VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 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 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Branislav Hatala ***ID:*** 195308

Ročník: 3 ***Akademický rok:*** 2019/20

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: 3.2.2020 ***Termín odevzdání:*** 3.8.2020

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

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

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 , 2020, 51 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 3.8.2020 .....

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 3.8.2020 .....

podpis autora

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.

Funkcionalita a proces používania výsledného zariadenia sa dá rozdeliť do 3 krokov:

1. Nastavenie úlohy spúšťanej stiskom tlačidla,
2. Vykonanie úlohy(samotný zber dát),
3. posielanie získaných dát.

Pričom funkcionalita 1 a 3 sú implementované pomocou USB rozhrania, kde sú porovnané rôzne spôsoby, ako tieto funkcionality implementovať.

V tejto práci sa do hĺbky rieši implementácia kompozitného zariadenia, ktoré spája emulátor klávesnice a virtuálne konzolové rozhranie na platforme od firmy NXP. 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 sprostredkovanie 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

Úlohu implementácie USB rozhrania v mikrokontroléri je možné zvládnuť na rôznych úrovniach. Najjednoduhšou možnosťou je vytvoriť USB zariadenie z obmedzenou funkcionalitou, ktoré je schopné enumerovať sa a plniť svoju základnú funkciu bez toho, aby splnilo USB štandard. Takéto zariadenie nie je možné uviesť na trh pod označením USB zariadenia a nie je možné preň získať vlastný VID a PID.Inou úrovňou implementácie USB zariadenia je vyhovenie USB štandardu.

1.1 USB štandard

Výrobcovia mikrokontrolérov sa snažia svojim klientom pri snahe vyhovieť USB štandardu pomôcť svojimi nástrojmi a dokumentáciou. Je možné nájsť dokumenty formátu zaškrtávacieho zoznamu, príkladom je dokument [14]. Splnenie veľkej časti podmienok definovaných v USB štandarde spadá na konštruktérov použitého mikrokontroléru, pričom sú medzi nimi požiadavky, ktoré treba zohladniť pri konštrukcií dosky plošného spoja. Ďalšie požiadavky sú kladené na programové vybavenie obstarávajúce USB funkcionalitu. V tejto práci sa do väčšieho detailu budeme zaoberať niekoľkými vybranými problémami, ktoré sa týkajú užívatela mikrokontrolérov najviac.

Samotné vlastníctvo VID je každoročný náklad vo výške 5000$. Uvedenie zariadenia na trh nevyžaduje priamo splnenie USB štandardu a vlastné VID, niektorý výrobcovia mikrokontrolérov ponúkajú možnosť použiť ich VID pre vývoj, testovanie a omedzenú produkciu. NXP ponúka túto možnosť vo forme **USB VID PID Program**, pričom ich obmedzenie na produkciu je 10000 jednotiek a len 3 rôzne PID môžu byť ponúknuté jednému zákazníkovi. Ďalšou nevýhodou tejto možnopsti je že nie je možné použiť logá spoločnosti USB-IF, ktoré členovia používajú na indikáciu jednotlivých štandardov. Túto možnosť v iných prevedeniach podporujú aj firmy ako:

- STMicroelectronics,
- Mikrochip,
- Texas Instruments,
- Luminary Micro,
- FTDI,
- Silicon Labs.

Členovia open-source komunity využívajú niekedy VID 0xF055. Je dôležité podotknúť, že tento fakt je skôr zaujímavosť, než metóda, kedže USB-IF nepodporuje tento prístup. [11]

1.1.1 Napájanie

USB zariadenie môže byť samonapájacie, alebo napájané zbernicou.USB špecifikácia diktuje limity v odbere energie v jednotkách prúdu pre zariadenia napájané zbernicou, pričom maximálna možná spotreba zariadenia cez jeden USB port je 500mA. Akékoľvek zariadenie musí byť schopné enumerácie a komunikácie zo zbernicou bez toho, aby presiahlo 100mA, toto obmedzenie platí aj pre zariadenia, ktoré potrebujú pre svoju funkciu napájací prúd do 500mA. Tento problém sa rieši pomocou dvoch rôznych konfigurácií, kde jedna slúži len na pripojenie zariadenia a komunikáciu s ním bez toho, aby plnilo svoju primárnu funkciu, na ktorú potrebuje väčší prúdový odber. Do druhej konfigurácií je prepnuté hostom po tom, čo bolo overené, či je možné, aby zbernica dodala zariadeniu jeho deklarovaný prúd pre jeho druhú konfiguráciu.

1.1.2 Suspend, resume, remote wakeup

Na zariadenie napájané zbernicou sú kladené ďalšie nároky týkajúce sa šetrenia energie. Jedným z nich je suspendovanie zariadnia. Ak zariadenie nezaznamená žiadnu komunikáciu na zbernici po dobu 3 ms, tak má ďalších 7 ms na to, aby sa dostalo do stavu, v ktorom odberá zo zbernice menej než 500uA. V prípade, že zariadenie podporuje a má zapnutú možnosť remote wakeup, tak je tento limit 2.5 mA, pričom tieto hodnoty prúdu udávajú priemernú spotrebu prúdu za dobu 1 s.

Remote wakeup je možnosť zariadenia, ktoré je suspendované zobudiť sa nezávisle od zbernice a vyslať do zbernice požiadavok na zobudenie zbernice.Táto možnosť sa používa na zobudenie uspatého počítača pomocou pripojeného zariadenia. Takéto zariadenie nesmie presiahnúť spomenutý 2.5 mA limit, pokiaľ je zbernica suspendovaná. Odsuspendovanie zbernice vykonáva host. Za týmto účelom sa má zariadenia po vyslaní remote wakeup signálu znovu uviesť do stavu, v akom odoberá nízky prúd.

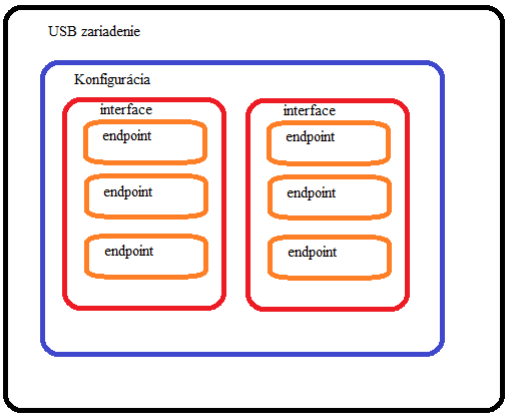
Ak zariadenia podporuje remote wakeup, tak je nutné programovo ošetriť nasledujúce situácie:

- Uspanie zbernice po zakázaní remote wakeup pre zariadenie,
- zapojenie do uspatej zbertnice,
- vyslanie remote wakeup signálu, ale host nevyhovie požiadavku

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, čislom 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 štrukturované.

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 rožširujú štandard, tým že prídávajú vrstvu nad základnou USB špecifikáciou. Jednotlivé triedy majú svoje vlastné špecifikácie, vrátane voľne dostupných dokumentov.

1.2 Základný koncept

V tejto kapitole je ukážkové zariadenie rozdelené na jeho 3 hlavné časti, ktoré plnia následné úlohy:

- komunikácia dát
- nastavenia merania,
- Získavanie a zpracovanie dát.

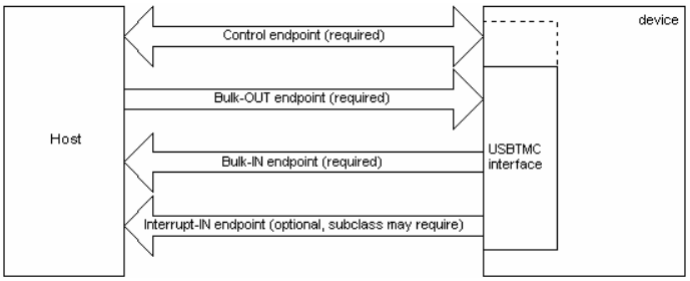
1.2.1 Rozhranie komunikácie dát

Úlohou tejto časti je zabezpečiť presun nameraných dát do pripojeného počítača. V tejto podkapitole sú uvedené rôzne možnosti realizácie tejto úlohy, pričom sú popísané výhody a nevýhody jednolivých možností a dôvod voľby medzi nimi.

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

[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

vstup a druhý pre výstup. Niektoré podtriedy vyžadujú interrupt[6](Prerušujúci prenos) endpoint na vstup[7]. Ďalšie 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.

Ďalšou nevýhodou tejto varianty je absencia podpory výrobcov MCU, oni neposkytujú nástroje na jeho implementáciu a tak SDK, middleware ani driver nie sú dostupné.

Zložitosť implementácie a nároky na špecializovaný software robia túto variantu značne nepraktickú.

USB VCOM

Táto varianta v skutočnosti pozostáva z 2 USB rozhraní. Jedná sa o virtuálnu konzolu, tá sa pre operačný sytém na pripojenom počítači javý podobne ako pripojené RS-232 rozhranie. Je možné sa na takéto zariadenie pripojiť ľubovolným terminálným rozhraním, ako sú PuTTY alebo Terra Term.

Data získané cez takéto rozhraní sú vypísané v okne terminálu a je nutné ich zaznamenať a sformátovať dodatočne.

Výhodou tejto varianty je možnosť zlúčenia úloh komunikácie dát a nastavenia merania. Takéto zlúčenie je možné vykonať aj separátne, výsledkom budú defakto 4 rozhrania tvoriace 2 separátne virtuálne konzoly.

Kedže týmto spôsobom je možné realizovať aj ďalšiu úlohu ukážkového zariadenia, pre zaujímavejší výsledok bude iná varianta uprednostnená.

Mass Storage Class

USB MSC je USB trieda, ktorú využívajú externé hard-disky, USB kľúče a iné zariadenia, do ktorých pamäti má host priamy prístup. Operačný program Windows využíva Explorer na obsluhu takýchto zariadení a tak odpadá problém zo závislosťou na externý software.

