VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Branislav Hatala

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

USB ROZHRANÍ IMPLEMENTOVANÉ V MIKROKONTROLÉRU
USB IN MICROCONTROLER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Branislav Hatala**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Zdeněk Bradáč, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Branislav Hatala ***ID:*** 195308
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

USB rozhraní implementované v mikrokontroléru

POKYNY PRO VYPRACOVÁNÍ:

1. Proveďte literární rešerši implementace USB rozhraní na malých výpočetních platformách.
2. Navrhněte koncepci implementace USB rozhraní do malého mikrokontroléru. Prezentujte blokovou strukturu HW a SW.
3. Zvolte vhodnou platformu, otestujte a oživte nezbytný HW.
4. Zvolte vhodný komunikační model a definujte profil komunikace. Vytvořte programové vybavení a otestujte jej.
5. Demonstrujte přenos dat mezi vaším funkčním vzorkem a jeho okolím přes USB. Zhodnoťte vaši realizaci.

DOPORUČENÁ LITERATURA:

Pavel Herout: Učebnice jazyka C, KOPP, 2004, IV. přepracované vydání, ISBN 80-7232-220-6

Dle pokynů vedoucího práce.

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 20.5.2019

Vedoucí práce: doc. Ing. Zdeněk Bradáč, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Táto práca sa zaoberá implementáciou USB rozhrania v miktrokontroléroch. Ako príklad vytvára ukážkové kompozitné zariadenie pozostávajúce z rozhrania USB klávesnice a virtuálneho konzolového rozhrania. Jedná sa o zariadenie určené na zber dát z meracích prístrojov, posielanie dát do PC je riešené emulovanou USB klávesnicou. Zatial čo virtuálna konzola slúži na výstupnú komunikáciu do prístrojov. Jedná sa len o ukážku USB komunikácie, zber dát je simulovaný posielaním predom pripravených reťazcov.

KĽÚČOVÉ SLOVÁ

USB, USB HID, mikrokontrolér, emulátor klávesnice, datové zberné zariadenie

ABSTRACT

This thesis deals with implementation of USB interface in microcontroller. As demonstration example it creates, a composite device that implements USB keyboard interface along with virtual console interface. The purpose of this device is to serve for data collection from measurement instruments. This device receives commands via virtual console and forwards data to a connected PC via emulated keyboard interface. This provides a practical example on the USB side, while measurement is simulated by set of predetermined strings.

KEYWORDS

USB, USB HID, microcontroller, keyboard emulator, data collection device

HATALA, Branislav. *USB rozhraní implementované v mikrokontroléru*. Brno, 2019, 32 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedúci práce: doc. Ing. Zdeněk Bradáč„ Ph.D.

VYHLÁSENIE

Vyhlasujem, že som svoju bakalársku prácu na tému „USB rozhraní implementované v mikrokontroléru“ vypracoval samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád bych poděkoval vedoucímu bakalárskej práce panu doc. Ing. Zdeněku Bradáčovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výzkum popsaný v tejto bakalárskej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno ....

podpis autora

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov

Úvod

Táto práca sa zaoberá Implementáciou USB. rozhrania v mikro kontroléroch. Ako ukážku implementácie používa zariadenie na zber dát z meracích prístrojov. Keďže rozsah tejto práce rieši len implementáciu USB rozrahnia, funkcionalita zberu dát tohoto ukážkového zariadenia je len simulovaná posielaním predpripravených číselných hodnôt.

Táto práca obsahuje porovnanie rôznych prístupov, ako implementovať takéto zariadenie. Tieto spôsoby sú:

- emulátor USB klávesnice,
- zariadenie HID triedy,
- zariadenie TMC triedy.

V tejto práci sa do hĺbky rieši implementácia emulátoru klávesnice. Toto rozhodnutie je odôvodnené faktom, že autor tejto práce sa podiela vo výkone svojho zamestnania na vývoji konfiguračných nástrojov, ktoré sú v tejto práci použité. Emulátor USB klávesnice využije práve tie nástroje a prostriedky, na ktorých sa najviac autor podielal a týmto ich prakticky otestuje.

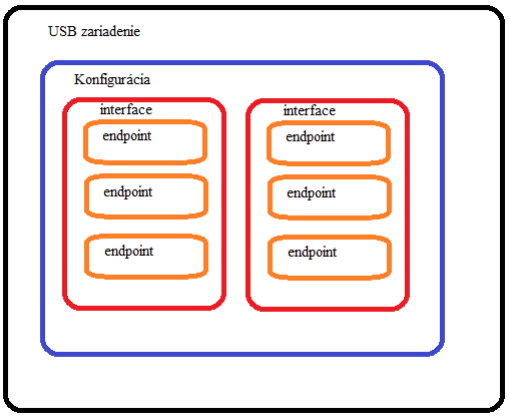
Programové vybavenie vytvorené v tejto práci slúži ako ukážka, ktorú je možné rozšíriť o kód na obsluhu jednotlivých pripojených meracích prístrojov, periférií a prípravkov. V kóde je určené presné miesto, kde sa má rozširovací kód dopísať.

Ukážkové zariadenie tiež rieši zprostredkovanie výstupných dát do prístrojov(SCPI príkazy) pre uživatelský kód, ten má k nim prístup v mieste určenom pre jeho rozšírenie. Táto úloha je riešená pomocou pridanej VCOM funkcionality.

1 Teoretická část studentské práce

USB zariadenie využíva na komunikáciu z okolím prostriedok zvaný endpoint. Túto komunikáciu riadi a iniciuje Host (Počítač na ktorý je zariadenie pripojené).

