ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA ELEKTROENERGETIKY A EKOLOGIE

DIPLOMOVÁ PRÁCE

ROZHRANÍ PRO DIAGNOSTIKU TV PŘIJÍMAČŮ

Václav Vůcha **2012**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2011/2012**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Václav VŮCHA**
Osobní číslo:	**E10N0038K**
Studijní program:	**N2644 Aplikovaná elektrotechnika**
Studijní obor:	**Aplikovaná elektrotechnika**
Název tématu:	**Rozhraní pro diagnostiku TV přijímačů**
Zadávající katedra:	**Katedra elektroenergetiky a ekologie**

Zásady pro vypracování:

1. Navrhněte a realizujte přípravek umožňující pomocí USB sběrnice zápis/čtení na sběrnici I2C a SPI.
2. Přípravek musí disponovat dalšími alespoň třemi číslicovými výstupy pro ovládání TV.
3. Přípravek musí podporovat I/O s rozsahy 5V a 3,3V. Přípravek by měl mít co možná nejmenší rozměry, aby zařízení bylo snadno přenositelné.
4. Vytvořte v jazyce Delphi pro Windows řídící aplikaci umožňující komunikaci s TV pomocí realizovaného přípravku.
5. Realizaci v práci velmi podrobně popište.

Rozsah grafických prací: **podle doporučení vedoucího**
Rozsah pracovní zprávy: **30 - 40 stran**
Forma zpracování diplomové práce: **tištěná/elektronická**
Seznam odborné literatury:

1. **Potřebnou literaturu dodá konzultant.**
2. **Další vhodnou literaturu si student vyhledá v dostupných pramenech.**

Vedoucí diplomové práce: **Doc. Ing. Martin Poupa, Ph.D.**
Katedra aplikované elektroniky a telekomunikací

Datum zadání diplomové práce: **17. října 2011**
Termín odevzdání diplomové práce: **11. května 2012**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

Západočeská univerzita v Plzni * Fakulta elektrotechnická *

Doc. Ing. Karel Noháč, Ph.D.
vedoucí katedry

V Plzni dne 17. října 2011

Anotace

Diplomová práce se zabývá návrhem a realizací zařízení, které obsahuje převodník ze sběrnice USB na sběrnice I2C a SPI. Zařízení primárně navrženo jako rozhraní mezi PC a TV přijímačem značky Panasonic. Práce je rozdělena na teoretickou a praktickou část. Teoretická část popisuje funkce sběrnic I2C a SPI a funkci převodníku FT4232H od firmy FTDI.
V praktické části jsou popsány základní příkazy pro ovládání převodníku, schéma zapojení a samotná realizace zařízení.

Abstract

The interface for diagnostic of TV sets.

This master's thesis deals with designing and implementation of device which contains a converter from the USB bus to the I2C and SPI buses. The device is primarily designed as interface between a PC and TV set made by Panasonic. The project is separated to a theoretical and practical part. The theoretical part describes the functions of the I2C and SPI buses and the function of the converter FT4232H from FTDI Company. There are described the primary commands of the converter, the wiring and realization of device in the practical part.

Klíčová slova

USB, I2C, SPI, FT4232H, FTDI, FT příkazy, D2XX, převodník, sběrnice

Keywords

USB, I2C, SPI, FT4232H, FTDI, FT commands, D2XX, converter, buses

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

V Plzni dne 11.5.2012 Václav Vůcha

……..

Poděkování

Rád bych poděkoval vedoucímu diplomové práce Doc. Ing. Martinu Poupovi, Ph.D. a konzultantovi Ing. Patriku Průchovi za cenné rady a připomínky při návrhu a realizaci zařízení.

Obsah

Úvod 9

Seznam symbolů a zkratek 10

1. Sběrnice I2C a SPI 11

1.1 Sběrnice SPI 11

1.1.1 Komunikace mezi uzly sběrnice 12

1.1.2 Vlastnosti sběrnice SPI 13

1.2 Sběrnice I2C 14

1.2.1 Komunikace mezi uzly sběrnice 14

1.2.2 Vlastnosti sběrnice I2C 16

2. Popis důležitých součástek zapojení 17

2.1 Převodník FT4232H 17

2.1.1 Zapojení převodníku 18

2.1.2 Ovládání převodníku 19

2.2 Multiplexer PCA9544A 20

2.3 Napěťový shifter PCA9517 22

2.4 Napěťový shifter 74LVC8T245 24

3. Schéma zapojení 25

3.1 Orientace ve schématu 25

3.2 Popis schématu zapojení 26

3.2.1 Napájení 27

3.2.2 Součástky kolem převodníku FT4232H 28

3.2.3 Zapojení sběrnice I2C 29

3.2.4 Zapojení sběrnice SPI 30

3.2.5 Obvod ovládání shifterů 30
3.2.6 Obvod digitálních vstupů a výstupů 30
3.2.7 Přepěťová ochrana vstupů rozhraní 32
4. Realizace rozhraní 33
4.1 Návrh 33
4.2 Výroba rozhraní 35
4.2.1 Výroba desky plošných spojů 35
4.2.2 Osazování součástek 36
4.2.3 Oživení 36
4.2.4 Krabička 37
4.3 Ovládací aplikace 37
4.3.1 Ovládání programu 38
4.3.2 Popis použitých funkcí 39
Závěr 43
Seznam použité a citované literatury 44
Příloha A - Schémata 45
Příloha B – Seznam součástek 47
Příloha C – Deska plošných spojů 49
Příloha D – Osazení součástek 51
Příloha E – Výpis zdrojového kódu ovládací aplikace 53
Příloha F – Obsah přiloženého CD 74

Úvod

Tato práce se zabývá návrhem a realizací rozhraní, které primárně slouží k propojení PC a TV přijímače po sběrnici I2C. Jedná se v podstatě o převodník z USB sběrnice na sběrnice I2C a SPI, takže je možné zařízení využít i k jiným účelům například programování sériových pamětí podporující jednu z těchto sběrnic nebo k ovládání a komunikaci s jinými elektronickými zařízeními pracujících s těmito sběrnicemi. Rozhraní dále obsahuje několik digitálních vstupů a výstupů, které slouží jako pomocné ovládací a signalizační obvody připojených periferií.

V teoretické části je ve stručnosti popsána funkce I2C a SPI sběrnic (průběhy signálů, adresování připojených zařízení atd.), jejich výhody, nevýhody, vlastnosti a použití. Dále je zde popsán převodník FT4232H od firmy FTDI, který je základem celého zapojení..

Praktická část obsahuje informace o některých použitých integrovaných obvodech, jejich funkcích a vhodném zapojení. Dále je zde schéma zapojení celého zařízení s vysvětlením funkce jako celku, k obvodu FT4232H jsou zde uvedeny základní příkazy k ovládání. Konec práce se zabývá samotnou realizací zařízení a jeho odzkoušení.

Seznam symbolů a zkratek

ACK	Acknowledge	Potvrzení příjmu dat
C		Označení kondenzátoru
D		Označení diody
DPS		Deska plošných spojů
GND	Ground	Zemní potenciál
I2C	Inter-Integrated Circuit	Druh sériové sběrnice
IO		Označení integrovaného obvodu
JTAG	Joint Test Action Group	Druh sériové sběrnice
LED	Light-Emitting Diode	Světlo emitující dioda
MISO	Master In, Slave Out	Signál sběrnice SPI
MOSI	Master Out, Slave In	Signál sběrnice SPI
MPSSE	Multi-Protocol Synchronous Serial Engine Interface	
		Rozhraní pro tvorbu sériových sběrnic
R		Označení rezistoru
RS232		Druh sériové sběrnice
RS422		Druh sériové sběrnice
RS485		Druh sériové sběrnice
S		Označení spínače
SCK, SCL	Serial Clock	Hodinový signál
SDA	Synchronous Data	Signál sběrnice I2C
SMD	Surface Mount Device	Povrchově montované součástky
SPI	Serial Peripheral Interface	Druh sériové sběrnice
SS	Slave Select	Signál sběrnice SPI
T		Označení tranzistoru
UART	Universal Asynchronous serial Reciever and Transmitter	
		Zařízení umožňující vysílání i příjem a synchronních sériových dat
USB	Universal Serial Bus	Univerzální sériová sběrnice
X		Označení konektoru
XT		Označení krystalu

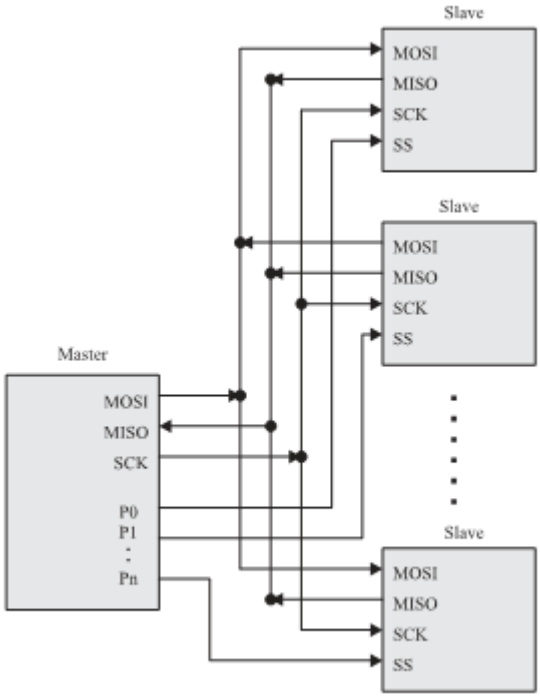
1. Sběrnice I2C a SPI

Sběrnice SPI (Serial Peripheral Interface) a I2C (Inter-Integrated Circuit) jsou dnes hojně rozšířené v mnoha digitálních zařízeních. Obě sběrnice používají sériový přenos dat, který minimalizuje potřebný počet propojovacích vodičů.

Velké rozšířen těchto sběrnic je zapříčiněno jejich jak softwarovou, tak hardwarovou jednoduchostí. Z hardwarového hlediska stačí ke komunikaci jen několik (u I2C dva, u SPI čtyři) vodiče. To dělá konstrukci periférií jednodušší s menšími rozměry. Díky malému počtu potřebných vodičů vychází také menší potřebný plošný spoj. Ze softwarového hlediska je použití sběrnic také poměrně jednoduché. Popis je v následujících odstavcích.

1.1 Sběrnice SPI

Sběrnice SPI je čtyřvodičová sériové sběrnice, kde vždy jeden uzel komunikačního obvodu funguje jako řadič sběrnice (master) a ostatní uzly jako slave. Propojení jednotlivých uzlů sběrnice je vidět na **Obrázku 1.**:



Obrázek 1.: *Propojení uzlů sběrnice SPI. Převzato z [1]*

Popis a funkce jednotlivých vodičů:

SCK (Serial Clock) – přenos hodinového signálu

MISO (Master In, Slave Out) – přenos dat z uzlu slave do master

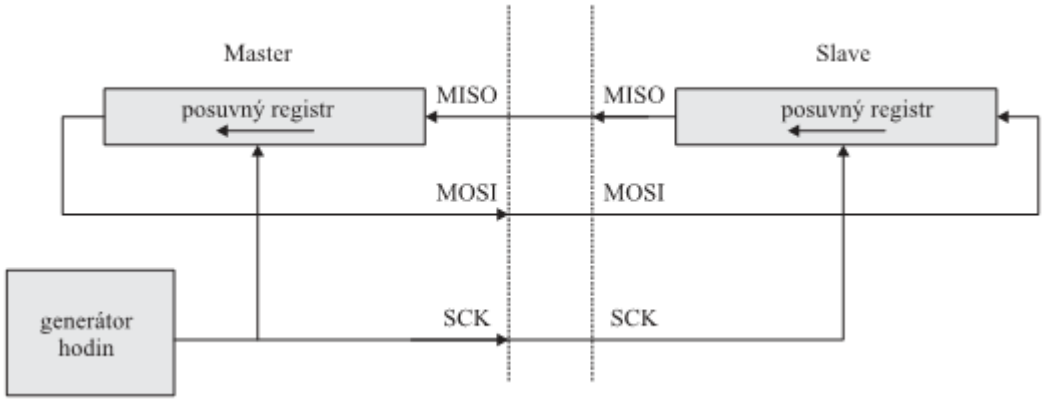
MOSI (Master Out, Slave In) – přenos dat z uzlu master do slave

SS (Slave Select) – výběr zařízení slave

Z obrázku a popisu jednotlivých vodičů je zřejmé, že pro přenos signálů si zařízení vystačí s jednosměrnými porty, což opět přispívá k zjednodušení hardwarové implementace.

1.1.1 Komunikace mezi uzly sběrnice

Přenos dat probíhá vždy mezi uzlem master a některým z uzlů slave. Výběr uzlu slave provádí uzel master aktivací vodiče SS příslušného uzlu slave. Pokud uzel slave není aktivován, udržuje vodič MISO ve vysoké impedanci. Oba uzly obsahují posuvné registry, jak je vidět na **Obrázku 2.:**



Obrázek 2.: *Propojení posuvných registrů. Převzato z [1]*

Uzel master vysílá hodinový signál po vodiči SCK, který je zaveden do všech uzlů slave. Při každém taktu hodinového signálu dojde k posunu bytů v registrech tak, že z registrů se vždy vysune bit s nejvyšší platností, tj. bit z registru uzlu slave se objeví na vodiči MISO, bit z registru uzlu master na vodiči MOSI. Bity jsou při dalším taktu zapsány do registru druhého

uzlu jako bity s nejnižší platností. Z toho plyne, že příjem a vysílání bitu mezi uzlem master a slave probíhá ve stejné době, tj. přenos je synchronní, obousměrný.

K posunu registru může dojít při náběžné nebo sestupné hraně hodinového signálu. Jak jednotlivé uzly na hodinový signál reagují, je dáno vlastnostmi připojeného uzlu a jsou buď konfigurovatelné, nebo nikoliv. Zařízení v uzlu master je obvykle programově řízené zařízení, které lze nakonfigurovat dle potřeby právě vybraného uzlu slave.

1.1.2 Vlastnosti sběrnice SPI

Jelikož z hardwarového hlediska je sběrnice v podstatě realizovaná posuvnými registry a přenos dat je synchronní, je možné na sběrnici dosáhnout větších přenosových rychlostí. Běžně se používá frekvence hodinového signálu 1 - 10MHz, ale při krátké vzdálenosti uzlů a malé kapacitě spoje je možné použít frekvenci až 70MHz.

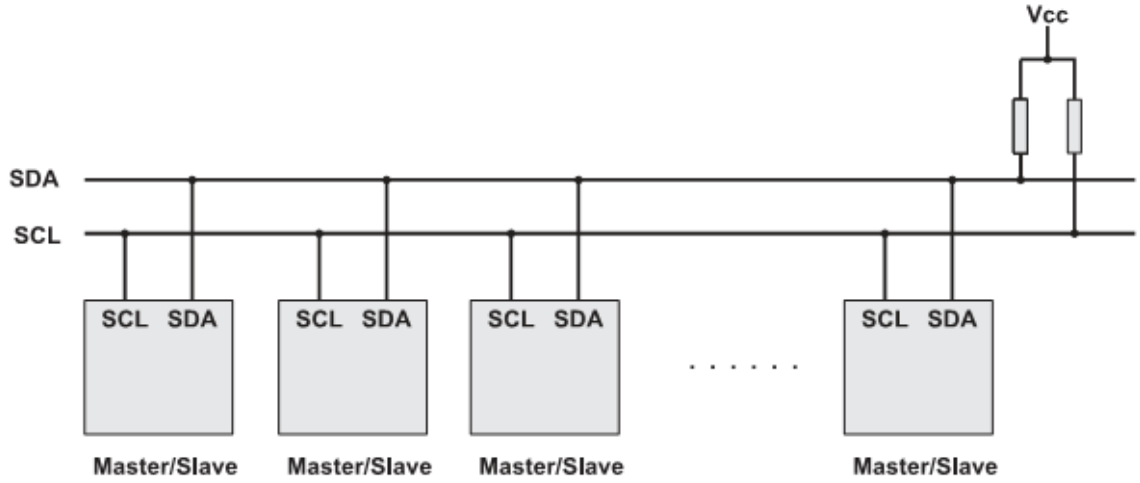
Mezi výhody sběrnice SPI určitě patří jednoduchá hardwarová implementace, kde, jak již bylo poznamenáno výše, pro příjem a vysílání dat postačí posuvné registry. Další výhodou je obousměrný přenos, nemusí tedy docházet k přepínání mezi vysíláním a příjmem, což zmenšuje nároky na softwarové řízení sběrnice.

Nevýhodou sběrnice je fakt, že v propojení komunikačních uzlů může existovat pouze jeden uzel master. Mezi nevýhody se také musí započítat, že vzdálenost mezi uzly musí být poměrně krátká (několik metrů). Důvodem je nutnost zachování synchronizace mezi hodinovým signálem a daty, kde všechny signály musí mít stejné zpoždění a neexistence signálu acknowledge, který by potvrzoval zpracování dat pomalejšími uzly. Další nevýhodou je různé zpracování hodinového signálu (náběžná nebo sestupná hrana), což může působit problémy při zapojování nových uzlů do sítě. Poslední nevýhodou je nutnost použití čtyř vodičů.

Sběrnice SPI se používá pro komunikaci mezi mikroprocesory, k připojení a programování pamětí atd.

1.2 Sběrnice I2C

Sběrnice I2C je dvouvodičová sériová sběrnice, kde každý uzel komunikační sítě má danou vlastní jedinečnou adresu. Propojení jednotlivých uzlů sběrnice je vidět na **Obrázku 3.**:



Obrázek 3.: *Propojení uzlů sběrnice I2C. Převzato z [1]*

Vodič SCL (Serial Clock) slouží k přenosu hodinového signálu, vodič SDA (Serial Data) k přenosu dat. Z obrázku je zřejmé, že pro přenos dat se používá pouze jeden vodič, tj. sběrnice I2C je poloduplexní.

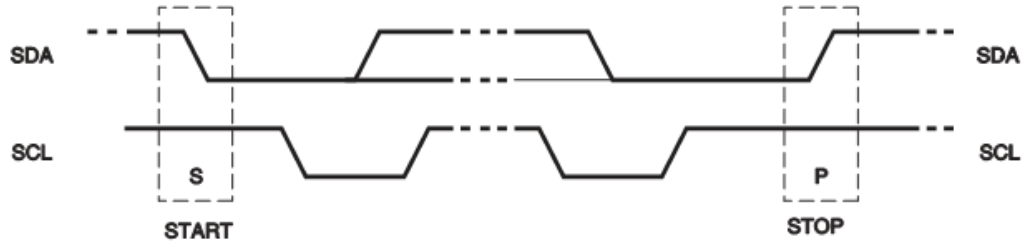
1.2.1 Komunikace mezi uzly sběrnice

Sběrnice I2C je typu multimaster. Každý uzel komunikační sítě může pracovat jako master i jako slave. To je umožněno tím, že každý uzel má vlastní sedmi nebo deseti bitovou adresu, která musí být v rámci komunikační sítě jedinečná. V klidovém stavu, kdy žádný uzel nevysílá, musí být oba vodiče na logické úrovni H. To je zajištěno zdvihacími rezistory připojenými k napájecímu napětí.

