VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

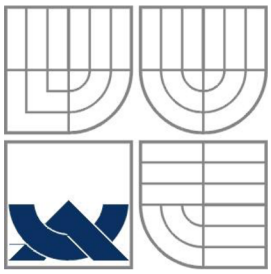
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

DIGITIZÉR AUDIOSIGNÁLU SE ZÁZNAMEM NA SD KARTU

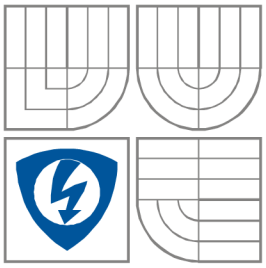
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Jan Harsa
AUTHOR

BRNO, 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

DIGITIZÉR AUDIOSIGNÁLU SE ZÁZNAMEM NA SD KARTU

AUDIO DIGITIZER USING RECORDING TO SD CARD

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JAN HARSA
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. JIŘÍ ŠEBESTA, Ph.D.
SUPERVISOR

BRNO, 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Diplomová práce

magisterský navazující studijní obor
Elektronika a sdělovací technika

Student:	Bc. Jan Harsa	***ID:***	106455
Ročník:	2	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Digitizér audiosignálu se záznamem na SD kartu

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s možnostmi digitalizace audiosignálu pomocí mikrokontroléru s ukládáním na SD kartu. Vyberte vhodný mikrokontrolér a navrhněte blokové schéma digitizéru audiosignálu ovládaného z PC pomocí USB rozhraní včetně možnosti přístupu na SD kartu z PC. Navrhněte kompletní schéma zapojení digitizéru audiosignálu a desku plošných spojů. Digitizér osaďte a proveďte jeho základní oživení. Proveďte rozbor firmware řídicího mikrokontroléru. Sestavte a odlaďte firmware řídicího mikrokontroléru a ověřte plnou funkčnost digitizéru audiosignálu a stanovte jeho parametry.

DOPORUČENÁ LITERATURA:

[1] FRÝZA, T. Mikroprocesorová technika. Elektronické skriptum. Brno: FEKT, VUT v Brně, 2008.

[2] FOUST, F. Secure Digital Card Interface for the MSP430. Application Note [online]. East Lansing: Michigen State University, 2004.- [cit. 10.1.2008]. Dostupné na www: http://alumni.cs.ucr.edu/~amitra/sdcard/Additional/sdcard_appnote_foust.pdf

Termín zadání:	11.2.2013	***Termín odevzdání:***	24.5.2013

Vedoucí práce: doc. Ing. Jiří Šebesta, Ph.D.
Konzultanti diplomové práce:

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práce pojednává o návrhu digitizéru, který je součástí monitorovacího přijímače pro letecké pásmo. Požadavek je dvoukanálová digitalizace audio signálů. Zařízení komunikuje s PC pomocí USB rozhraní. Dále umožňuje čtení a zápis dat na SD kartu. Pro návrh a testování funkce je použita vývojová deska STM32F4DISCOVERY s procesorem STM32F407VGT6. Tento kit doplňují další periferie na jiné desce (vstupní audio obvody a SD sloty). V práci je stručně rozebrána teorie k jednotlivým oblastem, s nimiž projekt souvisí. Dále je zde vysvětlen návrh hardwaru, popsán vývoj softwaru pro ovládání zařízení a největší pozornost je věnována popisu firmwaru pro MCU, který řídí AD převod, formátování zvukového signálu, USB komunikaci HID a Mass Storage, zápis a čtení z SD karty a další doplňující periferie.

KLÍČOVÁ SLOVA

vývojová deska STM32F4-Discovery, ARM procesor, SD karta, USB, HID, Mass Storage, A/D převod

ABSTRACT

This thesis deals about a digitizer design. The digitizer is part of air-band monitoring receiver. Requirement is digitalization of two audio signals. The device communicates with PC through USB interface and it provides reading and writing to SD card. The STM32F4DISCOVERY development board with processor STM32F407VGT6 was used for design and function testing. This development kit is supplemented with other peripherals on an extern board (input audio circuits and SD slots). The thesis describes briefly theory for each issue which this project deals with. One part is engaged to the hardware design. Then there is a description of the PC software for the device controlling. The main part of the thesis is about the development of the firmware for MCU, which manages AD conversion, formatting of the voice signal, USB communication (HID and Mass Storage class), recording and reading from the SD card and additional peripherals.

KEYWORDS

Development board STM32F4-Discovery, ARM processor, SD card, USB, HID, Mass Storage, AD conversion

HARSA, J. *Digitizér audiosignálu se záznamem na SD kartu*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav radioelektroniky, 2012. 46 s., 10 s. příloh. Diplomová práce. Vedoucí práce: doc. Ing. Jiří Šebesta, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svoji diplomovou práci na téma Digitizér audiosignálu se záznamem na SD kartu jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a~jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce doc. Ing. Jiřímu Šebestovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne ....

(podpis autora)

Výzkum realizovaný v rámci této diplomové práce byl finančně podpořen projektem CZ.1.07/2.3.00/20.0007 **Wireless Communication Teams** operačního programu **Vzdělávání pro konkurenceschopnost**.

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Finanční podpora byla poskytnuta Evropským sociálním fondem a státním rozpočtem České republiky.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 ÚVOD

Řešená aplikace je digitální část monitorovacího přijímače pro letecké pásmo (108 až 137 MHz). Přijímací vysokofrekvenční obvody přijímače zde nejsou řešeny. Nicméně tyto VF obvody (PLL atd.) jsou také řízeny digitální částí. Toto muselo být bráno v úvahu při návrhu zařízení – řídící mikroprocesor (MCU) nemohl být zcela vytížen, a to jak po stránce využití jeho vývodů, tak co se týče využití paměti programu.

Hlavní požadavky na zařízení jsou digitalizovat dva řečové audiosignály ve vhodném rozlišení, ukládat digitalizované sekvence na SD (Secure Digital) kartu ve vhodném formátu s podporou protokolu FAT (File Allocation Table) a poskytovat komunikaci s PC přes rozhraní USB (pro ovládání zařízení z PC) a komunikaci s SD kartou.

Z uvedeného plynou požadavky na MCU. Musí být ovladatelný z PC pomocí vhodného programu. Zařízení by se pro tuto komunikaci mělo chovat jako USB-HID (USB Human Interface Device). Tato komunikace by měla zprostředkovávat uživateli volby pro nahrávání audio sekvencí, což by mělo umožňovat zařízení i bez připojení k PC. Dále MCU musí digitalizovat dva audio kanály, formátovat digitalizované signály a ukládat je na SD kartu s podporou souborového systému. Tyto data musí MCU také číst, popř. přehrát. MCU musí také řídit obousměrný komunikační kanál PC ↔ SD karta.

2 MIKROKONTROLÉR PRO DIGITIZÉR

V této kapitole je diskutován výběr mikroprocesoru a stručně popsány vlastnosti konkrétního vybraného MCU. Dále je zde prezentována vývojová deska STM32F4-Discovery osazená vybraným procesorem.

Při výběru procesoru pro danou aplikaci je nutné brát ohled na jeho výkonnostní požadavky a další požadavky týkající se periferií, počtu pinů, atd. Je zřejmé, že výběr byl omezen na procesory architektury ARM (Advanced RISC Machine). Tato architektura nabízí mnohem vyšší výkon než běžné 8-bitové nebo 16-bitové procesory. Pokud přihlížíme k ceně, ARM procesory jsou jen o málo dražší než běžné RISC procesory. Dalším důležitým aspektem při volbě MCU bylo dostupné vývojové prostředí. Mnohé softwarové nástroje pro vývoj aplikací pro procesory s architekturou ARM jsou omezeny délkou kódu nebo časově. Plné verze vývojového prostředí jsou velmi drahé.

Na základě uvedených požadavků a omezení byl vybrán procesor STM32F407VGT6 od fy ST Microelectronics, který je také dostupný s vývojovou deskou STM32F4-Discovery. Návrh a vývoj zařízení byl postaven na práci právě s touto deskou. Pro zvolený procesor je k dispozici volně dostupné vývojové prostředí Atollic TrueSTUDIO for STM32.[1][2]

Poznámka: Během období vývoje tohoto zařízení firma Atollic změnila parametry volně dostupné (lite) verze vývojového prostředí, a to v omezení délky kódu na 32kB. Toto omezení využívá většina firem produkující vývojové prostředí pro ARM procesory. Vývoj pokračoval s prostředím µVision.

2.1 Procesor STM32F407VGT6

Zvolený MCU představila firma ST Microelectronics v září roku 2011. Obsahuje v současnosti nejnovější jádro ARM architektury Cortex-M4F s jednotkou FPU (Floating Point Unit) pro výpočty s plovoucí desetinnou čárkou. Jádro pracuje s maximálním hodinovým taktem 168MHz. V kombinaci s ART (Adaptive Real Time) akcelerátorem nabízí MCU velký početní výkon. MCU podporuje instrukční sadu Thumb-2 a DSP instrukce, což je velmi výhodné při implementaci algoritmů jako MP3, FLAC a dalších kompresních algoritmů pro zpracování zvuku.[3]

Mikroprocesory řady STM32F4 obsahují kromě nového výkonného jádra také větší paměť a celu řadu periferií. Stručný přehled zejména těch vlastností a periferií, které budou využité při navrhování digitizéru, jsou uvedené v následující tabulce (tab. 2.1).

Tab. 2.1: Přehled vlastností a periferií procesoru.[3]

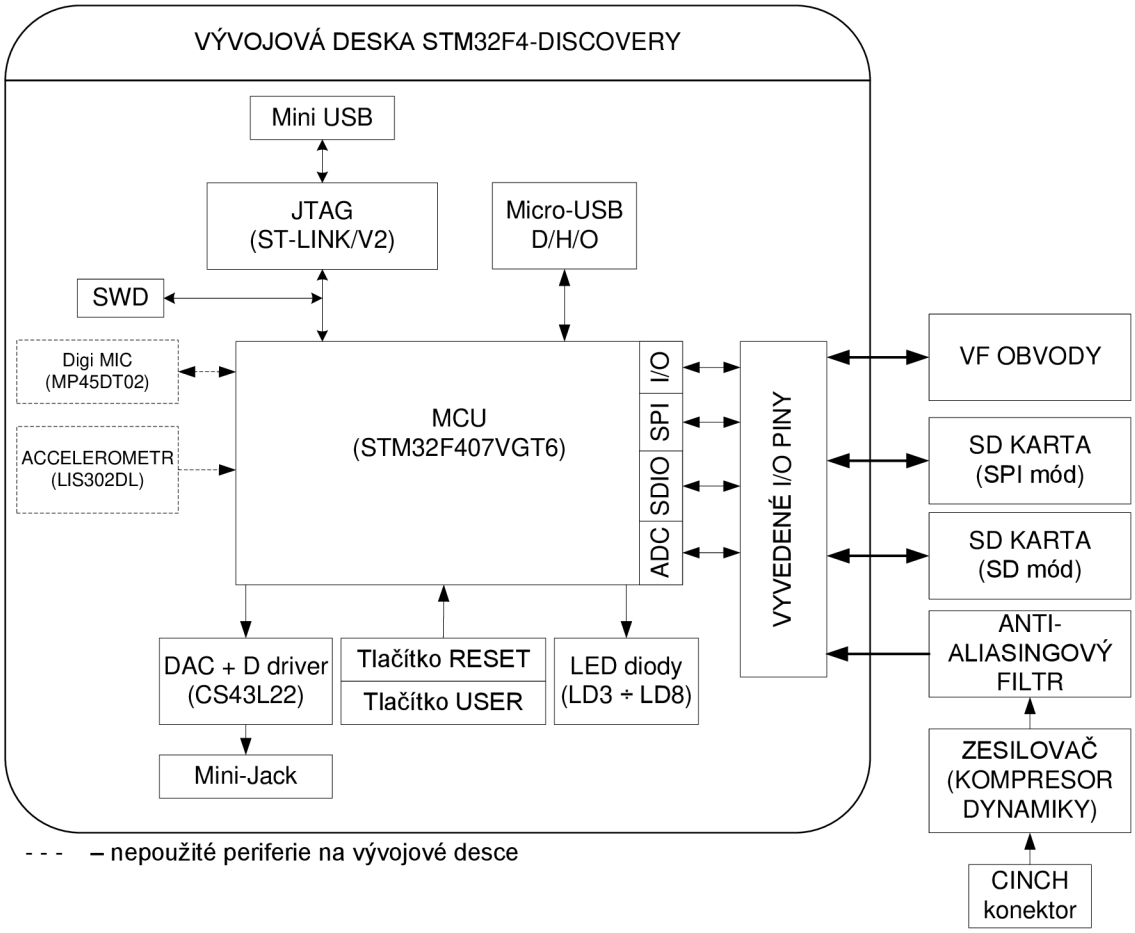
Flash paměť	1 MB
Paměť SRAM	196 kB
SPI	3x
SD/SDIO/MMC Interface	Specifikace verze 2.0
ADC	3x 12-bitový (2,4 MS/s)
USB	OTG (On-The-Go)
DMA	16-streams
Debug módy	SWD, JTAG

2.2 Vývojová deska STM32F4-Discovery

Kompletní návrh a realizace celého zařízení byla provedena na vývojové desce STM32F4-Discovery doplněné potřebnými periferiemi. Deska obsahuje jen několik základních periferií, z nichž byl využit programátor ST-LINK verze 2 připojený přes mini USB, uživatelské micro USB On-The-Go, tlačítka, LED diody a DA převodník pro přehrávání zvuku. Ostatní komponenty zařízení budou připojeny k desce pomocí kolíků, na které jsou vyvedené piny MCU.

Doplňujícími komponenty jsou vstupní audio obvody a SD sloty. Před vstupem AD převodníku musí být audio signál nejprve upraven. K tomu slouží zesilovač a anti-aliasingový filtr (v dalším vývoji je možné implementovat i kompresor dynamiky, který zlepší automatické vyrovnávání citlivosti). Vše je navrženo pro dva audio kanály. Další periferie jsou dva SD sloty. Jeden připojený k MCU přes SPI a druhý přes SDIO rozhraní. Takto bylo zvoleno pro možnost realizace obou typů komunikace s SD kartou (viz kap. 5.1).

Zařízení musí také ovládat VF obvody monitorovacího přijímače. Řízení těchto obvodů není součástí této práce. Na obr. 2.1 je blokové schéma popisující komponenty dostupné na vývojové desce a doplňující periferie.



Obr. 2.1: Blokové schéma zahrnující komponenty vývojové desky a doplňujících obvodů.

Význam zkratek v blokovém schématu (obr. 2.1):

SPI – Serial Peripherial Interface (Sériové rozhraní)
SDIO – Secure Digital Input/Output (rozhraní pro komunikaci v SD módu)
ADC – Analogově-digitální převodník
DAC – Digitálně-analogový převodník
SWD – Serial Wire Debug
D/H/O – Device/Host/On-The-Go

3 A/D PŘEVOD

3.1 Vstupní obvody

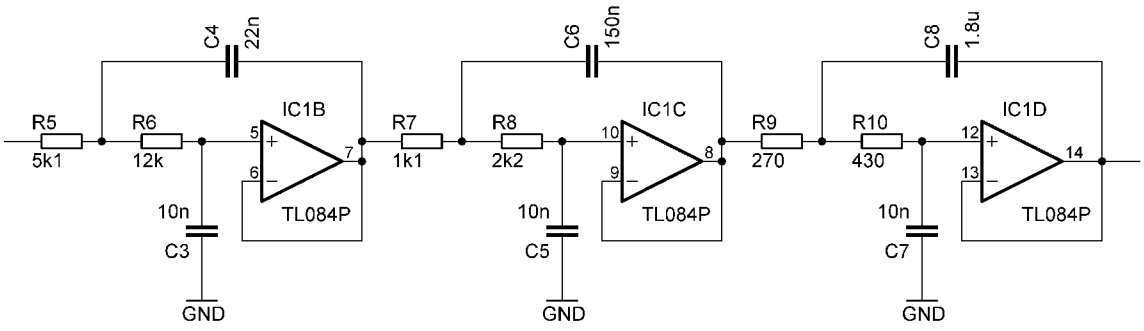
Obvody upravující analogový audio signál před samotným AD převodem se skládají ze zesilovače, anti-aliasingového filtru, ochranných prvků a obvodů pro úpravu stejnosměrné složky.

3.1.1 Anti-aliasingový filtr

Nejvhodnější typ anti-aliasingového filtru je integrovaný obvod využívající technologii spínaných kapacitorů. Hlavní výhody oproti klasickým filtrům jsou možnost ladění, jednoduchost výsledného zapojení a velmi dobré vlastnosti – vysoký řád filtru, velký útlum v nepropustném pásmu, atd. Mezi tyto obvody patří např. obvod LTC1569-7 od fy Linear Technology. Bohužel není dostupný příliš velký výběr těchto obvodů a jejich cena je poměrně vysoká. Klasické aktivní filtry mohou být také použity jako anti-aliasingové. Výhodou oproti integrované variantě je nízká cena a možnost vlastního návrhu „na míru“. Tento typ byl zvolen pro filtr v digitizéru.

Je doporučené používat v případě klasických aktivních filtrů nekaskádní strukturu, protože se neuplatňuje napěťová nesymetrie operačních zesilovačů. Pokud je ovšem v aplikaci brán zřetel pouze na střídavou složku, posunutí offsetu nevadí a může být použita i kaskádní struktura. Stejnosměrné oddělení je ošetřeno kondezátorem na výstupu filtru. Filtr byl navržen na mezní frekvenci 3,4 kHz, což je běžná maximální frekvence pro systémy zpracovávající řečový signál.

Pro návrh byl použit software FilterPro od fy Texas Instruments [7]. Byl vybrán aktivní filtr 6. řádu topologie Sallen-Key. Protože analogový signál je nekvalitní řečový signál, není nutné dosáhnout hladké frekvenční charakteristiky v propustném pásmu, a proto může být použita Chebyshevova aproximace (konkrétně byla zvolena aproximace Chebyshev 0,5dB). Schéma navrženého filtru je na obr. 3.1.

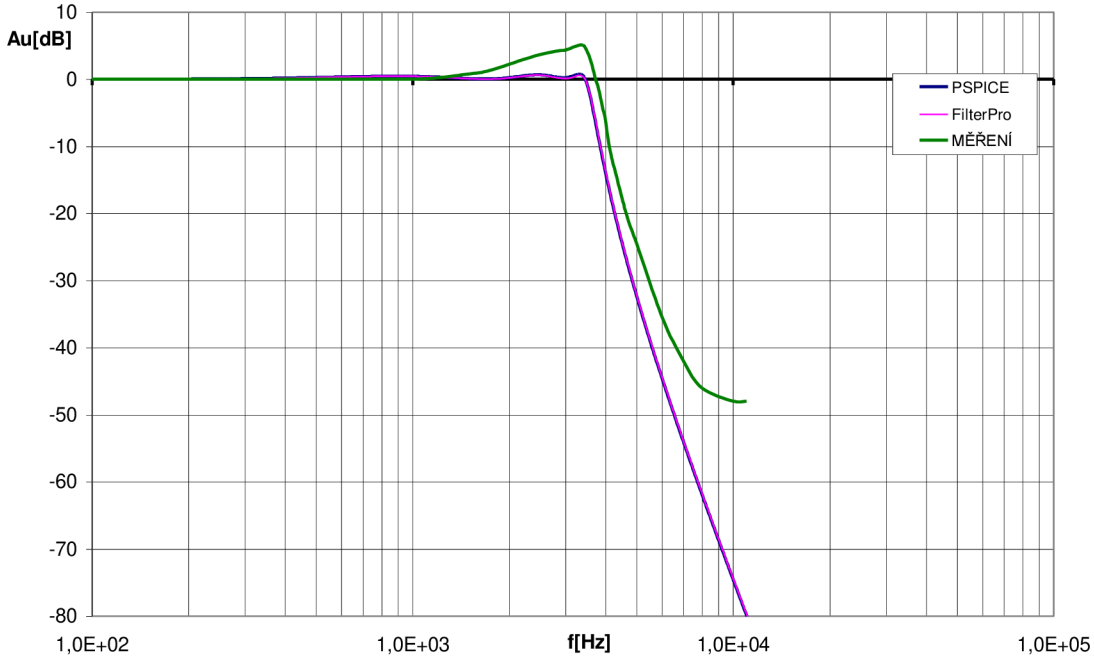


Obr. 3.1: Schéma anti-aliasingového filtru.

