Custom HID (HAL)

Petr Novák / novakpe@cvut.cz / 29.04.2022

Obsah

1 Úvod 1
2 Vytvoření Custom USB-HID pomocí STM32CubeMX 1
3 Komunikace 4
3.1 Komunikace na straně ARM 5
3.1.1 Příjem 5
3.1.2 Odeslání 5
3.2 Komunikace na straně PC 6
3.2.1 Nalezení všech dostupných USB-HID zařízení 6
3.2.2 Výběr cílového zařízení 6
3.2.3 Otevření vybraného zařízení 6
3.2.4 Odeslání dat 7
3.2.5 Příjem dat 7
4 Poznámky a postřehy 8

1 Úvod

Text popisuje postup vytvoření zařízení **Custom USB-HID** (Custom Human Interface Device Class) pomocí STM32CubeIDE a procesoru typu STM32. Jedná se pouze o stručný / orientační postup a může se lišit podle použitého procesoru.

Poznámky:

- Jsou uváděny pouze názvy souborů a názvy funkcí/dat nikoli řádky ve zdrojových souborech, ty se mohou lišit nejen podle typu použitého procesoru, ale i verze souborů / knihoven.
- Směr přenosu OUT / IN je vždy z pohledu PC (OUT – z PC / IN – do PC).

2 Vytvoření Custom USB-HID pomocí STM32CubeMX

Zařízení typu **Custom USB-HID** je vhodné vytvořit pomocí návrháře projektu v STM32CubeIDE (jde o celkem složitý proces). Stručný postup Vytvoření projetu je následující:

- Vytvořit základní projekt pro požadovaný procesor (nastavit frekvenci, SWD, …).
- Povolit / nastavit USB a vybrat typ USB jako „**Custom Human Interface Device Class** (HID)“. (Ne pouze „Human Interface Device Class (HID)“.)

    - U typu STM32F042 je nutno v **Systém Core** a dále **SYS** zatrhnout položku **Pins PA11/12 insteand of pins PA9/10**. Teprve potom lze povolit USB.
    - Povolit USB v **Connectivity** a dále v **USB** aktivací položky **Device (FS)**.
    - Nastavit typ USB v **Middleware** a dále **USB Device**, kde vybrat **Custom Human Interface Device Class (HID)**.
- Vygenerovat projekt a otevřít jej.

Vložení kódu do Custom Human Interface Device Class (HID):

Důležité soubory:

- **USB_DEVICE/Target/usbd_conf.h** – nastavení konstant / parametrů pro USB
- **USB_DEVICE/App/usbd_custom_hid_if.c** – zápis descriptoru USB zařízení
- **USB_DEVICE/App/usbd_desc.c** – VIP, PID, texty do názvu zařízení

Název HID zařízen:

V souboru **USB_DEVICE/App/usbd_desc.c** jsou konstanty tvořící popis zařízení. Z pohledu knihovny / aplikace se název zařízení skládá z těchto částí: **Manufacturer** / **Product** / **SerialNumber** / **ReportID**

- **Manufacturer** - USBD_MANUFACTURER_STRING
- **Product** - USBD_PRODUCT_STRING_FS
- **SerialNumber** - USBD_SERIALNUMBER_STRING_FS
- **ReportID** – Přiřazováno automaticky. Pokud není složený descriptor, tak není vůbec přítomno. Pokud je descriptor složen z více částí, tak je na konci názvu zařízeni „/ 01“ nebo „/ 02“ atd. podle počtu dílčích částí v descriptoru.

Descriptor / Report(y) – Popisuje tzv. **Report** zasílaný do / z HID zařízení z pohledu PC. Data do / z PC se vždy přenášejí pomocí definovaných reportů (bloku bytes).

- Vložit descriptor do souboru **USB_DEVICE/App/usbd_custom_hid_if.c** do pole **CUSTOM_HID_ReportDesc_FS[USBD_CUSTOM_HID_REPORT_DESC_SIZE]**.
- V souboru **USB_DEVICE/Target/usbd_conf.h** nastavit hodnotu položky **USBD_CUSTOM_HID_REPORT_DESC_SIZE** na délku vloženého descriptoru (počet bytes).
- Pokud descriptor obsahuje / popisuje více reportů, tak první část má ReportID=1, druhá ReportID=2, atd. Pokud však deskcriptor obsahuje pouze jeden report, tak se ReportID neuvádí (je „0“).

