**Code snippet USB HID Firmware for Cloud BT-2**

MCU Setting

-Clock for USB = 48 MHz

-Enable USB Device Pin

-Select USB Device Full Speed (USB 2.0)

-Select USB Class -> Custom Human Interface Device (CUSTOM\_HID\_USB)

STM32 USB Library Configure for CUSTOM HID

1. Increase HID report size and OUT report size in usbd\_conf.h

**#define USBD\_CUSTOMHID\_OUTREPORT\_BUF\_SIZE 64**

**#define USBD\_CUSTOM\_HID\_REPORT\_DESC\_SIZE 33**

1. In usbd\_customhid.h increase endpoint size to 0x40 or 64 Bytes and change USB customid structure (default structure message size is limited to 2

**#define** CUSTOM\_HID\_EPIN\_SIZE 0x40U

**#define** CUSTOM\_HID\_EPOUT\_SIZE 0x40U

**typedef** **struct** \_USBD\_CUSTOM\_HID\_Itf

{

uint8\_t \*pReport;

int8\_t (\* Init) (**void**);

int8\_t (\* DeInit) (**void**);

int8\_t (\* OutEvent) (uint8\_t\* );

} USBD\_CUSTOM\_HID\_ItfTypeDef;

1. In file usbd\_customhid.c is optional change of bInterval value to get faster response from the device

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Descriptor of Custom HID endpoints \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 27 \*/

0x07, /\*bLength: Endpoint Descriptor size\*/

USB\_DESC\_TYPE\_ENDPOINT, /\*bDescriptorType:\*/

CUSTOM\_HID\_EPIN\_ADDR, /\*bEndpointAddress: Endpoint Address (IN)\*/

0x03, /\*bmAttributes: Interrupt endpoint\*/

CUSTOM\_HID\_EPIN\_SIZE, /\*wMaxPacketSize\*/

0x00,

0xa, /\*bInterval: Polling Interval (10 ms)\*/

/\* 34 \*/

0x07, /\* bLength: Endpoint Descriptor size \*/

USB\_DESC\_TYPE\_ENDPOINT,/\* bDescriptorType: \*/

CUSTOM\_HID\_EPOUT\_ADDR, /\*bEndpointAddress: Endpoint Address (OUT)\*/

0x03,/\* bmAttributes: Interrupt endpoint \*/

CUSTOM\_HID\_EPOUT\_SIZE,/\* wMaxPacketSize\*/

0x00,

0xa,/\* bInterval: Polling Interval (10 ms) \*/

/\* 41 \*/

1. In file usbd\_customhid.c change call of OUT events

**static** uint8\_t **USBD\_CUSTOM\_HID\_DataOut** (USBD\_HandleTypeDef \*pdev, uint8\_t epnum)

{

USBD\_CUSTOM\_HID\_HandleTypeDef \*hhid = (USBD\_CUSTOM\_HID\_HandleTypeDef\*)pdev->pClassData;

((USBD\_CUSTOM\_HID\_ItfTypeDef \*)pdev->pUserData)->OutEvent(hhid->Report\_buf);

USBD\_LL\_PrepareReceive(pdev, CUSTOM\_HID\_EPOUT\_ADDR , hhid->Report\_buf,

USBD\_CUSTOMHID\_OUTREPORT\_BUF\_SIZE);

**return** *USBD\_OK*;

}

uint8\_t **USBD\_CUSTOM\_HID\_EP0\_RxReady**(USBD\_HandleTypeDef \*pdev)

{

USBD\_CUSTOM\_HID\_HandleTypeDef \*hhid = (USBD\_CUSTOM\_HID\_HandleTypeDef\*)pdev->pClassData;

**if** (hhid->IsReportAvailable == 1)

{

((USBD\_CUSTOM\_HID\_ItfTypeDef \*)pdev->pUserData)->OutEvent(hhid->Report\_buf);

hhid->IsReportAvailable = 0;

