VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

IMPLEMENTACE PROTOKOLU CAN PRO FITKIT

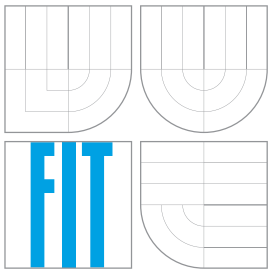
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ JANČO

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

IMPLEMENTACE PROTOKOLU CAN PRO FITKIT

FITKIT CAN IMPLEMENTATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ JANČO

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Dr. Ing. PETR HANÁČEK

BRNO 2014

Abstrakt

Tato bakalářská práce rozebíra principy komunikace na sběrnici CAN a návrh a implementaci řadiče této sběrnice. Řadič je implementovaný v jazyze VHDL pro školní vývojovou platformu FITKit. Dále práce popisuje návrh obvodů fyzické vrstvy pro připojení FITKit-u na sběrnici.

Abstract

This thesis describes main principles of communication on CAN bus, design and implementation of CAN bus controller. The controller is implemented in VHDL for school development platform FITkit. This work also describes design of CAN physical layer circuits for connecting FITKit to CAN bus.

Klíčová slova

CAN sběrnice, řadič, programovatelné hradlové pole, VHDL, FITkit

Keywords

CAN bus, controller, field-programmable gate array, VHDL, FITkit

Citace

Tomáš Jančo: Implementace protokolu CAN pro FITkit, bakalářská práce, Brno, FIT VUT v Brně, 2014

Implementace protokolu CAN pro FITkit

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Dr. Ing. Petra Hanáčka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Jančo
15. mája 2014

Obsah

Kapitola 1

Úvod

Napriek tomu, že protokol CAN bol predstavený už v roku 1986, podľa *CAN in Automation e.V.* [1], jeho robustnosť a univerzálnosť zabezpečuje neustály rast počtu zariadení komunikujúcich na zberniciach CAN.

Jedná sa o protokol komunikácie viacerých zariadení na jednoduchej sériovej zbernici. Medzi výhody patrí nízka latencia, distribuované prideľovanie zbernice bez nutnosti centrálneho arbitra a robustná detekcia chýb prenosu. Princípy a využitie tohto protokolu popisujem v kapitole 2.

Táto správa dokumentuje postup od návrhu po implementáciu radiča zbernice CAN. Implementácia vychádza zo špecifikácie Roberta Boscha *CAN Specification Version 2.0* [3] a je podrobne popísaná v kapitole 7. Pripojenie ku zbernici je realizované podľa štandardu ISO-11898 s využitím integrovaného budiča zbernice firmy Microchip MCP2551. Samotný radič je implementovaný v programovateľnom hradlovom poli (FPGA) školskej vývojovej dosky FITkit. S prvkami tejto vývojovej dosky zoznámim čitateľa v kapitole 3. Návrh rozhrania fyzickej vrstvy medzi FITkit-om a zbernicou CAN popisujem v kapitole 6.

1.1 Motivácia

Vývojová doska FITkit je pre študentov našej fakulty jedným z prostriedkov na zoznámenie sa s technológiami vstavaných zariadení. Existujú pre ňu rozširujúce komunikačné moduly na drôtovú komunikáciu na sieti typu Ethernet a bezdrôtovú komunikáciu v pásme 2.4 GHz [14].

Ďalší modul pripojiteľný k FITkit-u môže slúžiť študentom na experimentovanie s technológiami používanými v praxi. Príkladom môže byť diagnostika osobných automobilov, ktorá medzi inými využíva aj protokol CAN.

1.2 Ciele

Ciele tejto práce vychádzajú zo zadania bakalárskej práce a sú obohatené o body vyplývajúce z motivácie. Plnenie nasledujúcich cieľov je postupne popísané v jednotlivých kapitolách tejto práce.

- Zoznámiť čitateľa s fungovaním zbernice CAN.
- Priblížiť prvky dostupné na vývojovej doske FITkit.

- Preskúmať fungovanie dostupných implementácií protokolu CAN a zhodnotiť rôzne prístupy vzhľadom na možnosti poskytované dostupnou vývojovou doskou.
- Navrhnúť štruktúru a princíp fungovania radiča CAN pre FITkit, vhodne rozdeliť funkcie radiča medzi procesor a FPGA na vývojovej doske.
- Preskúmať možnosti realizácie pripojenia FITkit-u k CAN zbernici na úrovni fyzickej vrstvy. Zvoliť vhodný prístup a navrhnúť spôsob tohto pripojenia.
- Implementovať radič zbernice tak, aby bol schopný komunikovať za štandardom udávaných podmienok s inými zariadeniami na zbernici.
- Zhotoviť dosku plošného spoja a obvod fyzickej vrstvy pre pripojenie FITkit-u k zbernici.
- Otestovať funkčnosť implementácie.
- Vytvoriť jednoduchú demonštračnú aplikáciu komunikácie dvoch alebo viacerých zariadení na zbernici CAN.

Kapitola 2

Zbernica CAN

2.1 Použitie

Podľa W.Vossa [15] nachádza CAN uplatnenie v širokej škále priemyselných a spotrebných odvetví. Okrem automobilového priemyslu popisuje jeho výhody pri nasadení v letectve, námorníctve, vesmírnom výskume, automatizácií v priemyselnej výrobe, riadení výťahov a automatizácií v budovách.

2.2 Komunikačný protokol

Komunikačný protokol CAN umožňuje komunikáciu teoreticky neobmedzeného počtu rovnocenných uzlov. Praktické obmedzenia vyplývajú z fyzických obmedzení spoja (impedancia zbernice, oneskorenie).

Na rozdiel od iných zberníc, v CAN nie sú adresované jednotlivé zariadenia, ale správy sú rozlišované pomocou jedinečného identifikátora. Pre základný formát správy je identifikátor 11-bitový, umožňuje teda rozlíšiť $2^{11} = 2048$ rôznych správ. Pre rozšírený formát správy je dĺžka identifikátora 29 bitov (tzn. $2^{29} = 536870912$ rôznych správ). Rozšírený formát a jeho rozdiely oproti štandardnému formátu sú podrobnejšie rozobrané v časti 2.2.7 tejto kapitoly.

2.2.1 Vrstvový model

Systém používajúci zbernicu CAN môžeme rozdeliť podľa ISO/OSI [8] modelu do 7 vrstiev. Podľa špecifikácie CAN je linková vrstva (anglicky *Data Link Layer*) rozdelená na prenosovú (angl. *Transfer*) a objektovú vrstvu. Vyššie vrstvy špecifikácia nedefinuje, existujú však implementácie protokolov nad CAN, ktoré reprezentujú tieto vrstvy.

Fyzická vrstva CAN má viacero používaných štandardov. Tie sa líšia fyzickou realizáciou spoja medzi zariadeniami. Najčastejšie sa však jedná o dvojvodičový diferenciálny spoj [2]. Tento spoj je vždy obojsmerný a spoločný pre všetky zariadenia.

Prenosová vrstva má na starosti časovanie, vytváranie rámcov zo správ, arbitráciu, potvrdzovanie príjmu a ošetrovanie chýb. Objektová vrstva prevádza rámce z prenosovej vrstvy na správy (a naopak). Taktiež zodpovedá za filtrovanie správ. Bez použitia protokolov vyšších vrstiev zariadenie komunikujúce cez CAN pracuje priamo s CAN objektmi (správami).

<table>
<tr><th>CAN</th><th>ISO/OSI</th></tr>
<tr><td>Aplikačná</td><td>Aplikačná</td></tr>
<tr><td rowspan="4"></td><td>Prezentačná</td></tr>
<tr><td>Relačná</td></tr>
<tr><td>Transportná</td></tr>
<tr><td>Sieťová</td></tr>
<tr><td>Objektová</td><td rowspan="2">Linková</td></tr>
<tr><td>Prenosová</td></tr>
<tr><td>Fyzická</td><td>Fyzická</td></tr>
</table>

Obr. 2.1: Rozdelenie protokolu CAN do vrstiev ISO/OSI

2.2.2 Prenos správ

Správy môžu mať premenlivú dĺžku 0 až 8 bytov a sú identifikované jednoznačným identifikátorom `Message ID`. Správy sú kódované do rámcov s pevnou štruktúrou. Jednou z vlastností CAN je možnosť vyžiadania správy pomocou zaslania špeciálneho rámca, tzv. Remote Frame. Táto technika je známa ako dotazovanie (angl. polling). Arbitrácia (rozhodovanie o pridelení zbernice) je distribuovaná: priorita správy je odvodená od hodnoty `Message ID`. Technika arbitrácie je podrobnejšie rozobraná v podkapitole 2.2.3.

Štruktúra rámca

Rámec CAN začína poľom „Začiatok rámca“ (Start of Frame). Nasleduje pole arbitrácie, v ktorom sa prenesie identifikátor `Message ID` a príznak `RTR`[1]. Po dokončení odosielania tohto poľa je ukončená arbitrácia a na zbernicu zapisuje jediné zariadenie. Nasleduje riadiace pole (Control Field), ktoré určuje dĺžku prenášanej správy. V rámci tohto poľa je prenesený aj príznak rozšíreného formátu správy (viď 2.2.7). Za poľom Control Field nasleduje samotné telo správy a kontrolný súčet CRC. Po ich odoslaní nasleduje krátka pauza (dĺžka pauzy je zhodná s dĺžkou odosielania jedného bitu) a potvrdzovacie pole `ACK`. Krátka pauza slúži na to, aby všetky zariadenia na zbernici správu prijali a overili jej platnosť. Do poľa `ACK` potvrdí každý príjemca správy jej platnosť. Takto je možné rozlíšiť lokálne a globálne chyby prenosu. Po potvrdzovacom poli nasleduje pole konca rámca (EOF - End Of Frame) a medzirámcová pauza (IFS - Interframe Space). Tieto slúžia na korektné ukončenie prenosu a je možné počas nich indikovať chybu zaslaním chybového rámca (viď 2.2.5).

Kódovanie

Na úrovni fyzickej vrstvy sú prenášané informácie kódované do dvojúrovňového signálu, ktorý predstavuje takzvaný dominantný a recesívny bit. Ak ľubovoľné zariadenie na zbernici odošle dominantný bit, presadí tento stav zbernice a všetky zariadenia prijímajú dominantný bit zo zbernice. Ak zariadenie vysiela recesívny bit, môže prijímať recesívny bit (ak žiadne zariadenie nevysiela dominantný bit) alebo dominantný bit (ak aspoň jedno zariadenie vysiela dominantný bit). Tento princíp nazývame aj montážny súčin [16].

Nezhoda medzi vysielaným a prijímaným bitom môže znamenať stratu arbitrácie, chybový stav alebo príjem niektorého zo špeciálnych rámcov.