Dôležitou skutočnosťou pri posudzovaní a implementácií tejto možnosti je fakt, že Windows 10 priamo zapisuje vlastné meta dáta na každú pripojenú partíciu. Tento fakt spôsobil opakované znefunkčnovania vývojových kitov a debbugerov, na ktoré sa bootloader nahrával použitím USB MSC tried. Medzi ovplyvnené zariadenia patrí

[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

aj FRDM-k64, ktorá bola použitá v tejto práci. Tento problém bol spôsobený tým, že Windows 10 prepísal časť bootloader-u metadátami. Bolo to riešené v ďalšej z verzií programového vybavenia, nahrávajúcej jednotky zasiahnutej vývojovej dosky.

Nevýhoda tejto varianty spočíva v tom, že nejde o riešenie v reálnom čase a tak získané hodnoty vidieť až po opätovnom otvorení výstupného súboru a nie priebežne.

Human Interface Class

USB HID zariadenie sa podľa protokolu delí na:

- žiaden,
- klávesnica,
- myš.

Táto varianta sa zaoberá práve nevyužitím žiadneho protokolu, výrobcovia nazývajú takúto implementáciu tiež názvom **HID Generic**.

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 ďalšie dva descriptory: report descriptor (popisná štruktúra správy) a physical descripto (fyzická popisná štruktúra). Zariadenie môže 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 funkciu 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"(uzavierací prvok súboru prvkov) tento kód 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.

Nevýhodou je závislosť na netypickom programovom vybavení pre počíta ako sú driver a software.

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].

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.

Nevýhodou je nutnosť správneho fokusu v operačnom systéme pri získávaní dát, je možné odstrániť túto nevýhodu takým prevedením, kde sa po poslednom meraní pošlú všetky hodnoty. Takéto riešenie neumožnuje merané hodnoty sledovať priebežne.

Pre výhody tejto varianty a to najmä softwareovej nezávislosti a priamosti získavania dát bude v tejto práci použitá táto varianta, pričom nebude použitý kumulatívny spôsob komunikácie dát popísaný v predošlom odstavci.

1.2.2 Rozhranie nastavenia merania

Úlohou tejto časti je zabespečenie prispôsobitelnosti zariadenia pre rôzne úlohy. Zohladňujú sa možnosti vo variabilite počtu meraných hodnôt, spôsobe ich merania, meracích prístrojov a ich komunikačnými rozhraniami. Podobne ako v predošlej

kapitole, budú rozoberané výhody, nevýhody a dôvod voľby medzi jednotlivými variantami.

Niektoré riešenia zahrňujú použitie rozhraní už spomenutých v predošlej pod-kapitole, napriek tomu, že niektoré výhody a nevýhody ostávajú rovnaké. Platí skutočnosť že pre riešenie tejto úlohy sú rozhrania použité iným spôsobom.

Dôležitým kritériom pri výbere medzi jednotlivými variantami riešenia je značný vplyv na implementáciu samotného zberu a získavania dát, kedže spôsob akým je táto úloha riešená sa odvíja od implikácií jednotlivých variant rozoberaných v tejto pod-kapitole.

Mass storage class

Takéto riešenie by spočívalo v konštrukcií interpretovaného jazyka, ten by definoival úkony vykonávané pri jednotlivých opakovaniach merania, prípade úkony pred začatím prvého merania.

Výhodami takéhoto prístupu sú možnosť plnej automatizácie.Jeden súbor, ktorý sa prenáša do pamäti zariadenia môže definovať viac rôznych úkonov. Samotné nahrávanie je možné automatizovať použitím dávkového súboru.

Device Firmware Update

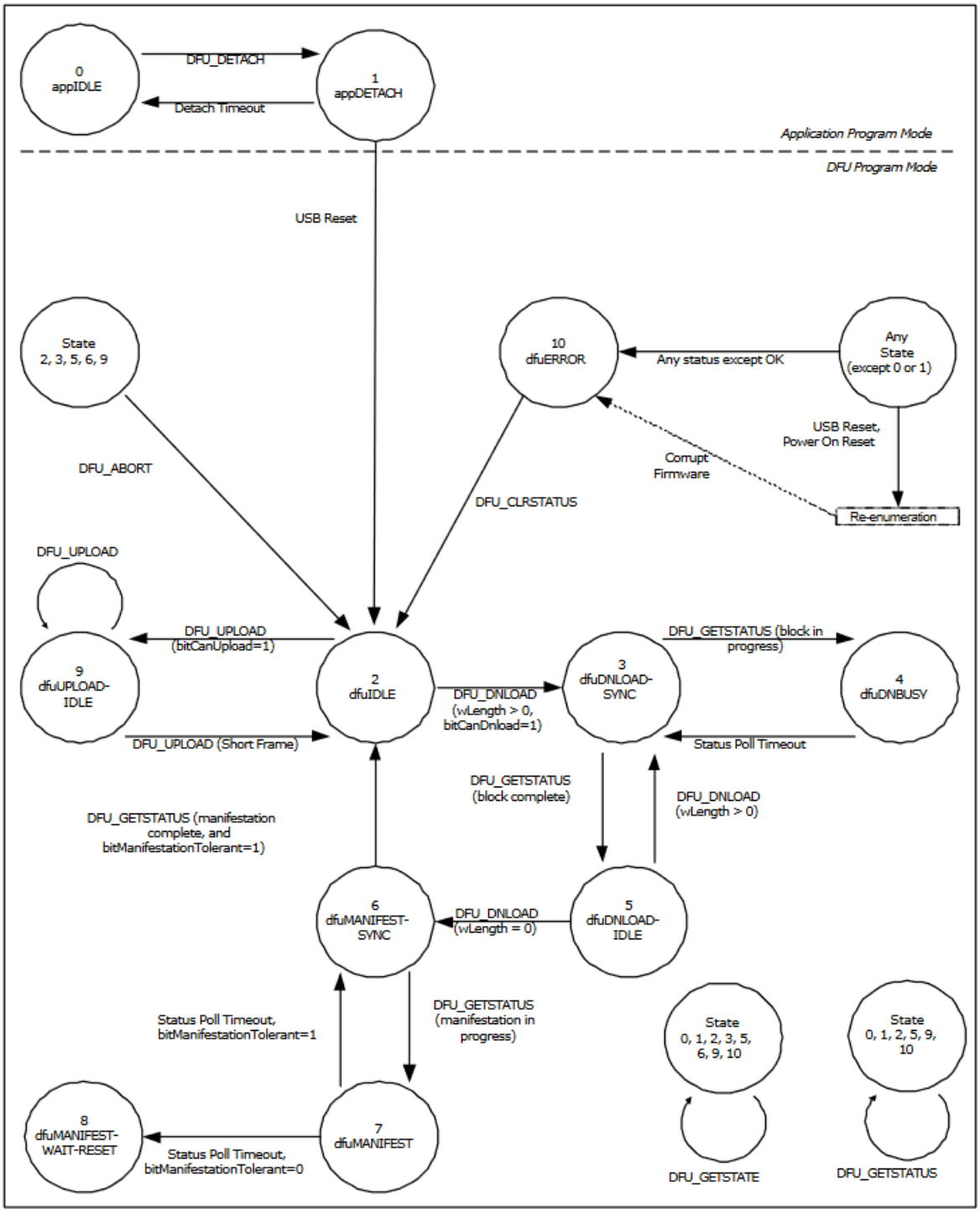
Táto trieda rozhrania umožnuje uživateľovi zariadenia aktualizovať firmware zariadenia. Výrobcovia zariadenia majú možnosť implementovať vlastný program na vykonávanie tohoto úkonu, alebo použiť dfu-util, jedná sa o software z GPL2 licenciou.

Dfu-util je hosťovská implementácia implementácia DFU1.0 a DFU1.1 špecifikácie. DFU je pužívaný na sťahovanie a nahrávanie firmware-u do zariadenia pripojeného cez USB. Rozsah použitia je od malých zariadení, ako sú mikrokontroléry až po mobilné telefóny.[13]

Táto varianta spočíva v implementácií programového vybavenia na mieru pre jednotlivé úlohy, ktoré zariadenie bude plniť. Funkcionalita zariadenia nebude nastaviteľná, namiesto toho sa bude meniť celý firmware.

Podobný výsledok sa dá dosiahnúť ale aj bez implementácie ďalšieho rozhrania. To je možné dosiahnúť použitím debugovacieho rozhrania, za použitia programátoru. Jedinou výhodou použitia DFU oproti tomuto spôsobu je možnosť prepísať firmware bez dodatočného hardware v podobe programátoru či debug sondy, nevýhodou spoliehania sa na DFU je absencia možnosti debugovania kódu.

Zariadenia implementujúce DFU rozhranie funguje ako stavový automat viz1.4. Prerušovaná čiara v stavovom diagrame rozdeluje dva režimy v ktorých zariadenie funguje.



Obr. 1.4: Stavový diagram implementácie DFU[12].

V aplikačnom režime (nad čiarou) zariadenie plní svoje funkcie, pritom DFU rozhranie je v pohotovostnom režime. Režim sa mení pomocou sledu dvoch udalostí,

požiadavok na DFU rozhranie a USB reset[8] . Po prechode do DFU režimu sa opakuje enumerácia zariadenia, pričom v tomto režime sa líšia descriptory zariadenia.

V DFU režime zariadenie pracuje ako čisto DFU zariadenie bez ďalších USB rozhraní, táto podmienka implikuje zmeny v konfiguračnom descriptore. Operačné systémy po prvotnom enumerovaní používajú na identifikáciu zariadenia výhradne kombináciu PID a VID, táto skutočnosť spôsobuje že je nutné, aby zariadenie v DFU režime bolo identifikované odlišným PID kódom ako v aplikačnom režime PIDa je súčasťou deskriptoru zariadenia.

Táto varianta má niekoľko nevýhod. Hlavnou z nich je, že samotná realizácia DFU je závislá od schopnosti platformy prepísovať flash pamäť, alebo v prípade kompromisu, schopnosť platformy vykonávať inštrukcie s procesorom zapisovateľnej pamäte. U malých mikrokontrolérоch tieto možnosti nie sú zaručené.

DFU postupne posiela a ukladá firmware v pamäti zariadenia až po tom, čo je prenesený celý nový firmware, nasleduje fáza manifestácie a zavádzania nového firmware. Vo fázi manifestácie je celý nový firmware uložený v pamäti spolu zo starým stále bežiacim, tento spôsob fungovania je značne pamäťovo náročný v prípade malých mikrokontrolérov.

