# 应用笔记

# Application Note

文档编号：**AN1163**

## APM32F402_F403_USB CDC 日志输出

版本：**V1.0**

# 1 引言

在 MCU 项目开发过程中，开发者常使用 UART 串口来打印调试信息。然而，当项目变得复杂时，UART 方案会面临线缆繁多、资源占用大以及带宽有限等问题。因此，开发者会寻求更优的解决方案，例如利用 USB 的特性，实现数据传输和日志打印的统一，仅通过一根线缆即可完成。

本应用笔记主要介绍如何利用 APM32F402/F403 芯片的 USB OTG 功能，将其配置为一个“复合设备 (Composite Device)”，使其同时支持 WinUSB（用于高速数据传输）和 CDC（虚拟串口，用于日志输出）两个接口。通过这种方式，仅需一根 USB 数据线，便可同时实现供电、数据交换和调试日志输出，实现“一线三用”的便捷开发体验。

例如，一个数据记录器（Data Logger）需要实时或定期采集传感器数据（如压力、流量、震动等）。现场维护人员可以通过笔记本电脑与设备建立 WinUSB 通道，高速下载大量的历史数据；同时，调试工程师可以在终端软件中打开 CDC 虚拟串口，实时查看 MCU 的运行状态或告警信息。整个过程仅需一根 USB 线，简化了物理连接，且不影响高带宽的数据传输。

本应用笔记适用于极海 APM32F402/F403，以下正文以 APM32F402 为例。

# 目录

# 2 选择 USB CDC 的原因

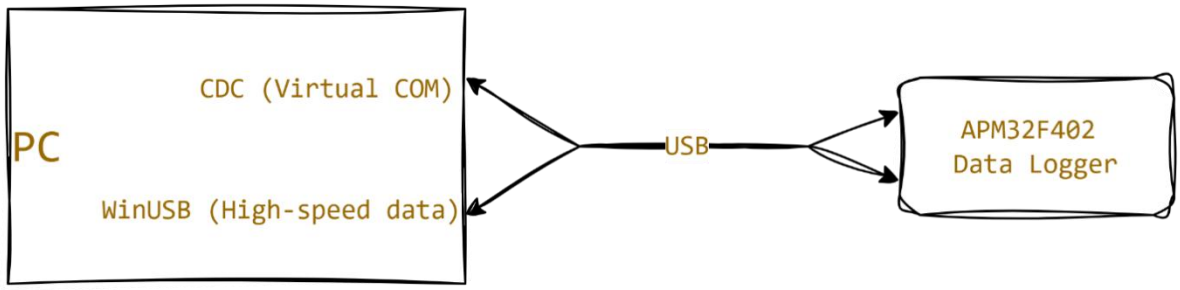
传统的 UART 串口虽然能满足基本的日志输出需求，但其缺点也同样明显：

- **带宽不足：** 常见的 115200bps 波特率在数据量大时可能成为瓶颈。提高波特率则对线材质量要求更高，且易受干扰。
- **需要额外转接：** 如果开发板仅提供 TTL 电平接口，则必须配备额外的 USB 转 TTL 线缆。
- **接线繁琐：** 电源线、调试线和串口线交织在一起，容易造成混乱。

相比之下，USB CDC（Communications Device Class，通信设备类）虚拟串口具备以下优势：

- **更高的传输速率：** 理论速率远高于常规 UART。
- **接线简洁：** 一根 USB 线即可同时负责供电、数据传输和调试。
- **灵活性高：** CDC 可以与其他 USB 功能（如 WinUSB、MSC、HID 等）作为复合设备共存，互不干扰。这使得在与 PC 高速交换数据的同时，还能通过虚拟 COM 端口查看实时日志成为可能。