[1] Remote Transmission Request - príznak vzdialeného rámca (žiadosť o zaslanie údajov)

Metóda bit stuffing

Zbernica CAN neobsahuje žiadne synchronizačné vodiče. Synchronizačný signál je odvodený z dátového vodiča na každom zariadení na zbernici. Časovanie je možné odvodiť len pri zmene stavu zbernice (z dominantného na recesívny alebo naopak), preto je dlhá postupnosť nemenného stavu zbernice nežiaduca.

Z tohto dôvodu sa do postupností po sebe idúcich rovnakých bitov vkladá bit inverznej polarity. V špecifikácií CAN je tento postup nazývaný aj `Bit Stuffing`. Najdlhšia povolená postupnosť rovnakej polarity je dlhá 5 bitov. Po odoslaní postupnosti piatich rovnakých bitov vysielajúce zariadenie vloží jeden inverzný doplnkový bit. Postupnosti dlhšie ako 5 bitov sú považované za chybu prenosu.

Táto metóda však nie je použitá na celý rozsah správy, iba na časť od poľa „Začiatok rámca“ (SOF) po CRC kontrolný súčet vrátane. Zvyšok správy má takú štruktúru, že nie je potrebné vkladať inverzné bity. Tu majú postupnosti viac ako 5 rovnakých bitov špeciálny význam (vloženie chybového rámca, vkladanie čakacích stavov, voľná zbernica). Ich význam bude vysvetlený v príslušných kapitolách.

2.2.3 Arbitrácia na zbernici

Z princípu dominantných a recesívnych stavov zbernice vyplýva možnosť súbežného zápisu na zbernicu viacerých zariadení, pričom každé zapisujúce zariadenie dokáže vyhodnotiť, či bol zápis úspešný. Z pohľadu zariadenia, ktoré zapisuje na zbernicu, môžu nastať tri prípady:

- Aspoň jedno zariadenie zapisuje na zbernicu dominantný bit D – stav zbernice je vždy dominantný.
- Zariadenie zapisuje na zbernicu recesívny bit r a stav zbernice je recesívny – odoslaná informácia má požadovanú hodnotu, ku kolízií nedochádza.
- Zariadenie zapisuje na zbernicu recesívny bit r a aspoň jedno zariadenie zapisuje dominantný bit D – stav zbernice bude dominantný. Z pohľadu zariadení zapisujúcich r došlo ku kolízií, zariadenia zapisujúce D kolíziu nedetegujú.

Na tomto princípe je postavená arbitrácia na zbernici CAN. Počas vysielania arbitračného poľa všetky zariadenia, ktoré chcú komunikovať počas tohto rámca postupne zapisujú na zbernicu identifikátor správy. Zariadenie, ktoré zistí nezhodu medzi vyslaným a prijatým bitom stráca arbitráciu a v komunikácií pokračujú len zariadenia, ktoré odoslali zhodný bit.

Z toho vyplýva, že najvyššiu prioritu bude mať správa s identifikátorom pozostávajúcim len z dominantných bitov. Naopak najnižšiu prioritu bude mať správa s identifikátorom z recesívnych bitov.

Po ukončení odosielania arbitračného poľa by na zbernicu malo zapisovať len jediné zariadenie. V opačnom prípade dôjde ku konfliktu, ktorý je z pohľadu CAN neriešiteľný a bude označený za chybu.

2.2.4 Kontrolný súčet CRC

Zbernica CAN je zabezpečená voči prenosovým chybám pomocou cyklického redundantného kontrolného súčtu CRC. Jeho hodnota je vypočítaná z polí rámca od SOF po dátové pole vrátane. Výpočet je možné opísať generujúcim polynómom:

$$X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$$

Hodnota CRC je prenesená v poli CRC rámca. Všetky zariadenia na zbernici overia tento kontrolný súčet. Ak je súčet platný pre danú správu, každé zariadenie odošle dominantný bit počas poľa ACK. Tak potvrdí vysielajúcemu zariadeniu platnosť odchádzajúcej správy. Vďaka princípu montážneho súčinu dokáže každé zariadenie zistiť, či správa bola správne doručená aspoň jednému zariadeniu. To umožňuje rozlišovať medzi globálnymi chybami (ani jedno zariadenie nedostalo správnu správu) a lokálnymi.

2.2.5 Ošetrenie chýb prenosu

Na zbernici CAN môžeme rozlíšiť tieto chybové stavy:

- Bitová chyba – nastáva, ak zariadenie zapisujúce na zbernicu zistí iný stav zbernice ako ten, ktorý na ňu zapísalo. Táto chyba však nenastáva ak sa tak stane počas arbitrácie alebo počas potvrdzovacieho poľa ACK.
- Chyba bit stuffingu – nastáva, ak zariadenie prijme 6 po sebe idúcich rovnakých bitov (len počas príjmu polí, na ktoré sa bit stuffing vzťahuje).
- Chyba CRC – nastáva, ak sa nezhoduje CRC vypočítaný príjemcom s hodnotou prijatou v poli CRC.
- Chyba formátu (angl. Form error) – nastáva, ak sa v poliach s pevne danou hodnotou nachádza neplatná hodnota.
- Chyba ACK – nastáva, ak zariadenie nezistí dominantný stav zbernice počas potvrdzovacieho poľa ACK.

Na detekciu chybového stavu zariadenie okamžite reaguje vyslaním chybového rámca. Ten pozostáva z 6 dominantných bitov. Odoslanie šiestich dominantných bitov porušuje pravidlo bit stuffingu, preto efektívne zneplatňuje správu práve posielanú na zbernici. Na tento stav zákonite reagujú ostatné zariadenia na zbernici a vysielajú vlastný chybový rámec. Po odoslaní 6 dominantných bitov zariadenie čaká na recesívny stav zbernice (zatiaľ odosiela recesívne bity - neovplyvňuje zbernicu). Keď sa stav zbernice zmení na recesívny, všetky zariadenia sú synchronizované a odošlú 7 recesívnych bitov. Tým je odosielanie chybového rámca ukončené.

Aby sa predišlo znefunkčneniu zbernice zariadením, ktoré neustále hlási chybu prenosu, zariadenia CAN majú tri stavy vzhľadom na ošetrovanie chybových stavov:

- Chybovo aktívne (angl. Error Active) – zariadenie reaguje na chybový stav zaslaním chybového rámca.
- Chybovo pasívne (angl. Error Pasive) – zariadenie na detekciu chyby reaguje „pasívnym chybovým rámcom“, ktorý pozostáva z 6 recesívnych bitov, teda neovplyvňuje stav zbernice. Po zotavení z poruchy môže opäť komunikovať na zbernici.
- Odpojené od zbernice (angl. bus-off) – zariadenie nezapisuje na zbernicu.

Stav zariadenia je odvodený od počtu chýb vysielania a chýb príjmu.

2.2.6 Časovanie zbernice

Keďže zbernica je asynchrónna, komunikujúce uzly musia dodržiavať časovanie pri vysielaní jednotlivých bitov.

Každý bit je odoslaný za dobu t_b, ktorá je daná rýchlosťou zbernice.

$$t_b = \frac{1}{BR}$$

kde BR je použitá symbolová rýchlosť (počet prenesených bitov za sekundu). Pre účely čítania zo zbernice je tento čas rozdelený na 4 segmenty:

1. Synchronizačný segment – slúži na synchronizáciu komunikujúcich zariadení. Počas neho by mala nastať zmena úrovne zbernice.
2. Propagačný segment – slúži na kompenzáciu oneskorenia spôsobeného konečnou rýchlosťou šírenia zmeny po zbernici a oneskoreniami logických prvkov na vstupe radičov zbernice.
3. Fázový segment 1 – slúži na kompenzáciu fázových posunov medzi zariadeniami. Tie sú spôsobené odchýlkami v rýchlosti systémových hodín jednotlivých zariadení.
4. Fázový segment 2

Prvý prechod zbernice z recesívneho na dominantný stav počas poľa začiatku rámca (SOF) slúži na tzv. tvrdú synchronizáciu. Pri tvrdej synchronizácií sa posunie stav prijímajúceho zariadenia tak, aby tento prechod ležal v synchronizačnom segmente.

Ostatné prechody sú použité na tzv. bitovú resynchronizáciu, kedy sú dĺžky fázových segmentov upravované (vždy sa jeden fázový segment skracuje a druhý predlžuje) tak, aby hrana nasledujúceho bitu bola posunutá do synchronizačného segmentu.

2.2.7 Rozšírený formát správy

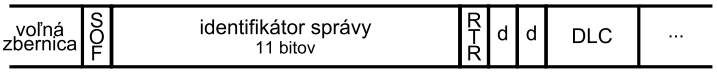
Rozšírenie protokolu CAN bolo špecifikované vo verzií CAN 2.0B[3] a prinieslo možnosť prenosu správ s 29-bitovým identifikátorom (namiesto pôvodných 11 bitov). Toto rozšírenie bolo štandardizované dodatkom ISO 11898.

Nový 29-bitový identifikátor správy sa skladá z dvoch častí: 11-bitového základného identifikátora a 18-bitového doplnku. Štruktúra rozšírenej správy je nasledovná: Pole `SOF` (začiatok rámca) a prvých 11 bitov identifikátora správy je zachovaných. V riadiacom poli sú v prípade rozšírenej správy prvé dva bity vždy recesívne. Prvý z nich by v prípade štandardného formátu označoval žiadosť o zaslanie vzdialeného rámca, v rozšírenom formáte je však vždy recesívny z dôvodu arbitrácie — správy s 11-bitovým identifikátorom sú prenášané prioritne. Druhý bol pôvodne rezervovaný (a odosielaný ako dominantný), teraz jeho recesívna úroveň značí rozšírený formát správy.

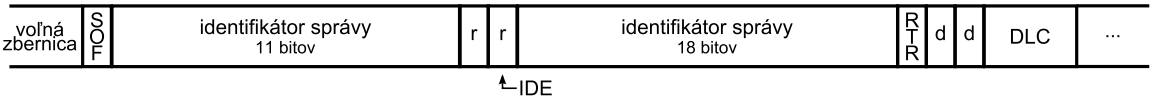
Ak je použitý rozšírený formát správy, po odoslaní dvoch bitov pôvodného riadiaceho poľa sa pokračuje odoslaním zvyšných 18 bitov identifikátora. Potom nasleduje riadiace pole zhodné s štandardným formátom a zvyšok rámca.

2.3 Fyzická vrstva CAN

Podľa dokumentov *CAN in Automation e.V.*[2], medzinárodného združenia užívateľov a výrobcov CAN zariadení sú najčastejšie používané štandardy fyzickej vrstvy CAN normy



(a) Štandardný formát správy



(b) Rozšírený formát správy

Obr. 2.2: Porovnanie formátov správ. `SOF` – Začiatok rámca (Start Of Frame), `RTR` – žiadosť o vzdialený rámec (Remote Transmission Request), `IDE` – príznak rozšíreného formátu (Identifier Extension), `DLC` – pole dĺžky správy (Data Length Code), `d` – dominantný bit, `r` – recesívny bit

ISO 11898-2, ISO 11898-3, SAE J2411 a ISO 11992. Prenosové médium môže byť elektrické alebo optické. V prípade elektrického média sa najčastejšie používa krútený pár vodičov vedúci diferenciálny signál.