Endpoint je unikátne odlíšitelná časť USB zariadenia, ktorá slúži ako zakončenie komunikačného prúdu medzi počítačom a USB zariadením. Každé USB logické zariadenie pozostáva zo série niekoľkých nezávislých endpointov. Každé logické USB zariadenie má svoju unikátnu adresu pridelenú systémom pri pripojení. Každý endpoint v zariadení má unikátnu adresu pridelenú pri jeho konštrukcií. Endpoint je determinovaný svojou adresou, číslom a smerom toku dát. Každý endpoint slúži na jednosmernú komunikáciu: vstup (smer zo zariadenia), alebo výstup (smer do zariadenia).[1]



Obr. 1.1: Logická štruktúra kompozitného USB zariadenia.

Každé USB zariadenie môže obsahovať niekoľko konfigurácií (Množín nastavení), pričom len jedna môže byť aktívna. Každá konfigurácia môže obsahovať niekoľko rozhraní. Každé rozhranie obsahuje niekoľko aktívnych endpointov. Obrázok 1.1 ilustruje hierarchiu, v akej je zariadenie štruktorované.

Každé USB zariadenie má dva endpointy 0, jeden na výstup a druhý na vstup. Tento endpoint je použitý na spravovanie štandardných funkcí USB zariadenia (enumerácia, konfigurácia, ovládanie nastavení) a nemá vlastný descriptor[1] (prvok popisu).

[1]Datová štruktúra nesúca údaje potrebné pre funkciu prvku hierarchie USB zariadenia

Každý descriptor nesie údaje o svojom type a dĺžke. Základná USB špecifikácia stanovuje nasledovné typy deskriptorov nesúce tieto údaje:

- Device,
  - BCD Kód USB verzie,
  - Kód triedy, podtriedy a protokolu[2],
  - Maximálnu velkosť packetu pre endpoint 0,
  - ID Vendora (pridelené USB organizáciou) a ID produktu,
  - BCD kód verzie zariadenia,
  - indexy v string descriptoru na:
    - meno výrobca,
    - meno produktu,
    - sériové číslo zariadenia.
  - počet možných konfigurácií.
- Configuration,
  - veľkosť celej konfiguračnej hierarchie,
  - počet rozhraní v konfigurácií,
  - číslo konfigurácie,
  - index v string descriptore na meno konfigurácie,
  - príznaky atribútov (samonapájanie, vzdialené prebudenie...),
  - maximálny prúd, ktorý zariadenie môže odoberať.
- Interface,
  - číslo rozhrania,
  - hodnota alternatívneho nastavenia,
  - počet endpointov,
  - kód triedy, podtriedy a protokolu[3],
  - index v string descriptoru na meno interfacu.
- Endpoint,
  - adresa (0-15) a smer,
  - typ prenosu a prípadné atribúty,
  - maximálna velkosť packetu,
  - interval vyvolávania.

[2]Od príslušnosti v týchto kategóriach závisí to, ako sa zariadenie bude správať a akým štandardom podlieha

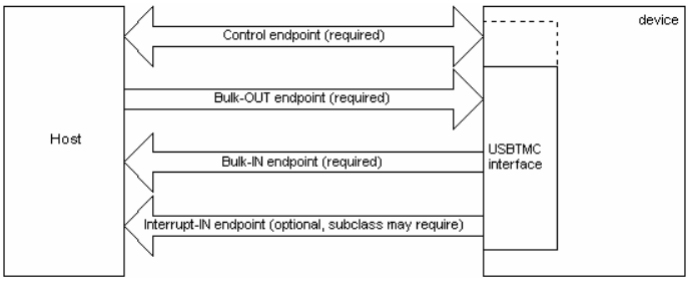
[3]Tieto údaje je možné definovať na úrovni rozhrania, táto metóda je preferovaná.

USB trieda, podtrieda a protokol vymedzujú účel a správanie zariadenia. Fungujú tak že roźširujú štandard tak, že definujú prídavnú vrstvu nad základnou USB špecifikáciou. Jednotlivé triedy majú svoje vlastné špecifikácie vrátane voľne dostupných dokumentov.

1.1 Test and Measurement class

USB TMC trieda rovnako ako ostatné USB triedy zariadení je špecifikácia prídavnej aplikačnej vrstvy nad USB štandardom definovanom v USB špecifikácií.

USB TMC trieda je vhodná pre inštrumentačné zariadenia, ktoré nevyžadujú garantované časovanie. Tieto zariadenia pozostávajú typicky z komponentov ako ADC, DAC, senzory a prevodníky. Môže sa jednať o samostatné zariadenia, alebo o karty v počítačoch.[4]



Obr. 1.2: USB TMC komunikačný model[3]

Táto trieda vyžaduje 2 endpointy[4] prenosu Bulk[5](Hromadný prenos) jeden pre vstup a druhý pre výstup. Niektoré podtriedy vyžadujú interrupt[6](Prerušujúci prenos) endpoint na vstup[7]. Dalšie prídavné endpointy sú prípadne podporované pri niektorých podtriedach a protokoloch.

Obsluha zariadenia je riešená pomocou štandardizovaných správ posielaných bulk endpoitmy, prípadne notifikácií cez interrupt endpoint. Použitie zariadení USB TMC vyžaduje špecializovaný software napr. NI visa, Labview, alebo test stand. Driver určený pre dané zariadenie je tiež vyžadovaný.

[4]Tieto 2 endpointy majú rovnaké číslo a preto sa javia ako jeden vstupno-výstupný

[5]typ USB prenosu dát určený pre posielanie väčších objemov dát z korekciou dát a bez časovania

[6]typ USB prenosu určený na rýchle periodické posielanie dát z garantovaným intervalom a korekciou dát

[7]v USB terminológií sa vstup/výstup označuje z perspektívy Host zariadenia

1.2 USB Human Interface Class

USB HID zariadenie sa podľa protokolu delí na:

- žiaden,
- klávesnica,
- myš.

Trieda HID je určená na aplikácie, kde je nutné periodicky posielať malý objem dát z garantovanou periódou a korekciou chýb. Pomocou tejto triedy nie sú realizované len myš, klávesnica, ovládacie prvky, ale aj niektoré senzory a medicínske zariadenia.

