HID Template Application Guide

Revision: V1.00 Date: March 29, 2024

www.holtek.com

Table of Contents

1. HID Template Overview

The Holtek USB Code Library Generator provides users with a simple and fast platform for USB solutions. This application guide will discuss the HID template, which can help users avoid the complex USB underlying communication protocols and which does not require any driver installation. This enables data communication between a USB device and a personal computer to be implemented solely using computer interface functions and program examples provided by this template.

2. HID Template Architecture

In addition to MCU programs, the HID template also provides software communication programs, a Windows Dynamic Link Library - HIDAPI.DLL and Link Library Usage Examples.

After 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 software program actively transmits 16 bytes of data, the USB device will return the received data to the computer. It demonstrates a simple application of data transmission between a USB host and a USB device.

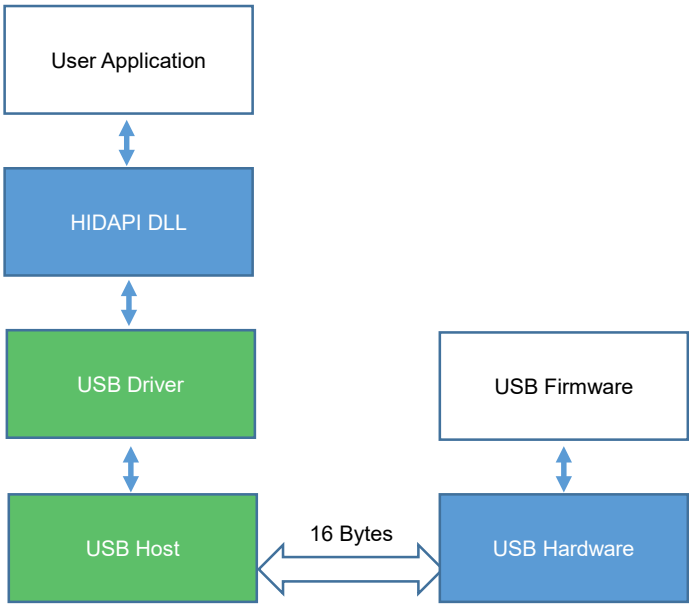

Fig 1. USB Stack

3. HID 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]**
  - -- **[HIDAPI]** HIDAPI.dll and header files
  - -- **[HID_INOUT]**
  - -- HID_INOUT.sln Project files for data loopback function
  - -- **[HID_INOUT]** Window source codes for data loopback function
  - -- **[ConsoleApplicationCS]** Console program developed in C#
  - -- **[ConsoleApplicationVB]** Console program developed in VB

