

Virtual COM Port Template Application Guide

Revision: V1.00 Date: March 29, 2024

www.holtek.com



Table of Contents

1. VCP (Virtual COM Port) Template Overview	3
2. VCP Template Architecture	3
3. VCP Template File Description	4
4. MCU Program Description	5
4.1 Program Main Flow	5
4.2 Data Transmission	6
4.3 define.h	9
4.4 mcu.h	9
5. Software Instructions	.10
5.1 Driver	10
5.2 COM Port Software	13



1. VCP (Virtual COM Port) Template Overview

The Holtek USB Code Library Generator provides a Virtual COM Port, the VCP template. This application type can support products that originally use a UART transmission interface. Modern day computers have gradually ceased supporting the UART interface. When the transmission interface is converted, the corresponding software must also be modified. If this device type is used for product development, the device will become a Virtual COM Port when it is connected to the computer therefore the software does not have to be modified.

2. VCP Template Architecture

In addition to MCU programs, the VCP template also provides a driver setup information file (INF). This allows users to manually setup a driver on the Windows operating system that cannot automatically setup the driver.

When the project has been generated, the device files will be located in the Device directory within the project directory and the computer files will be located in the Host directory.

This template demonstrates a simple data loopback function. When a computer COM port software actively transmits data with variable lengths, the USB device will return the received data to the computer. This means that the program only focuses on the development of the USB virtual serial port, and does not involve the UART timing. Users must complete this by themselves.





3. VCP Template File Description

[Project Dir]		
[Device]		
[Source]		
[application]		
main.c	Main program	
define.h	User define variables	
[hw]		
mcu.h	MCU related settings	
[os]		
[usb]		
CLS	USB class command related functions	
STD	USB enumeration and standard command	
	related functions	
USB_DESC.h	USB enumerate data	
USB_INT	USB interrupt	
USB_LIB	USB FIFO access related functions	
USB_FIFO.h	USB related definition	
xxxx.pjtx	HT-IDE3000 project files	
[Host]		
ht_cdc_usb.inf	Driver setup information file	
Fig 2. Directory Structure		

4. MCU Program Description



Fig 3. Program Main Flow

4.1 Program Main Flow

When the MCU has been powered-on, MCU_Init will be called to configure the system frequency and the output/input ports and USBInitial will be called to execute the USB related settings.

The program will then continuously determine whether any data needs to be accessed via the USB endpoints and then execute the following steps:

- AccessFIFO0 USB enumeration will be completed using endpoint 0. This document does not describe the USB specifications. Refer to the usb.org website for the relevant files.
- AccessFIFO1 When the RI/DSR/CD signal changes, its status will be returned to the host via endpoint 1.
- AccessFIFO2 Data will be received from the host via endpoint 2.
- AccessFIFO3 Data will be uploaded to the host via endpoint 3.



4.2 Data Transmission



Fig 4. Endpoint 2/3 Access Flow

4.2.1 USB device passively waits for a host access event

When an endpoint 2 access event has been detected, the data will be read out and the data length will be recorded using ucDataLen. When an endpoint 3 access event has been detected, if ucDataLen is not 0, it indicates that the data has been received from the host. After the data has been returned to the host by calling WriteFIFO, ucDataLen should be cleared to 0.

This template only demonstrates the data loopback function. The user can modify AccessFIFO2 in the usb_int.c file to transmit data via the UART timing when receiving data from the host. Similarly, when the data has been received from the UART, it will be transmitted to the host via AccessFIFO3.

4.2.2 RI/DSR/CD Status

These three signals are modulator/demodulator related signals. The corresponding status will be returned to the host via AccessFIFO1 when the signal changes. These signals do not have to be processed when this device is used in transparent transmission bridge applications.

Refer to the PSTN subclass of the CDC class provided at the usb.org for the return formats.



4.2.3 VCP Control Command

If the user want to develop a simple transparent transmission device with a fixed transmission speed and number of data bits, the following three commands do not have to be modified.



Fig 5. Endpoint 0 Access Flow

(A) SetLineCoding

The host can set the baudrate, stop bits, parity check code and number of data bits using the SetLineCoding command.

If necessary, add the UART control settings to SetupLineCoding in the usb_int.c file. The LineCoding consists of seven bytes, each of which are described in the following table:

Bytes	Description
0~3	Baudrate
4	Stop bits
5	Parity check code
6	Number of data bits

ł

void SetupLineCoding()

```
//LineCoding[0:3] : baudrate
//LineCoding[4] : stop bits 0->1 stop
17
                             1->1.5 stops
17
                             2->2 stops
//LineCoding[5] : parity
                             0->none
17
                             1->odd
17
                             2->even
17
                             3-≻mark
17
                             4->space
//LineCoding[6] : databits 5,6,7,8
17
//add your code here
77
```

(B) GetLineCoding

}

ł

}

The host can obtain a UART setting value using the GetLineCoding command. The device only returns 7-byte LineCoding without modification.

(C) SetControlLineState

The host can control flows using the SetControlLineState in the cls.c file. If necessary, modify this function to set RTS/DTR signal according to the transmitted values.