Příklad descriptoru obsahujícího pouze jeden report je následující:

```
/* 22B Custom CONTROL ID=0 (00) */
// z PC do ARM 10x 8bit hodnot (OUT), celková velikost reportu = 1 (ReportID) + 10 (data)
// z ARM do PC 10x 8bit hodnot (IN) , celková velikost reportu = 1 (ReportID) + 10 (data)
0x06, 0xAC, 0xFF,       // USAGE_PAGE (Vendor Defined Page 1)     // 3 B
0x09, 0x00,             // USAGE (Undefined)
0xA1, 0x01,             // COLLECTION (Application)
0x75, 0x08,             //   REPORT_SIZE (8) 8bits = 1 byte
0x95, 0x0A,             //   REPORT_COUNT (10) 10x byte
0x92, 0xA3, 0x01,       //   OUTPUT (Cnst, Var, Abs, NPrf, Vol, Buf)
0x75, 0x08,             //   REPORT_SIZE (8) 8bits = 1 byte
```

```
0x95, 0x0A,          //  REPORT_COUNT (10) 10x byte
0x82, 0xA3, 0x01,    //  INPUT (Cnst, Var, Abs, NPrf, Vol, Buf)
0xC0,                // END_COLLECTION
```

Descriptor popsuje pouze jeden report pro OUT a IN obsahující vždy 10 (0x0A) bytes. Celkový report má velikost 11 bytes, protože ve skutečnosti obsahuje ještě ReportID, které je vždy „0“. Maximální délka datové části reportu je 64 bytes.

Dále je potřeba nastavit:

- V souboru **USB_DEVICE/Target/usbd_conf.h** nastavit položku **USBD_CUSTOM_HID_REPORT_DESC_SIZE**. Jde o buffer, kam se ukládají data přijatá z PC než si je vyzvedne aplikace v ARMu. Pozor však na skutečnost, že teprve až je tento buffer plný, tak se vyvolá funkce pro vyzvednutí dat **USBD_CUSTOM_HID_DataOut(…)** (pospaná dále). Tedy pokud je tento buffer jako velikost reportu, tak se funkce **USBD_CUSTOM_HID_DataOut(…)** vyvolá po každém přijatém reportu. Avšak pokud je 2x velikost reportu, tak se funkce **USBD_CUSTOM_HID_DataOut(…)** vyvolá teprve, až při přijmu dvou reportů.
- V **Middlewares/ST/STM32_USB_Device_Library/Class/CustomHID/Src/usbd_customhid.h**. nastavit položky CUSTOM_HID_EPIN_SIZE a CUSTOM_HID_EPOUT_SIZE na celkovou velikost přenášených reportů. Tedy velikost datové části + ReportID. V tomto případě obě na 11.

I pro jedno HID zařízení však lze číslo REPORT ID uvést a to následovně

```
/* 24B Custom CONTROL ID=1 (01) */
// z PC do ARM 10x 8bit hodnot (OUT), celková velikost reportu = 1 (ReportID) + 10 (data)
// z ARM do PC 10x 8bit hodnot (IN) , celková velikost reportu = 1 (ReportID) + 10 (data)
0x06, 0xAC, 0xFF,    // USAGE_PAGE (Vendor Defined Page 1)      // 3 B
0x09, 0x00,          // USAGE (Undefined)
0xA1, 0x01,          // COLLECTION (Application)
0x85, 0x01,          //  REPORT_ID (1)
0x75, 0x08,          //  REPORT_SIZE (8)
0x95, 0x0A,          //  REPORT_COUNT (10)
0x92, 0xA3, 0x01,    //  OUTPUT (Cnst, Var, Abs, NPrf, Vol, Buf)
0x75, 0x08,          //  REPORT_SIZE (8)
0x95, 0x0A,          //  REPORT_COUNT (10)
0x82, 0xA3, 0x01,    //  INPUT (Cnst, Var, Abs, NPrf, Vol, Buf)
0xC0,                // END_COLLECTION
```

Pozor však na skutečnost, že pokud je REPORT ID uvedeno, tak nesmí být 0. V tomto případě má opět skutečný / celý report 11 bytes (REPORT ID 1B + DATA 10B), ale při komunikaci je nutno vždy nastavit **Report[0] = 1**, protože je v popisu deskriptoru zadáno REPORT_ID (1).