Následují popis komunikace je popsán při použití sedmibitové adresy.
Pro veškeré vysílání dat se dodržuje podmínka, že signál na vodiči SDA se smí měnit jen při logické úrovni L na vodiči SCL. To dává možnost přijímacím uzlům v případě pomalejšího

zpracování právě příchozího bitu podržet vodič SCL na logické úrovni L, a tím znemožnit vyslání dalšího bitu, dokud nebudou všechny stanice připraveny. Tato podmínka je porušena pouze při vysílání start bitu a stop bitu.

Vlastní komunikace je vždy zahájena vysláním tzv. start bitu. To se realizuje tak, že uzel master sníží napětí na vodiči SDA na logickou úroveň L, zatím co signál na vodiči SCL je ještě určitou dobu držen na logické úrovni H. Tato doba je dána frekvencí hodinového signálu SCL. Vše je názorně vidět na **Obrázku 4.:**

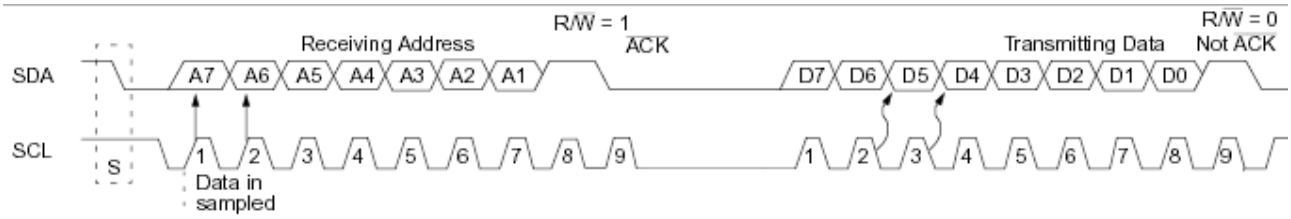


Obrázek 4.: *Průběhy signálů při vysílání start bitu a stop bitu. Převzato z [1]*

Ihned po vyslání start bitu, začne uzel master vysílat sedmibitou adresu uzlu, se kterým chce komunikovat. Osmý odeslaný bit pak značí, zda se bude z uzlu číst (logická 0) nebo do něj zapisovat (logická 1). Po odeslání celého bytu (sedmibitová adresa plus bite pro čtení nebo zápis) je na vodiči SDA přivedena logická úroveň H. Všechny uzly porovnají svoji adresu s adresou přijatou a v případě shody zvolený uzel v devátém časovém taktu vyšle signál ACK (acknowledge). Signál ACK se realizuje tak, že zvolený uzel přivede na vodič SDA logickou úroveň L. Tím vybraný uzel potvrdí, že je na sběrnici skutečně připojen a že je připraven přijmout nebo odeslat data.

Po přijetí ACK začne uzel master přijímat nebo vysílat data. Každý přenos bytu je potvrzován vysláním signálu ACK. Po přenosu všech dat je komunikace mezi uzly ukončena vysláním stop bitu. Stop bit je z hlediska změny úrovně signálu opakem start bitu, tj. při logické úrovni H na vodiči SCL je vodič SDA přiveden do logické úrovně H.

Celá výše popsaná komunikace je názorně vidět na **Obrázku 5.:**



Obrázek 5.: *Průběhy signálů při komunikaci na sběrnici I2C. Převzato z [2]*

1.2.2 Vlastnosti sběrnice I2C

Nejběžnější standardní přenosové rychlosti jsou 100kbps a 400kbps označované jako standard mode a fast mode. Větší standardní rychlosti 1Mbps a 3,4Mbps nejsou dnes ve větší míře podporovány.

Výhody sběrnice spočívají v tom, že každý uzel může ke sběrnici přistupovat jako master. To dělá sběrnici flexibilnější a rozšiřuje její použití. Další výhoda je v nutnosti použití pouze dvou signálových vodičů pro přenos dat mezi uzly a větší možná vzdálenost mezi uzly.
Nevýhodou sběrnice je složitější hardwarová implementace a softwarové ovládání, protože před samotnou komunikací musí dojít k vyslání a porovnání adresy uzlu slave, nastavení směru vysílání dat a po úspěšném přijetí dat pravidelně vysílat signál ACK, což již nelze jednoduše realizovat. Další nevýhodou jsou nižší přenosové rychlosti, které jsou oproti SPI na 10MHz skoro třetinové i při použití nejvyšší standardní rychlosti 3,4MHz.

2. Popis důležitých součástek zapojení

Základním prvkem celého rozhraní je převodník FT4232H od firmy FTDI. Ten zajišťuje veškerou komunikaci mezi počítačem a zbytkem rozhraní a celé rozhraní řídí. Pro zajištění potřebných funkcí rozhraní jsou použity další neméně důležité integrované obvody. Jelikož požadavkem na rozhraní byla minimálně dvoukanálová I2C sběrnice, je pro rozšíření jednoho I2C kanálu převodníku FT4232H použit multiplexer I2C, integrovaný obvod PCA9544A. Výstupní signály z převodníku FT4232H jsou na napěťové úrovni 3,3V. Protože rozhraní musí být schopné obsluhovat vnější sběrnice s napětím 3,3V i 5V, musí napěťová úroveň výstupních signálů z převodníku FT4232H převáděna na potřebnou napěťovou úroveň. O to se starají napěťové shiftery a budiče 74LVC8T245, které jsou použity pro buzení vstupů/výstupů a SPI sběrnice. Integrované obvody PCA9517D potom budí a převádí napětí pro jednotlivé kanály sběrnice I2C.

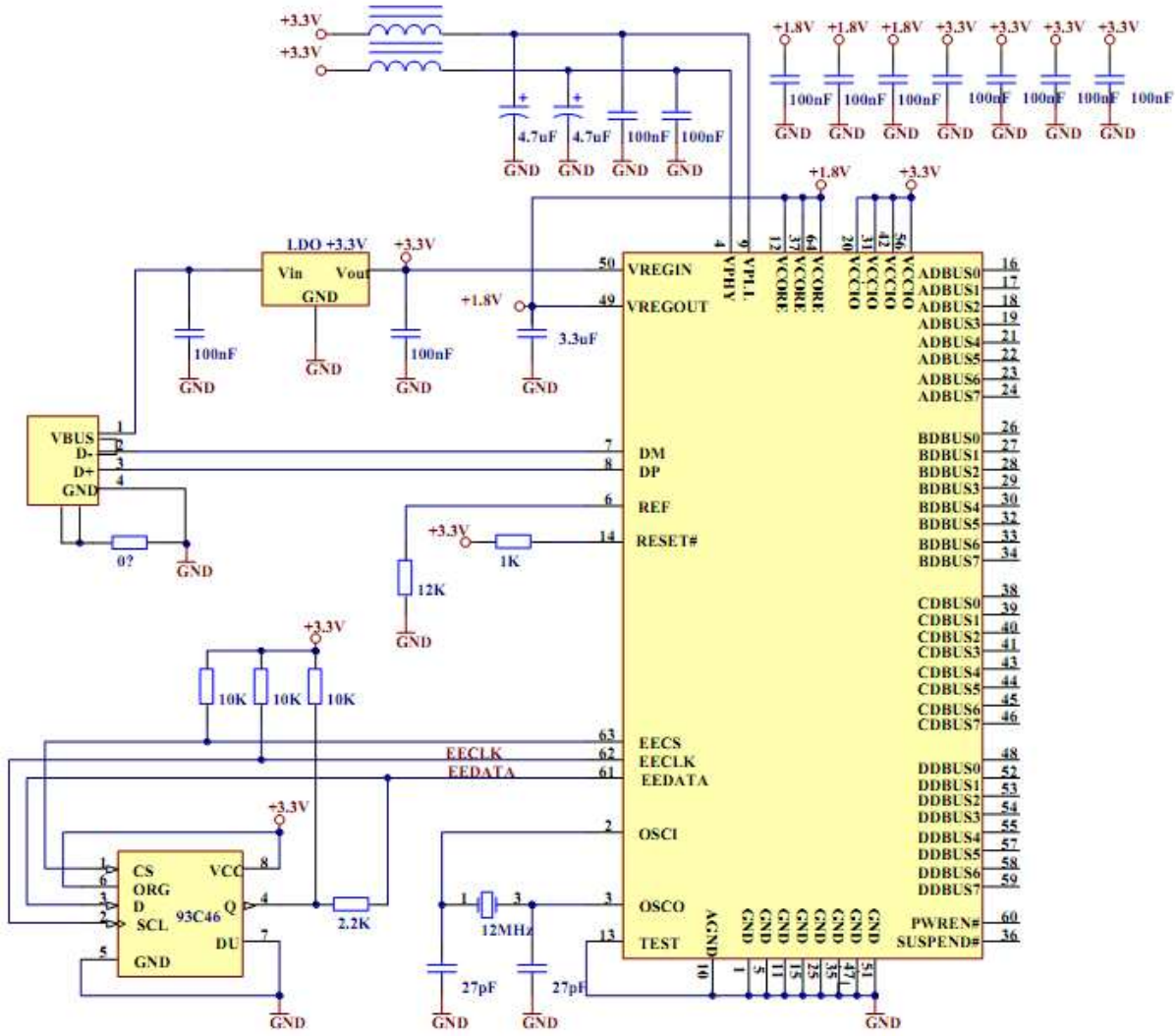
2.1 Převodník FT4232H

Integrovaný obvod FT4232H je univerzální vysokorychlostní převodník USB-UART/MPSSE IC kompatibilní s USB 2.0. Jedná se o čtyřkanálový převodník (jednotlivé kanály jsou označované písmeny A, B, C a D), kde porty A a B mohou využívat multi-protokolový synchronní sériový procesor (MPSSE). To umožňuje emulovat synchronní sériové sběrnice typu JTAG, SPI a I2C nebo jiné. Všechny porty také mohou pracovat jako nezávislé univerzální asynchronní rozhraní neboli UART. Tak je po přidání dalších vnějších obvodů možné získat až 4 sběrnice RS232, RS422 nebo RS485.

Všechny čtyři porty je také možné využívat v tzv. bit-bang režimu. V tomto režimu je možné nastavit každému pinu portu funkci digitálního vstupu nebo výstupu a stav těchto pinů jednoduše nastavit (v případě výstupů) nebo číst (v případě vstupů). Tím je možné softwarově implementovat vlastní libovolnou sběrnici. Bit-bang mód může pracovat v synchronním i asynchronním režimu, což tomuto převodníku dává ještě širší možnost využití.

2.1.1 Zapojení převodníku

Zapojení převodníku je vidět na **Obrázku 6.**



Obrázek 6.:*Zapojení převodníku FT4232H. Převzato z [3]*

Napájecí napětí převodníku je 3,3V. To je možné získat buď pomocí regulátoru z napájecího napětí USB, nebo externím napájením. Pro napájení mikroprocesorového jádra převodníku je potřeba napětí 1,8V. Toto napětí si převodník vyrábí sám zabudovaným regulátorem, který má vstup vyveden na pin 50. Napětí z regulátoru je dostupné na pinu 49 a musí být přivedeno na vstupní piny 12, 37 a 64.

Kladné napájení fyzické vrstvy portů je vyvedeno na piny 20, 31, 42 a 56. Pro správnou činnost je nutné, aby napájecí napětí bylo přivedeno na všechny čtyři piny. Fyzická vrstva převodníku je napájena přes vstupní piny 4 a 9. Je doporučeno použít předřazené tlumivky, protože při činnosti převodníku dochází k proudovým rázům, které vložené tlumivky eliminují.
Piny 1, 5, 11, 15, 25, 35, 47 a 51 slouží k přivedení nulového potenciálu. Pin číslo 13 se používá při testovacím režimu převodníku. Pro normální činnost musí být uzemněn.

Převodník také podporuje funkci ovládání snížené spotřeby. K tomuto účelu slouží pin 36, na pinu 60 se naopak nachází signál pro vybuzení externích obvodů k normální činnosti.

Jako každé USB zařízení musí mít i převodník vlastní deskriptory, které ho jednoznačně identifikují v operačním systému počítače. K uložení deskriptorů slouží vnější paměť. Paměť musí být 16 bitová sériová paměť. Doporučuje se použít paměť 93LC46B nebo podobnou, která může pracovat s napájecím napětím 3,3V. Paměť není pro činnost převodníku nutná. Pokud není zapojena, převodník se do operačního systému přihlašuje vlastními deskriptory, které jsou uložené v nepřepisovatelné paměti převodníku.

Paměť se programuje přímo přes převodník. K tomu slouží program FT Prog, který je na doprovodném CD ve složce FT4232H/FT_Prog_v2.6.6. V této složce je i manuál k tomuto programu pod názvem AN_124_User_Guide_For_FT_PROG.

2.1.2 Ovládání převodníku

Převodník je ovládán přes USB rozhraní. Po prvním připojení je nutné do počítače nainstalovat ovladače. Ovladače jsou dvojího druhu podle použití převodníku. Pokud bychom převodník chtěli používat jako převodník USB – RS232, nainstalujeme ovladače VCP. Potom se převodník z hlediska systému chová jako čtyři nezávislé sběrnice RS232. Pro naše použití nainstalujeme ovladače D2XX, které umožňují přímé ovládání převodníku. Ty jsou dostupné na stránkách firmy FTDI, která je výrobcem převodníku, pro všechny dnes používané operační systémy. Na doprovodné CD ve složce FT4232H/Ovladače jsou k dispozici ovladače pro operační systém Windows.

Na stránkách výrobce jsou také dostupné knihovny se všemi příkazy převodníku. Tyto knihovny je také možné nalézt na doprovodném CD ve složce FT4232H/Knihovny a unity

spolu s unitami pro Delphi. Unity obsahují několik užitečných procedur a funkcí, které lze dobře využít v ovládacím programu.

Jako hlavní knihovna, ve které jsou obsaženy všechny nejdůležitější příkazy převodníku, je knihovna FTD2XX.dll. Popis jednotlivých funkcí by byl příliš obsáhlý. Vše je možné najít v dokumentu D2XX_Programmer's_Guide(FT_000071) na doprovodném CD ve složce FT4232H/Manuály. To samé platí pro ostatní knihovny. Funkce, které jsou použité při ovládání rozhraní, jsou popsány v Odstavci 4.3.2 i s příklady použití v ovládací aplikaci rozhraní.

2.2 Multiplexer PCA9544A

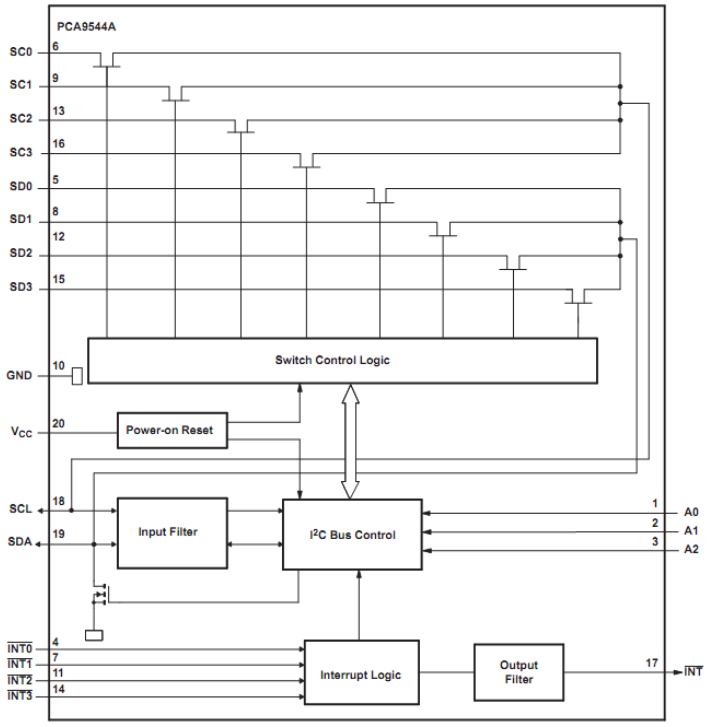
Integrovaný obvod PCA9544A je obousměrný čtyřkanálový I2C multiplexer od firmy Texas Instruments. Každý kanál má navíc vstup INT, který dokáže obsluhovat přerušení od zařízení daného kanálu, pokud připojené zařízení tuto funkci podporuje.

Maximální přenášená frekvence může být 400kHz, multiplexer tedy není možné použít pro větší standardní přenášené frekvence 1MHz a 3,4MHz.

Vnitřní schéma multiplexeru je vidět na **Obrázku 7**.

Vývody Vcc a GND slouží k napájení multiplexeru. Napájení se může pohybovat v rozmezí 2,5V až 5,5V. Na vývody SCL a SDA se připojuje vstup sběrnice I2C, z vývodů SC0 až SC3 a SD0 až SD3 pak vystupují jednotlivé kanály sběrnice I2C. Vývody INT0 až INT3 jsou vstupy přerušení všech čtyř kanálů a na výstup INT je pak přerušovací signál samotného multiplexeru.

Pomocí vývodů A0 až A2 se nastavují poslední 3 bity adresy multiplexeru pro identifikaci na sběrnici I2C (viz Odstavec 1.2.1). První 4 bity adresy jsou pevně dané. Celá adresa má hodnotu podle **Obrázku 8**. V zapojení rozhraní jsou vývody uzemněny, tudíž adresa multiplexeru v tomto konkrétním zapojení je 1110000.



Obrázek 7.: *Vnitřní schéma multiplexeru PCA 9544A. Převzato z [4]*

Pro přepínání výstupního kanálu, čtení nastavení a stavů vstupů přerušení slouží kontrolní registr multiplexeru. Struktura kontrolního registru je vidět na **Obrázku 9**.

V posledních 4 bitech kontrolního registru jsou zaznamenány stavy přerušujících vstupů. První 3 bity slouží k výběru výstupního kanálu. Výběr kanálu a čtení stavu, který kanál je právě vybrán, se řídí podle **Tabulky 1**. Čtení stavů přerušujících vstupů je vidět v **Tabulce 2**.

