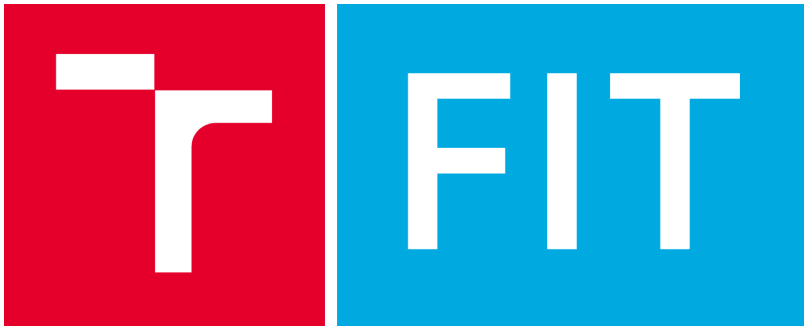
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



Mikroprocesorové a vestavěné systémy
Dokumentace projektu

Demonstrace využití rozhraní USB

Vedoucí projektu: Ing. Václav Šimek, simekv@fit.vutbr.cz

28. prosince 2018 Dominik Skála, xskala11@stud.fit.vutbr.cz

Obsah

1 Zadání

1.1 Demonstrace využití rozhraní USB

S využitím nezbytných prostředků jazyka C, knihoven dostupných na webu společnosti Freescale a vývojových nástrojů vytvořte jednoduchou demonstrační aplikaci pro ARM na FITkitu3, která bude vhodným způsobem ilustrovat možnosti využití rozhraní USB. Vytvořte prezentaci v ppt+pdf s názornými ukázkami tvorby aplikace tak, aby byla srozumitelná pro studenty BP a případně použitelná pro výukové účely.

2 Popis zadání

2.1 Demonstrace využití rozhraní USB

Pro demonstraci USB rozhraní jsem se rozhodl demonstrovat chování HID zařízení, které se chová jako standardní USB HID myš. Komunikace probíhá přes USB rozhraní typu B, které je připojeno k FITKITu a k zařízení, na kterém má být aplikace demonstrována.

3 Popis ovládání

3.1 Ovládání

Pro spuštění aplikace je nutné připojit FITKIT k zařízení ze kterého bude ovládáno přes rozhraní USB typu B. Aplikace využívá tlačítka **SW2**, **SW3**, **SW4**, **SW5** a **SW6**.

Tlačítka **SW2** - **SW5** jsou používána jako směrová tlačítka. Určují tedy směr pohybu myši. Orientace a směr pohybu je dána pozicí tlačítek.

Popis jednotlivých směrových tlačítek:

SW5 (btn4 ve zdrojovém kódu)	pohyb kurzorem nahoru
SW3 (btn2 ve zdrojovém kódu)	pohyb kurzorem dolů
SW4 (btn3 ve zdrojovém kódu)	pohyb kurzorem doleva
SW2 (btn1 ve zdrojovém kódu)	pohyb kurzorem doprava

Tlačítko **SW6** (btn5 ve zdrojovém kodu) je využito jako tlačítko kontrolní, a také jako tlačítko provádějící stisk "levého tlačítka myši".

Pokud dojde ke stisknutí tlačítka **SW6 5x** v řadě za sebou, aktivuje se režim nastavení (3.2.1).

3.2 Výstup

Pro informace o využití USB zařízení a pro konfiguraci (3.2.1) zařízení je využit výstup přes seriový port, který lze na COM portu v cílovém zařízení odchytávat a sledovat. Po spuštění aplikace dojde k vytištění základních informací o projektu a nastavení myši.

3.2.1 Nastavení

Po stisknutí tlačítka **SW6 5x** v řadě za sebou dojde k sepnutí režimu nastavení. V tomto režimu nelze ovládat myš, pro správnou konfiguraci je nutné připojit se ke COM portu na kterém je připojen FITKIT. Po spuštění režimu nastavení se zobrazí úvodní inforamce o nastavení myši, tedy jaké tlačítko je namapováno na **SW6** a také rychlost pohybu myši. Nejnižší hodnota je 0, nejvyšší je 10. Ve výchozím stavu je 5. Nastavení má 3 podsekce. Výpis nastavení USB zařízení, nastavení DPI zařízení a opuštění režimu nastavení. Po opuštění režimu dojde k nastavení parametrů a zařízení se opět chová jako USB HID myš.

4 Řešení projektu

4.1 Jazyk

Pro implementaci byl využit jazyk C včetně základních knihoven. Pro práci s projektem byly od RTOS převzaty knihovny **mqx**, **usb_descriptor** a **usb_hid** pro práci s USB HID rozhraním.

4.2 Vývojové prostředí

Pro implementaci byl využit verzovací nástroj git, zejména tedy jeho populární provozovatel Github s využitím studentské licence.

Pro vývoj v jazyce C byl využit nástroj Kinetis od společnosti NXP Semiconductors s využitím studentské licence.