Podľa ISO 11898-2 sa na signalizáciu recesívneho bitu používa rozdielové napätie −500 mV až 5 mV, na signalizáciu dominantného bitu 1.5 V až 3 V [10]. Použiteľné prenosové rýchlosti sú až do 1 Mbit/s, počet zariadení na zbernici je obmedzený iba elektrickými vlastnosťami použitých transcieverov. Topológia zbernice je vždy lineárna a vyžaduje ukončovacie rezistory o hodnote 120 Ω.

ISO 11898-3 štandardizuje rozhranie CAN odolné voči zlyhaniu (*fault-tolerant CAN*). Používa nižšie prenosové rýchlosti (najviac 125 kbit/s) a maximálny počet uzlov je 32. Dovoľuje však nelineárnu topológiu, v ktorej zlyhanie jednej prenosovej linky nemusí ovplyvniť ostatné zariadenia na zbernici.

Nie vždy je nutné využívať veľké prenosové rýchlosti a robustné vedenie zbernice, preto SAE J2411 špecifikuje fyzické médium s jedným dátovým vodičom (tzv. *single wire CAN*) — nepoužíva sa diferenciálna signalizácia. Prenosové rýchlosti sú najviac 33.3 kbit/s.

Posledný menovaný štandard, ISO 11992 definuje zbernicu CAN s nízkou prenosovou rýchlosťou a vysokou odolnosťou voči zlyhaniu s pripojením *point-to-point*.

Kapitola 3

Vývojová platforma FITkit

Študenti našej fakulty majú počas štúdia možnosť si zapožičať zariadenie FITkit[14], vývojovú dosku, pomocou ktorej sa môžu zoznámiť s technológiami používanými vo vstavaných zariadeniach. FITkit obsahuje mikrokontrolér Texas Instruments MSP430, programovateľné hradlové pole Xilinx FPGA Spartan 3. Okrem toho disponuje vysokým počtom vstupno/výstupných pinov, alfanumerickým LCD displejom, maticovou číselnou klávesnicou a rozhraniami známymi z osobných počítačov — port VGA, PS/2, audio vstup a výstup, sériové rozhranie RS232.

3.1 Mikrokontrolér MSP430

Na FITkit-e sa nachádza buď mikrokontrolér `MSP430F168IPM` alebo `MSP430F2617`, podľa verzie FITkit-u. Jedná sa o programovateľný mikrokontrolér s nízkou spotrebou, bohatý na periférie[12]. Procesorové jadro je 16-bitové typu RISC. Obe používané rady mikrokontroléru obsahujú časovače/čítače, analógovo-digitálne aj digitálne-analógové prevodníky, komunikačné rozhrania UART, SPI a I^2C. Väčšina univerzálnych výstupov kontroléru je vyvedených na hrebeňový konektor, teda plne dostupných užívateľovi. Rozhranie SPI je použité na komunikáciu s hradlový poľom popísaným nižšie.

3.2 FPGA Xilinx - programovateľné hradlové pole

Použité sú hradlové polia `XC3S50-4PQ208C` alebo `XC3S400-4PQ208C`[14]. Rada `XC3S50` obsahuje 50 tisíc logických hradiel, dve jednotky generátoru hodinového kmitočtu, pamäť RAM, dvojportové pamäte BRAM a 18-bitové násobičky. Vyššia rada `XC3S400` obsahuje 8-krát viac hradiel a väčší počet pamätí, násobičiek a hodinových jednotiek. Na FITkit-e tvorí toto FPGA rozhranie medzi mikrokontrolérom a ďalšími perifériami. 45 vstupno-výsupných pinov je vyvedených na port X FITkit-u, časť z nich je však zdieľaná s rozhraniami periférií.

3.3 Vývojové nástroje

Pre mikrokontrolér existuje prekladač jazykov C, C++ a asembleru `mspgcc`[1], voľne dostupný balík nástrojov pod licenciou GNU. Výstupom prekladu je binárny súbor, ktorý je potrebné nahrať do mikrokontroléru. Na FITkit-e sa to deje pomocou USB portu, nakoľko

[1] http://mspgcc.sourceforge.net/

FITkit obsahuje USB prevodník. Tento binárny súbor je kontroléru doručený sériovým rozhraním a nahraný do perzistentnej programovej pamäte samotným mikrokontrolérom (pomocou tzv. bootloader-u). Na nahrávanie programu do FITkit-u slúži nástroj `fkflash`.

Hradlové pole sa programuje vo vývojovom prostredí Xilinx ISE, v jazyku VHDL. Vývojové prostredie obsahuje nástroje na podporu návrhu, generovanie kódu, simuláciu a preklad (syntézu). Výsledný syntetizovaný kód je uložený v samostatnej pamäti FLASH a do hradlového poľa ho po každom resete nahráva mikrokontrolér.

Všetky spomínané nástroje je možné používať samostatne, alebo pomocou prekladového systému QDevKit, ktorý umožňuje organizáciu zdrojových súborov do projektov. Tie môžu byť pomocou grafického rozhrania preložené a nahrané do FITkit-u. QDevKit funguje na princípe generovania skriptov Makefile, ktoré sa vykonávajú pomocou interpretu `gmake`[2]. Ukážkové projekty sú dostupné cez voľne prístupný SVN repozitár.

[2] http://www.gnu.org/software/make/

Kapitola 4

Existujúce riešenia

V tejto kapitole popisujem existujúce implementácie protokolu CAN. Najčastejšie sa jedná o hardwarové radiče v podobe integrovaných obvodov, ale existujú aj softwarové implementácie a radiče, ktoré sú implementované ako vstavaná periféria v čipe mikrokontroléru.

Zo štruktúry a princípov použitých v existujúcich radičoch som čerpal pri návrhu implementácie radiča pre FITkit.

4.1 Hardwarové radiče

Obvody `MCP 2515` firmy Microchip a `SJA1000` firmy Philips sú bežne dostupné u predajcov elektronických súčiastok. V nasledujúcej časti podrobnejšie popisujem ich vlastnosti a vnútornú štruktúru.

4.1.1 Obvod MCP 2515

Tento obvod je integrovaným radičom CAN, ktorý implementuje prenosovú a objektovú vrstvu. Obvod pozostáva z viacerých blokov. Bloky plniace funkcie prenosovej vrstvy sú:

- Generátor hodín – slúži na vytvorenie hodinového signálu pre zvolenú symbolovú rýchlosť
- Časovacia logika – odvodzuje časovanie zo zmien stavu zbernice
- Vysielacia logika – zapisuje bity na zbernicu
- RX a TX posuvné registre – zabezpečujú (de-)serializáciu správy
- CRC modul a CRC komparátor – vykonáva výpočet kontrolného súčtu CRC a overenie CRC
- Konečný stavový automat protokolu – zapuzdruje celú logiku protokolu

Na úrovni objektovej vrstvy dochádza k filtrovaniu správ. Obvod disponuje dvoma registrami, do ktorých sa ukladajú správy po prechode filtrom. Filter je pre každý register samostatný a je definovaný maskou (tá určuje, ktoré bity správy sa použijú na filtrovanie) a pravidlami (tie určujú akceptované hodnoty zvolených bitov). Filtrovanie prebieha na základe zhody identifikátora správy `Message ID`.

Okrem týchto blokov obvod obsahuje registre na uloženie troch odchádzajúcich správ. Správy sú odosielané v poradí danom prioritami v riadiacom registri.

S nadradenou aplikáciou radič komunikuje pomocou rozhrania SPI.

4.1.2 Obvod SJA1000

Integrovaný radič `SJA1000` je podobný predchádzajúcemu v tom, že implementuje objektovú a prenosovú vrstvu CAN. Má však odlišnú štruktúru. Prijaté správy nie sú ukladané do registrov, ale do fronty. Tá je prístupná cez posuvné okno o veľkosti jednej správy (najviac 13 B). Register na uloženie správy pripravenej na odoslanie je len jeden.

Obvod komunikuje s nadradeným procesorom pomocou 8-bitovej zdieľanej zbernice. Oproti predchádzajúcemu obvodu poskytuje viac stavových informácií o prenose správ a chybách, prístupných cez stavové registre radiča.

4.2 Softwarové implementácie

4.2.1 MBCAN - CAN vo FPGA

Implementácia autorov Blagomira Doncheva a Marina Hristova [6] je ukážkou implementácie CAN pre FPGA v jazyku VHDL. Ich radič zodpovedá špecifikácií CAN verzie 2.0B, nepodporuje však prenos rámcov v rozšírenom formáte. Ich návrh sa skladá z dvoch blokov: *Protokolové jadro* a *Rozhranie*. Protokolové jadro je zložené z viacerých jednotiek:

- *Frame sequencer* - zabezpečuje (de-)serializáciu, (de-)kódovanie, monitorovanie zbernice a signalizuje stav ostatným jednotkám
- Logika CRC - počíta kontrolný súčet a overuje platnosť správy
- Bitová synchronizácia - zabezpečuje správne časovanie pri odosielaní bitov na zbernicu a správne vzorkovanie počas príjmu

Rozhranie slúži na komunikáciu s nadradenou aplikáciou. Využíva na to multiplexovanú paralelnú zbernicu. Ako rozhranie slúži dvojportová pamäť RAM, v ktorej sú uložené registre radiča

4.2.2 CAN Bus analyzér pre FPGA

Radek Holota vo svojej práci [7] popisuje analyzátor CAN vytvorený vo FPGA. Z hľadiska vrstvového modelu CAN implementuje len prijímaciu časť transportnej vrstvy. Jeho radič sa skladá z viacerých funkčných blokov, podobne ako ostatné radiče:

- `BRP` - delič hodinového signálu
- `BTL` - bitová synchronizácia
- `STUFF` - bit stuffing
- `ST_MACH` - stavový automat, ktorý riadi všetky ostatné bloky
- `STACK` - FIFO zásobník prijatých správ
- `COMM` - komunikačné rozhranie k počítaču

Na pripojenie k fyzickému médiu CAN zbernice používa obvod (transciever) `PCA82C250`. Komunikačné rozhranie slúži priamo na pripojenie osobného počítača pomocou paralelného portu.

4.2.3 TinyCAN - implementácia CAN nenáročná na zdroje

Autori implementácie TinyCAN [5] popisujú štruktúru radiča, ktorý je optimalizovaný vzhľadom na počet logických hradiel. Úsporu dosiahli čiastočne nedodržaním špecifikácie v prípade ošetrovania chybových stavov. Ich radič chyby prenosu deteguje, neoznamuje ich však odoslaním chybového rámca.