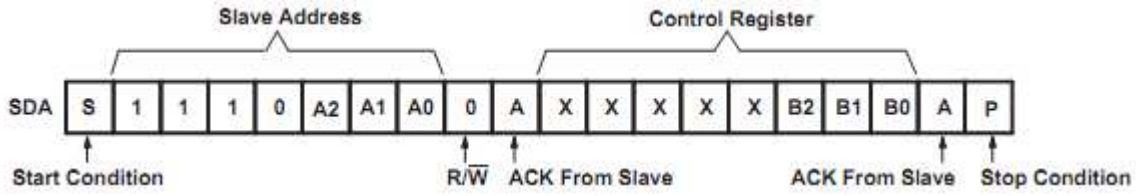
$\overline{INT3}$	$\overline{INT2}$	$\overline{INT1}$	$\overline{INT0}$	D3	B2	B1	B0	COMMAND
X	X	X	X	X	0	X	X	No channel selected
X	X	X	X	X	1	0	0	Channel 0 enabled
X	X	X	X	X	1	0	1	Channel 1 enabled
X	X	X	X	X	1	1	0	Channel 2 enabled
X	X	X	X	X	1	1	1	Channel 3 enabled
0	0	0	0	0	0	0	0	No channel selected, power-up default state

Tabulka 1.: *Hodnoty kontrolního registru při výběru jednotlivých kanálů multiplexeru PCA9544A. Převzato z [4]*

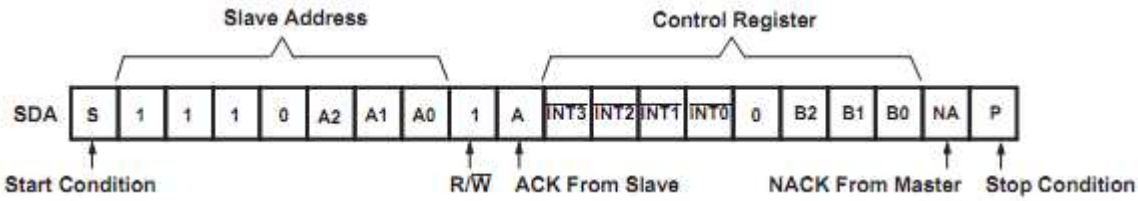
$\overline{INT3}$	$\overline{INT2}$	$\overline{INT1}$	$\overline{INT0}$	D3	B2	B1	B0	COMMAND
X	X	X	0	X	X	X	X	No interrupt on channel 0
			1					Interrupt on channel 0
X	X	0	X	X	X	X	X	No interrupt on channel 1
		1						Interrupt on channel 1
X	0	X	X	X	X	X	X	No interrupt on channel 2
	1							Interrupt on channel 2
0	X	X	X	X	X	X	X	No interrupt on channel 3
1								Interrupt on channel 3

Tabulka 2.: *Hodnoty kontrolního registru při aktivaci vstupu přerušení jednotlivých kanálů multiplexeru PCA9544A. Převzato z [4]*

Struktura příkazu pro zápis do registru multiplexeru je vidět na **Obrázku 8.**, příkaz pro čtení z registru je vidět na **Obrázku 9**.



Obrázek 8.: *Struktura příkazu pro zápis do kontrolního registru multiplexeru. Převzato z [4]*



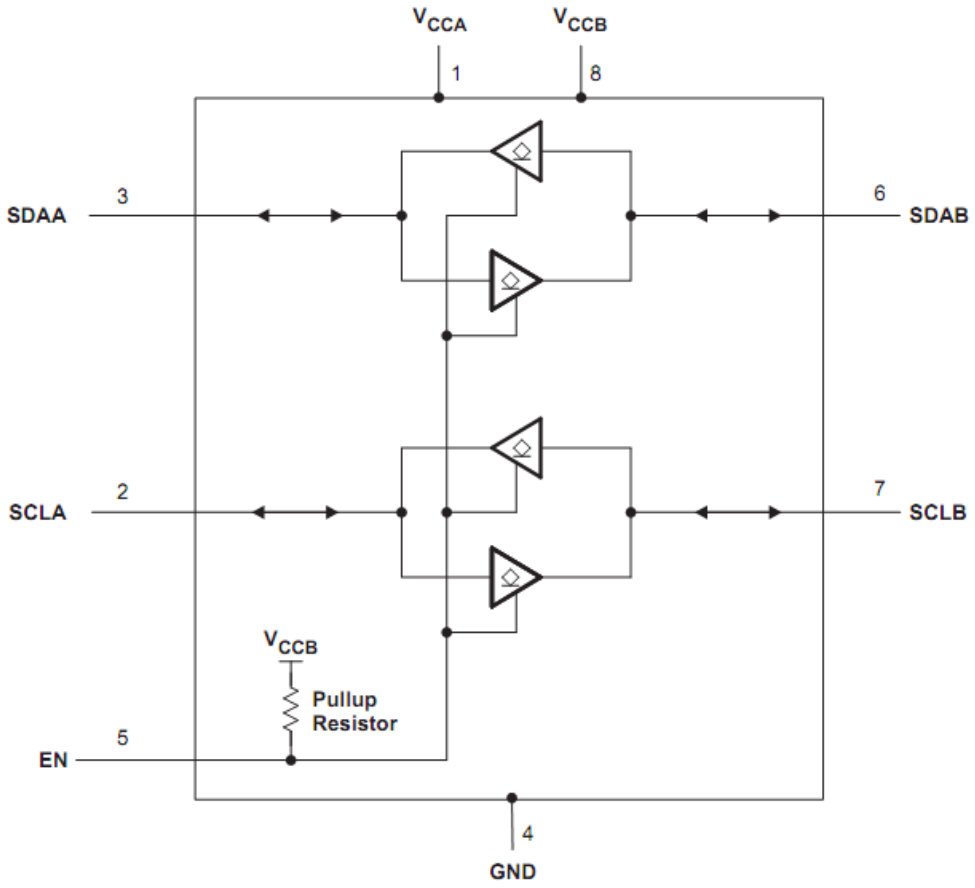
Obrázek 9.: *Struktura příkazu pro čtení z kontrolního registru multiplexeru. Převzato z [4]*

Datasheet multiplexeru je na přiloženém CD ve složce Datasheet pod názvem PCA9544A.

2.3 Napěťový shifter PCA9517

Napěťový shifter PCA9517 od firmy Texas Instruments byl vyvinut přímo pro práci se sběrnicí I2C. Kromě změny napěťové úrovně procházejícího signálu funguje jako třístavový budič. Maximální přenášená frekvence je, jako u multiplexeru, 400kHz a napájecí napětí 2,5V až 5,5V.

Vnitřní schéma je vidět na **Obrázku 10**.



Obrázek 10.: *Vnitřní schéma shifteru PCA 9517. Převzato z [5]*

Vývod V_{CCA} slouží jako jedna napěťová reference. Signál připojený na vývody SDAA a SCLA by měl mít stejnou napěťovou úroveň jako napětí na V_{CCA}. Na vývod V_{CCB} se připojuje druhá napěťová reference. I zde platí stejné pravidlo jako pro stranu A, že signály na vývodech SDAB a SCLB by měly mít stejnou napěťovou úroveň jako napětí na V_{CCB}. Nulový potenciál se zapojuje na vývod GND.

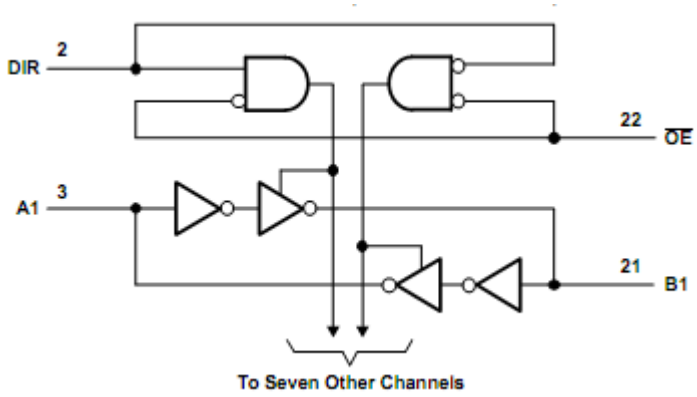
Ovládání třetího tedy izolujícího stavu se provádí vstupem označeným EN. Pro přechod do třetího stavu je nutné na tento vstup přivést logickou úroveň L.

Datasheet shifteru je na přiloženém CD ve složce Datasheet pod názvem PCA9517.

2.4 Napěťový shifter 74LVC8T245

Napěťový shifter 74LVC8T245 má osm jednosměrných přepínatelných linek. Stejně jako PCA9517 kromě napěťové změny úrovně procházejících signálů funguje jako třístavový budič. Napájecí napětí je v rozmezí 1,65V až 5,5V.

Vnitřní schéma jedné linky je na **Obrázku 11**.



Obrázek 10.: *Vnitřní schéma jedné linky shifteru 74LVC8T245. Převzato z [6]*

Referenční napětí pro stranu A se přivádí na vývod V_{CCA}, pro stranu B na V_{CCB}.

Shifter umožňuje obousměrný přenos signálu. Směr přenosu se řídí vstupem DIR. Při logické úrovni H na tomto vstupu je směr přenosu z A do B. Opačná logická úroveň změní i směr přenosu tedy z B do A.

Vstup OE ovládá třetí stav budiče. Pro normální práci budiče musí být na vstup přivedena logická úroveň L. Je doporučeno, aby vstupy DIR a OE byly ovládány logickými napěťovými úrovněmi jako má strana A.

Datasheet shifteru je na přiloženém CD ve složce Datasheet pod názvem SN74LVC8T245.

3. Schéma zapojení

3.1 Orientace ve schématu

Schéma je rozděleno na dva listy. Na prvním je možné nalézt všechny hlavní funkční celky (viz. **Obrázek 11**.), na druhém listě se pak nachází ochranné prvky vstupních a výstupních pinů (viz. **Obrázek 12**.). Vzhledem k tomu, že ochranné prvky nemají na funkci rozhraní vliv, jsou z důvodu přehlednosti zakresleny na samostatném listě. Tato část zapojení je v hlavním schématu naznačena jako obdélník v pravé části schématu, do kterého zprava vstupují konektory X2 a X3 a zleva jsou k němu přivedeny jednotlivé spoje. Pro jednoznačnou identifikaci spojů při přechodu mezi listy schématu, má každý spoj vlastní označení, kde je dodrženo, že spoje vstupující do druhého listu mají označení s indexem *a*, spoje ze schématu vystupující mají index *b* (u propojení nulového potenciálu se vyskytuje ještě index *c*).

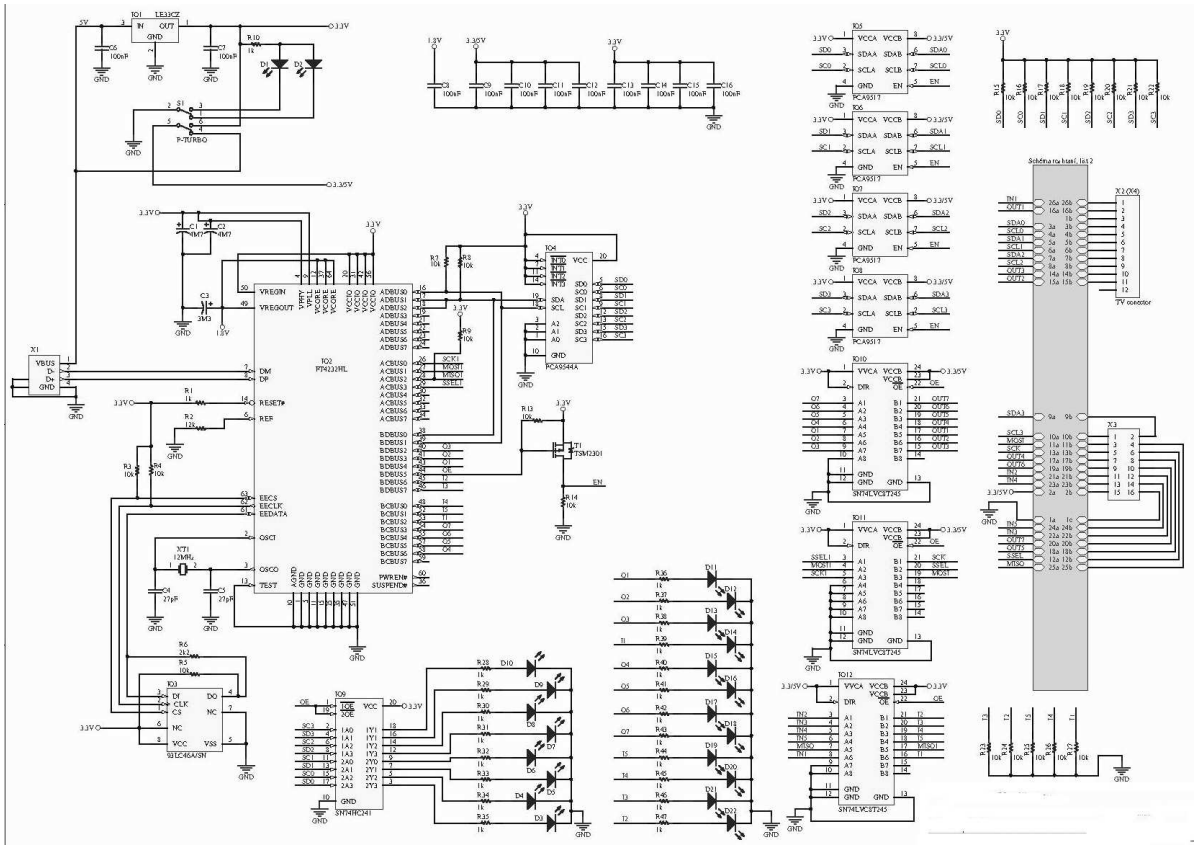
Většina spojů ve schématu je také provedena pomocí odkazů. To pomáhá lepší orientaci ve schématu a zlepšuje přehlednost. Jako příklad lze třeba uvést propojení spoje s názvem EN. Tento spoj vystupuje ze spoje mezi tranzistorem T4 a rezistorem R14 a je spojen se všemi místy ve schématu, které mají také označení EN, tj. s piny 5 integrovaných obvodů IO5 až IO8.

Značení jednotlivých součástek schématu je následující:

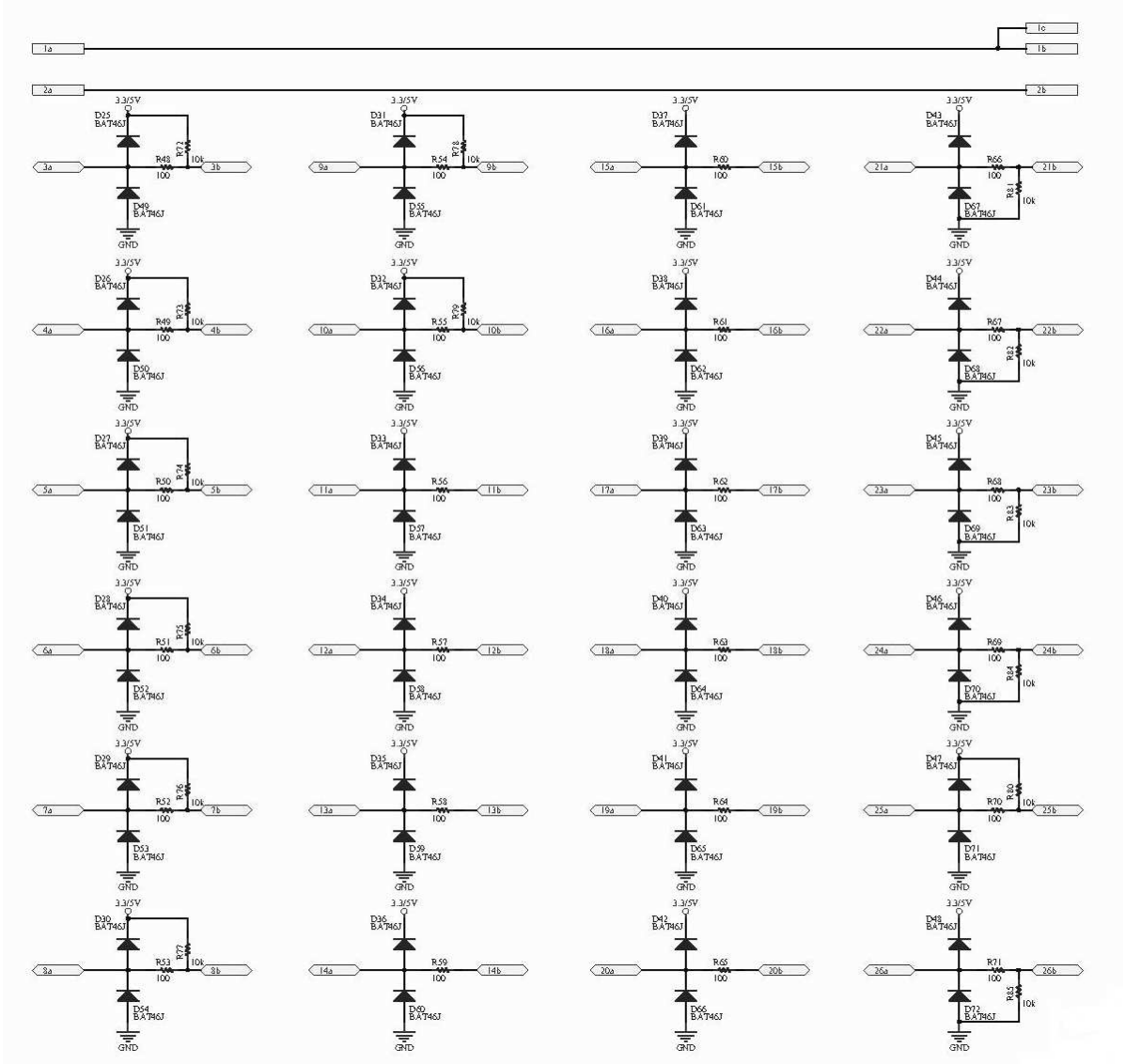
C kondenzátory
D diody
IO integrované obvody
R rezistory
S mechanický spínač
T tranzistory
X konektory
XT krystal

3.2 Popis schématu zapojení

Schéma zapojení je vidět na **Obrázku X**. a **Obrázku X**. Schéma je také v příloze A diplomové práce a na přiloženém CD. Zde je uložen celý projekt návrhu v Altium (složka Projekt/Projekt Altuim Designer), ve kterém je možné schéma editovat, nebo je zde schéma ve formátu PDF ve složce Projekt pod názvem Schéma list 1 a Schéma list 2. Hodnoty všech součástek jsou zapsány buď přímo ve schématu, nebo v seznamu součástek, který je uveden v příloze B diplomové práce a také na přiloženém CD ve složce Projekt pod názvem Seznam součástek.



Obrázek 11.: *Schéma zapojení – hlavní část*



Obrázek 12.: *Schéma zapojení – ochrany vstupů proti přepětí, vstupní zdvihací rezistory*

Schéma zapojení vychází z požadované funkce rozhraní a doporučeného zapojení jednotlivých prvků.

3.2.1 Napájení

Napájení pro celé rozhraní je získáváno přes konektor X1 přímo z USB sběrnice, která pracuje na napětí 5V. Napětí 3,3V, kterým se celé rozhraní napájí, je dosaženo použitím integrovaného stabilizátoru napětí LE33CZ (ve schématu značeném jako IO1). Pro správnou funkci stabilizátoru je nutné připojit blokovací kondenzátory 100nF (C6, C7) a filtrační

kondenzátor o velikosti 10µF. To je zde vyřešeno dvěma kondenzátory 4,7µF (C1, C2), každý pro jednu větev fyzické vrstvy převodníku FT4232H (IO2).

Jádro převodníku, jak již bylo popsáno v odstavci 2.1.1, potřebuje pro svou činnost napětí 1,8V. Toto napětí se získává přímo z integrovaného stabilizátoru převodníku. Napětí je filtrováno kondenzátorem o velikosti 3,3µF (C3) a rozvedeno na příslušné piny převodníku.