图 1 USB 复合设备连接示意图



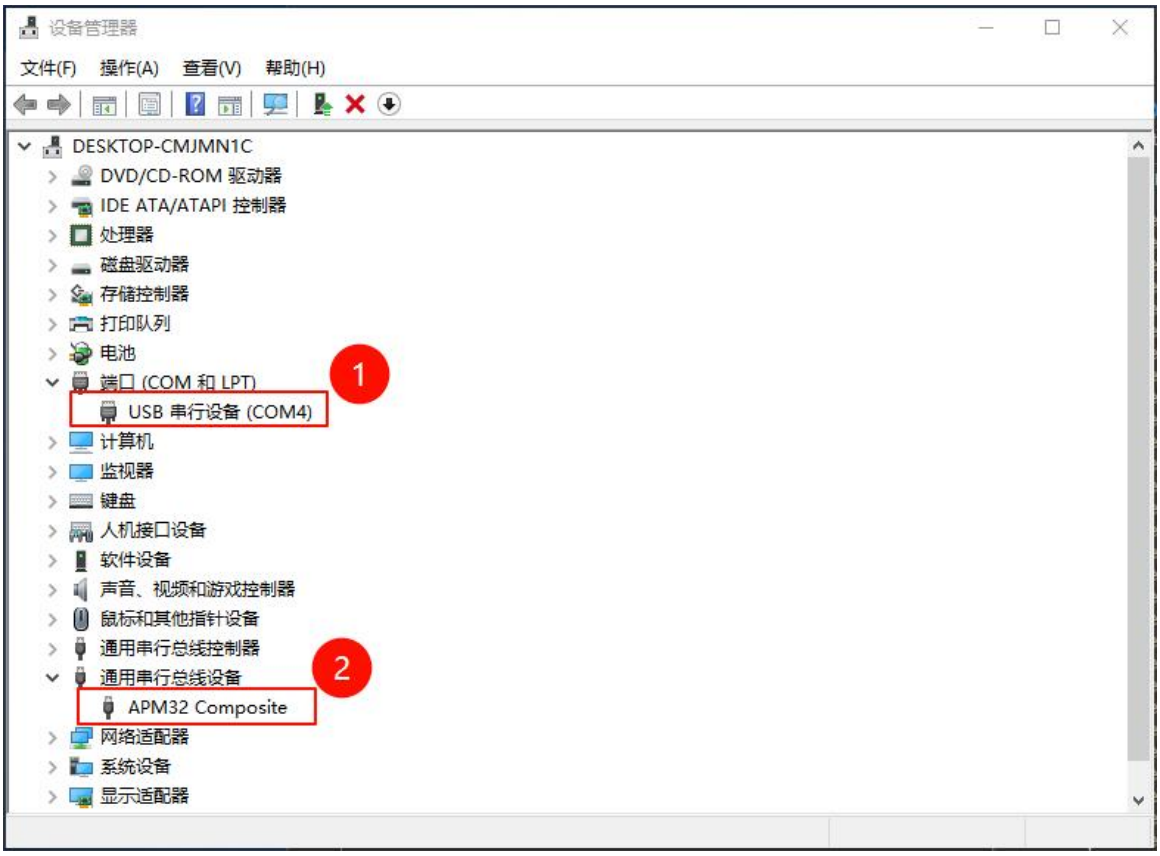
# 3 USB 复合设备简介

“USB 复合设备”是指在同一个物理 USB 接口上，通过配置描述符来开启多个逻辑功能接口。一个典型的组合如下：

- CDC 接口（虚拟串口）： 用于输出调试日志。
- WinUSB 接口（自定义协议）： 用于与上位机程序进行高速数据交互。
- 其他接口： 根据需求，只要资源允许，还可以加入 HID（人机接口设备）、MSC（大容量存储，U 盘模拟）、Audio（音频设备）等。

针对 APM32F402，极海官方提供了名为 OTGD_Composite_CDC_WINUSB 的例程。该例程已预先配置好复合设备的描述符和端点分配，开发者只需在其基础上进行少量修改，即可将 printf 函数的输出重定向至 CDC 虚拟串口，且不影响原有的 WinUSB 数据通道。

图 2 Windows 设备管理器中的复合设备识别



# 4 实现方法

## 4.1 基于官方例程

对于 APM32F402，极海提供了一个名为 OTGD_Composite_CDC_WINUSB 的例程。该例程已预先配置好复合设备的描述符和端点分配，用户只需在工程基础上进行少量修改，即可将 printf 函数的输出重定向至 CDC 虚拟串口，且不影响原有的 WinUSB 数据通道。

## 4.2 USB CDC 初始化

首先，请确保工程中已包含 USB 库的核心文件，如 usb_core、usb_init、usb_cdc 等驱动。在 main() 函数或自定义的系统初始化函数中，必须调用 USB_DeviceInit() 函数。此函数负责启动 USB 设备，并根据描述符向主机枚举 CDC 和 WinUSB 两个接口。

