VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Modul USB FM rádia

USB FM Radio Modul

2016

Bc. Pavel Kovář

Tuto stránku nahradíte v tištěné verzi práce oficiálním zadáním Vaší diplomové či bakalářské práce.

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava*.

V Ostravě 27. dubna 2016 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 27. dubna 2016 .....

Rád bych na tomto místě poděkoval zejména mému vedoucímu cenné rady a trpělivost.

Abstrakt

Tato práce popisuje návrh USB FM přijímače se dvěma tunery. Jeden tuner slouží pro přehrávání zvuku a druhý pro vyhledávání dalších stanic. přijímač je v systému reprezentován jako USB zvuková karta.
Příjem je realizován dvojicí integrovaných obvodů Si4735-DU. Tyto jsou přes I^2S a I^2C spojeny s MCU PIC32MX250F128B, který přes USB zajišťuje komunikaci s počítačem. V rámci firmware MCU je, po neúspěchu s Microchip harmony frameworkem, napsán vlastní USB stack.
Knihovna je napsána v jazyku C s využitím knihovny libusb. Poskytuje funkce pro tři úrovně přístupu k tunerům.
Demonstrační aplikace je ve formě grafického uživatelského rozhraní, napsaná v C++ s využitím QT frameworku.
Vše je funkční pod OS Linux i Windows.

Klíčová slova: FM rádio, USB, RDS, QT, libusb, PIC

Abstract

This work describes design of USB FM radio receiver with two tuners. One tuner is for radio playback, second one seeks new stations. In computer, device acts as sound card.
Receiving is done by couple of Si4735-DU integrated circuits, which are connected to MCU via I^2C and I^2S . MCU forwards data over USB to computer and back. Use of Microchip harmony framework was not successful so in firmware is USB stack written from scratch.
Library is written in C with use of libusb library. There are three levels of functions to access tuners.
Demo application has graphical user interface and is written in C++ in QT framework.
All works under Linux and Windows.

Keywords: FM radio receiver, USB, RDS, QT, libusb, PIC

Seznam použitých zkratek a symbolů

AM	–	Amlitudová Modulace (Rozhlasové vysílání v pásmu dlouhých vln)
CD	–	Compact disc
DAB	–	Digital Audio Broadcasting (Digitální pozemní rozhlasové vysílání)
DIP	–	Dual Inline Package
FM	–	Rozhlasové vysílání v pásmu velmi krátkých vln
I^2C	–	Inter-Integrated Circuit
I^2S	–	Integrated Interchip Sound
LW	–	Long Waves (Rozhlasové vysílání v pásmu dlouhých vln)
MCU	–	Microcontroller unit
PCM	–	Pulse-code modulation
RDS	–	Radio Data System
SPI	–	Serial Peripheral Interface
SSOP	–	Shrink Small-Outline Package
SW	–	Short Waves (Rozhlasové vysílání v pásmu krátkých vln)
USB	–	Universal Serial Bus
UTF-16	–	Způsob kódování znaků ISO 10646/Unicode
QFN	–	Quad Flat No-leads package

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

2 Výběr součástek

Vzhledem k tomu, že není možné se cenou zařízení přiblížit zavedeným výrobcům elektroniky, rozhodl jsme se výběr součástek a konstrukci modulu přizpůsobit tak, aby bylo možné modul vyrobit v domácích podmínkách.

2.1 Způsob příjmu rozhlasového vyslání

Jednou možností je řešení přijímačem složeným z diskrétních součástek a nebo pomocí analogových IO. Ovšem toto je příliš komplikované.

Na trhu je řada integrovaných obvodů, které zajišťují samotný příjem vysílání včetně vyhledávání static, měření kvality signálu a přijmu RDS a to s minimem potřebných externích součástek. Tyto IO se typicky ovládají pomocí I^2C nebo SPI a zvuk poskytují digitálně přes rozhraní I^2S popřípadě analogově. Bohužel drtivá většina těchto IO je dostupná pouze v pouzdru QFN, které se velmi obtížně pájí a je možné je sehnat pouze v minimálním množství 1000 kusů. Výjimkou je SI4735-D60 od výrobce SILICON LABS, který je dostupný v pouzdru SSOP24 a je možné jej u nás zakoupit i po jednotlivých kusech. IO neumožňuje přijímat DAB, ale umí následující:

- Pásma: FM, SW, MW, LW.
- Vzorkovací frekvence až do 48kHz.
- Rozlišení vzorku kanálu až do 24bitů.
- Stereofonní příjem.
- Příjem RDS.

2.2 Volba rozhraní pro spojení modulu a počítače

Po tomto rozhraní se budou přenášet dva druhy informací a to samotný zvuk a ovládání tunerů.

V současné době je prakticky jediným schůdným řešením použití rozhraní USB díky celé řadě výhod, které nabízí. Zejména jeho širokým rozšířením na téměř všech počítačích, od osobních přes servery až po jednodeskové či průmyslové počítače. Stejně tak je k dispozici velké množství součástek se zabudovanou podporou tohoto rozhraní. USB dále poskytuje možnost napájení připojených zařízení až do příkonu 2,5 W. Má zabudovanou podporu pro různé druhy přenosů včetně isochronních (garantovaný periodický přenos předem dohodnutého množství dat). Specifikace USB zavádí standardní třídy funkcí v zařízení. V době psaní tohoto textu sice neexistuje třída pro ovládání tuneru, ale existuje třída popisující zvuková zařízení. Díky tomuto není potřeba vyvíjet vlastní ovladač zvukové karty na straně počítače.

2.2.1 Verze USB specifikací

V současné době je možné se setkat s USB verze 1.0, 1.1, 2.0 a 3.0. Dobrou zprávou je zpětná kompatibilita všech verzí. tj. zařízení podle specifikace 1.0 by mělo fungovat s jakýmkoliv hostem. Rychlost full speed definuje už první specifikace, její maximální propustnost 12 Mbit je pro věrný přenos dvoukanálového zvuku více než dostatečná. Novější verze nepřinášejí žádnou vlastnost, která by byla pro tento projekt přínosná.

Odlišná situace je v případě specifikací třídy USB audio. Existují vzájemně nekompatibilní verze 1.0 a 2.0. Ani zde mladší verze nepřináší žádný benefit, který bych mohl využít. Navíc doposud nemá nativní podporu ani ve Windows 10. Z tohoto důvodu není použití USB audio 2.0 příliš vhodné.

2.3 Napojení tuneru na USB

Požadavky:

- Schopnost přenášet dvoukanálový zvuk bez znatelného zkreslení. Zvolil jsem PCM formát o vzorkovací frekvenci 48 kHz a rozlišení 16 bitů na jeden kanál. Pro srovnání audio CD používá 44,1 kHz / 16bitů.
- Alespoň jedno rozhraní I^2S schopné přijímat zvuk a fungující v režimu master.
- Podporu pro USB audio. To implikuje nutnost podpory full speed USB a nebo rychlejší. Low speed nepodporuje isochronní přenosy, které jsou nezbytné pro přenos zvuku.
- Rozhraní I^2C master pro ovládání tunerů.
- Kompatibilita s 3,3 V logikou tunerů.

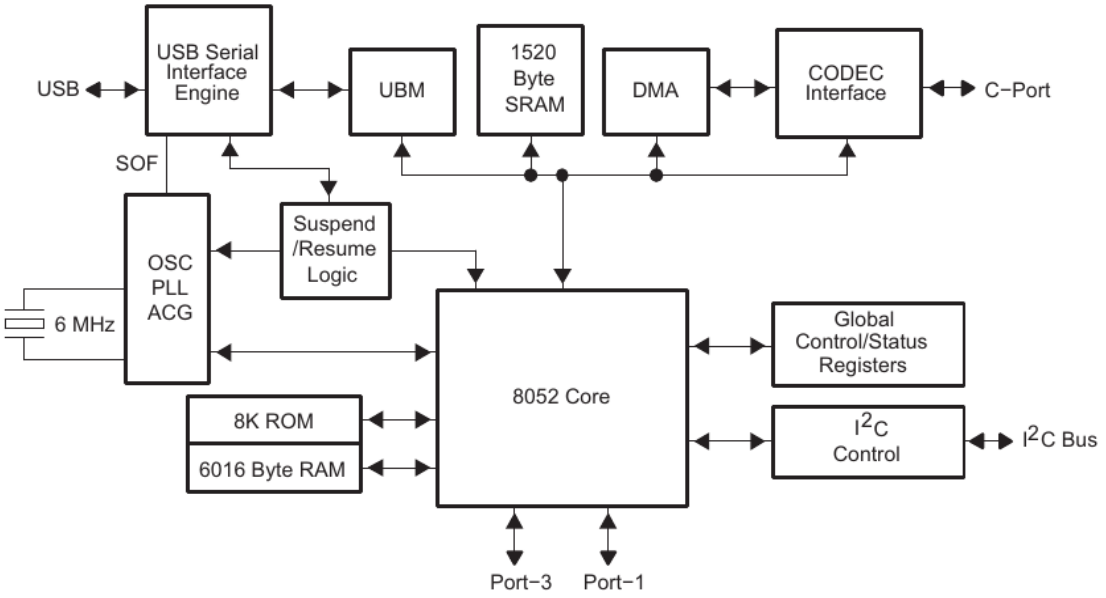
2.3.1 TAS1020b

Jak je patrné z obrázku 1, jedná se o USB I^2S zvukovou kartu a MCU v jednom. Na rozdíl od většiny MCU nemá interní paměť programu. Program se načítá při spuštění buď z E^2PROM paměti připojené přes I^2C a nebo přes USB ze zařízení, ke kterému je obvod připojen.

Obvod podle specifikace [15] podporuje všechno potřebné. Full speed USB 1.1 včetně USB audio 1.0, 14 endpointů z toho až dva mohou být isochronní. Dále nabízí až dvě vstupní I^2S rozhraní a jednu I^2C sběrnici. Nevýhodou je absence programové paměti, kusová dostupnost obvodu pouze ve formě vzorků a v mém případě také fakt, že s tímto druhem obvodů nemám žádné zkušenosti.

2.3.2 PIC16F1454

PIC16F1454 je osmi bitový MCU od firmy Microchip s podporou full speed USB 2.0. Obvod obsahuje továrně kalibrovaný oscilátor a umí pracovat při napájecím napětí 2,3-5,5V.



Obrázek 1: Blokové schéma TAS1020b. (Převzato z [15])

Díky tomu obvod nepotřebuje prakticky žádné externí součástky. V podstatě k němu stačí připojit pouze USB kabel.

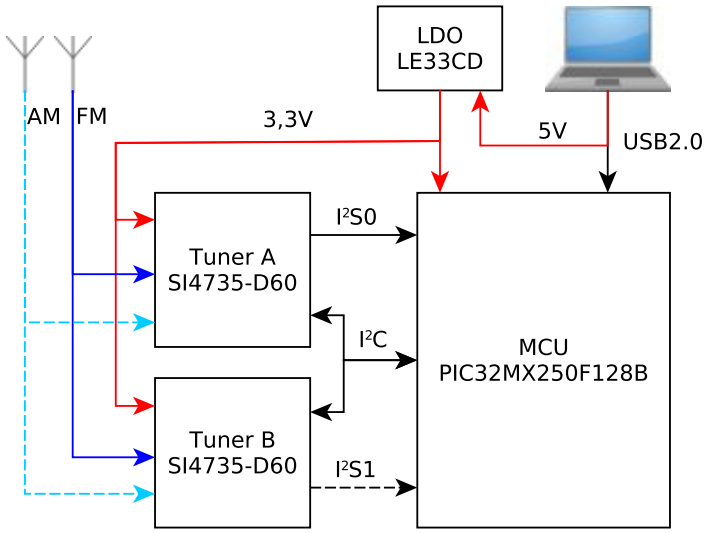
Výrobce poskytuje k tomuto MCU knihovnu Microchip Library for aplications, která mimo jiné obsahuje implementaci USB audio 1.0. Navíc jedním ze vzorových projektů u této knihovny je i USB mikrofón, který řeší přenos zvuku do počítače.