Ďalšou nevýhodou je to akým spôsobom je DFU použitá na plnenie svojej úlohy v tomto prípade. Pri každej ďalšej meracej úlohe je nutné vykonať nasledujúce kroky:

1. Vyvinie sa a odladí frimware pre danú meraciu úlohu,
2. Výsledný binárny súbor sa doplní o CRC sekvenciu,
3. Použie sa dfu-util na aktualizáciu zariadenia.

VCOM

Podobne ako pri použití MSC, aj v tomto prípade riešenie spočíva v použití interpretovaného jazyka. Výhodou oproti MSC pri zadávaní príkazov postupne cez konzoilové okno je okamžitá spätná väzba v podobe chybových hlášok a oznamoch o stave zariadenia cez konzolové okno.

Nevýhodou je závislosť na software, pre pouyžívanie je potrebný konzolový emulátor. Táto výhoda je mierna, kedže potrebný software je dostupný ako share-ware pre rôzne platformy.

V tejto práci bude implementovaný práve tento prístup.

[8]Reset USB rozhrania nevyžaduje reset zariadenia

1.3 Prehľad implementácie USB rozhrania

1.3.1 Software implementácia

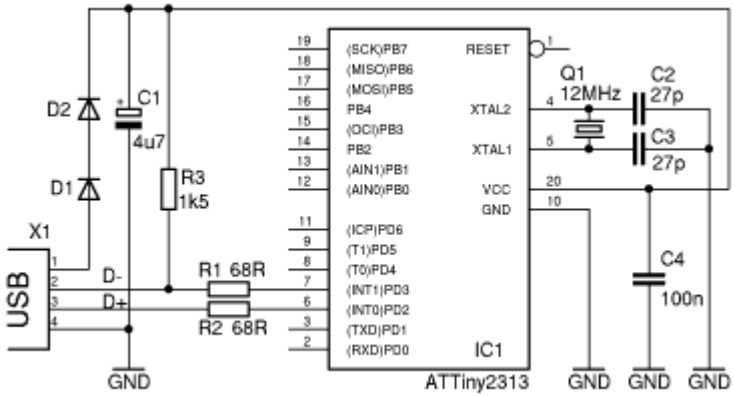
Existuje niekoľko rôznych prevedení ktoré implementujú USB rozhranie cez software, najznámejšie sú:

- V-USB[8],
- grainuum[7],
- USBtiny[9].

Spoločným znakom týchto prevedení je práca v low-speed(Nízka rýchlosť) režime. Tento režim okrem iných obmedzení má maximálnu rýchlosť 1.5 Mbit/s a rýchlosť posielania packetov(Polling Rate) je 125 Hz. Ďalším problémovým faktorom ktoré software implementácia prináša je väčšie využitie zdrojov mikrokontroléru na obsluhu USB rozhrania.

V-USB je čisto software-ová implementácia low-speed USB rozhrania pre Atmel AVR mikrokontroléry umožňujúca tvorbu USB zariadenia na skoro akejkoľvek AVR platforme bez nutnosti prídavného integrovaného obvodu.[8]

VUsbObvod



Obr. 1.5: Obvod pre použitie V-USB[8].

Na web stránke V-USB sú uvedené výhody použitia V-USB oproti využitiu mikrokontroléru z využívajúcemu USB hardware:[8]

- Štandardné AVR mikrocontroléry sú jednoduhšie dostupné,
- Väčšina MCU sú dostupné len v SMD púzdrach,
- V-USB má voľne prístupný zdieľaný VID PID pár ,
- Dostupnosť vývojových prostriedkov pre AVR,
- AVR mikrokontroléry majú integrovanú EEPROM pamäť

Všetky tieto uvedené dôvody sú zastaralé, zaťial čo jednotková cena ATTiny2313 použitej v ukážke zapojenia je približne tretinová oproti najlacnejšiemu Cortex-M0+ mikrokontroléru z integrovaným USB hardware, vývojové prostriedky a proces vývoja je optimálnejší pre platformu použitú v tejto práci.

Zatiaľ čo poznámka o SMD púzdrach je pravdivá, výsledkom použitia povrchovej montáže je v prípade mikrokontrolérov závislosť na nespájkovatelnej maske dosky plošného spoja. Tento problém je dnes už eliminovaný dostupnosťou prototypovej výroby dosky plošného spoja.

Výrobcovia riešia problém z VID PID číslami ako už bolo v tejto práci spomenuté pomocou zprostredkovacích programov.

Dnešné vývojové prostriedky rôznych firiem implementujú rôzne konfiguračné nástroje do svojich integrovaných vývojových prostriedkov. Tieto nástroje šetria čas tým, že umožnujú nastavenie periférií mikrokontroléru bez úplnej znalosti ich implementácií. Dnešným trendom je posúvanie vývoju aplikácií mikrokontrolérov do vyššých vrstiev abstrakcie.

AVR mikrokontroléry boli medzi prvými čo implementovali integrovanú EEPROM pamäť, dnes sa jedná už o samozrejmosť. Vývoj v prístupe k pamäti a programovaniu naďalej zdostupňuje proces debuggovania a programovania. Vývojové kity NXP majú integrovaný debugger ktorý môže využiť DAP alebo J-link. Zatiaľ čo mikrokontroléry samotné môžu využiť SWD10 konektor pre prístup cez PE micro sondu.

USBtiny je určená pre AVR platformy z fekvenciou hodinových pulzov 12 MHz, s takouto frekvenciou má bit USB zbernice dĺžku 8 hodinových taktov.[9]

Grainuum USB je navrhnutá pre Crotex-M0+ procesory z taktovacou frekvenciou 48MHz a jednocyklovými vstupno-výstupnými operáciami. Prakticky to znamená že Grainuum funguje pre širokú škálu zariadení rady Kinetis.[7]

Otázne je do akej miery Software implementácie USB zbernice spĺňajú požiadavky na kvalitu signálov zbernice. USB štandard definuje toleranciu časovania pre low-speed 1.5

1.3.2 Hardware implementácia

USB rozhranie je v mikrokontroléroch typicky implementované vo forme hardware modulu. Rôzne moduly majú odlišné vlastnosti, vo výsledku rôzny výrobcovia mikrokontrolérov dosahujú vzájomne odlišujúce sa parametry USB modulov. V rámci jedného výrobcu sa môžu vyskytovať rôzne USB moduly. V táto kapitola sa bude zaoberať vysvetlením a porovnaním vlastností USB modulov od výrobcov:

- NXP,
- Microchip,

- STMicroelectronics.

Elektrické vlastnosti

Dôležitým parametrom pri návrhu USB zariadenia je impedancia rozdielového vedenia. Tá má byť podľa štandardu prispôsobená na 90 Ohmov. Z tejto skutočnosti vyplýva, že dôležitou vlastnosťou je impedancia, ktorú USB hardware vnáša do obvodu a tá sa líši medzi výrobcami. Nominálna impedancia sa dosahuje pripojením rezistorov do dráhy rozdielového vedenia, príklady hodnôt rezistorov sú NXP: 33 Ohm, STMicroelectronics: 22 Ohm. USB obvody mikrokontrolérov microchip nevyužívajú prídavné rezistory.

Pull-up a pull-down rezistory pripojené na vodiče D+ a D- datovej zbernice USB majú zároveň signalizačný účel. Zariadenie implementuje na svojej strane jeden pull-up rezistor o veľkosti 1.5 kOhm na napätie 3.3 V, low-speed zariadenie tento rezistor pripája na vodič D-, full-speed a high-speed zariadenie na D+ vodič. Host zapája pull-down rezistory o veľkosti 15 KOhm na oba vodiče zbernice. Tieto rezistory sú implementované vnútrne v USB module mikrokontroléru a je možné ich zapojenie meniť za behu zariadenia.

Napájanie

Niektoré mikrokontroléry implementujú integrovaný regulátor na získávanie napájacieho napätia zo zbernice USB.

Ďalšou vlastnosťou ohladne napájanie je prítomnosť modulu pre detekciu nabíjania u STM zariadení označovaná ako BCD, u NXP ako DCD.

USB nabíjacia špecifikácia pre batérie definuje limity, detekciu, riadenie a oznamovací mechanizmus, ktoré dovolujú zariadeniu odoberať väčší prúd než USB2.0 špecifikácia dovoluje.[6]

