Application Note

Document No.: AN1163

APM32F402_F403_USB CDC Log Output

Version: V1.0

1 Introduction

During the development of MCU projects, developers often use UART serial ports to print debugging information. However, when the project becomes complex, the UART solution will face problems such as numerous cables, high resource occupation, and limited bandwidth. Therefore, developers will seek a better solution. For example, by taking advantage of the characteristics of USB, data transmission and log printing can be unified, which can be completed with just one cable.

This application note mainly introduces how to use the USB OTG function of the APM32F402/F403 chip and configure it as a "Composite Device" to support both the WinUSB (for high-speed data transmission) and CDC (virtual serial port for log output) interfaces simultaneously. In this way, with just one USB data cable, power supply, data exchange, and debugging log output can be realized simultaneously, providing a convenient development experience of "one cable for three purposes".

For example, a Data Logger needs to collect sensor data (pressure, flow, vibration, etc.) in real time or periodically. On-site maintenance personnel can establish a WinUSB channel with the device through a laptop to download a large amount of historical data at high speeds. At the same time, debugging engineers can open the CDC virtual serial port in the terminal software to view the MCU's operating status or alarm information in real time. The whole process only requires one USB cable, which simplifies the physical connection without affecting high-bandwidth data transmission.

This application note applies to the Geehy APM32F402/APM32F403. The APM32F402 is used as an example throughout this document.

Contents

2 Reasons for Choosing USB CDC

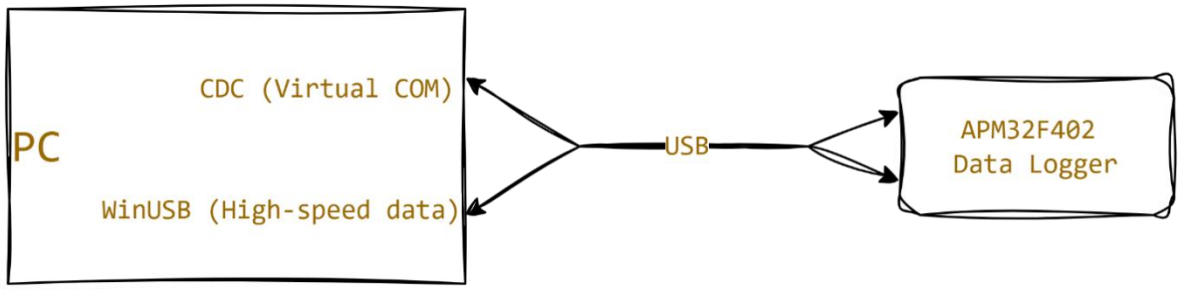
Although the traditional UART serial port can meet the basic log output requirements, its drawbacks are also obvious:

- **Insufficient bandwidth:** The common baud rate of 115200 bps may become a bottleneck when the data volume is large. Increasing the baud rate requires higher-quality wires and is more susceptible to interference.
- **Additional conversion is needed:** If the development board only provides a TTL-level interface, an additional USB-to-TTL cable must be provided.
- **Complicated wiring:** The power cable, debugging cable, and serial cable are intertwined, which can easily cause confusion.

In contrast, the USB CDC (Communications Device Class) virtual serial port has the following advantages:

- **Higher transmission rate:** The theoretical rate is much higher than that of the conventional UART.
- **Simple wiring:** A single USB cable can be used for power supply, data transmission, and debugging simultaneously.
- **High flexibility:** CDC can coexist with other USB functions (WinUSB, MSC, HID, etc.) as a composite device, which do not interfere with each other. This makes it possible to exchange data with the PC at high speeds and view real-time logs through the virtual COM port at the same time.