Bohužel tento MCU nemá podporu I^2S a na jeho softwarovou implementaci je příliš pomalý.

2.3.3 PIC32MX250F128

Tento 32 bitový MCU, taktéž od firmy Microchip, je vybaven všemi potřebnými rozhraními. Full speed USB 2.0, 2x nezávislé I^2S , 2x I^2C . Pracuje v rozmezí napájecích napětí 2,3-3,6V. V MCU jsou k dispozici čtyři DMA kanály, které je možné řetězit (po ukončení jednoho kanálu se automaticky spustí druhý). Podobně jako u PIC16F1454 je i k tomuto čipu k dispozici framework Harmony [14] s podporou pro USB Audio 1.0. Vyrábí se v různých pouzdrech, dokonce i v DIP, které je možné přímo zapojit do nepájivého pole.

Pro modul jsem vybral právě tento MCU. Případně je možné něco málo ušetřit a použít PIC32MX220F032. Liší se pouze menšími velikostmi pamětí, konkrétně 32kB programové paměti místo 128kB a 8kB datové paměti namísto 32kB.



Obrázek 2: Blokové schéma zapojení.

2.4 Výsledná konstrukce

Propojení jednotlivých komponent je naznačeno na obrázku 2. Jak je patrné, zahrnul jsem i propojení tuneru B s MCU přes I^2S a také vyvedení anténních AM vstupů tunerů (v obrázku čárkovaně). Aktuálně nejsou využity, ale v budoucnu bude možné zařízení rozšířit o podporu příjmu všech pásem, které tunery podporují a nebo přidat režim, kdy se modul bude chovat jako dvě nezávislé zvukové karty.

Celé zařízení je napájeno z USB přes jediný lineární stabilizátor LE33CD, který snižuje napájecí napětí na 3,3V. Dokáže poskytnout až 100mA a je odolný proti nadproudu a přehřátí [16]. MCU je zapojen podle doporučení v katalogovém listu [8] kapitola 2.1. Stejně tak tunery jsou zapojeny podle doporučení v katalogovém listu [6] kapitola 2.2.

Tady bych chtěl zmínit vstup RCLK. Tuner potřebuje v době ladění hodinový signál. V katalogovém listu je několikrát zmíněno, že tento signál je nutné buď přivést přímo na vstup RCLK a nebo se získá zapojením 32,768 kHz krystalu. Až v programovací příručce [7] v popisu parametru REFCLK_PRESCALE je možnost odvodit hodinový signál od vstupu hodinového signálu I^2S DCLK. Této možnosti jsem samozřejmě využil.

Kompletní schéma zapojení je součástí příloh.

3 Tuner

Jak jsem zmínil v kapitole 2.4 k přijmu rozhlasového vysílání je použit integrovaný obvod SI4735-D60. Zvuk je do MCU přenášen přes rozhraní I^{2}S , samotné ovládání tuneru je realizováno přes sběrnici I^{2}C .

3.1 Zvukové rozhraní I^{2}S

MCU	Směr	Tuner	Význam
SDI	$\Rightarrow$	DOUT	Datový signál
SCK	$\Rightarrow$	DCLK	Hodiny
SS	$\Leftarrow$	DFS	Signál určení kanálu

Tabulka 1: Popis I^{2}S signálů.

Rozhraní I^{2}S je v případě jednosměrného přenosu v podstatě rozhraní SPI doplněné o další datový signál. Význam signálů včetně odpovídajících názvů pinů na MCU a tuneru shrnuje tabulka 1. V případě I^{2}S se nejedná o sběrnici. Vždy komunikují právě dvě zařízení kde jedno je master a druhé slave. Master vždy vysílá datový signál a signál přepínání kanálů. Datový signál vysílá samozřejmě zřízení, které je zdrojem zvuku. Je možný i obousměrný přenos, potom je nutné použití dvou datových linek.

Tuner umí pracovat pouze v režimu slave, takže hodinový signál a signál přepínání kanálů je generován z MCU. Vzorkování zvuku v tuneru je řízeno jeho vlastním interním oscilátorem. Díky tomuto pravděpodobně občas dojde k zahození některého vzorku pokud MCU čte pomaleji než tuner provádí vzorkování, a nebo dojde k přečtení náhodných dat v opačném případě. Nicméně v praxi jsem s tímto nezaznamenal žádný problém.

Vyjma standardního formátu dat popsaného v I^{2}S specifikace [18] vzniklo ještě několik dalších formátů. Použil jsem standardní formát, tak jak je zobrazen na obrázku 3.

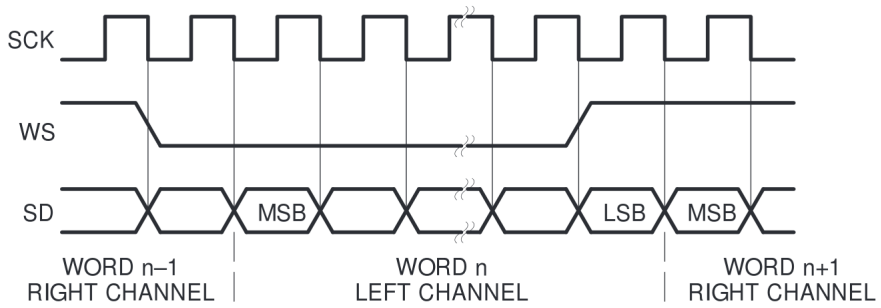
3.2 Ovládací rozhraní I^{2}C

Ovládání tuneru se provádí přes rozhraní I^{2}C . Toto rozhraní je součástí drtivé většiny mikrokontrolérů, není ho třeba blíže popisovat. Tuner se do režimu I^{2}C přepne připojením pinů GPO1 na napájecí napětí a GPO2 k zemi.

SEN	Adresa čtení	Adresa zápisu
0	0x23	0x22
1	0xC7	0xC6

Tabulka 2: I^{2}C adresy.

Základní informací nezbytnou pro komunikace s tunerem je I^{2}C adresa. Tuner umožňuje změnu adresy pomocí změny úrovně na pinu SEN. Adresy jsou v tabulce 2.



Obrázek 3: Časový diagram I²S přenosu. (Převzato z [18])

3.3 Ovládání tuneru

Tuner se ovládá zapisováním příkazů a případným čtením odpovědí. Dále je zde dvojice příkazů pro čtení a zápis proměnných. (properties) tuneru. Kompletní popis všech příkazů a proměnných je v programovací příručce k tuneru [7].

Název pole	Délka	
CMD	1 B	Příkaz.
ARG1 - ARG7	0 - 7 B	Případný argument příkazu.

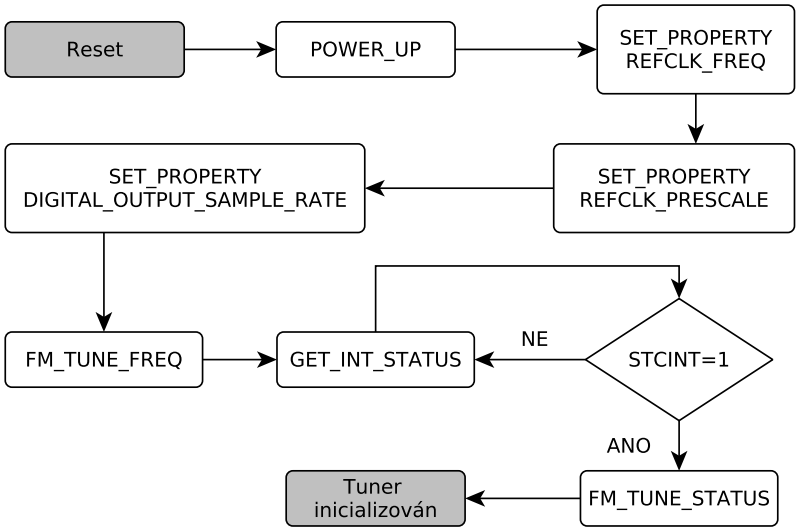
Tabulka 3: Formát příkazu pro tuner.

Název pole	Délka	
STATUS	1 B	Status tuneru.
RESP1 - RESP15	0 - 15 B	Případná další data odpovědi.

Tabulka 4: Formát odpovědi tuneru.

Všechny příkazy mají jednotný formát vyobrazený v tabulce 3. Každý příkaz je jednou transakcí zápisu na I²C sběrnici. Případnou odpověď na příkaz je možné získat následnou transakcí čtení z I²C sběrnice. Formát odpovědi je v tabulce 4. Jak je patrné odpověď na každý příkaz obsahuje vždy minimálně jeden byte se stavem zařízení (STATUS). Význam jednotlivých bitů v tomto byte je následující:

- Bit 7 **CTS** - Pokud je v 1 je možné do tuneru odeslat další příkaz.
- Bit 6 **ERR** - Pokud je v 1 signalizuje chybu provádění příkazu.
- Bity 5 - 3 **Rezervováno** - Hodnoty těchto bitů nejsou specifikovány.
- Bit 2 **RDSINT** - Pokud je v 1 signalizuje vznik přerušení od přijmu RDS.



Obrázek 4: Diagram inicializace tuneru.

- Bit 1 **ASQINT** - Pokud je v 1 signalizuje dokončení měření parametrů přijmu.
- Bit 0 **STCINT** - Pokud je v 1 signalizuje dokončení ladění nebo vyhledávání stanice.

Bity CTS a ERR se aktualizují vždy. Pro čtení nejnižších třech bitů je potřeba aktualizovat jejich hodnoty příkazem GET_INT_STATUS. Tento příkaz má číslo 0x14 a nemá žádné argumenty. Tuner umožňuje nakonfigurovat signalizaci změny některého z těchto bitů změnou úrovně na výstupu GPO2/INT. Této možnosti jsem se ale rozhodl nevyužít.

Jak jsem zmínil výše, kromě příkazů se parametry tuneru nastavují a nebo čtou pomocí proměnných. jedná se vždy o 16-ti bitové hodnoty, které se identifikují taktéž 16-ti bitovým číslem.

Zápis hodnoty se provádí příkazem SET_PROPERTY (0x12), za kterým následuje jeden nulový byte, za ním horní a poté dolní byte čísla proměnné. Dále se vyšle horní byte a poté dolní byte vlastní hodnoty.

Čtení se provádí obdobně, příkazem GET_PROPERTY (0x14) za kterým se vyšle nulový byte následovaný horním a dolním byte čísla zapisované proměnné. Hodnota se získá následným čtením z I^2C sběrnice. Horní byte hodnoty je až ve druhém bytu argumentu ARG-2 (viz. tab. 4), dolní byte v je v ARG-3.

3.3.1 Inicializace tuneru

Po spuštění MCU vždy provádí inicializaci obou tunerů. Průběh inicializace shrnuje diagram 4. Z důvodu zkrácení doby spuštění modulu probíhá inicializace formou jednoduchého "multitaskingu". Pokud je jeden tuner zaneprázdněn testuje se stav druhého tuneru a v případě, že není zaneprázdněn dojde k vyslání příkazu, pokud je zaneprázdněn

testuje se znova první tuner a tak dále. Ze stejného důvodu je v příkazu FM_TUNE_STATUS nastaven příznak zrychleného nepřesného ladění.

Bit	7.	6.	5.	4.	3.	2.	1.	0.
CMD	0x01							
ARG1	CTSIEN = 0	GPO2OEN = 0	PATCH = 0	XOSCEN = 0	FUNC = 0			
ARG2	OPMODE = 0b10110000							

Tabulka 5: Příkaz spuštění tuneru POWER_UP.

Inicializace začíná spuštěním tuneru příkazem POWER_UP. Formát příkazu je v tabulce 5. Součástí příkazu jsou tyto parametry:

- **CTSIEN** - Možnost hardwarové signalizace přerušení v okamžiku kdy je možné odeslat další příkaz. Hardwarové přerušení nepoužívám.
- **GPO2OEN** - Nastavení pinu GPO2 jako výstupu.
- **PATCH** - Umožňuje načtení novějšího firmware do integrovaného obvodu tuneru.
- **XOSCEN** - Použití krystalu jako zdroje hodinové frekvence. Místo krystalu používám přímo hodinový kmitočet rozhraní I^2S .
- **FUNC** - Použitá hodnota nastaví režim přijmu FM rádia.
- **OPMODE** - Nastavení výstupu zvuku. Použitá hodnota znamená "pouze digitální výstup I^2S ".