Výstupní napětí rozhraní je možné přepínat mezi 3,3V a 5V. K tomu slouží mechanický přepínač S1, na jehož přepínací kontakt je přivedeno napětí 3,3V a 5V. Ze společného kontaktu je pak získáváno napětí 3,3V nebo 5V podle polohy přepínače. Výstupní napětí je pro zamezení omylu signalizováno zelenými nízkoodběrovými světelnými diodami (D1, D2). Dioda D1 signalizuje výstupní napětí 3,3V, dioda D2 pak 5V. Napětí pro diody je sníženo předřadným rezistorem R10 o hodnotě 1kΩ. Signalizační diody jsou přepínány druhým kontaktem mechanického přepínače S1.

Napěťové špičky v napájecím napětí, které vznikají při činnosti integrovaných obvodů, jsou eliminovány blokovacími kondenzátory 100nF (C8 – C16), které jsou rozmístěné po celé desce plošných spojů.

Podle doporučeného zapojení převodníku FT4232H by měly být v napájení fyzické vrstvy zařazeny tlumivky. Tlumivky v zapojení zkušebního vzorku byly zapojeny, ale následným měřením osciloskopem byly na rozvodu napájecího napětí 3,3V zjištěny kmity, které znemožňovaly práci celého rozhraní. Jako původce kmitů byly zjištěny právě tyto tlumivky, a proto byly z finálního návrhu odstraněny.

3.2.2 Součástky kolem převodníku FT4232H

Signál z USB sběrnice je přiveden z konektoru X1 přímo do převodníku. Přes rezistor R1 je zapojen resetovací vstup převodníku. Převodník se resetuje uzemněním tohoto vstupu, proto pro normální činnost musí být zapojen na kladné napájecí napětí. Rezistor R2 2kΩ slouží k nastavení proudové reference a pro správnou činnost musí být zapojen proti zemi. Rezistory R3 až R6 jsou zdvihací a pomocné rezistory sběrnice, po které převodník komunikuje s externí pamětí 93LC46B (IO3).

Taktovací signál pro vlastní převodník je získáván z připojeného krystalu XT1, který pracuje na frekvenci 12MHz. Ke krystalu jsou pro omezení šíření kmitů zapojeny blokovací kondenzátory 27pF (C4, C5).

3.2.3 Zapojení sběrnice I2C

Zapojení je navrženo tak, že jako sběrnice I2C pracuje port A převodníku FT4232H. Rezistory R7 a R8 (hodnota 10kΩ) jsou zdvihací rezistory vodičů sběrnice I2C, které jsou pro správnou funkci sběrnice I2C nezbytné (viz Odstavec 1.2.1).

Sběrnice je dále vedena do integrovaného obvodu PCA9544A (IO4), což je kanálový multiplexer a rozdělí sběrnici do čtyř kanálů. Signály jednotlivých kanálů sběrnice poté prochází napěťovými shiftery PCA9517 (IO5 až IO8), které obousměrně mění napěťovou hodnotu signálu na úrovně přivedené na piny 1 a 8. Jelikož shiftery pracují současně jako třístavové budiče, je možné vodiče sběrnice I2C více proudově zatížit, aniž by došlo k poškození samotného multiplexeru. Z napěťových shifterů jsou kanály sběrnice vedeny na konektory X2 a X3. Připojení kanálů sběrnice na oba konektory je podle **Tabulky 3**. a **Tabulky 4**.

Rezistory R15 až R22 a R72 až R80 o hodnotě 10kΩ zapojené na jednotlivých kanálech I2C před i za napěťovými shiftery potom plní stejnou funkci jako rezistory R7 a R8.

K signalizaci činnosti kanálů sběrnice jsou použity žluté nízkoodběrové signalizační LED diody. Přidělený LED diod k jednotlivým vodičům kanálů sběrnice je uvedeno v **Tabulce 3**. a **Tabulce 4**. Aby nedocházelo k ovlivňování činnosti sběrnice, jsou LED diody zapojeny přes budič 74HC241. Rezistory R28 až R35 s hodnotou 1kΩ jsou předřazené odpory diod D3 až D10.

Protože převodník FT4232H může pracovat na sběrnici I2C pouze jako uzel master (vysvětlení viz Odstavec 1.2.1), jsou pro sledování dění na sběrnici oba vodiče sběrnice zavedeny do portu C převodníku, kde piny 38 a 39 jsou nakonfigurovány jako vstupy. Tím je možné softwarovým snímáním stavů těchto vstupů a následným správným zpracováním, získat data posílána jiným uzlem master na sběrnici.

3.2.4 Zapojení sběrnice SPI

Jako sběrnice SPI je nakonfigurován port B převodníku FT4232H. Z toho signály jednotlivých vodičů opět procházejí přes napěťové shiftery, ale tentokráte typu 74LVC8T245 (IO11, IO12). Jelikož tyto shiftery nejsou obousměrné, tak všechny výstupní signály sběrnice SPI (SCK, MOSI a SSEL viz Odstavec 1.1) jsou vedeny přes shifter IO11 a vstupní signál (MISO) přes shifter IO12. Odtud jsou jednotlivé vodiče sběrnice vyvedeny na konektor X3. Připojení vodičů ke konektoru je podle **Tabulky 4**.

Rezistor R9 (10kΩ) plní funkci zdvihacího rezistoru, aby vstup MISO měl definovanou napěťovou úroveň.

3.2.5 Obvod ovládání shifterů

Všechny použité shiftery jsou třístavové. Třetí tedy izolující stav je ovládám pomocí pinu 44 převodníku FT4232H. Shiftery IO9 až IO12 potřebují ke své činnosti logickou hodnotu L na příslušných pinech. Tato hodnota je brána přímo z pinu 44 převodníku. Shiftery IO5 až IO8 naopak pracují s úrovní opačnou tj. s logickou úrovní H. Aby bylo možné všechny shiftery ovládat jedním společným výstupem převodníku, je signál pro shiftery IO5 až IO8 negován připojeným tranzistorem T1. Při logické úrovni H na pinu 44 převodníku je tranzistor T1 uzavřen a přes rezistor R14 (10kΩ)je na shiftery IO5 až IO8 přiváděna logická úroveň L, čímž je navozen třetí izolující stav. Shiftery IO9 až IO12 jsou uzavřené též, protože se na ně dostává logická úroveň H přímo z pinu 44 převodníku.

Pro otevření shifterů je nutná logická úroveň L na pinu 44 převodníku. Tím dojde jednak k sepnutí shifterů IO9 až IO12 a jednak k sepnutí tranzistoru T1, přes který se pak dostane logická úroveň H na shiftery IO5 až IO8. Rezistor R13 (10kΩ)zapojení mezi vývody drive a source tranzistoru T1 zajišťuje spolehlivé uzavření tranzistoru při logické úrovni H na vývodu drive.

3.2.6 Obvod digitálních vstupů a výstupů

Digitální vstupy a výstupy rozhraní jsou ovládány přímo převodníkem FT4232H. Celkem má rozhraní 5 vstupů a 7 výstupů. Ty jsou vyvedeny na porty C a D převodníku. Odtud jdou signály opět přes shiftery 74LVC8T245 (IO10 a IO12), které zajišťují převod signálů na požadované napěťové úrovně. Shiftery jsou jednosměrné, proto přes shifter IO10 jsou vedeny

výstupy, přes shifter IO12 potom vstupy. Všechny vstupy a výstupy jsou vyvedeny na konektory X2 a X3 podle **Tabulky 3**. a **Tabulky 4**.

Stav digitálních vstupů a výstupů je signalizován červenými nízkoodběrovými LED diodami D11 až D22, které jsou k vodičům připojeny přímo bez použití budiče. Přidělení LED diod k jednotlivým vstupům a výstupům je vidět v **Tabulce 3**. a **Tabulce 4**. Předřazené rezistory k těmto diodám mají označení R36 až R47 a hodnotu 1kΩ.

Aby vstupy měly definovaný stav i při nezapojeném signálu, jsou ke vstupům připojeny zdvihací rezistory R81 až R85 (10kΩ) připojených na vodiče z konektorů X2 a X3. Protože při přechodu shifterů do třetího stavu by byly vstupy opět bez definované úrovně, jsou rezistory o stejné hodnotě zapojeny i na vodiče mezi převodníkem a shifterem. Tyto rezistory mají označení R23 až R27.

Signál	Pin konektoru	Označení LED diody
Dig. vstup 1	1	D14
Dig. výstup 1	2	D11
Nulový potenciál	3	
I2C - SDA0	4	D3
I2C - SCL0	5	D4
I2C - SDA1	6	D5
I2C - SCL1	7	D6
I2C - SDA2	8	D7
I2C - SCL2	9	D8
Dig. výstup 3	10	D13
Dig. výstup 2	11	D12
	12	

Tabulka 3.: *Zapojení pinů konektoru X2 a přidělení LED diod signálům*

Signál	Pin konektoru	Označení LED diody
I2C - SCL3	1	D10
I2C - SDA3	2	D9
SPI - MOSI	3	
SPI - MISO	4	
SPI - SCK	5	
SPI - SSEL	6	
Dig. výstup 4	7	D15
Dig. výstup 5	8	D16
Dig. výstup 6	9	D17
Dig. výstup 7	10	D18
Dig. vstup 2	11	D22
Dig. vstup 3	12	D21
Dig. vstup 4	13	D20
Dig. vstup 5	14	D19
3,3/5V	15	
Nulový potenciál	16	

Tabulka 4.: *Zapojení pinů konektoru X3 a přidělení LED diod signálům*

3.2.7 Přepěťová ochrana vstupů rozhraní

Při připojování a odpojování konektorů X2 a X3 může dojít k napěťovým přechodovým jevům, které by mohly poškodit shiftery rozhraní přepětím. Aby se přepětí nedostalo na piny shifterů, jsou u každého vstupu a výstupy zapojeny dvě shottkyho diody D25 až D72, jedna proti kladnému a druhá proti zápornému potenciálu. Při přepětí (kladném či záporném) dojde k otevření schottkyho diody BAT-46J a k odvedení přepětí k opačnému potenciálu než je potenciál přepětí. Při otevření diody dochází k přímému spojení opačných potenciálů tj. zkratu, který je doprovázen velkým proudem. Velký proud by mohl poničit schottkyho diody,

proto je před diodami zapojen předřazený rezistor, který procházející proud omezuje. Omezovací rezistory mají označení R49 až R71 a hodnotu 100Ω.

4. Realizace rozhraní

4.1 Návrh

Návrh vycházel z požadavků na funkci rozhraní. Jako hlavní prvek byl zvolen převodník FT4232H, který dokáže realizovat všechny potřebné funkce, které jsou na rozhraní kladeny. Ostatní integrované obvody a součástky již plní pouze podpůrnou funkci jako ochranu proti přepětí, převod napěťové úrovně signálů na požadovanou úroveň, buzení signálů a signalizaci stavů pomocí LED diod. Ale i tak součástky musely být vybrány tak, aby zapojení vyhovovaly. Jeden ze základních parametrů byl rozsah napájecího napětí. Dále bylo nutné součástky vybírat podle dovolené přenosové rychlosti, která musela odpovídat předpokládaným použitým rychlostem.

Návrh začal nakreslením schématu. Schéma je nakresleno ve vývojovém softwaru pro elektroniku Altium Designer. Jedná o profesionální návrhářský program s mnoha funkcemi. Jednotlivé prvky schématu jsou k dispozici ve velmi rozsáhlé knihovně. Navíc je možné si od vývojářů programu Altium stáhnout a nainstalovat aktualizované knihovny, které obsahují i nejnovější elektronické prvky. Pokud přesto nějaký prvek chybí, je možné si jej lehce vytvořit v integrovaném editoru.

Po nakreslení schématu byly všechny součástky exportovány do editoru pro návrh desky plošných spojů. Zde byl pro navržení parametrů plošného spoje využit skvělý průvodce. V jednotlivých krocích lze nastavit rozměr desky, počet vrstev spojů, tloušťka spojů a převažující druh součástek podle druhu montáže.
Nejprve byly stanoveny rozměry desky. Na rozhraní byl dán požadavek co nejmenšího rozměru, aby bylo snadno přenositelné. Jako nejmenší možný rozměr se po zkušebním rychlém rozmístění součástek ukázal rozměr minimálně 100x60mm. Na základě tohoto kroku byla vybrána nejbližší větší krabička, do které bude rozhraní umístěno. Krabička U-KP24A se

ukázala jako vhodná nejen kvůli rozměrům, ale také kvůli přítomnosti uchycovacích sloupků. Deska podle náčrtku od výrobce krabičky (náčrtek je k nalezení na přiloženém CD ve složce Datasheet pod názvem U-KP24A) mohla mít maximální rozměry 104x63mm. Rozměr plošného spoje byl tedy stanoven na 102x62mm. Krabička nebyla v požadavcích na rozhraní vyžadována, ale pro lepší mechanickou ochranu je řešení s krabičkou lepší. Nic méně nic nebrání tomu, aby deska místo do krabičky nebyla jen podložena distančními sloupky. Distanční sloupky se v tomto případě přišroubují v místech, kde by byla deska uchycena do krabičky.

Počet vrstev byl stanoven kvůli rozměrům desky na dvě. Vzhledem k pouzdrům integrovaných obvodů (obzvláště převodníku FT4232H) byla vybrána tloušťka spojů 0,3mm.

Před rozmísťováním jednotlivých součástek bylo nutné ještě na desku zanést uchycovací otvory a otvory pro průchod sloupků krabičky, které slouží k sešroubování krabičky dohromady.

Po vyřešení mechanických otvorů desky nadešlo rozmísťování součástek na desku tak, aby spojovací cesty vyšly co nejlépe. Jako nejlepší se ukázalo umístit LED diody po stranách desky, kde nikde nepřekážely a převodník přibližně do středu desky.

Jednotlivé spoje se v Altium kreslí velice snadno, program sám hlídá izolační vzdálenosti mezi spoji a nedovolí propojit součástky jinak, než jak je zakresleno ve schématu. Je také samozřejmě možné nakreslit plošný spoj bez předchozího nakreslení schématu, ale pak chybí zpětná kontrola správného propojení.

Spoje, které nebylo možné zakreslit na stranu SMD součástek, byly rozvedeny na druhé straně desky. Jedním z dalších požadavků na rozhraní byla DPS bez prokovených otvorů. Proto jsou na desce k propojení obou stran použity drátové propojky. Při návrhu byla věnována pozornost tomu, aby spoje sběrnic neprocházely drátovými propojkami, které by do nich mohly zanést přechodové odpory a hlavně kapacity, které by signál deformovaly. Skrz propojky byly tedy vedeny jen spoje napájení a digitálních vstupů a výstupů, u kterých parametry spojů nejsou kritické.

Návrh DPS byl ukončen exportováním výrobních podkladů DPS a vytvořením integrované knihovny projektu, která obsahuje všechny prvky použité v návrhu. Odpadá tak nutnost, aby při otvírání projektu na jiném počítači byly nainstalovány stejné knihovny prvků.

Vyexportované podklady pro výrobu obsahují nákresy spojů, rozmístění součástek, nepájivou masku a výstupní soubor pro vrtačku, kde jsou zaznamenány polohy a rozměry děr v desce.

Celý návrh (schémat i DPS) je na přiloženém CD ve složce Projekt/Projekt Altuim. Ve složce Projekt je pak možné nalézt vyexportované soubory ve formátu PDF.

4.2 Výroba rozhraní

4.2.1 Výroba desky plošných spojů

Deska plošných spojů byla pro zkušební vzorek rozhraní vyrobena v domácích podmínkách formou fotocesty. Byla zakoupena deska s oboustranně naplátovanou mědí a vrstvou fotorezistu. Předlohy pro spoje byly získány z projektu Altia. Před ozařováním desky ultrafialovým světlem bylo nutné vyřešit, jak umístit předlohy z obou stran tak, aby nebyly vzájemně posunuté. Nakonec se ukázalo jako nejlepší řešení vyvrtat do desky malým vrtákem (0,8mm) otvory ve středu budoucích uchycovacích otvorů. Tyto otvory pak sloužily jako záchytné body pro správné usazení předloh pro obě strany. Při osazování předloh bylo také nutné dbát na správnou orientaci předlohy, a to nejen podélnou, ale také stranovou, aby výsledné spoje nebyly zrcadlově otočeny.

Pro osvětlení desky ultrafialovým světlem byly použity aktinické ultrafialové zářivky, čímž bylo dosaženo rovnoměrného osvětlení celé desky oproti použití bodového zdroje ultrafialového světla.

Po osvětlení byla deska vyvolána v hydroxidu sodném a vyleptána v chloridu železitém. Následovala kontrola na celistvost spojů a zkraty mezi spoji. Jelikož mezi některými spoji zůstala tenká neodleptaná místa, musela být tato místa mechanicky rozdělena. Deska byla nakonec ošetřena pájitelným lakem, který spoje chrání proti korozi.

Nakonec došlo k vyvrtání všech děr. Pro vrtání děr pro součástky byl použit vrták o průměru 0,8mm, uchycovací otvory jsou vyvrtány vrtákem o průměru 3mm a otvory pro sloupky krabičky mají průměr 7mm.

4.2.2 Osazování součástek

Součástky byly pájeny odporovou pájkou s hrotem průměru 0,3mm. Postup pájení integrovaných obvodů vycházel ze zkušenosti. Pájené plošky byly potřeny pájecí kapalinou a lehce pocínovány. Pájecí kapalina je složena z kalafuny rozpuštěné v ethanolu a zabraňuje při pájení propojení pájecích plošek cínem mezi sebou. Integrovaný obvod byl usazen na správné místo a na několika místech lehce připájen, aby nedošlo k jeho posunutí. Na piny byla nanesena pájecí kapalina a pomalým posunem hrotu pájky po jednotlivých pinech byly piny přiletovány.

Po zaletování byla provedena u všech součástek kontrola na zkraty (zvláště u integrovaných obvodů) a dobré elektrické a mechanické spojení součástky se spojem.

4.2.3 Oživení

Při prvním připojení rozhraní k počítači bylo změřeno digitálním multimetrem napětí na napájecích pinech všech integrovaných obvodů. Oproti očekávanému napětí 3,3V z integrovaného stabilizátoru byly naměřeny pouze 3V. Za příčinu bylo později za pomoci osciloskopu diagnostikováno použití tlumivek 100µH zapojených v napájení fyzické vrstvy převodníky. Tyto tlumivky tvořily s filtračními kondenzátory oscilátor, který rozkmital celé napájení. Po vyřazení tlumivek bylo napětí podle osciloskopu hladké se správnou hodnou napětí. Tlumivky byly poté vyřazeny i z návrhu rozhraní, tudíž při konstrukci dalších kopií s nimi není počítáno. Také bylo vyhodnoceno, že napájecí spoje by měly být silnější, proto byl návrh upraven a napájecí spoje rozšířeny na 0,6mm.