Filtr byl simulován v programu PSPICE. Frekvenční charakteristika je totožná s výsledkem z programu FilterPro. Následující graf (obr. 3.2) prezentuje kmitočtové

charakteristiky získané simulacemi a měřením filtru. Naměřená charakteristika se trochu liší od simulovaných. Hodnoty na vyšších frekvencích, kde má filtr velký útlum, byly dostupnými přístroji neměřitelné. V této oblasti má velký vliv na měření také šum.

Podle simulace je dynamika filtru s nesymetrickým napájecím napětím 5V je dostačující pro maximální rozkmit signálu na vstupu AD převodníku (0 až V_{REF} = 3V). Toto bohužel nebylo potvrzeno měřením. Výstupní signál je značně zkreslen už při V_{pp} = 1 V. Je tedy nutné použít buď vyšší napájecí napětí, nebo použít jiný typ OZ.



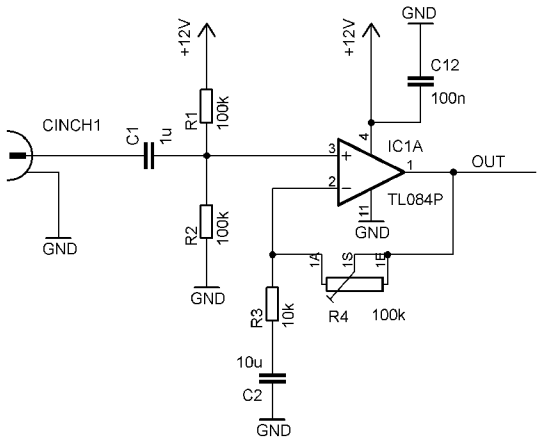
Obr. 3.2: Simulované a změřené frekvenční charakteristiky anti-aliasingového filtru.

3.1.2 Kompletní zapojení vstupních audio obvodů

Před anti-aliasingovým filtrem musí být zesilovač, který upraví velikost audio signálu tak, aby byla vhodná pro vstup AD převodníku v MCU. Ten vyžaduje vstupní napětí v rozsahu 0V až V_{REF}. Hodnota referenčního napětí je rovna napájecímu napětí MCU, což jsou 3V (na stabilizátoru na vývojové desce bylo naměřeno 2,92V). Zesilovač je realizován jako neinvertující s operačním zesilovačem TL084 doplněný kapacitorem tak, aby nezesiloval stejnosměrnou složku signálu.

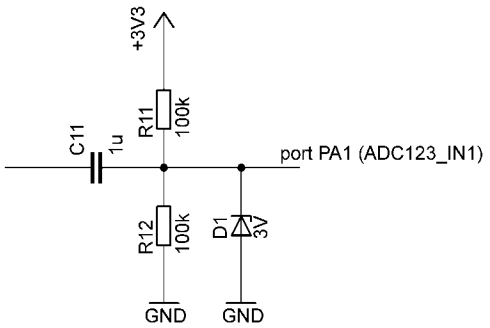
Napájecí napětí pro OZ je z externího zdroje (kvůli zkreslení filtru při malém napájecím napětí). Tento nedostatek není velkou závadou, protože filtr bude použit pouze pro vývoj zařízení (tudíž externí zdroj není na závadu). Ve výsledném zařízení signál už dostatečně filtrován ve vstupních obvodech monitorovacího přijímače.

Jelikož je napětí nesymetrické, není možné zpracovávat signál jako střídavý. Před zpracováním signálu (zesílením a filtrací) je tedy posunuta DC složka na polovinu napájecího napětí pomocí odporového děliče. Zapojení zesilovače a děliče napětí je na následujícím obrázku.



Obr. 3.3: Zapojení neinvertujícího zesilovače s TL084.

Za zesilovačem následuje výše diskutovaný filtr. Před vstupem AD převodníku je upravena stejnosměrná složka na $V_{REF}/2$, tedy přibližně na 1,5 V. Vstup ADC je ještě doplněn ochrannou Zenerovou diodou, jak je vidět na následujícím zapojení.



Obr. 3.4: Zapojení obvodu před vstupem AD převodníku mikrokontroléru.

Kompletní zapojení dvou kanálů vstupních audio obvodů včetně motivu desky plošných spojů a nákresu osazení je v příloze A. Vstupní audio obvody jsou na jedné desce společně se sloty pro SD karty. Deska byla navržena jako jednostranná.

3.2 AD převod pomocí MCU

Procesor obsahuje tři dvanáctibitové A/D převodníky s postupnou aproximací. Každý převodník má k dispozici až 16 kanálů. Pro aplikaci jsou využity ADC3 s kanálem 1 (port PA1) a ADC2 s kanálem 15 (port PC5). Bylo tak zvoleno s ohledem na využití pinů procesoru.

AD převod realizovaný pomocí smyčky čekající na dokončení převodu, je velice nevhodný. Procesor by nemohl současně vykonávat další operace. Vhodnější je použít převod s přerušením nebo s pomocí DMA (Direct Memory Access) kontroléru, kdy se MCU může věnovat jiným operacím a AD převod obsluhuje DMA kontrolér. Byla zvolena možnost s DMA.

Budou využity dva AD převodníky a každý bude používat pouze jeden svůj kanál. To znamená, že mohou pracovat v continuous módu. V nastavení AD převodu lze

definovat periodu mezi jednotlivými převody pomocí počtu cyklů. Toto nastavení je vhodné pouze pro snížení rychlosti ADC, kterou je vhodné snížit, pokud není třeba vysoké vzorkovací frekvence. Bylo by zbytečné, aby AD převodníky vzorkovaly plnou rychlostí 2,4MS/s, jestliže vzorkujeme pouze zvukový signál. Nastavení provede následující příkaz:

```
ADC_RegularChannelConfig(ADC3, ADC_Channel_1, 1, ADC_SampleTime_28Cycles);
```

Přesnou periodu vzorkovací frekvence ovšem udává časovač a při přerušení je pomocí DMA uložen vzorek. Každá periferie má vyhrazený konkrétní kanál DMA. Pro ADC2 je to DMA2 channel2 stream0 a ADC3 je věnován DMA2 channel1 stream2. ADC může obsluhovat pouze časovač 1 až 5 a časovač 8. Pro obsluhu prvního kanálu byl zvolen časovač 5 (TIM5) a pro druhý kanál časovač 4.

Časovač musí být inicializován při každém spuštění záznamu, protože uživatel má možnost volby vzorkovací frekvence. Ta je určena nastavením časovače, které lze změnit jeho opětovnou inicializací.

Digitalizované 12-bitové vzorky se ukládají do 16-bitové proměnné se zarovnáním doleva. Je to z toho důvodu, že pro zápis do zvukového souboru jsou vhodnější vzorky s celým počtem bajtů. Jeden vzorek má tedy 2 bajty. V DMA přerušení je postupně ukládáno 16 vzorků do bufferu (32 bajtů). Po naplnění bufferu jsou data zapsána na SD kartu. Ukládání dat do zvukového souboru na kartu je diskutováno v kap. 6. Následující zdrojový kód je obsluha přerušení DMA pro první kanál.

```
void DMA2_Stream0_IRQHandler(void)
{
DMA_ClearFlag(DMA2_Stream0,DMA_FLAG_TCIF0 );
if(DoSample1==0) return;              // kontrola informace z časovače

DoSample1=0;
pBuf1[InternalBufferSize1] = ADC3ConvertedValue<<4;   //uložení vzorku
InternalBufferSize1++;                    //inkrementace indexu bufferu

 if(InternalBufferSize1>=16)
 {
      InternalBufferSize1=0;              //vynulování bufferu
      Data_Status1 = 1;                   //informace pro zápis dat
 }
}
```

4 USB KOMUNIKACE S PC

Vývojová deska STM32F4-Discovery obsahuje konektor USB micro-B a další potřebné periferie pro USB komunikaci s PC. Vzhledem k požadavkům ze zadání bylo zařízení navrženo jako USB device (procesor má na čipu USB On-The-Go, takže může být zařízení konfigurováno i jako host). Digitizér musí být ovladatelný z PC. Toto ovládání poskytuje uživateli možnost nahrát definovaný zvukový úsek. Z toho plyne, že pro tuto funkci se zařízení bude jevit jako USB HID (Human Interface Device). Zařízení ovšem musí zajišťovat přístup na SD kartu přes USB, tudíž funkci čtečky karet. V tomto případě se chová jako USB zařízení třídy Mass Storage. Z uvedeného je zřejmá nutnost použití dvou vstupních a dvou výstupních koncových bodů (Endpointů) a jednoho obousměrného koncového bodu pro řídící kanál (Control Pipe).

V dalším vývoji může být přidána třída USB Audio, která bude digitalizovaný zvuk přímo odesílat do PC, kde jej bude možné přehrát. Procesor umožňuje použití až tří vstupních a tří výstupních endpointů, takže realizace všech tříd pracujících současně je možná.

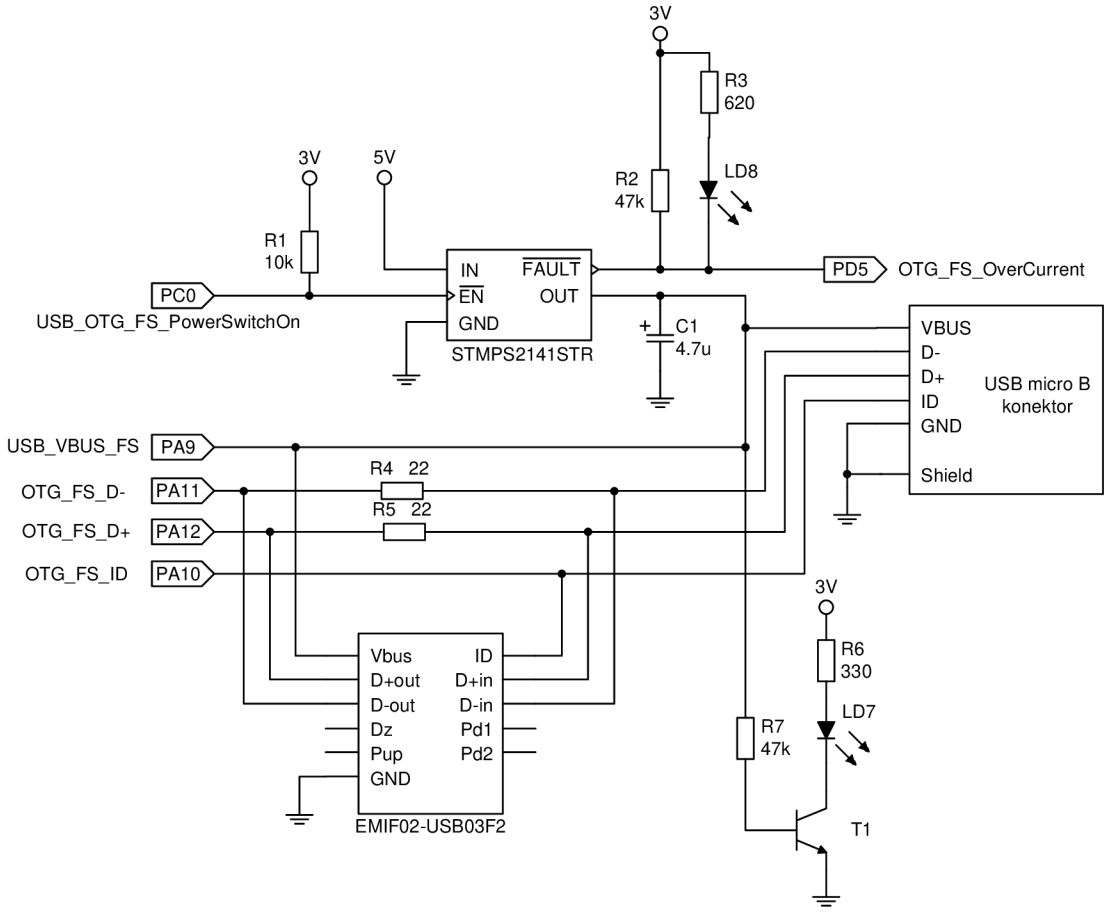
Pro přenos pomocí sběrnice USB je použit symetrický pár. Standard určuje kódování NRZI (Non-return-to-zero Inverted) s použitím bit stuffingu (vhodný způsob pro udržení synchronizace – maximálně 6 bitů (logická 1) po sobě následujích beze změny signálu). Mimo datové vodiče má sběrnice USB také zemní a napájecí vodič V_{BUS}, na kterém musí být napětí v rozsahu 4,4 – 5,25 V.

4.1 Připojení USB k procesoru

Jelikož je USB OTG (On-The-Go) driver hardwarovou součástí procesoru, není třeba příliš mnoho externích součástek. Micro-AB USB konektor je přítomen na vývojové desce Discovery. Schéma připojení konektoru k procesoru je na obr. 4.1. Způsob připojení umožňuje pouze komunikaci ve full speed (FS) módu. To znamená, že maximální frekvence na sběrnici je 25 MHz. USB High Speed komunikuje na frekvenci 50 MHz. Obvody pro napájení sběrnice jsou pro řešenou aplikaci nevyužité, protože USB je konfigurováno jako device (deska umožňuje i konfiguraci jako host).

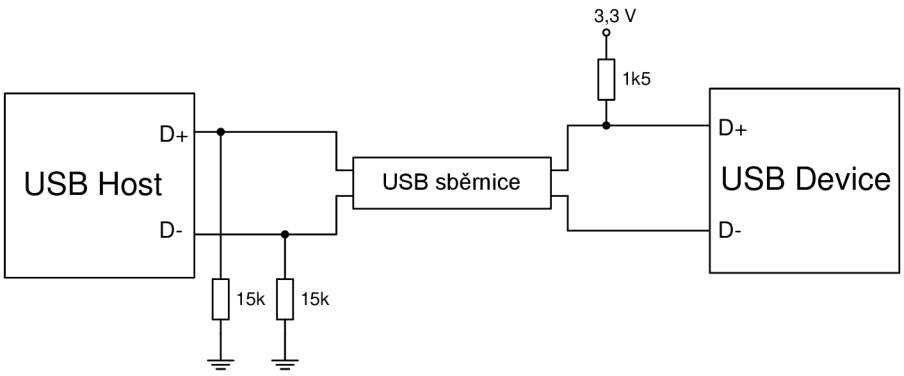
Pin ID konektoru USB je buď nezapojený (device konfigurace) nebo připojen na zem (host). Integrovaný obvod EMIF02-USB03F2 zajišťuje ESD ochranu a filtraci EMI (elektromagnetické interference). Rezistory R4 a R5 jsou použity pro dobré impedanční přizpůsobení USB sběrnice. Jejich hodnota by se měla pohybovat mezi 22 a 33 Ω. Zelená LED dioda LD7 indikuje připojení USB hostu ke sběrnici.

Zařízení s vývojovou deskou není možné napájet z micro USB konektoru. Vývojový kit to neumožňuje. Ve finálním zařízení pro tuto možnost stačí zapojit ochrannou diodu mezi VBUS a rozvod napájení (kvůli úbytku na diodě je k dispozici 4,5 V). V případě napájení ze sběrnice USB není použit obvod STMPS2141STR, který slouží pro napájení USB device v případě, že zařízení je konfigurováno jako USB host.



Obr. 4.1: Připojení USB konektoru k MCU (převzato z [11]).

USB host (popř. USB hub) pozná připojení zařízení USB device podle toho, že se na datových pinech sběrnice USB změní napěťové poměry. Jestliže není device připojen, je na sběrnici stav „single-ended-zero“ (SE0), což znamená, že jsou oba datové piny v logické nule – díky 15 kΩ pull-down rezistorům (viz obr. 4.2). Po připojení zařízení USB device přejde sběrnice do stavu „J“, kdy je díky 1,5 kΩ rezistoru na pinu D+ logická 1. Na USB jsou během komunikace stavy „J“ (D+ log 1, D- log 0), „K“ (D+ log 0, D- log 1) a SE0 pro ukončení paketu (pouze po dobu 2bitů). Stav SE1, kdy jsou oba datové piny v log 1, je zakázaný. Odpor 1,5 kΩ je součástí hardwaru USB OTG, který je na čipu mikrokontroléru. [8]



Obr. 4.2: Propojení USB device ve full-speed módu k USB host.

4.2 Deskriptory a proces enumerace

Po připojení jakéhokoli zařízení přes USB dojde k procesu tzv. enumerace, kdy se USB host dozví veškeré informace potřebné k vytvoření přenosového kanálu pomocí rour (pipes). Informace jsou předány tzv. deskriptory.

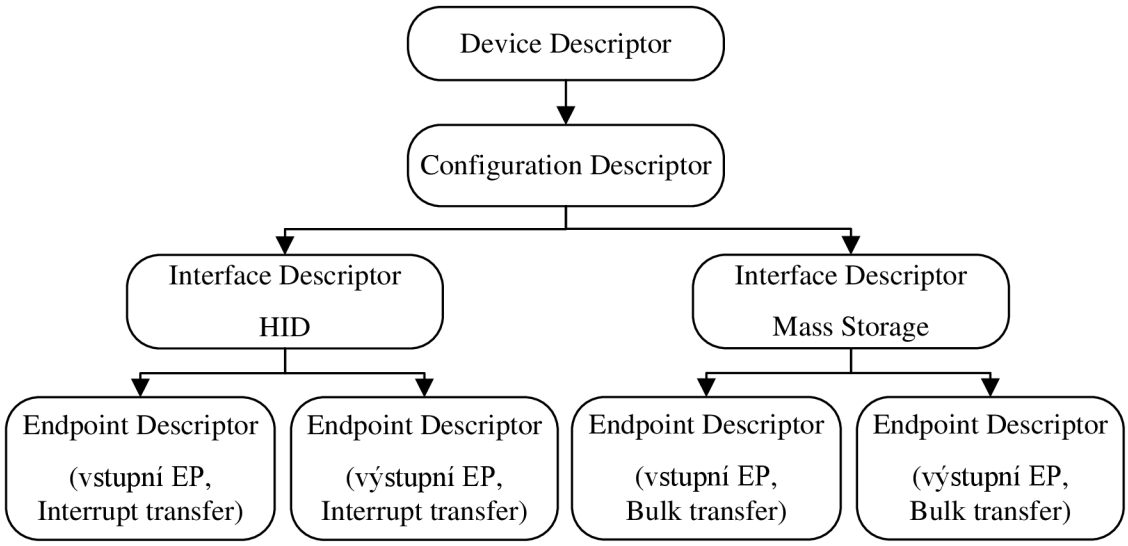
Základní typy deskriptorů jsou [8]:

- Device Descriptor (mimo jiné obsahuje VID a PID – pro všechna zařízení od firmy STM lze použít 0x0483, 0x5710)
- Configuration Descriptor (určuje způsob napájení, maximální odběr apod.)
- Interface Descriptor (obsahuje informaci o třídě, ve které bude zařízení komunikovat – HID, Mass Storage, Vendor defined,...)
- Endpoint Descriptor (popisuje komunikaci přes daný koncový bod – směr komunikace, maximální velikost paketu, typ synchronizace, použití endpointu a typ přenosu – Control, Isynchornous, Bulk, Interrupt)

Descriptory vytváří stromovou strukturu hierarchicky podle pořadí v uvedeném seznamu. Strukturu je možné libovolně větvit (např. jedno zařízení může mít definováno několik konfigurací, z nichž každá může mít několik endpointů resp. endpoint deskriptorů pro různé přenosy).

Některá zařízení mají více device deskriptorů. Toho se často využívá při potřebě nastavení různého odběru proudu pro různá nastavení (třídy). V takovém případě se zařízení jeví jako USB hub, i když fyzicky se jedná o jedno zařízení. Toto nastavení není nutné pro řešenou aplikaci. Nejvhodnější je využít dva interface deskriptory, které definují USB třídy HID a Mass Storage.

Struktura deskriptorů vypadá tak, jak je uvedeno v diagramu na obr. 4.3. Konfigurační deskriptor obsahující interface deskriptory pro HID a Mass Storage a endpoint deskriptory je v příloze B. Detailněji je implementace USB composite device popsána v kap. 4.7.



Obr. 4.3: Struktura deskriptorů pro nakonfigurování USB komunikace.