Příklad descriptoru vytvářející dvě HID zařízení

```
/* 24B Custom HID s ReportID = 1 */
// z PC do ARM 10x 8bit hodnot (OUT), celková velikost reportu = 1 (ReportID) + 10 (data)
// z ARM do PC 10x 8bit hodnot (IN) , celková velikost reportu = 1 (ReportID) + 10 (data)
0x06, 0xAC, 0xFF,       // USAGE_PAGE (Vendor Defined Page 1)      // 3 B
0x09, 0x00,             // USAGE (Undefined)
0xA1, 0x01,             // COLLECTION (Application)
0x85, 0x01,             //   REPORT_ID (1)
0x75, 0x08,             //   REPORT_SIZE (8)
0x95, 0x0A,             //   REPORT_COUNT (10)
0x92, 0xA3, 0x01,       //   OUTPUT (Cnst, Var, Abs, NPrf, Vol, Buf)
0x75, 0x08,             //   REPORT_SIZE (8)
0x95, 0x0A,             //   REPORT_COUNT (10)
0x82, 0xA3, 0x01,       //   INPUT (Cnst, Var, Abs, NPrf, Vol, Buf)
0xC0,                   // END_COLLECTION
/* 24B Custom HID s ReportID = 2 */
// z PC do ARM 10x 8bit hodnot (OUT), celková velikost reportu = 1 (ReportID) + 10 (data)
// z ARM do PC 10x 8bit hodnot (IN) , celková velikost reportu = 1 (ReportID) + 10 (data)
0x06, 0xAC, 0xFF,       // USAGE_PAGE (Vendor Defined Page 1)      // 3 B
0x09, 0x00,             // USAGE (Undefined)
0xA1, 0x01,             // COLLECTION (Application)
0x85, 0x02,             //   REPORT_ID (2)
0x75, 0x08,             //   REPORT_SIZE (8) [mimo ReportID]
0x95, 0x0A,             //   REPORT_COUNT (10)
0x92, 0xA3, 0x01,       //   OUTPUT (Cnst, Var, Abs, NPrf, Vol, Buf)
0x75, 0x08,             //   REPORT_SIZE (8)
0x95, 0x0A,             //   REPORT_COUNT (10) [mimo ReportID]
0x82, 0xA3, 0x01,       //   INPUT (Cnst, Var, Abs, NPrf, Vol, Buf)
0xC0,                   // END_COLLECTION
```

První hodnota v popisu / definici reportu musí být vždy **ReportID**.

(Jak definovat hodnoty pro velikost reportů?)
Poznámka: Více reportů je vhodné pro rozdělení přenášených dat podle nějakých například komunikačních kanálů do/z zařízení. Místo více reportů doporučuji však vytvářet pouze jeden report (klidně delší) a jeho první datový byte použít jako identifikaci o jaký komunikační kanál / povel / data se jedná, tedy v podstatě jako náhradu za hodnotu ReportID (toto je mnohem jednodušší pro konfiguraci HID zařízení).

Na straně ARM i PC musí být dodržena přesná / stejná velikost reportu, jinak komunikace selhává!

3 Komunikace

Pro komunikaci je nutno v ARMu udělat nějaké úpravy a na PC/Win použít připravené zdrojové soubory.

3.1 Komunikace na straně ARM

3.1.1 Příjem

Pro příjem dat je určena funkce **CUSTOM_HID_OutEvent_FS(…)** nacházející se v souboru **Middlewares/ST/STM32_USB_Device_Library/Class/CustomHID/Src/usbd_customhid_if.c**. Funkce je volána automaticky od USB při přijmu reportu z PC (přesněji řečeno po naplnění bufferu o velikosti **USBD_CUSTOMHID_OUTREPORT_BUF_SIZE**).

Funkce je definována jako **static**. Pro její použití doporučuji tento postup:

- Funkci **CUSTOM_HID_OutEvent_FS(…)** překopírovat do vlastního souboru a v souboru **usbd_customhid_if.c** ji zcela za-komentovat.
- V deklaraci formátu funkce **CUSTOM_HID_OutEvent_FS(…)** odstranit označení **static** (například /* static */).
- Její dva parametry **uint8_t event_idx, uint8_t state** změnit (například) na jeden parametr typu **uint8_t *buff** (jak v implementaci, tak i deklaraci hlavičky funkce). Velikost bufferu není potřeba předávat, protože odpovídá velikosti definovaného reportu (ale zle ji samozřejmě do hlavičky funkce doplnit).
- Parametry funkce je rovněž potřeba změnit i tam kde se funkce volá a to:
    - Soubor **usbd_customhid_if.c** ve funkci **USBD_CUSTOM_HID_DataOut(…)**. Změnit její volání na **->OutEvent(hhid->Report_buf)**.
    - Soubor **usbd_customhid_if.c** ve funkci **USBD_CUSTOM_HID_EP0_RxReady(…)**, zle její volání zcela za-komentovat.

Ve své (překopírované) funkci **CUSTOM_HID_OutEvent_FS(…)** je vhodné přijatá data okamžitě překopírovat do svého (jiného) bufferu a vytvořit si nějaký příznak příjmu dat a data zpracovat až v hlavní smyčce programu. Nezpracovávat data v události od jejich přijmu (brzdila by se USB komunikace).

