



**ČESKÉ
VYSOKÉ
UČENÍ
TECHNICKÉ
V PRAZE**

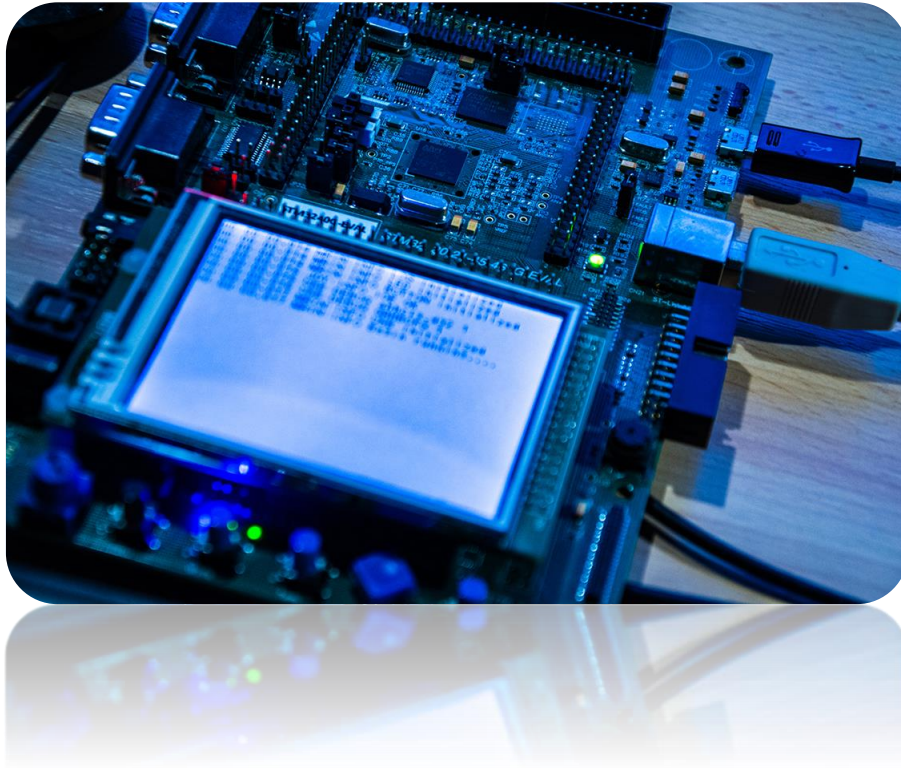
České Vysoké Učení Technické v Praze
Fakulta Elektrotechnická

KRP – Komunikační rozhraní počítačů

USB Device

Závěrečná zpráva

USB Device Mouse/Joystick pro SMT3240G-EVAL



Zadání

Cílem semestrální práce je naprogramovat USB zařízení v MCU STM32F407. Podmínkou je implementace enumeračního procesu a aplikační vrstvy. Aplikační vrstva by měla splňovat určitou třídu zařízení (např. HID), po dohodě se cvičícím je možné navrhnout i vlastní aplikační vrstvu.

Za každý týden zpoždění je stržen jeden bod z konečného bodového zisku. Podmínkou udělení zápočtu je odevzdání všech částí zmíněných v tabulce harmonogram.

Nedílnou částí každé semestrální práce je odevzdání závěrečné zprávy, která popisuje vývoj zařízení, vystihuje zajímavé použité algoritmy, obsahuje vývojový diagram a obsahuje návod na práci se zařízením.

Popis

- Blikání LED pomocí přerušení: MCU bude blikat LEDkami, hlavní programová smyčka bude prázdná a samotné blikání bude řešeno softwarově v obsluze časovače.
- Ladicí konzole: K dispozici je implementovaná grafická knihovna pro LCD, úkolem je implementovat textovou konzoli pro výpis ladicích informací. Program bude schopen vypsát textové informace a čísla v šestnáctkové, volitelně i jiné, soustavě.
- Zachycení resetu a parsování setup rámců: Pomocí přerušení bude zachycena událost resetu od hosta, bude otevřen endpoint 0 pro příjem setup rámců a bude implementován náznak parsování přijatých požadavků.