Mimo uvedené typy deskriptorů existuje ještě Report deskriptor, který definuje jednotlivé zprávy přenášené HID třídou. Je možné definovat pouze jeden vstupní a jeden výstupní report, ale pro přehlednost a praktičnost je lepší definovat více typů zpráv. Záleží na aplikaci. Pro navrhované zařízení byl napsán následující report deskriptor. Obsahuje vzory zpráv pro data z MCU (informace o stisku tlačítek a o stavu AD převodníků) a zprávy z PC do MCU, kdy je v šestnácti bajtech odeslána informace o kanálu, názvu souboru a zapnutí či vypnutí ADC (pro veškerou komunikaci z PC do MCU byl definován jeden report). [5]

```
__ALIGN_BEGIN static uint8_t
CustomHID_ReportDescriptor[CUSTOMHID_SIZ_REPORT_DESC] __ALIGN_END =
{
  0x06, 0xFF, 0x00,      /* USAGE_PAGE (Vendor Page: 0xFF00) */
  0x09, 0x01,            /* USAGE (Discovery Kit) */
  0xA1, 0x01,            /* COLLECTION (Application)       */
  /* VŠECHNY ODCHOZÍ PŘÍKAZY*/
  0x85, 0x01,            /*     REPORT_ID (1)              */
  0x09, 0x01,            /*     USAGE (1)                */
  0x15, 0x00,            /*     LOGICAL_MINIMUM (0)     */
  0x25, 0xff,             /*     LOGICAL_MAXIMUM (255) */
  0x75, 0x08,            /*     REPORT_SIZE (8 bitů)    */
  0x95, 0x0f,            /*     REPORT_COUNT (15) */
  0x91, 0x02,            /*     OUTPUT (Data,Var,Abs,Vol) =Variable */
  /* STISK A UVOLNĚNÍ TLAČÍTEK */
  0x85, 0x05,            /*     REPORT_ID (5)              */
  0x09, 0x05,            /*     USAGE (5)                */
  0x15, 0x00,            /*     LOGICAL_MINIMUM (0)         */
  0x25, 0x03,           /*  LOGICAL_MAXIMUM (2) - 4 stavy*/
  0x75, 0x08,            /*     REPORT_SIZE (24bits)            */
  0x95, 0x01,            /*     REPORT_COUNT (1)       */
  0x81, 0x02,            /*     OUTPUT (Data,Var,Abs,Vol) =Variable */
  /* STATUS DIGITIZÉRU */
  0x85, 0x06,            /*     REPORT_ID (6)              */
  0x09, 0x06,            /*     USAGE (6)                */
  0x15, 0x00,            /*     LOGICAL_MINIMUM (0)         */
  0x25, 0x03,            /*   LOGICAL_MAXIMUM (3)  = 4 různé příkazy*/
  0x75, 0x08,            /*     REPORT_SIZE (8)             */
  0x95, 0x01,            /*     REPORT_COUNT (1)       */
  0x81, 0x02,           /*     OUTPUT (Data,Var,Abs,Vol) = Variable */
  /* POČET NAHRANÝCH SOUBORŮ NA SD KARTĚ */
  0x85, 0x07,            /*     REPORT_ID (7)              */
  0x09, 0x07,            /*     USAGE (7)                */
  0x15, 0x00,            /*     LOGICAL_MINIMUM (0)         */
  0x25, 0xff,           /*     LOGICAL_MAXIMUM (255) */
  0x75, 0x08,            /*     REPORT_SIZE (8bits)            */
  0x95, 0x03,            /*     REPORT_COUNT (3) */
  0x81, 0x02,            /*     INPUT (Data,Var,Abs,Vol) =Variable */
  0xc0                /*     END_COLLECTION                  */
};
```

Jak již bylo zmíněno v popisu endpoint descriptoru, pro USB zařízení je definováno několik typů přenosu dat:

- Control Transfer (řízení komunikace, obousměrný kanál)
- Interrupt Transfer (po přerušení se posílají malé objemy dat, velikost paketů maximálně 64 B (FS), 1024 B (HS), garantuje se rychlost a latence)
- Bulk Transfer (pro velké množství dat, maximální délka paketů 64 B (FS), 512 B (HS), není garantovaná rychlost ani zpoždění, ale je zajištěna bezchybnost přenosu pomocí CRC a ARQ)
- Isochronous Transfer (pro velké množství dat, velikost paketů se může měnit během komunikace, maximální délka 1023 B (FS), 1024 B (HS), oproti předchozímu se garantuje rychlost a zpoždění a není zajištěna bezchybnost přenosu – vhodné pro live audio přehrávání, kde občasné chyby při přenosu nejsou velkou závadou)

V aplikaci je použit interrupt transfer pro třídu HID, Mass Storage využívá bulk transfer a v případě dalšího vývoje a implementace třídy Audio by byl využit i isonchronous transfer.

4.3 Komunikační protokol USB

Komunikace po USB 2.0 je rozdělena na rámce dlouhé 1 ms (pro low speed a full speed) nebo na mikrorámce 125 μs (pro high speed). Každý rámec nebo mikrorámec začíná přenosem paketu Start-of-Frame (SOF), který podrobněji popisuje tab. 4.2.

Každý přenos se skládá z jedné nebo více tzv. transakcí. Během běžné transakce jsou posílány 3 typy paketů – token paket určující směr komunikace a endpoint, datový paket (nepovinný) a handshake pro kontrolu přenosu. Jejich strukturu popisuje následující tabulka. Pakety jsou stejné pro všechny typy přenosu dat. Liší se zabezpečení, délky paketů, formát dat atd. [8]

Tab. 4.1: Struktura transakce a jednotlivých paketů.

Transakce													
Token paket						Datový paket					Handshake paket		
S	PID	ADDR	EP	CRC5	EOP	S	PID	DATA	CRC16	EOP	S	PID	EOP

Tab. 4.2: Struktura SOF paketu.

SOF paket				
S	PID	Číslo rámce (11b)	CRC5	EOP

Význam zkratek použitých v popisu paketů:

S – Synchronizace (8b)
PID – Packet identifier (4b) – určuje typ paketu
ADDR – Adresa zařízení (7b)
EP – Adresa endpointu (4b)
CRC – Cyklický zabezpečovací kód
EOP – Konec paketu

Existují tři typy token paketů – In, Out, Setup. Dělí se podle směru komunikace a třetí (Setup paket) je využit pro inicializaci řízeného přenosu. Z předchozího je patrné, že handshake pakety obsahují v podstatě jen PID, který informuje o jednom ze tří stavů (ACK – zařízení připraveno, NAK – zařízení nepřipraveno, STALL – chyba). [8]

4.4 Implementace USB HID v MCU

Na stránkách ST Microelectronics jsou dostupné knihovny pro USB, které je hardwarově přítomné přímo na čipu procesoru. Na nejnižší úrovni jsou to zejména knihovny Device Controller Driver (*usb_dcd.c, usb_dcd_int.c* – obsluha přerušení a *usb_core.c*). Pokud je zařízení konfigurováno jako USB device (procesor podporuje i možnost USB host), obsluhu podporují knihovny *usbd_core.c*, *usbd_ioreq.c*, *usbd_req.c* a *usbd_desc.c*, obsahující device deskriptor. Dále je třeba zahrnout do projektu knihovny pro použité třídy USB. Tyto knihovny se liší podle aplikace a programátor do nich zpravidla musí zasáhnout a přizpůsobit si je.

Pro HID třídu je v projektu soubor *usbd_hid_core.c*. Obsahuje konfigurační deskriptor (viz příloha B – společný deskriptor pro MSC) a report deskriptor, které jsou popsány v kap. 4.1. Dále obsahuje všechny funkce obsluhující komunikaci přes HID endpoint (dohromady s deskriptory jsou nazývány tzv. HID callbacks). Nejdůležitějšími funkcemi jsou `USBD_HID_DataOut`, která zpracovává přijatá data z PC (příkazy pro zapnutí a vypnutí jednotlivých kanálů digitizéru a příkazy pro odeslaní statusu zařízení), a funkce `USBD_HID_SendReport`, pomocí níž lze odeslat z MCU krátký dvou-bajtový příkaz.

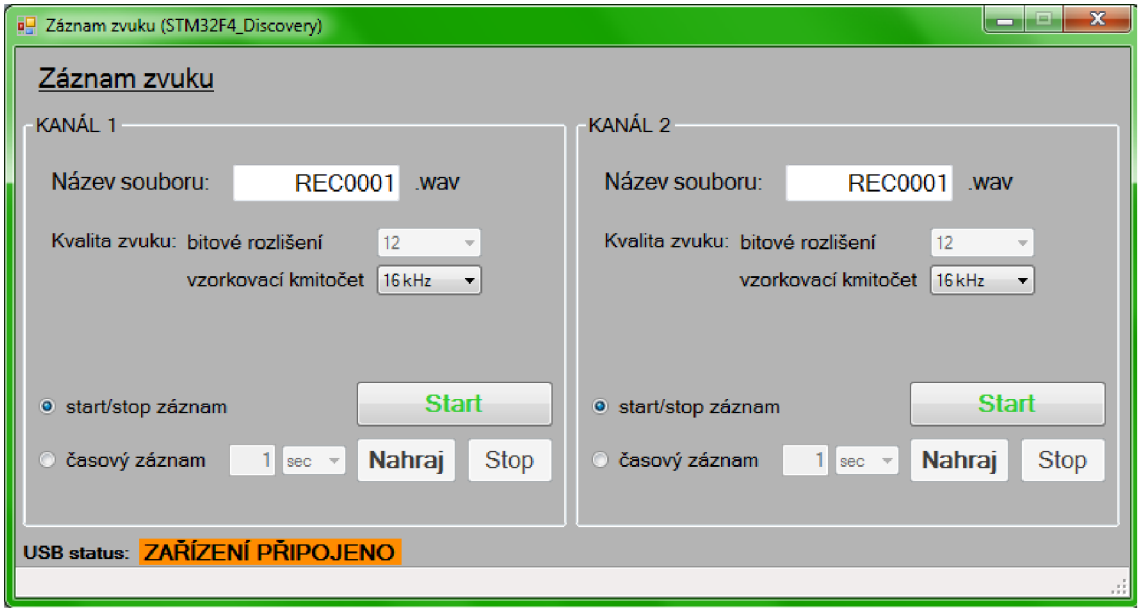
Následující tabulka ukazuje formát, v jakém jsou data z PC do MCU posílána v případě startu AD převodu. Ostatní příkazy jsou velice krátké (například jeden bajt), protože se neposílají žádná další data jako v případě startu ADC.

Tab. 4.3: Formát odesílaných příkazů z PC do MCU.

bajt	0	1	2	3 až 13	14	15
význam	report ID	příkaz (např. start)	délka názvu souboru	název souboru	bitové rozlišení	vzorkovací frekvence

4.5 HID aplikace pro ovládání zařízení z PC

Aplikace je naprogramovaná v jazyce C#. Pro tento jazyk existuje vhodná volně dostupná knihovna pro USB HID [10]. Knihovna obsahuje základní funkce potřebné pro komunikaci v přídě HID. Třída HID má tu výhodu, že uživatel nemusí instalovat žádné ovladače, protože od Windows XP výše jsou ovladače HID součástí systému. Vzhled aplikace je na následujícím obrázku.



Obr. 4.4: Vzhled aplikace po spuštění.

Uživatel má možnost ovládat z počítače oba kanály. Pro každý si může zvolit kvalitu zvuku, název souboru, pod jakým se uloží digitalizovaný záznam na SD kartu a k dispozici má dva způsoby spuštění záznamu – spuštění digitalizace, které se ukončí, až je přeruší sám uživatel (maximální délka nahrávky je však 5 minut) a spuštění se zadáním časového intervalu, který chce nahrát.

Aplikace je synchronizována s ovládáním na zařízení. Pokud je spuštěn záznam ze zařízení, daný kanál není možné ovládat z aplikace. Naopak je to možné, ale aplikace vždy dostane informaci o stavu zařízení a podle toho se nastaví tlačítka apod. V případě připojení zařízení jsou ze zařízení odeslány statusy obou AD převodníků, aby aplikace nabídla uživateli správné volby. Nastavení tlačítek podle aktuálního stavu zařízení (zjištěný při připojení nebo v případě ovládání zařízení přímo) je prováděno přímo ve funkci, která přijímá data o stavu zařízení. V tomto případě je nutné operaci volat příkazem Invoke a mít definovaný objekt, ke kterému potom lze přistupovat jako delegate. Následně je ukázán příklad s komponentou group box.

Definice objektu groupBox1:

```
delegate void MyDelegat(object groupBox1);
```

Volání funkce lockgp1, pro zamčení objektu groupBox1, pomocí Invoke:

```
Invoke(new MyDelegat(lockgp1), e);
```

Aplikace při spuštění zkontroluje, zda je digitizér připojen či nikoli. Tato kontrola probíhá i během celé doby, kdy je aplikace spuštěna. Tento proces aktualizuje aplikaci tak, aby byla vhodně přizpůsobená na ovládání a informuje uživatele o připojení či odpojení USB. Mimo to odešle digitizéru žádost o zaslání informací o stavu AD převodníků a podle toho se také přizpůsobí. V tomto procesu po připojení se také odešle do zařízení aktuální datum a čas. Následující kód prezentuje odeslání těchto informací z PC do zařízení.

```
Buffer2[0] = 1; // report 1
Buffer2[1] = 6; // index žádosti o zaslání statusu digitizéru
year = DateTime.Now.Year;
Buffer2[2] = (byte) (year >> 8);                    //rok (vyšší bajt)
Buffer2[3] = (byte)(year & byte.MaxValue);          //rok (nižší bajt)
Buffer2[4] = Convert.ToByte(DateTime.Now.Month);  //měsíc
Buffer2[5] = Convert.ToByte(DateTime.Now.Day);    //den v měsíci
Buffer2[6] = Convert.ToByte(DateTime.Now.Hour);   //hodina
Buffer2[7] = Convert.ToByte(DateTime.Now.Minute); //minuta
Buffer2[8] = Convert.ToByte(DateTime.Now.Second); //sekunda
usbhid.write(Buffer2);   // odeslání informací přes HID endpoint
```

Ve spodní části okna aplikace je uživatel informován o chybách. Těmi může být např. neplatné zadání názvu souboru (nepovolené znaky), neplatné zadání délky nahrávky apod. Tento informační řádek byl vytvořen pomocí komponenty status strip.

Uživatel má možnost si zvolit libovolný název souboru. Ten je ovšem omezen na 7 znaků bez přípony, protože ovládání FAT systému nepodporuje Long File Name. Pokud uživatel nezvolí název souboru, je přednastaven název REC0001. Při dalším nahrávání je název souboru automaticky nastavován na REC0002, REC0003 atd. Nahrané soubory jsou rozděleny do složek podle kanálu (CHANNEL1 a CHANNEL2).

Aplikace společně s projektem pro Visual Studio 2010 je na přiloženém CD. Spustitelný soubor se jmenuje PC_HID_record.exe.

4.6 USB Mass Storage třída

Jak bylo zmíněno v úvodu kapitoly, pro komunikaci s SD kartou je třeba využít další dva endpointy podporující bulk transport. Pro USB komunikaci je definována další třída a to Mass Storage class (MSC), která je určena k přenosům velkých dat, tudíž k přenosům mezi paměťovými zařízeními jako je např. hard disk nebo SD karta.

Pro třídu HID bylo nutné přidat soubor *usbd_hid_core.c* obsahující deskriptory a funkce pro komunikaci v třídě HID. V případě třídy MSC je nezbytné přidat čtyři .c soubory s jejich hlavičkovými soubory. Pochopitelně mezi ně patří soubor *usbd_msc_core.c* s descriptory a funkcemi pro MSC třídu, dále *usbd_msc_bot.c* s funkcemi pro podporu přenosového protokolu BOT (Bulk Only Transport), *usbd_msc_data.c* pro načtená data a příkazy čekající na zpracování a soubor *usbd_msc_scsi.c* pro podporu protokolu USB Attached SCSI (Small Computer System Interface). Tento protokol umožňuje transport dat mezi počítačem a paměťovým zařízením připojeným přes USB (nejčastěji ve třídě MSC). [9] Dalším nutným souborem pro správnou funkci MSC třídy je *usbd_storage_msd.c*. Tato knihovna je konektivitou mezi USB MSC třídou a paměťovým médiem (SD kartou), resp. knihovnou pro komunikaci s daným paměťovým zařízením.

Zmíněné soubory potřebné pro správnou komunikaci s paměťovým zařízením přes USB jsou dostupné na stránkách fy STM [1] v knihovně příkladů STM32F4-Discovery_FW_V1.1.0 pro zvolený vývojový kit. Samozřejmě je třeba upravit deskriptory (adresu endpointu apod.).

Kromě souborů pro řízení komunikace přes USB je třeba zajistit přístup na SD kartu tak, aby ji uživatel viděl mezi paměťovými zařízeními. Driver

stm324xg_eval_sdio_sd.c (v projektu byl modifikován a pojmenován jako *stm32f4_discovery_sdio_sd.c*) obsahující příkazy pro operace s SD kartou je také dostupný na stránkách fy STM [1]. Je součástí USB knihovny STM32_USB-Host-Device_Lib_V2.1.0. Driver je určen pro vývojový kit STM3240G-EVAL obsahující stejný procesor jako vývojová deska Discovery, ale navíc má i další periferie jako SD slot, displej apod. Driver komunikuje s kartou přes SDIO rozhraní. Pro správnou funkci bylo nutné zkombinovat knihovny pro Discovery a Evaluation desku určující GPIO nastavení portů podle připojeného hardwaru (*stm32f4_discovery.c, stm324xg_eval.c*).

Jestliže se zařízení chová jako čtečka karet, pro SD kartu je master PC (popř. jiné zařízení, ke kterému je připojena čtečka. To je důležitý fakt, který se projeví v obsluze FAT systému. Ten je řízen vždy masterem, tudíž počítačem. V tomto případě není třeba řešit souborový systém na úrovni MCU. Ten musí zprostředkovat pouze přenos dat pomocí USB a přístup na kartu. FAT systém obsluhuje mikrokontrolér, pokud sám zapisuje na paměťové zařízení (např. digitalizovaná data).

SD karta a zápis na ni je podrobně rozebrán v kap. 5. Souborovému systému FAT je věnována kapitola 6.

4.7 USB composite device

USB třídy HID a Mass Storage byly během vývoje realizovány samostatně ve dvou různých aplikacích. Pro finální zařízení je ovšem vhodné konfigurovat USB tak, aby měl uživatel přístup na SD kartu a současně mohl ovládat zařízení z PC pomocí softwaru, tedy inicializovat obě dvě třídy pro jedno zařízení. Takové zařízení se nazývá USB composite device. Deskriptory pro toto zařízení jsou popsány v kap. 4.2 a v příloze B jsou uvedeny samotné deskriptory (resp. konfigurační deskriptor obsahující interface a endpoint deskriptory). Dvě třídy se inicializují využitím dvou interface deskriptorů.

Následující funkce inicializuje USB po startu programu MCU.

```
USBD_Init(&USB_OTG_dev, //proměnná popisující veškeré dění na USB OTG
    USB_OTG_FS_CORE_ID, //zařízení ve full speed módu
    &USR_desc, //device a string deskriptory (ID zařízení, popis atd.)
    &USBD_class_cb, //ukazatel na deskriptory a callback funkce
    &USR_cb);     //uživatelské callbacky (události po konfiguraci
                    USB, odpojení, připojení apod.)
```

Ukazatel *USBD_class_cb* předá funkci deskriptory a callback funkce pro obsluhu jednotlivých tříd. Ve [12] je uvedeno, že USB knihovna pro procesory řady STM32F4xx je vhodná pro realizaci USB composite device. Firma ST ovšem neposkytuje žádný příklad této realizace a pravděpodobně mnoho vývojářů pracujících s STM32 procesory zatím nevytvořilo něco podobného. Přitom realizace složeného zařízení USB je poměrně běžná.