USB HID špecifikácia definuje dalšie dva descriptory: report descriptor (popisná štruktúra správy) a physical descriptor (fyzická popisná štruktúra). Zariadenie smie použiť na komunikáciu len interrupt endpoint. Posiela ním dáta s predom definovanou štruktúrou zvanou report (správa). Jej presná štruktúra a vlastnosti jednotlivých dát ktoré obsahuje, sú definované v report descriptore. Physical descriptor je nepovinný, pre funckiu USB rozhrania nie je podstatný.

Fyzická popisná štruktúra nesie informácie o tom ktorou časťou ľudského tela je ovládaný špecifický ovládací prvok alebo skupina ovládacích prvkov. Napríklad, môže indikovať že, palec pravej ruky je určený na používanie tlačidla 5. Aplikácia môže využiť túto informáciu na priradenie funkcie ovládacím prvkom zariadenia.[2]

Report descriptor1.3 je séria pozostávajúca z párov (kód,hodnota), kde kód určuje význam svojej hodnoty. Výnimkou je ënd collection"(uzvierací prvok súboru prvkov) kód ktorý nie je nasledovaný hodnotou. Jeden report descriptor definuje štruktúru vstupných aj výstupných dát a zároveň aj nastavenia (feature), ktorými zariadenie komunikuje.

Výhodou USB HID zariadení je jednoduchá implementácia, ale pre ich používanie je nutný host driver a software. Tento software je podstatne jednoduhšie implementovať na rozdiel od USB TMC .

1.3 Emulátor USB klávesnice

Existuje rada zariadení, ktoré v sebe implementujú rozhranie štandardnej USB klávesnice a ich výstupom sú virtuálne stlačenia klávies. Typickým príkladom sú skanery čiarových kódov. Veľkou výhodou týchto zariadení je, že využívajú driver obsiahnutý v každom desktop systéme a nevyžadujú žiadny prídavný software.Taktéto zariadenie môže byť použité na sadzbu výstupných hodnôt priamo do tabuľkového kalkulátoru, alebo vyžadovaných textových polí.

Výstupom tejto semestrálnej práce je práve takéto zariadenie.

Item		Value (Hex)
Usage Page (Generic Desktop),		05 01
Usage (Mouse),		09 02
Collection (Application),		A1 01
Usage (Pointer),		09 01
Collection (Physical),		A1 00
Usage Page (Buttons),		05 09
Usage Minimum (01),		19 01
Usage Maximun (03),		29 03
Logical Minimum (0),		15 00
Logical Maximum (1),		25 01
Report Count (3),		95 03
Report Size (1),		75 01
Input (Data, Variable, Absolute),	;3 button bits	81 02
Report Count (1),		95 01
Report Size (5),		75 05
Input (Constant),	;5 bit padding	81 01
Usage Page (Generic Desktop),		05 01
Usage (X),		09 30
Usage (Y),		09 31
Logical Minimum (-127),		15 81
Logical Maximum (127),		25 7F
Report Size (8),		75 08
Report Count (2),		95 02
Input (Data, Variable, Relative),	;2 position bytes (X & Y)	81 06
End Collection,		C0
End Collection		C0

Obr. 1.3: Report Descriptor štandardnej počítačovej myši[2].

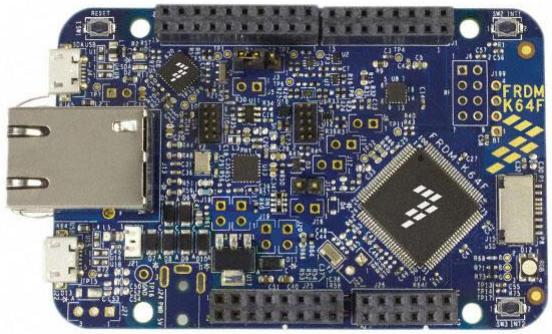
2 Výsledky studentské práce

Úloha implementácie USB rozhrania môže byť pojatá na rôznych úrovniach. Vytvoriť zariadenie, ktoré prejde enumeráciou a plní svoju funkciu je podstatne jednoduhšie, ako urobiť zariadenie, ktoré je schopné vyhovieť pokročilejším štandardným USB požiadavkom a fungovať robustne v bežných situáciach[1]. V prípade použitia prostriedkov ako v tomto projekte by tento krok vyžadoval zásah do kódu v USB stack komponente. Vôbec najťažšou úlohou v implementovaní USB rozhrania je vytvorenie zariadenia ktoré vyhovuje USB štandardu a môže byť uznané USB organizáciou.

Cieľom v tejto práci je vytvoriť zariadenie ktoré je schopné základnej USB funkcionalite, a jeho odpovede na štandardné USB požiadavky sú situačné tak, aby pri priamom pripojení fungovalo spolahlivo.

2.1 Doska Plošného Spoja

Táto práca je realizovaná na skúšobnej doske FRDM-K64F Obr.2.1., ktorá je komerčne dostupná od firmy NXP, MCU použité v tejto doske je MK64FN1M0VLL12. Pre funkciu USB modulu tento MCU nevyžaduje žiadne externé obvody.