## 4.3 printf 重定向

在 MDK、IAR 等编译环境中，C 库函数 printf 的底层实现依赖于 fputc() 或 _write() 这类函数。只需重写这个底层函数，将其输出目标从默认的 UART 更改为 USB CDC 发送函数即可。
为了避免频繁调用 USB 发送函数（逐字符发送效率低下），可以引入一个静态缓冲区。将待发送的字符先存入缓冲区，当缓冲区满或遇到换行符 \n 时，再将整个缓冲区的数据通过 USB 一次性发送出去。

## 4.4 核心代码示例

以下是一个带有缓冲机制的 fputc() 函数重写示例。请注意，具体的函数名、宏定义等需与您的工程环境保持一致。

```
#define CDC_TX_BUF_SIZE  (128)

/*!
 * @brief       Redirect C Library function printf to serial port.
 *              After Redirection, you can use printf function.
 *
 * @param       ch:  The characters that need to be send.
 *
 * @param       *f:  pointer to a FILE that can recording all information
 *              needed to control a stream
 *
 * @retval      The characters that need to be send.
 *
 * @note
```

```
 */
int fputc(int ch, FILE* f)
{
    /* 原始的 UART 发送代码可以注释或移除 */
    // USART_TxData(DEBUG_USART, (uint8_t)ch);
    // while (USART_ReadStatusFlag(DEBUG_USART, USART_FLAG_TXBE) == RESET);

    /* 定义静态发送缓冲区和计数器 */
    static uint8_t s_cdcTxBuf[CDC_TX_BUF_SIZE];
    static uint16_t s_cdcTxCount = 0;

    /* 可选：根据需要将 '\n' 转换为 Windows 习惯的 "\r\n" */
    if (ch == '\n')
    {
        // 在添加 '\r' 前，检查缓冲区是否会溢出
        if (s_cdcTxCount >= CDC_TX_BUF_SIZE)
        {
            USBD_FS_CDC_ItfSend(s_cdcTxBuf, s_cdcTxCount);
            s_cdcTxCount = 0;
        }
        // s_cdcTxBuf[s_cdcTxCount++] = '\r'; // 如果需要，取消此行注释
    }

    /* 将当前字符存入缓冲区 */
    s_cdcTxBuf[s_cdcTxCount++] = (uint8_t)ch;

    /* 当缓冲区满或遇到换行符时，立即发送数据 */
    if (s_cdcTxCount >= CDC_TX_BUF_SIZE || ch == '\n')
    {
        USBD_FS_CDC_ItfSend(s_cdcTxBuf, s_cdcTxCount);
        s_cdcTxCount = 0; // 发送后清空计数器
    }

    return (ch);
}
```