V ich návrhu sa nachádzajú klasické prvky:

- Delič hodinového signálu
- Procesor bitového toku (BSP[1]) - obsahuje stavový automat implementujúci celú protokolovú logiku, implementuje rozhranie k nadradenej aplikácií pomocou paralelnej zbernice. Komunikácia prebieha na úrovni čítania a zápisu do registrov.
- Blok bitového časovania - zabezpečuje synchronizáciu, synchrónne prenáša stav zbernice do BSP v okamihu vzorkovania, prenáša výstup z BSP na zbernicu v okamihu platnosti bitu.

4.3 Mikrokontroléry s radičom CAN

Na trhu sú bežne dostupné mikrokontroléry, ktoré obsahujú periférnu jednotku s radičom CAN. Výstupy radiča bývajú vyvedené na niektoré zo vstupno/výstupných pinov mikrokontroléru. Inicializácia a ovládanie radiča prebieha zhodne ako ovládanie ostatných periférií - pomocou registrov. Tie sú buď mapované do adresového priestoru pamäte, alebo na ich modifikáciu slúžia špecializované inštrukcie, podľa architektúry mikrokontroléru.

4.3.1 STM32F3

Mikrokontrolér `STM32F3` je výrobok firmy STMicroelectronics[2]. Je postavený na jadre 32-bitového procesora ARM Cortex-M4 a ponúka veľké množstvo periférnych jednotiek. Okrem iných, ponúka aj jednotku `bxCAN`, ktorá implementuje funkciu radiča CAN zbernice. Táto jednotka implementuje prenosovú a objektovú vrstvu CAN a ponúka: [11]

- Podporu odosielania a príjmu štandardných a rozšírených CAN rámcov (podľa CAN 2.0 A a B).
- Prenosovú rýchlosť až do 1 Mbit/s.
- Tri schránky na odosielané správy. Správy sú odosielané v poradí podľa nastaviteľnej priority.
- Prijaté správy sú ukladané do dvoch FIFO bufferov, každý má kapacitu na uloženie 3 správ.
- Správy sú filtrované pomocou 28 konfigurovateľných akceptačných filtrov. Tie umožňujú filtrovanie na základe identifikátora správy a príznakov z kontrolného poľa. Filter pracuje buď v maskovanom režime, kedy maska určuje porovnávané bity, alebo v režime zoznamu identifikátorov, kedy sa porovnáva celý identifikátor. Druhý režim umožňuje jedným filtrom porovnávať správu s dvojicou identifikátorov.

[1] BSP - Bit Stream Processor
[2] http://www.st.com/

- Podporu prerušení pri prijatí správy, dokončení odosielania, chybe a ďalších udalostiach.

Komunikácia s radičom prebieha pomocou sady registrov mapovaných do pamäte. Nájdeme v nej registre na prístup k schránkam správ, stavové a riadiace registre. Niektoré riadiace registre sú počas normálnej činnosti uzamknuté a ich modifikácia je možná iba v inicializačnom režime. Toto opatrenie zabraňuje nechcenej zmene napr. prenosovej rýchlosti, ktorá by mohla spôsobiť dočasné znemožnenie komunikácie ostatných zariadení na zbernici.

Výstup radiča je možné pomocou konfigurácie GPIO pripojiť na viacero vstupno/výstupných pinov mikrokontroléru. Tento výstup je nutné pripojiť k budiču zbernice.

Tento mikrokontrolér sa nachádza napríklad na vývojovej doske `STM32 F3 Discovery`, ktorá bola použitá pri ladení a testovaní implementácie môjho radiča.

4.3.2 Microchip PIC

Mikrokontroléry PIC spoločnosti Microchip Technology Inc. sú 8, 16 alebo 32-bitové kontroléry. Podľa zvolenej rady a typu obsahujú rôzne periférie a disponujú rôznym množstvom vstupno-výstupných pinov. Jedným z mikrokontrolérov 8-bitovej rodiny PIC18 je aj PIC18F2580. Ten disponuje perifériou CAN radiča s názvom `ECAN`. Podporovaný je CAN protokol vo verzii 2.0A a B. Implementovaná je objektová a prenosová vrstva. Na fyzickú zbernicu CAN sa obvod pripája pomocou vhodného obvodu fyzickej vrstvy (transcieveru). Podporované rýchlosti sú až do 1 Mbit/s. Organizácia radiča sa nelíši od obvodu Microchip MCP2515 opísaného v kapitole 4.1.1, radič však obsahuje vyšší počet dostupných bufferov a filtrov: až 6 bufferov je použiteľných na uloženie celej odchádzajúcej alebo prichádzajúcej správy a je dostupných až 16 akceptačných filtrov.

Komunikácia s aplikáciou v procesore prebieha pomocou riadiacich a dátových registrov mapovaných do pamäte podobne, ako v prípade periférie `bxCAN` z predchádzajúcej podkapitoly. Periféria podporuje prerušenie procesora pri dokončení odosielania / prijímania správy ako aj pri vzniku chybového stavu.

Radič má niekoľko režimov: konfiguračný režim, v ktorom je možné nastaviť kritické parametre prenosu (rýchlosť, akceptačné filtre, ...), režim spánku, režim normálneho prenosu, režim pasívneho načúvania, kedy radič nezasahuje do komunikácie na zbernici ani v prípade chyby a režim spätnej slučky (tzv. *loopback*). Zaujímavosťou je režim rozoznávania chýb, kedy prijaté správy obsahujúce chybu nie sú zahodené, ale sú uložené do príjmových registrov a môžu byť použité na odhalenie problému komunikácie.

Okrem týchto režimov radič podporuje viacero funkčných úrovní (najmä z dôvodu spätnej kompatibility so staršími implementáciami tejto periférie). V základnom režime sú dostupné funkcie takmer zhodné so spomínaným hardwarovým radičom MCP2515. Vo vyšších režimoch pribúdajú dostupné registre na uloženie správ a akceptačných filtrov. Taktiež pribudne možnosť filtrovania správ na základe obsahu dátovej časti. Toto má umožniť jednoduchšiu implementáciu protokolov vyšších vrstiev nad CAN.

Kapitola 5

Návrh radiča

Pri návrhu radiča som postupoval metódou zhora nadol, teda dekompozíciou problému na čiastkové podproblémy.

5.1 Dekompozícia problému

Z popisu protokolu CAN vyplývajú tieto požiadavky:

- Radič musí obsahovať objektovú vrstvu, ktorá bude zabezpečovať filtrovanie správ. Počas jej návrhu je potrebné určiť:
  - Spôsob špecifikácie filtrov správ
  - Spôsob aplikácie filtrov na tok správ a blok zodpovedný za túto činnosť
- Radič musí obsahovať prenosovú vrstvu, zodpovednú za (de-)kódovanie správ, ich príjem a odosielanie. Pre jej fungovanie treba navrhnúť:
  - Spôsob obaľovania správy do rámcov – blok logiky prenosovej vrstvy
  - Overovanie správ CRC kontrolným súčtom – generátor CRC a komparátor
  - Kódovanie správ na postupnosť bitov a naopak – serializér a deserializér, blok „bit stuffing-u“
  - Dodržiavanie časovania správ – zdroj hodinového signálu, synchronizačné obvody
  - Vzorkovanie stavu zbernice

5.2 Popis prvkov radiča

V tejto časti práce sú podrobnejšie špecifikované úlohy prvkov, u ktorých potreba existencie vyplynula z dekompozície problému. Spôsob implementácie je opísaný v kapitole 7. Bloková schéma zobrazujúca prepojenie prvkov radiča sa nachádza v prílohe B.

5.2.1 Komunikačný modul

Na najvyššej úrovni z pohľadu vrstvového modelu CAN bude modul zabezpečujúci komunikáciu s aplikačnou vrstvou. Jeho úlohou bude nastavovať parametre podriadených modulov (nastavenie filtrov, časovania) a prenášať správy prijaté z CAN do aplikačnej vrstvy a naopak. Pre tento účel bude použitá sada registrov:

- Dátové registre - budú obsahovať prijaté alebo odoslané správy.
- Riadiace registre - budú obsahovať konfiguračné slová, ktoré budú nastavovať parametre prenosu: prenosovú rýchlosť, dĺžky časových segmentov, nastavenie prerušení, voľbu režimov radiča a iné.
- Stavové registre - ich obsah bude nastavovaný samotným radičom a bude reflektovať stav radiča pomocou príznakov: príznaky chyby, prijatej správy, odosielania správy a iné.

Táto sada registrov bude dostupná pomocou rozhrania SPI medzi FPGA a mikrokontrolérom FITkit-u.

5.2.2 Filtrovanie správ

Modul filtrovania správ porovnáva prijatú správu s akceptačným filtrom. V prípade zhody ju pripraví na prenos pomocou komunikačného modulu. V opačnom prípade sa správa zahodí a nebude ďalej spracovávaná. Akceptačný filter sa skladá z dvoch hodnôt: identifikátora a masky. Maska určuje, ktoré bity identifikátora budú porovnávané na zhodu. Každý filter bude mať priradené zodpovedajúce registre, v ktorých bude uložený identifikátor a filtrovacia maska.

5.2.3 Logika prenosovej vrstvy

Tento blok obsahuje stavový automat, ktorého stavy zodpovedajú jednotlivým poliam rámca CAN. Stavový automat takisto zabezpečuje arbitráciu počas arbitračnej fázy a prechod do chybového stavu v prípadoch opísaných v 2.2.5. Výstupom je sada riadiacich signálov pre podriadené bloky, najmä povoľovacie signály na zápis na zbernicu, signály určujúce typ synchronizácie, aplikácie metódy bit stuffing a signály označujúce aktuálnu fázu prenosu resp. voľnú zbernicu.

5.2.4 Kontrolný súčet CRC

V tomto bloku je vypočítaná hodnota kontrolného súčtu na základe sekvenčného vstupu odosielanej správy. Výstupom je táto hodnota CRC. Ďalší blok zabezpečuje komparáciu (porovnanie hodnoty) očakávaného CRC a prijatej hodnoty.

5.2.5 Zdroj hodinového signálu

Blok hodinového signálu obsahuje preddeličku systémového hodinového signálu, ktorá je nastaviteľná na rôzne frekvencie, zodpovedajúce používaným symbolovým rýchlostiam CAN.

5.2.6 Synchronizačné obvody

Synchronizačné obvody zabezpečujú bitovú synchronizáciu pri čítaní zo zbernice. Vstupom je aktuálny stav zbernice, výstupom sú impulzy označujúce okamih platnosti dát, kedy je stav zbernice navzorkovaný. Tieto obvody sú riadené riadiacimi registrami, ktoré určujú dĺžky jednotlivých časových úsekov (ako bolo vysvetlené v časti 2.2.6). Keďže synchronizácia prebieha iba pri príjme, obvod musí obsahovať aj povoľovací vstupný signál.