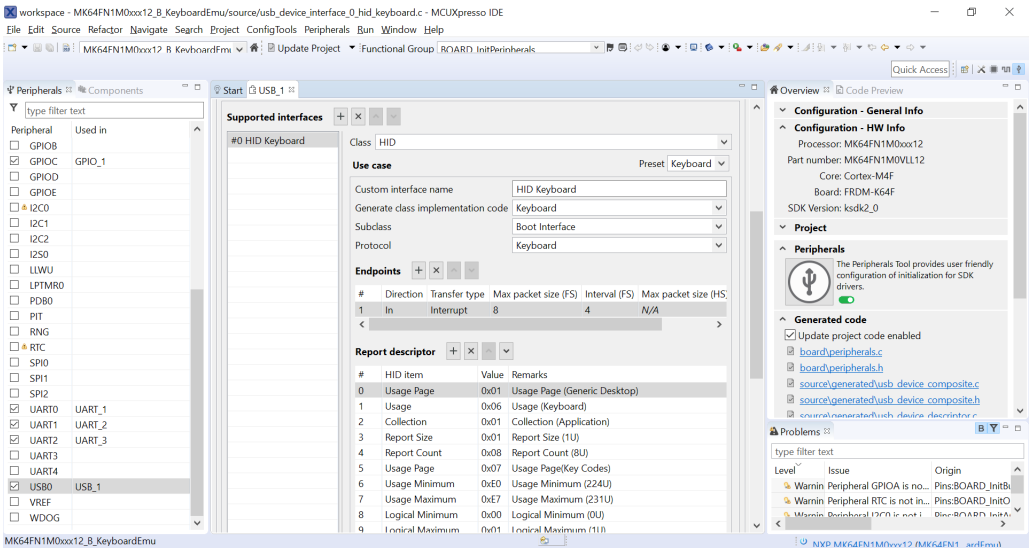
Obr. 2.1: skúšobná doska FRDM-K64F

2.2 Implementácia USB periférie v kóde

Pri práci s vývojovou doskou FRDM-K64F bolo použité natívne vývojové prostredie od NXP, MCUXpresso. Toto vývojové prostredie obsahuje nástroje na konfiguráciu Obr.2.2 periférnych zariadení v MCU, jeho vstupom sú nastavenia uživatelom cez grafické rozhranie a výstupom je vygenerované ukážkové súbory .c/.h. Tieto súbory

[1] Takéto situácie môžu zahrňovať suspendovanie zariadenia alebo funkciu pripojením cez rozbočovač

sú v stave obsahujúci základnú funkcionalitu a ďalším krokom je ich úprava na uživatelom vyžadovanú funkcionalitu.



Obr. 2.2: konfiguračný nástroj v IDE MCUXpresso

Tento konfiguračný nástroj pre USB periférium generuje zdrojové súbory funkčné všeobecne pre USB a jeden konkrétny pre protokol, šablóna na základe ktorej generuje sa volí pomocou parametru „Generate class implementation code“.

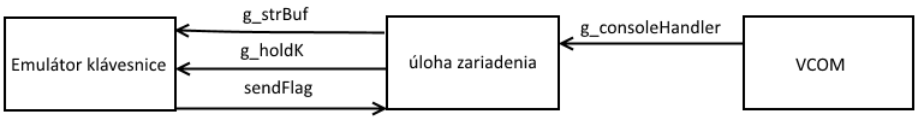
Výsledkom tejto práce je kompozitné zariadenie, implementuje viac než jednu triedu. Takéto zariadenie pozostáva z niekolkých rozhraní, kde jednotlivé zariadenia sú implementované vo svojom rozhraní. Za týmto účelom je nutné definovať pre každé zariadenie triedu, podtriedu a protokol na úrovni rozhrania.

Konfiguračné nástroje integrované pre MCUXpresso sa správajú ku všetkým zariadeniam bez ohľadu na to či implementuje viac rozhraní ako ku kompozitnému zariadeniu. Tetno prístup zjednodušuje prácu z týmto zariadením a aj implementáciu nástroja samotného. Vďaka tomuto prístupu stačí pre implementáciu kompozitného zariadenia len pridať ďalšie rozhrania a upraviť ich funkcionalitu na požadovanú.

2.2.1 Štruktúra projektu

Ukážkové zariadenie sa skladá z 3 hlavných častí, tieto časti navzájom komunikujú pomocou 4 globálnych premenných. Zloženie a spôsob interakcie jednotlivých blokov je ilustrovaný na Obr.2.8.

Emulátor klávesnice - posiela do PC výstup zo zariadenia vo forme kódov stlačenia kláves. Na vstup používa premennú g_strBuf, táto premenná obsahuje jeden



Obr. 2.3: Zloženie zariadenia

reťazec znakov, ktorý bude poslaný a následne vynulovaný. Po vynulovaní tohoto reťazca je nulovaná aj synchronizačná premenná sendFlag. Posielanie jednotlivých obsahu g_strBuf cez USB prebieha len ak hodnota synchronizačnej premennej g_holdk je nulová, inak je posielanie pozastavené. Synchronizačná premenná sendFlag slúži na to, aby sa dokončilo volanie callback funkcie než začne dalšie.

VCOM - slúži na získanie konfiguračného reťazca pre jednotlivé zariadenia. Tento reťazec sa jednorázovo zadáva pri konfigurácií zariadenia na danú laboratórnu úlohu. Tento reťazec pozostáva z čiastkových reťazcov, ktoré sú určené pre jednotlivé merania na jednotlivých prístrojoch. Čiastkové reťazce sú oddelené znakom '$' následovaným jednoznakovou adresou prístroja. Čiastkový reťazec môže obsahovať akékoľvek znaky z výnimkou '$' a 0x00. Výstupom tohoto bloku je aktualizovanie globálnej premennej g_consoleHandler, tá obsahuje samotný konfiguračný reťazec a údaje o jeho dĺžke a momentálnej polohe kurzoru slúžiaceho na implementáciu navigácie v reťazci z konzolového vstupu.

Úloha zariadenia - obsluhuje prerušenie stlačenia tlačidla. Jeho hlavná časť je funkcia ktorá prejde postupne konfiguračný reťazec, naplní statickú premennú finalString čiastkovým reťazcom a zavolá callback funkciu pre jeho adresu. Tento proces sa opakuje pre každý jeden čiastkový reťazec obsiahnutý v konfiguračnom reťazci. Súčasťou tohto procesu je aj práca zo synchronizačnými globálnimy premennými.