Figure 1 Schematic Diagram of USB Composite Device Connection

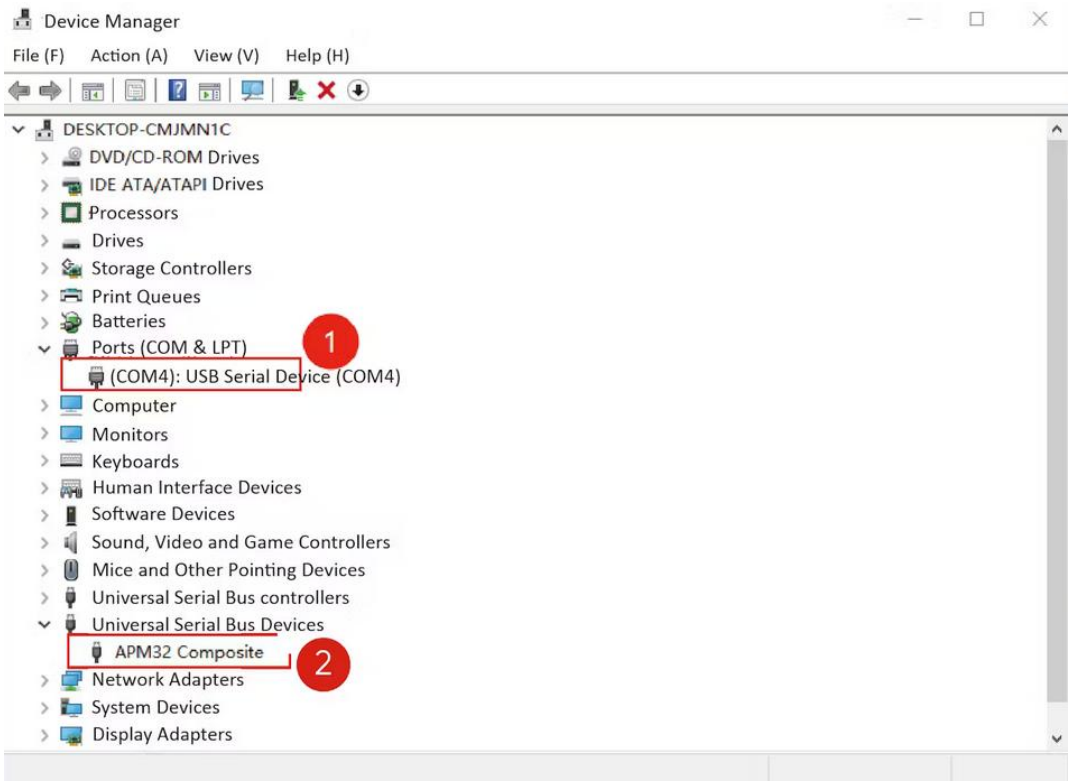


3 Introduction to USB Composite Device

USB composite device: Used to enable multiple logical function interfaces through configuration descriptors on the same physical USB interface. A typical combination is as follows:
CDC interface (virtual serial port): Used to output debugging logs.

- WinUSB interface (custom protocol): Used for high-speed data interaction with the host computer program.
- Other interfaces: According to requirements, as long as resources permit, HID (human-machine interface device), MSC (mass storage, U-disk emulation), Audio (audio device), etc. can also be added.
  For the APM32F402, Geehy official provides an example named OTGD_Composite_CDC_WINUSB. This example has pre-configured the descriptors and endpoint allocations of the composite device. Developers only need to make a few modifications on this basis to redirect the output of the printf function to the CDC virtual serial port without affecting the original WinUSB data channel.

Figure 2 Recognition of Composite Device in Windows Device Manager



4 Implementation Method

4.1 Based on the official example

For APM32F402, Geehy provides an example named OTGD_Composite_CDC_WINUSB. This example has pre-configured the descriptors and endpoint allocations of the composite device. Users only need to make a few modifications on the basis of the project to redirect the output of the printf function to the CDC virtual serial port without affecting the original WinUSB data channel.

4.2 USB CDC initialization

First, please ensure that the project includes the core files of the USB library, such as drivers like usb_core, usb_init, and usb_cdc. In the main() function or a custom system initialization function, the USB_DeviceInit() function must be called. This function is responsible for starting the USB device and enumerating the CDC and WinUSB interfaces to the host according to the descriptors.