- Přijetí požadavku set address: Zařízení nastaví přijatou adresu a odešle potvrzení.
- Enumerace: Dokončení procesu enumerace, tedy odeslání zbylých deskriptorů a konfigurací.
- Aplikační vrstva: Implementace aplikační třídy dle dokumentace tříd.

Zvolené řešení

Jako nejvíce vhodné se mi zdálo implementovat zařízení typu HID (myš), které bude schopno simulovat pomocí joysticku a dvou tlačítek klasickou myš.

Popis postupu práce

HW Nastavení desky STM3240G-EVAL

Power related jumpers

- JP18 Nastaven na STlk, čili tak, aby deska byla napájena z ST-LINK/V2 USB, nikoliv z připojeného USB OTG FS.
- Je spojen JP32 aby MCU_VDD bylo připojeno k 3.3V napětí.
- Switche boot0 a boot1 jsou ve stavu 00, takže deska startuje z User Flash
- Ostatní nastavení jumperů nemá na chod aplikace vliv.

SW Nastavení HW vnitřních periférií

Nastavení hodin

- HSE ON
- AHB clock(HCLK) = SYSCLK
- Low Speed APB clock = HCLK/4
- High Speed APB clock = HCLK/2
- PLL clock source = 48Mhz
- Source clock = PLL source

Timer

Povolil jsem zdroj hodin pro periférii APB1. Zvolil jsem Timer3 a nastavil periodu a předděličku tak, aby přerušovala každých 100ms.

LED

Povolil jsem hodiny na portech GPIOG, GPIOC, GPIOI a nastavil piny PG6, PG8, PC7, PI9. Všechny jako PullUp s rychlostí 50Mhz

Button

- Tlačítka Tamper a Key
- AHB1 clk
- Porty GPIOG a GPIOC
- Piny PG15 a PC13
- PullUp rezistory

Joystick

Jelikož je joystick připojeno pomocí IO Expanderu, byla pro komunikaci použita již hotová knihovna `stm324xg_eval_ioe`, která dokáže komunikovat přes I²C s IO Expanderem a získávat pohyb joysticku. Z knihovny jsem použil pouze funkce `IOE_Config()` pro inicializaci a `IOE_JoystickGetState()` pro získání stavu(stisku) joysticku.

USB

- Povolit hodiny na AHB1 pro port GPIOA
- Piny PA8, PA9, PA11, PA12, PA10
- Všechny piny na port GPIOA
- Piny 8,11,12 jsem nakonfiguroval na
 - alternativní funkci GPIO_AF_OTG1_FS
 - GPIO rychlost 100Mhz
 - No pull
- Pin 9 - VBUS
 - Místo alternativní funkce MODE IN
 - No pull
 - GPIO rychlost 100Mhz
- Pin 10 - ID
 - Pull up
 - Rychlost 100Mhz
 - alternativní funkci GPIO_AF_OTG1_FS
- Povolit High Speed APB2 pro SYSCFG
- Povolit Low Speed AHB2 pro OTG FS
- Nastavit NVIC přerušení od OTG_FS_IRQn
 - Priority = 1
 - Subpriority = 3

Struktura projektu

Projekt je rozdělen do několika hlavních částí, které mezi sebou spolupracují.