Dále bylo zkontrolováno napětí 1,8V vycházející z integrovaného stabilizátoru převodníku a bylo vyzkoušeno přepínání výstupního napětí 3,3/5V. Zde již bylo vše v pořádku.

Po zprovoznění zkušební aplikace ovládání rozhraní bylo dále přezkoušeno, zda vstupy a výstupy reagují na jednotlivé signály a byla přezkoušena funkce sběrnice I2C a SPI. Sběrnice byly zkoušeny na pamětech 24C01 (u I2C) a 93C46 (SPI). U sběrnice I2C bylo zjištěno, že při návrhu schématu byly u připojení integrovaného obvodu PCA9544A prohozeny linky SCL a SDA. Bylo to způsobeno chybou v knihovně Altia, kde u popisu pinů této součástky byly prohozeny popisy. Proto byly tyto vodiče na DPS přerušeny a pomocí tenkých vodičů signály přehozeny.

Na zkušebním vzorku bylo při hledání závad nutné některé spoje přerušit. Tyto spoje byly následně opět spojeny drátovými propojkami.

Nakonec byla odzkoušena funkce digitálních vstupů a výstupů a ovládání třetího stavu shifterů.

4.2.4 Krabička

Jak již bylo psáno výše, použití krabičky není nutné.

Krabička U-KP24A byla upravena podle náčrtku Úprava krabičky, který je na přiloženém CD ve složce Projekt. Byly vyvrtány otvory pro LED diody, otvor pro aretační tlačítko S1 a vypilovány otvory pro konektor X1 a X3. S konektorem pro připojení TV (konektor X2) bylo původně počítáno přímo do DPS. Ale protože konektor by byl v krabičce hodně zapuštěný a byl by k němu špatný přístup, byl nakonec přilepen na víčko krabičky a s deskou spojen přes pomocný konektor X4. Protože krabička nemusí být použita a tudíž konektor X2 by byl zapájen přímo do DPS, není pomocný konektor X4 ve schématu přímo zakreslen.

Deska je ke krabičce přišroubována k uchycovacím nálitkům. Aby deska byla ve správné výšce, musí být podložena distančními sloupky o výšce 4mm.

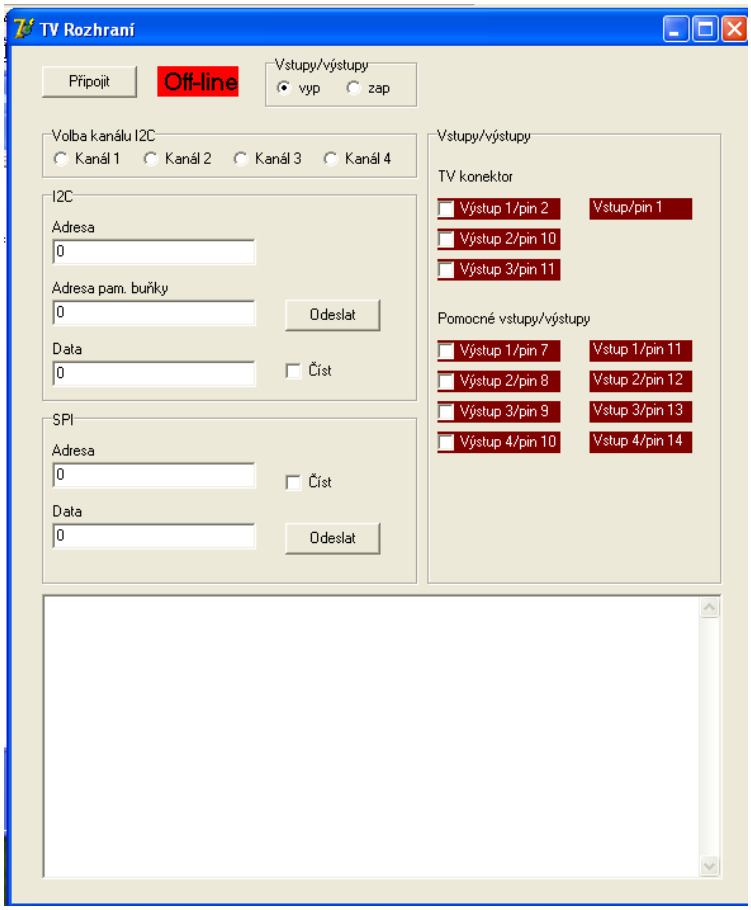
Nakonec byla pro krabičku vyrobena samolepka, na které jsou popisy jednotlivých LED diod a pinů konektorů.

4.3 Ovládací aplikace

Ovládací aplikace je vytvořena v programovacím jazyce Delphi, což byl jeden z bodů zadání diplomového práce. Aplikace slouží pouze k odzkoušení funkce rozhraní, jedná se o ukázkový program pro budoucí uživatele.

Ovládání sběrnice I2C a SPI je předpřipraveno k zapsání nebo přečtení dat na zvolené adrese EEPROM paměti. Na sběrnici I2C je ovládací aplikace připravena komunikovat s pamětí 24C01, ovládání sběrnice je připraveno pro komunikaci s pamětí 93LC46B. Není to tedy univerzální ovládání sběrnic, protože to by bylo kvůli množství nastavení parametrů sběrnic příliš složité. Program má budoucím uživatelů pouze ukázat, jak si připravit vlastní aplikace, jaké použít funkce, a jak funkce používat. Mnoho komentářů k aplikaci je již v samotném zdrojovém kódu.

Náhled aplikace je na **Obrázku 13**.



Obrázek 13.: *Náhled ovládací aplikace rozhraní*

4.3.1 Ovládání programu

K navázání spojení s rozhraním slouží tlačítko Připojit. Po úspěšném spojení se vedle tlačítka Připojit změní nápis Off-line na On-line na zeleném podkladu a tlačítko změní funkci na Odpojit. Pokud by se spojení nepodařilo, aplikace vypíše v samostatném okně chybu.

Napravo od tlačítka Připojit se nachází ovládání třetího stavu shifterů. Při volbě vyp je na pinu 44 převodníku logická úroveň H a všechny shiftery jsou ve třetím stavu.

Pro volbu kanálu I2C slouží box pod tlačítkem Připojit. K volbě je použita komponenta RadioGroup, která dovoluje aktivaci pouze jedné možnosti, což koresponduje s funkcí I2C multiplexeru.

Ovládání digitálních výstupů a indikace digitálních vstupů je v pravém boxu aplikace. Vstupy a výstupy jsou rozděleny podle rozmístění v konektorech rozhraní. Aktivace výstupu se provádí zaškrtnutím políčka příslušné komponenty CheckBox.

Ke komunikaci po sběrnici I2C slouží část pod boxem volby kanálu I2C. Zde jsou komponenty Edit, které slouží k zadávání adres a posílaných dat. Do políčka s popisem Adresa zařízení, se zadává adresa EEPROM paměti, se kterou chceme komunikovat, políčko Adresa v pam. buňky slouží k zadání adresy paměťové buňky, do které chceme zapisovat, nebo z které chceme číst. Zda se budou data zapisovat, nebo číst, se řídí CheckBoxem s popisem Číst. Funkce je zřejmá. Při zaškrtnutí políčka se data posílají s osmým bitem nastaveným pro čtení (viz Odstavec 1.2.1).

Ovládání sběrnice SPI je podobné jako u sběrnice I2C. Rozdílem je pouze počet komponent Edit pro zadávání adres. Protože sběrnice SPI není adresná (viz. Odstavec 1.1.1), zadává se zde pouze adresa paměťové buňky EEPROM paměti, do které chceme zapisovat, nebo z ní číst. Volba zápisu nebo čtení se opět ovládá CheckBoxem s popisem číst.

4.3.2 Popis použitých funkcí

Pro ovládací aplikaci jsou použity příkazy buď přímo importované z výrobcem dodávaných knihoven, (viz. Odstavec 2.1.2) nebo z výrobcem dodávaných unit pro Delphi.

Po stisku tlačítka Připojit dochází k otevírání portů C a D a jejich nastavení do bit-bang režimu:

```
ftStatusC := Open_USB_Device_By_Device_Description('Quad RS232-HS C');
ftHandleC := FT_HANDLE;
ftStatusC := Set_USB_Device_BitMode($00, $00);
ftStatusC := Set_USB_Device_BitMode($3C, $01);
```

Otevírání portů je v ovládací aplikaci prováděno za pomoci funkce *Open_USB_Device_By_Device_Description* z připojené unity D2XXUnit. Jediným parametrem funkce je název otevíraného portu. Při otevírání každého portu převodníku je portu přiřazen vlastní identifikační znak, pomocí kterého je pak port volán jednotlivými příkazy nebo funkcemi. Toto číslo se ukládá do proměnné FT_HANDLE. Protože se zde používá funkce z unity, musí se obsah proměnné FT_HANDLE uložit do jiné proměnné, aby nebyla hodnota přepsána při otevírání dalšího portu. Hodnota z FT_HANDLE je tedy po otevření portu C uložena do proměnné ftHandleC.

Každý příkaz převodníku vrací hodnotu, jestli byl příkaz vykonán dobře či ne. Návratová hodnota je uložena v proměnné FT_STATUS. Hodnota 0 značí úspěšné provedení, jiná návratová hodnota značí chybu. Seznam všech návratových hodnot je popsán v manuálu D2XX_Programmer's_Guide(FT_000071) na přiloženém CD ve složce FT4232H.

Po otevření portu C a D dochází k nastavení Bit-bang režimu. To se provádí funkcí *Set_USB_Device_BitMode(x, x).* Prvně musí být stejnou funkcí proveden reset portu, jak je vidět na části zdrojového kódu výše, kde parametry této funkce mají nulovou hodnotu. Tím dojde k resetu portu a jeho přípravu pro nastavení bit-bang režimu. Dalším použitím funkce dojde k nastavení jednotlivých pinů jako vstupy nebo výstupy. V našem zapojení jsou u portu C nastaveny piny 38, 39, 45 a 46 jako vstupy a 40, 41, 43 a 44 jako výstupy. Jak budou které piny nastaveny, záleží na hodnotě prvního parametru funkce *Set_USB_Device_BitMode(x, x).* Ten se nazývá maska portu, jeho jednotlivé bity stanovují, jak budou piny nastaveny. Pokud bude příslušný pin nastaven na logickou 0, funguje jako vstup, pokud na logickou 1, pak funguje jako výstup. V našem případě musí být hodnota parametru vyjádřena binárně: 00111100, v hexadecimální soustavě je tato hodnota pak představována číslem 3C, což odpovídá zdrojovému kódu programu.

Druhý parametr funkce pak nastavuje, v jakém módu bude port pracovat. Hodnota 1 značí asynchronní přenos, další hodnoty i s vysvětlením jsou nalezeny ve výše zmíněném manuálu.

Nastavení výstupů se provádí zápisem příslušné hodnoty na port. K tomu slouží funkce *FT_Write(ftHandleD, @FT_Out_Buffer, BufferSizeD, @res).* Funkce FT_Write má celkem čtyři parametry. První je identifikace portu, na který chceme zapisovat, druhý parametr je odkaz na data, která chceme zapisovat, a třetí udává počet bytů k zápisu. Poslední parametr vrací hodnotu, kolik bytů bylo ve skutečnosti zapsáno.

K načtení stavů vstupů slouží pro port C funkce *Get_USB_Device_BitMode(FT_In_BufferC)*. Tato funkce přečte aktuální bitovou hodnotu všech pinů portu a uloží ji do proměnné FT_In_BufferC. Tato hodnota ale obsahuje všechny bity z celého portu, proto je nutné získat hodnotu jen toho bitu, která nás zajímá. To se v ovládací aplikace provádí testováním výsledku logického součtu:
if (FT_In_BufferC or 191) = 191
Zde se testuje nastavení pinu 45 na portu C. Pin 45 je sedmý bit celého portu, proto je logický součet proveden s číslem 191, což odpovídá binární hodnotě 10111111. Pokud není bit nastaven, výsledkem je opět číslo 191.

Příkazy pro ovládání portů sběrnic a řízení toku samotných informací jsou podobné výše popsaným funkcím.

U sběrnice I2C je port otvírán příkazem *I2C_OpenChannel(0, @ftHandleA).* Příkaz je opět přebrán ze zdrojového kódu ovládací aplikace. První parametr udává index portu s MPSSE procesorem, který chceme otevřít. U rozhraní je sběrnice I2C připojena k portu A, tj. prvnímu portu s MPSSE. Proto je index roven 0. Druhý parametr potom vrací znak portu, kterým bude identifikován.

Před započetím čtení nebo zápisu po I2C je nutné port inicializovat. K tomu je v ovládací aplikaci použit příkaz *I2C_InitChannel(ftHandleA, @ChannelConfig).* Parametr ChannelConfig je pointer na strukturu dat, kde je uložena přenosová frekvence, délka času latency timeru a další konfigurace sběrnice. Všechny funkce I2C a popis jejich parametrů je možné nalézt v souboru AN_177_User_Guide_For_LibMPSSE-I2C na přiloženém CD.

Zápis dat na sběrnici se provádí příkazem *I2C_DeviceWrite(ftHandleA, deviceAddress, 1, @data1, @bytesToTransfered, 1).* První parametr je zřejmý z předchozího popisu. Druhý parametr představuje adresu slave zařízení na sběrnici, do kterého chceme zapisovat, nebo z něj chceme číst. Další udává počet přenesených bytů. Následující parametr je pointer na místo, kde jsou uložena data, která chceme přenést. Předposlední parametr je pointer na počet skutečně přenesených bytů, poslední parametr slouží k nastavení, zdali bude vysílán startbit, stopbit, signál ACK atd. Hodnota tohoto parametru je podrobně popsána v manuálu AN_177_User_Guide_For_LibMPSSE-I2C.

K čtení dat přes I2C sběrnici se používá příkaz *I2C_DeviceRead(ftHandleA, deviceAddress, 1, @data1, @bytesToTransfered, 15).* Parametry mají stejnou funkci jako u příkazu *I2C_DeviceWrite.*

U sběrnice SPI se pro otevření kanálu používá příkaz *SPI_OpenChannel(1, @ftHandleB).* Parametry mají stejnou funkci jako u výše popsaného příkazu *I2C_OpenChannel(0, @ftHandleA).* Zde je ale hodnota indexu porta rovna jedné, protože sběrnice SPI je u rozhraní připojena na port B.

Před započetím jakékoliv komunikace je opět třeba sběrnici inicializovat. Příkaz *SPI_InitChannel(ftHandleB, @ChannelConfig)* je prakticky totožný s příkazem pro inicializaci sběrnice I2C. I příkaz pro zápis má hodně podobnou strukturu. Rozdíl je jen v počtu parametrů, kde u příkazu pro zápis na sběrnici SPI chybí parametr adresy zařízení. Zápis v ovládací aplikaci se na sběrnici SPI se provádí příkazem *SPI_Write(ftHandleB, @Out_Buff, 11, @res, 7),* čtení potom příkazem *SPI_Read(ftHandleB, @In_Buff, 2, @res, 4).*

Popis všech příkazů pro sběrnici SPI je možné nalézt v manuálu AN_178_User Guide for LibMPSSE-SPI na přiloženém CD.

Po ukončení práce s porty je nutné je uzavřít, jinak by zůstaly v operačním systému počítače jako otevřené a při opětovném připojení by byly nedostupné. Uzavření se provádí v ovládací aplikaci funkcí *Close_USB_Device* u portů C a D, u sběrnice I2C příkazem *I2C_CloseChannel(ftHandleA)* a u sběrnice SPI *SPI_CloseChannel(ftHandleB).*

Závěr

Návrh rozhraní odpovídá požadavkům na něj kladených. Je napájeno z USB portu, obsahuje čtyři kanály sběrnice I2C a umí komunikovat i po sběrnici SPI. Dále obsahuje dostatečný počet digitálních vstupů a výstupů. Veškerá činnost na vodičích sběrnice I2C je signalizována LED diodami, také jsou signalizovány stavy digitálních vstupů a výstupů.

Rozhraní také umožňuje volbu výstupního napětí 3,3V nebo 5V. Všechny vstupní/vstupní linky jsou zapojeny přes budiče, takže nehrozí poškození převodníku FT4232H nadměrným proudem portů. Všechny vstupní a výstupní linky jsou chráněny shottkyho diodami proti přepětí vznikajícím při připojování rozhraní k periferiím.

Po realizaci rozhraní bylo shledáno několik nedostatků. Prvně se neosvědčila domácí výroba DPS s takto jemnými spoji. Dále bylo shledáno, že napájecí cesty by měly být silnější, aby nedocházelo k velkému úbytku napětí při proudových rázech. Při kontrolním měření osciloskopem bylo zjištěno kmitání napětí 3,3V, které způsobovaly tlumivky zapojené v obvodu napájení fyzické vrstvy převodníku FT4232H. Nakonec bylo vyhodnoceno jako nevhodné použití budiče LED diod 74HC244, protože LED diody při nečinnosti sběrnice stále svítí. Problém se vyřeší použitím budiče 74HC241, který invertuje procházející signál. Nakonec byla shledána závada při návrhu připojení zdvihacích rezistorů u kanálů I2C sběrnice, které jsou připojeny na spojích mezi shiftery a konektory. Rezistory byly omylem připojeny k nulovému potenciálu. Na zkušebním vzorku rozhraní byly spoje nulového potenciálu příslušných rezistorů přerušeny a kladný potenciál byl připojen pomocí dvou krátkých vodičů.

Některé tyto vady byly odstraněny na zkušebním vzorku, všechny vady jsou pak odstraněny v návrhu schématu a DPS. Další vyráběné zařízení budou tedy již bez závad. Rozhraní je z hardwarového hlediska plně funkční a dokazuje, že návrh rozhraní je v pořádku.

Ze softwarového hlediska jsou zde možnosti, jak aplikaci vylepšit. Zkušební aplikace není bez chybná, komunikace po I2C i SPI má několik vad, které by bylo třeba odladit. Ale všechny hlavní funkce jsou správné, aplikace tedy plní funkci příkladu pro uživatele.