2.2.2 Implementácia rozhrania klávesnice

Šablónu pre protokol klávesnice tvoril sám autor. Tieto šablóny sa vonkajšou funkcionalitou odvíjajú od svojich predchodcov, SDK ukážokových projektov. Narozdiel od nich ponúkajú priamu generáciu kódu ktorá má štruktúru takú, ako je v praxi zaužívané. Zatiaľ čo ukážkové projekty nie sú vhodné na priamu implementáciu.

Vygenerovaný kód pre USB klávesnicu je schopný enumerácie a posiela v cykloch striedavo kód stlačenia dvoch kláves, PAGEUP a PAGEDOWN. Tento kód bol následne prepísaný na prevádzanie dát určených na poslanie na klávesové kódy a ich postupné posielanie v packetoch po jednom. Implementácia USB rozhrania sa skladá z nasledovných častí:

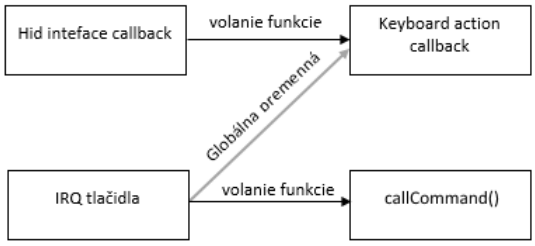
- USB driver,

- USB stack[2],
- generovaný kód,
- časť generovaného kódu prepísaná uživatelom.

Obr. 2.4 Popisuje volanie hlavných častí programu, kde funkcia HID interface Callback Obr. A.1 je volaná cez USB driver. Slúži ako rozbočník na volanie iných funkcií, ktoré obsluhujú iné udalosti v generovanom súbore a ktoré nie sú všetky automaticky generované (prípadne sú autorom dopísané).

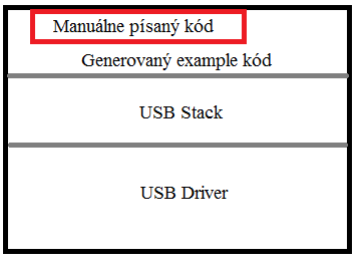
Keyboard action callback Obr. A.2 vykonáva samotnú úlohu zariadenia, jedno volanie tejto funkcie pošle jeden znak reťazca zapísaného v globálnej premennej g_StrBuf v prípade, že všetky znaky tohoto reťazca boli poslané, je vyprázdnený.

Do reťazca g_StrBuf prepisuje hodnoty určené na poslanie funckia volaná úlohou zariadenia. Sprosredkovanie úlohy zariadenia je volané funkciou callCommand().



Obr. 2.4: schéma vzájomného volania kódu

Obr. 2.5 ilustruje spôsob akým sa tvoria zariadenia pomocou konfiguračných nástrojov, rovnaký postub je použitý pre rozhrania USB klávesnice a VCOM.



Obr. 2.5: Štruktúra implementácie kódu

Úloha emulátoru klávesnice sa vykonáva callback funkciou USB_DeviceHidKeyboardAction(void), jej zprehladnený kód vyzerá nasledovne:

[2]USB stack slúži ako medzivrstva vložená medzi uživatelský kód a USB driver

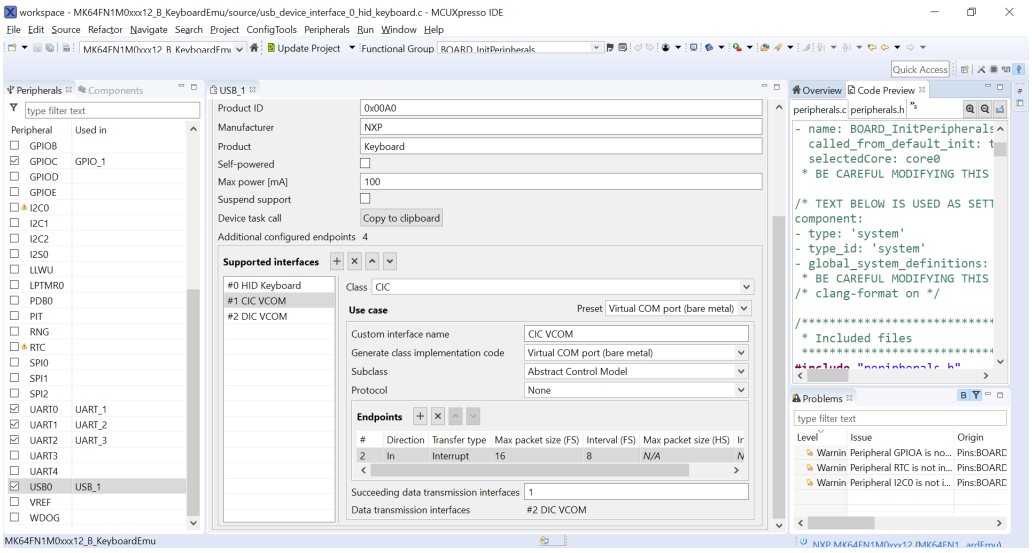
```
static usb_status_t USB_DeviceHidKeyboardAction(void)
{
    static int x = 0U;// pozicia v retazci
    s_UsbDeviceHidKeyboard.buffer[2] = 0x00U;
    if(g_StrBuf[x]=='\0')// posielanie dokoncene.
    {
        for(int i = 0;i<20;i++)
        {
            g_StrBuf[i] = '\0';
        }
        x = 0U;
        sendFlag = 0;// umozni dalsie meranie
    }
    else if(!g_holdk)// synchronizacia
    {
        // naplnenie bufferu na odoslanie znaku
        s_UsbDeviceHidKeyboard.buffer[2] =
            cNumericToKey(g_StrBuf[x]);
        x++;// posuv pozicie v retazci
    }
    // navratova hodnota je vysledok USB prenosu
    return USB_DeviceHidSend(
        s_UsbDeviceComposite->interface0HidKeyboardHandle,
        USB_INTERFACE_0_HID_KEYBOARD_EP_1_INTERRUPT_IN,
        s_UsbDeviceHidKeyboard.buffer,
        USB_INTERFACE_0_HID_KEYBOARD_INPUT_REPORT_LENGTH);
}
```