Fig 2. Directory Structure

4. MCU Program Description

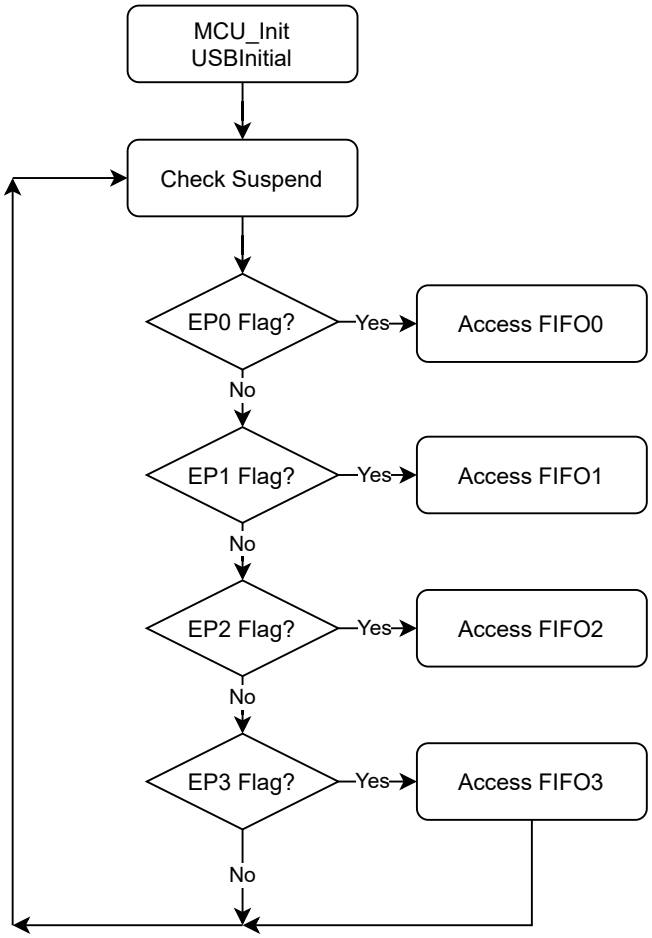

Fig 3. Program Main Flowchart

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 Data will be uploaded to the host via endpoint 1.

AccessFIFO2 Data will be received from the host via endpoint 2.

4.2 Data Transmission

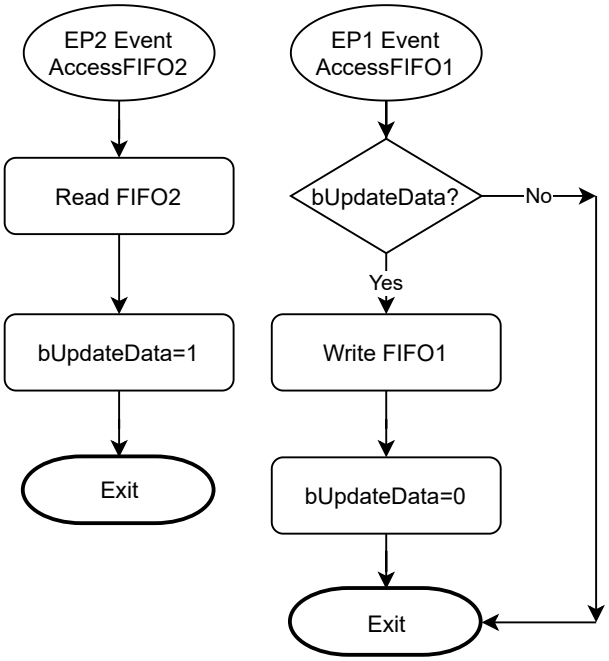

Fig 4. Endpoint Access Flowchart

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 bUpdateData flag will be set high. When an endpoint 1 access event has been detected, if the bUpdateData flag is 1, it indicates that the data has been received from the host. The bUpdateData flag should be cleared to 0 after sending the data back.

Users can modify AccessFIFO1 and AccessFIFO2 in the usb_int.c file and add product communication command protocols according to their functional requirements.

4.2.2 SetFeature/GetFeature

The USB HID also provides a path to transmit data via endpoint 0, which provides additional user flexibility during protocol definitions. For example:

The USB device can receive commands from the host using SetFeature. AccessFIFO2 is used to receive the data. Instead, GetFeatureReport is used to transmit the status. AccessFIFO1 is used to transmit the data.

Note: The directionality of the USB commands is described from the perspective of the host. The transmit direction of Get command is from device to host. The transmit direction of Set command is from host to device.