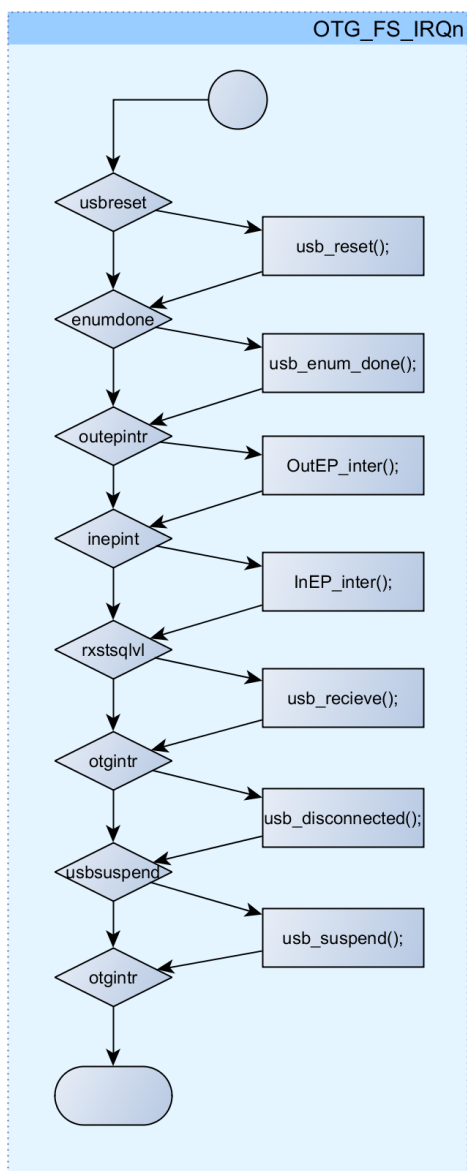
- HAL – Hardware Abstrakt Library – zde jsou vytknuta všechna nastavení periférií
- HW TIMER – 100ms timer s možností registrovat funkce, které se budou volat každých 100ms
- JOYSTICK – Podpora práce s joystickem
- LCD – Podpora výpisů na displej
- LED – Podpora práce s LED diodami. Možnost blikání, zapínání a vypínání.
- LOG – Podpora výpisů s časem, jak na displej, tak do sériové linky. Možnost určit úroveň a zdroj výpisu.
- SERIAL – podpora sériové linky
- USB – Knihovna pro práci s USB

Celý projekt má „Entry point“ v souboru main.cpp ve funkci main()

USB

Pro práci s USB registry jsem použil již hotové struktury definované v souboru usb_regs.h

1. Nejprve jsem nastavil daným proměnným, které reprezentují dané druhy registrů, příslušné adresy v paměti
 - a. např. usbDREGS = USB_D_BASE; kde USB_D_BASE je makro vracející adresu 0x50000800
2. Podle manuálu jsem nastavil registry GUSBCFG, AHBCFG, GUSBCFG
3. Provedl počáteční nastavení EP
4. A nastavil, které interrupty chci přijímat v GINTMSK
5. Nakonec jsem počkal na USB RESET a tím začíná kolečko s IRQ přerušeními



IRQ USB

V obsluze přerušení se pak volají funkce, které dle manuálu procesoru obsluhují daný interrupt.

Usb Reset

- Smaže všechny interrupty na IN a OUT EP
- Resetuje adresu USB
- Připraví EP0 pro příjem SETUP packetu
- A aktivuje EP0 IN a EP0 OUT

Usb enum

- Nastaví registr GUSBCFG na správnou rychlost (FULL SPEED)
- Nastaví EP0 rychlost enumerace

Usb recieve

- Přejme příchozí paket z příslušné FIFO. V případě aplikace z EP0 OUT

OUT EP interrupt

- Zde probíhá parsování dat z přijatého setup packetu do struktury *SUsbSetupPckt*, která má stejnou strukturu jako USB SETUP packet

- bmRequest
- bRequest
- wValue
- wIndex
- wLength

Pro položky větší než 2B prohazují byty podle endianness.

Na základě bmRequest se volá příslušná funkce, která zpracovává obsluhu SETUP packetu.

- DEVICE
 - Get short descriptor

- Set address
 - Nastavím v DCFG registru adresu dle wValue
- Get long descriptor
 - Device descriptor
 - Configuration descriptor
 - String descriptor – neobsluhuji, i když jsem to zkoušel
- Set configuration
- INTERFACE
 - Při dotazu na get descriptor odešlu HID descriptor myši/joysticku

IN EP interrupt

Všetchna data, která jsem si připravil ve fázi OUT EP interrupt, teď odešlu na EP0 IN. (Zkopíruji do příslušné FIFO)

- CNAK = 1
- EPENA = 1
- A povolit přerušení pro daný EP

Struktura dat

ZLP – zero length packet

Nemá žádnou délku, tak ani žádnou strukturu. Použito pro oznámení ukončení přenosu.