注意：

（1） s_cdcTxBuf[]: 用于累积待发送数据的静态缓冲区，避免频繁调用 USBD_FS_CDC_ItfSend()。

（2） s_cdcTxCount: 记录当前缓冲区中的字节数。

（3） 当缓冲区满或遇到换行符 \n 时，触发 USBD_FS_CDC_ItfSend() 函数，将缓冲区内的数据一次性发送给主机。

## 4.5 编译与测试

(1) 将上述 fputc 函数添加到您的工程中。

(2) 编译代码并将其下载到 APM32F402 开发板。

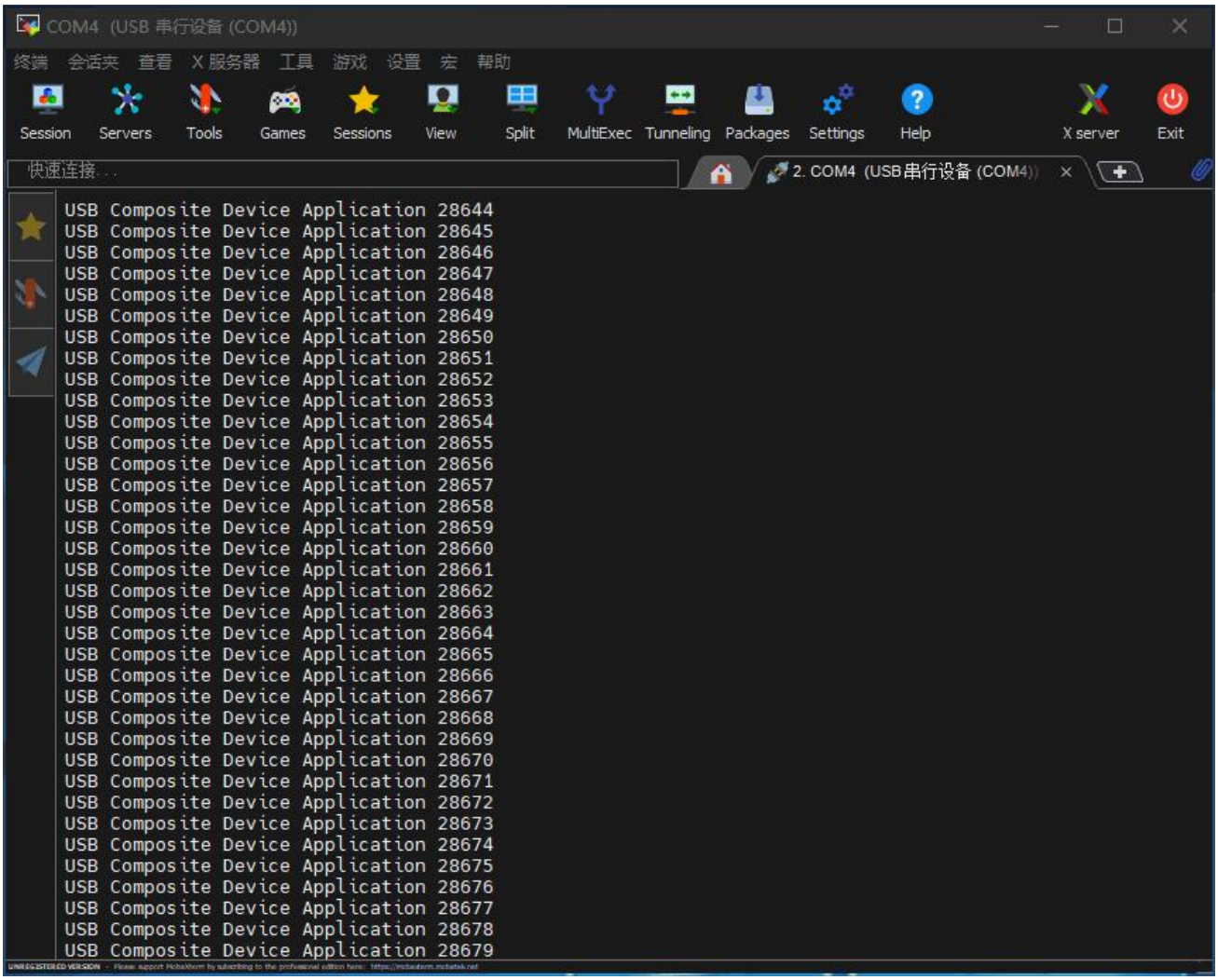
(3) 使用 USB 线连接开发板与电脑。

(4) 打开电脑的“设备管理器”，检查是否同时出现一个新的“虚拟 COM 端口”和一个“WinUSB”设备。

(5) 如果设备枚举成功，使用任意串口调试助手（如 PuTTY、Tera Term 等）打开该虚拟 COM 端口。波特率设置通常不影响通信，因为 USB CDC 的速率由 USB 总线决定，而非传统的波特率配置。

(6) 此时，MCU 中所有通过 printf 输出的信息都将实时显示在串口调试助手的窗口中。

图 3 通过串口助手查看 CDC 日志输出



# 5 总结

通过在极海官方 OTGD_Composite_CDC_WINUSB 例程的基础上进行少量修改，可以成功地将 printf 函数重定向到 USB CDC 虚拟串口，从而获得以下优势：

- 简化连接：仅用一根 USB 线即可完成开发、供电和调试所有工作。
- 节省资源：释放了硬件 UART 资源，可用于连接其他需要硬件流控或特定波特率的串口外设。
- 双通道并行：WinUSB 通道可以继续高效传输大数据，而 CDC 通道则专用于日志输出，两者互不干扰，提高了开发和调试效率。

这种 USB CDC + WinUSB 的融合方案，不仅降低了线缆成本，还统一了数据和日志的管理接口，特别适用于工业、医疗或消费电子等领域的复杂设备。在此基础上，还可以进一步扩展，例如引入 DMA 传输或在 RTOS 环境下构建多线程发送队列，以实现更高级的功能。