3.1.2 Odeslání

Odeslání dat se vykonává pomocí funkce **USBD_CUSTOM_HID_SendReport_FS(uint8_t *report, uint16_t len)** v souboru **USB_DEVICE/App/usbd_custom_hid_if.c**:

- Funkci je potřeba od-komentovat a raději zrušit označení jako **static** (například /* static */).
- Ve vlastních souborech, kde je použita funkce **USBD_CUSTOM_HID_SendReport_FS(…)** pro odeslání dat, vložit její deklaraci **int8_t USBD_CUSTOM_HID_SendReport_FS(uint8_t *report, uint16_t len)**.

Vstup:

- **report** - buffer s daty pro odeslání (bytes).
- **len** - počet odesílaných bytes z bufferu.

Výstup:

- USBD_OK (0x00) / USBD_BUSY.

Pokud je vytvořen report bez ReportID, tak má v ARMu přijatý / vysílaný report délku pouze jeho datové části. Tedy do funkce **CUSTOM_HID_OutEvent_FS(…)** vstupuje report obsahující pouze datové bytes, nikoli s prvním byte obsahujícím ReportID. Funkce **USBD_CUSTOM_HID_SendReport_FS** se volá rovněž s bufferem obsahujícím pouze datovou část reportu, ReportID se neuvádí.

3.2 Komunikace na straně PC

Vytvořeny zdrojové soubory pro základní komunikaci s **Custom USB-HID**

3.2.1 Nalezení všech dostupných USB-HID zařízení

// detekuje všechna připojená **Custom USB-HID** zařízení
List<XUtils.USBHID.USBHIDInfo> devices = XUtils.USBHID.USBHIDMain.GetAll()
// snadno lze zobrazit základní informace o nalezených zařízeních
devices[x].ToString()

3.2.2 Výběr cílového zařízení

Třída **USBHIDInfo** obsahuje položky, podle kterých lze hledat / detekovat požadovaná zařízení:
Vid, **Pid**, **Manufacturer**, **Product**, **SerialNumber**
(ideální jsou **Manufacturer**, **Product**, **SerialNumber**)

3.2.3 Otevření vybraného zařízení

Otevřít požadované **Custom USB-HID** lze například pomocí
// zadáním odkazu na **USBHIDInfo**
device = Commun.HID.HIDUSBMain.OpenDevice(devices[x]);
nebo
// zadáním popisu
// (tento popis lze uložit do konfiguračního souboru aplikace pro pozdější otevření tohoto zařízení)
device = Commun.HID.HIDUSBMain.OpenDevice(devices[x].ToString());
// pokud se zařízení podařilo úspěšně otevřít
if (device != null)
{

```
    // pokud je potřeba velikost OUT reportu (PC -> ARM)
    … = device.devInfo.OutputReportBytesLength;
    // pokud je potřeba velikost IN reportu (ARM -> PC)
    … = device.devInfo.InputReportBytesLength;

    // pokud je potřeba, tak spuštění přijmu samostatným vláknem / událostí
    device.ReceiveByThread = true;                  // příjem pomoci vlákna
    device.OnReceiveReportIn = ReceiveReportIn; // metoda pro obsluhu příjmu
    device.Start();                                 // spuštěni obsluhy příjmu
```

}

Otevřené zařízení obsahuje buffery pro vysílání / příjem dat pod názvy **ReportOutBytes[…]** a **ReportInBytes[…]**. Jejich potřebná velikost je detekována automaticky z aktuálně otevřeného zařízení. Stačí tedy do **ReportOutBytes[…]** data vložit a poté vyslat, nebo po příjmu data z **ReportInBytes[…]** vyzvedávat.
Pozor na skutečnost, že report na PC/Win obsahuje vždy ReportID. Na první pozici reportu **ReportOutBytes[…]** před jeho odesláním, je nutno vložit správné ReportID (pokud není ReportID v deskriptoru uvedeno, tak vložit hodnotu „0“). Při příjmu na první pozici reportu **ReportInBytes[…]** je rovněž vždy ReportID (pokud není ReportID v deskriptoru uvedeno, tak je zde „0“).