Bits	Description	
D7 to D2	Reserved	
D1	DCE transmit function control	0 – RTS Off
		1 – RTS On
D0	Notification of DTE ready state	0 – DTR Off
		1 – DTR On

```
static void SetControlLineState()
```

```
//Indicates to DCE if DTE is present or not
if(FIF0_OUT[FIF0_wValL]&0x01)
;//_pin_DTR=0;
else ;//_pin_DTR=1;
//activate/deactivate Tx carrier
if(FIF0_OUT[FIF0_wValL]&0x02)
;//_pin_RTS=0;
else ;//_pin_RTS=1;
```



4.3 define.h

#define DESC_IDVENDOR	(0x04D9)
#define DESC_IDPRODUCT	(0x806C)
#define DESC_BCDDEVICE	(0x0100)

The define.h file provides the following parameters for users to modify as required.

a. USB vendor ID - DESC_IDVENDOR

The identifier 0x04D9 is the Holtek dedicated USB ID. The user can use this or change it to their own applied for vendor ID.

b. USB product ID - DESC_IDPRODUCT

Different products can be configured with different product IDs to identify the pairing device using software.

c. Program version - DESC_BCDDEVICE

4.4 mcu.h

This template selects an internal oscillator as the system frequency by default. The external oscillators used by different ICs have different settings. The user can modify the parameters in the mcu.h file to change the settings.

```
#define _HXT_ 0
#define _HXTEN_ 0
#define _PLL_ 0
#define _LXT_ 0
#define _LXT_ 0
#define _LXTEN_ 0
```

If _HXT_ has a value of 0, this indicates that this device does not support the programmable selection of an external high speed oscillator as the high frequency clock source or that it does not support an external oscillator.

Similarly, if <u>LXT</u> has a value of 0, this indicates that this device does not support the programmable selection of an external low speed oscillator as the low frequency clock source or that it does not support an external oscillator.

Refer to the device datasheet for detailed usage.

#define _HX1	r_	1	
#define	_нхлем_		0
#define	_hxp [^] 0		_pds00
#define	_hxp1		_pds01
#define	_hxp2		_pds02
#define	_hxp3		_pds03
#define	_PLL_		0
<pre>#define _LX1</pre>	ſ_	1	
#define	_LXTEN_		0
#define	_1xp0 _		_pes06
#define	_1xp1		_pes07
#define	_1xp2		_pes10
#define	_1xp3		_pes11



When <u>HXT</u> has a value of 1, if the user wants to select an external oscillator as the high frequency clock source, then set <u>HXTEN</u> to 1.

Similarly, if _LXTEN_ has been set high, the low frequency will be sourced from an external oscillator.

Other defined values in the mcu.h file are related to the device's own characteristics and should not be arbitrarily changed.

5. Software Instructions

5.1 Driver

On Windows 8 and higher operating systems, the virtual COM port drivers are usually installed automatically. When the driver setup has been completed, the COM port number will be displayed in the Device Manager as shown in the following figure.





If the operating system does not support automatic COM port driver setup, this is displayed as shown in the following figure:



Now execute the following steps to install manually.

1. Right-click on "Holtek VCP Demo" and choose [Update Driver Software].



2. Select [Browse my computer for driver software] in the window that appears.



3. Select the Host directory in the project directory.

G 🗓 Update Driver Software - Holtek VCP Demo	
Browse for driver software on your computer	
Search for driver software in this location:	
C:\USBSW\myVCP\Host Browse	
Let me pick from a list of device drivers on my computer This list will show installed driver software compatible with the device, and all driver software in the same category as the device.	
Next Car	ncel



4. Finish installing.

	—
😡 📱 Update Driver Software - Holtek USB VCP Demo (COM4)	
Windows has successfully updated your driver software	
Windows has finished installing the driver software for this device:	
Holtek USB VCP Demo	
	Close

5.2 COM Port Software

The general COM port communication software can be used to test VCP devices. Take PuTTY as an example.

5.2.1 Enable COM Port

- Enter the number as shown in Device Manager in "Serial line"
- Enter any value in "Speed", because this template is only used for a data loopback test and does not actually implement a UART timing.
- Click [Open].





5.2.2 Data Loopback Test

Enter any string in the window using the keyboard. If it displays what was entered, this indicates that the data has been correctly returned. If the data is not returned to the host, the entered string will not be displayed in the window.





Copyright[®] 2024 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

The information provided in this document has been produced with reasonable care and attention before publication, however, HOLTEK does not guarantee that the information is completely accurate. The information contained in this publication is provided for reference only and may be superseded by updates. HOLTEK disclaims any expressed, implied or statutory warranties, including but not limited to suitability for commercialization, satisfactory quality, specifications, characteristics, functions, fitness for a particular purpose, and non-infringement of any thirdparty's rights. HOLTEK disclaims all liability arising from the information and its application. In addition, HOLTEK does not recommend the use of HOLTEK's products where there is a risk of personal hazard due to malfunction or other reasons. HOLTEK hereby declares that it does not authorise the use of these products in life-saving, life-sustaining or safety critical components. Any use of HOLTEK's products in life-saving/sustaining or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold HOLTEK harmless from any damages, claims, suits, or expenses resulting from such use. The information provided in this document, including but not limited to the content, data, examples, materials, graphs, and trademarks, is the intellectual property of HOLTEK (and its licensors, where applicable) and is protected by copyright law and other intellectual property laws. No license, express or implied, to any intellectual property right, is granted by HOLTEK herein. HOLTEK reserves the right to revise the information described in the document at any time without prior notice. For the latest information, please contact us.