Seznam použité a citované literatury

[1] Dudáček, K. Sériová rozhraní SPI, Microwire, I2C a CAN
Dostupný z WWW: http://home.zcu.cz/~dudacek/NMS/Seriova_rozhrani.pdf

[2] Tišňovský, P. Komunikace po sériové sběrnici I2C [online].2012
Dostupný z WWW: www.root.cz

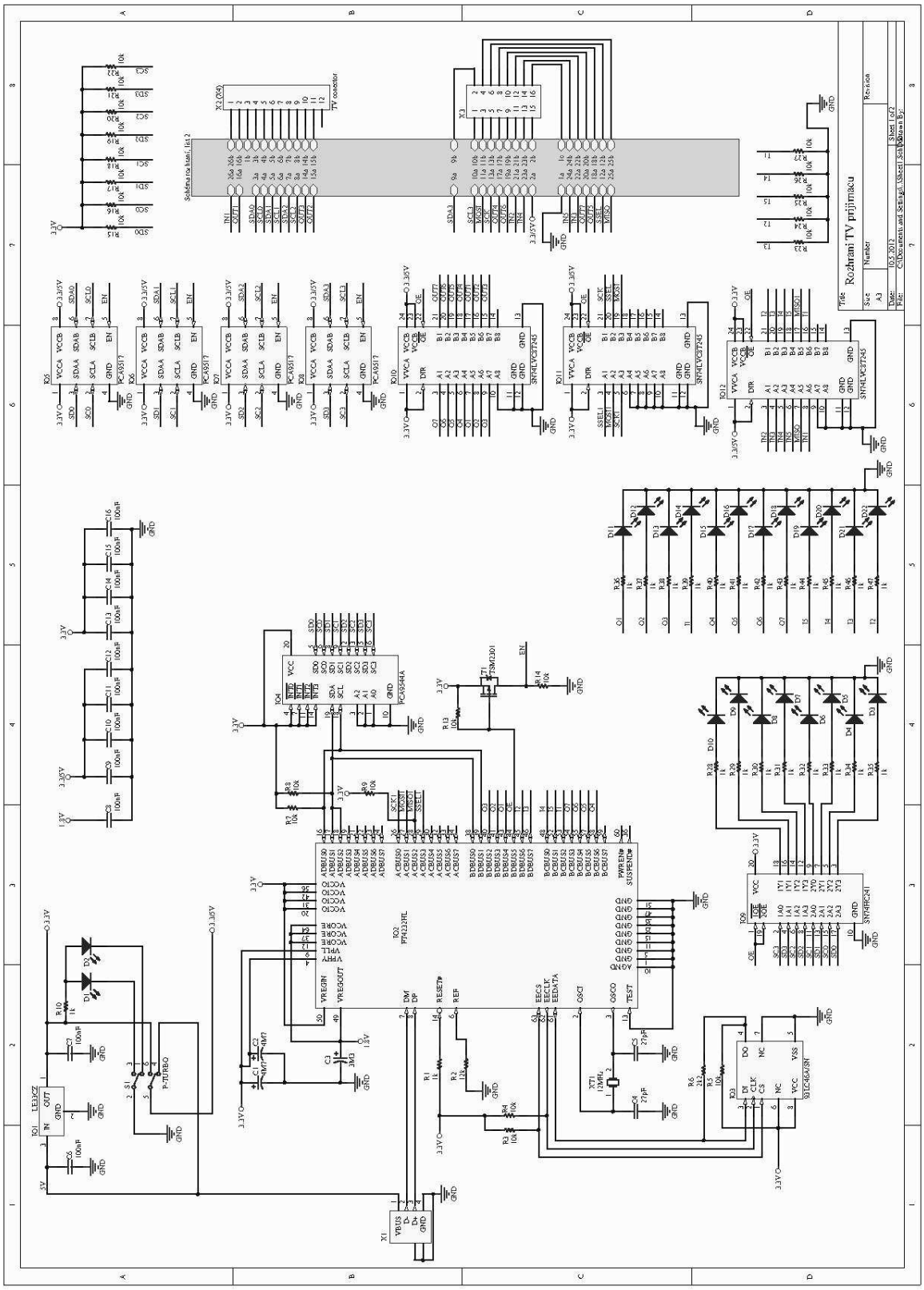
[3] FTDI Datasheet FT4232H [online].2012
Dostupný z WWW: www.ftdi.cz

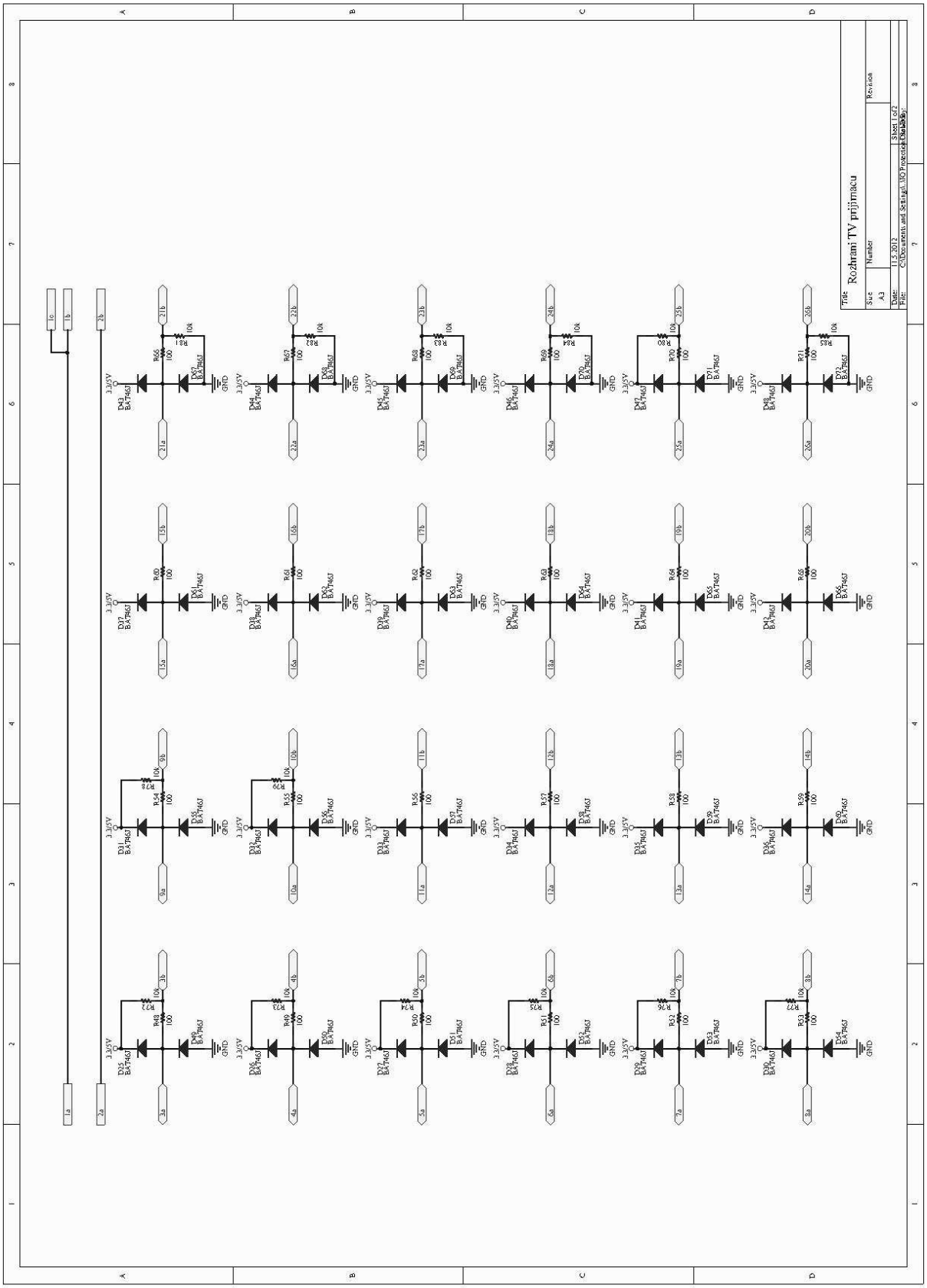
[4] Texas Instruments Datasheet PCA9544 [online].2012
Dostupný z WWW: www.ti.cz

[5] Texas Instruments Datasheet PCA9517 [online].2012
Dostupný z WWW: www.ti.cz

[6] Texas Instruments Datasheet SN74LVC8T245 [online].2012
Dostupný z WWW: www.ti.cz

Příloha A - Schémata




Rozhraní TV přijímacu
Size
A3
Number
Revision
Date:
11.5.2012
Sheet 1 of 2
File:
C:\Documents and Settings\...\MCP Protocols\Schematic1.SchDoc

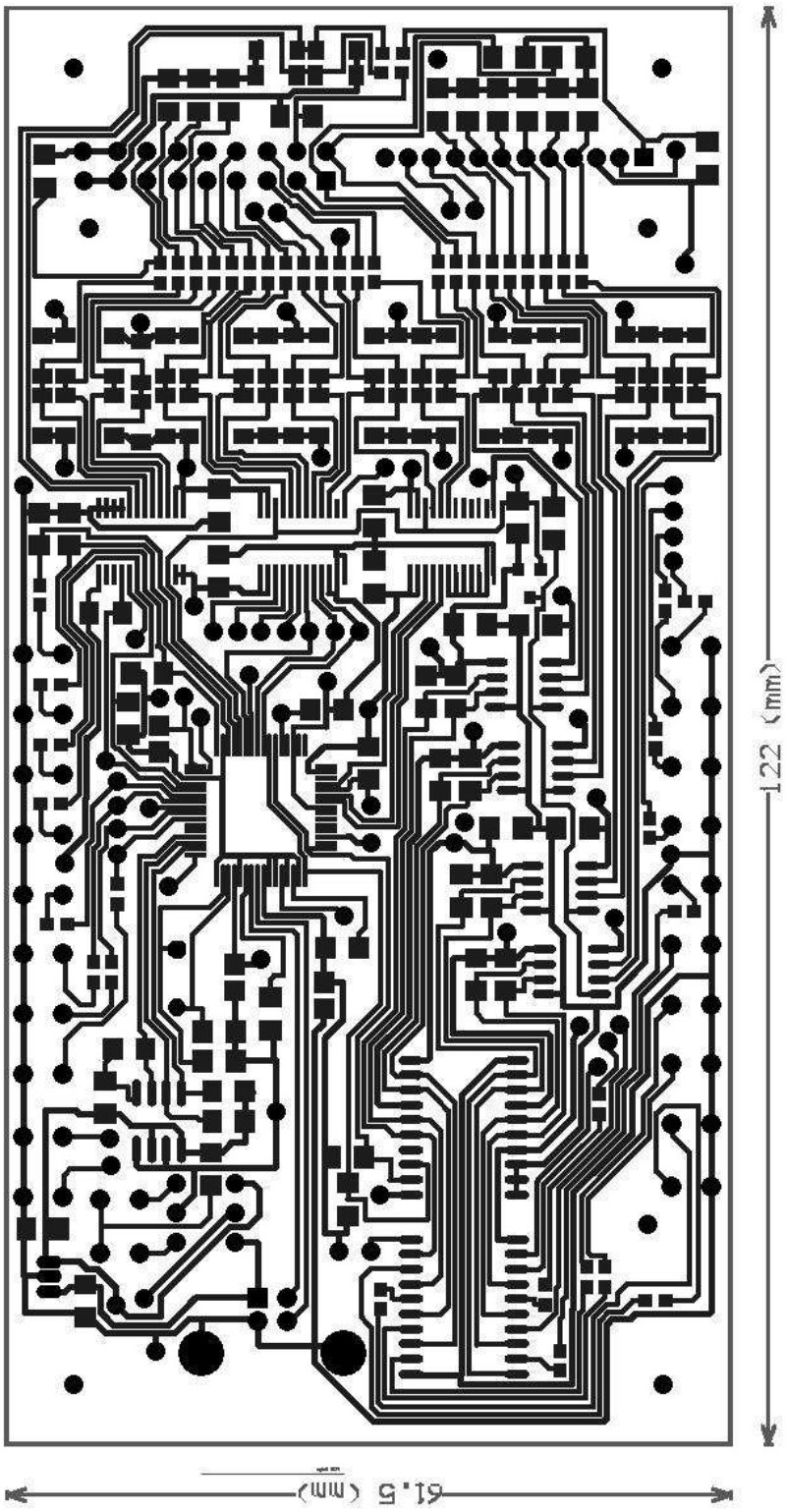
Příloha B – Seznam součástek

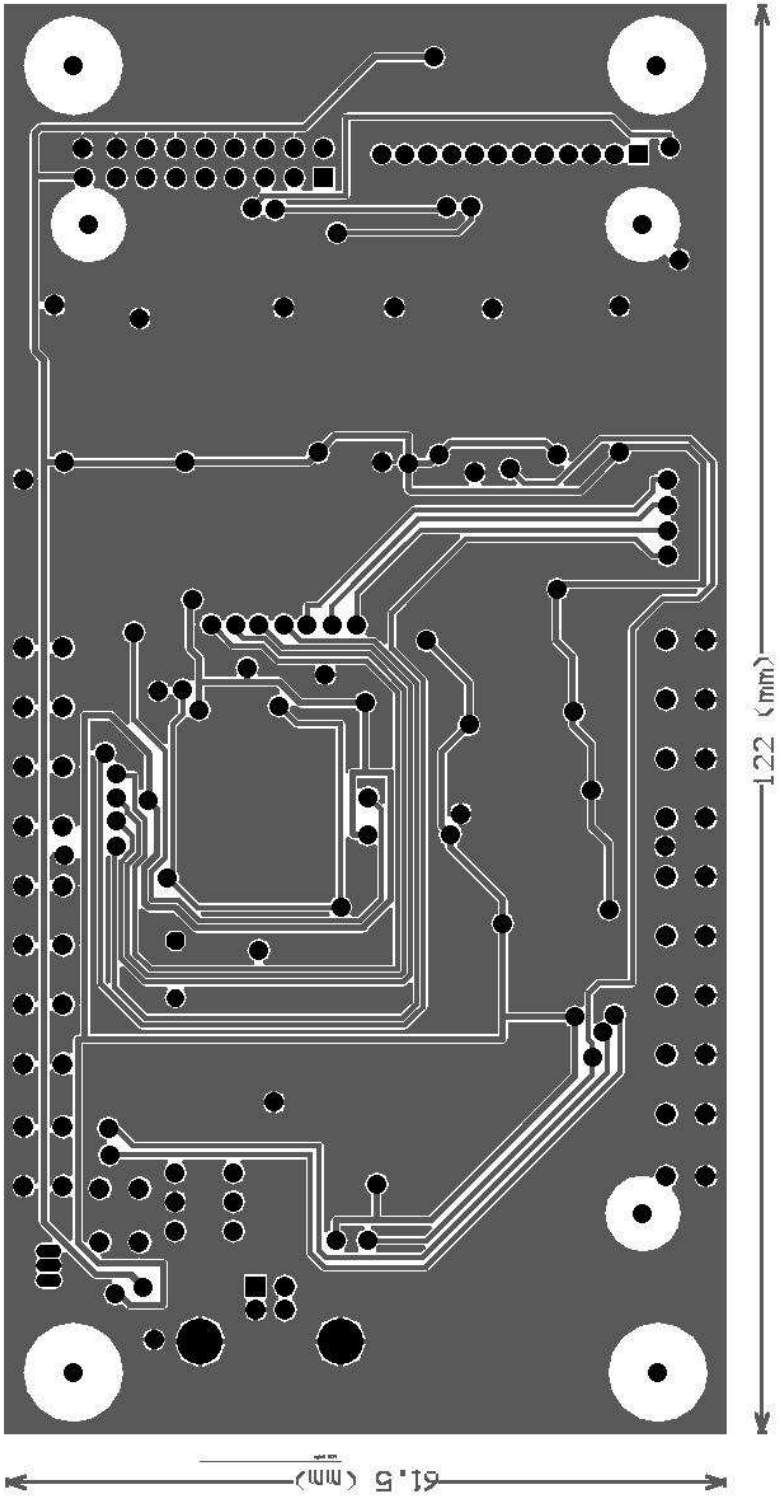
Označení	Typ	Poznámky
C1, C2	KONDENZÁTOR 4,7µF CTS 4M7/16V A	
C3	KONDENZÁTOR 3,3µF CTS 3M3/16V A	
C4, C5	KONDENZÁTOR 27pF CK1206 27P/50V NPO	
C6 – C16	KONDENZÁTOR 100nF CK1206 100N/50V NPO	
D1, D2, D14	DIODA L-HLMP-1790	ø3mm; zelená; 1,8V; 2mA
D3 – D10	DIODA L-HLMP-1719	ø3mm; žlutá; 1,8V; 2mA
D11 – D22	DIODA L-HLMP-1700	ø3mm; červená; 1,8V; 2mA
D25 – D72	DIODA TMMBAT46	dodává TME
IO1	INTEG. STABILIZÁTOR LE33CZ	3,3V; 100mA; TO-92
IO2	INTEG. OBVOD FT4232H	dodává TME
IO3	INTEG. OBVOD 93LC46B-I/SN	dodává TME
IO4	INTEG. OBVOD PCA9544AD	dodává TME
IO5 – IO8	INTEG. OBVOD PCA9517D	dodává TME
IO9	INTEG. OBVOD CD74HC241 SMD	dodává TME
IO10 – IO12	INTEG. OBVOD SN74LVC8T245 PW	dodává TME
R1	REZISTOR 1kΩ R1206 1K0	
R2	REZISTOR 12kΩ R1206 12K	
R3 - R5	REZISTOR 10kΩ R1206 10K	
R6	REZISTOR 2,2kΩ R1206 2K2	

R7 – R9	REZISTOR 10kΩ R1206 10K	
R10	REZISTOR 1kΩ R1206 1K0	
R11 – R27	REZISTOR 10kΩ R1206 10K	
R28 – R47	REZISTOR 1kΩ R0603 1K0	
R48 – R71	REZISTOR 100Ω R0603 100R	
R72 – R85	REZISTOR 10kΩ R1206 10K	
S1	TLAČÍTKO S ARETACÍ P-TURBO	dodává GME
T1	TRANZISTOR TSM 2301CX	
X1	USB KONEKTOR USB1X90B PCB	
X2	KONEKTOR PRO PŘIPOJENÍ TV	
X3	OBOUSTRANNÝ KOLÍK ASS22020G	
X4*	DUTINKOVÁ LIŠTA BTK15G + OBOUSTRANNÝ KOLÍK S2G10	
XT1	KRYSTAL 12MHz HC49S Q 12.000MHz	