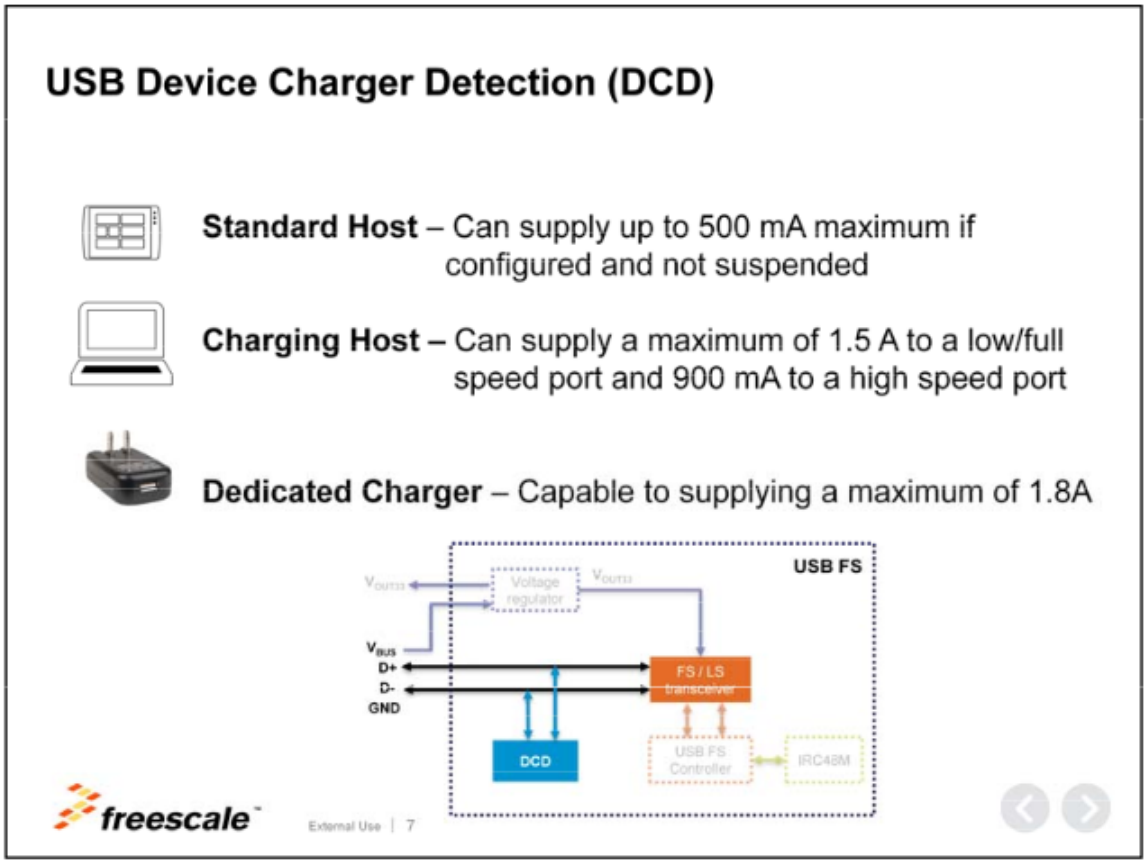
USB rýchlosti

Väčšina výrobcou ponúka v dnešnej dobe mikrokontroléry z USB obvodmi podľa štandardu USB2.0, v rýchlostich:

- low-speed, 1.5Mbpa,
- full-speed 12Mbps,
- high-speed 480Mbps.

Host a OTG

Ďalšie možnosti ponúkané modulmi USB sú možnosti fungovania ako HOST alebo OTG. Tieto možnosti vyžadujú prispôsobenie obvodového riešenia pri implementácií



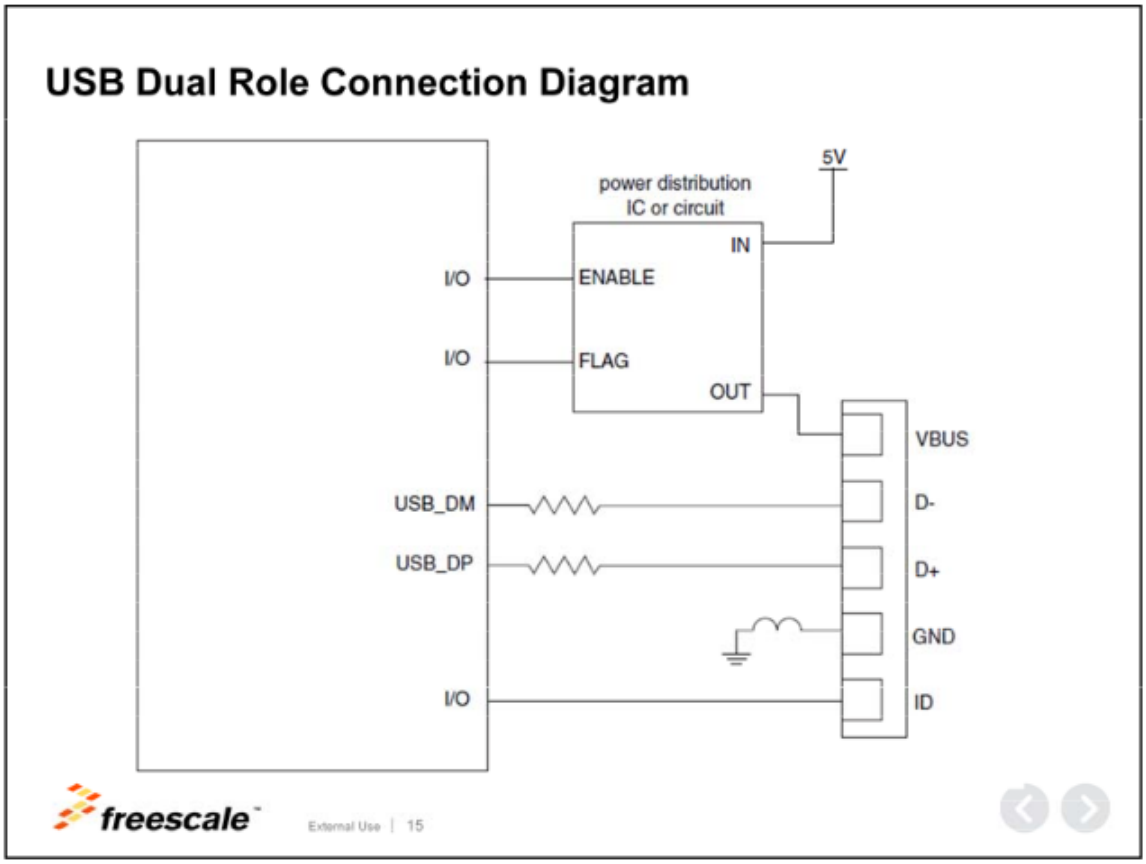
Obr. 1.6: Typy USB portov podľa DCD[6].

mikrokontroléru, nutné je implementovať schopnosť napájania zbernice, v prípade OTG schopnosť rozoznať pozíciu(host/zariadenie).

Host režim umožňuje pripájanie USB zariadení k mikrokontroléru.

Zatiaľ čo full-speed je považovaná za štandardnú rýchlosť, podpora high-speed závisí od konkrétneho mikrokontroléru alebo jeho rodine.

OTG režim umožnuje zariadeniu fungovať striedavo ako host, alebo zariadenie. Tento režim využívajú tablety a mobilné telefóny. Umožňuje im to používať USB zariadenia a zároveň pripojenie k počítaču za účeom prenášania súborov a nabíjania. To či je k zariadeniu cez USB zbernicu pripojené zariadenie alebo host sa určuje pomocou ID vodiča to, že sa má zariadenie správať ako host je indikované nízkou logickou úrovňou napätia. Príklad obvodu OTG zariadenia je uvedený na obrázku1.7.



Obr. 1.7: príklad zapojenia OTG zariadenia[6].

Korekcia Hodinového signálu

USB rozhranie definuje prísné požiadavky na časovanie komunikácie. Výrobcovia mikrokontrolérov v snahe ušetriť náklady na používanie ich produktov implementujú mechanizmus umožňujúci použiť vnútorný oscilátor na časovanie USB zbernice. Tento obvod je možné použiť len v režime zariadenia.

Spomenutý výrobcovia tento mechanizmus implementujú bežne a je nazývaný ako **clock recovery circuit**.

Tento mechanizmus využíva SOF signál posielaný hostom v pravidelných intervaloch, USB modul detekuje tento signál a porovnáva čas jeho príchodu s jeho predpokladaným príchodom podľa časovača taktovaného vnútorným oscilátorom. Na základe výsledného rozdielu je vykonaná korekcia(trim) oscilátoru, tento proces prebieha kontinuálne za celej operácie zariadenia.

Obsluha USB modulu

Software plní úlohu obsluhy USB modulu tým, že plní buffer dátami. V bežných prevedeniach má každý endpoint USB zariadenia vlastný buffer. Typycky sa jedná o dvojité bufferovanie implementované hardwareovo. Vo výsledku USB modul má obmedzený počet endpointov[9].

USB modul generuje prerušenia pre udalosti ohľadom posielania, stavu zbernice a USB žiadostí obdrdžaných.

Software výrobcov mikrokontrolérov USB driver a stack nasledujú trend posúvania vývoja USB zariadenia do vyšších abstretkných vrstiev. Vďaka tomuto prístupu uživatelský kód na obsluhu USB rozhrania pozostáva z implementácie callback funkcií, ktorých parametre dosadzuje driver/stack. Tento prístup umožňuje zároveň odabstrakovať rozdiely medzi USB modulmi tak, že uživatelská implementácia je v prípade migrácie indeferentná.

Software pre obsluhu USB modulu plní svoju úlohu tým, že v reakcií na prerušenia plní buffer vyžadovanými datami a vykonáva obdržané požiadavky. Príkladom môže byť fungovanie vstupného zariadenia, buffer vstupného endpointu je plnený datami nezávislo od diania na zbernici. V prípade že je zahájený USB prenos a buffer je prázdny, tak USB modul pošle ako odpoveď v prenose **NAK**[10] kód.

1.4 Obvodové riešenie USB rozhrania

USB zbernica na prenos signálu využíva diferenciálny pár vodičov. Štandard USB 2.0 dosahuje rýchlosť do 480 Mbps, to činí frekvenciu pulzov 240MHz.[10] Na posúdenie kvality obvodu diferenciálneho páru sa používa takzvaný "vzor oka"(eye pattern). Ide o normu, ktorá vyhradzuje zakázané oblasti v centrovanom grafe priebehu signálov na diferenciálnom páre. Parametre tejto zakázanej oblasti závisia od toho, ktorý z meracích bodov je použitý.

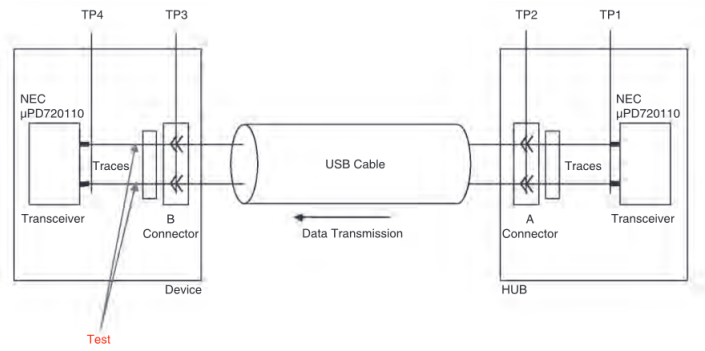
Ďalším dôležitým kritériom pri návrhu obvodu pre USB je odpor diferenciálneho páru, ten má byť podľa USB štandardu 90 Ohmov . Obe kritéria je možné splniť vhodne zvolenou prúdovo kompenzovanou tlmivkou, zapojenou na diferenciálny pár vodičov.