Odpovědí na tento příkaz je pouze jeden byte se statusem tuneru.

Dále je potřeba nastavit dvě proměnné. REFCLK_PRESCALE hodnotu děličky hodinového kmitočtu a do REFCLK_FREQ výslednou frekvenci, která se musí pohybovat v rozsahu $31,130 - 34,406Khz$. Jak už jsem zmínil jako zdroj hodinové frekvence je použita hodinová frekvence I^2S rozhraní. Tato frekvence vyplývá ze součinu počtu kanálů, jejich rozlišení a vzorkovací frekvence. $2 \cdot 16 \cdot 48 = 1536kHz$. Dělící poměr tedy vychází $1536/32 = 48$ při frekvenci $32000kHz$.

Zatímco REFCLK_FREQ obsahuje pouze 16-ti bitovou hodnotu frekvence v Hz v REFCLK_PRESCALE jsou pro hodnotu dělícího poměru vyhrazeny bity 0-11. Bity 13-15 musí být nulové, bit 12 je nastaven na hodnotu 1, což znamená, že se jako zdroj hodinové frekvence použije I^2S .

Poslední proměnná, kterou je potřeba nastavit, je DIGITAL_OUTPUT_SAMPLERATE. Obsahuje hodnotu vzorkovací frekvence. V tomto případě je to hodnota 48000.

Proměnné se nastavují pomocí příkazu SET_PROPERTY popsaného v kapitole 3.3. Výsledné hodnoty a čísla proměnné jsou shrnuty v tabulce 6.

Poslední částí inicializace tuneru je naladění frekvence 93,7 MHz příkazem FM_TUNE_FREQ. Formát příkazu včetně použitých hodnot je v tabulce 7. Význam parametrů je následující:

Název	Číslo	Hodnota
REFCLK_FREQ	0x0201	0x7D00
REFCLK_PRESCALE	0x0202	0x102A
DIGITAL_OUTPUT_SAMPLE_RATE	0x0104	0xBB80

Tabulka 6: Proměnné použité pro inicializaci tuneru.

Bit	7.	6.	5.	4.	3.	2.	1.	0.
CMD	0x20							
ARG1	0x00						FREEZE = 0	FAST = 1
ARG2	$FREQ_H$ = 0x24							
ARG3	$FREQ_L$ = 0x94							
ARG4	ANTCAP = 0x00							

Tabulka 7: Příkaz spuštění tuneru FM_TUNE_FREQ.

- **FREEZE** - Nastavení způsobí pozvolný přechod zvuku po přeladění.
- **FAST** - Nastavení způsobí rychlé ale nepřesné přeladění.
- **FREQ**$_H$ - Horní byte frekvence v desetinách MHz.
- **FREQ**$_L$ - Dolní byte frekvence v desetinách MHz.
- **ANTCAP** - Nastavení kapacity vstupního kondenzátoru antény. Hodnota 1-191 pF. Hodnota 0 znamená automatické nastavení.

Odpovědí na tento příkaz je pouze jeden byte obsahující status tuneru. Dokončení ladění je signalizováno nastavením bitu STCINT ve statusu. K aktualizaci hodnoty tohoto bytu je nutné vždy vyslat příkaz GET_INT_STATUS. Tento příkaz se skládá z jediného byte s číslem příkazu 0x14. Odpovědí na tento příkaz je rovněž jediný byte se statusem tuneru kde jsou aktualizovány hodnoty bitů signalizujících přerušení RDSINT, ASQINT a STCINT.

Bit	7.	6.	5.	4.	3.	2.	1.	0.
CMD	0x22							
ARG1	0x00						CANCEL = 0	INTACK = 1

Tabulka 8: Příkaz zjištění stavu ladění FM_TUNE_STATUS.

Ke smazání bitu STCINT je použit příkaz FM_TUNE_STATUS viz. tabulka 8. Kromě smazání tohoto bitu nastavením parametru INTACK je možné nastavením tohoto parametru zrušit probíhající ladění nebo vyhledávání stanice. Jak je vidět z tabulky 9, odpověď na příkaz obsahuje kromě statusu tuneru následující informace:

Bit	7.	6.	5.	4.	3.	2.	1.	0.
SATUS	CTS	ERR	X	X	RSQINT	RDSINT	X	STCINT
RESP1	BLTF	X	X	X	X	X	AFCRL	VALID
RESP2	READFREQ$_H$							
RESP3	READFREQ$_L$							
RESP4	RSSI							
RESP5	SNR							
RESP6	MULT							

Tabulka 9: Odpověď na příkaz zjištění stavu ladění FM_TUNE_STATUS.

- **BLTF** - Je nastaven pokud vyhledávání stanice přeteklo přes hodnotu maximální nebo podteklo hodnotu minimální laditelné frekvence.
- **AFCRL** - Je nastaven pokud je automatické dolaďování aktivní.
- **VALID** - Naladěná frekvence byla vyhodnocena jako validní.
- **READFREQ$_H$** - Horní byte naladěné frekvence v desetinách MHz.
- **READFREQ$_L$** - Dolní byte naladěné frekvence v desetinách MHz.
- **RSSI** - Indikátor síly přijímaného signálu v dBμV.
- **SNR** - Odstup signálu od šumu v dB.
- **MULT** - Indikátor míry odrazů v signálu.

3.3.2 Vyhledávání stanic

Bit	7.	6.	5.	4.	3.	2.	1.	0.
CMD	0x22							
ARG1	0x00				SEEKUP	WRAP	0x00	

Tabulka 10: Příkaz zjištění stavu ladění FM_TUNE_STATUS.

Vyhledávání stanic se provádí pomocí příkazu FM_SEEK_START popsaném v tabulce 10. Příkaz lze parametrizovat pouze pomocí dvou bitů. Nastavením bitu SEEKUP se bude vyhledávat směrem k vyšším frekvencím. Nastavením bitu WRAP bude při dosažení nejvyšší frekvence vyhledávání pokračovat znova od nejnižší frekvence, v případě sestupného vyhledávání se bude po dosažení nejnižší frekvence pokračovat od horní frekvence. Pokud bit WRAP není nastaven, při dosažení nejnižší nebo nejvyšší frekvence dojde k zastavení vyhledávání.

Ukončení vyhledávání je signalizováno nastavením bitu STCINT ve statusu odpovědi tuneru. Detekce ukončení vyhledávání se provádí shodně jako detekce ukončení ladění popsaná v kap. 3.3.1. Naladěnou frekvenci a informace o kvalitě signálu je možné přečíst příkazem FM_TUNE_STATUS, který byl taktéž popsán v kapitole 3.3.1. Tyto informace se vždy aktualizují po naladění, opětovné volání příkazu FM_TUNE_STATUS vrací vždy stejné hodnoty.

3.3.3 Čtení RDS z tuneru

RDS je rozšíření pozemního rozhlasového vysílání o souběžné vysílání různých digitálních informací. Z nejznámějších informací je to RT (Radio Text) - přenos až 64 znaků textu, typicky s názvem právě přehrávané skladby a nebo názvem rádiové stanice. Potom AF (Alternative frequency) seznam dalších frekvencí, na kterých je možné stanici naladit, což lze s využitím informace PI (Programme Identification), který stanici jednoznačně identifikuje k automatickému přelaďování na frekvenci s lepším příjmem. Kompletní popis je v [19].

Proud přenášených dat se dělí do třiceti skupin označených 1A, 1B, 2A ... 15A, 15B. Každá skupina má pevně danou velikost 104 bitů a skládá se ze čtyř stejně dlouhých bloků. Chtěl bych upozornit, že v normě [19] jsou tyto bloky označeny čísly 1 až 4, kdežto v programovací příručce tuneru [7] jsou označeny písmeny A až D. Každý blok nese 16 bitů užitečných dat a 10-ti bitové kontrolní slovo. Vyhodnocování chyb v přijatých datech a jejich případnou korekci zajišťuje tuner. Pro příjem je v tuneru k dispozici fronta na 14 skupin.

Bit	15.	14.	13.	12.	11.	10.	9.	8.
Horní byte	BLETHA = 2		BLETHB = 2		BLETHC = 2		BLETHD = 2	
Bit	7.	6.	5.	4.	3.	2.	1.	0.
Dolní byte	0x00							RDSEN = 1

Tabulka 11: Proměnná konfigurace příjmu RDS FM_RDS_CONFIG.

Příjem RDS v tuneru je potřeba nejprve nakonfigurovat a povolit. Díky nepoužití hardwarového přerušení, je možné konfiguraci přerušení přeskočit. Konfigurace přijmu RDS se zjednoduší na nastavení jediné proměnné FM_RDS_CONFIG, jejíž formát se nachází v tabulce 11. Nastavením bitu RDSEN této proměnné dojde k povolení přijmu RDS. Pole BLETHA až BLETHD specifikují dovolenou míru chybovosti bloků A až D přijímaných skupin a to takto:

- 0 - Pokud je v bloku jakákoliv chyba celá skupiny bude zahozena.
- 1 - Skupina bude přijata, pokud v bloku došlo k opravě nejvýše dvou bitů.
- 2 - Skupina bude přijata, pokud v bloku došlo k opravě nejvýše pěti bitů.
- 3 - Jakékoliv, i neopravitelné, množství chyb v bloku nezpůsobí zahození skupiny.

Bit	7.	6.	5.	4.	3.	2.	1.	0.
CMD	0x24							
ARG1	0x00					STATUSONLY = 0	0x00	INTACK = 1

Tabulka 12: Příkaz přečtení přijaté RDS skupiny FM_RDS_STATUS.

Vyčítání přijatých skupin se potom provádí příkazem FM_RDS_STATUS, uvedeným v tabulce 12. Nastavením bitu STATUSONLY v argumentu tohoto příkazu je možné pouze vyčíst informace o stavu příjmu RDS bez odebrání nejstarší přijaté skupiny z fronty. Nastavením druhého bitu INTACK dojde ke smazání příznaku přerušení RDSINT ve statusu tuneru. Vzhledem k nepoužití přerušení nastavení bitu INTACK nemá význam.

Bit	7.	6.	5.	4.	3.	2.	1.	0.
SATUS	CTS	ERR	X	X	RSQINT	RDSINT	X	STCINT
RESP1	X	X	RDSNEW BLOCKB	RDSSYNC BLOCKA	X	RDSSYNC FOUND	RDSSYNC LOST	RDSRECV
RESP2	X	X	X	X	X	GRPLOST	X	RDSSYNC
RESP3	RDSFIFOUSED							
RESP4	$BLOCKA_H$							
RESP5	$BLOCKA_L$							
RESP6	$BLOCKB_H$							
RESP7	$BLOCKB_L$							
RESP8	$BLOCKC_H$							
RESP9	$BLOCKC_L$							
RESP10	$BLOCKD_H$							
RESP11	$BLOCKD_L$							
RESP12	BLEA		BLEB		BLEC		BLED	

Tabulka 13: Odpověď na příkaz přečtení přijaté RDS skupiny FM_RDS_STATUS.

Odpověď na tento příkaz v tabulce 13 je podle očekávání poněkud obsáhlejší. První byte obstahuje jako vždy status tuneru. V následujících dvou bytech za statusem jsou bity jejihž LOG 1 má následující význam:

- **RDSNEWBLOCKB, RDSNEWBLOCKA** - Byl přijat validní blok B nebo A.
- **RDSSYNCFOUND** - Příjem RDS je synchronizován.
- **RDSSYNCLOST** - Ztracena synchronizace příjmu RDS, např. z důvodu špatného signálu.
- **RDSRECV** - Nastaven pokud je ve frontě počet skupin větší nebo roven hodnotě proměnné FM_RDS_INT_FIFO_COUNT. Tato proměnná byla ponechána ve výchozím nastavení na hodnotě 0, takže tento bit bude nastaven vždy.
- **GRPLOST** - Přetečení fronty přijatých skupin.