5.2.7 Vzorkovanie stavu zbernice

Obvod vzorkovania je prvým obvodom na vstupe zo zbernice. Zabezpečuje prečítanie stavu zbernice v okamihu určenom špecifikáciou CAN (tzv. sample point).

5.2.8 Bit stuffing

Obvod bit stuffing-u označuje, ktorý bit v priebehu príjmu správy je doplnkový a vyhodnocuje chybu bit stuffing-u. Pri odosielaní správy označuje miesto vloženia doplnkového bitu a určuje jeho hodnotu. Keďže bit stuffing sa aplikuje iba v niektorých častiach CAN rámca, obvod obsahuje vstupný povoľovací signál.

5.2.9 Radič prerušení

Okrem komunikácie cez rozhranie SPI by mal radič CAN obsahovať možnosť vyvolať prerušenie MCU. Udalosti, pri ktorých bude prerušenie vyvolané budú: príjem správy, dokončenie odosielania správy, vznik chyby. Pre tento účel bude implementácia obsahovať radič prerušení, ktorý pri vyvolaní niektorej z týchto udalostí vyvolá prerušenie MCU. Po prečítaní stavovej slabiky sa príznak prerušenia deaktivuje do doby vzniku novej udalosti.

Kapitola 6

Návrh obvodov fyzickej vrstvy

6.1 Cieľové vlastnosti

Obvod fyzickej vrstvy by mal umožňovať:

- Jednoduché pripojenie k FITkit-u
- Prevod medzi signálmi s úrovňou TTL 3.3 V a diferenciálnym signálom zbernice CAN
- Voliteľná možnosť napájania FITkit-u z napájacích vodičov vedených so zbernicou CAN
- Možnosť vloženia terminačných rezistorov (impedančné prispôsobenie) do linky CAN
- Štandardné rozhranie na pripojenie ku CAN zbernici

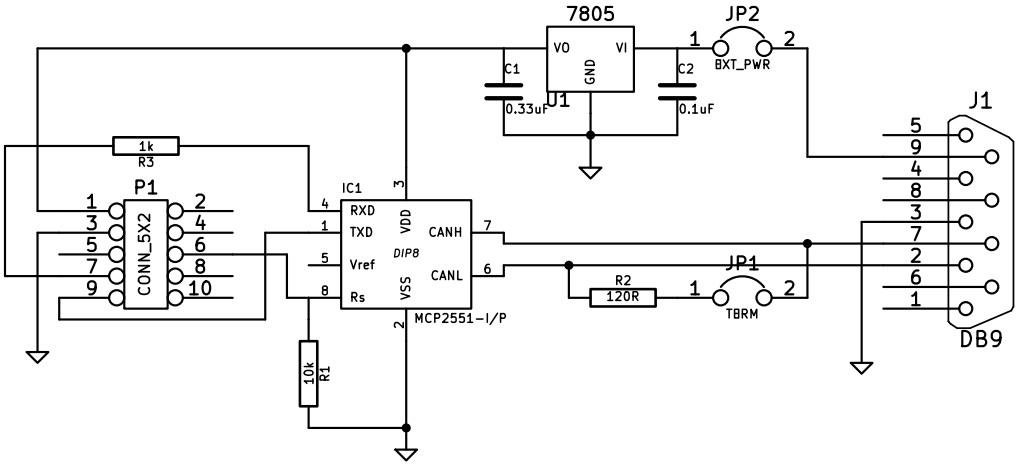
6.2 Budiče zbernice

Z dostupných integrovaných budičov zbernice bol zvolený obvod `MCP2551` firmy Microchip, ktorý vykonáva prevod TTL na diferenciálny signál CAN podľa štandardu ISO-11898 a umožňuje rýchlosť komunikácie až do 1 Mb/s.

6.3 Obvod

Srdcom obvodu je spomínaný integrovaný budič zbernice `MCP2551`. Obvod ďalej obsahuje konektor `DB-9` zapojený podľa odporúčania CiA DS-102 [4]. Tento konektor slúži na pripojenie k fyzickému médiu CAN. Na druhej strane obvodu je konektor na pripojenie FITkit-u. Poradie vývodov bolo zvolené tak, aby sa dal pripojiť plochým vodičom s 10-pinovým konektorom na krajnú, ľavú, dolnú pozíciu rozhrania X na FITkit-e. Prepojovací konektor má niektoré vodiče nevyužité, nakoľko sú zodpovedajúce vývody FITkit-u vyhradené pre signály prerušenia a komunikáciu s perifériou audio kodeku, ktorý je súčasťou FITkit-u. Napriek tomu, že audio kodek nie je v tejto práci nijako využitý, vývody, ktoré sú s ním zdieľané, nie je možné použiť.

Obvod obsahuje dve skratovacie prepojky. Prepojka `JP1` vkladá do zbernice CAN terminačný rezistor o odporúčanej hodnote 120 Ω [4]. Prepojka `JP2` slúži na privedenie napájania z prídavných vodičov zbernice CAN k FITkit-u. Napájacie napätie 5 V pre obvod `MCP2551`



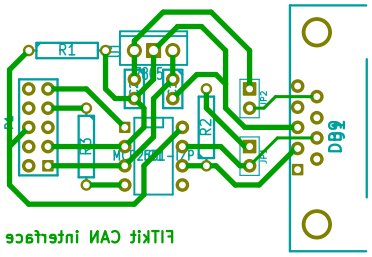
Obr. 6.1: Schéma obvodu fyzickej vrstvy CAN

zabezpečuje buď FITkit, alebo samostatný lineárny regulátor `7805`. Integrované obvody sú zapojené štandardným spôsobom podľa katalógu.

Obvod zapojený podľa mojej schémy umožňuje voľbu troch režimov. Voľba režimu sa uskutočňuje cez vývod číslo 6 prepojovacieho konektoru. Ak je vývod nepripojený (resp. pripojený k výstupu v stave vysokej impedancie), rezistor R1 obmedzuje strmosť hrany výstupu budiča. V prípade, že sa na tento vývod privedie potenciál 0 V, budič pracuje vo vysokorýchlostnom režime - hrany výstupného signálu budú strmšie, čo umožní vyššie prenosové rýchlosti za cenu zvýšeného elektromagnetického vyžarovania. Ak sa na tento výstup privedie logická hodnota 1, budič zbernice sa prepne do režimu spánku, výstup na zbernicu je deaktivovaný a príjmacie obvody pracujú s nižším príkonom. Informácie o režimoch činnosti boli prevzaté z katalógového listu obvodu `MCP2551` [9].

6.4 Doska plošných spojov

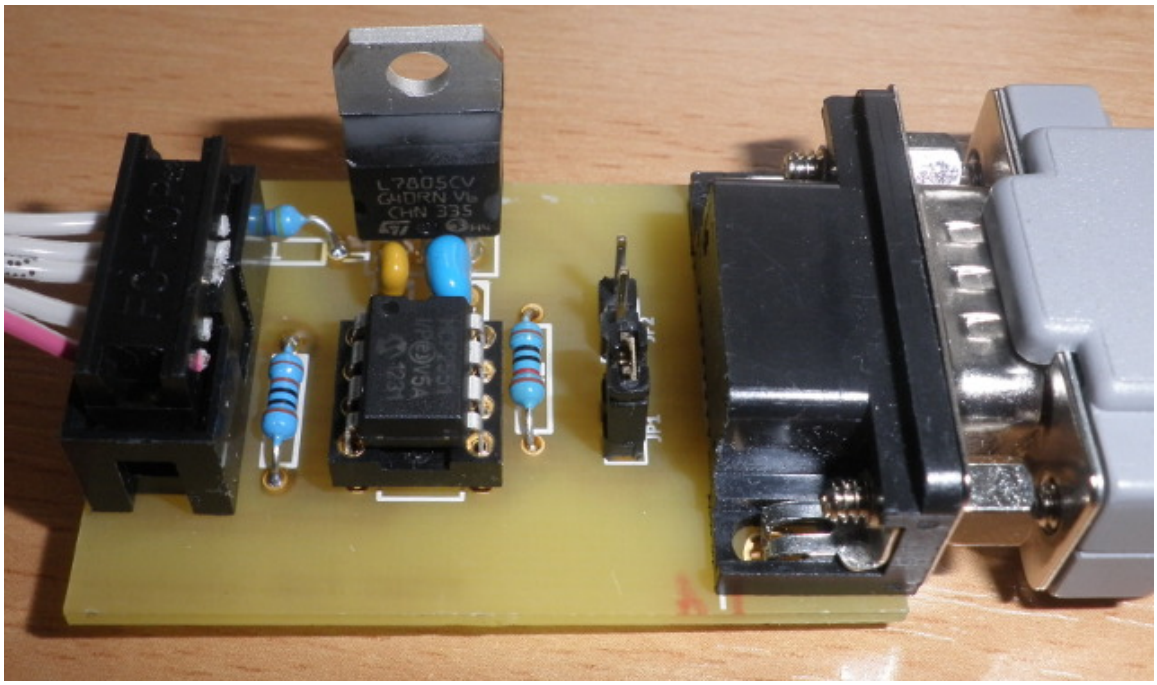
Pre obvod fyzického rozhrania bola navrhnutá jednovrstvová doska plošných spojov. Doska je osadená klasickými súčiastkami s drôtovými vývodmi (angl. *Through-hole technology*, THT). Integrovaný obvod je vložený v pätici pre púzdro DIP8. Konektor pre rozhranie zbernice je CANNON9 v uhlovom prevedení 90°. Rozhranie pre pripojenie k FITkit-u je vyvedené na hrebeňový konektor so zámkom, na ktorý sa pripojí prepojovací vodič. Návrh dosky v zdrojovom formáte sa nachádza na priloženom disku CD.



