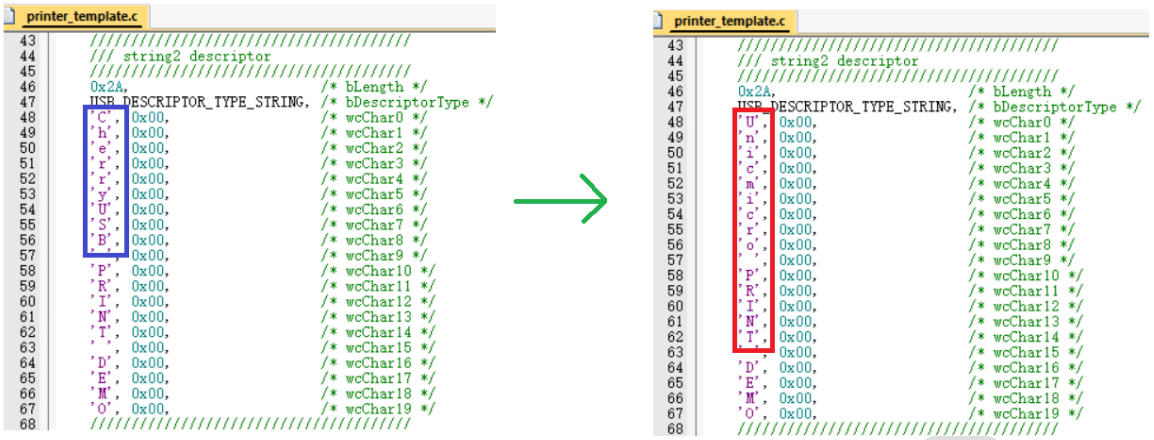
打印机端口状态定义如下

Bit(s)	Field	Description
7 .. 6	Reserved	Reserved for future use; device shall return these bits reset to zero.
5	Paper Empty	1 = Paper Empty, 0 = Paper Not Empty
4	Select	1 = Selected, 0 = Not Selected
3	Not Error	1 = No Error, 0 = Error
2 .. 0	Reserved	Reserved for future use; device shall return these bits reset to zero.

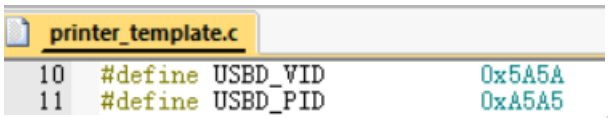
31. 此实例厂商字符串改为“Unicmicro”，注意 bLength 值要根据实际长度给出值。



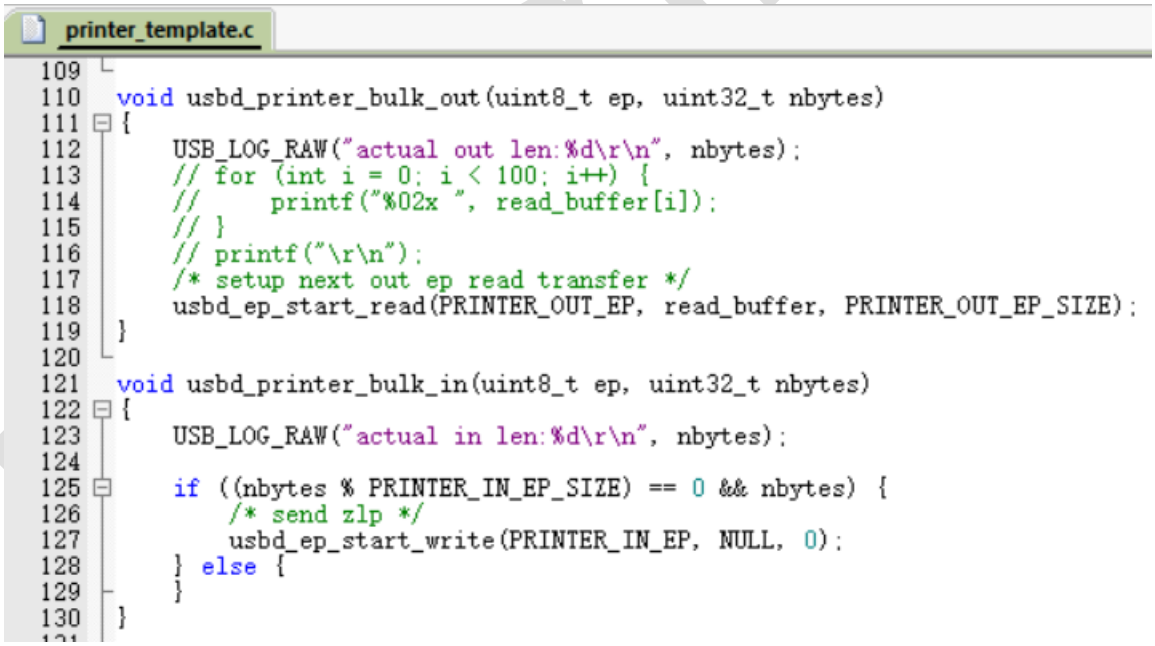
32. 此实例产品字符串改为“Unicmicro PRINT DEMO”，注意 bLength 值要根据实际长度给出值