4.3 printf redirection

In compilation environments such as MDK and IAR, the underlying implementation of the C library function printf depends on functions like fputc() or _write(). Just rewrite this underlying function and change its output target from the default UART to the USB CDC sending function. In order to avoid frequent calls to the USB sending function (sending characters one by one has low efficiency), a static buffer can be introduced. First store the characters to be sent in the buffer. When the buffer is full or a newline character \n is encountered, send all the data in the buffer through USB at once.

4.4 Core code example

The following is an example of rewriting the fputc() function with a buffering mechanism. Please note that the specific function names, macro definitions, etc. need to be consistent with your project environment.

```
#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)
{
    /* The original UART sending code can be commented out or removed */
    // USART_TxData(DEBUG_USART, (uint8_t)ch);
    // while (USART_ReadStatusFlag(DEBUG_USART, USART_FLAG_TXBE) == RESET);

    /* Define a static sending buffer and a counter */
    static uint8_t s_cdcTxBuf[CDC_TX_BUF_SIZE];
    static uint16_t s_cdcTxCount = 0;

    /* Optional: Convert '\n' to "\r\n" as per Windows convention if needed */
    if (ch == '\n')
    {
        // Check if the buffer will overflow before adding '\r'
        if (s_cdcTxCount >= CDC_TX_BUF_SIZE)
        {
            USBD_FS_CDC_ItfSend(s_cdcTxBuf, s_cdcTxCount);
            s_cdcTxCount = 0;
        }
        // s_cdcTxBuf[s_cdcTxCount++] = '\r'; // Uncomment this line if needed
    }

    /* Store the current character in the buffer */
    s_cdcTxBuf[s_cdcTxCount++] = (uint8_t)ch;

    /* Send data immediately when the buffer is full or a newline character is
encountered. */
    if (s_cdcTxCount >= CDC_TX_BUF_SIZE || ch == '\n')
    {
        USBD_FS_CDC_ItfSend(s_cdcTxBuf, s_cdcTxCount);
        s_cdcTxCount = 0; // Clear the counter after sending
    }

    return (ch);

}
```

Note:

(1) s_cdcTxBuf[]: A static buffer used to accumulate data to be sent, avoiding frequent calls to

USBD_FS_CDC_ItfSend().

(2) s_cdcTxCount: Records the number of bytes currently in the buffer.

(3) When the buffer is full or a newline character \n is encountered, trigger the USBD_FS_CDC_ItfSend() function to send the data in the buffer to the host at once.

4.5 Compilation and testing

(1) Add the above fputc function to your project.

(2) Compile the code and download it to the APM32F402 development board.