Pro inicializaci USB composite device je třeba nějakým způsobem předat funkci *USBD_Init* callback funkce pro více tříd. Byl učiněn pokus inicializovat společně s deskriptory všechny HID i Mass Storage callback funkce. Tedy proměnná *USBD_class_cb* obsahovala dvakrát více callback funkcí. Tato proměnná je typu *USBD_Class_cb_TypeDef*, který je definován v *usb_core.h* a původně to byla tato struktura:

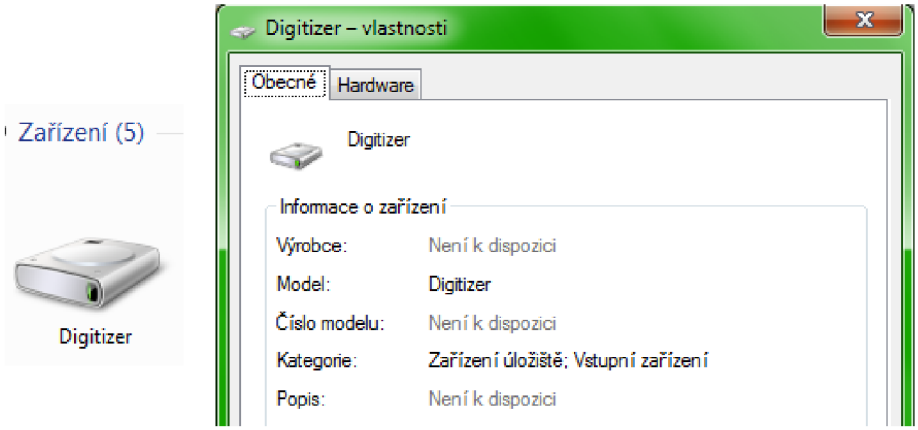
```
typedef struct _Device_cb
{
  uint8_t  (*Init)          (void *pdev , uint8_t cfgidx);
  uint8_t  (*DeInit)        (void *pdev , uint8_t cfgidx);
  /* Control Endpoints*/
  uint8_t  (*Setup)         (void *pdev , USB_SETUP_REQ  *req);
  uint8_t  (*EP0_TxSent)    (void *pdev );
  uint8_t  (*EP0_RxReady)   (void *pdev );
  /* Class Specific Endpoints*/
  uint8_t  (*DataIn)        (void *pdev , uint8_t epnum);
  uint8_t  (*DataOut)       (void *pdev , uint8_t epnum);
  uint8_t  (*SOF)           (void *pdev);
  uint8_t  (*IsoINIncomplete)  (void *pdev);
  uint8_t  (*IsoOUTIncomplete)  (void *pdev);

  uint8_t  *(*GetConfigDescriptor)(uint8_t speed ,uint16_t *length);
} USBD_Class_cb_TypeDef;
```

Proto muselo být zasaženo i do těchto USB knihoven a tato struktura byla rozšířena, aby mohla obsahovat funkce pro 2 třídy. Bohužel USB se po takto provedené inicializaci nekonfigurovalo správně, takže toto řešení není pravděpodobně vhodné. Navíc podle [12] lze vytvářet s použitou knihovnou složené zařízení USB, takže by se nemělo zasahovat do souborů na nižší úrovni, což je i *usb_core.h* v *usb_core.c*.

Druhou možností realizace composite device je vytvoření univerzálních callback funkcí, které obsahují kód pro Mass Storage i HID. Obě třídy využívají tyto callback funkce: *Init, DeInit, Setup, DataIn* a *DataOut*. Ostatní funkce, které jsou definovány ve struktuře `USBD_Class_cb_TypeDef` nejsou využity. Callbacky *Init* a *DeInit* obsahují příkazy pro obě třídy neodděleně. Pouze v inicializaci je na začátku nutné deinicializovat samotný Mass Storage class. Do funkce *Setup* byly zahrnuty requesty od obou tříd. Většina requestů je určena pouze pro jednu třídu, takže je není třeba nějak oddělovat. Pouze requesty USB_REQ_GET_INTERFACE a USB_REQ_SET_INTERFACE jsou společné. Pro tyto funkce lze provést společný příkaz *USBD_CtlSendData* posílající data na control pipe přes endpoint 0. Tento příkaz pracuje se společnou proměnnou pro HID a Mass Storage. Callback funkce *DataIn* a *DataOut* mají oddělené routiny pro MSC a HID pomocí adresy endpointu. Řešení funkce *DataIn* ukazuje následující zdrojový kód.

```
static uint8_t  USBD_HID_DataIn (void  *pdev, uint8_t epnum)
{
if (epnum==0x01)  // adresa HID endpointu
{
  DCD_EP_Flush(pdev, HID_IN_EP);//Vyprázdnění FIFO zásobníku na HID EP
}
else  // adresa Mass Storage class endpointu
{
  /*přenos dat do PC pomocí Bulk Only Transfer protokolu*/
  MSC_BOT_DataIn(pdev , epnum);
}
return USBD_OK;
}
```

Enumerace takto konfigurovaného zařízení proběhla správně zařízení se přihlásilo jako složené zařízení USB. Je možný přístup na SD kartu a současně je možná komunikace s PC aplikací přes HID třídu. Položku Digitizér je možné najít mezi zařízeními a tiskárnami. V jejích vlastnostech jsou uvedeny 2 kategorie (třídy).



Obr. 4.5: Přihlášení digitizéru ve Windows.

4.8 Testování USB

Během vývoje konfigurace USB je třeba měnit sériové číslo zařízení. Pokud se konfigurace jednou nezdaří, zařízení se nekonfiguruje správně ani po opravě firmwaru pro enumeraci USB. Systém Windows si ukládá sériová čísla a informace o všech USB zařízeních, která byla připojena k danému počítači. Proto je třeba vždy při změně deskriptorů, callback funkcí, apod. přidělit zařízení jiné sériové číslo, aby systém provedl celou konfiguraci zařízení.

Sériové číslo s dalšími údaji popisující zařízení je nastaveno v souboru *usbd_desc.c*. Soubor obsahuje také funkce, které tyto údaje předávají počítači při enumeraci. Všechny tyto údaje jsou inicializovány společně s Device deskriptorem.

```
USBD_DEVICE USR_desc =
{
  USBD_USR_DeviceDescriptor,
  USBD_USR_LangIDStrDescriptor,
  USBD_USR_ManufacturerStrDescriptor,
  USBD_USR_ProductStrDescriptor,
  USBD_USR_SerialStrDescriptor,
  USBD_USR_ConfigStrDescriptor,
  USBD_USR_InterfaceStrDescriptor,
};
```

Tyto údaje jsou vlastně také deskriptory daného USB zařízení. Konkrétní údaje o zařízení jsou uvedeny následně.

```
#define USBD_VID                     0x0483  //STM VID
#define USBD_PID                     0x5710  //STM PID
#define USBD_LANGID_STRING            0x409  //English
```

```
#define USBD_MANUFACTURER_STRING          "UREL project"
#define USBD_PRODUCT_FS_STRING_HID        "Digitizer (HID+MSC)"
#define USBD_SERIALNUMBER_FS_STRING_HID   "000000000A1F"
#define USBD_CONFIGURATION_FS_STRING_HID  "HID+MSC Config"
#define USBD_INTERFACE_FS_STRING_HID      "HID+MSC Interface"
```

Zařízení USB composite device bylo správně konfigurováno a USB pracuje správně. S touto konfigurací ovšem vznikl problém. Třída Mass Storage pracuje sice správně, ale ovlivňuje natolik zařízení, že ostatní periferie nepracují správně. Jedná se zejména o hlavní smyčku stavového automatu, který obsluhuje ADC, tlačítka, LED diody, zápis digitalizovaných souborů, atd. Z debugování bylo vypozorováno, že na USB endpointy třídy Mass Storage jsou neustále odesílána a přijímána data. Tato aktivita na USB sběrnici byla zaznamenána i v nepřítomnosti SD karty. Toto chování třídy MSC pravděpodobně způsobuje, že zařízení nepracuje správně.

Z tohoto důvodu byla USB komunikace vyřešena způsobem popsaným v následující kapitole.

4.9 USB s přepínanými třídami HID a MSC

Kvůli uvedeným problémům nebylo možné použít konfiguraci USB composite device. Alternativním řešením je přepínání mezi konfigurací USB HID a MSC.

Přepnutí z jedné USB třídy do druhé a zpět je možné pomocí dlouhého stisku tlačítka na portu PA0 (user button na desce Discovery). Po stisku tlačítka je třeba nejprve softwarově odpojit USB a poté inicializovat novou třídu stejně jako po resetu zařízení. Bez odpojení by USB host (PC) nerozpoznal, že USB zařízení změnilo svoji konfiguraci a nebyla by provedena enumerace. Softwarové odpojení lze provést nastavením bitu SDIS do logické 1 v registru OTG_FS_DCTL (device control registr). S využitím definicí z USB knihovny vypadá funkce pro odpojení následovně.

```
void Disconnect_USB(USB_OTG_CORE_HANDLE  *pdev)
{
  pdev->regs.DREGS->DCTL |= 0x02;
}
```

pdev – ukazatel na USB device
regs – USB registry

Před inicializací nové konfigurace USB je vhodné použít zpoždění, aby bylo zařízení odpojeno nějakou dobu. Minimální doba, kdy host bezpečně detekuje odpojení je 1,0025 ms. [4]

Třídy mají konfigurační deskriptor a callback funkce v oddělených souborech (*usbd_hid_core.c* a *usbd_msc_core.c*).

Přepnutí do konfigurace s USB MSC (pro přístup na kartu) je možné pouze v případě, kdy ani jeden ADC kanál není spuštěn.

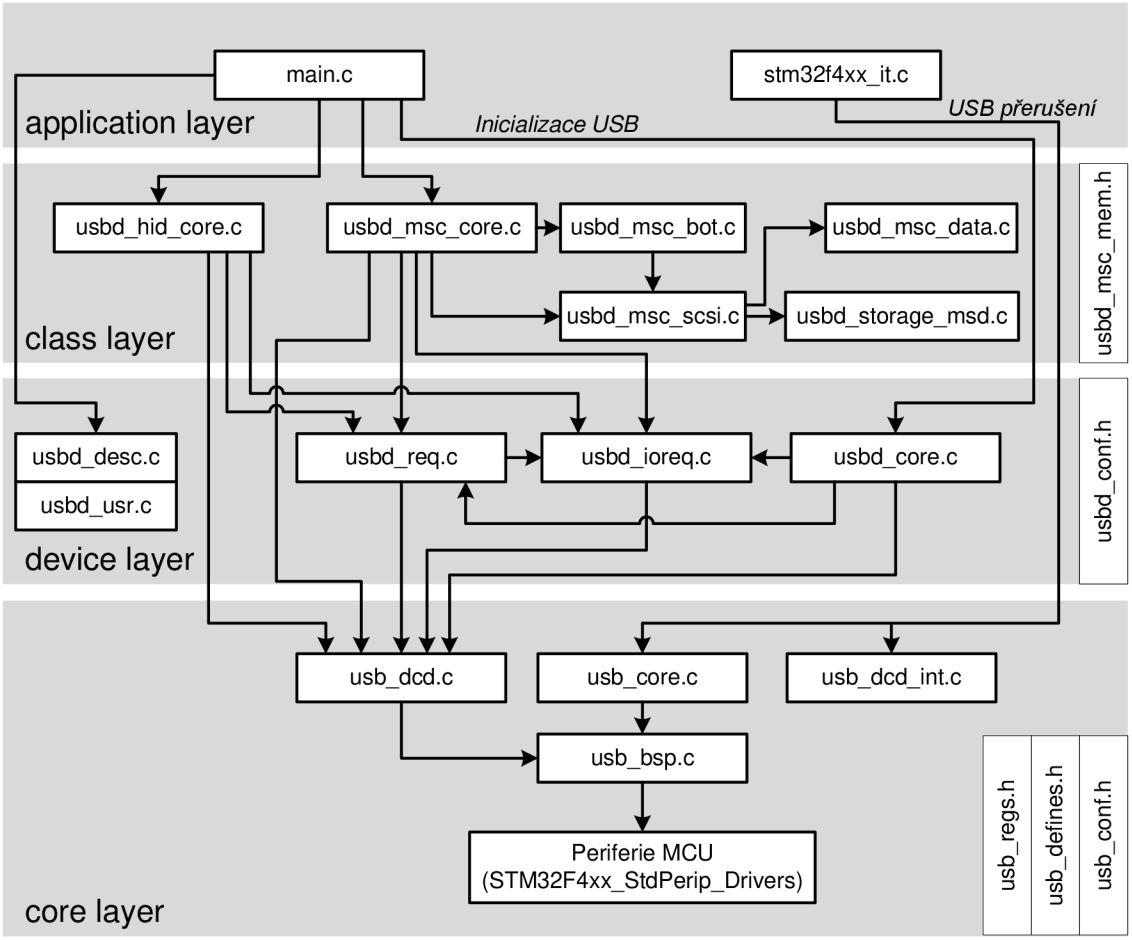
Toto alternativní řešení má za následek to, že uživatel nemá přístup na SD kartu současně s možností ovládání zařízení z aplikace v počítači. Nicméně obě funkce jsou k dispozici. Přepnutí trvá přibližně 1 až 2 vteřiny. Musí totiž dojít k nové enumeraci a inicializaci paměťového zařízení.

4.10 Struktura USB knihoven v projektu aplikace

Protože modul USB skýtá poměrně velké množství souborů, z nich některé byly popsány výše, je vhodné uvést strukturu propojení všech USB .c souborů, které jsou součástí projektu.

Struktura je uvedena na následujícím obrázku. Odpovídá verzi projektu s přepínanými USB třídami. Šipky znamenají volání funkcí mezi jednotlivými soubory (lze říci sestupování v hierarchii na nižší vrstvy). Každý .c soubor má samozřejmě hlavičkový soubor .h se stejným jménem. Výjimkou je soubor *usbd_storage_msd.c,* jehož hlavičkový soubor se jmenuje *usbd_msc_mem.h*.

Modul USB zahrnuje i .h soubory, které nejsou hlavičkovým souborem pro konkrétní .c soubor, ale obsahují definice a konstanty pro celou řadu souborů (nebo pro celou vrstvu komunikace USB). Tyto soubory jsou uvedeny v pravé části obr. 4.6. Soubor *usb_defines.h* obsahuje definice a makra pro soubory pracující na nejnižší vrstvě, *usb_conf.h* obsahuje konfiguraci nižších vrstev komunikace a konstanty pro konfiguraci USB tříd a device vrstvy jsou v souboru *usbd_conf.h*. Soubor *usb_regs.h* obsahuje definice USB registrů a jejich adresy.



Obr. 4.6: Struktura souborů USB modulu.

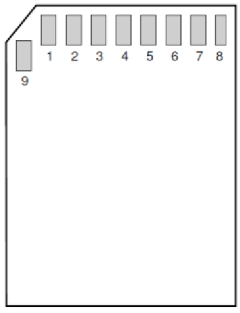
5 SD KARTA

V následující kapitole je rozebrán standard SD (Secure Digital) karty, protokoly pro komunikaci s SD kartou a připojení karty jako periferie k MCU.

SD karty jsou přenositelné paměti typu flash. V dnešní době patří mezi nejpopulárnější paměťové karty. Konkurují jim zejména standardy CF (Compact Flash), CF+, Sony Memory Stick a MMC (MultiMedia Card). SD karty mají malé rozměry, mají relativně jednoduché fyzické rozhraní, oproti zmíněným standardům mají menší spotřebu (pod 100mA), což je důležité při použití v mobilních zařízeních a jejich cena je také příznivá. Navíc se standard SD karet neustále vyvíjí a výrazně se zvyšuje jejich kapacita. Původní SD karty byly omezeny do velikosti 2 GB. Technologie SDHC (SD High Capacity) přinesla zvýšení maximální kapacity na 32 GB a zrychlení zápisu a čtení dat z karty. Podle této rychlosti se SDHC karty dělí do tříd (2,4,6,8,10), jejichž označení odpovídá rychlosti zápisu v MB/s. V roce 2009 byla představena další technologie, a to SDXC (SD eXtended Capacity). Standard přináší možnost výroby až 2TB karet. V současnosti jsou zákazníkům k dispozici SDXC karty do maximální kapacity 128 GB. Přenosová rychlost tohoto standardu může dosahovat až 300MB/s. Standard využívá nový systém souborů exFAT.[13][14]

5.1 Standard a protokoly

Standard pro SD kartu vychází ze staršího standardu MultiMediaCard (MMC). Byl vyvinut firmami Toshiba, MEI a SanDisk [14]. Licenci SD standardu vlastní organizace SD Card Association [15]. Na obr. 5.1 je zobrazeno rozmístění pinů SD karty a popis k nim pro všechny tři používané komunikační módy je v tab. 5.1.



Obr. 5.1: Rozmístění pinů na SD kartě (převzato z [13]).

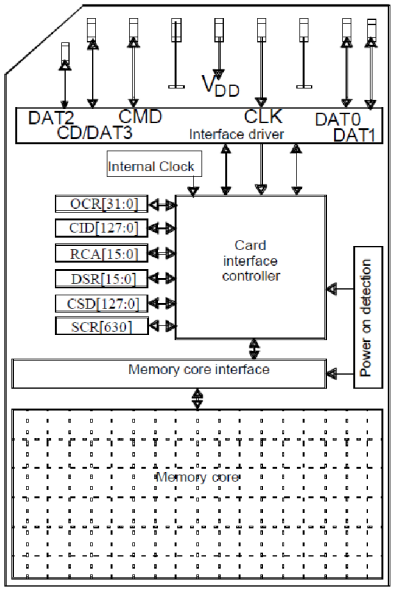
Tab. 5.1: Popis funkce jednotlivých pinů SD karty pro používané módy.[13]

	SD mód (4-bitový)		SD mód (1-bitový)		SPI mód	
Pin	Jméno pinu	Funkce	Jméno pinu	Funkce	Jméno pinu	Funkce
1	DAT3	Data Line 3	NC	Nevyužito	CS	Chip select
2	CMD	Command Line	CMD	Command Line	DI	Data In (příkazy a data)
3	VSS1	Zem	VSS1	Zem	VSS1	Zem
4	VDD	Napájení	VDD	Napájení	VDD	Napájení
5	CLK	Hodinový signál	CLK	Hodinový signál	CLK	Hodinový signál
6	VSS2	Zem	VSS2	Zem	VSS2	Zem
7	DAT0	Data Line 0	DAT	Data Line	DO	Data Out (data a status)
8	DAT1	Data Line 1	IRQ	Interrupt	IRQ	Interrupt
9	DAT2	Data Line 2	R/W	Read/Write	NC	Nevyužito

Vnitřní struktura SD karty je uvedena na obr. 5.2. Odesílání odpovědí a řízení přenosu dat je obsluhováno řadičem SD karty. Ten přistupuje přes určité rozhraní k flash paměti karty, k výstupním pinům a k registrům. Ty jsou uvedeny společně s jejich významem v následující tabulce. Jejich obsahem je zejména popis vlastností karty a její aktuální parametry.

Tab. 5.2: Registry SD karty.

Označení registru	Název registru	Popis
OCR	Operation Condition Register	Údaje o napájení SD karty (hodnota napájení je popsána od 1,6 V do 3,6 V s rozlišením 0,1 V)
CID	Card Identification Register	Identifikace karty – ID výrobce, OEM (Original Equipment Manufacturer), sériové číslo, jméno karty, datum výroby, verze produktu (naprogramováno ve výrobě, nelze měnit obsah)
CSD	Card Specific Data	Informace o operacích s kartou (registr je nutný pro přístup k datům v paměti, obsahuje například informace o ochraně proti zápisu – další možností je přepínač, který je ovšem jen na standardních SD kartách)
RCA	Relative Card Address	Lokální systémová adresa přidělené masterem po inicializaci (pouze v SD módu)
SCR	SD	Informace o speciálních funkcích SD karty



Obr. 5.2: Architektura SD karty. (převzato z [13])

SD protokol funguje na principu jednoduchého command-response protokolu. Master vždy řídí komunikaci a posílá příkazy. SD karta (slave) reaguje odpověďmi v tzv. rámcích a poté začne například čtení či zápis dat.

SD karta podporuje dva režimy, tzv. SD mód a SPI mód. Ty se liší ve fyzickém zapojení a v řízení komunikace (inicializační sekvence, formát příkazů a odpovědí). Detailněji je komunikace pro každý mód popsána níže.

Mód SPI (Serial Peripheral Interface) komunikuje po klasické sériové sběrnici. Díky tomu je možné připojit SD kartu prakticky k libovolnému procesoru. Na druhou stranu s tímto režimem se dosahuje menších přenosových rychlostí.

5.1.1 Řídící příkazy

V obou komunikačních módech je komunikace řízena pomocí příkazů dvou typů – hlavní řídící rámec (ve formátu CMDXX) a aplikační řídící rámec (ve formátu ACMDXX). Za XX se skrývá číslo příkazu. Řídící rámce mají velikost 6 bytů. Všechny příkazy a odpovědi mají strukturu big endian (MSB first). Struktura rámce je v následující tabulce.

Tab. 5.3: Struktura řídícího rámce (příkazu).[14]

Byte 1			Byte 2 – 5	Byte 6	
0	1	Číslo příkazu	Argument (big endian)	CRC7	1

První dva bity (01) jsou startovní sekvence. Rámec je ukončen stop bitem 1. Cyklický zabezpečovací kód je v režimu SPI volitelný.

Seznam důležitých příkazů, které jsou nejčastěji používané pro komunikaci s SD kartou, jsou uvedeny v následující tabulce.