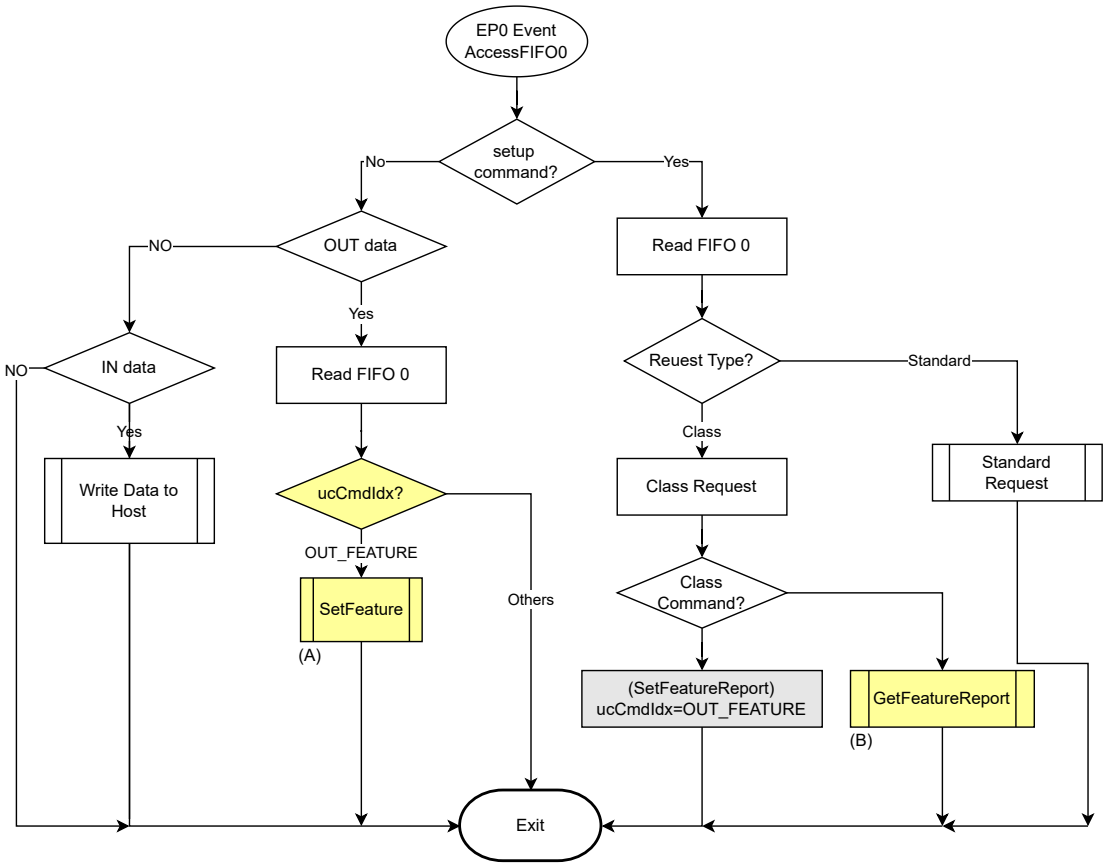

Fig 5. Endpoint 0 Access Flowchart

(A) SetFeature (usb_int.c)

When the program has called this function, the data from the host has already been read out and stored in the FIFO_OUT buffer.

```
void SetFeature()
{

  //**modify code here
  //**data was ready in FIFO_OUT buffer


  ucCmdIdx=0;
  WriteFIFO(0,0);                    //handshake
}
```

The user only has to read the data out from FIFO_OUT buffer to use.

In this template the Feature length is 2. To change the Feature length, modify its definition in the define.h file. Refer to the next section for details.

(B) GetFeature (cls.c)

When the program has called a GetReport function in the cls.c file, it means that the host has issued a GetFeatureReport command. The data, which will be returned to the host, just needs to be placed in the FIFO_OUT buffer by the user. The length remains at 2.

```
void GetReport()
{
  if((FIFO_OUT[FIFO_Type] & 0x01))
  {
    if(FIFO_OUT[FIFO_wValH]==1)
    {
    {
      //GetFeatureReport();
      if(FIFO_OUT[FIFO_wLenH]!=0 ||
         FIFO_OUT[FIFO_wIdxL]!=0 ||
         FIFO_OUT[FIFO_wLenL]!=FEATURE_SIZE)
        _stli |= 0x01;
      else
      {
        //**add code here**
        //**put data in FIFO_OUT buffer**
        WriteFIFO(0,FEATURE_SIZE);
      }
    }
  }else
  {
    _stli |= 0x01;
  }
}
```

4.3 define.h

```
#define FEATURE_SIZE        2
#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. Feature length - FEATURE_SIZE

The length can be set up to 8.

Now the data to be transmitted via endpoints 1/2 has a fixed length of 16 and cannot be modified by users.

The USB HID class does not support data transmission with variable lengths, it means that the data to be transmitted via endpoint 0 and endpoint 1/2, has a fixed length with the FEATURE_SIZE and the EP_LEN.

b. 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.

c. USB product ID - DESC_IDPRODUCT

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

d. 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 _LXTEN_       0
```

If _HXT_ has a value of 0, this indicates that this IC 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, a _LXT_ value of 0 also indicates that this IC 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 IC datasheet for detailed usage.

```
#define _HXT_          1
    #define _HXTEN_       0
    #define _hxp0         _pds00
    #define _hxp1         _pds01
    #define _hxp2         _pds02
    #define _hxp3         _pds03

    #define _PLL_         0

#define _LXT_          1
    #define _LXTEN_       0
    #define _lxp0         _pes06
    #define _lxp1         _pes07
    #define _lxp2         _pes10
    #define _lxp3         _pes11
```

When _HXT_ has a value of 1, if the user wants to select an external oscillator as the high frequency clock source, then set _HXTEN_ 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 IC's own characteristics and should not be arbitrarily changed.