Deskriptory

Deskriptory se posílají jako pole bajtů, které se do FIFO kopíruje po 4B, u 2B prvků deskriptorů se na nižším indexu pole bajtů uvádí bajt s menší hodnotou.

Pohyb myši

Pohyb myši je strukturován do pole (uint8_t buff[4]), kde:

- 0. Bajt a jeho 0. a 1. Bit indikují stisk tlačítek
- 1. Bajt pohyb po ose x
- 2. Bajt pohyb po ose y
- 3. Bajt je 0

Enpointy

V aplikaci jsou použity pouze 3 EP.

- EP 0 OUT – pro příjem kontrolních dat
- EP 0 IN – pro odesílání kontrolních dat (např. deskriptorů)
- EP 1 IN – pro odesílání pohybu myši

Vývoj a Závěr

Vývoj aplikace probíhal v softwaru IAR Embedded Workbench, kde byla prováděna převážně kompilaci a debug, a v software MS Visual Studio 2015 s pluginem VisualAssistX, pro lepší orientaci v kódu, přecházení mezi funkcemi a definicemi a dalšími výhodami.

Pro debug aplikace byl také použit další SW, který se snažilo zachytit komunikaci mezi deskou a počítačem, ale žádný z nich nezachytával usb pakety na tak nízké úrovni, aby byl použitelný. Proto se nejlépe osvědčil osciloskop ve škole.

Celkový vývoj probíhal s použitím dokumentace pro STM32F407, uživatelským manuálem pro desku STM3240G-EVAL a ukázkovými zdrojovými kódy nalezenými na fórech na internetu.

Pro práci s USB registry byly použity již hotové struktury definované v souboru usb_regs.h

Příloha

Device descriptor

```
__IO static uint8_t DevDescM[USB_SIZ_DEVICE_DESC] =
{
    0x12,                ///< bLength
    EVD_DEVICE,          ///< bDescriptorType
    0x00,                ///< bcdUSB
    0x02,                ///<
    0x00,                ///< bDeviceClass
    0x00,                ///< bDeviceSubClass
    0x00,                ///< bDeviceProtocol
```

```

64,                                     ///< bMaxPacketSize
LOWBYTE(VENDOR_ID),                    ///< idVendor
HIGHBYTE(VENDOR_ID),                   ///< idVendor
LOWBYTE(PRODUCT_ID),                   ///< idProduct
HIGHBYTE(PRODUCT_ID),                  ///< idProduct
0x00,                                  ///< bcdDevice
0x02,                                  ///<
0x00,                                  ///< Index of manufacturer string
0x00,                                  ///< Index of product string
0x00,                                  ///< Index of serial number string
1                                       ///< bNumConfigurations
}; ///< Device Descriptor