Výpis 2.1: Callback úlohy klávesnice

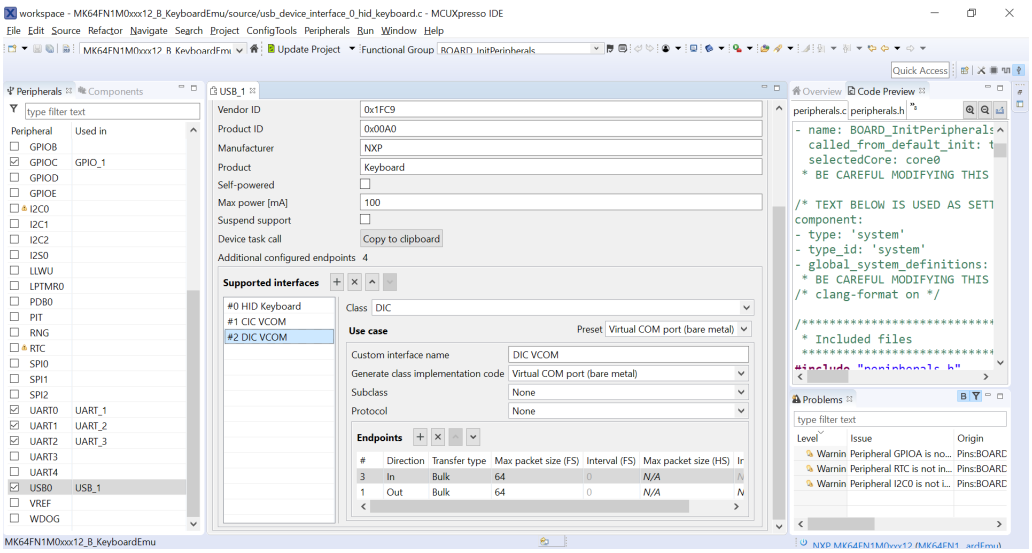
2.2.3 Implementácia VCOM

Rovnakým spôsobom ako bola generovaná a prepísaná implementácia HID klávesnice bol vygenerovaný aj ukážkový projekt pre VCOM. Pre pridanie VCOM funkcionality do projektu je nutné pridať 2 USB rozhrania, CIC a DIC. Nástroje MCUXpresso generujú USB projekty ako kompozitné. To umožňuje pridávať ďalšie USB rozhrania, Pričom najprísnejšie obmedzenie v MCU vychádza z obmedzeného počtu endpointov.

Vygenerovaný kód pre VCOM je schopný enumerácie, pripojenia sa cez konzolový software (napr. PuTTY), príjmať znaky a zároveň ich posielať spať. Takto sa javí



Obr. 2.6: Nastavenia rozhrania CIC



Obr. 2.7: Nastavenia rozhrania DIC

konzolové okno ako funkčné. V tomto stave ale konzolová aplikácia nie je schopná správne reagovať na niektoré klávesy ako del, backspace a šípky, tiež je v režime delete a nie insert.

Pre naše účely je nutné prepísať úlohu zariadenia, konkrétne miesto kódu na prepísanie je v súbore usb_device_interface_1_cic_vcom vo funkcií void USB_DeviceInterface1CicVcomTask(void).

Obsluha konzole spociva v posielani vstupnych znakov na konzolový výstup, toto ich zobrazí. A skladaniu reťazca zloženého príkazu z príchozích znakov, ten

sa finalizuje zaznamenaním znaku noveho riadku. Na výslednom reťazci je nutné vykonávať operácie príslušné špeciálnymi znakmi, alebo ich ignorovanie.

Je úryvok z kódu ktorý spravuje konzolový vstup, funkcie deleteChar() a addChar() formujú reťazec z príchozieho konzolového vstupu:

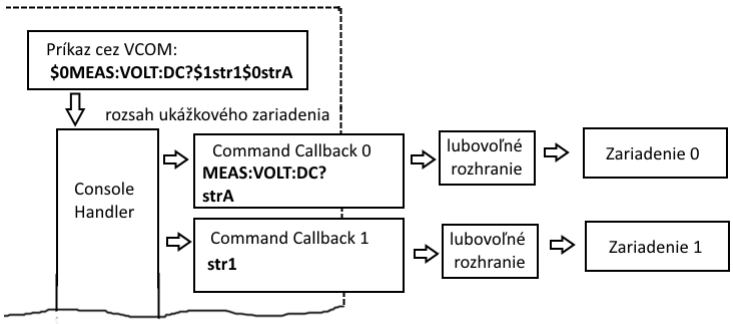
```
void USB_DeviceInterface1CicVcomTask(void)
{
    // inicializacia ukazatelu na retazec zo vstupom
    if(g_consoleHandler.inputStr == NULL)
    {
        g_consoleHandler.inputStr = s_currRecvBuf;
    }
    // navratova hodnota
    usb_status_t error = kStatus_USB_Error;
    /* kontrola pripojenia zariadenia
     a rozhrania konzoloveho vstupu */
    if ((1 == s_UsbDeviceComposite->attach)
        && (1 == s_UsbInterface1CicVcom.startTransactions))
    {
        // prisiel vstup?
        if ((0 != s_recvSize) && (0xFFFFFFFFU != s_recvSize))
        {
            int32_t i;
            // pozice startu výpisu do konzole
            char escapedPos = 0;
            // kopírovanie vstupu na výstup
            for (i = 0; /* i < s_recvSize */; i++)
            {
                // dosiahol sa koniec
                if(s_currRecvBuf[i] == '\0')
                {
                    break;
                }
                // osetrenie specialnych znakov
                i = resolveChar(i);
                // zaistenie preposlania
                s_currSendBuf[s_sendSize++] = s_currRecvBuf[i];
                // nacitanie do prikazu
```

```
34                  if(escapedPos<=i)
35                  {
36                      if(s_currRecvBuf[i]==127) // zmazanie
37                      {
38                          deleteChar();
39                      }
40                      else // pridanie znaku
41                      {
42                          addChar(i);
43                      }
44                  }
45              }
46              s_recvSize = 0;
47          }
48          // odosielanie je povodny generovany kod
49          if (s_sendSize)
50          {
51              uint32_t size = s_sendSize;
52              s_sendSize = 0;
53
54              error = USB_DeviceCdcAcmSend(
55                  s_UsbInterface1CicVcom.cdcAcmHandle,
56                  USB_DIC_VCOM_IN_ENDPOINT, s_currSendBuf,
57                  size);
58
59              if (error != kStatus_USB_Success)
60              {
61                  // posielanie zlyhalo
62              }
63          }
64      }
65  }
```