5. Software Instructions

5.1 HID_INOUT

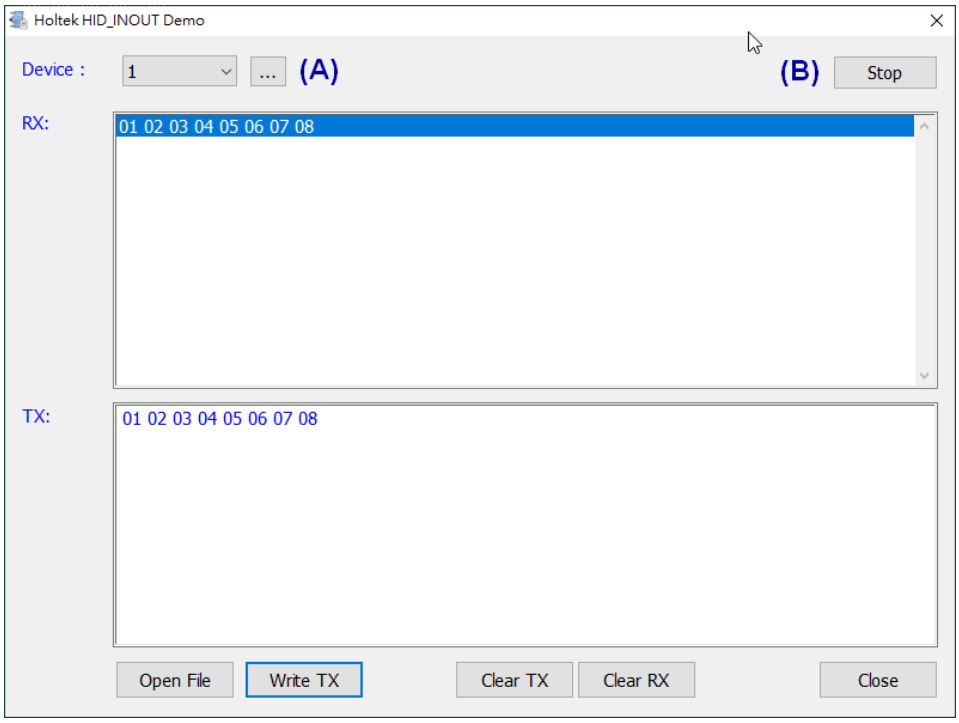

Fig 6. Holtek HID Demo Data Loopback Test

The USB Code Library Generator provides a simple window software for the HID template, which is developed in Visual Studio C++ and is used to demonstrate the data loopback via USB interface.

The steps are as follows:

(A) When the […] button is clicked, the OpenFirstHIDDevice/OpenNextHIDDevice will be called to locate all the USB HID devices with the specified VID/PID in the host.

(B) Similarly, the OpenFirstHIDDevice/OpenNextHIDDevice will be called to start the specified USB HID device. Then the background thread will be started to monitor the returned data at any time.

(TX) Enter hexadecimal numbers in this TX window at the bottom of the screen and separate the individual values with a space. Then click the [Write TX] button. At this time, the WriteFile will be called by software. The data will be transmitted to the USB HID device using AccesssFIFO2. Refer to the HIDAPI Dynamic Link Library Function Description for details.

(RX) When the USB HID device has returned data to the software using AccessFIFO1, the data which has been read back, will be shown in the RX window using the ReadFile. Refer to the HIDAPI Dynamic Link Library Function Description for details.

It should be noted that the USB HID class does not support data transmission with variable lengths. This means that the data of each transmission has a fixed length, the EP_LEN (endpoints 1/2).