```

Configuration descriptor

```

__IO static uint8_t CfgDesc[USB_CONFIG_DESC_SIZ] =
{
    0x09,
    ///< bLength: Configuration Descriptor size
    EVD_CONFIGURATION,          ///< bDescriptorType: Configuration
    USB_CONFIG_DESC_SIZ,        ///< wTotalLength: Bytes returned
    0x00,
    0x01,                        ///< bNumInterfaces: 1 interface
    0x01,                        ///< bConfigurationValue: Configuration value
    0x00,                        ///< iConfiguration: Index of string descriptor describing t
he configuration
    0xE0,                        ///< bmAttributes: bus powered and Support Remote Wake-up
    0x32,                        ///< MaxPower 100 mA: used for detecting Vbus

    //=== Deskriptor Joystick/Mys interfacu =====//
    /* Size 09 */
    0x09,                        ///< bLength: Interface Descriptor size
    EVD_INTERFACE,              ///< bDescriptorType: Interface descriptor type
    0x00,                        ///< bInterfaceNumber: Number of Interface
    0x00,                        ///< bAlternateSetting: Alternate setting
    0x01,                        ///< bNumEndpoints
    0x03,                        ///< bInterfaceClass: HID
    0x01,                        ///< bInterfaceSubClass
    0x02,                        ///< nInterfaceProtocol
    0,                            ///< iInterface: Index of string descriptor

    //=== Descriptor of Joystick Mouse HID =====//
    /* 18 */
    0x09,                        ///< bLength: HID Descriptor size
    0x21,                        ///< bDescriptorType
    0x11,                        ///< bcdHID: HID Class Spec release number
    0x01,                        ///<
    0x00,                        ///< bCountryCode: Hardware target country
    0x01,                        ///< bNumDescriptors: HID descs
    0x22,                        ///< bDescriptorType
    74,                          ///< wItemLength: Total length of Report descriptor
    0x00,

    //=== Descriptor of Mouse endpoint =====//
    /* 27 */
    0x07,                        ///< bLength: Endpoint Descriptor size
    EVD_ENDPOINT,               ///< bDescriptorType:

    0x81,                        ///< bEndpointAddress: Endpoint Address
    0x03,                        ///< bmAttributes: Interrupt endpoint

```

```

4,                ///< wMaxPacketSize: 4 Byte max
0x00,            ///<
0x0A,            ///< bInterval: Polling Interval
/* 34 */ Celková velikost
};

```

Mouse Report Descriptor

```

__IO static uint8_t MouseReportDesc[HID_MOUSE_REPORT_DESC_SIZE] =
{
    0x05, 0x01,    ///< USAGE_PAGE (Generic Desktop)
    0x09, 0x02,    ///< USAGE
    0xA1, 0x01,    ///< COLLECTION (Application)
    0x09, 0x01,    ///< USAGE (Pointer)

    0xA1, 0x00,    ///< COLLECTION (Physical)
    0x05, 0x09,    ///< USAGE_PAGE (Button)
    0x19, 0x01,    ///< USAGE_MINIMUM (Button 1)
    0x29, 0x03,    ///< USAGE_MAXIMUM (Button 3)

    0x15, 0x00,    ///< LOGICAL_MINIMUM (0)
    0x25, 0x01,    ///< LOGICAL_MAXIMUM (1)
    0x95, 0x03,    ///< REPORT_COUNT (3)
    0x75, 0x01,    ///< REPORT_SIZE (1)

    0x81, 0x02,    ///< INPUT (Data,Var,Abs)
    0x95, 0x01,    ///< REPORT_COUNT (1)
    0x75, 0x05,    ///< REPORT_SIZE (5)
    0x81, 0x01,    ///< INPUT (Cnst,Var,Abs)

    0x05, 0x01,    ///< USAGE_PAGE (Generic Desktop)
    0x09, 0x30,    ///< USAGE (X)
    0x09, 0x31,    ///< USAGE (Y)
    0x09, 0x38,    ///< USAGE

    0x15, 0x81,    ///< LOGICAL_MINIMUM (-127)
    0x25, 0x7F,    ///< LOGICAL_MAXIMUM (127)
    0x75, 0x08,    ///< REPORT_SIZE (8)
    0x95, 0x03,    ///< REPORT_COUNT (3)

    0x81, 0x06,    ///< INPUT (Data,Var,Rel)
    0xC0, 0x09,    ///< END_COLLECTION
    0x3c, 0x05,    ///<
    0xff, 0x09,    ///<

    0x01, 0x15,    ///<
    0x00, 0x25,    ///<
    0x01, 0x75,    ///<
    0x01, 0x95,    ///<

    0x02, 0xb1,    ///<
    0x22, 0x75,    ///<
    0x06, 0x95,    ///<
    0x01, 0xb1,    ///<

    0x01, 0xc0    ///<
};
///< Report deskriptor - Nastaveni mysi/joysticku

```