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
```

2. 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;
```

3. 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 */
```

4. 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;
}
```

5. 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;
```

6. 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              */
```

7. 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

8. 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);
	}

}
```