- **RDSSYNC** - Příjem RDS je synchronizován. (Aktuální stav.)

V bytu RDSFIFOUSED je počet skupin ve frontě. Pokud je tato hodnota větší než nula, v následujících osmi bytech se nachází data nejstarší přijaté skupiny. V posledním byte odpovědi jsou informace o chybovosti přijmu jednotlivých bloků skupiny BLEA až BLED. Význam hodnot chybovosti je tento:

- 0 - Blok přijat bez chyby.
- 1 - V bloku došlo k opravě jednoho až dvou bitů.
- 2 - V bloku došlo k opravě tří až pěti bitů.
- 3 - Blok nebylo možné opravit. Neobsahuje platná data.

Aby nedoházelo k přetečení fronty přijatých skupin, je potřeba frontu vyčítat v dostatečně krátkých intervalech. Zde je možné využít z pevné délky skupiny 104 b a rychlosti přenosu 1187,5 kHz. Přenos skupiny tedy trvá 87,5 ms. K naplnění 14-ti prvkové fronty tedy dojde nejdříve za 1,2 s. Pokud bude interval vyčítání vždy kratší, k přetečení fronty nedojde.

4 USB

Na rozdíl od například populárního rozhraní UART je USB podstatně komplikovanější. Je to určitou daní za jeho univerzálnost. Vzhledem k velkému rozsahu specifikace USB [2] se omezím pouze na popis částí nezbytných pro implementaci modulu.

S vydáním specifikace USB 2.0 byly předchozí specifikace označeny jako zastaralé a neměly by se používat pro nové konstrukce. Následující text se tedy týká USB 2.0 a rychlosti full-speed.

4.1 Stručný úvod do full-speed USB 2.0

4.1.1 Topologie

TODO Obrázek s pyramidou topologie

Ačkoliv název rozhraní (Univerzální Sériová Sběrnice) napovídá, že jde o sběrnici, jedná se o zapojení typu hvězda. Přesněji, jak je vidět z obrázku **??**, připojená zařízení a rozbočovače tvoří strom jehož kořenem je hostitel. Tento implementuje tzv. kořenový rozbočovač (root hub), ke kterému je možné buď přímo připojit jedno zařízení a nebo rozbočovač a do něj dalších až osm zařízení/rozbočovačů. Je možné takto za sebe zřetězit až pět rozbočovačů. Celkově je možné na jeden kořenový rozbočovač připojit až 127 zařízení (včetně rozbočovačů).

Velkou výhodou je, že zařízení je od topologie odstíněno. Vždy se z jeho pohledu komunikuje přímo s hostem.

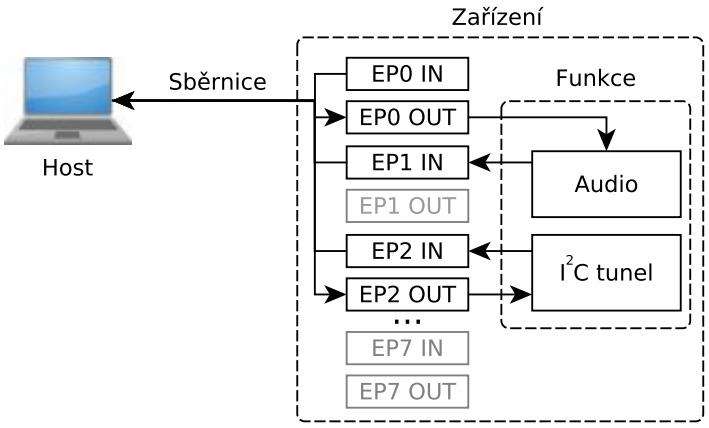
Ve specifikaci je také zohledněn fakt, že zařízení zpravidla nezastává pouze jedinou funkci. To je konec konců případ i tohoto modulu. Obsahuje dvě funkce - zvukovou kartu a I^2C tunel pro komunikaci s tunery.

4.1.2 Komunikace

	Latence	Vyhrazená šířka pásma	Spolehlivý přenos	Typ dat
Isochroní přenos	Minimální	až 90%	Ne	Proud
Hromadný přenos	Negarantovaná	Ne	Ano	Proud
Přenos přerušení	Minimální	až 90%	Ano	Proud
Řídící přenos	Negarantovaná	až 10%	Ano	Zprávy

Tabulka 14: Druhy USB přenosů.

Aby bylo možné uspokojit nároky na přenos (transfer) dat rozdílné povahy různými funkcemi, zavádí specifikace koncové body (endpoint). Osm výstupních (OUT) pro směr z hostitele do zařízení a osm pro směr opačný (IN). Směr je vždy určován právě z pohledu hostitele. Každému bodu je možné přiřadit jeden ze čtyř druhů přenosu podle tabulky



Obrázek 5: USB koncové body a funkce.

14. Výjimkou je vstupní a výstupní koncový bod nula. Tyto vždy slouží pro řídící přenosy a na rozdíl od ostatních bodů je musí podporovat všechna zařízení. Full-speed USB podporuje čtyři druhy komunikace uvedené v tabulce 14.

Body nula jsou využívány jednak k inicializaci a správě vlastního zařízení, ale také můžou být využívány zároveň i funkcemi. Například popisovaný modul jej využívá pro ovládání audio funkce. Modul dále využívá vstupní koncový bod 1 pro přenos zvuku do hostitele a potom dvojici koncových bodů 2 v obou směrech pro komunikaci s tunery viz. obrázek 5.

Přenos vždy sestává z alespoň jedné transakce (transaction), která se dále dělí na pakety (packets). Transakce je z pohledu zařízení vždy vyřizována od počátku do konce bez přerušení jinou transakcí. Je vždy iniciována vysláním token paketu hostitelem, který takto může řídit šířku pásma přidělovanou jednotlivým zařízením na sběrnici. Token pakety mohou být podle druhu transakce následujících třech druhů:

- **IN** (Vstupní) - Následuje přenos ze zařízení do hostitele.
- **OUT** (Výstupní) - Následuje přenos z hostitele do zařízení.
- **SETUP** - Následuje řídící přenos.

Za tímto paketem následuje nula nebo více paketů s daty. Aby bylo možné detekovat výpadek nebo duplikaci některého z paketů jsou specifikovány hned dva typy paketů. A to DATA0 a DATA1. U izochronních transakcí se vždy posílají pakety DATA0. U ostatních transakcí se vyšle nejprve DATA0 a poté se tyto druhy paketů střídají nezávisle na transakcích. Znovu se od DATA0 začne pouze v následujících případech:

- Na začátku každé řídící transakce.
- Po následujících žádostech hostitele (bude popsáno dále):

- Přiřazení konfigurace.
- Zrušení zastavení koncového bodu.
- Nastavení rozhraní.

Za datovými pakety následuje potvrzování transakce protistranou. Izochronní přenosy potvrzování nepodporují, tudíž datovými pakety jejich transakce končí. Transkace hromadných přenosů a přenosu přerušení jsou vždy zakončeny jedním potvrzovacím (handshake) paketem. Transkace řídících přenosů mají před potvrzovací paket vložen jeden datový paket nulové délky (zero length packet často zkracovaný ZLP), ale opačného směru než všechny předchozí datové pakety. Potvrzovací paket vždy vysílá zařízení. Má vždy jeden z následujících typů:

- **ACK** (Úspěch) - Úspěšné ukončení transakce.
- **NAK** (Neúspěch) - Typicky poškozená přijímaná data a nebo častěji zařízení nemá připravena data k odeslání.
- **STALL** (Chyba) - Zařízení takto reaguje na požadavek, který nepodporuje.

Rozdíl mezi NAK a STALL je, že po NAK potvrzení bude hostitel požadavek opakovat (počet opakování není explicitně specifikován), kdežto STALL signalizuje nemožnost vyřízení požadavku a tudíž jej opakovat nemá smysl.

Formátování a rozpoznávání paketů řeší přímo USB modul v mikrokontroléru, není tedy nutné se jím hlouběji zabývat. Detailní popis je v kapitole 8 USB specifikace [2].

4.1.3 Deskriptory

USB specifikace zavádí deskriptory (descriptors). Jedná se o unifikovaný způsob jak může zařízení informovat hostitele o svých schopnostech a požadavcích. Na základě právě těchto informací může operační systém vybrat pro funkce zařízení odpovídající ovladače, řadič v hostiteli se dozví kolik dat, jak často bude přenášet po jednotlivých koncových bodech a jakou má tomuto přenosu přiřadit prioritu a podobně.

Při vývoji zařízení je základním rozdělením deskriptorů rozdělení podle požadavků hostitele:

- **Deskriptor zařízení** (Sevice descriptor) - Nejnutnější informace pro správu zařízení na sběrnici.
- **Deskriptory řetězců** (String dscriptors) - Pole textových řetězců a informace o dostupných lokalizacích.
- **Deskriptory konfigurace** (Configuration descriptor) - Struktura deskriptorů s veškerými dalšími informacemi.

Popis všech deskriptorů všech možných funkcí je zcela mimo rozsah tohoto textu. Dále se omezím pouze na deskriptory a jejich hodnoty použité v modulu.

Název pole	Délka	Hodnota	
bLength	1B	18	Délka deskriptoru.
bDescriptorType	1B	0x01	Typ deskriptoru.
bcdUSB	2B	0x0200	Verze USB specifikace implementovaná zařízením. (2.0)
bDeviceClass	1B	0x00	Třída zařízení.0x00 znamená, že třídu specifikuje každé rozhraní zvlášť.
bDeviceSubClass	1B	0x00	Podtřída zařízení. Pokud je bDeviceClass 0x00, musí být i toto pole 0x00.
bDeviceProtocol	1B	0x00	Protokol zařízení. Pokud je bDeviceClass 0x00, musí být i toto pole 0x00.
bMaxPacketSize	1B	64	Největší délka data, kterou je možné odeslat koncovým bodem 0.
idVendor	2B	0x04D8	ID výrobce zařízení.
idProduct	2B	0xF32C	ID zařízení.
bcdDevice	2B	0x0100	Verze zařízení 1.0.
iManufacturer	1B	1	Odkaz na řetězec s názvem výrobce.
iProduct	1B	2	Odkaz na řetězec s názvem zařízení.
iSerialNumber	1B	0	Odkaz na řetězec se sériovým číslem zařízení. 0 znamená nespecifikován.
bNumConfigurations	1B	1	Počet konfigurací zařízení.

Tabulka 15: Deskriptor zařízení.

4.1.4 Deskriptor zařízení

V tabulce 15 je uveden deskriptor zařízení tak, jak je použit v modulu. Myslím, že popis významu polí v tabulce je dostatečný. Za zmínku stojí ID výrobce a zařízení. Jejich účel je stejný jak například MAC adresa síťových zařízení a to jednoznačně identifikovat druh zařízení. Oficiální cesta je požádat o přiřazení ID výrobce USB implementers fórum, což v době psaní toho textu stojí 5000 amerických dolarů [17]. Výrobci programovatelných součástek s podporou USB, ale naštěstí zpravidla nabízejí možnost zdarma získat ID zařízení z jejich rozsahů. Jednou z podmínek bývá nutnost použít přidělení ID právě na jejich součástce. Pro modul jsem získal ID produktu od firmy Microchip 0xF32C. ID výrobce je 0x04D8.

Pole iManufacturer, iProduct a iSerialNumber nesou indexy na deskriptory textových řetězců. Všechna pole jsou nepovinná. V případě jejich vynechání se použije index s hodnotou 0. Jak napovídají názvy jednotlivých polí, je možné takto přidat popisek výrobce a zařízení ve formě lidsky čitelného textu a sériové číslo daného kusu zařízení, které může využít ovladač v hostiteli například pro načtení posledního nastavení po opětovném připojení zařízení.

4.1.5 Deskriptory řetězců

TODO:Tabulka se zkrácenými string deskriptory.

Deskriptory řetězců jsou organizovány jako pole indexované od nuly. Každý jeden řetězec začíná hlavičkou deskriptoru, ve které je určen typ deskriptoru a jeho celková délka v bytech. Poté následuje samotný text zakódovaný podle normy unicode, konkrétně UTF-16. Je tedy možné použití i národních znaků.