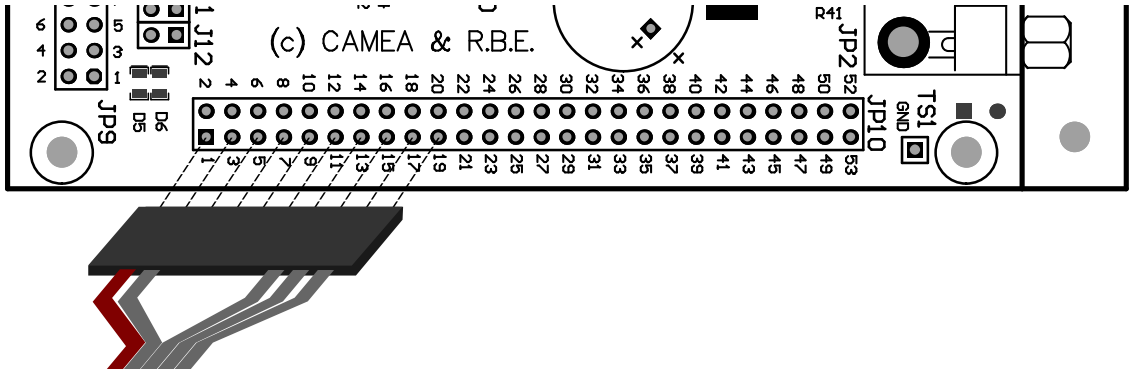
Obr. 6.2: Návrh dosky plošných spojov fyzickej vrstvy CAN

6.5 Vyhotovenie obvodu

Po pripravení dosky plošných spojov bola táto osadená súčiastkami. Pri oživovaní obvodu nedošlo k žiadnym problémom, obvod fungoval na prvé zapojenie. Bola otestovaná možnosť napájania obvodu z FITkit-u aj možnosť napájania obvodu a FITkitu z externého zdroja. Pre tento účel bol konektor zbernice CAN vybavený konektorom na pripojenie externého napájania.



Obr. 6.3: Hotový obvod s pripojeným prepojovacím vodičom



Obr. 6.4: Spôsob pripojenia k FITkit-u[1]

[1]Podklad k obrázku prevzatý zo stránok projektu FITkit: http://merlin.fit.vutbr.cz/FITkit

Kapitola 7

Implementácia radiča

7.1 Jadro radiča

Jadrom radiča je jednotka `can_core`. Na jej vstupy sú privedené vstupné bity zo zbernice (resp. z jednotky vzorkovania) a stavové signály z ostatných jednotiek. Jej srdcom je stavový automat, ktorého stavy zodpovedajú jednotlivým častiam CAN rámca. Okrem toho obsahuje čítač, ktorý počíta počet prenesených bitov v aktuálnej časti správy.

7.1.1 Stavová logika

Nasledovný stav automatu je určený kombinačnou logikou na základe súčasného stavu a vstupov. Stav sa zmení s vzostupnou hranou hodinového signálu `CLK`, ak je aktívny signál `CE` a zároveň je tento hodinový takt označený ako začiatok prenosu nového bitu aktiváciou signálu `TX_POINT`. Do registra `pState`, označujúceho súčasný stav, sa vtedy zapíše hodnota zo signálu `nState`. Tá je určená spomínanou kombinačnou logikou.

Stavy automatu sú zobrazené na diagrame v prílohe C.2. K prechodu do nasledujúceho stavu dochádza po dosiahnutí istej hodnoty počítadla bitov, ktorá zodpovedá počtu bitov v danej časti CAN rámca. K ďalším zmenám stavu dochádza pri príjme neplatného bitu, či už v rámci chyby bit stuffing-u, alebo prijatiu inej, ako očakávanej hodnoty v nemenných častiach správy daných štandardom.

7.1.2 Počítadlo bitov

Počítadlo bitov je obvyklou implementáciou čítača so synchrónnym nulovaním a povoľovacím signálom. Čítač je inkrementovaný pri vzostupnej hrane hodín za nasledovných podmienok:

- blok `can_core` má aktívny povoľovací signál `CE`
- v tomto hodinovom cykle je aktivovaný signál `TX_POINT`, ktorý označuje okamih začiatku vysielania nového bitu
- aktuálny bit nie je vloženým doplnkovým bitom (bit stuffing), t.j. nie je aktívny signál `RX_DESTUFF`
- bitový čítač má aktívny povoľovací signál `bit_cnt_enable`
- nie je aktívny signál nulovania čitača `bit_cnt_reset`

K nulovaniu čítača dochádza pri vzostupnej hrane hodín, ak je aktívny signál `bit_cnt_reset`.

7.1.3 Vstupná a výstupná logika

Nastavenie hodnôt výstupu a spracovanie hodnôt vstupu zo zbernice je implementované vo viacerých procesoch jednotky `can_core`.

Vstupná logika ukladá prijaté bity správy do posuvných registrov `messageID`, `controlField`, `data` a `crcField`. K posuvu dochádza s vzostupnou hranou hodín pri aktívnom signáli `RX_POINT`, ktorý označuje okamih čítania nového bitu zo zbernice. K posuvu však nedochádza, ak sa jedná o doplnkový bit vložený bit stuffing-om, ktorý je označený aktívnym signálom `RX_DESTUFF`.

Výstupná logika má podobnú štruktúru: výstupy posuvných registrov (ich najvyššie bity - správy na zbernici CAN sú odosielané s najvyšším bitom ako prvým, čo označujeme ako „MSB first“) sú vyvedené na signál `txBit`, ten je ďalej privedený na výstup `TX`. K posuvu dochádza synchrónne s hodinovým signálom, tentoraz však pri aktívnom signáli `TX_POINT`.

7.2 Bit stuffing

Funkciu bit stuffingu zabezpečuje jednotka `can_stuff`. Využíva sa pri príjme aj pri vysielaní správy.

Modul má vstup hodinového signálu `CLK`, asynchrónny resetovací signál `RESET`, povoľovací signál `CE`, vstup hodnoty zo zbernice `BIT_IN`.

Na výstupe `BIT_OUT` sa nachádza hodnota, ktorá by mala nasledovať v dátovom prúde. Pri príjme sa táto hodnota používa na porovnanie s prijatou hodnotou a následné prípadné vyvolanie chyby bit stuffing-u. Pri odosielaní je táto hodnota zapisovaná na výstup radiča a následne na zbernicu.

Vstupný dátový prúd je vzorkovaný v čase vzostupnej hrany hodinového signálu. V prípade, že sa takto prijme (alebo odošle) 5 rovnakých bitov, automat prejde do stavu `sStuff` a na výstup `BIT_OUT` zapíše inverzný bit a aktivuje signál `DO_STUFF`.

Funkciu bit stuffingu je možné naviac ovládať signálom `STUFF_EN`, jeho deaktivácia spôsobí zotrvanie automatu v stave `s0` a všetky vstupné bity sú prenášané na výstup.

Jednotka je pripojená k výstupu jednotky vzorkovania `can_sample` (vstup dát) a k jadru radiča `can_core` (stavové signály).

7.3 Kontrola chýb a CRC

Kontrola formátu správy sa odohráva v riadiacej logike stavového automatu jadra radiča `can_core`. Integrita správy je overovaná kontrolným súčtom. Výpočet tohoto kontrolného súčtu zabezpečuje jednotka `can_crc`. Je to modifikovaný posuvný register o dĺžke 15 bitov. Pomocou operácie `xor` sa s každým hodinovým cyklom vypočíta nová hodnota CRC na základe vstupnej hodnoty a hodnoty uloženej v registri. Na výpočet sa používa algoritmus

uvedený priamo v špecifikácií CAN[3].

```
CRC_REGISTER ← 0;
repeat
    CRCNXT ← vstupný bit xor CRC_REGISTER(14);
    CRC_REGISTER(14:1) ← CRC_REGISTER(13:0) ; // bitový posuv doľava
    CRC_REGISTER(0) ← 0 ;
    if CRCNXT = 1 then
        CRC_REGISTER(14:0) ← CRC_REGISTER(14:0) xor 0x4599 ;
    end
until koniec časti podliehajúcej CRC;
```

Algoritmus 1: Výpočet CRC

Jednotka výpočtu CRC má nasledovné vstupy a výstupy:

- `RESET` - synchrónny reset
- `CLK` - hodinový signál
- `CE` - povoľovací vstup
- `CRC_OUT` - výstupná hodnota CRC (15 bitov)
- `BIT_IN` - vstupný bit

Táto jednotka je pripojená k jadru radiča `can_core` a poskytuje výpočet CRC pri prijímaní aj odosielaní správy.

Pri prijímaní správy sa s každým prijatým bitom tento zapíše na vstup `BIT_IN` a aktivuje sa signál `CE`. Pri nasledovnom hodinovom takte sa na výstup `CRC_OUT` zapíše nová hodnota CRC. Tá sa po prijatí celej správy porovná s hodnotou v poli kontrolného súčtu a následne sa (ne-)odošle potvrdzovací bit ACK. Pri odosielaní správy sú do jednotky zapisované odosielané bity a v poli kontrolného súčtu sa odošle hodnota z `CRC_OUT`.

7.4 Vzorkovanie zbernice

Vzorkovanie zbernice zabezpečuje jednotka `can_sampler`. Tá má štandardné vstupy synchrónneho bloku - hodinový signál `CLK`, povoľovací vstup `CE` a asynchrónny reset `RESET`. Vstupný signál zo zbernice je prijímaný cez vstup `RX`, navzorkovaná hodnota je vystavená na výstup `BIT_OUT`.

Platnosť prijatého bitu je oznámená aktiváciou signálu `RX_POINT`. K tej dochádza na prelome prvého a druhého fázového segmentu (2.2.6). Časový okamih zápisu na zbernicu je označený aktiváciou signálu `TX_POINT` - ten sa nachádza v prvej - synchronizačnej fáze. Oba signály sú aktívne po dobu jedného hodinového taktu.

Dĺžka jednotlivých fáz sa nastavuje pomocou vstupov `PROP_CNT_REG`, `PHASE1_CNT_REG` a `PHASE2_CNT_REG`. Maximálna dĺžka skoku je nastavená vstupom `SJW`.

Počas príjmu prvého bitu správy dochádza k tzv. tvrdej (*hard*) synchronizácií. Túto aktivuje jadro radiča pomocou signálu `SYNC_HARD`. Počas vysielania správy k synchronizácií nedochádza - je blokovaná deaktiváciou signálu `SYNC_EN`.

Vzorkovacia jednotka je stavový automat, ktorý v každej fáze príjmu jedného bitu inkrementuje počítadlo časových kvánt. Toto počítadlo je komparované s hodnotami v riadiacich

registroch a v prípade zhody sa stavový automat presunie do ďalšieho stavu. Synchronizácia je zabezpečená skracovaním a predlžovaním dĺžok fázových segmentov. Po vykonaní synchronizácie (tvrdej synchronizácie, alebo synchronizácie počas jedného z fázových segmentov) je aktivovaný príznak `PHASE_COMPENSATED` a ďalšia synchronizácia je blokovaná. Tento príznak je vynulovaný s príchodom ďalšieho bitu (tj. počas synchronizačnej fázy).

Diagram stavového automatu v notácií ASM sa nachádza v prílohe C.1

7.5 Časovanie a preddelička

Zdrojom časovania pre rôzne prenosové rýchlosti je obvod preddeličky implementovaný v module `can_baud`. Je to synchrónny čítač s nastaviteľnou hranicou čítania. Po dosiahnutí tejto medznej hodnoty je počítací register vynulovaný a čítanie pokračuje od nuly. Zároveň je aktivovaný výstupný signál `CE_OUT`, ktorý slúži ako povoľovací vstup pre ďalšie obvody. Hranicu čítania je možné nastaviť vystavením novej hodnoty na zbernicu `DATA_IN` a aktiváciou signálu `WE`. Naopak, pri aktívnom signále `OE` sa na výstup `DATA_OUT` zapíše aktuálna hranica čítania.