Dalším dôležitým konštrukčným prvkom je ochrana proti elektrostatickému napäti. V praxi používaným riešením je zapojenie transilu medzi jednotlivé vodiče a tienenie USB konektoru.

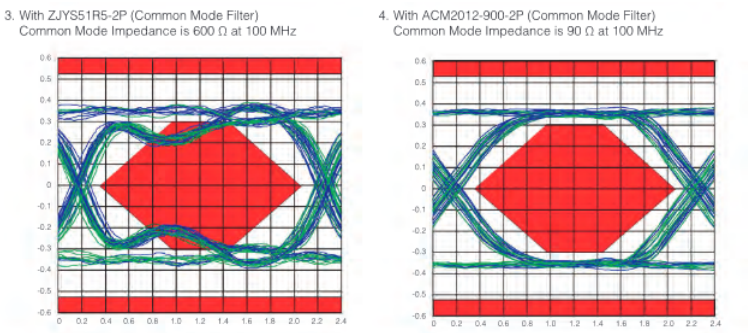
USB zbernica je schopná napájania zariadení USB 2.0, definuje maximum 500mA odberaného prúdu cez port zariadenia. Nutným konštrukčným prvkom je ochrana

[9]V prípade NXP Kinetis platformy je to 16 endpointov

[10]NAK kód ako odpoveď na vstup signalizuje že zariadenie nemá data na poslanie



Obr. 1.8: Meracie body pre eye pattern[10].



Obr. 1.9: ukážka nevyhovujúceho a vyhovejúceho eye pattern vzoru[10].

pred nárazovými prúdmi a tá je riešená bežňe zapojením napätia zbernice do zariadenie cez tlmivku.

2 Výsledky studentské práce

Úloha implementácie USB rozhrania môže byť pojatá na rôznych úrovniach. Vytvoriť zariadenie, ktoré prejde enumeráciou a plní svoju funkciu. Je to 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 Výber platformy

Kritickým požiadavkom na mikrokontrolér pre tvorbu ukážkového zariadenia je prítomnost USB modulu. Pre výber platformy sú dôležité aj vlastnosti, ktoré majú dopad na jednotkovú cenu, cenu vývoja a čas vývoja. Dostupnosť vývojových nástrojov a ich kvalita zohráva dôležitú úlohu pri výbere platformy. Ďalším v praci zohľadňovaným faktorom sú skúsenosti vývojárov a kompatibilita z už vlastnenými prostriedkami[2].

Ďalšie požiadavky brané do úvahy pri výbere mikrokontroléru sú:

- I2C rozhranie,
- SPI rozhranie,
- Funkčnosť bez externého oscilátoru.

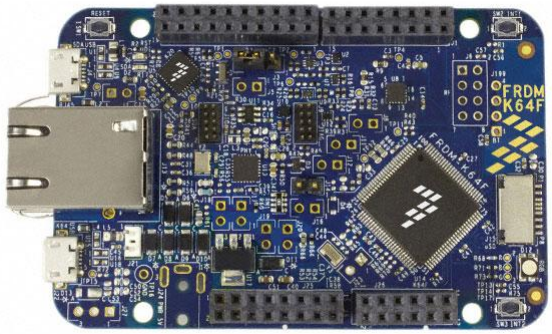
Kedže tvorca bakalárskej práce v čase jej tvorby sa podielal na vývoji konfiguračných nástrojov vo firme NXP, bude v tejto práci zvolená práve platforma od tohoto výrobcu. Veľkou prednosťou tohoto výrobcu je flexibilita jeho natívnych vývojových prostriedkov.

Táto práca sa zaoberá implementáciou USB rozhrania v malých mikrokontroléroch, z toho dôvodu je pre ukážkové zariadenie zvolená Kinetis platforma. Tá disponuje procesormi ARM-Cortex: M0+, M3 a M4. Programové vybavenie ktoré je výsledkom tejto práce je prenášatelné vrámci celej platformy Kinetis.

Software je vyvynutý na skúšobnej doske FRDM-K64F Obr.2.1. Jedná sa o rozpočtovú vývojovú dosku. Táto doska využíva mikrokontrolér MK64FN1M0VLL12,

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

[2] Jedná sa o software a hardware predošle vyvynutý.



Obr. 2.1: skúšobná doska FRDM-K64F

ten má ARM CORTEX-M4 jadro a obsahuje moduly, ktoré sú pre ukážkové zariadenie nadbytočné napr. zbernicu CAN a Ethernet. Použitie jadra ARM Cortex-M4 je tiež nadbytočné, kedže ukážkové zariadenie nevyužíva nasledovné:

- DSP inštrukcie,
- Aritmetiku z nasýtením,
- inštrukcie delenia,
- inštruckie násobenia.

ARM Cortex-M instruction variations

Arm Core	Cortex M0[2]	Cortex M0+[3]	Cortex M1[4]	Cortex M3[5]	Cortex M4[6]	Cortex M7[7]	Cortex M23[8]	Cortex M33[12]	Cortex M35P	Cortex M55
ARM architecture	ARMv6-M[9]	ARMv6-M[9]	ARMv6-M[9]	ARMv7-M[10]	ARMv7E-M[10]	ARMv7E-M[10]	ARMv8-M Baseline[15]	ARMv8-M Mainline[15]	ARMv8-M Mainline[15]	Armv8.1-M
Computer architecture	Von Neuman	Von Neumann	Von Neumann	Harvard	Harvard	Harvard	Von Neumann	Harvard	Harvard	Harvard
Instruction pipeline	3 stages	2 stages	3 stages	3 stages	3 stages	6 stages	2 stages	3 stages	3 stages	4 to 5 stages
Thumb-1 instructions	Most	Most	Most	Entire	Entire	Entire	Most	Entire	Entire	Entire
Thumb-2 instructions	Some	Some	Some	Entire	Entire	Entire	Some	Entire	Entire	Entire
Multiply instructions 32x32 = 32-bit result	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Multiply instructions 32x32 = 64-bit result	No	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Divide instructions 32/32 = 32-bit quotient	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Saturated instructions	No	No	No	Some	Yes	Yes	No	Yes	Yes	Yes
DSP instructions	No	No	No	No	Yes	Yes	No	Optional	Optional	Optional
Single-Precision (SP) Floating-point instructions	No	No	No	No	Optional	Optional	No	Optional	Optional	Optional
Double-Precision (DP) Floating-point instructions	No	No	No	No	No	Optional	No	No	No	Optional
Half-Precisions (HP)	No	No	No	No	No	No	No	No	No	Optional
TrustZone instructions	No	No	No	No	No	No	Optional	Optional	Optional	Optional
Co-processor instructions	No	No	No	No	No	No	No	Optional	Optional	Optional
Helium technology	No	No	No	No	No	No	No	No	No	Optional
Interrupt latency (if zero-wait state RAM)	16 cycles	15 cycles	23 for NMI 26 for IRQ	12 cycles	12 cycles	12 cycles	15 no security ext 27 security ext	TBD	TBD	TBD

Obr. 2.2: Porovnanie ARM Cortex jadier.[15]

Pre zníženie jednotkovej ceny zariadenia s ohľadom na predošle spomenuté sku-

točnosti, je zvolený mikrokontrolér s jadrom ARM Cortex-0+, konkrétne mikrokontrolér typu MKL27Z. Voľba jadra je uskutočnená na základe porovnania vlastností ARM Cortex jadier v tab.2.2. Voľba typu mikrokontroléru je uskutočnená pomocou nástroja **Product Selector**[16].

Mikrokontrolér MKL27Z je dostupný z rôznymi veľkosťami pamätí podľa výpisu2.3 je možné vybrať potrebnú veľkosť pamäti pre danú verziu programového vybavenia. Tiež je nutné zvážiť rozdiel vo velkosti spotreby pamäti zmenou architektúry mikrokontroléru a ďalším vývojom programového vybavenia v budúcnosti.

Všetky pamäťové varianty MKL27Z si sú vzájomne plne kompatibilné z hľadiska obvodu na doske plošného spoja a sú zamienateľné bez obmedzenia. Táto kompatibilita sa z istými obmedzeniami vzťahuje aj na ostatné mikrokontroléry z rodiny MKL2X s lqfp64 púzdrom.

```
Building target: MK64FN1M0xxx12_B_KeyboardEmu.axf
Invoking: MCU Linker
arm-none-eabi-gcc -nostdlib -Xlinker -Map="MK64FN1M0xxx12_B_KeyboardEmu.map" -
Memory region         Used Size  Region Size  %age Used
   PROGRAM_FLASH:       40488 B         1 MB      3.86%
      SRAM_UPPER:       10872 B       192 KB      5.53%
      SRAM_LOWER:          0 GB        64 KB      0.00%
        FLEX_RAM:          0 GB         4 KB      0.00%
Finished building target: MK64FN1M0xxx12_B_KeyboardEmu.axf

make --no-print-directory post-build
Performing post-build steps
arm-none-eabi-size "MK64FN1M0xxx12_B_KeyboardEmu.axf"; # arm-none-eabi-objcopy
   text    data     bss     dec     hex filename
  39976     512   10360   50848    c6a0 MK64FN1M0xxx12_B_KeyboardEmu.axf


14:43:40 Build Finished. 0 errors, 4 warnings. (took 4s.68ms)
```

Obr. 2.3: Výpis build príkazu pre projekt realizovaný na FRDM-k64.