Jedinou výjimkou je deskriptor s indexem 0. V zařízení je možné mít více sad textů v různých jazycích. Seznam dostupných jazyků (jazykových kódů) je právě v tomto deskriptoru. Seznam těchto kódů popisuje [5]. V modulu jsem se omezil pouze na angličtinu (kód 0x0409).

4.1.6 Konfigurace zařízení

Konfigurace zařízení je soubor deskriptorů, který popisuje schopnosti zařízení a také jeho nároky na přenos, popřípadě definují sady parametrů z nichž si může hostitel vybrat tu, která mu v daný okamžik nejvíce vyhovuje. Zařízení musí specifikovat minimálně jednu konfiguraci, ale také více. Hostitel potom zařízení jednu přidělí, případně ji může kdykoliv změnit za jinou.

Každá jedna konfigurace začíná deskriptorem konfigurace, za kterým následují deskriptory rozhraní a koncových bodů, do kterých mohou být zanořeny deskriptory další. Tvoří takto stromovou strukturu. V případě mého modulu je možná pouze jediná konfigurace. Její struktura vypadá následovně:

1. Deskriptor řídícího rozhraní zvuku.
   (a) Deskriptor řídícího rozhraní zvuku - hlavička.
   (b) Deskriptor řídícího rozhraní zvuku - vstupní terminál přijímač rádia.
   (c) Deskriptor řídícího rozhraní zvuku - výstupní terminál odesílání zvuku přes USB.
2. Deskriptor rozhraní pro odesílání zvuku - varianta s vypnutým přenosem.
3. Deskriptor rozhraní pro odesílání zvuku - varianta se zapnutým přenosem.
   (a) Deskriptor rozhraní pro odesílání zvuku - obecný deskriptor.
   (b) Deskriptor rozhraní pro odesílání zvuku - popis formátování dat.
   (c) Deskriptor koncového bodu - odesílání zvuku.
4. Deskriptor rozhraní specifikovaného výrobcem - I^2C tunel.
   (a) Deskriptor koncového bodu - odesílání dat hostu.
   (b) Deskriptor koncového bodu - příjem dat z hosta.

Název pole	Délka	Hodnota	
bLength	1B	9	Délka deskriptoru.
bDescriptorType	1B	0x02	Typ deskriptoru.
wTotalLength	2B	127	Celková délka všech deskriptorů konfigurace.
bNumInterfaces	1B	3	Počet rozhraní v konfiguraci.
bConfigurationValue	1B	1	Index této konfigurace.
iConfiguration	1B	0	Odkaz na řetězec s popisem konfigurace.
bmAttributes	1B	0b10000000	Bitová maska s atributy.

Tabulka 16: Deskriptor konfigurace.

4.1.7 Deskriptor konfigurace

Jak je vidět z tabulky 16, tento deskriptor obsahuje informace pro identifikaci konfigurace. A to zejména index iConfiguration. To je hodnota, kterou poté pošle host do zařízení v požadavku o přidělení konfigurace. Dále následuje pole wTotalLength s celkovou délkou konfigurace. Hostitel obdrží všechny případné konfigurace v jednom bloku. Na základě této hodnoty rozliší, kde jednotlivé konfigurace začínají a končí.

Za zmínku také stojí pole bmAttributes. Nejvyšší bit musí být z důvodu kompatibility s USB 1.0 nastaven na 1, nejnižší bity 0-4 jsou rezervovány pro budoucí použití a musí být nastaveny na 0. Bit 6 signalizuje, že zařízení není napájeno z USB sběrnice. Bit 5 signalizuje, že zařízení chce využívat mechanizmus vlastního probuzení a informování hostitele o události. Modul má oba atributy nastaveny na hodnotu 0.

4.1.8 Deskriptory konfigurace vztažené k USB audio 1.0

Největší část konfigurace zabírají deskriptory popisující část přenosu zvuku. Je to dáno také tím, že specifikace USB audio [3] nepopisuje pouze přenos audia po USB. Nabízí také prostředky pro popis topologie vstupů, výstupů, různých efektových jednotek přepínačů, směšovačů a podobně včetně jejich ovládání. Bylo by například možné takto realizovat kompletní ovládání mixážního, kde by přes USB mohlo být realizováno pouze několik vstupů a výstupů a nebo i žádný. pultu Topologie modulu rádia z pohledu této specifikace je nejjednodušší možná. Je zde pouze jeden výstupní terminál přenosu zvuku přes USB, který má jako vstup nastaven vstupní terminál přijímač rádia.

Název pole	Délka	Hodnota	
bLength	1B	9	Délka deskriptoru.
bDescriptorType	1B	0x04	Typ deskriptoru.
bInterfaceNumber	1B	0	Pořadové číslo rozhraní.
bAlternateSetting	1B	0	Identifikátor alternativní nastavení.
bNumEndpoints	1B	0	Počet koncových bodů v tomto rozhraní.
bInterfaceClass	1B	1	Třída rozhraní. (Audio)
bInterfaceSubClass	1B	1	Podtřída rozhraní. (Control device)
bInterfaceProtocol	1B	0	Vždy 0.
iInterface	1B	0	Index na textový řetězec.

Tabulka 17: Deskriptor řídícího rozhraní zvuku.

Název pole	Délka	Hodnota	
bLength	1B	9	Délka deskriptoru.
bDescriptorType	1B	0x24	Typ deskriptoru.
bDescriptorSubType	1B	0x01	Podtyp deskriptoru. (Hlavička)
bcdADC	2B	0x0100	Verze USB Audio specifikace.
wTotalLength	2B	30	Celková délka deskriptorů tohoto rozhraní.
bInCollection	1B	1	Počet rozhraní pro odesílání zvuku.
baInterfaceNr(1)	1B	1	Index rozhraní pro odesílání zvuku.

Tabulka 18: Deskriptor řídícího rozhraní zvuku - hlavička.

Název pole	Délka	Hodnota	
bLength	1B	12	Délka deskriptoru.
bDescriptorType	1B	0x24	Typ deskriptoru.
bDescriptorSubType	1B	0x02	Podtyp deskriptoru. (Vstupní terminál)
bterminalID	1B	1	Identifikátor terminálu.
wTerminalType	2B	0x0710	Typ terminálu. (Přijímač rádia)
bAssocTerminal	1B	0	Přidružený terminál. (Nejedná se o zvukové propojení)
bNrChannels	1B	2	Počet zvukových kanálů.
wChannelConfig	2B	0x0003	Bitová mapa konfigurace kanálů.
iChannelNames	1B	0	Index na textový řetězec s názvem kanálů.
iTerminal	1B	0	Index na textový řetězec s popisem terminálu.

Tabulka 19: Deskriptor řídícího rozhraní zvuku - vstupní terminál.

Název pole	Délka	Hodnota	
bLength	1B	9	Délka deskriptoru.
bDescriptorType	1B	0x24	Typ deskriptoru.
bDescriptorSubType	1B	0x03	Podtyp deskriptoru. (Výstupní terminál)
bterminalID	1B	2	Identifikátor terminálu.
wTerminalType	2B	0x0101	Typ terminálu. (Odesílání přes USB)
bAssocTerminal	1B	0	Přidružený terminál. (Nejedná se o zvukové propojení)
bSourceID	1B	1	Identifikátor připojeného vstupního terminálu.
iTerminal	1B	0	Index na textový řetězec s popisem terminálu.

Tabulka 20: Deskriptor řídícího rozhraní zvuku - výstupní terminál.

Řízení probíhá přes řídící rozhraní zvuku (Audio control interface), které je vždy napojeno na koncový bod 0. V popisu tohoto rozhraní se skrývá i topologie. Skládá se z deskriptorů v tabulkách 17, 18, 19 a 20.

První deskriptor pouze určuje, že bude následovat popis rozhraní řízení zvuku třídy audio.

Následující deskriptor (tabulka 18) tvoří hlavičku třídně specifického popisu rozhraní. Specifikuje jednak použitou verzi USB Audio specifikace a také seznam rozhraní pro odesílání zvuku přes USB. Počet položek v tomto seznamu určuje hodnota pole bInCollection. Poté se na konec deskriptoru pro každé rozhraní přidá jedna položka. V tomto případě je použito pouze jediné rozhraní.

Zbylá dvojice deskriptorů (tabulka 19 a 20) popisuje vlastní topologii. Výstupní terminály mají vždy pole bSourceID, které se nastaví na hodnotu bTerminalType vstupního terminálu, se kterým je propojen.

Dále následuje sada deskriptorů popisující rozhraní pro odesílání zvuku (audio streaming interface). Stejně jako řídící rozhraní zde se začíná standardním deskriptorem roz-

Název pole	Délka	Hodnota	
bLength	1B	9	Délka deskriptoru.
bDescriptorType	1B	0x04	Typ deskriptoru.
bInterfaceNumber	1B	1	Pořadové číslo rozhraní.
bAlternateSetting	1B	0	Identifikátor alternativní nastavení.
bNumEndpoints	1B	0	Počet koncových bodů v tomto rozhraní.
bInterfaceClass	1B	1	Třída rozhraní. (Audio)
bInterfaceSubClass	1B	2	Podtřída rozhraní. (Streaming)
bInterfaceProtocol	1B	0	Vždy 0.
iInterface	1B	0	Index na textový řetězec.

Tabulka 21: Deskriptor rozhraní odesílání zvuku.

Název pole	Délka	Hodnota	
bLength	1B	9	Délka deskriptoru.
bDescriptorType	1B	0x04	Typ deskriptoru.
bInterfaceNumber	1B	1	Pořadové číslo rozhraní.
bAlternateSetting	1B	1	Identifikátor alternativní nastavení.
bNumEndpoints	1B	1	Počet koncových bodů v tomto rozhraní.
bInterfaceClass	1B	1	Třída rozhraní. (Audio)
bInterfaceSubClass	1B	2	Podtřída rozhraní. (Streaming)
bInterfaceProtocol	1B	0	Vždy 0.
iInterface	1B	0	Index na textový řetězec.

Tabulka 22: Deskriptor rozhraní odesílání zvuku.

hraní, který nese základní informace o rozhraní.

V tomto případě jsou použity hned dva deskriptory (tabulka 21 a 22). Liší se pouze ve dvou polích a to bAlternateSetting a bNumEndpoints. Druhé z těchto polí určuje počet koncových bodů použitých v rozhraní. Díky tomu, že je v prvním deskriptoru tento počet nulový, má host možnost zvolit toto alternativní nastavení vždy, když nemá zájem o zvuková data. K identifikaci tohoto nastavení slouží hodnota v prvním ze zmíněných polí bAlternateSettings.

Název pole	Délka	Hodnota	
bLength	1B	7	Délka deskriptoru.
bDescriptorType	1B	0x24	Typ deskriptoru.
bDescriptorSubType	1B	1	Podtyp deskriptoru.
bTerminalLink	1B	2	Index výstupního terminálu.
bDelay	1B	1	Zpoždění v paketech.
wFormatTag	2B	1	Formát přenášených dat. (PCM)

Tabulka 23: Deskriptor formátu zvuku - hlavička.

Název pole	Délka	Hodnota	
bLength	1B	11	Délka deskriptoru.
bDescriptorType	1B	0x24	Typ deskriptoru.
bDescriptorSubType	1B	2	Podtyp deskriptoru.
bFormatType	1B	1	Typ formátu.
bNrChannels	1B	2	Počet kanálů.
bSubFrameSize	1B	2	Velikost jednoho podrámce v bytech.
bSubFrameResolution	1B	16	Počet platných bitů jednoho podrámce.
bSamFreqType	1B	1	Způsob určení vzorkovací frekvence. (Diskrétní)
tSamFreq(1)	3B	48000	Vzorkovací frekvence v Hz.

Tabulka 24: Deskriptor formátu zvuku.