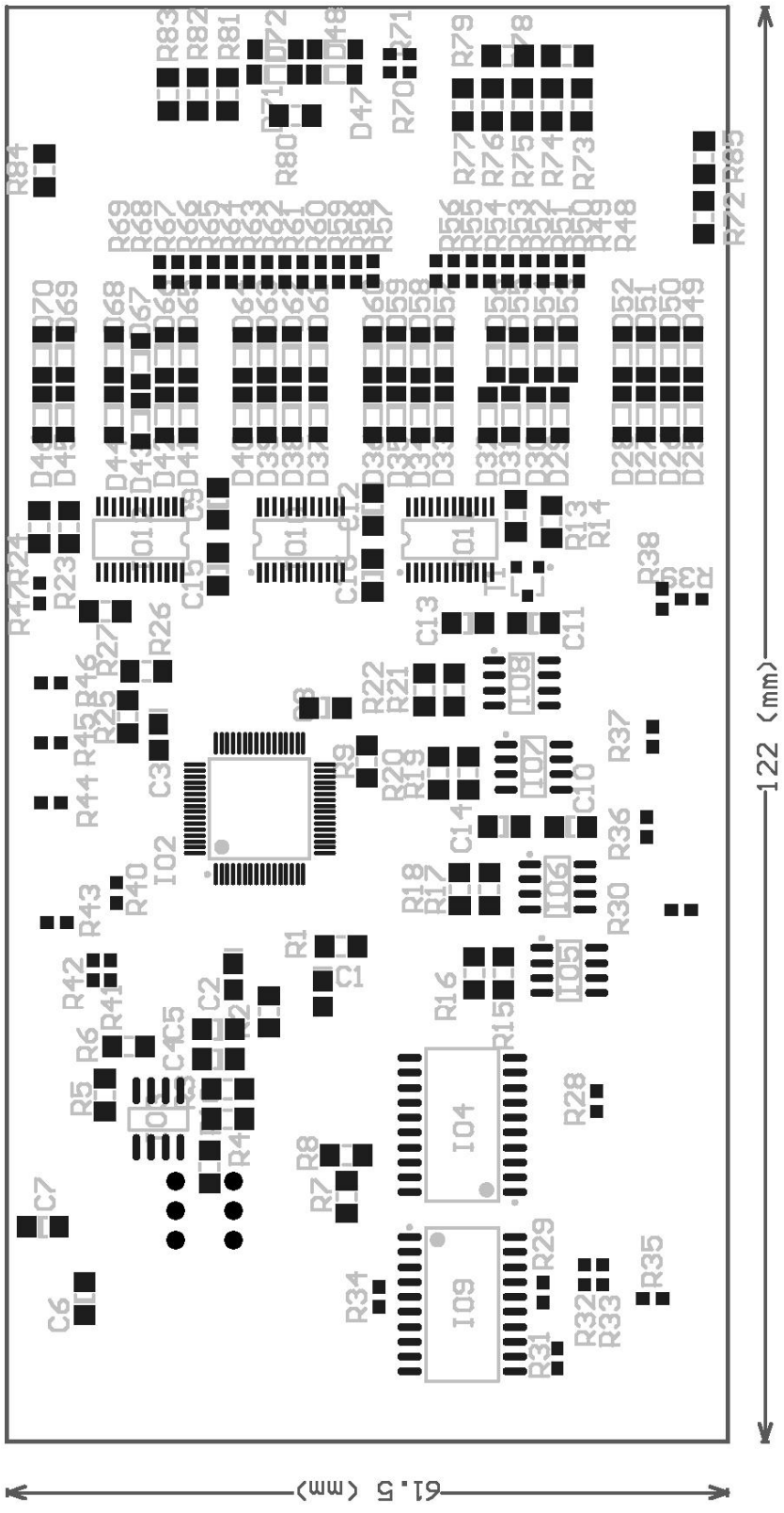
* použit jen při umístění rozhraní do krabičky

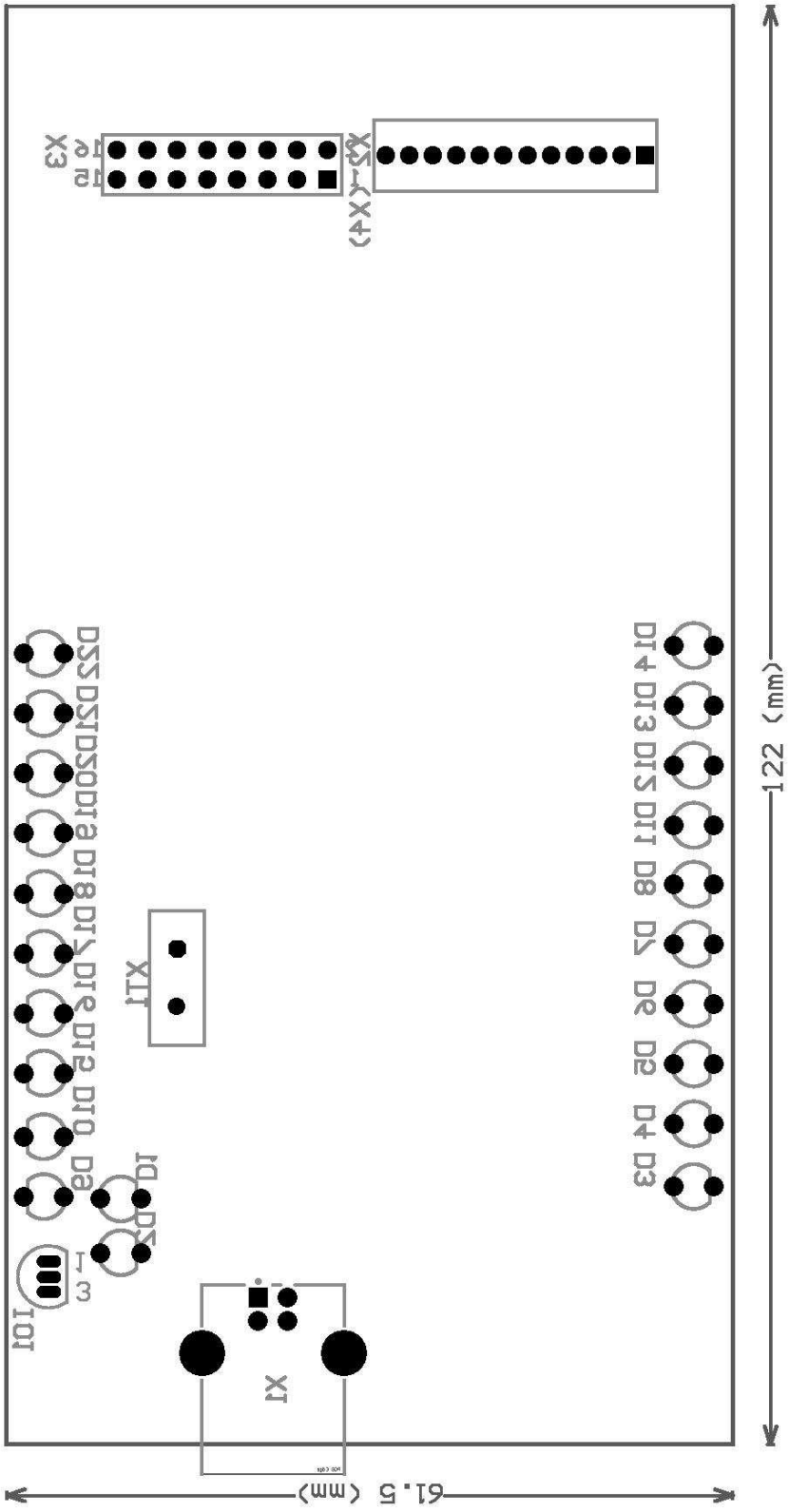
Příloha C – Deska plošných spojů




122 (mm)
61,5 (mm)

Příloha D – Osazení součástek




122 (mm)
61.5 (mm)

Příloha E – Výpis zdrojového kódu ovládací aplikace

```
unit Unit1;

interface

uses

  D2XXUnit, D2XXUnit2, DSI2C, DSSPI, FT2232HSPIUnit, Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,

  Dialogs, StdCtrls, ExtCtrls;

type

  //FT_HANDLE = DWord;

  FT_STATUS = DWord;

const

  FT_DLL_Name = 'FTD2XX.dll';

  IC_DLL_Name = 'libMPSSE.dll';

  SPI_DLL_Name = 'libMPSSE.dll';

  FT2232SPI_DLL_Name = 'ftcspi.dll';


  FT_OPEN_BY_SERIAL_NUMBER = 1;

  FT_OPEN_BY_DESCRIPTION = 2;


  FT_OK = 0;


type

  TForm1 = class(TForm)

    lbl_1: TLabel;

    btnPripojit: TButton;

    rdgZapIO: TRadioGroup;

    Timer1: TTimer;

    rdgKanal: TRadioGroup;

    grpI2C: TGroupBox;

    edAdresaI2C: TEdit;
```

edData1I2C: TEdit;

btnOdeslatI2C: TButton;

chcCistI2C: TCheckBox;

lblAdresaI2C: TLabel;

lblData1I2C: TLabel;

grpSPI: TGroupBox;

edAdresaSPI: TEdit;

lblAdresaSPI: TLabel;

lblDataSPI: TLabel;

edDataSPI: TEdit;

btnOdeslatSPI: TButton;

chcCistSPI: TCheckBox;

grpIO: TGroupBox;

lblTVkonektor: TLabel;

chcTVvystup2: TCheckBox;

chcTVvystup3: TCheckBox;

chcTVvystup1: TCheckBox;

lblPomIO: TLabel;

chcVystup1: TCheckBox;

chcVystup2: TCheckBox;

chcVystup3: TCheckBox;

chcVystup4: TCheckBox;

Memo1: TMemo;

lblTVVstup: TLabel;

lblIOVstup1: TLabel;

lblIOVstup2: TLabel;

lblIOVstup3: TLabel;

lblIOVstup4: TLabel;

edData2I2C: TEdit;

lblDataI2C: TLabel;

```
    procedure btnPripojitClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure btnOdeslatSPIClick(Sender: TObject);
    procedure btnOdeslatI2CClick(Sender: TObject);

  private
    ftHandleA:Dword;
    ftHandleB:Dword;
    ftHandleC:Dword;
    ftHandleD:Dword;
    //FT_HANDLE:Dword;
    ftStatusA:FT_STATUS;
    ftStatusB:FT_STATUS;
    ftStatusC:FT_STATUS;
    ftStatusD:FT_STATUS;
    BufferoutC : integer;
    Bufferout : integer;
    BufferinC : integer;
    BufferinD : integer;
  public
  end;

var
  Form1: TForm1;
  My_Names : Names;
  PortAIsOpen : boolean;
  ICChannel : Dword;
  ICChannelSPI : Dword;
  data : byte;
  list: Dword;
```

```
 res : byte;
 FT_Out_Buffer : byte;
 ChannelConfig : ARRAY[0..256] of Dword;
 LastItemIndex : integer;

implementation

{$R *.dfm}

function FT_CreateDeviceInfoList(lpdwNumDevs:pointer) : FT_STATUS ; stdcall ; External FT_DLL_Name;
function FT_OpenEx(pvArg1:Pointer;dwFlags:Dword;ftHandle:Pointer) : FT_STATUS ; stdcall ; External
FT_DLL_Name name 'FT_OpenEx';
function FT_ResetPort(ftHandle:Dword) : FT_Status ; stdcall ; External FT_DLL_Name;
function FT_SetBitMode (ftHandle:Dword; ucMask:Byte; ucMode:Byte) : FT_STATUS;  External
FT_DLL_Name;
function FT_SetDtr(ftHandle:Dword) : FT_Result ; stdcall ; External FT_DLL_Name name 'FT_SetDtr';
function FT_ClrDtr(ftHandle:Dword) : FT_Result ; stdcall ; External FT_DLL_Name name 'FT_ClrDtr';
function FT_Write(ftHandle:Dword; FTOutBuf : Pointer; BufferSize : LongInt; ResultPtr : Pointer ) : FT_Status
; stdcall ; External FT_DLL_Name;
function FT_Read(ftHandle:Dword; FTInBuf : Pointer; BufferSize : LongInt; ResultPtr : Pointer ) : FT_Status ;
stdcall ; External FT_DLL_Name name 'FT_Read';
function FT_ResetDevice(ftHandle:Dword) : FT_Status ; stdcall ; External FT_DLL_Name name
'FT_ResetDevice';
function I2C_GetNumChannels(numChannels:pointer) : FT_Status ; stdcall ; External IC_DLL_Name;
function I2C_GetChannelInfo(index:dword; FT_DEVICE_LIST_INFO_NODE:pointer) : FT_Status ; stdcall ;
External IC_DLL_Name;
function I2C_OpenChannel(index:integer; fthandle: pointer) : FT_Status ; stdcall ; External IC_DLL_Name;
function I2C_InitChannel(fthandle: Dword; ChannelConfig: pointer) : FT_Status ; stdcall ; External
IC_DLL_Name;
function I2C_CloseChannel(fthandle:Dword) : FT_Status ; stdcall ; External IC_DLL_Name;
function I2C_DeviceWrite(fthandle: dword; deviceAddress:Dword; bytesToTransfer: Dword; buffer: pointer;
bytesTransfered: pointer; options: Dword ) : FT_Status ; stdcall ; External IC_DLL_Name;
function I2C_DeviceRead(fthandle: dword; deviceAddress:Dword; bytesToTransfer: Dword; buffer: pointer;
bytesTransfered: pointer; options: Dword ) : FT_Status ; stdcall ; External IC_DLL_Name;
function SPI_GetNumChannels(numChannels:pointer) : FT_Status ; stdcall ; External SPI_DLL_Name;
```

```
function SPI_OpenChannel(index:integer; fthandle:pointer) : FT_Status ; stdcall ; External SPI_DLL_Name;

function SPI_InitChannel(fthandle: Dword; ChannelConfig: pointer) : FT_Status ; stdcall ; External
SPI_DLL_Name;

function SPI_CloseChannel(fthandle:Dword) : FT_Status ; stdcall ; External SPI_DLL_Name;

function SPI_Read(fthandle: dword; buffer: pointer; sizeToTransfer: Dword; bytesTransfered: pointer; options:
Dword ) : FT_Status ; stdcall ; External SPI_DLL_Name;

function SPI_Write(fthandle: dword; buffer: pointer; sizeToTransfer: Dword; bytesTransfered: pointer; options:
Dword ) : FT_Status ; stdcall ; External SPI_DLL_Name;

function SPI_GetNumHiSpeedDevices(pNumHiSpeedDevices: pointer): FTC_Status; stdcall ; External
FT2232SPI_DLL_Name name 'SPI_GetNumHiSpeedDevices';

function SPI_GetHiSpeedDeviceNameLocIDChannel(DeviceNameIndex: Dword; pDeviceNameBuffer:
PDeviceName; DeviceNameBufferSize: Dword; pLocationID: PDword; pChannelBuffer: PChannel;
ChannelBufferSize: Dword; pHiSpeedDeviceType: PDword): FTC_STATUS; stdcall ; External
FT2232SPI_DLL_Name name 'SPI_GetHiSpeedDeviceNameLocIDChannel';

function SPI_OpenHiSpeedDevice(DeviceName: String; LocationID: Dword; Channel: String; pFtHandle:
PDword): FTC_STATUS; stdcall ; External FT2232SPI_DLL_Name name 'SPI_OpenHiSpeedDevice';

function SPI_Close(fthandle: Dword): FTC_STATUS; stdcall ; External FT2232SPI_DLL_Name name
'SPI_Close';

function SPI_InitDevice(fthandle: Dword; dwClockFrequencyValue: Dword): FTC_STATUS; stdcall ; External
FT2232SPI_DLL_Name name 'SPI_InitDevice';

function SPI_SetClock(fthandle: Dword; dwClockDivisor: Dword; pdwClockFrequencyHz: PDword):
FTC_STATUS; stdcall ; External FT2232SPI_DLL_Name name 'SPI_SetClock';


procedure TForm1.btnPripojitClick(Sender: TObject);

begin

if btnPripojit.Caption = 'Připojit'

 then

  begin

    Timer1.Enabled := False;

    ftStatusC:= Open_USB_Device_By_Device_Description('Quad RS232-HS C'); //Otevírání portu C

    ftHandleC := FT_HANDLE; //Ukládání získaného Handle do HandleC pro pozdější použití

    ftStatusC:= Set_USB_Device_BitMode($00, $00); //Nastavení Bit-bang modu -reset

    ftStatusC:= Set_USB_Device_BitMode($3C, $01); //Nastavení Bit-bang modu - nastavení které pny budou
vstupy a které výstupy

    ftStatusD:= Open_USB_Device_By_Device_Description('Quad RS232-HS D');

    ftHandleD := FT_HANDLE;
```

```
   ftStatusD:= Set_USB_Device_BitMode($00, $00);
   ftStatusD:= Set_USB_Device_BitMode($78, $01);
   If (ftStatusC=0) and (ftStatusD=0)  //Kontrola otevření portů
    then
     begin
      btnPripojit.Caption := 'Odpojit';
      lbl_1.Caption := 'On-line';
      lbl_1.Color := clLime;
      Timer1.Enabled := True;
      LastItemIndex := 4;
     end;

 end
else
 begin
   FT_HANDLE := ftHandleC;
   Close_USB_Device;
   FT_HANDLE := ftHandleD;
   Close_USB_Device;
   btnPripojit.Caption := 'Připojit';
   lbl_1.Caption := 'Off-line';
   lbl_1.Color := clRed;
 end;
end;

procedure TForm1.Timer1Timer(Sender: TObject);   //Slouží ke kontrole všech CheckBoxů a zda-li není signál
na nějakém fig. vstupu
var   FT_Out_Buffer : byte;
      FT_In_BufferC : byte;
      FT_In_BufferD : byte;
      BufferSizeC : integer;
```

```
    BufferSizeD : integer;
    res : byte;
    ValueMUXControlRegister : integer;
    bytesToTransfered : Dword;
begin
 if lbl_1.Caption = 'On-line'
  then
   //Nastavení hodnoty BufferSize podle zatržených CheckBoxů - port D (slouží k nastavení dig. výstupů
   begin
   FT_Out_Buffer := 0;
   if (chcVystup1.Checked = false) and (chcVystup2.Checked = false) and (chcVystup3.Checked = false) and
(chcVystup4.Checked = false)
    then BufferSizeD := 1;

   if (chcVystup1.Checked = true) and (chcVystup2.Checked = false) and (chcVystup3.Checked = false) and
(chcVystup4.Checked = false)
    then BufferSizeD := 8;

   if (chcVystup1.Checked = false) and (chcVystup2.Checked = true) and (chcVystup3.Checked = false) and
(chcVystup4.Checked = false)
    then BufferSizeD := 26;

   if (chcVystup1.Checked = false) and (chcVystup2.Checked = false) and (chcVystup3.Checked = true) and
(chcVystup4.Checked = false)
    then BufferSizeD := 12;

   if (chcVystup1.Checked = false) and (chcVystup2.Checked = false) and (chcVystup3.Checked = false) and
(chcVystup4.Checked = true)
    then BufferSizeD := 7;

   if (chcVystup1.Checked = true) and (chcVystup2.Checked = false) and (chcVystup3.Checked = true) and
(chcVystup4.Checked = true)
    then BufferSizeD := 6;
```

if (chcVystup1.Checked = true) and (chcVystup2.Checked = false) and (chcVystup3.Checked = false) and (chcVystup4.Checked = true)

then BufferSizeD := 22;

if (chcVystup1.Checked = true) and (chcVystup2.Checked = true) and (chcVystup3.Checked = false) and (chcVystup4.Checked = false)

then BufferSizeD := 70;

if (chcVystup1.Checked = true) and (chcVystup2.Checked = true) and (chcVystup3.Checked = true) and (chcVystup4.Checked = false)

then BufferSizeD := 152;

if (chcVystup1.Checked = true) and (chcVystup2.Checked = true) and (chcVystup3.Checked = true) and (chcVystup4.Checked = true)

then BufferSizeD := 11;

if (chcVystup1.Checked = false) and (chcVystup2.Checked = true) and (chcVystup3.Checked = true) and (chcVystup4.Checked = true)

then BufferSizeD := 18;

if (chcVystup1.Checked = false) and (chcVystup2.Checked = false) and (chcVystup3.Checked = true) and (chcVystup4.Checked = true)

then BufferSizeD := 202;