Výpis 2.2: Callback úlohy virtuálnej konzole

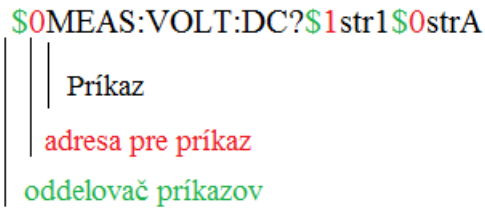
2.2.4 Implementácia úlohy zadania

Obr. 2.8 Ilustruje úlohu zariadenia. Posledný reťazec prijatý cez terminál je uložený v pamäti. Každým stlačením tlačidla je tento príkaz po úsekoch rozdeleným znakom $ a jeho adresou rozdelený na jednotlivé príkazy. Následne je volaný callback na



Obr. 2.8: výstupný komunikačný model

základe adresy jednotlivého príkazu, pritom je tejto callback funkcií sprístupnení jej prislúchajúci príkaz.



Obr. 2.9: Tvar vstupu z konzolového okna

Na opačný smer komunikácie je určená globálna premenná g_StrBuf. vo funkcií commandCallbacks() v súbore console_handler.c na miestach, kde uživatel je mienený dopisovať vlastný kód obsluhujúci jednotlivé zariadenia, sa nachádza volanie funkcie getSData(g_StrBuf). Táto funkcia v ukážke slúži ako príklad posielania reťazcov na výstup emulovanej klávesnice.

Medzi jednotlivými adresami rozlišuje jedna switch štruktúra, kam uživateľ môže pridať vlastné adresy. Pri rozširovaní o ďalšie adresy je nutné okopírovať prácu so synchronizačnými premennými (g_holdK a sendflag) z predpísaných case blokov. Tieto synchronizačné hodnoty slúžia na synchronizáciu vstupno výstupných operácií. To zaručuje postupné vykonávanie jednotlivých príkazov v správnom poradí bez prerušenia.

V súbore console_handler.c má vo funkcií commandCallbacks() prístup k relevantnému jednotlivému reťazcu konfiguračného reťazca cez statickú premennú singleCommand a dĺžku tohoto reťazca má uloženú v statickej premennej singleCommandLength. V prípade potreby pristúpiť vo svojej callback funkcií k príkazu smerovanému na svoju adresu, môže použiť tieto dve statické premenné na prístup k nej.

kód ktorý rozlišuje adresy príkazov a obsahuje miesto pre uživatelský kód. Ten je ohraničený synchronizačnými opatreniami v podobe práce z globálnimy premennými. V rámci synchronizácie je nutné zabrániť klávesnicovému výstupu posielať reťazec z nedokoncenej operácie, a zaistiť že další uživaťelský callback sa volá až po dokončení výstupu dát aktuálneho.

```
1  static void commandCallbacks()
2  {
3      int i = 0;
4      switch (address) {
5          case '0': // callback pre adresu 0
6              if (!sendFlag)
7              {
8                  // zabranenie klavesnicoveho vystupu
9                  g_holdk = 1;
10
11                 // uživatelský kód
12                 getSData(g_StrBuf);
13
14                 // synchronizácia
15                 g_holdk = 0; // data pripravene, cas odosielat
16                 // zablokovanie uzivatelskych callbackov
17                 sendFlag = 1;
18                 /* sendFlag je nulovany v callback
19                  funkcii klavesnice
20                  po tom co je odosielanie hotove */
21             }
22             break;
23         case '1': // callback pre adresu 1
24             if (!sendFlag)
25             {
26                 // zabranenie klavesnicoveho vystupu
27                 g_holdk = 1;
28
29                 // uživatelský kód
30                 getSData(g_StrBuf);
31
32                 // synchronizácia
33                 g_holdk = 0;
```

```
34                      sendFlag = 1;
35                  }
36                  break;
37              default:
38                  break;
39      }
40  }
```

Výpis 2.3: Callback úlohy zariadenia

3 Závěr

Implementácia USB rozhrania v mikrokontroléroch je podstatne zjednodušená konfiguračnými nástrojmi, ktoré generujú kód, ktorý rieši inicializáciu periferií zariadenia a umožňujú pri vývoji sústrediť sa viac na úlohu zariadenia. Tieto nástroje sú flexibilná alternatíva na predošlé vývojovové metódy, ktoré zahŕňali prepis ukážkových projektov. Vďaka týmto prostriedkom si dokáže uživatel generovať kód na inicializáciu a základnú obsluhu aj iných periferných zariadení. Táto základná obsluha je na úrovni predošle dostupných ukážkových projektov.

Generovaný kód je riešený ako kompozitné zariadenie, vďaka tomu je možné ľubovolne generovať prídavné rozhrania. To pri vývoji z SDK ukážok nebolo úplne možné. Každá SDK ukážka implementuje jednu konkrétnu triedu zariadenia alebo kombináciu zariadení. Spôsob akým sú utvorené neponúka flexibilitu v ich rozširovaní o iné triedy implementované v separátnych projektoch.