V tabulce 23 a 24 jsou deskriptory popisující formátování a způsob přenosu zvukových dat. Formátování dat je specifikováno v [4]. Podle tohoto dokumentu se proud zvukových dat dělí do paketů, které se odesílají každou 1 ms. Paket je složen z rámců (frames), které reprezentují úrovně všech kanálů zachycených v jeden okamžik. Samotné úrovně jednotlivých kanálů se nazývají podrámce (subframes).

Deskriptor třídně specifické hlavičky v tabulce 23 nese kromě polí pro identifikaci sebe sama pouze tři zajímavá pole.

První je bTerminalLink. Identifikuje, ke kterému výstupnímu terminálu se popis formátu vztahuje.

Pole bDelay je hodnota prodlevy mezi zachycením vzorku a jeho odesláním. Veličinou je počet paketů. Hodnota tohoto pole má význam pouze v případě simultánního záznamu zvuku z více zvukových karet a podobně. V případě přijímače rádia na ní nezáleží.

Posledním polem je wFormatTag, které určuje, že hodnoty jednotlivých vzorků budou ve formátu PCM jako čísla se znaménkem. I zde nabízí USB Audio specifikace značnou volnost a kromě formátu PCM je možné přenášet zvuk komprimovaný různými kodeky. Detailnější specifikace je v [4].

Následující deskriptor, vyobrazený v tabulce 24, blíže specifikuje parametry PCM formátu. Toto specifikuje hodnota pole bFormatType.

Jak název napovídá, pole bNrChannels určuje počet kanálů. Pole bSubframeSize obsahuje velikost jednoho podrámce (vzorku) v Bytech. Zároveň pole bSubFrameResolution nese počet platných bitů v podrámci.

Konec deskriptoru je věnován specifikaci vzorkovacích frekvencí. Pomocí hodnoty v poli bSamFreqType se určí způsob specifikace frekvencí. V případě modulu je použita pouze jediná diskrétní frekvence, která je specifikována v poli tSamFreq(1). Je možné zde specifikovat více diskrétních frekvencí, ze kterých si poté hostitel vybírá, a nebo specifikovat interval.

Název pole	Délka	Hodnota	
bLength	1B	7	Délka deskriptoru.
bDescriptorType	1B	0x25	Typ deskriptoru.
bEndpointAddress	1B	0x81	Číslo koncového bodu.
bmAttributes	1B	0b00000001	Bitová mapa s atributy.
wMaxPacketSize	2B	384	Maximální velikost paketu v bytech.
bInterval	1B	1	Interval Odesílání.
bRefresh	1B	0	Frekvence synchronizace. (Nepoužito)
bSynchAddress	1B	0	Adresa synchronizačního koncového bodu. (Nepoužito)

Tabulka 25: Deskriptor koncového bodu odesílání zvuku.

V tabulce 25 je deskriptor koncového bodu popisující koncový bod odesílání zvuku (audio streaming endpoint). Prvním zajímavým polem je adresa koncového bodu v poli bEndpointAddress. Jako adresa je použito číslo koncového bodu s tím, že v případě vstupního koncového bodu (směr do hostitele) se sedmý (nejvyšší) bit nastaví na 1. U opačného směru tento bit zůstává nulový.

Pole bmAttributes obsahuje bitovou mapu atributů koncového bodu. Význam bitů 0 a 1 je následující:

- 00 - Řídící koncový bod.
- 01 - Izochronní koncový bod.
- 10 - Hromadný koncový bod.
- 11 - Koncový bod přerušení.

Zbylé bity popisují způsob synchronizace izochronních koncových bodů, která ale není použita. Jejich popis je v kapitole 5.12.4 USB specifikace [2].

Pole wMaxPacketsize určuje největší možnou velikost paketu. V tomto případě se spočítá z deskriptoru v tabulce 24 jako $bNrChannels \cdot bSubFrameSize \cdot \frac{tSamFreq(1)}{1000}$ Vypočtenou hodnotu je nutné vždy zaokrouhlit nahoru. Například v případě vzorkovací frekvence 44,1 kHz bude ve většině paketů 44 rámců, ale každý desátý jich bude mít 45.

Pole bInterval určuje periodu čtení dat z koncového bodu. Hodnota 1 znamená, že se bude hostitel pokoušet číst data z koncového bodu každou 1 ms. Zbývající dvě pole se vztahují k synchronizaci, která nebyla použita a tudíž jsou nastavena na hodnotu 0.

Název pole	Délka	Hodnota	
bLength	1B	9	Délka deskriptoru.
bDescriptorType	1B	0x05	Typ deskriptoru.
bDescriptorSubType	1B	1	Podtyp deskriptoru.
bmAttributes	1B	0b00000000	Bitová mapa s atributy.
bLockDelayUnits	1B	0	Jednotky prodlevy stabilizace synchronizace. (Nepoužito).
wLockDelay	2B	0	Doba prodlevy stabilizace synchronizace. (Nepoužito)

Tabulka 26: Deskriptor koncového bodu odesílání zvuku specifický pro danou třídu.

Posledním deskriptorem popisujícím přenos zvuku je třídně specifický deskriptor koncového bodu v tabulce 26. Za běžnými uvozujícími poli se nachází bitová mapa s atributy bmAttributes. Význam bitů je následující:

- Bit 0 - Koncový bod podporuje žádost o počáteční nastavení vzorkovací frekvence.
- Bit 1 - Koncový bod podporuje dynamickou změnu vzorkovací frekvence za běhu.
- Bit 2-6 - Vyhrazeno - musí být nulové.
- Bit 7 - Pokud je nastaveno musí se přenášená data vždy doplnit nulami tak, aby měl každý paket velikost wMaxPacketSize.

Dvě pole na konci deskriptoru umožňují specifikovat dobu potřebnou pro stabilizaci synchronizace. Synchronizace koncového bodu není použita, takže obě hodnoty jsou nulové.

4.1.9 Deskriptory konfigurace vztažené k USB I²C tunelu

Název pole	Délka	Hodnota	
bLength	1B	9	Délka deskriptoru.
bDescriptorType	1B	0x04	Typ deskriptoru.
bInterfaceNumber	1B	2	Číslo rozhraní.
bAlternateSettngs	1B	0	Identifikátor alternativního nastavení.
bNumEndpoints	1B	2	Počet koncových bodů v tomto rozhraní.
bInterfaceClass	1B	255	Třída rozhraní (Vendor specific).
bInterfaceSubClass	1B	0	Podtřída rozhraní.
bInterfaceProtocol	1B	255	Protokol rozhraní (Vendor specific).
iInterface	1B	0	Index řetězce s popisem rozhraní (Nepoužit).

Tabulka 27: Deskriptor rozhraní USB - I²C tunelu.

Deskriptor ozhraní USB - I²C tunelu v tabulce 27 je podobný jako zvuková rozhraní. Liší se samozřejmě třídou zařízení, kde třída 255 určuje, že se nejedná o žádné ze standardizovaných rozhraní, ale je definováno výrobcem. Stejný význam má protokol číslo 255.

Název pole	Délka	Hodnota	
bLength	1B	9	Délka deskriptoru.
bDescriptorType	1B	0x05	Typ deskriptoru.
bEndpointAddress	1B	0x82 / 0x02	Číslo koncového bodu.
bmAttributes	1B	0b00000010	Bitová mapa s atributy.
wMaxPacketSize	2B	32	Maximální velikost paketu v bytech.
bInterval	1B	0	Interval Odesílání. (Nemá význam)
bRefresh	1B	0	Frekvence synchronizace. (Nemá význam)
bSynchAddress	1B	0	Adresa synchronizačního koncového bodu. (Nemá význam)

Tabulka 28: Deskriptor koncových bodů USB - I²C tunelu.

V tabulce 28 jsou oba deskriptory koncových bodů pro USB I²C tunel. Liší se pouze v adrese. Pole bmAttributes obsahuje pouze informaci o tom, že se jedná o koncový bod hromadného přenosu dat (bulk endpoint). Poslední tři pole mají v případě tohoto druhu koncového bodu vždy hodnotu 0.

Maximální velikost paketu wMaxPacketSize může být 8, 16, 32 nebo 64 B (viz. [2] kap. 5.8.3). Vzhledem k velikosti nejdelšího možného příkazu a odpovědi odesílaného přes I²C , jsem zvolil velikost 32 B.

4.2 Microchip Harmony framework

Harmony framework obsahuje mimo jiné také ovladače pro USB včetně USB Audio 1.0. Součástí je i vzorový projekt pro USB reproduktor, ze kterého jsem původně vycházel.

Poté co jsem projekt upravil tak, aby zvuková data nepřijímal, ale odesílal, se začal MCU restartovat. Z fóra firmy Microchip (`http://www.microchip.com/forums/FindPost/827487`) jsem zjistil, že jde o již nahlášený problém chybné inicializace, v důsledku které dochází k dělení nulou a k resetu MCU. Na fóru je také uveřejněno, jak tuto chybu opravit.

Opravením této chyby jsem se dostal k odesílání zvuku. I když modul správně prošel enumerací a konfigurací, do hostitele se přenášelo pouze ticho. Pomocí USB analyzátoru jsem zjistil, že modul od hostitele dostává IN tokeny, ale posílá pouze pakety nulové délky, jako by neměl žádná data k odeslání.

Rozhodl jsem se implementovat ovladač USB Audia jako vendor endpoint (koncový bod specifikovaný výrobcem). Pro potřeby modulu stačí pouze zajistit odesílání zvuku přes izochronní koncový bod a reagovat na požadavek změny alternativního nastavení funkce povolením nebo zakázáním izochronního koncového bodu. Došel jsem ke zcela totožnému chování modulu jako s původním ovladačem.

Upravil jsem tedy vlastní ovladač tak, aby izochronní koncový bod byl povolen neustále. Po této úpravě se začal přenášet zvuk, ale implementace vendor endpointu je pravděpodobně příliš pomalá, takže přibližně každý druhý paket byl následován paketem nulové délky.

4.3 Vlastní implementace USB

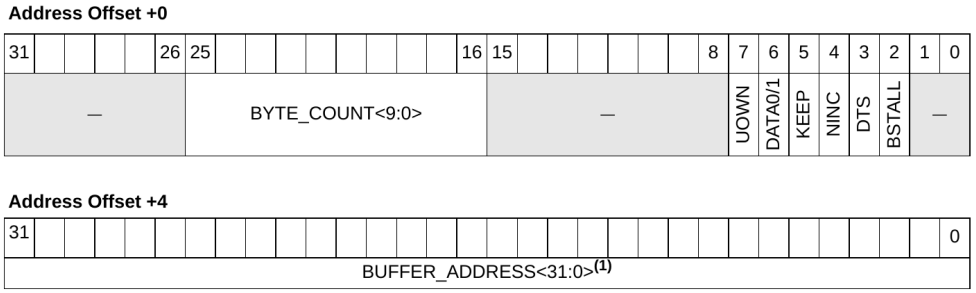
Kvůli zmiňovaným problémům s Harmony frameworkem, které se mi nepodařilo vyřešit, jsem se rozhodl napsat ovladač v MCU sám. Ke zprovoznění USB modulu je potřeba, aby MCU podporoval pouze několik požadavků hostitele a implementovat jednoduchý stavový stroj.

4.3.1 USB modul v MCU

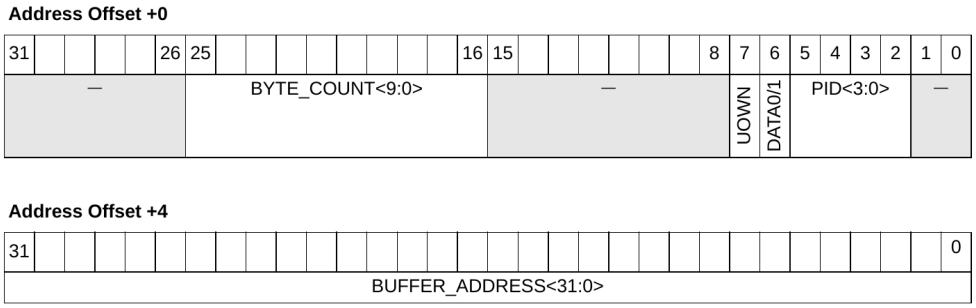
Dokumentace USB v MCU se nachází v obecném manuálu [12] pro celou řadu těchto 32 bitových MCU. Detaily pro konkrétní typ jsou v manuálu [8]. Pravděpodobně díky existenci Harmony frameworku je dokumentace relativně stručná.