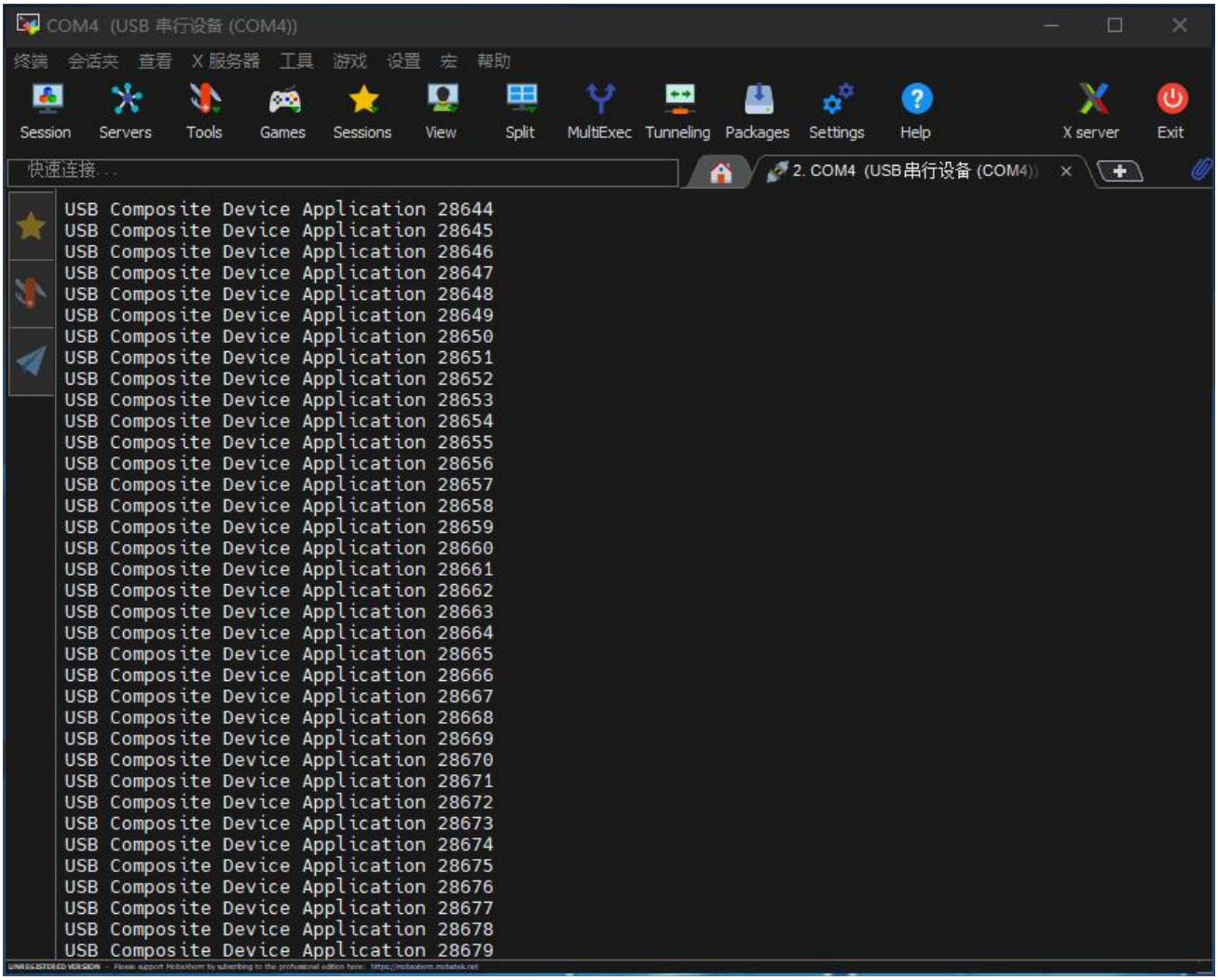
(3) Connect the development board to the computer using a USB cable.

(4) Open the "Device Manager" on the computer and check if a new "Virtual COM Port" and a "WinUSB" device appear simultaneously.

(5) If the device enumeration is successful, use any serial port debugging assistant (PuTTY, Tera Term, etc.) to open the virtual COM port. The baud rate setting usually does not affect communication because the speed of USB CDC is determined by the USB bus rather than the traditional baud rate configuration.

(6) At this time, all information output by the printf function in the MCU will be displayed in real time in the window of the serial port debugging assistant.

Figure 3 Viewing CDC Log Output through Serial Port Assistant



5 Summary

By making a few modifications based on the official Geehy OTGD_Composite_CDC_WINUSB example, the printf function can be successfully redirected to the USB CDC virtual serial port, thus obtaining the following advantages:
Simplified connection: All tasks of development, power supply, and debugging can be completed with just one USB cable.

- Resource saving: The hardware UART resources are released and can be used to connect other serial peripherals that require hardware flow control or specific baud rates.
- Dual-channel parallel: The WinUSB channel can continue to efficiently transmit large amounts of data, while the CDC channel is used only for log output. The two do not interfere with each other, so that the efficiency of development and debugging can be improved.