Tab. 5.4: Důležité příkazy.[13]

Označení příkazu	Argument	Typ odpovědi	Popis
CMD0	bez argumentu	R1	Reset karty a vstup do režimu idle
CMD3	bez argumentu	R6	Žádost o odeslání RCA (pouze pro SD mód)
CMD9	bez argumentu	R2	Žádost o odeslání CSD (pouze pro SD mód)
CMD10	bez argumentu	R2	Žádost o odeslání CID (pouze pro SD mód)
CMD16	délka bloku (32 bitů)	R1	Nastavení velikosti bloku
CMD17	adresa bloku (32 bitů)	R1	Čtení bloku dat
CMD18	adresa bloku (32 bitů)	R1	Čtení více bloků dat (přenos ukončen až příkazem CMD12)
CMD24	adresa bloku (32 bitů)	R1	Zápis bloku dat
CMD25	adresa bloku (32 bitů)	R1	Zápis více bloků dat (až do přerušení příkazem CMD12)
CMD55	bez argumentu	R1	Příští příkaz bude aplikační (ACMD)
CMD58	bez argumentu	R3	Čtení OCR registru
ACMD41	bez argumentu	R1	Inicializace karty

Na každý příkaz reaguje karta odezvou. Typy příkazů, odezvy a další komunikace mezi kartou a zařízením se liší pro SPI mód a SD mód. Proto je popis rozdělen do dvou podkapitol.

5.1.2 Příkazy a odezvy v SPI módu

Pro SPI mód jsou definovány tři typy odpovědních rámců (R1, R2, R3), jejichž struktura je zobrazena v tabulkách 5.5, 5.6 a 5.7. Všechny tyto data jsou přenášeny přes CMD pin stejně jako příkazy. Rámce jsou oproti SD módu poněkud kratší a nejsou zabezpečeny pomocí CRC kódů.

Tab. 5.5: Struktura rámce R1.[14]

Byte	Bit	Význam
1	7	Start bit (0)
	6	Chyba parametru příkazu
	5	Chyba adresy
	4	Ztráta sekvence
	3	Chyba CRC
	2	Neplatný příkaz
	1	Přerušení mazání dat
	0	V režimu idle

Tab. 5.6: Struktura rámce R2.[14]

Byte	Bit	Význam
1	0-7	Rámec R1
2	7	Karta je plná
	6	Neplatná volba parametru mazání
	5	Porušení ochrany zápisu – příkaz se pokoušel zapsat data
	4	Chyba ECC (Error Correction Code)
	3	Chyba řadiče karty
	2	Nespecifikovaná chyba
	1	Příkaz se pokoušel smazat chráněný sektor dat nebo bylo zadáno chybné heslo pro zamčení/odemčení karty
	0	Karta je zamčená (pomocí přepínače)

Tab. 5.7: Struktura rámce R3.[14]

Byte	Bit	Význam
1	0-7	Rámec R1
2-5	všechny	OCR (Operating Conditions Register)

Před přenosem uživatelských dat je nutné aplikovat příkaz pro čtení či zápis a startovací sekvenci, která inicializuje buď přenos jednoho bloku dat (11111110) nebo mnohonásobný přenos bloků dat (11111100). Po ní následuje samotný přenos datového bloku, který je typicky dlouhý 512 bytů. Každý datový blok je zakončen 16-bitovým kontrolním kódem CRC. Pro zabezpečení dat musí být CRC kódy použity i v SPI módu. V případě mnohonásobného přenosu bloků je nutné pro ukončení přenosu použít ukončovací sekvenci „11111101“ (pro zápis) nebo příkaz CMD12 (pro čtení). Po každém zápisu karta odpoví odesláním tzv. Write status rámce, jehož struktura a možné odpovědi jsou popsány v následující tabulce.

Tab. 5.8: Write status rámec.[13]

Bit	Význam
7-5	nevyužity
4	0
3-1	010 – data byla přijata 101 – data nebyla přijata (CRC chyba) 110 – data nebyla přijata (chyba zápisu)
0	1

Čtení z karty se liší tím, že po přenosu dat karta posílá další rámec pouze v případě chyby. Tento rámec je nazýván Read error rámec a významy jeho jednotlivých bitů jsou v následující tabulce (tab. 5.9).

Tab. 5.9: Read error rámec.[13]

Bit	Význam
7-5	vždy 0
4	Karta je zamčena
3	Mimo rozsah
2	Chyba ECC (Error Correction Code)
1	Chyba řadiče karty
0	Nespecifikovaný chyba

V SPI módu jsou z registrů SD karty dostupné pouze OCR (Operating conditions register), CSD (Card specific data) a CID (Card identification number).

5.1.3 Komunikace s SD kartou v SD módu

Pro komunikaci v SD módu slouží jednobitový a čtyřbitový protokol. Jednobitový protokol využívá pouze pin DAT0 pro přenos dat. V SD módu je datový přenos zabezpečen pomocí 16-ti bitového CRC (Cyclic redundancy check) kódu pro každý datový vodič. Generující polynom je

$$G(x) = x^{16} + x^{12} + x^{5+1}.$$

Odezvy a příkazy jsou zabezpečeny 7 bitovým CRC kódem, jehož generující polynom vypadá následovně:

$$G(x) = x^7 + x^3 + 1$$

V 1-bitovém SD módu je datový paket odeslán od MSB po LSB. Ve 4-bitovém módu jsou data rozdělena po jednotlivých bitech na 4 datové porty, jak ukazuje následující tabulka (příklad s 4096 bitovým paketem). Každý datový pin má extra CRC zabezpečení.

Tab. 5.10: Odesílání dat ve 4-bitovém SD módu.

DAT3	4095	4091 ÷ 7	3	CRC	1
DAT2	4094	4090 ÷ 6	2	CRC	1
DAT1	4093	4089 ÷ 5	1	CRC	1
DAT0	4092	4088 ÷ 4	0	CRC	1

V SD módu jsou odpovědi poněkud komplexnější a je jich více typů oproti SPI módu. Standard definuje odezvy R1, R1b (busy), R2, R3 a R6. Odezvy R4 a R5 využívají pouze MMC karty. Formáty odezev jsou uvedeny v následujících tabulkách. Odezva R1b (busy) má stejný formát jako odezva R1, ale navíc se vysílá extra signál přes datové piny.

Tab. 5.11: Odezva R1 v SD módu.

Byte	Bit	Význam
1	7	Start bit (0)
	6	Přenosový bit (0)
	5-0	Index příkazu
2-5		Status karty
6	7-1	CRC7
	0	stop bit (1)

Tab. 5.12: Struktura odezvy R2.

Byte	Bit	Význam
1	7	Start bit (0)
	6	Přenosový bit (0)
	5-0	Nevyužito (111111)
2-15		CID nebo CSD registr (data zahrnují CRC7)
16	7-1	
	0	stop bit (1)

Odezva (R3) na příkaz CMD1 obsahuje registr OCR. Tento odpovědní rámec není zabezpečen CRC kódem.

Tab. 5.13: Struktura odpovědního rámce R3.

Byte	Bit	Význam
1	7	Start bit (0)
	6	Přenosový bit (0)
	5-0	Nevyužito (111111)
2-5		OCR registr
6	7-1	Nevyužito (111111)
	0	stop bit (1)

Tab. 5.14: Struktura rámce R6.

Byte	Bit	Význam
1	7	Start bit (0)
	6	Přenosový bit (0)
	5-0	Nevyužito (111111)
2-5		RCA registr
6	7-1	Nevyužito (111111)
	0	stop bit (1)

Proces čtení a zápisu na SD kartu není v SD módu řízen pomocí bitových sekvencí, ale pouze pomocí příkazů. Po zápisu dat karta odešle CRC rámec, v kterém sdělí, zda došlo k chybě (odešle sekvenci „101“) či nikoli (odešle „010“). CRC rámec je odesílán pouze přes pin DAT0. Princip je podobný jako s odesíláním Write status rámce v SPI módu (viz tab. 5.8). V případě čtení z karty jsou data také zabezpečena pomocí kódu CRC16. Read error rámec ani žádný jeho ekvivalent není v SD módu aplikován.

5.2 Připojení SD slotu k MCU

K vývojové desce SMT32F4-Discovery je připojena deska s vstupními audio obvody (viz kap. 3.1) a se dvěma SD sloty. Jeden slot je připojen přes SPI a druhý k rozhraní SDIO využívající čtyřbitový SD mód. Bylo tak zvoleno pro možnost experimentování a ladění obou režimů komunikace s SD kartou.

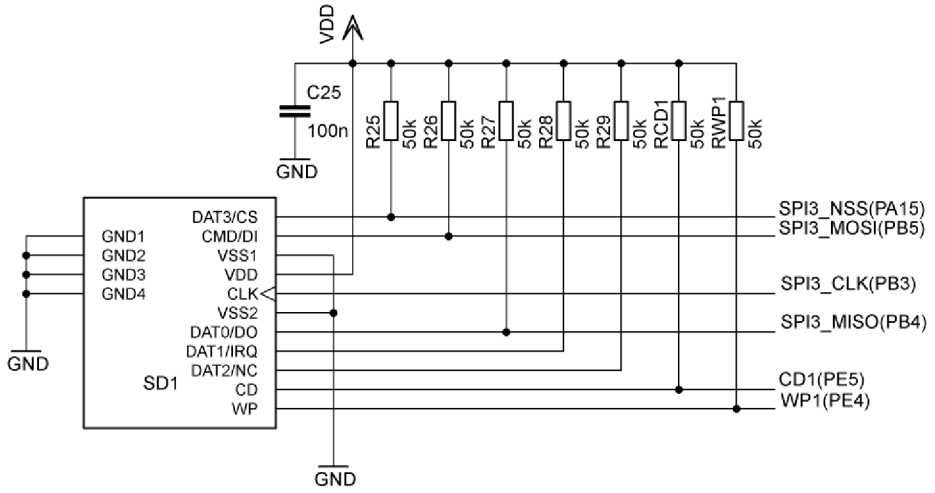
SD slot má oproti samotné SD kartě (piny karty jsou popsány v tab. 5.1) ještě další dva výstupy, a to WP (Write Protect – ochrana proti zápisu realizovaná přepínačem na SD kartě) a CD (Card Detection – detekce karty). CD je vhodné připojit na pin MCU, na kterém je dostupné přerušení. V případě zvoleného procesoru je možné nastavit externí přerušení na jakémkoli I/O pinu.

Všechny datové piny, pin CMD, CD a WP musí být připojeny přes pull-up rezistory. To platí i pro nevyužité datové piny v případě připojení přes SPI. Mohou být použity i vnitřní rezistory MCU. Volba mezi vnitřními a externími pull-up rezistory

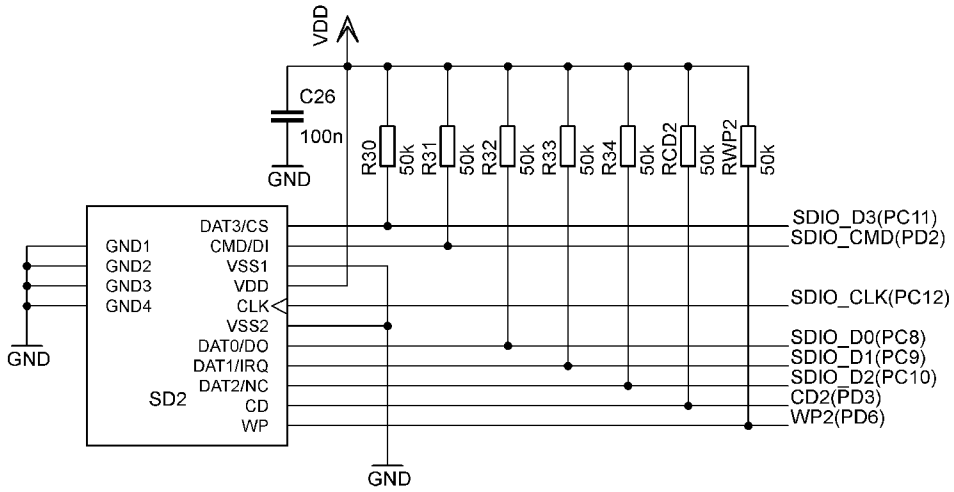
může ovlivnit maximální rychlost zápisu a čtení z karty, proto byly externí pull-up odpory uvažovány při návrhu desky plošných spojů a v případě potřeby mohou být zapojeny.[14][16]

Pro detekci karty existují dvě metody. První (preferovanější) je založena na použití mechanického spínače uvnitř SD slotu. Při vložení karty tento spínač připojí pin CD (card detection) na zem. Tento způsob detekce byl použit pro digitizér. Druhá metoda využívá vnitřní pull-up rezistor připojený k pinu DAT3 SD karty. Není potřeba výstup navíc, ale zaplatí se za to jistými omezeními.[17]

Na následujících dvou obrázcích jsou uvedeny použité způsoby připojení SD karty přes SPI rozhraní (obr. 5.3) a pomocí SDIO rozhraní (obr. 5.4). Ve schématech jsou popsány i jednotlivé piny, na které jsou SD karty připojeny. K SD slotu je nezbytné také připojit blokovací kondenzátor 100 nF.



Obr. 5.3: Připojení SD karty k MCU přes rozhraní SPI.



Obr. 5.4: Připojení SD karty k MCU přes rozhraní SDIO.

Celkové zapojení i s audio obvody a motiv desky plošných spojů je v příloze A.

5.3 Firmware pro SD kartu

Komunikaci s SD kartou řídí driver *stm32f4_discovery_sdio_sd.c* původně určen pro Evaluation vývojovou desku (více popsáno v kap. 4.6). Tento driver obsluhuje funkce pro inicializaci, výběr karty, čtení, zápis, mazání dat, atd., čili posílá příkazy kartě, zpracovává odpovědi a data. Tyto funkce jsou vytvořeny tak, aby byla knihovna univerzální. Proto tyto funkce volají driver *stm32f4xx_sdio.c* ze standardní periferní knihovny pro STM32F4 procesory. Tento soubor už přímo pracuje s registry SDIO periferie.

Driver *stm32f4_discovery_sdio_sd.c* umožňuje čtení a zápis na standardní a SDHC karty. Pro operace s SDHC kartami byly přidány další funkce pro čtení a zápis (*SD_ReadMultiBlocksSDHC()* a *SD_WriteMultiBlocksSDHC()*, jejichž argumentem je adresa, která není v bajtech, ale v blocích. To umožňuje operovat s pamětí větší než 2 GB, což je omezení standardních SD karet. SDHC karty mají kapacitu až 32 GB. Funkčnost driveru byla otestována i s kartami s maximální kapacitou. Autorem tohoto způsobu adresace je uživatel fóra STMicroelectronics Clive1.[21]

Přenos dat z a na SD kartu využívá DMA kontrolér. Pro SDIO periferii jsou určeny stream 3 a stream 6 kanálu 4 DMA2 kontroléru. V aplikaci je použit stream 3. Inicializace DMA je spolu s nastavením SDIO periferie a výstupních pinů procesoru pro SD kartu v knihovně *stm32f4_discovery.c*, která obsahuje také definice a inicializaci periferií na Discovery kitu.

Při vývoji zařízení nebyl uvažován pin SD slotu WP (Write Protect), protože tento pin je pouze na standardním SD slotu, ale v dalším vývoji zařízení bude pravděpodobně použit micro SD slot, který tento pin nemá.

Proces ukládání dat na SD kartu je řízen přes modul FAT, který je součástí projektu. Zápis je více rozeprán v následující kapitole. Pokud je zařízení v režimu čtečky SD karet, je FAT systém obsluhován počítačem a k SD driveru se přistupuje přes USB MSC třídu a soubor *usbd_storage_msd.c*.

6 ZÁPIS DIGITALIZOVANÝCH DAT

Kapitola obsahuje informace o formátu wave pro zvukový záznam a systému souborů FAT (file alocation table), který umožňuje uživateli běžnou práci se složkami a soubory v paměťovém zařízení.

6.1 Zvukový formát wave

Wave je jedním z nejjednodušších formátů zvuku. Digitalizovaný zvuk může být uložen přímo jako PCM, popř. DPCM, ADPCM a další jednoduché formy digitálního zvuku. Zvukový záznam wave může být dále procesorem zpracován či komprimován (MP3).

Zvukový soubor wave začíná hlavičkou o 44 bajtech. Hlavička je rozdělena na tři oddíly (RIFF deskriptor, popis formátu a datový oddíl – obsahuje samotná data uvedená krátkou hlavičkou). Následující tabulka popisuje jednotlivé prvky hlavičky souboru. V posledním sloupci jsou hodnoty použité pro řešenou aplikaci.[18]

Tab. 6.1: Hlavička souboru wave. [18]

Oddíl	Offset	název prvku hlavičky	popis	endianita	hodnota
RIFF deskriptor	0	ChunkID	„RIFF“ v ASCII kódu	big	0x52494646
	4	ChunkSize	velikost souboru bez prvních dvou prvků	little	
	8	Format	„WAVE“ v ASCII	big	0x57415645
formát	12	Subchunk1ID	“fmt “ v ASCII	big	0x666d7420
	16	Subchunk1Size	velikost oddílu formát bez prvních dvou položek	little	16
	20	AudioFormat	formát zvuku (PCM = 1)	little	1
	22	NumChannels	počet kanálů	little	1
	24	SampleRate	vzorkovací frekvence	little	dle volby uživatele (f_{VZ})
	28	ByteRate	datový tok [B/s]	little	1x f_{VZ} x 2
	32	BlockAlign	počet bajtů na jeden vzorek	little	2
	34	BitsPerSample	počet bitů na vzorek	little	16
data	36	Subchunk2ID	„data“ v ASCII	big	0x64617461
	40	Subchunk2Size	velikost samotných dat	little	
	44	data		little	

6.2 Systém FAT

Tento souborový systém byl poprvé představen na začátku osmdesátých let. Od té doby prošel velkým vývojem, během něhož bylo představeno několik verzí tohoto systému. V současnosti nejnovější jsou verze exFAT a FAT+. Ovšem nejrozšířenější verzí stále zůstává FAT32. Tento systém limituje velikost disku na 8TiB a maximální velikost souboru na 4 GB a nepodporuje nejnovější karty SDXC. Pro řešenou aplikaci je vzhledem k velikosti ukládaných dat tento souborový systém dostačující.

FAT je poměrně jednoduchý systém, který podporují téměř všechny operační systémy a je hojně používán pro přenositelná média (kromě CD). Pro tento projekt bude použita verze FAT32. Umístění clusterů je určeno 32-bitovou adresou.

V každém paměťovém médiu využívajícím systém FAT32 je začátek paměti obsazen tzv. Master Boot Recordem (MBR). Je to úplně první sektor fyzické paměti (512 B). Obsahuje bootovací kód. Na konci MBR je popis jednotlivých svazků disku (až 4, ale většinou postačí jeden). Důležitá je hlavně adresa začátku svazku a informace o tom, jaký souborový systém využívá.

Začátek svazku se souborovým systémem FAT32 vypadá následovně.

Tab. 6.2: Rozdělení paměti v médiu se systémem FAT32.

Bootovací sektor	rezervovaná paměť	FAT (primární)	FAT (primární)	Uživatelská data

Bootovací sektor obsahuje základní informace o médiu (typ – přenosný, pevný disk apod.) a základní parametry souborového systému (počet bajtů na sektor, počet sektorů na cluster atd.). Je to první logický sektor svazku.

Paměť je rozdělena na clustery, které jsou zmapovány ve FAT tabulce (je v ní uveden status clusteru a adresa následujícího clusteru). Clustery jsou dále děleny na sektory. Počet sektorů v jednom clusteru je volitelný (1, 2, 4, 8, 16, 32, 64 nebo 128).

Důležitým obsahem paměti je kmenový adresář. Obsahuje informace o všech souborech a adresářích, které jsou na vrcholu souborové hierarchie. Jsou to jméno souboru, velikost v bajtech a adresa prvního clusteru. U starších verzí systému FAT měl kmenový adresář pevné umístění v paměti, ale od verze FAT32 je v paměti fyzicky rozložen do clusterů tak, jako běžné soubory (proto není v tab. 6.2). Díky tomu je jeho velikost pružná.[19]

6.3 Firmware pro zpracování zvuku

Protože formát wave nepoužívá žádnou kompresi dat, je zpracování digitalizovaných dat poměrně jednoduché. Zápis je řízen z hlavního programu a z přerušení od DMA a časovače pro patřičný AD převodník. V souboru *wav_rec.c* jsou funkce pro zpracování a nahrávání zvuku. Mezi nejdůležitější patří

```
void WaveRecStart(void);      //začátek nahrávání
void WaveRecStoring(void);    //funkce pro nahrání bloku dat do souboru
void WaveRecStop(void);       // konec nahrávání, přepsání hlavičky
```

Tyto funkce jsou pouze pro první kanál. Pro druhý mají index 2.