USB modul v MCU PIC32MX je velmi podobný USB modulu v 8 bitovém MCU PIC16F1454, na kterém jsem se snažil původně modul tuneru postavit.

Prvním překvapením pro mne bylo, že podle USB specifikace nesmí být na sběrnici přivedeno žádné napětí, pokud není sběrnice napájena z hostitele. U zařízení napájených z USB sběrnice toto není třeba řešit. MCU PIC32 má vyhrazený vstup VBUS právě pro snímání napětí na USB sběrnici. I když je ho možné pomocí konfiguračních slov deaktivovat, je nutné jej použít. Pokud na tento vstup není přivedeno napětí, USB modul v MCU nepracuje.



Obrázek 6: Formát deskriptoru USB bufferu předaného do USB modulu. (Převzato z [12].)



Obrázek 7: Formát deskriptoru USB bufferu předaného do aplikace. (Převzato z [12].)

Vlastní konfigurace USB modulu je v případě USB zařízení v celku jednoduchá. Bitem USBPWR v registru U1PWRC se provede spuštění USB modulu.

Dále je potřeba do registrů U1BDTP1-3 nastavit dolní tři byty ukazatele na tabulku deskriptorů bufferů. Každému koncovému bodu, pro každý směr, jsou přiřazeny dva záznamy v této tabulce, lichý a sudý. Tvoří takto dvouprvkovou FIFO frontu. Každý záznam v této tabulce nese ukazatel na buffer pro příjem nebo odesílání dat a další nezbytné informace. Tabulka může být umístěna kdekoliv v zapisovatelné paměti MCU, ale její adresa musí být zarovnaná na 512 B. Jádro MCU používá rozdílné adresy pro přístup k datům z programu (virtuální adresy) a přístup k datům z hardwarových modulů (fyzické adresy). Adresu ukládanou do registrů U1BDTP Je tudíž potřeba přeložit. To se provede logickým součinem s hodnotou 0x1FFFFFFF.

Dále je potřeba zinicializovat samotnou tabulku deskriptorů bufferů. Jak je patrné z obrázků 6 a 7, význam dat v jednotlivých záznamech tabulky je mírně odlišný v případě, že tabulku předáváme do USB modulu a v případě, že USB modul zpracoval token pro daný koncový bod a tabulku vrátil aplikaci. Význam jednotlivých polí je následující:

- **BYTE_COUNT** - Počet bytů k přijetí nebo odeslání.
- **UOWN** - Pokud je nastaven na 1, buffer patří USB modulu a nelze jej měnit. Po zpracování paketu bude tento bit nastaven USB modulem zpět na hodnotu 0.

- **DATA0/1** - Určuje jestli se paket odešle s PID DATA0 nebo DATA1. V případě příjmu určí jestli se očekává paket s PID DATA0 nebo DATA1.
- **KEEP** - Pokud je nastaven na 1, USB modul si tento buffer ponechá napořád.
- **NINC** - Pokud je nastaven na adresa DMA nebude inkrementována.
- **DTS** - Pokud je nastaven na 1, USB modul bude při přijmu ignorovat pakety, kde PID (DATA0/1) neodpovídá hodnotě pole DATA0/1. Pokud je nastaven na 0, tato kontrola nebude prováděna.
- **BSTALL** - Pokud je nastaven na 1, byl přijat nebo bude vyslán STALL.
- **PID** - Druh přijatého tokenu.
- **BUFFER_ADDRESS** - Fyzická adresa na buffer s data k odeslání nebo přijmu.

Jak je patrné z předchozího výčtu je nezbytné inicializovat přinejmenším bit UOWN na hodnotu 0.

Po tomto kroku je možné spustit USB modul nastavením bitu USBEN v registru U1CON na 1.

Aplikace si musí udržovat informace o tom jestli má používat lichý nebo sudý buffer. USB modul umožňuje resetovat všechny koncové body v obou směrech tak, aby znova začaly sudým bufferem, nastavením bitu PBRST v registru U1CON na 1.

Ještě je nutné nakonfigurovat používané koncové body v registrech U1EP0 - U1EP15. Nastavením bitů EPTXEN a nebo EPRXEN odesílání a nebo příjem, pokud se nejedná o koncový bod pro řídící přenosy, je dobré zakázat příjem SETUP tokenů nastavením bitu EPCONDIS. A pokud se nejedná o koncový bod pro isochronní přenos dat povolit handshake nastavením bitu EPHSK.

Pro příjem a odesílání stačí nastavit příslušný záznam v tabulce deskriptorů bufferů s nastavením bitu UOWN jej předat do USB modulu. Po vyřízení přijmu nebo odesílání USB modul nastaví bit TRNIF v registru U1IR a současně nastaví v registru U1STAT tato pole:

- **ENDPT** - Číslo koncového bodu 0-15.
- **DIR** - Směr. Pokud má hodnotu 1 poslední transakce bylo odesílání (IN transakce), pokud 0 poslední transakce byla příjem (OUT transakce).
- **PPBI** - Pokud má hodnotu 1 byl vyřízen lichý buffer, pokud 0, byl vyřízen sudý buffer.
- **2 bity mezera.**

Pakliže je hodnota tohoto registru posunuta o dva bity doprava, je ji možné přímo použít jako index v tabulce deskriptorů bufferů.

Při přijmu setup tokenu (zahájení řídící transakce) dojde k nastavení bitu PKTDIS registru U1CON. Dokud je tento bit nastaven je zastaveno zpracování všech tokenů. Po připravení odpovědi je nutné tento bit nastavit na hodnotu 0.

Aplikace by dále měla monitorovat v registru U1CON tyto bity: (Více v kapitole 4.3.3.)

- **IDLEIF** - Zařízení bylo suspendováno.
- **URSTIF** - Zařízení bylo resetováno.

4.3.2 Standardní USB požadavky

Název pole	Délka	
bmRequestType	1B	Bitová mapa s obecným popisem požadavku.
bRequest	1B	Specifikace požadavku.
wValue	1B	Obecná hodnota.
wIndex	1B	Obecný index.
wLength	1B	Velikost dat, které je nutné dále přijmou nebo odeslat.

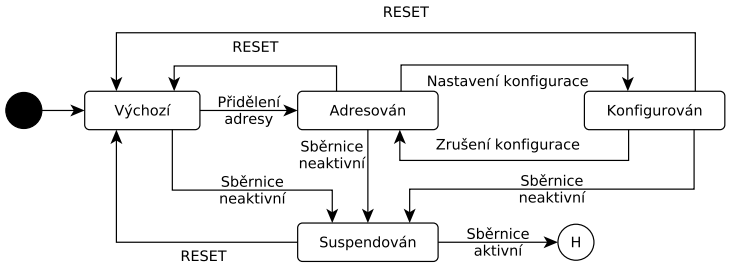
Tabulka 29: Hlavička USB požadavků.

Standardní požadavky na zařízení jsou detailně popsány v [2] v kapitole 9.3. Implementoval jsem pouze požadavky nutné k funkci modulu rádia.

Standardní požadavky odesílané z hostitele mají vždy stejnou hlavičku popsanou v tabulce 29. Význam polí wValue a wIndex je vždy specifický danému požadavku. Význam bitů pole bmRequestType je náledující:

- Bit 7: Směr případné datové části.
  - 1 - Z hostitele do zařízení,
  - 0 - Ze zařízení do hostitele.
- Bity 6-5: Typ požadavku.
  - 0 - Standardní požadavek.
  - 1 - Požadavek specifický pro danou třídu zařízení.
  - 2 - Výrobcem specifikovaný požadavek.
  - 3 - Vyhrazeno.
- Bity 4-0: Příjemce požadavku.
  - 0 - Zařízení.
  - 1 - Rozhraní.
  - 2 - Koncový bod.
  - 3 - Jiný.
  - 4-31 - Vyhrazeno.

Byly implementovány následující požadavky: (V závorkách jsou uvedeny hodnoty bRequest)



Obrázek 8: Zjednodušený diagram stavů USB zařízení.

- **GET_STATUS** (0): Odpovědí na tento požadavek jsou dva byty. Předposlední bit určuje, jestli se zařízení smí probudit, když je suspendováno (remote wakeup). Toto není podporováno, bit je vždy nula. Poslední nejnižší bit určuje, jestli má zařízení vlastní napájení. Tento bit je taktéž vždy nula. Ostatní bity jsou rezervovány a jsou vždy nulové.
- **SET_ADDRESS** (5): V poli wValue zařízení obdrží novou adresu.
- **GET_DESCRIPTOR** (7): Hostitel žádá o deskriptor. V horním byte wValue je typ deskriptoru, v dolním byte index deskriptoru. wIndex je buď nula nebo specifikuje ID jazyka. wLength určuje maximální délku odpovědi. Hostitel může požadovat pouze začátek deskriptoru.
- **GET_CONFIGURATION** (8): Hostitel žádá o index aktuálně používané konfigurace. Ta je odeslána jako jeden byte.
- **SET_ADDRESS** (5): V poli wValue zařízení obdrží index nové konfigurace.
- **SET_INTERFACE** (11): V poli wValue zařízení obdrží index nového alternativní nastavení pro rozhraní s indexem v poli wIndex. (Implementováno v ovladači USB audio.)

4.3.3 Stavy USB zařízení

Na obrázku 8 je stavový diagram USB zařízení. Diagram je zjednodušen pouze na stavy důležité pro modul.

- **Výchozí:** Stav po zapnutí a inicilizaci zařízení. Při vstupu do tohoto stavu se nastaví adresa zařízení na výchozí hodnotu 0.
- **Adresován:** Do tohoto stavu se přejde po zpracování požadavku nastavení adresy. Veškerá další komunikace bude probíhat na této adrese. Ovšem adresu do registru MCU U1ADDR je nutné nastavit až po dokončení transakce - odeslání paketu nulové délky.

- **Konfigurován:** Do tohoto stavu se přejde na základě požadavku nastavení konfigurace.
- **Suspendován:** K suspendování zařízení může dojít obecně kdykoliv. Do tohoto stavu se přejde pokud na sběrnici není zaznamenána aktivita po dobu 3 ms. Toto detekuje hardware MCU. V tomto stavu musí zařízení snížit svoji spotřebu na nejvýše 0,5 mA.

Dále je potřeba počítat s faktem, že zařízení může být kdykoliv resetováno a vrátí se tak do výchozího stavu, není stanoveno pořadí, v jakém budou žádosti přicházet a kterákoli transakce může být předčasně ukončena vysláním dalšího setup tokenu.

4.3.4 Rozhraní implementace USB

Zkonzultovat a dodělat t...

Implementace se nachází v projektu firmware ve složce drv.

V souboru config.h se nastaví počet konfigurací, celkový počet deskriptorů řetězců, nejvyšší použitý koncový bod a specifikuje se jestli má zařízení vlastní napájení.

V souboru usb.h je vlastní rozhraní.

4.4 USB I²C tunel

USB - I²C tunelu je zjednodušen tak, aby vyhovoval potřebám ovládání tunerů. Není obecně použitelný pro jakoukoliv I²C komunikaci. Umožňuje pouze odeslání jedné ze čtyř pevně specifikovaných adres, následované čtením nebo zápisem až 16-ti bytů.

Název pole	Délka	
id	1 B	Identifikátor požadavku.
tuner	1 B	Číslo tuneru.
type	1 B	Typ požadavku.
rw_size	1 B	Velikost pole cmd v bytech.
cmd	0 - 16 B	Případná data odesílána po I²C vyjma adresy.

Tabulka 30: Formát požadavku USB - I²C tunelu.

Tabulka 30 popisuje formát požadavku pro čtení nebo zápis dat na I²C sběrnici. Význam polí je následující:

- **id** - Hodnota 0-255, kterou je možné použít pro přiřazení odpovědi k požadavku.
- **tuner** - Výběr tuneru na modulu rádia. 0 pro hlavní tuner A, ze kterého je přenášen zvuk, 1 pro tuner B.
- **type** - Typ požadavku. 0 Pro zápis na I²C sběrnice, 1 pro čtení z I²C sběrnice, 2 pro ping - modul v odpovědi pouze vrátí přijatá data.