# 6 版本历史

表格 1 文件版本历史

| 日期 | 版本 | 变更历史 |
| --- | --- | --- |
| 2025.08 | 1.0 | 新建 |

# 声明

本手册由珠海极海半导体有限公司（以下简称“极海”）制订并发布，所列内容均受商标、著作权、软件著作权相关法律法规保护，极海保留随时更正、修改本手册的权利。使用极海产品前请仔细阅读本手册，一旦使用产品则表明您（以下称“用户”）已知悉并接受本手册的所有内容。用户必须按照相关法律法规和本手册的要求使用极海产品。

1、权利所有

本手册仅应当被用于与极海所提供的对应型号的芯片产品、软件产品搭配使用，未经极海许可，任何单位或个人均不得以任何理由或方式对本手册的全部或部分内容进行复制、抄录、修改、编辑或传播。

本手册中所列带有“®”或“™”的“极海”或“Geehy”字样或图形均为极海的商标，其他在极海产品上显示的产品或服务名称均为其各自所有者的财产。

2、无知识产权许可

极海拥有本手册所涉及的全部权利、所有权及知识产权。

极海不应因销售、分发极海产品及本手册而被视为将任何知识产权的许可或权利明示或默示地授予用户。

如果本手册中涉及任何第三方的产品、服务或知识产权，不应被视为极海授权用户使用前述第三方产品、服务或知识产权，也不应被视为极海对第三方产品、服务或知识产权提供任何形式的保证，包括但不限于任何第三方知识产权的非侵权保证，除非极海在销售订单或销售合同中另有约定。

3、版本更新

用户在下单购买极海产品时可获取相应产品的最新版的手册。

如果本手册中所述的内容与极海产品不一致的，应以极海销售订单或销售合同中的约定为准。

4、信息可靠性

本手册相关数据经极海实验室或合作的第三方测试机构批量测试获得，但本手册相关数据难免会出现校正笔误或因测试环境差异所导致的误差，因此用户应当理解，极海对本手册中可能出现的该等错误无需承担任何责任。本手册相关数据仅用于指导用户作为性能参数参照，不构成极海对任何产品性能方面的保证。

用户应根据自身需求选择合适的极海产品，并对极海产品的应用适用性进行有效验证和测试，以确认极海产品满足用户自身的需求、相应标准、安全或其它可靠性要求；若因用户未充分对极海产品进行有效验证和测试而致使用户损失的，极海不承担任何责任。

5、合规要求

用户在使用本手册及所搭配的极海产品时，应遵守当地所适用的所有法律法规。用户应了解产品可能受到产品供应商、极海、极海经销商及用户所在地等各国有关出口、再出口或其它法律的限制，用户（代表其本身、子公司及关联企业）应同意并保证遵守所有关于取得极海产品及/或技术与直接产品的出口和再出口适用法律与法规。

6、免责声明

本手册由极海“按原样”（as is）提供，在适用法律所允许的范围内，极海不提供任何形式的明示或暗示担保，包括但不限于对产品适销性和特定用途适用性的担保。

极海产品并非设计、授权或担保适合用于军事、生命保障系统、污染控制或有害物质管理系统中的关键部件，亦非设计、授权或担保适合用于在产品失效或故障时可导致人员受伤、死亡、财产或环境损害的应用。

如果产品未标明“汽车级”，则表示不适用于汽车应用。如果用户对产品的应用超出极海提供的规格、应用领域、规范，极海不承担任何责任。

用户应该确保对产品的应用符合相应标准以及功能安全、信息安全、环境标准等要求。用户对极海产品的选择和使用负全部的责任。对于用户后续在针对极海产品进行设计、使用的过程中所引起的任何纠纷，极海概不承担责任。

7、责任限制

在任何情况下，除非适用法律要求或书面同意，否则极海和/或以“按原样”形式提供本手册及产品的任何第三方均不承担损害赔偿责任，包括任何一般、特殊因使用或无法使用本手册及产品而产生的直接、间接或附带损害（包括但不限于数据丢失或数据不准确，或用户或第三方遭受的损失），这涵盖了可能导致的人身安全、财产或环境损害等情况，对于这些损害极海概不承担责任。

8、适用范围

本手册的信息用以取代本手册所有早期版本所提供的信息。