Aby celá aplikace fungovala jako stavový automat, byl celý proces nahrávání rozdělen do uvedených tří funkcí a funkce *WaveRecStoring()* je volána pouze tehdy, je-li připraven blok digitalizovaných dat o velikosti 16 vzorků (32 bajtů). Do souboru jsou ukládány vzorky jako 16-ti bitové, i když rozlišení AD převodníků je pouze 12 bitů. Je vhodnější, když vzorky mají celý počet bajtů.

Při spuštění nahrávání se nastaví potřebné ukazatele a proměnné a vytvoří se hlavička souboru voláním pomocné funkce

```
uint32_t WaveRecHeader(uint8_t* pHeadBuf);  //vytvoření hlavičky
```

Hlavička obsahuje mimo jiné informace o velikosti souboru. Do tohoto pole se zatím uloží nuly. Po ukončení nahrávání se tyto informace přepíší, což zajišťuje funkce *WaveRecStop()*. Všechny parametry, které jsou popsány v hlavičce wave souboru jsou v tab. 6.1.

Za 44 bitovou hlavičku se doplní bity, které vytvoří spolu s hlavičkou blok 512 bitů. Tyto bity už jsou skutečná data a je vhodné je nastavit tak, aby představovaly střední hodnotu digitalizovaného signálu. Tím zabráníme nepříjemnému prasknutí na začátku nahrávky. Střední hodnota je polovina 3,3 V. Po digitalizaci, máme-li 16 bitové vzorky, je to tedy 0x8000.

Pro zápis jsou použity dva buffery, které se po zápisu bloku dat vymění (resp. přepnou se ukazatele na ně. Do jednoho bufferu se zapisují vzorky z AD převodníku v obsluze přerušení DMA a z druhého se data zapisují do souboru na SD kartě. Tento způsob umožňuje to, že se současně pracuje s oběma buffery (pracuje AD převodník i ukládání na kartu) a nedochází k žádné kolizi či přepsání užitečných dat.

Zmíněné funkce v souboru *wav_rec.c* vykonávají také samotný zápis na SD kartu s podporou souborového systému. Tyto záležitosti jsou popsány v následující kapitole.

6.4 Firmware pro zápis s podporou FAT

Pro embedded systémy je zdarma dostupný modul podporující FAT systém [20]. Je vytvořen nezávisle na platformě, takže může být použit v aplikaci s mikrokontrolérem jakékoli architektury. Modul neobsluhuje nižší vrstvy komunikace s paměťovým médiem, což umožňuje implementaci tohoto driveru při použití jakéhokoli paměťového zařízení se systémem FAT. V řešeném projektu jej lze najít pod názvem *ff.c* s hlavičkou *ff.h*.

Modul podporuje souborové systémy FAT12, FAT16 a FAT32. Pro funkci využívá minimálně 6,5 kB paměti při implementaci v ARM-CortexM3 (podle [20]). Otevřené soubory vyžadují samozřejmě další paměťový prostor. Počet otevřených souborů není limitován modulem, ale dostupnou pamětí. Modul umožňuje použití long file name (LFN), ale z důvodu rychlosti aplikace (zejména při vyhledávání v souborech) nebyla tato funkce použita. Názvy souborů mohou mít tedy maximální 11 znaků včetně přípony.

Modul nabízí celou řadu funkcí pro operace se soubory a složkami. Důležité funkce použité v aplikaci jsou uvedené v následující tabulce. Jsou prezentovány příklady použití funkcí.

Tab. 6.3: Důležité funkce podporující FAT systém.

Název funkce a její parametry	Popis
`f_mount( 0, &fatfs );`	inicializace/deinicializace média
`f_open(&file, filename[0], FA_CREATE_ALWAYS | FA_WRITE);`	otevření souboru, lze otevřít s různými parametry (např. pro čtení, zápis, vytvoření souboru)
`f_write (&file, RecBufHeader, 512, (void *)&bytesWritten);`	zápis dat RecBufHeader do otevřeného souboru
`f_mkdir("CHANNEL1");`	vytvoření složky
`f_unlink (filename[0])`	smazání souboru
`f_lseek(&file, 0);`	posuv ukazatele na místo v souboru (0 – začátek souboru)
`f_close (&file);`	zavření souboru

file – ukazatel na soubor

filename[0] – název souboru včetně cesty, definováno jako pole kvůli použití pro více kanálů

Zmíněné funkce jsou volány z knihovny *wav_rec.c*, která obsluhuje i formátování zvuku zmíněné v předchozí kapitole. Při spuštění nahrávání se nejprve vytvoří složky pro jednotlivé kanály (CHANNEL1 a CHANNEL2) do nichž se ukládají soubory wave. Poté se ověří zda je soubor s požadovaným názvem není přítomen na kartě. Jestliže ano, inkrementuje se číslo v názvu souboru. Při nahrávání spuštěném přímo ze zařízení se názvy souborů nastavují od REC0001.WAV do REC1999.WAV. Pokud jsou na kartě soubory zahrnující celý zmíněný rozsah názvů, nahraje se soubor RECLAST.WAV. Ten se vždy při spuštění dalšího nahrávání přepisuje, dokud uživatel nezkopíruje či nesmaže záznamy na kartě. Rozsah názvů je omezen z důvodu rychlosti vyhledávání na kartě. Kontrola přítomnosti souboru s daným názvem na kratě probíhá pomocí *f_open*. Pokud je záznam spuštěn z aplikace v PC, může si uživatel zvolit jakýkoli název.

Po nalezení nepoužitého jména souboru se soubor otevře pomocí *f_open* s parametry FA_CREATE_ALWAYS a FA_WRITE, tedy vytvoří se a otevře pro zápis. Ve funkci *WaveRecStoring()* se postupně zapisují digitalizovaná data pomocí *f_write.* Po ukončení nahrávání se aktualizuje hlavička skutečnou velikostí souboru tak, že se nastaví ukazatel v souboru pomocí *f_lseek* na začátek. Nakonec se soubor zavře pomocí *f_close*.

Jak již bylo řečeno, modul je oddělen od nižších vrstev pro přístup k paměťovému médiu. To jej činí univerzálním, ale současně vzniká nutnost doplnit modul o knihovnu komunikující s paměťovým médiem, tedy s SD kartou. Soubor se v projektu jmenuje *diskio.c*. Obsahuje funkce *disk_initialize*, *disk_status, disk_read, disk_write, disk_ioctl* a *get_fattime*. Tyto funkce komunikují přímo s knihovnou *stm32f4_discovery_sdio_sd.c* pro operace s SD kartou (popsáno v kap. 5) a tím spojují modul FAT s nižšími vrstvami.

Funkce *get_fattime* vrací aktuální datum a čas. Tyto informace se zapisují do souboru po jeho zavření, tedy po ukončení nahrávání. Všechny globální proměnné

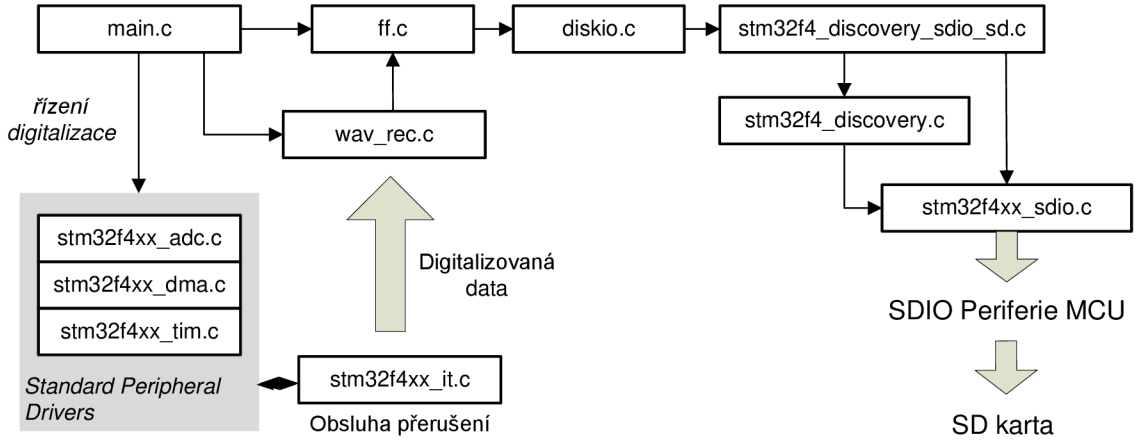
s datem a časem se zapíší do 32 bitové proměnné typu unsigned long. Výsledek vypadá následovně.

Tab. 6.4: Formát data a času pro modul FAT.

bity	31÷25	24÷21	20÷16	15÷11	10÷5	4÷0
význam	počet let od r. 1980	měsíc	den v měsíci	hodina	minuta	sekunda / 2

6.5 Struktura souborů řídících digitalizaci a zápis

Následující obrázek (obr. 6.1) prezentuje strukturu souborů, které řídí zápis na SD kartu a digitalizaci audiosignálu. Šipky označují volání funkcí mezi soubory.



Obr. 6.1: Struktura souborů pro zápis na SD kartu.

7 VÝVOJ APLIKACE PRO MCU

Hlavní soubor firmwaru pro digitizér je *main.c*. Program pro mikrokontrolér byl sestaven jako stavový automat, řízený systémovými hodinami. V přerušení od systémových hodin, které je nastavené na 1 ms, je inkrementována proměnná sys_time, která je používána v jednotlivých funkcích stavového automatu pro řízení času.

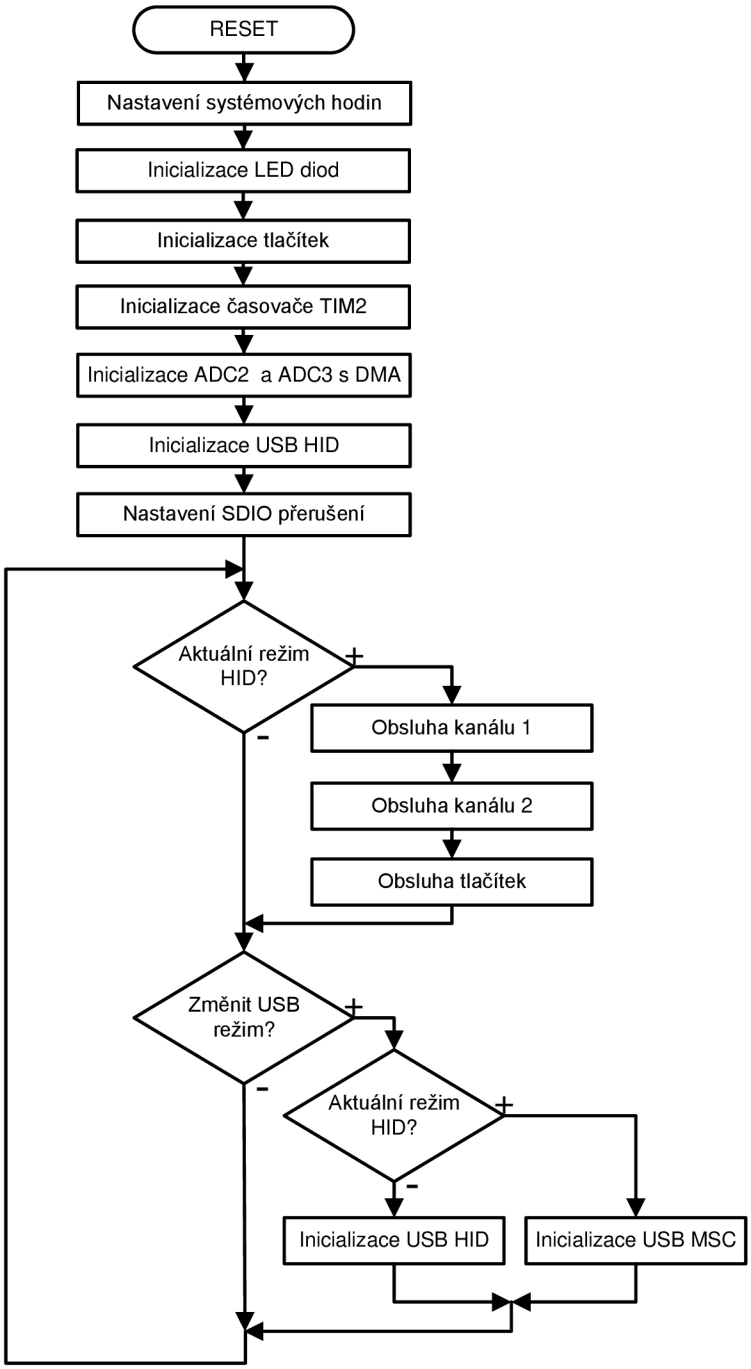
```
void SysTick_Handler(void)
{
  sys_time++;
}
```

Vždy se využívá pouze rozdíl času, takže aktuální hodnota proměnné sys_time není důležitá. Jelikož je sys_time datového typu unsigned long long s 64-bitovou délkou, není možné, aby došlo k přetečení a tím k chybnému výpočtu rozdílu času. Pro zajímavost lze vypočítat, že k přetečení dojde přibližně za 584,5 milionu let běhu aplikace bez resetu.

Na začátku programu je provedena inicializace všech potřebných interních a externích periferií procesoru. Poté program vstoupí do hlavní nekonečné smyčky. Tu opustí při přerušení od SDIO, USB, časovačů, DMA nebo systémových hodin. Hlavní kostru programu popisuje vývojový diagram na obr. 7.1.

Nekonečná smyčka, obsluhující stavový automat, vypadá následovně:

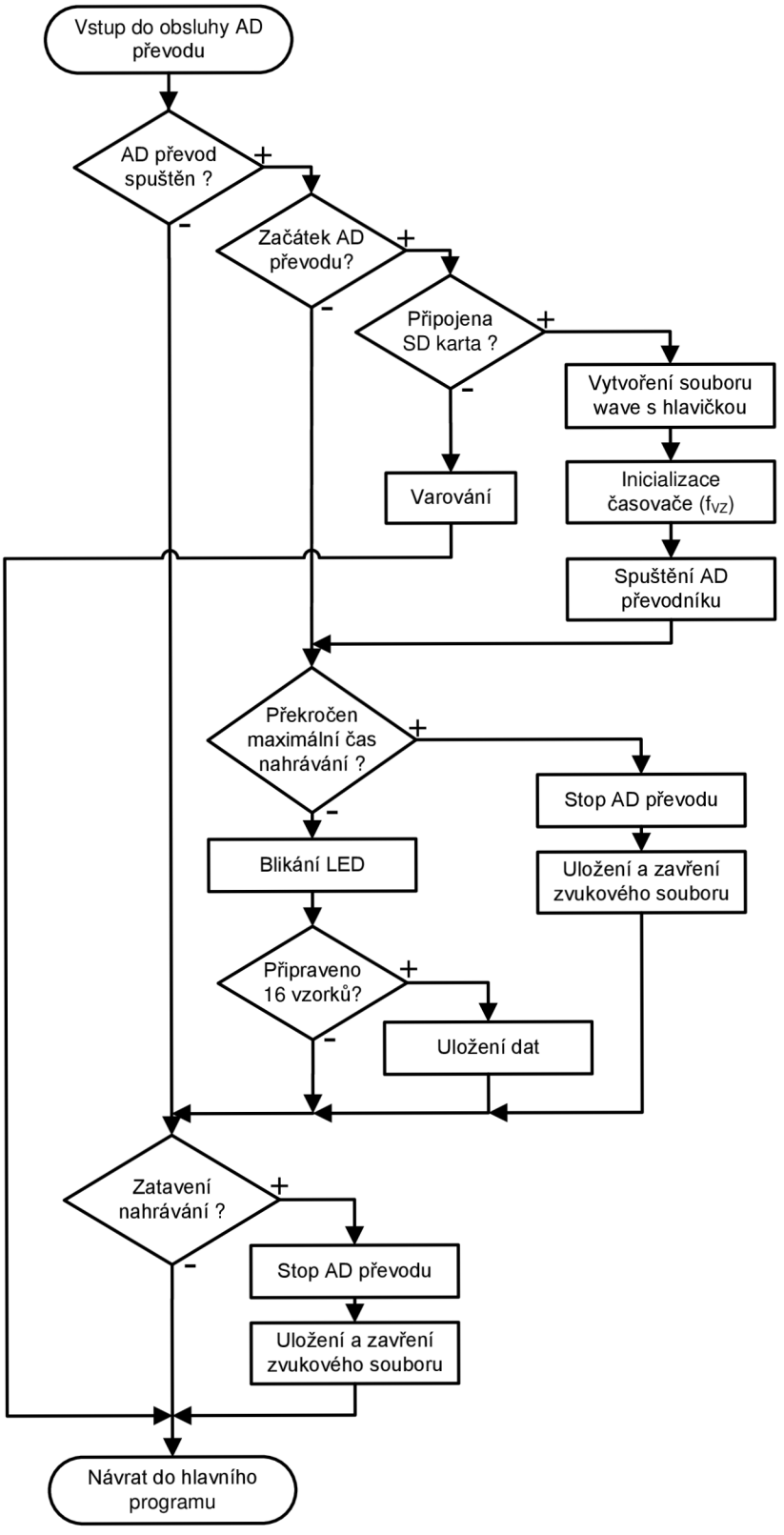
```
while (1)
{
    if(USB_mode == HID)
    {
        ad_conv_ch1();        //ADC kanál 1
        ad_conv_ch2();        //ADC kanál 2
        button1();            //start/stop kanál 1
        button2();            //start/stop kanál 2
    }
    if((adc_enable[0]==0) && (adc_enable[1]==0))
    {
        user_button();        //přepnutí na HID/MSC
    }
}
```



Obr. 7.1: Vývojový digram s inicializací a se zjednodušenou hlavní smyčkou.

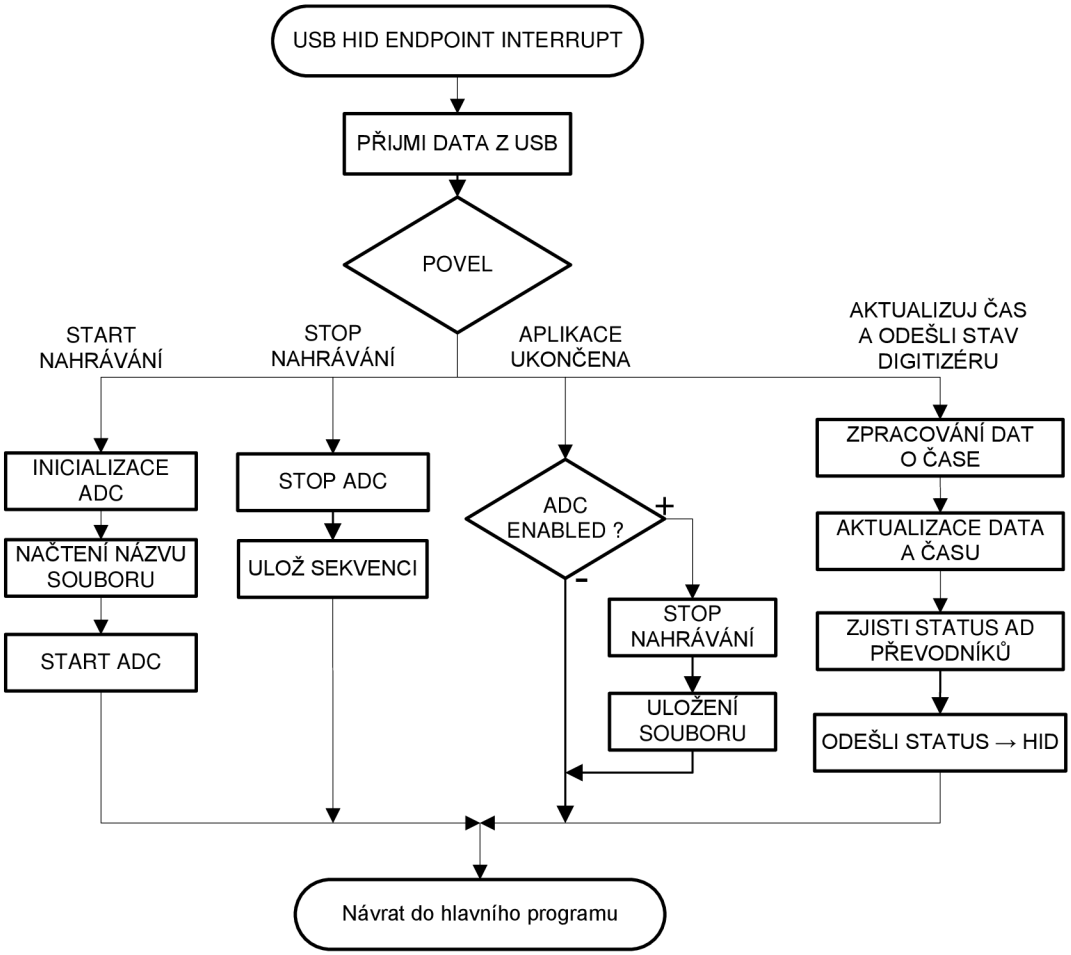
Každé tlačítko je softwarově ošetřeno proti vlivu zákmitů. Filtrační kondenzátory nebyly použity. Samozřejmě při čekání na další zjištění stavu nebo na uvolnění tlačítka program nestojí, ale díky tomu, že funguje jako stavový automat, může během této doby obsluhovat další kód.