- **rw_size** - V případě zápisu velikost pole cmd v bytech, v případě čtení počet bytů, které budou přečteny. Hodnota 0 - 16.
- **cmd** - V případě zápisu data vysílaná na I^{2}C sběrnici.

Název pole	Délka	
id	1 B	Identifikátor požadavku.
error	1 B	Číslo tuneru.
reply	0 - 16 B	Případná odpověď tuneru z I^{2}C sběrnice.

Tabulka 31: Formát odpovědi USB - I^{2}C tunelu.

- **id** - Hodnota 0-255, kterou je možné použít pro přiřazení odpovědi k požadavku.
- **error** - Vyjadřuje, jestli při vyřizování požadavku došlo k chybě či nikoliv.
  - 0 - Nedošlo k chybě.
  - 1 - I^{2}C sběrnice je zaneprázdněna, probíhá komunikace.
  - 2 - Nevalidní požadavek, neplatné číslo tuneru nebo příliš velké množství dat.
  - 3 - Tuner není spuštěn a inicializován.
  - 4 - Předchozí požadavek ještě nebyl vyřízen.
  - 5 - Neplatný typ požadavku.
  - 128 - Obecná chyba.
- **reply** - V případě úspěšného požadavku čtení, odpověď tuneru.

Komunikace z hostitele se provádí pomocí požadavků a odpovědí popsaných v tabulkách 30 a 31. Výjimkou je požadavek s typem 2 (ping), kdy modul rádia odešle zpět stejná data, která přijal. K práci s USB - I^{2}C tunelem bych chtěl podotknout, že není možné odeslat další požadavek před vyřízením předchozího. Po spuštění modulu rádia dochází k počáteční inicializaci tunerů. Podobu inicializace modul odpovídá chybovým kódem 3.

Vlastní modul I^{2}C v MCU je dobře zdokumentován, jednak obecně pro celou rodinu MCU v [10], detaily specifické pro mnou použitý typ MCU jsou potom v [8].

Je zbytečné popisovat zde konfiguraci I^{2}C modulu. Chtěl bych ale upozornit na dvě hardwarové chyby, které se modulu týkají. Obě jsou popsané v [13].

Při spuštěném modulu I2C1 nelze používat vývody RA0 a RA1, stejně tak při spuštěném modulu I2C2 nelze používat vývody RB5 a RB6. Druhou chybou je možnost dvojitého zápisu v případě, že během zápisu došlo k přerušení. Toto chování ve většině případů není problém. V případě I^{2}C modulu se se zápisem spouští jednotlivé akce, takže může dojít například k dvojímu zápisu bytu na sběrnici. Řešením je před zápisem do registrů náchylných na tuto chybu zakázat přerušení.

5 Knihovna

DFD diagram asi není rozumný, ale určitě sem minimálně nakreslit rozvrstvení včetně vrstev v tuneru.

Dotaz asi na vedoucího - Přepisovat dokumentaci do češtiny a nebo se na ni jenom odkázat a omezit se na souhrn funkcí s vhodnými příklady.

5.1 Nízko úrovňové funkce

5.2 Středně úrovňové funkce

5.3 Vysoko úrovňové funkce

5.3.1 RDS dekodér

5.4 Překlad

6 Závěr

TODO:

- Zmínit že příjem na PCB je mnohem kvalitnější než v nepájivém poli a změřit to pomocí SNR
- Nemožnost dostat se v suspendu do 0,5 mA
- Test mód neimplementován. Nutný pouze pro získání USB loga.

Bc. Pavel Kovář

7 Reference

[1] AXELSON, Jan. *USB complete: the developer's guide.* 4th ed. Madison, WI: Lakeview Research, 2009, xxiii, 504 p. ISBN 1-931448-08-6.

[2] Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. *Universal Serial Bus Specification: Revision 2.0* [online] 2000-04-27 [2015-12-26] http://www.usb.org/developers/docs/usb20_docs/usb_20_0702115.zip

[3] Gal Ashour, Billy Brackenridge, Oren Tirosh, Altec Lansing, Craig Todd, Remy Zimmermann, Geert Knapen. *Universal Serial Bus Device Class Definition for Audio Devices: Release 1.0* [online] 1998-03-18 [2015-12-26] http://www.usb.org/developers/docs/devclass_docs/audio10.pdf

[4] Gal Ashour, Billy Brackenridge, Oren Tirosh, Altec Lansing, Craig Todd, Remy Zimmermann, Geert Knapen. *Universal Serial Bus Device Class Definition for Audio Data Formats* [online] 1998-03-18 [2016-02-20] http://www.usb.org/developers/docs/devclass_docs/frmts10.pdf

[5] USB Implementers' Forum. *Universal Serial Bus Language Identifiers (LANGIDs): Version 1.0* [online] 2000-03-26 [2016-02-09] http://www.usb.org/developers/docs/USB_LANGIDs.pdf

[6] Silicon Laboratories, *Si4730/Si4731/Si4734/Si4735-D60 Broadcast AM/FM/SW/LW Radio Receiver: Rev. 1.2 8/13* [online] 2013-08-08 [2015-12-26] https://www.silabs.com/Support%20Documents/TechnicalDocs/Si4730-31-34-35-D60.pdf

[7] Silicon Laboratories, *AN332: Si47xx Programming Guide: Rev. 1.0 9/14* [online] 2014-09-10 [2015-12-26] http://www.silabs.com/Support%20Documents/TechnicalDocs/AN332.pdf

[8] Microchip Technology Inc. *PIC32MX1XX/2XX Family Data Sheet: Revision H* [online] 2015-07-29 [2015-12-26] http://ww1.microchip.com/downloads/en/DeviceDoc/60001168H.pdf

[9] Microchip Technology Inc. *PIC32 Family Reference Manual, Sect. 23 Serial Peripheral Interface* [online] 2011-10-11 [2015-12-26] http://ww1.microchip.com/downloads/en/DeviceDoc/61106G.pdf

[10] Microchihttp://ww1.microchip.com/downloads/en/DeviceDoc/61116F.pdp Technology Inc. *PIC32 Family Reference Manual, Sect 24. Inter-Integrated Circui* [online] 2013-03 [2015-12-26] http://ww1.microchip.com/downloads/en/DeviceDoc/61116F.pdf

[11] Microchip Technology Inc. *PIC32 Family Reference Manual, Sect. 31 DMA Controller* [online] 2013-11-15 [2015-12-26] http://ww1.microchip.com/downloads/en/DeviceDoc/60001117H.pdf

[12] Microchip Technology Inc. *PIC32 Family Reference Manual, Sect. 27 USB On-The-Go* [online] 2011-04-13 [2015-12-26] http://ww1.microchip.com/downloads/en/DeviceDoc/61126F.pdf

[13] Microchip Technology Inc. *PIC32MX1XX/2XX 28/36/44-pin Family Silicon Errata and Data Sheet Clarification* [online] 2015-07-29 [2015-12-26] http://ww1.microchip.com/downloads/en/DeviceDoc/80000531G.pdf

[14] Microchip Technology Inc. *MPLAB Harmony USB Libraries Help* [online] 2012-11-15 [2015-12-26] http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB%20Harmony%20USB%20Libraries%20%28v1.06.02%29.pdf

[15] *TAS1020B USB Streaming Controller* [online] 2011-05 [2015-12-29] http://www.ti.com/lit/ds/symlink/tas1020b.pdf

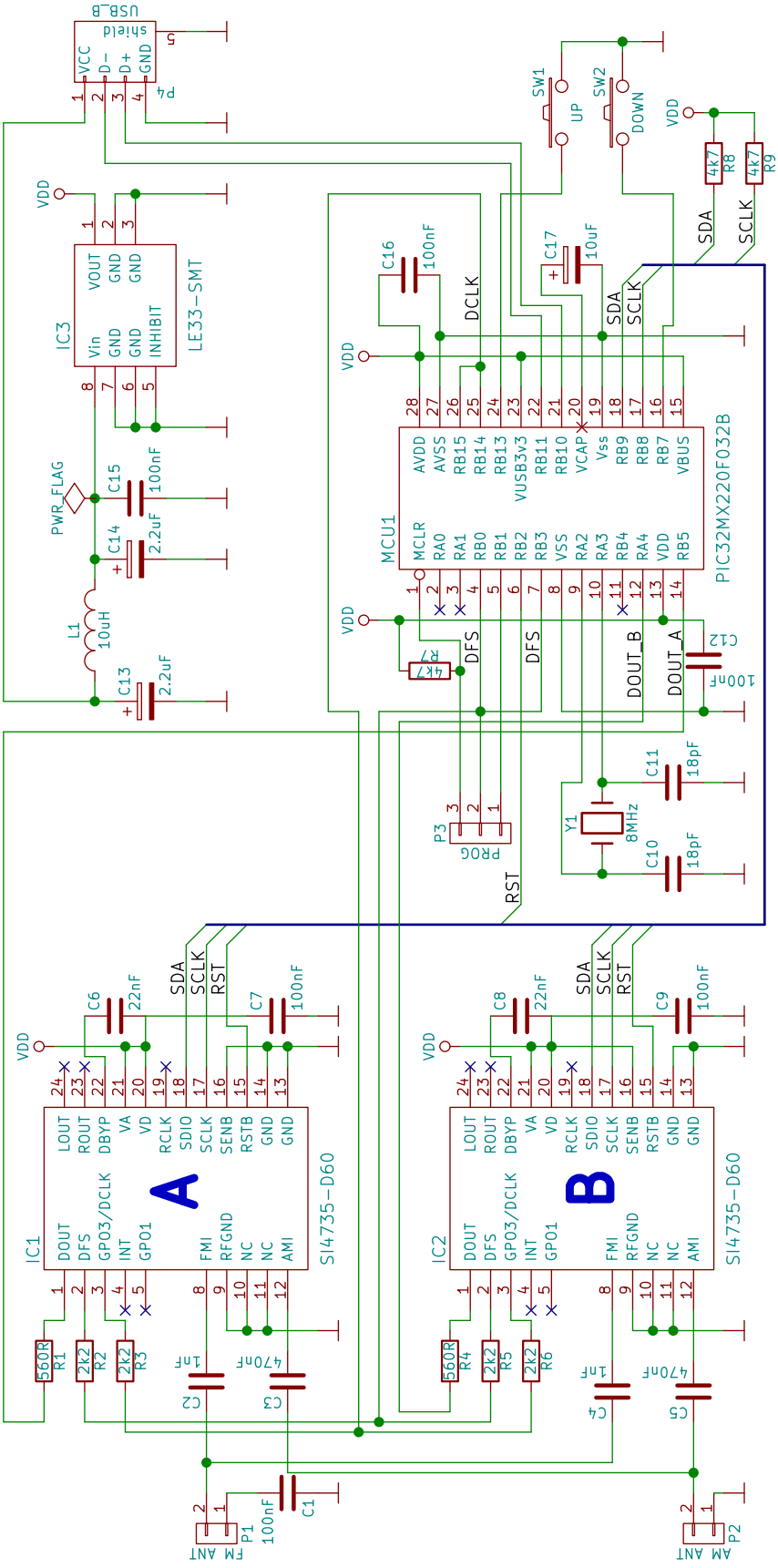
[16] STMicroelectronics *Very low-dropout voltage regulator with inhibit function* [online] 03-2014 [2016-01-09] http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00000545.pdf

[17] USB.org *Getting a Vendor ID* [online] [2016-01-16] http://www.usb.org/developers/vendor/

[18] Philips Semiconductors *I^2S bus specification* [online] 05-06-1996 [2016-02-27] http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00000545.pdf

[19] EN 50067 *Specification of the radio data system (RDS) for VHF/FM sound broadcasting in the frequency range from 87,5 to 108,0 MHz* CELNEC 04-1998 [2016-04-01]

A Schéma zapojení modulu



B Rozpiska součástek