4.3 Popis řešení

Hlavní funkcí, která obsluhuje rutinu ovládání myši je funkce: `mouseRun()`, která nainicializuje jednotlivá tlačítka FITKITu, vytiskne uvítací zprávu a následně čeká na stisk tlačítka. Inicializace tlačítek je prováděna funkcí `initButton()`, která přijímá vstupní strukturu **LWGPIO_STRUCT**, kterou nastaví jako tlačítko 1 - 5, zadané druhým parametrem. Popis jednotlivých tlačítek je v sekci 3.1. Po úspěšné inicializaci tlačítek naplní strukturu **Mouse_settings** úvodními daty, která lze právě měnit přes režim nastavení. Tato struktura se aktualizuje při každém stisku tlačítka, buď se do ní ukládají směrová data, tedy kam se má myš pohnout a nebo jsou do ní ukládány eventy stisku tlačítka. Po úspěšném uložení dat do struktury je zavolána funkce: `sendMouseCommand()`. Tato funkce správně nakonfiguruje položku **rpt_buf** struktury **g_mouse** (která je typu: MOUSE_GLOBAL_VARIABLE_STRUCT). **rpt_buf** je pole které má 4 prvky. První prvek obsahuje informace o tom, jaké tlačítko bylo stisknuto. Následující dva prvky určují směr na ose x a y ve kterém má být kurzor posunut. Poslední prvek pole není využit. Následně dojde k odeslání dat přes `USB_Class_HID_Send_Data()` funkci, která přijímá právě výše zmíněnou strukturu **rpt_buf** a aplikační ovladač pro myš. Po odeslání dat se reinicializuje struktura (smaže se příznak pohybu a stisku tlačítka). Kontrola zda nebylo stisknuto nějaké tlačítko se provádí periodicky, každých 5 ms. Toto řešení jsem zvolil protože se jedná o jednodušší řešení, kdy mohu naráz vyvolat úpravu nastavení směru pohybu více než jedním tlačítkem myši. Toto však komplikuje stav, kdy je nutné zaručit, že bylo tlačítko stisknuto pouze jednou. Variantou byl vlastní ovladač pro zjišťování stavu tlačítka a nebo použití semaforů a přerušení, které je však na MQX výrazně složitější a logika s více tlačítky by dala více práce. U pohybu myši toto řešeno nijak není, pro pohyb se pouze kontroluje zda je tlačítko aktivně drženo. U režimu nastavení je toto řešeno následovně:

Pro kontrolu jsou tedy vždy volány dvě funkce: `updateButtonState()`, která vždy zaktualizuje aktuální stav tlačítka na **BUTTON_PRESSED** nebo **BUTTON_RELEASED**. Po této funkci je doporučeno volat funkci: `buttonPressedAndReleased()`, která po uvolnění tlačítka vrací hodnotu **TRUE** pokud bylo tlačítko stisknuté a následně uvolněno.

4.3.1 Struktury

Aplikace využívá vlastní struktury **Mouse_settings**, která obsahuje data která ze kterých se vytváří jednotlivé požadavky pro USB HID rozhraní.

Tato struktura je definována následovně:

```
typedef struct mouse_settings_struct{
    int MOUSE_DIR;
    int MOUSE_SECOND_DIR;
    int MOUSE_SPEED;
    int SECOND_PRESS;
    int BUTTON;
    int BUTTON_PRESS;
} Mouse_settings;
```

Struktura obsahuje:

int MOUSE_DIR primární směr pohybu kurzoru

int MOUSE_SECOND_DIR případný sekundární směr (pohyb do 8 směrů)

int MOUSE_SPEED rychlost pohybu kurzoru (slouží pro úpravu DPI)

int SECOND_PRESS proměnná určující zda bylo stisknuto druhé tlačítko pro pohyb v druhém směru

int BUTTON pro jaké tlačítko je tlačítko **SW6** nakonfigurováno

int BUTTON_PRESS proměnná určující zda bylo tlačítko stisknuto

4.3.2 Rozšíření

Aplikace rozšiřuje původní příklady MQX, tedy zejména je převzata inicializace tlačítek a inicializace USB_HID rozhraní z ukázkových příkladů od MQX RTOS.

5 Závěrečná shrnutí

Samotná aplikace vcelku dobře zvládá změnu rychlosti pohybu kurzoru i samotný pohyb. Známým problémem však je stisk tlačítek určených pro opačné směry. Kurzor se poté pohybuje jinam než by reálně měl (ve skutečnosti by měl zůstat na jednom místě, když jsou dvě protisměrná tlačítka stisknuta naráz). Dozajista by šlo zlepšit implementaci stisku levého tlačítka. Variantou první je používat přerušení a semaforu. Druhou variantou je, že se po aktivním držení zavolá metoda pouze jednou a vícekrát ne dokud by tlačítko nebylo opět povoleno. Neimplementovanou součástí je možnost změny mapování tlačítek. Aktuálně je aplikace nachystána tak, aby se dalo ve zdrojovém kódu změnit mapování tlačítka **SW6** na jiné tlačítko, v režimu nastavení toto ale není možné měnit.

6 Materiály

Aplikace byla vyvinuta na základě zdrojových kódů aplikace **MQX hid_mouse_dev_fitkit** ukázkového příkladu. Tento příklad byl získán jako součást zadání ze strany Ing. Šimka.