Následující diagram popisuje obsluhu jednoho kanálu digitizéru. Druhý je téměř totožný. Samotný AD převod je obsluhován v přerušení, odkud se posílají data k zápisu na SD kartu. Jednotlivé podfunkce z následujícího digramu, které jsou součástí knihovny *wav_rec.c*, jsou podrobněji popsány v kap. 6.



Obr. 7.2: Vývojový diagram obsluhy nahrávání.

Většina informací přenášených přes USB HID třídu směřuje z PC do digitizéru. Opačným směrem jsou posílána pouze některé informace o stavu zařízení, zapnutí či vypnutí nahrávání apod., které slouží pro přizpůsobení aplikace pro ovládání digitizéru. Informace pro zařízení posílaná z PC (aplikací PC_HID_record.exe) a jejich zpracování je vhodné ukázat na vývojovém diagramu. Zpracování přijatých dat, které provádí callback funkce USBD_HID_DataOut, zjednodušeně prezentuje obr. 7.3. Povely jsou pouze pro jeden kanál digitizéru. Zpracování je větveno pomocí příkazu switch.

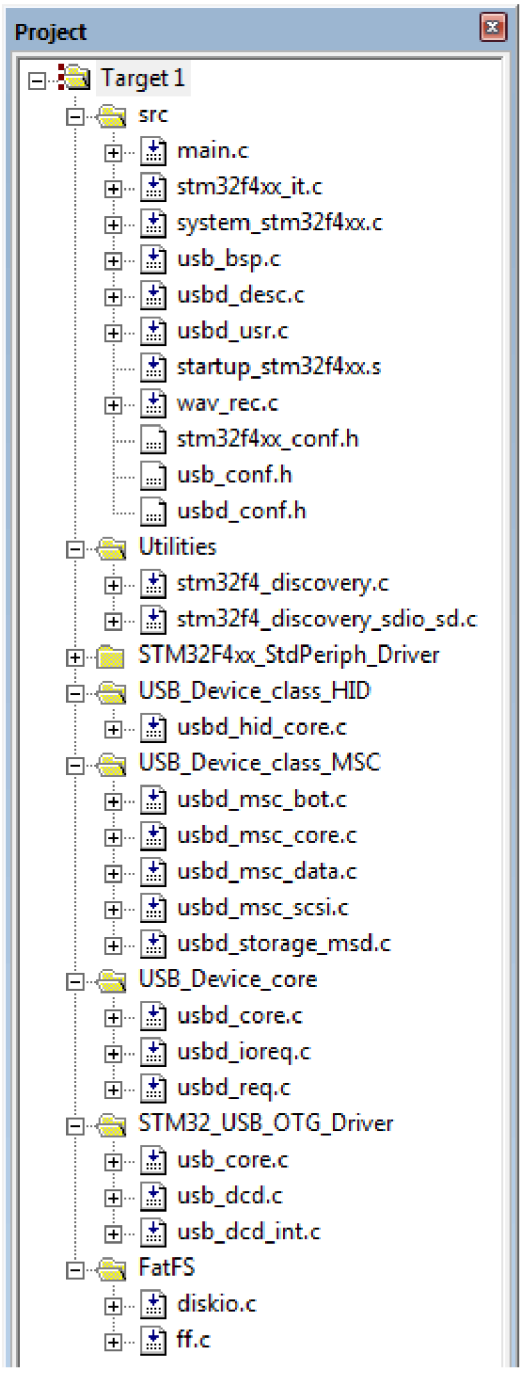


Obr. 7.3: Zpracování přijatých dat na přes USB HID.

Protože při uložení souboru se do něj ukládají informace o čase (viz kap. 6.4), je vhodné implementovat do firmwaru nějaké hodiny. RTC (Real Time Clock) využit nebyl. Bylo zvoleno jednodušší řešení s časovačem TIM2, který měří čas po resetu od 00:00 1.1.2013. Po připojení USB a spuštění aplikace pro ovládání digitizéru se ihned datum a čas aktualizuje podle času v PC. Toto řešení je dostačující, protože informace o čase jsou využity pouze pro vlastnosti nahraných souborů. Bez implementace hodin do firmwaru by soubory obsahovaly čas vytvoření 00:00 1.1.1980. Více o aktualizaci času je v kap. 4.5.

Firmwaru pro digitizér byl zpočátku vyvíjen v prostředí Attolic. V průběhu práce byl tento software nedostačující kvůli jistým omezením, zejména délky kódu. Proto

další vývoj pokračoval v prostředí µVision od fy Keil. Kompletní projekt pro zařízení vytvořený v prostředí µVision je na přiloženém CD. Následující obrázek ukazuje strukturu souborů v projektu. Hlavičkové soubory (kromě konfiguračních, které nemají přidružený .c soubor) jsou skryté. Je vidět, že USB komunikaci je díky její komplexnosti věnována většina souborů. Složka STM32F4xx_StdPeriph_Driver obsahuje základní drivery pro většinu periferií na čipu kontroléru.



Obr. 7.4: Organizace souborů v µVision projektu pro digitizér.

Velikost kódu firmwaru pro MCU bez optimalizace je 46,2 kB.

8 VLASTNOSTI A OVLÁDÁNÍ DIGITIZÉRU

V této kapitole je stručně popsáno ovládání digitizéru (tedy prototypu s vývojovou deskou Discovery) a jeho parametry.

V režimu, kdy se digitizér chová jako čtečka karet, byla změřena rychlost přenosu dat. Karta pro testování měla poměrně dobré parametry, co se týče rychlosti, takže maximální rychlost byla dána zařízením.

Tab. 8.1: Maximální rychlost zápisu a čtení z SD karty.

Zápis	600 kB/s
Čtení	780 kB/s

Pro testování nahrávání byl použit jako zdroj audiosignálu audio výstup počítače. Propojení bylo realizováno pomocí kabelu s redukcí na konektory CINCH, které jsou připojeny ke vstupům obou kanálů. V tomto případě se jedná v podstatě o levý a pravý kanál stereo signálu. Vstupní audio obvody jsou napájeny 12 V externím zdrojem. Zbytek digitizéru je napájen 5 V přes USB. Toto dvojí napájení je pouze pro testování samotného digitizéru, protože vstupní audio obvody v kompletním monitorovacím přijímači jsou řešeny jinak.

Při testování je vhodné nastavit výstupní hlasitost PC přibližně na 20 %, aby se vstup zesilovače nepřebudil (při 100 % hlasitosti PC je výstupní signál 3,5 V_{PP}). Zesílení vstupního zesilovače bylo nastaveno přibližně 3x (potenciometr na 20 kΩ). Nahrávání bylo testováno pro oba kanály (také současně). Nahraný zvukový signál obsahuje určitou úroveň šumu, který vzniká ve vstupních analogových obvodech, které jsou pouze pro testování. Co se týče digitalizace a nahrávání na kartu, vše funguje bezproblémově.

Pro ovládání zařízení z PC je třeba připojit zařízení přes USB micro konektor. Ovládání aplikace je intuitivní.

Digitizér je možné ovládat i pomocí tlačítek na zařízení. Přepnutí režimu USB (HID/MSC) je možné dlouhým stiskem modrého tlačítka na desce Discovery. Zařízení bylo doplněno v průběhu vývoje o další čtyři tlačítka. Dvě slouží pro ovládání nahrávání zvuku (každé pro jeden kanál). Další dvě byly přidány pro ovládání přehrávání, které bude pravděpodobně implementováno v dalším vývoji. Tlačítka jsou na malé desce plošných spojů, která je připevněna k vývojové desce Discovery pomocí pinů. Veškerá dokumentace k tomuto doplňku jsou v příloze C.

Fotografie kompletního zařízení s deskou Discovery je v příloze D.

9 ZÁVĚR

Cílem práce byla kompletní realizace digitizéru, který je součástí monitorovacího přijímače pro letecké pásmo. Požadavkem byl návrh kompletního schéma digitizéru s vhodným mikrokontrolérem, realizace digitizéru, jeho oživení a vytvoření firmwaru pro MCU a softwaru pro ovládání.

Práce je rozdělena tématicky do několika kapitol. K dané problematice je vždy v úvodu kapitoly uvedena teorie a poté je diskutován návrh hardwaru či firmwaru pro danou periferii nebo část zařízení.

V první části práce je diskutována volba MCU. Při výběru hrála velkou roli dostupnost vývojového prostředí pro MCU. Dále bylo navrženo blokové schéma ukazující způsob řešení s vývojovým kitem STM32F4-Discovery, jehož součástí je zvolený MCU od firmy STMicroelectronics. Tato vývojová deska byla použita jako základ pro vývoj digitizéru. Ke kitu bylo nutné navrhnout přídavný modul. První částí tohoto modulu jsou vstupní audio obvody, které zahrnují zesilovač, anti-aliasingový filtr a další komponenty. Filtr byl navržen jako kaskádní, simulován v programu PSpice a jeho funkce ověřena měřením. Další částí modulu je připojení SD slotů. Byly realizovány dva způsoby připojení, a to přes SPI a SDIO rozhraní mikrokontroléru. V průběhu vývoje se ukázalo, že je vhodnější použít SDIO připojení. Komunikace přes SPI nebyla realizovaná. Modul umožňuje připojení externích pull-up rezistorů, ale zařízení bylo realizováno s vnitřními.

V další části práce je popsáno řešení USB komunikace s počítačem. USB periferie je součástí Discovery kitu. USB může komunikovat pouze ve Full-Speed módu, což není příliš velkým nedostatkem pro digitizér. Snahou bylo vytvořit USB composite device, který umožní ovládání digitizéru přes HID třídu se současným přístupem na SD kartu díky Mass Storage třídě. USB třídy byly nejprve realizovány samostatně a poté implementovány do složeného zařízení USB. Zařízení s USB composite device nepracovalo správně. Mass Storage třída ovlivňovala ostatní součásti programu. Proto bylo realizováno alternativní řešení s přepínanými třídami HID a Mass Storage. Není tedy možné současně přistupovat z počítače k souborům na kartě a ovládat zařízení pomocí aplikace v PC.

Pro ovládání zařízení z počítače byla vytvořena aplikace v jazyce C#. Pro vývoj bylo použito MS Visual Studio. Aplikace poskytuje uživateli základní volby pro ovládání dvou kanálů pro záznam zvuku. Navíc program aktualizuje čas v digitizéru, který je použit při ukládání parametrů nahraných souborů.

Digitalizace je realizována pomocí dvou 12-bitových AD převodníků, které jsou řízeny časovači. Pro přenos digitalizovaných dat je využit DMA kontrolér. Přenos dat mezi SD kartou a MCU využívá taktéž DMA. Firmware pro obsluhu zápisu využívá knihovny pro SDIO rozhraní a pro komunikace s SD kartou, které jsou k dispozici od firmy ST. Zápis na SD kartu řízený mikrokontrolérem podporuje souborový systém FAT32. V případě přístupu na kartu z počítače, FAT systém řídí počítač. Digitalizovaný audiosignál je nahrán ve formátu wave. Na SD kartě jsou data ukládána do složek podle kanálu digitizéru.

Funkčnost digitizéru byla otestována. Vlastnosti zařízení splňují požadavky zadání.

Práce prezentuje příklad použití ARM procesorů obsahujících běžně mnoho periferií, které jsou výhodou při vývoji jakýchkoli embedded aplikací. Řada procesorů STM32F4 nabízí nejvýkonnější 32-bitové procesory od fy STMicroelectronics. V následujícím vývoji je počítáno s implementací kodéru a dekodéru MP3, tudíž výkon MCU bude plně využit. Firma ST podporuje vývoj embedded aplikací se svými produkty poskytováním knihoven pro různé periferie, publikováním příkladů aplikací, poměrně dobrou technickou dokumentací a v neposlední řadě velkou nabídkou vývojových desek. Jsou to zejména řady Discovery (obsahují pouze několik základních periferií a programátor) a Evaluation (obsahuje více komponent). Discovery kity jsou velice levné, často je lze pořídit za přibližně stejnou cenu jako samotný procesor. Při vývoji digitizéru byla využita deska STM32F4-Discovery doplněná o navržené potřebné periferie na doplňujícím modulu. Při tvorbě firmwaru byly využity dostupné drivery pro periferie MCU poskytované firmou ST. Nicméně použitá knihovna pro USB má určité nedostatky. Autoři knihovny uvádějí, že lze s knihovnou vytvářet složená zařízení USB. Během vývoje však bylo třeba modifikovat i soubory obsluhující USB na nízké úrovni. Podobné nedostatky knihoven jsou dány tím, že zvolený procesor je na trhu poměrně nový.

LITERATURA

[1] STMicroelectronics. www.st.com

[2] Atollic. www.atollic.com

[3] *STM32F407xx*. Datasheet, STMicroelectronics (www.st.com), Document ID 022152, Rev 2, 2012, 167 s.

[4] *RM0090 Reference Manual.* STMicroelectronics (www.st.com), Document ID 018909, Rev 1, 2011, 1316 s.

[5] *Začínáme s STM32F4 Discovery 7* [online]. [cit. 2012-03-14]. MCU.CZ, ISSN 1213-8045. Dostupné z www: <http://mcu.cz/comment-n2848.html>

[6] VRBA, R. *Teorie vzájemného převodu analogového a číslicového signálu.* Skripta, VUT v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010, 139 s.

[7] *Active Filter Design Application* [online]. [cit. 2012-04-03]. Texas Instruments, Dostupné z www: < http://www.ti.com/tool/filterpro>

[8] AXELSON, J. *USB Complete – The Developer's Guide, Fourth Edition*. Lakeview Research LLC, ISBN13 978-1-931448-08-6, 2009, 504 s.

[9] STEVENS, C. *USB Attached SCSI Protocol (UASP)*. USB Implementers Forum, Inc., Revision 1.0, červen 2009, 27 s.

[10] LEITNER, F. *HID USB Driver / Library for .Net / C#,* [online]. [cit. 2012-10-25]. Dostupný z www: <http://www.florian-leitner.de/index.php/2007/08/03/hid-usb-driver-library/>

[11] *STM32F4 high-performance discovery board, user manual UM1472*. STMicroelectronics (www.st.com), Document ID 022256, Rev 2, 2012, 38 s.

[12] *On-The-Go host and device library, user manual UM1021.* STMicroelectronics (www.st.com), Document ID 18153, Rev 3, květen 2012, 107 s.

[13] SanDisk. *SanDisk SD Card Prouct Manual.* Version 2.2, Document No. 80-13-00169, listopad 2004, 116 s.

[14] FOUST, F. *Secure Digital Card Interface for the MSP430*. Application Note. East Lansing: Michigen State University, 2004, 23 s.

[15] SD Card Association. <www.sdcard.org>

[16] *SD Memory Card Specifications, Part 1: PHYSICAL LAYER SPECIFICATION*. SD Group, Version 1.0, květen 2000, 117 s.

[17] *SD(HC)-memory card and MMC interface conditioning.* Application note. NXP Semiconductors, Rev. 1, 29.4.2010, 37 s.

[18] WILSON, S. J. *WAVE PCM soundfile format.* [online]. [cit. 2012-11-10]. 2003, Dostupný z www: <https://ccrma.stanford.edu/courses/422/projects/WaveFormat/>

[19] *Microsoft Extensible Firmware Initiative FAT32 File System Specification.* Microsoft Corporation, 2000, version 1.03, 34 s.

[20] *FatFs - Generic FAT File System Module.* [online]. [cit. 2013-03-12]. Dostupný z www: <http://elm-chan.org/fsw/ff/00index_e.html>

[21] *STM32F4-Discovery USB Mass storage.*. STe2eCommunities, STM forum, [online]. [cit. 2013-04-05]. Dostupný z www: <https://my.st.com/public/STe2ecommunities/mcu/Lists/STM32Discovery/DispForm.aspx?ID=2842&RootFolder=%2Fpublic%2FSTe2ecommunities%2Fmcu%2FLists%2FSTM32Discovery%2FSTM32F4-Discovery%20USB%20Mass%20storage>

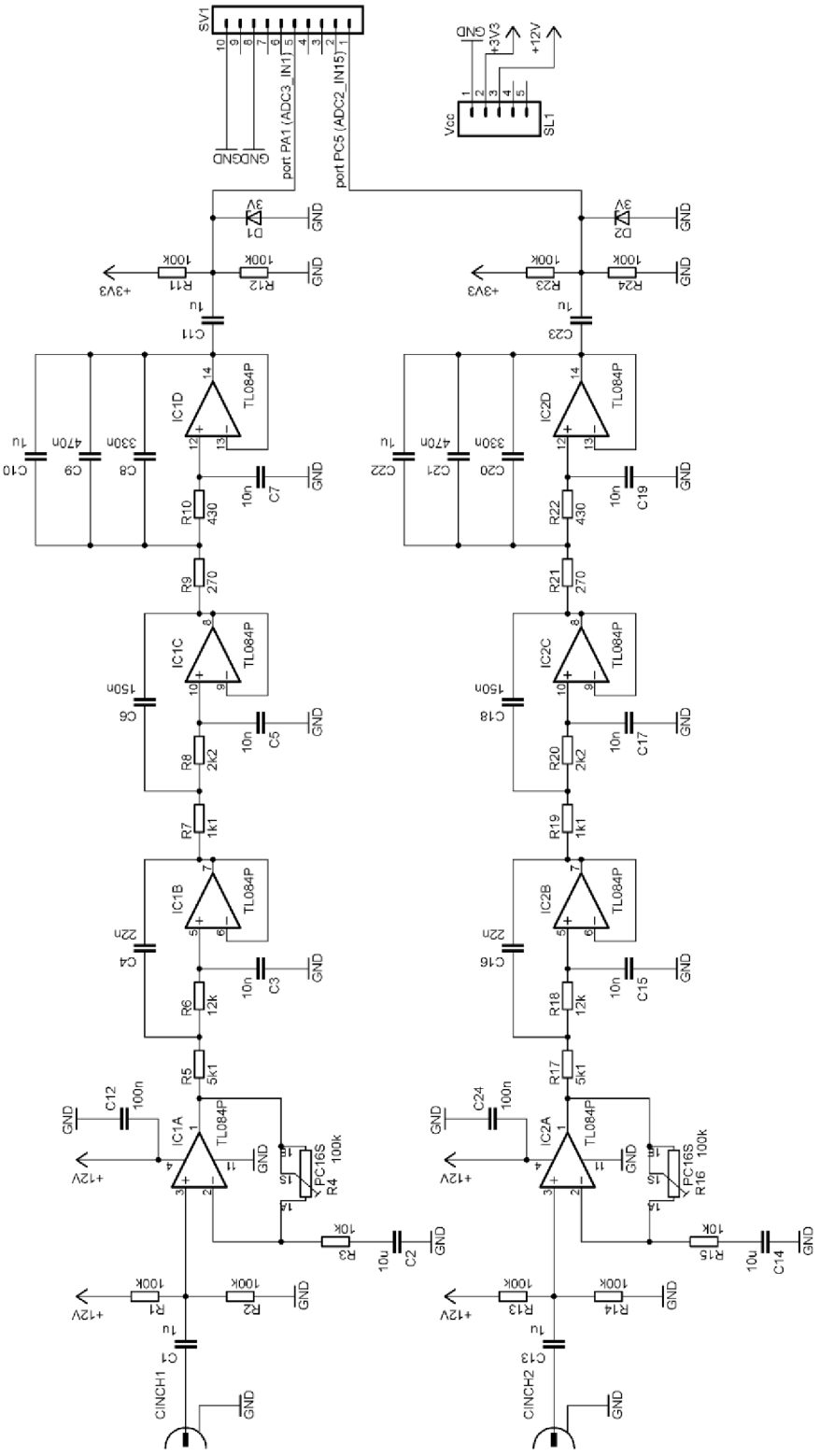
SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ADC	Analogově-digitální převodník
BOT	Bul Only Transport
BSP	Board Support Package
DAC	Digitálně-analogový převodník
DCD	Device Controller Driver
ECC	Error Correction Code
FAT	File Allocation Table
HID	Human Interface Device
JTAG	Joint Test Action Group
MCU	Microcontroller Unit (Mikrokontrolér)
MMC	Multi Media Card
MSC	Mass Storage Class (třída Mass Storage)
OCR	Operating Conditions Register
SCSI	Small Computer System Interface
SD	Secure Digital
SDIO	SD Input/Output
SPI	Serial Peripheral Interface
SWD	Serial Wire Debug
USB	Universal Serial Bus
V_{PP}	Napětí špička-špička
V_{REF}	Referenční napětí pro AD převodník