3.2.4 Odeslání dat

```
// nejprve připravit buffer pro OUT report
// první byte je vždy REPORT ID
device.ReportOutBytes[0] = ReportID;
// další bytes jsou DATA (maximálně podle hodnoty v HID deskriptoru v ARMu)
device.ReportOutBytes[1] = data1;
…
device.ReportOutBytes[N] = dataN;
// odeslání na USB-HID zařízení (report je odeslán okamžitě)
device.ReportSend();
```

3.2.5 Příjem dat

Příjem lze uskutečnit dvěma způsoby:

- A) Dotazováním – periodické dotazování zda něco přišlo
- B) Pomocí události – automatická událost při příjmu (běžícím na pozadí)

Obě metody lze samozřejmě kombinovat

A) Pomocí pouze periodického dotazování lze využívat například posloupnosti metod:

```
// …
device.ReportOutSend();          // odeslání reportu
device.ReportInReceive();        // čekání na příjem reportu
// …
```

Toto je vhodné většinou pro komunikaci stylem dotaz – odpověď. Tedy kdy je na každý zaslaný report do USB-HID zařízení očekávána nějaká odpověď. Vysílaný report OUT je vždy v **device.ReportOutBytes[…]** a přijatý report IN je vždy v **device.ReportInBytes[…]**. Při tomto stylu použití komunikace se nesmí nastavit **device.ReceiveByThread = true** a nesmí se volat metoda **device.Start()**, tedy nesmí být spuštěn příjem pomocí vlákna / události!

B) Příjem pomocí události je poněkud univerzálnější. Výstupní report (**device.ReportOutBytes[…]**) lze samozřejmě odeslat kdykoli metodou **device.ReportSend()**. Při příjmu vstupního (IN) reportu je automaticky volaná metoda zadaná pomocí **device.OnReceiveReportIn = ReceiveReportIn**. Volaná metoda má tvar **void OnReportIn(XUtils.USBHID.USBHIDInfo devInf, byte[] report)**. Přijatý report

je dostupný pouze pomocí parametru předaného do této metody, tedy nikoli v **device.ReportInBytes[…]**. Přijatý report IN je vhodné co nejrychleji odebrat a neblokovat volanou událost. Samozřejmě je nutno nastavit **device.ReceiveByThread = true** a pro spuštění tohoto typu příjmu volat metodu **device.Start()**. Parametr metody nazvaný **devInf** obsahuje informace o USB-HID zařízení, které událost / příjem dat vyvolalo. Při otevření několika USB-HID zařízení, lze tedy všechny události přesměrovat na jednu obslužnou metodu a například pomocí výrobního / sériového čísla detekovat zdroj události / dat.
Pokud je spuštěn příjem pomocí události / vlákna a přesto je potřeba vyslat report a snadno počkat na odpověď (typem dotaz – odpověď), tak lze použít metodu **device.ReportSendAndReceive(…)**. Report OUT je vyslán okamžitě a lze vykonávat vlastní kontrolu na příjem odpovědi z průběžně přicházejících paketů. Kontrolu na příjem požadované odpovědi si lze definovat pomocí metody vložené do **dev.IsThisAnswer = IsThisAnswer**, kde volaná metoda bude mít tvar / obsah například:

```
bool IsThisAnswer(byte[] reportOut, byte[] reportIn)
{
        // pouze pokud souhlasí „Cmd“ a „CmdSub“, tak je to očekávaná odpověď
        return ((reportOut[1] == reportIn[1]) && (reportOut[2] == reportIn[2]);
}
```

Pokud report IN obsahuje očekávanou odpověď, tak je vráceno řízení z metody **device.ReportSendAndReceive()** s návratovou hodnotou **USBHIDConsts.EnCmnStatus.OK**, jinak se čeká dále. Přijatá odpověď, tedy přesněji řečeno report IN, je uložena v **device.ReportInBytes[…]**. Pokud však do zadaného časového intervalu nepřišla očekávaná odpověď, tak je vráceno **USBHIDConsts.EnCmnStatus.TimeOut**.
Toto je vhodné například pokud asynchronně přicházejí IN reporty s daty (průběžně se zpracovávají) a současně (mezi nimi) je potřeba do zařízení vysílat povely (spuštění, zastavení, nastavení) a vždy snadno počkat na potvrzení jejich vykonání.

4 Poznámky a postřehy

Zde jsou uvedeny některé poznámky a postřehy vztahující se zejména k tématu **Custom USB-HID** pro procesory typu STM32:

- Nepovedlo se mi úspěšně zprovoznit **Custom USB-HID** na procesoru STM32F042. Lze data kontinuálně libovolně krát vysílat a kontinuálně libovolně krát přijímat. Pokud se však střídá vysílání / příjem, tak po druhém kole vytvořená komunikace zatuhne. Zřejmě se někde něco přepíše. Na WWW jsem našel informaci, že je potřeba zvětšit přijímací buffer, ale nezjistil jsem kde / jaký / jak.