2.2 Doska Plošného Spoja

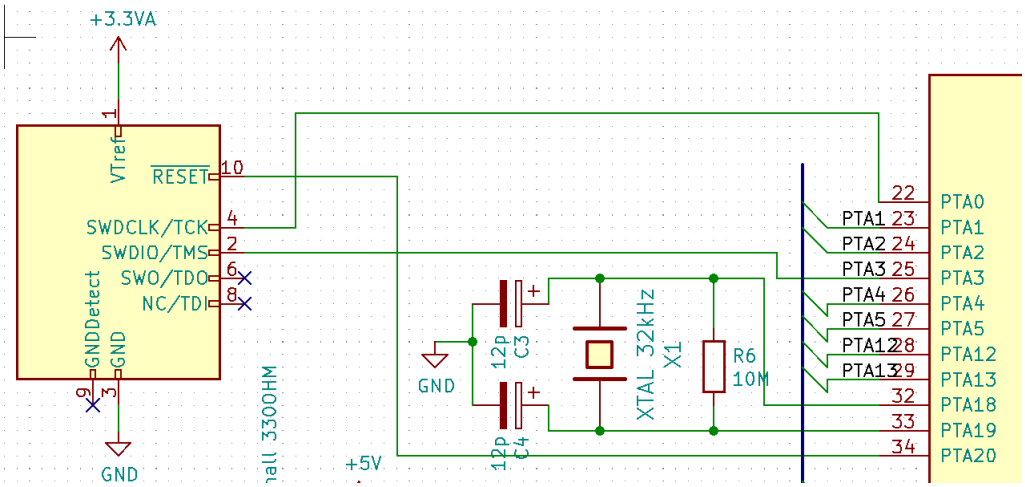
Doska plošného spoja využívajúca MKL27Z navrhnutá v tejto práci využíva bezkryštálový beh USB modulu a používa vnútorný zdroj hodinových impulzov na taktovanie jadra. Doska obsahuje 32kHz kryštálový oscilátor2.4 zaojený na XTAL0 vstup, ten slúži ako zdroj hodinových puzlov pre RTC modul.

Obvod je napájaný cez USB port, je možné voliť medzi interným a externým regulátorom napätia. Použitý mikrokontrolér má napájacie napätie 3.3V.

Na účely debugovania a nahrávania programu je použitý SWD-10 konektor2.4, Na ten je možné sa pripojiť P&E micro sondou. SWD-10 konektor pripája len nasledovné vývody:

- VRef,
- SWDCLK/TCK,

- SWDIO/TMS,
- GND.



Obr. 2.4: Zapojenie SWD a oscilátoru

USB obvod2.5 sa odvíja od obvodu použitého na vývojovej doske FRDM-K64. Rezistor R3 (10 Ohm) slúži na meranie odoberanéhi prúdu cez USB port.

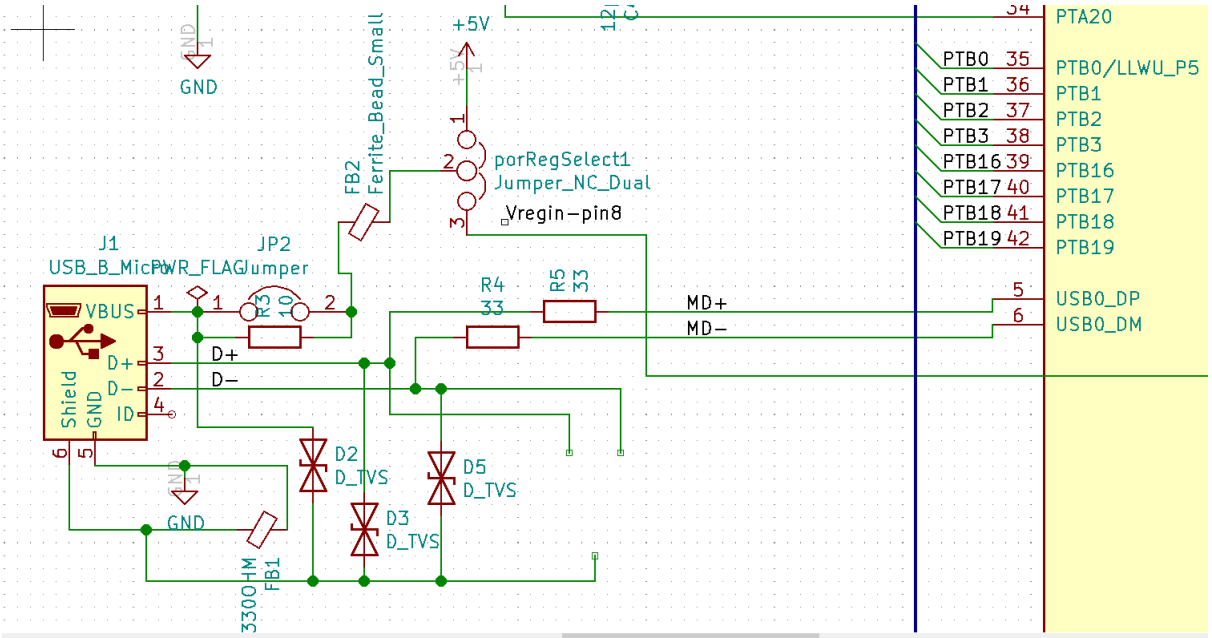
Transily DTVS slúžia na ochranu pred elektrostatickým napätím. FB2 tlmivka slúži ako ochrana pred nárazovými prúdmi. R3 a R4 (33 Ohm) upravujú impedanciu datovej zbernice USB. Spôsob vedenia diferenciálneho páru datovej zbernice je ukázaný na obr2.6.

2.3 Implementácia USB periférie v kóde

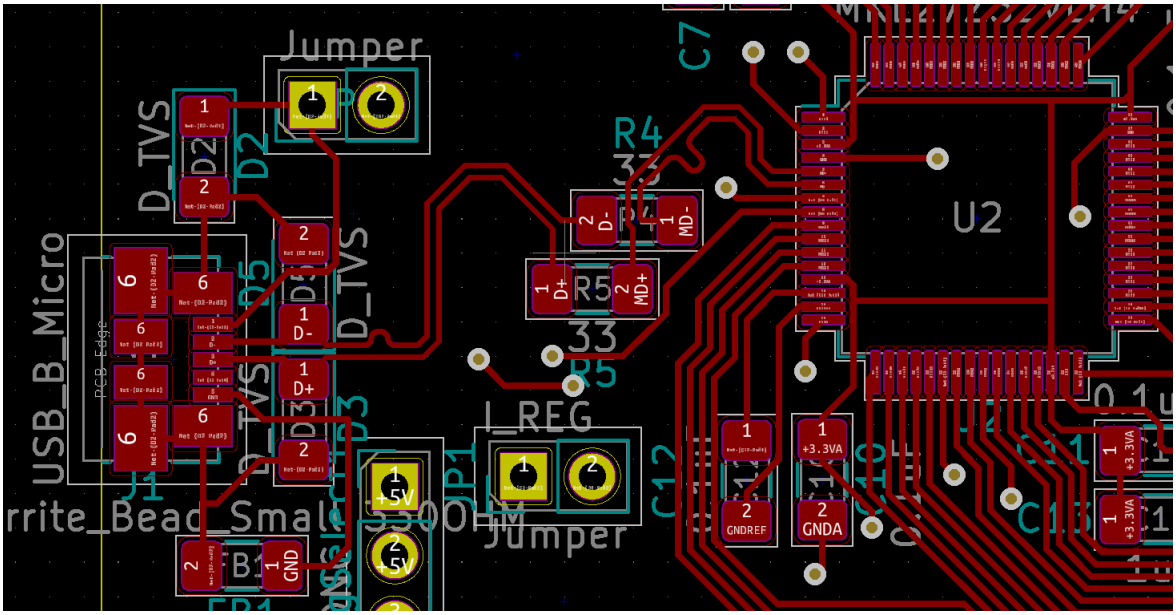
Pri práci s vývojovou doskou FRDM-K64F bolo použité natývne vývojové prostredie od NXP, MCUXpresso. Toto vývojové prostredie obsahuje nástroje na konfiguráciu Obr.2.7 periférnych zariadení v MCU, jeho vstupom sú nastavenia užívateľom cez grafické rozhranie a výstupom sú vygenerované súbory .c/.h. Tieto súbory sú v stave obsahujúci základnú funkcionalitu a ďalším krokom je ich úprava na uživatelom vyžadovanú funkcionalitu.

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, ktoré implementuje viac než jednu triedu. Takéto zariadenie pozostáva z niekoľkých 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.

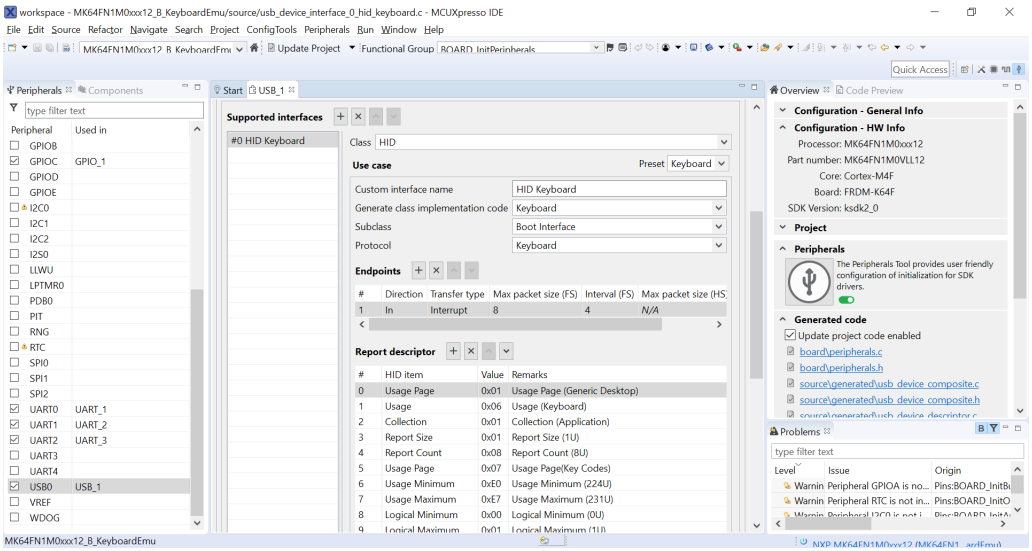


Obr. 2.5: Obvod USB rozhrania



Obr. 2.6: Vedenie usb signálu[3]

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 s 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.



Obr. 2.7: konfiguračný nástroj v IDE MCUXpresso

Konfiguračné nástroje poskytnú uživaťelovi začiatočný kód, ten je generovaný tak že USB zariadenie sa enumeruje a každé jeho USB rozhranie funguje tak, ako ukážkový projekt na ktorom je založené. Rozhranie klávesnice posiela opakovane PGUP a PGDWN kódy cez rozhranie a VCOM rozhrania posielajú späť znaky obdržané. Výsledok generovania kódu je zariadenie pozostávajúce z nezávislých rozhraní.