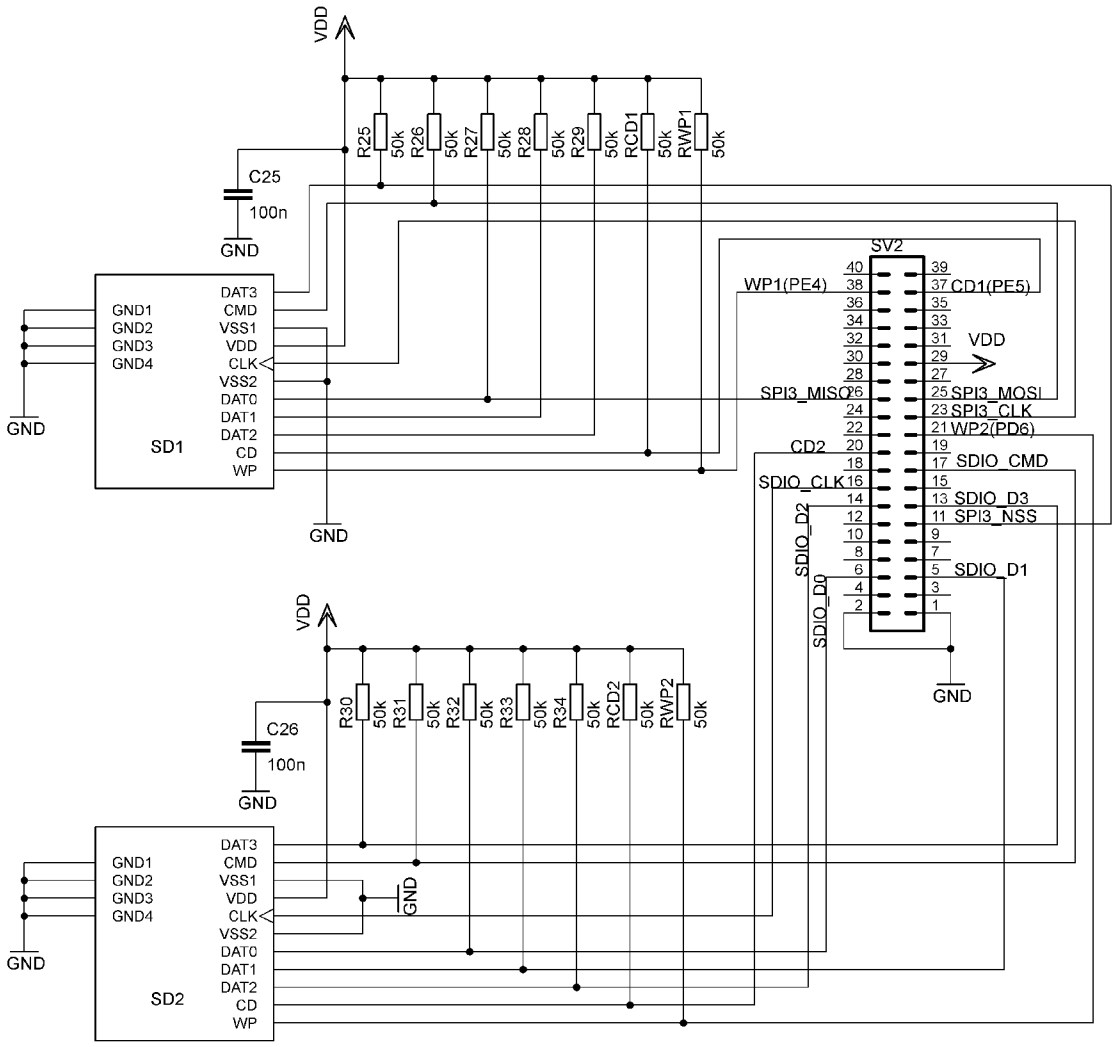
SEZNAM PŘÍLOH

A NÁVRH ZAŘÍZENÍ K VÝVOJOVÉ DESCE

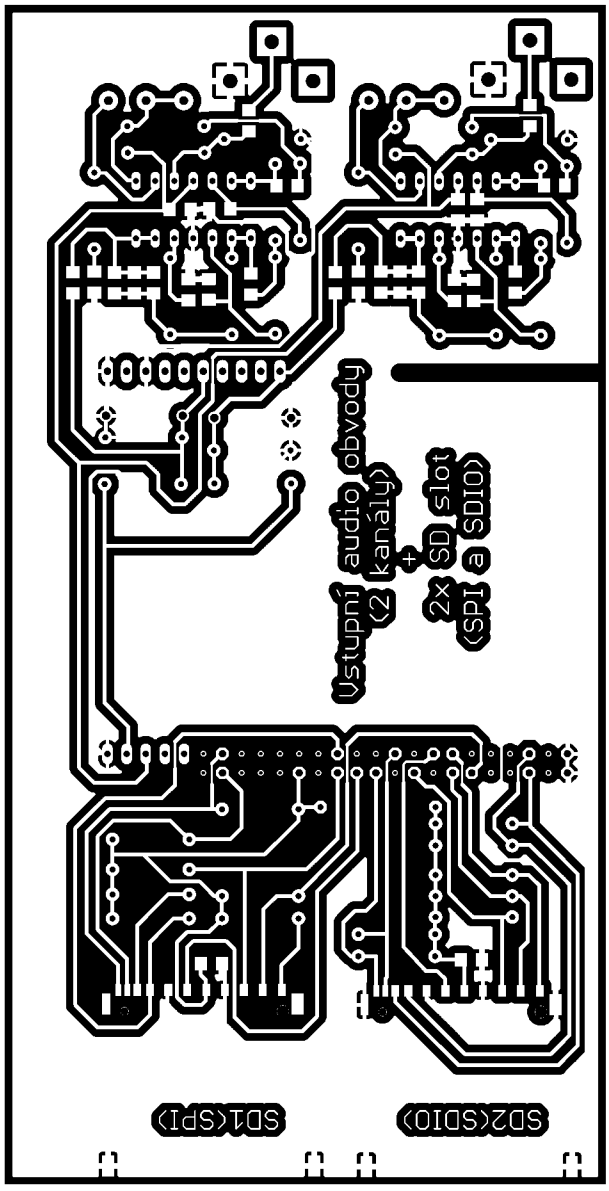
A.1 Schéma vstupních audio obvodů



A.2 Schéma zapojení SD slotů k MCU

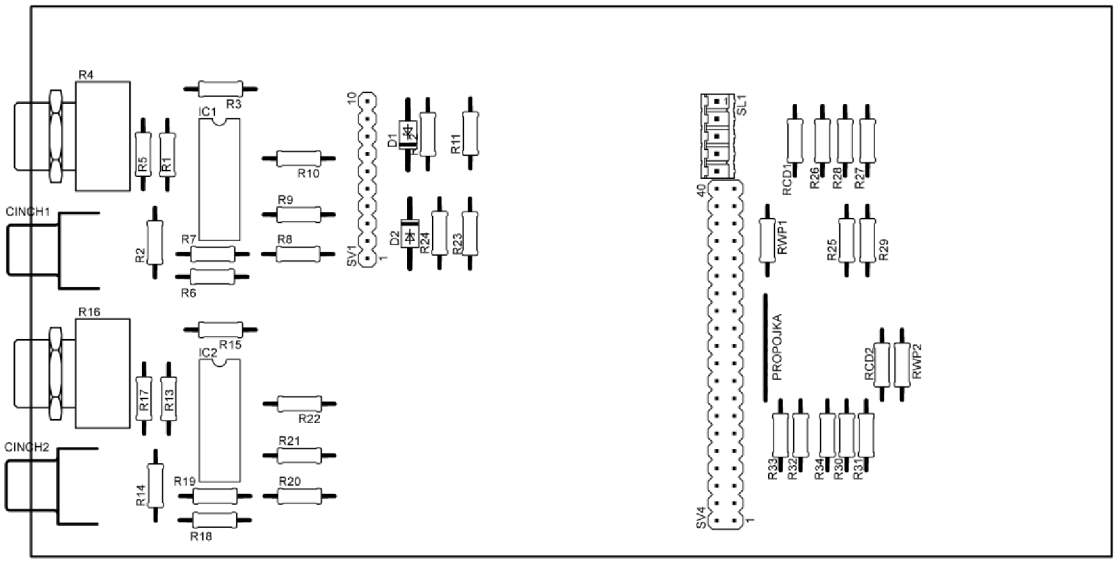


A.3 Deska plošného spoje

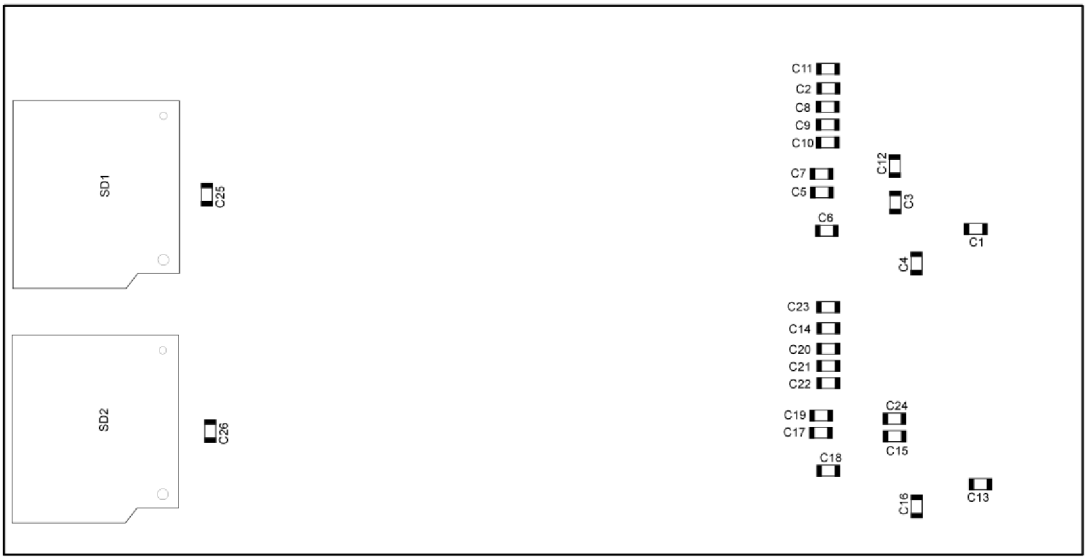


Rozměr desky 157 x 80 [mm], měřítko M 1:1

A.4 Osazovací nákres – top



A.5 Osazovací nákres – bottom



A.6 Seznam součásek

Součástka	Hodnota	Pouzdro	Součástka	Hodnota	Pouzdro
C1	1u	C1206	R13	100k	0207/10
C2	10u	C1206	R14	100k	0207/10
C3	10n	C1206	R15	10k	0207/10
C4	22n	C1206	R16	100k	PC16S
C5	10n	C1206	R17	5k1	0207/10
C6	150n	C1206	R18	12k	0207/10
C7	10n	C1206	R19	1k1	0207/10
C8	330n	C1206	R20	2k2	0207/10
C9	470n	C1206	R21	270	0207/10
C10	1u	C1206	R22	430	0207/10
C11	1u	C1206	R23	100k	0207/10
C12	100n	C1206	R24	100k	0207/10
C13	1u	C1206	R25	50k	0207/10
C14	10u	C1206	R26	50k	0207/10
C15	10n	C1206	R27	50k	0207/10
C16	22n	C1206	R28	50k	0207/10
C17	10n	C1206	R29	50k	0207/10
C18	150n	C1206	R30	50k	0207/10
C19	10n	C1206	R31	50k	0207/10
C20	330n	C1206	R32	50k	0207/10
C21	470n	C1206	R33	50k	0207/10
C22	1u	C1206	R34	50k	0207/10
C23	1u	C1206	RCD1	50k	0207/10
C24	100n	C1206	RCD2	50k	0207/10
C25	100n	C1206	RWP1	50k	0207/10
C26	100n	C1206	RWP2	50k	0207/10
R1	100k	0207/10	SD1		SD030
R2	100k	0207/10	SD2		SD030
R3	10k	0207/10	SL1		BL805G
R4	100k	PC16S	SV1		BL810G
R5	5k1	0207/10	SV2		BL840GD
R6	12k	0207/10	CINCH1		736880-49
R7	1k1	0207/10	CINCH2		736880-49
R8	2k2	0207/10	D1	3V	DO41Z10
R9	270	0207/10	D2	3V	DO41Z10
R10	430	0207/10	IC1	TL084P	DIL14
R11	100k	0207/10	IC2	TL084P	DIL14
R12	100k	0207/10			

B USB DESKRIPTORY

B.1 Konfigurační deskriptor (HID + Mass Storage)

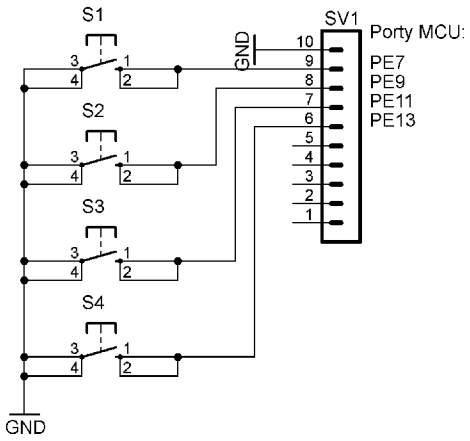
Obsahuje Interface deskriptory pro HID a Mass Storage a Endpoint deskriptory.

```
__ALIGN_BEGIN static uint8_t USBD_HID_CfgDesc[USB_HID_CONFIG_DESC_SIZ]
__ALIGN_END =
{
  0x09, /* bLength: velikost konfiguračního deskriptoru */
  0x02, /* bDescriptorType: konfigurační deskriptor */
  64,   /* wTotalLength: velikost celého konfiguračního deskriptoru */
  0x00,
  0x02, /* bNumInterfaces: 2 interface deskriptory */
  0x01, /* bConfigurationValue: konfigurační hodnota */
  0x00, /* iConfiguration: index řetězce popisujícího konfiguraci*/
  0xE0, /* bmAttributes: Bus powered and Support Remote Wake-up */
  0x32, /* MaxPower 100mA: maximální proud */

  /*********    INTERFACE 0:  HID interface ***************/
  0x09, /* bLength: velikost HID interface descriptoru */
  0x04, /* bDescriptorType: Interface deskriptor */
  0x00, /* bInterfaceNumber: Označení tohoto inteface deskriptoru */
  0x00, /* bAlternateSetting */
  0x02, /* bNumEndpoints: Počet endpointů pro tento interface */
  0x03, /* bInterfaceClass: třída HID */
  0x00, /* bInterfaceSubClass : 1=BOOT, 0=no boot */
  0x00, /* nInterfaceProtocol : 0=žádný protokol,1=klávesnice,2=myš*/
  0,    /* iInterface: Index řetězce popisujícího HID interface */
  /*******    Deskriptor Custom HID   ********************/
  0x09, /* bLength: velikost deskriptoru Custom HID */
  0x21, /* bDescriptorType: HID deskriptor */
  0x10, /* bcdHID: specifikace HID verze 1.10 */
  0x01,
  0x00, /* bCountryCode: kód cílové země - nedefinováno */
  0x01, /* bNumDescriptors: počet HID deskriptorů */
  0x22, /* bDescriptorType: Report deskriptor */
  120,  /* wItemLength: celková délka Report deskriptoru */
  0x00,
  /********   HID endpoint deskriptory ***************/
  /* VSTUPNÍ EP */
  0x07, /* bLength: velikost endpoint deskriptoru */
  0x05, /* bDescriptorType: Endpoint deskriptor   */
  0x81, /* bEndpointAddress: adresa endpointu (vstupní EP číslo 1) */
  0x03, /* bmAttributes: Interrupt endpoint (interrupt trasfer) */
  2,    /* wMaxPacketSize: maximální velikost paketu pro tento EP */
  0x00,
  0x20, /* bInterval: Polling interval (32 ms) */
  /* VÝSTUPNÍ EP */
  0x07, /* bLength: velikost endpoint deskriptoru */
  0x05, /* bDescriptorType: Endpoint deskriptor   */
  0x01  /* bEndpointAddress: adresa endpointu (výstupní EP číslo 1) */
  0x03, /* bmAttributes: Interrupt endpoint (interrupt trasfer) */
  16,   /* wMaxPacketSize: maximální velikost paketu (z PC se posílá
  0x00,    více informací) */
  0x20,     /* bInterval: Polling interval (32 ms) */
```

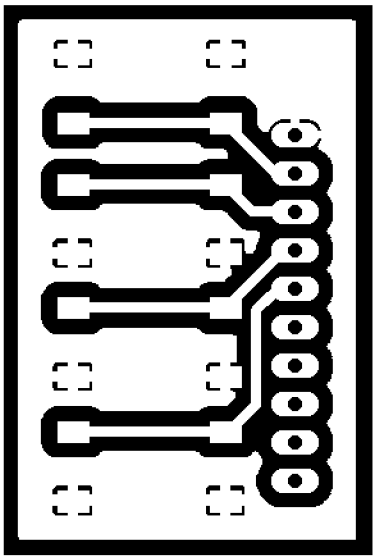
```
/********    INTERFACE 1:  Mass Storage interface **************/
  0x09,   /* bLength: velikost HID interface descriptoru */
  0x04,   /* bDescriptorType: Interface deskriptor */
  0x01,   /* bInterfaceNumber: Označení tohoto inteface deskriptoru */
  0x00,   /* bAlternateSetting */
  0x02,   /* bNumEndpoints: Počet endpointů pro tento interface */
  0x08,   /* bInterfaceClass: třída Mass Storage  */
  0x06,   /* bInterfaceSubClass : podtřída - SCSI transparent */
  0x50,   /* nInterfaceProtocol: Mass Storage Bulk only transport */
  0x05,   /* iInterface: Index řetězce popisujícího MSC interface */
  /********************  Mass Storage Endpoint deskriptory *********/
  /* VSTUPNÍ EP */
  0x07,   /* velikost endpoint deskriptoru */
  0x05,   /* bDescriptorType: Endpoint deskriptor   */
  0x82,   /*Endpoint address: adresa endpointu (vstupní EP číslo 2) */
  0x02,   /*Bulk endpoint */
  0x40,    /* wMaxPacketSize: maximální velikost paketu */
  0,
  0x00,   /*Polling interval se pro MSC nedefinuje */
  /* VÝSTUPNÍ EP */
  0x07,   /* velikost endpoint deskriptoru 7 */
  0x05,   /* bDescriptorType: Endpoint deskriptor   */
  0x02,   /*Endpoint address: adresa endpointu (vstupní EP číslo 2) */
  0x02,   /*Bulk endpoint */
  0x40,    /* wMaxPacketSize: maximální velikost paketu */
  0,
  0x00     /* Polling interval se pro MSC nedefinuje */
};
```

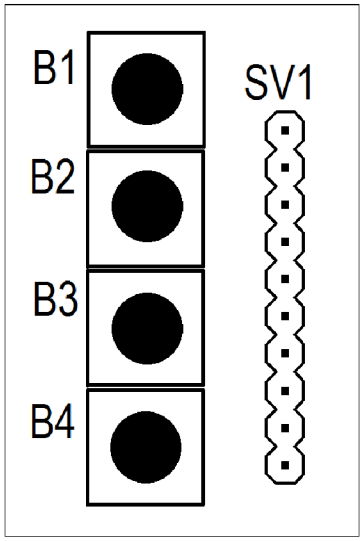
C EXTERNÍ TLAČÍTKA

C.1 Schéma připojení tlačítek k MCU



C.2 Deska plošných spojů (top) a osazovací nákres (top)





Rozměr desky 36 x 25 [mm]
měřítko M 2:1

Seznam součástek:

Součástka	Popis
B1 ÷ B4	mikrospínač SMD
SV1	BL810G

D FOTOGRAFIE ZAŘÍZENÍ