if (chcVystup1.Checked = false) and (chcVystup2.Checked = true) and (chcVystup3.Checked = false) and (chcVystup4.Checked = true)

then BufferSizeD := 10;

if (chcVystup1.Checked = true) and (chcVystup2.Checked = false) and (chcVystup3.Checked = true) and (chcVystup4.Checked = false)

then BufferSizeD := 1; //doplnit

if (chcVystup1.Checked = true) and (chcVystup2.Checked = true) and (chcVystup3.Checked = false) and (chcVystup4.Checked = true)

```
 then BufferSizeD := 50;


 if (chcVystup1.Checked = false) and (chcVystup2.Checked = true) and (chcVystup3.Checked = true) and
(chcVystup4.Checked = false)

 then BufferSizeD := 74;


 FT_HANDLE := ftHandleD;

 If (BufferSizeD - Bufferout) <>0 //Porovnání, jestli se hodnota změnila oproti minulé kontrole

 then

 begin

  Bufferout := BufferSizeD;

  ftStatusD:= FT_Write(ftHandleD, @FT_Out_Buffer, BufferSizeD, @res ); //Zápis na port D - nastavení
výstupů dle zatržených CheckBoxů

  If ftStatusD = 0

   then

    begin

     //Nastavení podbarvení výstupů, které značí, že je výstup aktivován

     if chcVystup1.Checked = true

      then chcVystup1.Color := clRed

      else chcVystup1.Color := clMaroon;

     if chcVystup2.Checked = true

      then chcVystup2.Color := clRed

      else chcVystup2.Color := clMaroon;

     if chcVystup3.Checked = true

      then chcVystup3.Color := clRed

      else chcVystup3.Color := clMaroon;

     if chcVystup4.Checked = true

      then chcVystup4.Color := clRed

      else chcVystup4.Color := clMaroon;

    end

   else MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
```

```
    end;


  //Nastavení výstupů pro port C

  FT_Out_Buffer := 0;

  if (chcTVvystup1.Checked = false) and (chcTVvystup2.Checked = false) and (chcTVvystup3.Checked = false)
and (rdgZapIO.ItemIndex = 0)

   then BufferSizeC := 1;


  if (chcTVvystup1.Checked = true) and (chcTVvystup2.Checked = false) and (chcTVvystup3.Checked = false)
and (rdgZapIO.ItemIndex = 0)

   then BufferSizeC := 4;


  if (chcTVvystup1.Checked = false) and (chcTVvystup2.Checked = true) and (chcTVvystup3.Checked = false)
and (rdgZapIO.ItemIndex = 0)

   then BufferSizeC := 78;


  if (chcTVvystup1.Checked = false) and (chcTVvystup2.Checked = false) and (chcTVvystup3.Checked = true)
and (rdgZapIO.ItemIndex = 0)

   then BufferSizeC := 14;


  if (chcTVvystup1.Checked = false) and (chcTVvystup2.Checked = false) and (chcTVvystup3.Checked = false)
and (rdgZapIO.ItemIndex = 1)

   then BufferSizeC := 26;


  if (chcTVvystup1.Checked = true) and (chcTVvystup2.Checked = false) and (chcTVvystup3.Checked = true)
and (rdgZapIO.ItemIndex = 1)

   then BufferSizeC := 74;


  if (chcTVvystup1.Checked = true) and (chcTVvystup2.Checked = false) and (chcTVvystup3.Checked = false)
and (rdgZapIO.ItemIndex = 1)

   then BufferSizeC := 178;


  if (chcTVvystup1.Checked = true) and (chcTVvystup2.Checked = true) and (chcTVvystup3.Checked = false)
and (rdgZapIO.ItemIndex = 0)
```

```
   then BufferSizeC := 1;  //doplnit


  if (chcTVvystup1.Checked = true) and (chcTVvystup2.Checked = true) and (chcTVvystup3.Checked = true)
and (rdgZapIO.ItemIndex = 0)

   then BufferSizeC := 6;


  if (chcTVvystup1.Checked = true) and (chcTVvystup2.Checked = true) and (chcTVvystup3.Checked = true)
and (rdgZapIO.ItemIndex = 1)

   then BufferSizeC := 11;


  if (chcTVvystup1.Checked = false) and (chcTVvystup2.Checked = true) and (chcTVvystup3.Checked = true)
and (rdgZapIO.ItemIndex = 1)

   then BufferSizeC := 50;


  if (chcTVvystup1.Checked = false) and (chcTVvystup2.Checked = false) and (chcTVvystup3.Checked = true)
and (rdgZapIO.ItemIndex = 1)

   then BufferSizeC := 51;


  if (chcTVvystup1.Checked = false) and (chcTVvystup2.Checked = true) and (chcTVvystup3.Checked = false)
and (rdgZapIO.ItemIndex = 1)

   then BufferSizeC := 10;


  if (chcTVvystup1.Checked = true) and (chcTVvystup2.Checked = false) and (chcTVvystup3.Checked = true)
and (rdgZapIO.ItemIndex = 0)

   then BufferSizeC := 43;


  if (chcTVvystup1.Checked = true) and (chcTVvystup2.Checked = true) and (chcTVvystup3.Checked = false)
and (rdgZapIO.ItemIndex = 1)

   then BufferSizeC := 46;


  if (chcTVvystup1.Checked = false) and (chcTVvystup2.Checked = true) and (chcTVvystup3.Checked = true)
and (rdgZapIO.ItemIndex = 0)

   then BufferSizeC := 7;
```

```
 FT_HANDLE := ftHandleC;
 If (BufferSizeC - BufferoutC) <>0
 then
  begin
   BufferoutC := BufferSizeC;
   ftStatusC:= FT_Write(ftHandleC, @FT_Out_Buffer, BufferSizeC, @res );
   If ftStatusC = 0
    then
     begin
      if chcTVvystup1.Checked = true
       then chcTVvystup1.Color := clRed
       else chcTVvystup1.Color := clMaroon;
      if chcTVvystup2.Checked = true
       then chcTVvystup2.Color := clRed
       else chcTVvystup2.Color := clMaroon;
      if chcTVvystup3.Checked = true
       then chcTVvystup3.Color := clRed
       else chcTVvystup3.Color := clMaroon;
     end
    else MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
    end;
//Kontrola nastavení vstupů
FT_HANDLE := ftHandleC;
Get_USB_Device_BitMode(FT_In_BufferC);  //Sejmuní aktuální hodnoty pinů na portu C
if (FT_In_BufferC - BufferinC) <> 0
 then
  begin
   if (FT_In_BufferC or 191) = 191         //Test, zda-li je příslušný bit nastaven
    then lblIOVstup1.Color := clMaroon      //Nastavení podbarvení vstupů, které značí aktivaci vstupu
    else lblIOVstup1.Color := clRed;
```

```
   if (FT_In_BufferC or 127) = 127
    then lblIOVstup2.Color := clMaroon
    else lblIOVstup2.Color := clRed;
   BufferinC := FT_In_BufferC;
  end;
//Kontrola vstupů pro port D
FT_HANDLE := ftHandleD;
Get_USB_Device_BitMode(FT_In_BufferD);
if (FT_In_BufferD - BufferinD) <> 0
 then
  begin
   if (FT_In_BufferD or 253) = 253
    then lblIOVstup4.Color := clMaroon
    else lblIOVstup4.Color := clRed;
   if (FT_In_BufferD or 254) = 254
    then lblIOVstup3.Color := clMaroon
    else lblIOVstup3.Color := clRed;
   if (FT_In_BufferD or 251) = 251
    then lblTVVstup.Color := clMaroon
    else lblTVVstup.Color := clRed;
   BufferinD := FT_In_BufferD;
  end;

if rdgKanal.ItemIndex <>  LastItemIndex
 then
  begin
   if rdgKanal.ItemIndex = 0
    then
     begin
      ftStatusA:= I2C_OpenChannel(0, @ftHandleA);        //Otevření kanálu
```

```
    if ftStatusA <> 0
     then MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
    ChannelConfig[0] := 400000;
    ChannelConfig[1] := 255;
    ftStatusA := I2C_InitChannel(ftHandleA, @ChannelConfig);      //Inicializace
    if ftStatusA <> 0
     then MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
    ValueMUXControlRegister :=  4;
    ftStatusA := I2C_DeviceWrite(ftHandleA, 112, 1, @ValueMUXControlRegister, @bytesToTransfered, 3 );
//Zápis, kam se bude zapisovat/číst
    if ftStatusA <> 0
     then MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
    LastItemIndex := 0;
    I2C_CloseChannel(ftHandleA);
  end;

  if rdgKanal.ItemIndex = 1
   then
    begin
    ftStatusA:= I2C_OpenChannel(0, @ftHandleA);        //Otevření kanálu
    if ftStatusA <> 0
     then MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
    ChannelConfig[0] := 400000;
    ChannelConfig[1] := 255;
    ftStatusA := I2C_InitChannel(ftHandleA, @ChannelConfig);      //Inicializace
    if ftStatusA <> 0
     then MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
    ValueMUXControlRegister :=  5;
    ftStatusA := I2C_DeviceWrite(ftHandleA, 112, 1, @ValueMUXControlRegister, @bytesToTransfered, 3 );
//Zápis, kam se bude zapisovat/číst
    if ftStatusA <> 0
```

```
      then MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
     LastItemIndex := 1;
     I2C_CloseChannel(ftHandleA);
   end;

   if rdgKanal.ItemIndex = 2
    then
     begin
     ftStatusA:= I2C_OpenChannel(0, @ftHandleA);        //Otevření kanálu
     if ftStatusA <> 0
      then MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
     ChannelConfig[0] := 400000;
     ChannelConfig[1] := 255;
     ftStatusA := I2C_InitChannel(ftHandleA, @ChannelConfig);      //Inicializace
     if ftStatusA <> 0
      then MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
     ValueMUXControlRegister :=  6;
     ftStatusA := I2C_DeviceWrite(ftHandleA, 112, 1, @ValueMUXControlRegister, @bytesToTransfered, 3 );
//Zápis, kam se bude zapisovat/číst
     if ftStatusA <> 0
      then MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
     LastItemIndex := 2;
     I2C_CloseChannel(ftHandleA);
   end;

   if rdgKanal.ItemIndex = 3
    then
     begin
     ftStatusA:= I2C_OpenChannel(0, @ftHandleA);        //Otevření kanálu
     if ftStatusA <> 0
      then MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
```

```
      ChannelConfig[0] := 400000;
      ChannelConfig[1] := 255;
      ftStatusA := I2C_InitChannel(ftHandleA, @ChannelConfig);      //Inicializace
      if ftStatusA <> 0
       then MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
      ValueMUXControlRegister :=  7;
      ftStatusA := I2C_DeviceWrite(ftHandleA, 112, 1, @ValueMUXControlRegister, @bytesToTransfered, 3 );
//Zápis, kam se bude zapisovat/číst
      if ftStatusA <> 0
       then MessageDlg('Chyba komunikace', mtError, [mbOk], 0);
      LastItemIndex := 3;
      I2C_CloseChannel(ftHandleA);
    end;
  end;
 end;


end;


procedure TForm1.btnOdeslatSPIClick(Sender: TObject);
var address: byte;
    dataout: Dword;
    datain: Dword;
    ChannelConfig : ARRAY[0..256] of Dword;
    Out_Buff : Array[0..63] of byte;
    In_Buff : Array[0..63] of byte;
    i : integer;
    j : integer;
begin
//Data pro konfiguraci kanálu SPI
ChannelConfig[0] := 1;     //Frekvence
ChannelConfig[1] := 255;   //Latency čas
```

```
ftStatusB := SPI_OpenChannel(1, @ftHandleB);     //Otvírání kanálu
if ftStatusB = 0
 then Memo1.Lines.Add('Kanál SPI otevřen')
 else Memo1.Lines.Add('Chyba otvírání SPI: ' + HexWrdToStr(ftStatusB));   //Výpis chyby při neotevření
Memo1.Lines.Add('Handle SPI kanálu: ' + HexWrdToStr(ftHandleB));

ftStatusB := SPI_InitChannel(ftHandleB, @ChannelConfig);    //Inicializace kanálu
if ftStatusB = 0
 then Memo1.Lines.Add('Inicializace úspěšná')
 else Memo1.Lines.Add('Inicializace neúspěšná, kód chyby: ' + HexWrdToStr(ftStatusB));

address:= strtoint(edAdresaSPI.Text);
dataout:= strtoint(edDataSPI.Text);
datain:= $00;

if chcCistSPI.Checked = false
 then
  begin
   //Zápis do paměti
   Out_Buff[0] := $9F;
   Out_Buff[1] := $FF;
   ftStatusB := SPI_Write(ftHandleB, @Out_Buff, 11, @res, 7 );     //Aktivace CS
   if ftStatusB = 0
    then Memo1.Lines.Add('Data SPI zapsána')
    else Memo1.Lines.Add('Chyba SPI zápisu: ' + HexWrdToStr(ftStatusB));

   i := $A0;
   i := i or ((address div 8) AND $0F);
   Out_Buff[0] := i;
```

```
ftStatusB := SPI_Write(ftHandleB, @Out_Buff, 1, @res, 2 );      //Zápis adresy + příkazu pro paměť
if ftStatusB = 0
 then Memo1.Lines.Add('Data SPI zapsána')
 else Memo1.Lines.Add('Chyba SPI zápisu: ' + HexWrdToStr(ftStatusB));

Out_Buff[0] := (address and $07) div 8;

ftStatusB := SPI_Write(ftHandleB, @Out_Buff, 3, @res, 1 ); //Zápis posledních 3 bitů
 if ftStatusB = 0
 then Memo1.Lines.Add('Data SPI zapsána')
 else Memo1.Lines.Add('Chyba SPI zápisu: ' + HexWrdToStr(ftStatusB));

Out_Buff[0] := dataout and $FF;
Out_Buff[1] := (dataout and $FF00) div 8;
dataout := (In_Buff[1] * 256) or In_Buff[0];

ftStatusB := SPI_Write(ftHandleB, @dataout, 2, @res, 4 );        //Zápis dat
 if ftStatusB = 0
  then Memo1.Lines.Add('Data SPI zapsána')
  else Memo1.Lines.Add('Chyba SPI zápisu: ' + HexWrdToStr(ftStatusB));
end
else
begin
//Čtení z paměti
i := $c0;
i := i or ((address div 8) AND $0F);
Out_Buff[0] := i;

ftStatusB := SPI_Write(ftHandleB, @Out_Buff, 1, @res, 2 );      //Zápis instruční sady pro paměť
```

```
  if ftStatusB = 0
   then Memo1.Lines.Add('Data SPI zapsána')
   else Memo1.Lines.Add('Chyba SPI zápisu: ' + HexWrdToStr(ftStatusB));

  Out_Buff[0] := (address and $07) div 8;

  ftStatusB := SPI_Write(ftHandleB, @Out_Buff, 4, @res, 1 );  //Zápis adresy
  if ftStatusB = 0
   then Memo1.Lines.Add('Data SPI zapsána')
   else Memo1.Lines.Add('Chyba SPI zápisu: ' + HexWrdToStr(ftStatusB));

  ftStatusB := SPI_Read(ftHandleB, @In_Buff, 2, @res, 4 );    //Čtení z paměti
   datain := (In_Buff[1] * 256) or In_Buff[0];
   if ftStatusB = 0
    then Memo1.Lines.Add('Data SPI přečtena: ' + HexWrdToStr(datain))
    else Memo1.Lines.Add('Chyba SPI čtení: ' + HexWrdToStr(ftStatusB));
  end;

SPI_CloseChannel(ftHandleB);
end;

procedure TForm1.btnOdeslatI2CClick(Sender: TObject);
var bytesToTransfer : Dword;
   bytesToTransfered : Dword;
   buffer : ARRAY[0..256] of word;
   deviceAddress : integer;
   ChannelConfig : ARRAY[0..256] of Dword;
   data : integer;
   data1 : integer;
   data2 : integer;
```

```
begin
ftStatusA:= I2C_OpenChannel(0, @ftHandleA);        //Otevření kanálu
if ftStatusA = 0
 then Memo1.Lines.Add('Port otevřen')
 else Memo1.Lines.Add('Port neotevřen, kód chyby: ' + HexWrdToStr(ftStatusA));

FT_Handle := ftHandleA;
data1 := 0;
data2 := 0;
bytesToTransfer := 1;
bytesToTransfered := 0;
deviceAddress := strtoint(edAdresaI2C.Text);
data1 := strtoint(edData1I2C.Text);
data2 := strtoint(edData2I2C.Text);
ChannelConfig[0] := 400000;        //Frekvence
ChannelConfig[1] := 255;

ftStatusA := I2C_InitChannel(ftHandleA, @ChannelConfig);      //Inicializace
if ftStatusA = 0
 then Memo1.Lines.Add('Inicializace úspěšná')
 else Memo1.Lines.Add('Inicializace neúspěšná, kód chyby: ' + HexWrdToStr(ftStatusA));
ftStatusA := FT_ResetDevice(ftHandleA);

ftStatusA := I2C_DeviceWrite(ftHandleA, deviceAddress, 1, @data1, @bytesToTransfered, 1 );      //Zápis, kam
se bude zapisovat/číst
   if ftStatusA = 0
    then Memo1.Lines.Add('Zápis úspěšný')
    else Memo1.Lines.Add('Zápis neúspěšný, kód chyby: ' + HexWrdToStr(ftStatusA));

if chcCistI2C.Checked = true
```

```
 then

  begin

   ftStatusA := I2C_DeviceRead(ftHandleA, deviceAddress, 1, @data, @bytesToTransfered, 15 );     //V případě čtení rovnou bez nutnosti předchozího zápisu adresy

    if ftStatusA = 0

     then Memo1.Lines.Add('Čtení úspěšné, obdržená data: ' + HexWrdToStr(data1) + ', Přenesených bytů: ' + HexWrdToStr(bytesToTransfered))

     else Memo1.Lines.Add('Čtení neúspěšné, kód chyby: ' + HexWrdToStr(ftStatusA));

  end

 else

  begin

   ftStatusA := I2C_DeviceWrite(ftHandleA, deviceAddress, 1, @data2, @bytesToTransfered, 1 );

   if ftStatusA = 0

    then Memo1.Lines.Add('Zápis úspěšný')

    else Memo1.Lines.Add('Zápis neúspěšný, kód chyby: ' + HexWrdToStr(ftStatusA));

  end;


I2C_CloseChannel(ftHandleA);

end;


end.
```

Příloha F – Obsah přiloženého CD

Diplomová práce: Rozhraní pro diagnostiku TV přijímačů

Datasheety k použitým součástkám

Manuály k převodníky FT4232H

Knihovny příkazů k převodníku FT4232H

Unity pro Delphi k převodníku FT4232H

Ovladače pro operační systém Windows k převodníku FT4232H

Program FT Prog pro naprogramování paměti převodníku

Projekt návrhu v programu Altium Designer

Schéma zapojení

Deska plošných spojů

Osazení součástek

Seznam součástek

Seznam materiálu

Náčrtek samolepky

Náčrtek úpravy krabičky

Projekt ovládací aplikace v programu Delphi