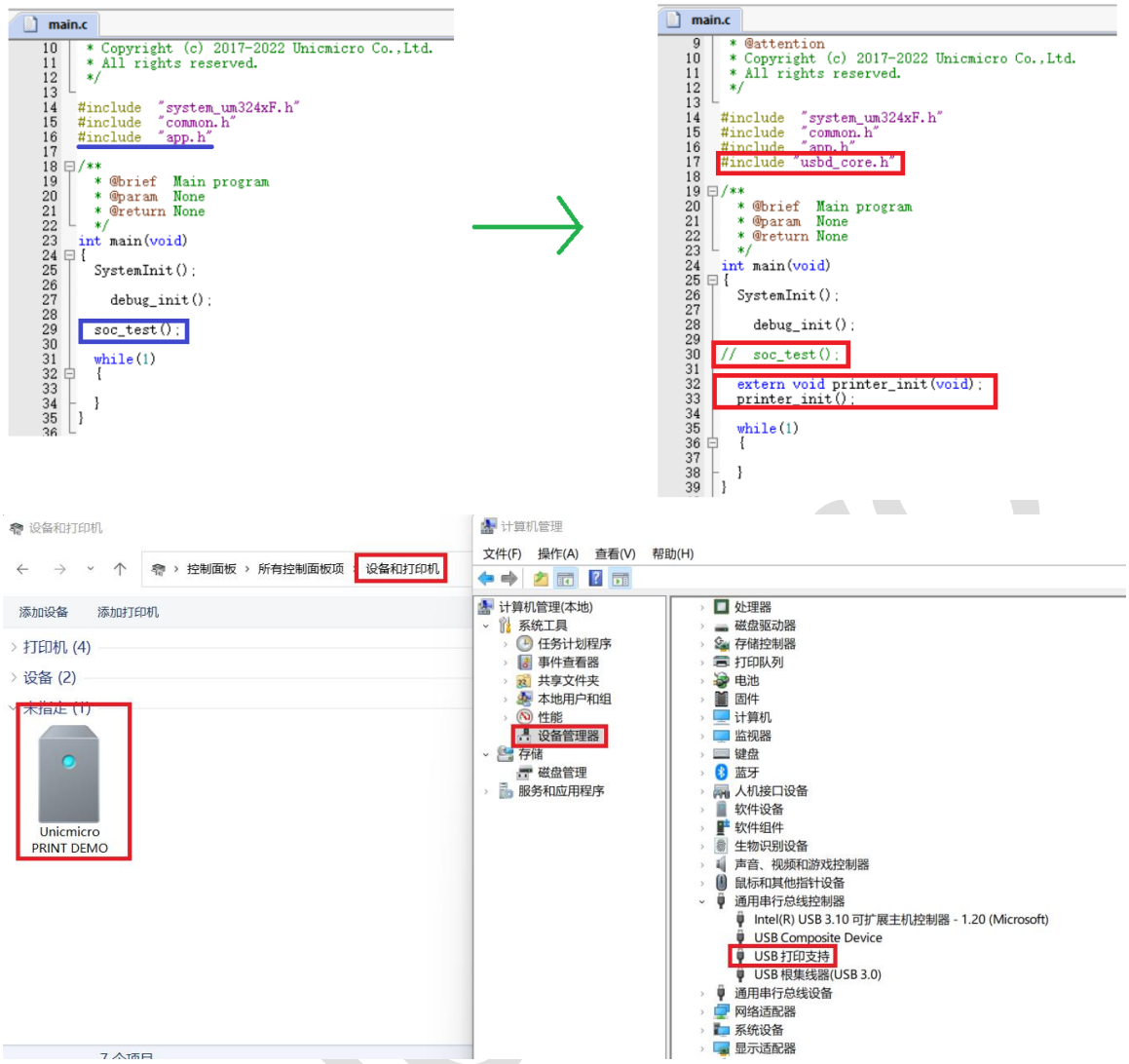
33. VID 是由供应商向 USB-IF(Implementers Forum 应用者论坛)申请。每一个供应商的 VID 是唯一，PID 是由供应商自行决定。此实例按照 CheeryUSB 的，没去修改。



34. 接收到数据进入 “usbd_printer_bulk_out” 函数，数据个数为 nbytes，暂存在 read_buffer 缓冲区，要发送数据，调用“usbd_ep_start_write”函数，数据发送完毕后进入“usbd_printer_bulk_in”函数。



35. 一切准备就绪，在 main 函数里调用打印机初始化的函数，然后编译下载，接上 USB 数据线连接电脑，已可以识别到打印机（本人电脑 windows 11）。



36. 对应此实例打印机 ID 的驱动程序，只找到 XP 系统的，在 XP 系统安装好驱动后，接上 USB 数据线连接电脑

5 参考样例

此应用笔记的 demo 及打印机驱动（XP 系统）请联系我司 FAE 人员索要。

6 鸣谢

感谢 CheeryUSB 的作者 sakumisu 大神提供高性能的、可免费使用的、开源的 USB 协议栈，免去开发人员去详读繁琐复杂的 USB 协议规范，从而降低了嵌入式软件工程师的开发难度。

7 版本修订

版本	日期	描述
V1.0	2023.04.04	初始版