7.6 Filtre správ

Radič obsahuje dva nezávislé akceptačné filtre. Tie porovnávajú hodnotu `MessageId` práve prijatej správy o jej akceptovaní rozhodnú na základe porovnania so zadanou hodnotou. Porovnanie sa uplatňuje len na bity určené maskou. Oba filtre sú inštanciou modulu `can_filter`. Rovnomenná entita definovaná v tomto module je kombinačný obvod so vstupmi `FILTER`, `MASK` a `DATA` o premenlivej šírke určenej generickým parametrom `WIDTH`. Výstup `MATCH` je aktívny (log. 1) v prípade, že hodnota na vstupe `DATA` je po vymaskovaní bitov podľa vstupnej masky zhodná s hodnotou na vstupe `FILTER`.

Nakoľko sú filtre náročné na prostriedky (je potrebné uložiť identifikátor o dĺžke 29 bitov, akceptačnú masku o rovnakej dĺžke a vytvoriť komparátor), bol vo finálnom zostavení vysyntetizovaný iba jeden filter. Syntézu druhého filtra je možné zapnúť generickým parametrom `GEN_FILTER1` entity `can`.

7.7 Komunikačné rozhranie s MCU

Komunikácia s mikrokontrolérom prebieha pomocou rozhrania SPI. Na tento účel boli využité komponenty používané v demonštračných aplikáciach FITkit-u [13].

7.7.1 Komunikačný systém

Rozhranie SPI je vysokorýchlostné synchrónne sériové rozhranie, umožňujúce komunikovať jednému nadriadenému (master) zariadeniu s viacerými podriadenými (slave) zariadeniami. Komunikácia prebieha výmenou obsahov dvoch 8-bitových posuvných registrov medzi master a slave zariadením. Na FITkit-e je v úlohe master mikrokontrolér, FPGA má úlohu podriadeného zariadenia. V rámci FPGA je možné adresovať viacero cieľových jednotiek využitím správ s nasledovným formátom:

1. **Odoslanie operačného kódu** - operačný kód určuje požadovanú operáciu - čítanie alebo zápis

2. **Odoslanie adresy** - adresa určuje cieľovú jednotku v FPGA, počet bitov adresy je možné zvoliť

3. **Dátová časť** - prenos ľubovoľnej dĺžky dát

V FPGA je správa spracovaná radičom SPI (jednotka `SPI_ctrl`), ktorý dáta z SPI prenáša na internú sériovú zbernicu. Tam sú dáta analyzované dekóderom SPI (jednotka `SPI_adc`). Ak správa SPI adresuje priestor tohoto dekóderu, ten na paralelné rozhranie vystaví adresu (offset) registra a dáta správy. Následne aktivuje signály povoľujúce zápis alebo čítanie registra.

7.7.2 Stavové a riadiace registre

Registre slúžiace na komunikáciu s MCU sú implementované v bloku `can`. Ten obsahuje dekóder adries registrov a samotné registre. Registre sú implementované v module `univ_register`. Ten reprezentuje register na všeobecné použitie, s voliteľnou šírkou a trojstavovým výstupom. Naviac majú aj sekundárny výstup, ktorý je trvale aktívny (nepodlieha povoľovaciemu signálu `OUTPUT ENABLE`).

Registre určené na zápis zo strany MCU (t.j. radič z nich len číta, MCU môže zapisovať aj čítať) sú pripojené vstupom a výstupom na vnútornú dátovú zbernicu a signály na povolenie čítania a zápisu sú privedené z dekódera adries. Sekundárny výstup je privedený na vstupy jadra radiča.

Registre, do ktorých zapisuje radič, ale ktoré sú pre MCU určené len na čítanie (napr. registre prijatej správy), majú vstup pripojený na výstupné signály jadra radiča a povolenie zápisu aktivuje samotný radič.

Mapa registrov a schéma organizácie registrov sa nachádza v prílohe D.

Stavový register CAN_STATUS

Tento register slúži na zistenie stavu radiča ako aj na zvolenie režimu práce. Podrobný popis jednotlivých bitov registra sa nachádza v prílohe D. Dostupné režimy, okrem normálneho režimu sú :

- SLEEP - režim spánku, transciever pracuje s nižším príkonom
- SUSPEND - režim bez činnosti - správy nie sú prijímané ani odosielané
- HISPEED - režim vysokej rýchosti - transciever pracuje s ostrejšími hranami výstupného signálu, čo umožňuje vyššie prenosové rýchlosti na úkor vyžarovaného rušenia
- LOOPBACK - spätnosľučkový režim - odoslané správy sú spracované ako prijaté

Okrem voľby režimov tento register poskytuje informáciu o prijatí novej správy, prípadne výskyte chybového stavu. Zápisom na príslušný bit registra sa spustí odosielanie správy zapísanej v príslušných dátových registroch.

Aby bolo možné modifikovať obsah registra po bitoch, je k tomuto registru pridružený register `CAN_STATUSMASK`, ktorého obsah určuje, ktoré bity zapísané na adresu registra `CAN_STATUS` sa do neho skutočne prenesú. Zápisu do samotného registra teda zväčša predchádza zápis do maskovacieho registra, ktorý určí rozsah modifikácie stavového registra.

Chybová slabika

Informácie o chybe prenosu môže nadradená aplikácia získať z registra `CAN_ERROR`. Každý bit registra označuje jeden druh chyby prenosu. Register je prepísaný vždy po ukončení prenosu. Pri úspešnom dokončení prenosu sa chybové príznaky vynulujú.

7.7.3 Registre prijatej správy

Prijatá správa sa skladá v registroch vnútri jadra radiča, ktoré nie sú prístupné pomocou komunikačného rozhrania. Po dokončení príjmu správy sa celá správa presunie do príjmových registrov a nastaví sa bit `CAN_RXMSG` v stavovom registri, čím sa oznámi príjem novej správy. Na uloženie prijatej správy slúži 13 registrov: 4 registre na uloženie identifikátora správy `CAN_RXMSGID...`, register riadiaceho poľa správy `CAN_RXCTRL` a 8 registrov dátovej časti správy `CAN_RXDATA0` až `CAN_RXDATA7`. Ich podrobný popis sa opäť nachádza v prílohe D.

7.7.4 Registre odosielanej správy

Registre odosielanej správy majú rovnakú štruktúru ako registre prijatej správy. Ich názvy však začínajú na `CAN_TX`. Pri zápise dátovej časti správy je potrebné dáta zapísať zarovnané doprava, tj. v prípade, že správa dĺžky n je kratšia ako 8 bytov, registre `CAN_TXDATA0` až `CAN_TXDATA`$(7-n)$ ostanú prázdne a správa sa zapíše do registrov `CAN_TXDATA7` až `CAN_TXDATA`$(8-n)$. Tento spôsob zápisu pre užívateľa transparentne rieši knižnica `fitkitcan` opísaná v nasledujúcej kapitole.

7.7.5 Registre časovania

Päť registrov nastavuje parametre časovania zbernice. Register `CAN_BAUD` nastavuje deliaci pomer preddeličky hodinového signálu, registre `CAN_TPROP`, `CAN_TPHASE1` a `CAN_TPHASE2` nastavujú dĺžky jednotlivých časových fáz prenosu jedného bitu. Ich hodnoty zodpovedajú počtu taktov hodín od začiatku bitu (nie od začiatku danej fázy), tj. pre nastavenie dĺžok segmentov na 2, 12, 2 je potrebné nastaviť hodnoty 2, 14, 16.

7.7.6 Registre filtrov správ

Prijaté správy je možné filtrovať na základe ich identifikátora. Pre jeden filter je vyhradených 8 registrov: 4 slúžia na uloženie identifikátora a 4 na uloženie masky.

7.7.7 Počítadlá chýb

Štandard CAN vyžaduje od radiča počítať množstvo chýb pri príjme a odosielaní správy. Tieto hodnoty sú dostupné po prečítaní registrov `CAN_REC` a `CAN_TEC`. Prvý z nich obsahuje hodnotu počítadla chýb príjmu, druhý chýb odosielania. Zápisom ľubovoľnej hodnoty na adresu počítadla dôjde k jeho vynulovaniu.

7.7.8 Radič prerušení

Jednoduchý radič prerušení je implementovaný v module `can`. Ten sleduje stav príznakov novej správy, odosielania správy a výskytu chyby. V prípade, že dôjde k zmene daného príznaku (vzostupnej hrane signalizujúcej príjem novej správy alebo vznik chyby, resp.

zostupnej hrane v prípade dokončenia odosielania správy) aktivuje výstupný signál `IRQ`. Ten je privedený na vstupný pin MCU na ktorom môže byť povolené prerušenie. Aby nedochádzalo k prerušeniu aktívneho prenosu pomocou rozhrania SPI, je aktivácia signálu `IRQ` blokovaná po dobu aktívneho signálu `SPI_CS`.

Signál `IRQ` je deaktivovaný po prečítaní stavového registra `CAN_STATUS` - pre jednoduchosť nemá radič prerušení vyhradený vlastný register (tzv. interrupt vector), nakoľko sa všetky informácie o vzniku prerušenia nachádzajú v stavovom registri.

Kapitola 8

Demonštračná aplikácia

Na ukážku funkcií radiča som vytvoril aplikáciu pre mikrokontrolér FITkit-u. Tá umožňuje príjem a odosielanie CAN správ pomocou konzolového rozhrania. Aplikácia sa skladá z komunikačnej knižnice `fitkitcan`, ktorá obsahuje funkcie na inicializáciu a ovládanie radiča CAN, a samotného programu s textovým užívateľským rozhraním.

8.1 Knižnica fitkitcan

Táto knižnica poskytuje dve sady funkcií: vysokoúrovňové funkcie, ktoré pracujú s dátovými štruktúrami predstavujúce CAN správy a nízkoúrovňové funkcie slúžiace na priame nastavenie a čítanie registrov radiča.

8.1.1 Vysokoúrovňová komunikácia

Základnými funkciami radiča je príjem a odosielanie správy. Správa je reprezentovaná dátovou štruktúrou `CanMsg`, ktorá obsahuje nasledujúce položky:

- `CanMsgId_t` **id** - identifikátor správy (štandardný alebo rozšírený)
- `bool` **ide** - príznak rozšíreného identifikátora
- `bool` **rtr** - príznak vzdialeného (*remote*) rámca
- `CanLen_t` **dlc** - dĺžka správy
- `CanData_t` **data**[**8**] - dáta správy

Na odoslanie správy slúži funkcia `send(*CanMsg)`, ktorá podľa údajov z dátovej štruktúry správy naplní registre odchádzajúcej správy radiča a aktivuje signál na odoslanie správy v riadiacom registri.