This integrated solution of USB CDC + WinUSB not only reduces the cable cost but also unifies the management interfaces for data and logs. It is particularly suitable for complex devices in fields such as industry, medical, or consumer electronics. On this basis, further expansion is possible. For example, DMA transmission is introduced or a multi-thread sending queue is built in an RTOS environment to serve more advanced functions.

6 Revision History

Table 1 Revision History

Date	Version	Revision History
August, 2025	1.0	New

Statement

This manual is formulated and published by Zhuhai Geehy Semiconductor Co., Ltd. (hereinafter referred to as "Geehy"). The contents in this manual are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to correct and modify this manual at any time. Please read this manual carefully before using the product. Once you use the product, it means that you (hereinafter referred to as the "users") have known and accepted all the contents of this manual. Users shall use the product in accordance with relevant laws and regulations and the requirements of this manual.

1. Ownership of rights

This manual can only be used in combination with chip products and software products of corresponding models provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this manual for any reason or in any form.

The "Geehy" or "Geehy" words or graphics with "®" or "TM" in this manual are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No intellectual property license

Geehy owns all rights, ownership and intellectual property rights involved in this manual.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale and distribution of Geehy products and this manual.

If any third party’s products, services or intellectual property are involved in this manual, Geehy shall not be deemed to authorize users to use the aforesaid third party’s products, services or intellectual property, nor shall it be deemed to provide any form of guarantee for third-party products, services, or intellectual property, including but not limited to any

non-infringement guarantee for third-party intellectual property, unless otherwise agreed in sales order or sales contract of Geehy.

3. Version update

Users can obtain the latest manual of the corresponding products when ordering Geehy products.

If the contents in this manual are inconsistent with Geehy products, the agreement in Geehy sales order or sales contract shall prevail.

4. Information reliability

The relevant data in this manual are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment are unavoidable. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this manual. The relevant data in this manual are only used to guide users as performance parameter reference and do not constitute Geehy's guarantee for any product performance.

Users shall select appropriate Geehy products according to their own needs, and effectively verify and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If losses are caused to users due to the user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Compliance requirements

Users shall abide by all applicable local laws and regulations when using this manual and the matching Geehy products. Users shall understand that the products may be restricted by the export, re-export or other laws of the countries of the product suppliers, Geehy, Geehy distributors and users. Users (on behalf of itself, subsidiaries and affiliated enterprises) shall agree and undertake to abide by all applicable laws and regulations on the export and re-export

of Geehy products and/or technologies and direct products.

6. Disclaimer

This manual is provided by Geehy on an "as is" basis. To the extent permitted by applicable laws, Geehy does not provide any form of express or implied warranty, including without limitation the warranty of product merchantability and applicability of specific purposes.

Geehy products are not designed, authorized, or guaranteed to be suitable for use as critical components in military, life support, pollution control, or hazardous substance management systems, nor are they designed, authorized, or guaranteed to be suitable for applications that may cause injury, death, property, or environmental damage in case of product failure or malfunction.

If the product is not labeled as "Automotive grade", it means it is not suitable for automotive applications. If the user's application of the product is beyond the specifications, application fields, and standards provided by Geehy, Geehy will assume no responsibility.

Users shall ensure that their application of the product complies with relevant standards, and the requirements of functional safety, information security, and environmental standards. Users are fully responsible for their selection and use of Geehy products. Geehy will bear no responsibility for any disputes arising from the subsequent design and use of Geehy products by users.

7. Limitation of liability

In any case, unless required by applicable laws or agreed in writing, Geehy and/or any third party providing this manual and the products on an "as is" basis shall not be liable for damages, including any general or special direct, indirect or collateral damages arising from the use or no use of this manual and the products (including without limitation data loss or inaccuracy, or losses suffered by users or third parties), which cover damage to personal safety, property, or environment, for which Geehy will not be responsible.

8. Scope of application

The information in this manual replaces the information provided in all previous versions of the manual.