Dostupnosť MCU jednotiek, ktoré pre funkciu USB rozhrania nepotrebujú externé obvody, naďalej zjednodušujú jeho implementáciu. Z požiadavok na návrh DPS pre USB zariadenie ostávajú len požiadavky limitujúce maximálnú dĺžku vodivých ciest, parazitickú kapacitu a zarušenie. Pri implementácií USB a oživovaní periférnych zariadení je výhodné použiť vývojovú dosku. Tá pridáva funkcionality ako JTAG, rozhranie pre debugger, tiež aj rôzne obvody, ktoré môžu byť použité na testovanie komunikačných rozhraní (I2C kompas, akcelerometer.)

Zariadenie vytvorené v tejto práci je možné rozšíriť o RS-232 rozhrania ktoré by umožnili pripojiť meracie prístroje alebo akékolvek iné periférie ktoré umožňujú vykonávať merania (I2C,ADC...), a stlačením tlačidla vykonať hromadný zber dát. Periférie, množstvo zdrojov dát, formát, a ako má byť nimi naložené pri výstupe emulovanou klávesnicou, To všetko je možné meniť z niekolkých presne vymedzených miest v kóde ktorými sú callback funckie obsluhujúce funkcie zariadenia. Vďaka VCOM rozhraniu je možné meniť konfiguráciu zariadenia za behu pomocou konfiguračného reťazca. Takéto zariadenie môže nájsť svoje uplatnenie nie len v automatizácií úkonov v laboratórnych úlohách, Ale kdekoľvek kde je potrebné vykonať zápis alebo posielanie údajov na prenosné zariadenia, bez inštalácie akéhokoľvek programového vybavenia na cielené PC.

Literatúra

[1] *Universal Serial Bus Specification* [online]. Revision 2.0 April 27, 2000. URL: <http://sdphca.ucsd.edu/Lab_Equip_Manuals/usb_20.pdf>.

[2] *Device Class Definition for Human Interface Devices (HID)* [online]. Firmware Specification—6/27/01 Version 1.11 USB Implementers' Forum URL: <https://www.usb.org/sites/default/files/documents/hid1_11.pdf>.

[3] *Universal Serial Bus Test and Measurement Class Specification (USBTMC)* [online]. Revision 1.0 April 14, 2003 USB Implementers' Forum URL: <http://sdpha2.ucsd.edu/Lab_Equip_Manuals/USBTMC_1_00.pdf>.

[4] Jan Axelson USBComplete *Everything You Need to Develop Custom USB Peripherals*, 3. vyd. Lakeview Research, 2005. 572 s ISBN13 978-1-931448-03-1.

Zoznam symbolov, veličín a skratiek

DIC	Data interface class - trieda datového rozhrania
CIC	Communication interface class - trieda komunikačného rozhrania
VCOM	Virtual Communications Port - Virtuálne hardwarové rozhranie
SCPI	Standard Commands for Programmable Instruments – štandardné príkazy pre programovatelné inštrumenty
USB	Universal Serial Bus – univerzálna sériová zbernica
HID	Human Interface Device – Trieda Pre rozhranie s človekom
TMC	Test & Measurement Class – trieda určená pre testovacie a meracie aparatúry
BCD	Binary Coded Decimal – B dvojkovo kódovaná dekadická číslica
ADC	Analog to Digital Converter – Analógovo číslicový prevodník
DAC	Digital to Analog Converter – Digitálne Analógový prevodník

Zoznam príloh

A Hlavné časti programu pre implementáciu USB rozhrania

```
/* Interface callback */
usb_status_t USB_DeviceInterface0HidKeyboardCallback(class_handle_t handle, uint32_t event, void *param)
{
    usb_status_t error = kStatus_USB_Error;
    switch (event)
    {
        case kUSB_DeviceHidEventSendResponse:
            if (s_UsbDeviceComposite->attach)
            {
                return USB_DeviceHidKeyboardAction();
            }
            break;
        case kUSB_DeviceHidEventGetReport:
        case kUSB_DeviceHidEventSetReport:
        case kUSB_DeviceHidEventRequestReportBuffer:
            error = kStatus_USB_InvalidRequest;
            break;
        case kUSB_DeviceHidEventGetIdle:
        case kUSB_DeviceHidEventGetProtocol:
        case kUSB_DeviceHidEventSetIdle:
        case kUSB_DeviceHidEventSetProtocol:
            break;
        default:
            break;
    }

    return error;
}
```

Obr. A.1: Callback obsluhujúci rozhranie USB klávesnice.

Item		Value (Hex)
Usage Page (Generic Desktop),		05 01
Usage (Keyboard),		09 06
Collection (Application),		A1 01
Usage Page (Key Codes);		05 07
Usage Minimum (224),		19 E0
Usage Maximum (231),		29 E7
Logical Minimum (0),		15 00
Logical Maximum (1),		25 01
Report Size (1),		75 01
Report Count (8),		95 08
Input (Data, Variable, Absolute),	;Modifier byte	81 02
Report Count (1),		95 01
Report Size (8),		75 08
Input (Constant),	;Reserved byte	81 01
Report Count (5),		95 05
Report Size (1),		75 01
Usage Page (Page# for LEDs),		05 08
Usage Minimum (1),		19 01
Usage Maximum (5),		29 05
Output (Data, Variable, Absolute),	;LED report	91 02
Report Count (1),		95 01
Report Size (3),		75 03
Output (Constant),	;LED report padding	91 01
Report Count (6),		95 06
Report Size (8),		75 08
Logical Minimum (0),		15 00
Logical Maximum(101),		25 65
Usage Page (Key Codes),		05 07
Usage Minimum (0),		19 00
Usage Maximum (101),		29 65
Input (Data, Array),	;Key arrays (6 bytes)	81 00
End Collection		C0

Obr. A.2: Report descriptor štandardnej klávesnice [2].