}

**return** *USBD\_OK*;

}

1. In file usbd\_custom\_hid\_if.c add buffer for user USB message and declared UART handle type define

uint8\_t buffer[64];

UART\_HandleTypeDef huart1;

1. And in file usbd\_custom\_hid\_if.c add HID report descriptor

/\* USER CODE BEGIN 0 \*/

0x06, 0x00, 0xff, //Usage Page(Undefined )

0x09, 0x01, // USAGE (Undefined)

0xa1, 0x01, // COLLECTION (Application)

0x15, 0x00, // LOGICAL\_MINIMUM (0)

0x26, 0xff, 0x00, // LOGICAL\_MAXIMUM (255)

0x75, 0x08, // REPORT\_SIZE (8)

0x95, 0x40, // REPORT\_COUNT (64)

0x09, 0x01, // USAGE (Undefined)

0x81, 0x02, // INPUT (Data,Var,Abs)

0x95, 0x40, // REPORT\_COUNT (64)

0x09, 0x01, // USAGE (Undefined)

0x91, 0x02, // OUTPUT (Data,Var,Abs)

0x95, 0x01, // REPORT\_COUNT (1)

0x09, 0x01, // USAGE (Undefined)

0xb1, 0x02, // FEATURE (Data,Var,Abs)

/\* USER CODE END 0 \*/

0xC0 /\* END\_COLLECTION \*/

1. And modify CUSTOM\_HID\_OutEvent\_FS function declaration and definition

// declaration

**static** int8\_t **CUSTOM\_HID\_OutEvent\_FS**(uint8\_t\* state);

// definition

**static** int8\_t **CUSTOM\_HID\_OutEvent\_FS**(uint8\_t\* state)

{

/\* USER CODE BEGIN 6 \*/

// Copy Received data to the buffer

**memcpy**(buffer, state, 64 \* **sizeof**(uint8\_t));

// this function return data was sent from HID Terminal to display on "received data" box

USBD\_CUSTOM\_HID\_SendReport(&hUsbDeviceFS,(uint8\_t\*)buffer,64);

// send an array of data to process and Set the new Bluetooth device name

USBD\_HID\_SetBluetoothname(buffer);

**return** (0);

/\* USER CODE END 6 \*/

}

**Implemented Core Function is below**

1. In file usbd\_custom\_hid\_if.c and function that receive data from USB Terminal to stored mcu buffer then process and transmit to Bluetooth module via UART Protocol

**void** **USBD\_HID\_SetBluetoothname**(uint8\_t\* usbbuffer)

{

uint8\_t Packet[64];

// usbbuffer[0] stored data of len that received from USB Terminal and plus UART command\_id stored 1 Byte.

uint8\_t command\_len = usbbuffer[0] + 1U;

// Packet[Checksum\_Position] is a position of the checksum is next by 3 Bytes from the command length

uint8\_t chksum\_pos = command\_len + 3U;

// Packet[end\_of\_packet] is the last index of an array are transmitted by UART

uint8\_t packet\_end = chksum\_pos + 1U;

Packet[0] = 0xAA; // BT\_Module UART Packet Header

Packet[1] = 0x00; // Packet Lenth \*MSB\*

// Packet Lenth \*LSB\* -> Command\_ID (1 Byte) + Parameters (Shouldn't over 32 Bytes)

Packet[2] = command\_len;

// UART Command 0x05 \*Change Bluetooth device name on discovery mode\*

Packet[3] = 0x05;

// copy buffer that receives from USB to new buffer to send via UART Protocol.

**memcpy**(&Packet[4] , &usbbuffer[1], chksum\_pos);

// Add checksum at the tail of the UART Packet.

Packet[chksum\_pos] = calculateChecksum(&Packet[2],&Packet[packet\_end]);

// Transmit Command Packet via MCU UART

// from head(packet[0]) to the tail(checksum)

**for**(uint8\_t i = 0; i < packet\_end; i++)

{

HAL\_UART\_Transmit(&huart1, &Packet[i], 1, 100);

}

}