2.3.1 štruktúra generovaného kódu

Kód generovaný konfiguračným nástrojom je rozdelený medzi všeobecnú obsluhu USB rozhrania a súbory jednotlivých rozhraní. Súbory nachádzajúce sa v zložke source/generated slúžia na všeobecnú obsluhu USB rozhrania. Obsah týchto súborov je ovplyvnený jednotlivými nastaveniami v rozhraniach.

Každé USB rozhranie má vlastné súbory[4], tie sa nachádzajú priamo v zložke source.

Súbory usb_device_decsriptor, obsahujú descritpory a funkcie na ich získanie.

Súbor kompozitného zariadenia

Tento súbor obsahuje kód, ktorý pre svoju funkciu volá funkcie jednotlivých rozhraní a k tým má prístup ako k externým funkciám. Tieto funkcie sú:

- USB_DeviceInterface[...]Init
- USB_DeviceInterface[...]Callback

[4]Výnimkiou je DIC rozhranie použité pre implementáciu VCOM, ono zdiela súbor z rozhraním CIC

- USB_DeviceInterface[...]SetConfiguration
- USB_DeviceInterface[...]SetInterface

Tento súbor definuje štruktúru g_UsbDeviceComposite, tá obsahuje nasledovné polia: usb_device_handle deviceHandle - obsahuje údaje o vyrovnávacích pamätiach používaných pre USB prenosy class_handle_t interface[...]Handle - odkaz na štruktúru každého jednotlivého rozhrania uint8_t currentConfiguration - číslo momentálnej USB konfigurácie uint8_t speed - kód nastavenej rýchlosti uint8_t attach - stav pripojenia zariadenia

V tomto súbore sú implementované funkcie:

- void USB0_IRQHandler(void) - Jedná sa o rutinu prerušenia ktorá funkciu USB driveru na obsluhu prerušenia
- void USB_DeviceIsrEnable(void)- Nastavuje USB modul a jeho prerušenie
- static usb_status_t USB_DeviceCallback(usb_device_handle handle, uint32_t event, void *param) - Tento callback je volaný nižžou vrstvou obsluhy USB modulu parameter event obsahuje výčtovú hodnotu udalosti ktorá na USB ropzhraní nastala, Táto funkcia implementuje obsluhu jednotlivých udalostí. Takouto udalosťou môže byť požiadavka na descriptor, reset zbernice, alebo uspanie zbernice
- usb_status_t USB_UpdateInterfaceSetting(uint8_t interface, uint8_t alternateSetting) - Pre dané USB rozhranie nastavý jeho alternatívne nastavenie[5].
- void USB_DeviceApplicationInit(void) - inicializácia zariadenia, jednotlivých rozhraní, alokácia vyrovnávacích pamätí.
- void USB_DeviceTasks(void) - Volanie obslužných callbackov rozhraní, obsluha niektorých typov rozhraní je volaná z tejto funkcie. Niektoré typy rozhraní majú svoju callback rutinu zaregistrovanú pre priame volanie USB stackom.

generovaný kód rozhrania

Obsluha USB rozhraní všeobecne prebieha pomocou nasledovných fukcií:

- USB_DeviceInterface[...]Init - Inicializuje statickú referenciu na kompozitné zariadenie a svoj ukazateľ na vyrovnáciu pamäť,
- USB_DeviceInterface[...]Action - plní samotnú funkciu zariadenia, volá funkciu ktorá slúži na odoslanie dát cez USB rozhranie.
- USB_DeviceInterface[...]Callback - IDentifikuje a obsluhuje udalosti rozhrania, Volá funckiu USB_DeviceInterface[...]Action
- USB_DeviceInterface[...]SetConfiguration - V prípade že existuje požadovaná konfigurácia, volá funkciu USB_DeviceInterface[...]Action. Inak vráti chybový

[5]Zariadenie môže mať niekoľko nastavení, ktoré definujú endpointy a ich vlastnosti rozlične.

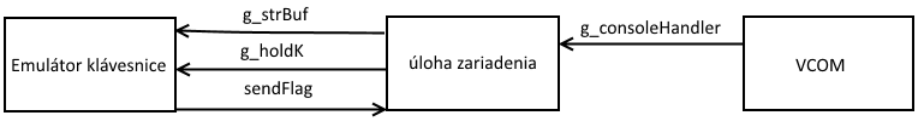
kód.

- USB_DeviceInterface[...]SetInterface - volá funkciu USB_DeviceInterface[...]Action a vracia kód chyby ako návratovú hodnotu

Vo vygenerovanom kóde volanie USB_DeviceInterface[...]Action slúži na upozornenie nižších vrstiev o prenose a udržania periodického posielania dát.

2.3.2 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.2.13.



Obr. 2.8: Zloženie zariadenia

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

2.3.3 Implementácia rozhrania klávesnice

Šablónu pre protokol klávesnice tvoril sám autor. Tieto šablóny sa vonkajšou funkcionalitou odvíjajú od svojich predchodcov, SDK ukážokových projektov. Na rozdiel 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.

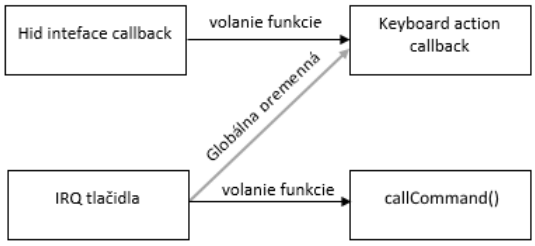
Vygenerovaný kód pre USB klávesnicu je schopný enumerácie a posiela v cykloch striedavo kód stlačenia dvoch kláves, PAGEUP a PAGEDOWN. Tento kód bol následne prepísaný na prevádzanie dát určených na poslanie na klávesové kódy a ich postupné posielanie v packetoch po jednom. Najväčšie zmeny sú urobené vo funkcií static usb_status_t USB_DeviceHidKeyboardAction(void), tam je nutné nahradiť kód ukážky cieleným kódom. Implementácia USB rozhrania sa skladá z nasledovných častí:

- USB driver,
- USB stack[6],
- generovaný kód,
- časť generovaného kódu prepísaná uživatelom.

Obr. 2.9 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.

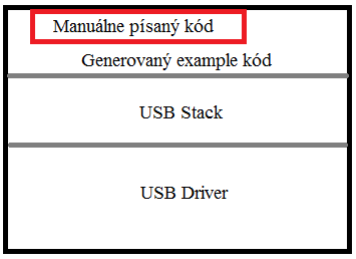
Do reťazca g_StrBuf prepisuje hodnoty určené na poslanie funckia volaná úlohou zariadenia. Sprosredkovanie úlohy zariadenia je volané funkciou callCommand().



Obr. 2.9: schéma vzájomného volania kódu

Obr. 2.10 ilustruje spôsob akým sa tvoria zariadenia pomocou konfiguračných nástrojov, rovnaký postub je použitý pre rozhrania USB klávesnice a VCOM.

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



Obr. 2.10: Štruktúra implementácie kódu

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

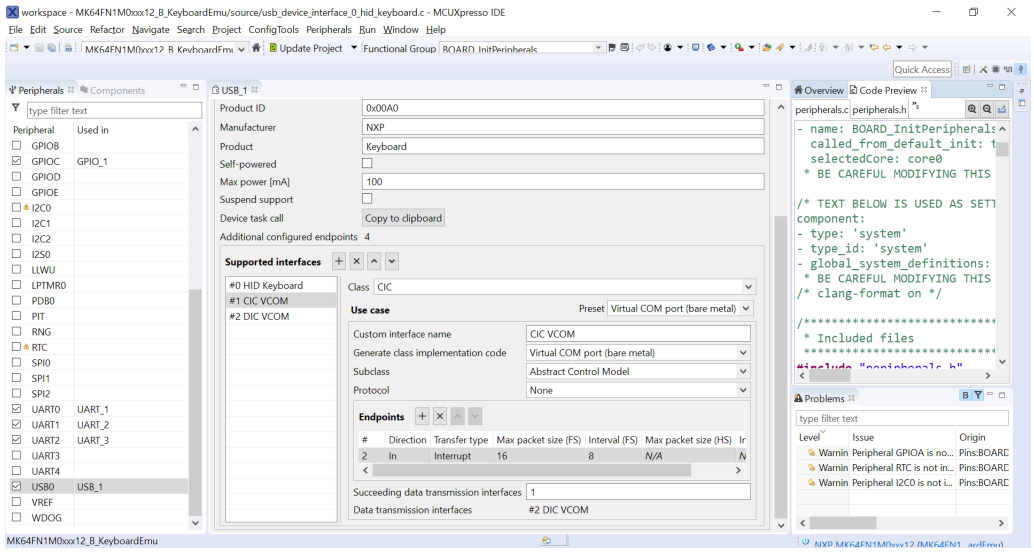
```
static usb_status_t USB_DeviceHidKeyboardAction(void)
{
    static int x = 0U;// pozicia v retazci
    s_UsbDeviceHidKeyboard.buffer[2] = 0x00U;
    if(g_StrBuf[x]=='\0')// posielanie dokoncene.
    {
        for(int i = 0;i<20;i++)
        {
            g_StrBuf[i] = '\0';
        }
        x = 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);
```

```
27 }
```

Výpis 2.1: Callback úlohy klávesnice

2.3.4 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.