Therefore, in the HID_INOUT software, the first byte is regarded as the data length to achieve transmission with variable lengths. The maximum length in a data transmission can be up to EP_LEN-1.

D0	D1	D2~D16
Report ID(0)	Len	Data

5.2 ConsoleApplicationCS

5.3 ConsoleApplicationVB

In addition to C++, the HIDAPI.dll file can also be called in other programming languages. The USB Code Library Generator provides console programs in C# and VB, in which the USB device can be opened, written to, read out and closed. This console program is mainly used to demonstrate the DLL function loading and calling.

6. HIDAPI Dynamic Link Library Function Description

6.1 Functional Description

Function	**HANDLE OpenFirstHIDDevice(DWORD wVID, DWORD wPID, DWORD wUsagePage, DWORD wUsage, BOOL bSync);** **HANDLE OpenNextHIDDevice(DWORD wVID, DWORD wPID, DWORD wUsagePage, DWORD wUsage, BOOL bSync);**	
Parameter	wVID	The vendor ID of the USB device to open.
	wPID	The product ID of the USB device to open.
	wUsagePage wUsage	The UsagePage/Usage described by the Report descriptor (refer to the HID specification). If it is not specified, enter a 0. When the same VID/PID has more than one USB interface, specify the UsagePage/Usage, otherwise it will return to the first interface handle.
	bSync	Synchronous transmission or asynchronous transmission. For synchronous transmission the function will return after I/O completion.
Return	The open device handle, if this fails, return Null.	

Description:

When there the same VID/PID USB devices exist, use an OpenFirstHIDDevice to obtain the first one. Then with OpenNextHIDDevice, continue to open until the return value is Null.

Then execute a ReadFile/WriteFile action. The ReadFile/WriteFile are the standard functions of the Windows API. Refer to MSDN for details.

```
BOOL WriteFile(HANDLE                   hFile,
               LPCVOID                  lpBuffer,
               DWORD                    nNumberOfBytesToWrite,
               LPDWORD                  lpNumberOfBytesWritten,
               LPOVERL                  APPED lpOverlapped);
BOOL ReadFile(HANDLE                    hFile,
              LPVOID                    lpBuffer,
              DWORD                     nNumberOfBytesToRead,
              LPDWORD                   lpNumberOfBytesRead,
              LPOVERLAPPED              lpOverlapped);
```

The length of data to be read (nNumberOfBytesToRead)/written (nNumberOfBytesToWrite) must be the length defined by the F/W plus one. The length is EP LEN+1 in this template.

The first byte of the incoming lpBuffer must be set to the Report ID. Without the Report ID then enter 0.

When the WriteFile is called by the host, the device will obtain the data from AccessFIFO2.

When ReadFile is called by the host, the host will obtain the data transmitted by the device from AccessFIFO1.

Function	void CloseHIDDevice(HANDLE hDevice)	
Parameter	hDevice	The handle of the device to close.
Return	Null	

Description:

Close the specified USB device.

Function	BOOL SetFeature(HANDLE hDevice, LPVOID pData,DWORD nLen)	
Parameter	hDevice	The handle of the open device.
	pData	The buffer of the feature data that will be written to the device. The first byte must be the Report ID. Without the Report ID then enter 0.
	nLen	The length of the feature data that will be written to the device. The length must be the Feature Report length defined by the F/W plus one. The length is FEATURE_SIZE+1 in this template.
Return	1: write succeeds 0: write fails	

Description:

When the SetFeature is called by the host, the device can obtain the data from the SetFeature. Refer to the MCU Program Description in Section 4.2.2 (A).

Function	BOOL GetFeature(HANDLE hDevice, LPVOID pData, DWORD nLen)	
Parameter	hDevice	The handle of the open device.
	pData	The buffer of the feature data that is read out from the device. The first byte must be the Report ID. Without the Report ID then enter 0.
	nLen	The length of the feature data that is read out from the device. The length must be the Feature Report length defined by the F/W plus one. The length is FEATURE_SIZE +1 in this template.
Return	1: read succeeds 0: read fails	

Description:

The host can call GetFeature to obtain the data that is written by the USB device in the GetReport. Refer to the MCU Program Description section 4.2.2 (B).