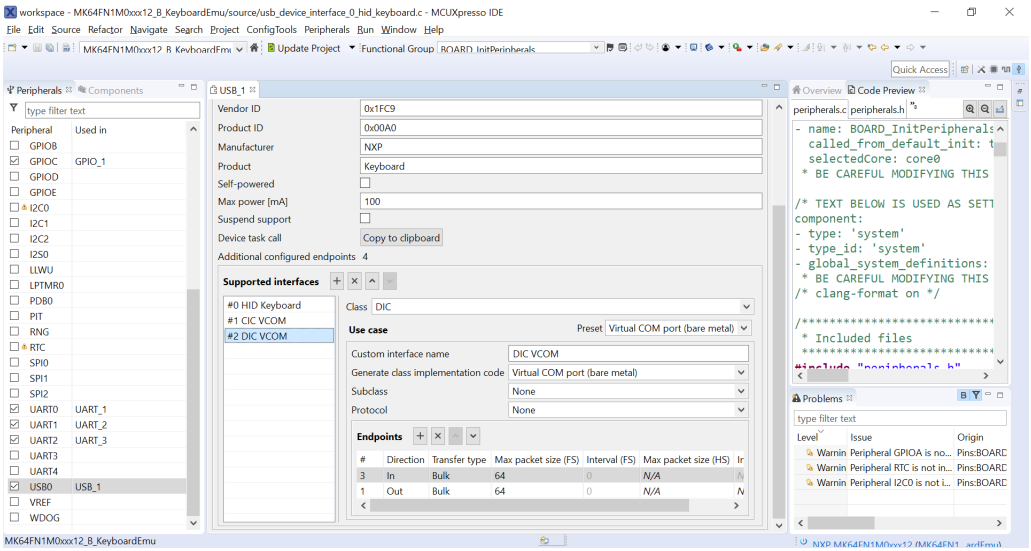


Obr. 2.11: Nastavenia rozhrania CIC

Vygenerovaný kód pre VCOM je schopný enumerácie, pripojenia sa cez konzolový software (napr. PuTTY), príjmať znaky a zároveň ich posielať späť. Takto sa javí 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 spočíva v posielaní vstupných znakov na konzolový výstup a toto ich zobrazí. Ku skladaniu reťazca zloženého príkazu z príchozích znakov sa finalizuje zaznamenaním znaku nového riadku. Na výslednom reťazci je nutné vykonávať operácie príslušné špeciálnymi znakmi, alebo ich ignorovanie.



Obr. 2.12: Nastavenia rozhrania DIC

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

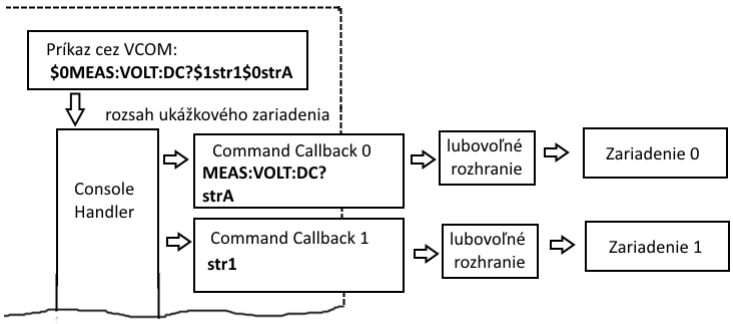
```
void USB_DeviceInterface1CicVcomTask(void)
{
    // inicializacia ukazatelu na retazec zo vstupom
    if(g_consoleHandler.inputStr == NULL)
    {
        g_consoleHandler.inputStr = s_currRecvBuf;
    }
    // navratova hodnota
    usb_status_t error = kStatus_USB_Error;
    /* kontrola pripojenia 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
```

```
22              for (i = 0; /* i < s_recvSize */; i++)
23              {
24                  // dosiahol sa koniec
25                  if(s_currRecvBuf[i] == '\0')
26                  {
27                      break;
28                  }
29                  // osetrenie specialnych znakov
30                  i = resolveChar(i);
31                  // zaistenie preposlania
32                  s_currSendBuf[s_sendSize++] = s_currRecvBuf[i];
33                  // 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  }
```

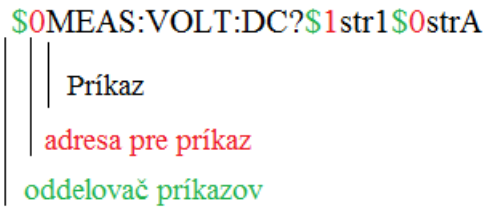
Výpis 2.2: Callback úlohy virtuálnej konzole

2.3.5 Implementácia úlohy zadania



Obr. 2.13: výstupný komunikačný model

Obr. 2.13 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 základe adresy jednotlivého príkazu, pritom je tejto callback funkcií sprístupnený jej prislúchajúci príkaz.



Obr. 2.14: 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 a môže použiť tieto dve statické premenné na prístup k nej.

Kód ktorý rozlišuje adresy príkazov 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 nedokončenej operácie a zaistiť, že další uživatelský callback sa volá až po dokončení výstupu dát aktuálneho.

```
static void commandCallbacks()
{
    int i = 0;
    switch (address) {
        case '0': // callback pre adresu 0
            if (!sendFlag)
            {
                // zabranenie klavesnicoveho vystupu
                g_holdk = 1;

                // uživatelský kód
                getSData(g_StrBuf);

                // synchronizácia
                g_holdk = 0; // data pripravene, cas odosielat
                // zablokovanie uzivatelskych callbackov
                sendFlag = 1;
                /* sendFlag je nulovany v callback
                 funkcii klavesnice
                 po tom co je odosielanie hotove */
            }
```

```
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 prostriekami vyvýjanými výrobcami mikrokontrolérov. Štruktúra ktorú tvorý driver a USB stack posúvajú úlohu uživaťela mikrokontroléra do vyšších abstraktných vrstiev a riešia problematiku harwaru. Snahy výrobcov zvýšiť jednoduchosť a znížiť vývojové náklady na ich platforme vedie ku konfiguračným nástrojom, ktoré poskytujú uživateľovi lepší začiatočný bod vo vývoji ako ukážkové projekty. Dobre implementované konfiguračné nástroje odstraňujú nutnosť podrobnej znalosti konkrétneho prevedenia periférií, hodinových sytémov a napájania vývodov. Tieto výdobytky sú zároveň schopné nahradiť značnú časť práce z dokumentáciou mikrokontrolérov. Spolu z vlastnosťami ako bezkryštálový provoz USB modulu, zabudovaný modul pre napájanie cez USB, detekcia nabíjania a rýchlosťami podporovanými dostupnosť a kvalita konfiguračných nástrojov sa stávajú rozhodujúcim faktorom vo výbere platformy pre tvorbu USB rozhrania.

V porovnaní nástrojov popredných výrobcov ako STMicroelectronics, microchip a NXP, práve nástroje od NXP ponúkajú najväčšiu flexibilitu v práci z USB rozhraním. Konfiguračné nástroje od firmy NXP umožňujú generovať kompozitné zariadenie lubovolných typov rozhraní bez obmedzení, zatiaľ čo u ostatných výrobcov je možné použiť len fixne predpripravené ukážkové projekty.Tent postup vývoja z SDK ukážok sa dá považovať dnes už za zastaralý. Spôsob vývoja z ukážkové projektov neponúka flexibilitu v ich rozširovaní o iné triedy implementované v separátnych projektoch.

Z požiadaviek 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.) A až následne preniesť vývoj na cielený mikrokontrolér a vývoj dosky plošného spoja.

Ukážkové zariadenie v tejto práci sa po pripojení na USB zbernicu enumeruje ako kompozitné zariadenie. Po pripojení je možné vidieť jeho rozhrania ako klávesnicu a virtuálnu konzolu COMX[1]. Na rozhranie konzoly je možné pripojiť sa programmi ako teraterm alebo PuTTY. Po pripojení je možné zadať reťazec príkazov, ktoré majú byť v prípade stlačťenia tlačida vykonané. Po tom čo bol príkaz zadaný, stlačenie tlačidla spôsobíF zapisovanie hodnôt cez rozhranie USB klávesnice. Počet jednoltlivých hodnôt závisý od počtu príkazov a každá hodnota je oddelená

[1]Kde X je číslo konzoly a to závisý od USB portu.

kódom stlačenia tabulátoru, za poslednou hodnotou nasleduje kód stlačenie klávesy enter.

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 niekoľkých 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 netypyckého 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.

[5] *FRDM-K64F Freedom Module User's Guide* [online]. Document Number: FRDMK64FUG Rev. 1,08/2016 NXP Semiconductors URL: <https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kineti FRDM-K64F>.

[6] *USB 2.0 for Kinetis MCUs* [online]. URL: <https://www.nxp.com/docs/en/supporting-information/Universal-Serial-Bus-Training.pdf>.

[7] *Grainuum USB* [online]. URL: <https://github.com/xobs/grainuum>.

[8] *V-USB* [online]. URL: <https://www.obdev.at/products/vusb/index.html>.

[9] *USBtiny* [online]. URL: <https://dicks.home.xs4all.nl/avr/usbtiny/>.

[10] *USB2.0 Technical Manual* [online]. TDK Corporation Magnetics Business Group Mitsuharu Mizutani URL: <https://product.tdk.com/en/products/emc/guidebook/eemc_practice_04.pdf>.

[11] *USB implementers forum wikipedia page* [online]. URL: <https://en.wikipedia.org/wiki/USB_Implementers_Forum>.

[12] *Device Firmware Upgrade 1.1* [online]. URL: <https://www.usb.org/sites/default/files/DFU_1.1.pdf>.

[13] *DFU util homepage* [online]. URL: <http://dfu-util.sourceforge.net/>.

[14] *USB compliance checklist* [online]. July 19, 1999 Motorola URL: <https://www.nxp.com/docs/en/supporting-information/MPC823SI2.pdf>.

[15] *ARM Cortex-M* [online]. URL: <https://en.wikipedia.org/wiki/ARM_Cortex-M>.

[16] *NXP product selector* [online]. URL: <https://www.nxp.com/parametricSearch#/category/c731_c1770>.

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
VID	Vendor Idetnification Number – kód výrobcu
PID	Product Identification Number – kód produktu
MCU	Microcontroller Unit – Jednotka mikrokontroléru
SMD	Surface Mount Device – zariadenie s povrchovou montážou
SDK	Software Development kit – Balík vývoja softwareu
BCD	Battery Charger Detect – Detekcia nabíjania batérií
DCD	Device Charger Detect – Detekcia nabíjania zariadenia
OTG	On-the-Go – označenie pre zariadenie schopné dynamicky fungovať ako Host alebo zariadenie
SOF	Start Of Frame – Začiatok rámca
SMD	Durface mount device – zariadenie z povrchovou montážou
RTC	Real time clock – hodiny skutočného času

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].