Prijatú správu je možné prečítať funkciou `receive(*CanMsg)`. Tá vyčíta údaje z registrov prijatej správy a naplní predanú dátovú štruktúru.

Ďalšou funkciou je nastavenie prenosovej rýchlosti. Volaním `baud(long baud)` sa vypočíta vhodné nastavenie časových registrov (preddelička a registre dĺžky fázových segmentov) a toto sa zapíše do registrov radiča.

8.1.2 Nízkoúrovňové funkcie

Táto sada funkcií slúži na priamu manipuláciu s obsahom registrov radiča. Väčšina funkcií má sémantiku `register(hodnota)` pre zápis a `register()` pre čítanie hodnoty. Knižnica obsahuje nasledovné funkcie:

- `status(), status(mask, value)` - čítanie a nastavenie stavového registra
- `prop(), prop(value)` - nastavenie dĺžky propagačného segmentu
- `t1(), t1(value)` - nastavenie dĺžky prvého fázového segmentu
- `t2(), t2(value)` - nastavenie dĺžky druhého fázového segmentu
- `sjw(), sjw(value)` - nastavenie dĺžky synchronizačného skoku
- `rxId()` - prečítanie ID prijatej správy
- `txId(id)` - nastavenie ID odosielanej správy
- `txData(len, *data)` - nastavenie dát odosielanej správy
- `rxData(len, *data)` - prečítanie dát prijatej správy
- `txControl()` - zapísanie riadiaceho poľa odosielanej správy

Komunikácia prebieha pomocou rozhrania SPI ako bolo vysvetlené v kapitole 7.7.1. Na ovládanie rozhrania SPI sa používajú funkcie z knižnice fitkitlib, konkrétne `FPGA_SPI_RW_A8_D8` slúži na zápis alebo čítanie 8-bitového obsahu registra adresovaného 8-bitovou adresou. Táto funkcia je obalená do funkcií `spiread(addr)` a `spiwrite(addr, value)` na čítanie a zápis do registrov daných relatívnou adresou (offsetom) `addr`. Bázová adresa radiča CAN sa uvedie pri inicializácií knižnice.

8.2 Konzolová aplikácia

Ako rozhranie medzi užívateľom a radičom slúži konzolová aplikácia, ktorá beží na MCU FITkit-u. Tá poskytuje základné príkazy na inicializáciu radiča, príjem a odosielanie správ a zisťovanie stavu radiča pomocou intuitívnej syntaxe.

8.2.1 Ovládanie

Všetky príkazy majú rovnakú syntax `<príkaz> parameter = hodnota parameter2 = hodnota ...`. Hodnoty parametrov majú zápis podľa typu parametra. Čísla môžu byť zapísané v desiatkovom tvare alebo hexadecimálnom vo formáte `0xab`. Reťazce je možno zapísať priamo alebo ich uzavrieť do úvodzoviek či apostrofov alebo je ich možné zapísať ako postupnosť hexadecimálnych čísel v tvare `0xabcd`. Logické hodnoty (*boolean*) môžu byť zapísané ako `on - off`, `1 - 0` alebo `true - false`. V názvoch príkazov a parametrov nezáleží na veľkosti písmen.

8.2.2 Podporované príkazy

Dostupné príkazy sú zhrnuté v nasledujúcej tabuľke.

príkaz	parameter	typ	význam
set			nastaví parametre prenosu
	baud	číslo	prenosová rýchlosť v b/s
	prescaler	číslo	hodnota preddeličky
	prop	číslo	trvanie (počet časových kvánt) propagačného segmentu
	t1	číslo	trvanie prvého fázového segmentu
	t2	číslo	trvanie druhého fázového segmentu
	loop	boolean	zapne alebo vypne Loopback režim
	sleep	boolean	zapne alebo vypne režim úspory energie
	hispeed	boolean	zapne alebo vypne vysokorýchlostný režim
send			pošle správu podľa parametrov
	id	číslo	identifikátor správy
	eid	číslo	rozšírený identifikátor správy (nie je možné kombinovať s parametrom id)
	dlc	číslo	dĺžka správy (ak je vyplnený parameter data, určí sa automaticky)
	data	reťazec	obsah dátovej časti správy
	rtr	boolean	ak je 1, správa bude odoslaná ako *Remote Frame*
filter			nastaví parametre filtra správ
	id	číslo	identifikátor správy
	mask	číslo	bitová maska maska
	n	číslo	číslo filtra (nepovinné, implicitná hodnota 0)
	enable	boolean	(de)aktivácia filtra
test			
	enable	boolean	povolí testovací režim - aplikácia bude na prijaté správy reagovať odoslaním potvrdzovacej správy, po ktorej odošle náhodnú testovaciu správu
	master	boolean	zariadenie označené ako *master* bude iniciovať test - ako prvé odošle náhodnú testovaciu správu. V prípade straty správy (pri chybe prenosu) po určitej časovej pauze znova odošle testovaciu správu.

Tabuľka 8.1: Podporované príkazy

Kapitola 9

Overenie funkčnosti

Na overenie funkčnosti mojej implementácie boli vykonané dve sady testov. V prvej sade bola testovaná schopnosť komunikovať medzi dvoma inštanciami mojej implementácie. Pre tento účel boli použité dve vývojové dosky FITkit, ktoré boli prepojené fyzickou vrstvou podľa kapitoly 6. V druhej sade bol jeden z FITkitov nahradený vývojovou doskou Discovery spomínanou v kapitole 4.3.1, čím som otestoval kompatibilitu s inou implementáciou CAN.

V prípade oboch testov bola vytvorená aplikácia, ktorá generovala náhodné správy s premenlivou dĺžkou a príznakmi. Na príjem správy reagovalo druhé zariadenie odoslaním potvrdzovacej správy, ktorá verifikovala správne prijatie. Po prenose takejto sady správ sa úlohy oboch zariadení vymenili. Prípadné chyby prenosov sa počítali a zapisovali na terminál.

Testy prebiehali pri bežných prenosových rýchlostiach 250 kbit/s, 500 kbit/s a 1000 kbit/s. V každom teste bolo prenesených aspoň 10000 správ, potom bol test ručne zastavený.

9.1 Výsledky testov

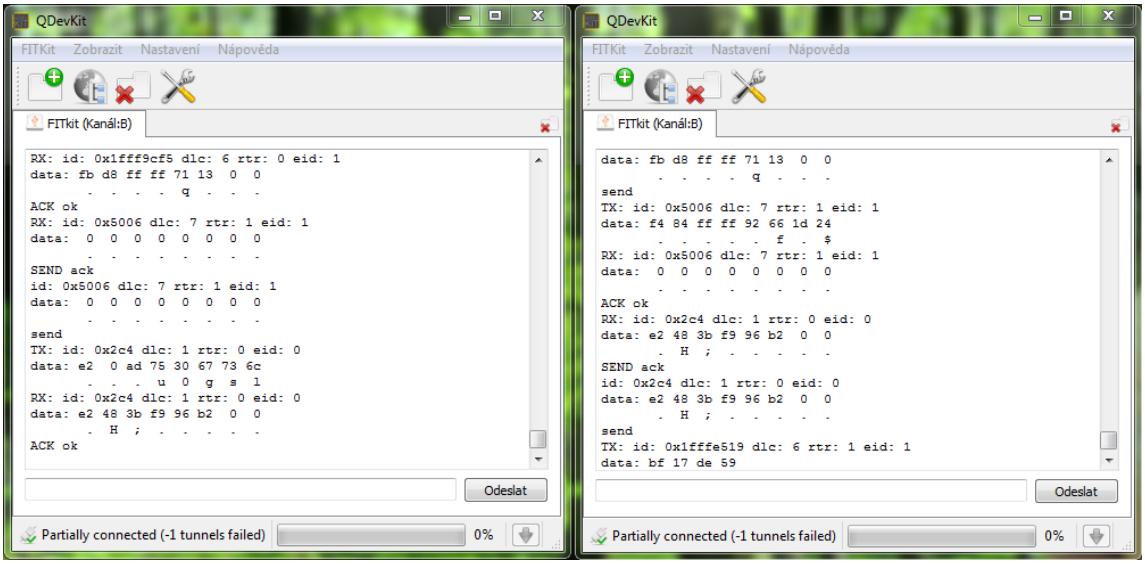
Testovanie prebiehalo kontinuálne počas vývoja radiča, v tejto kapitole je však opísaná podoba a výsledky testov na konci vývoja. Počas vývoja testy odhalili niektoré chyby implementácie, ktoré boli neskôr opravené.

9.1.1 Testovanie samotnej implementácie

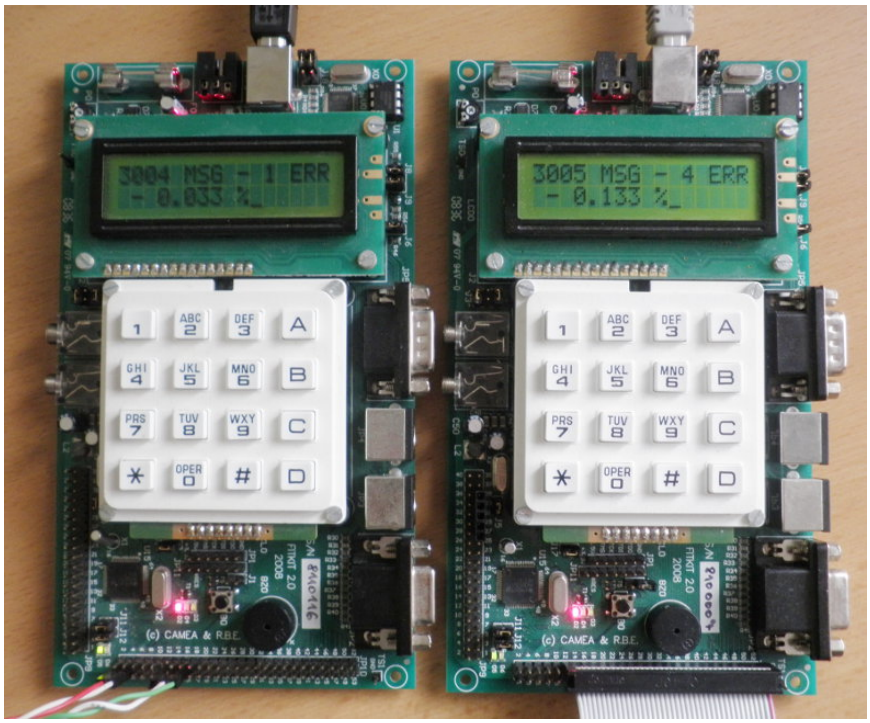
V tomto teste medzi sebou komunikovali dva FITkit-y s mojou implementáciou radiča CAN. Výsledky testu sú zhrnuté v tabuľke.

prenosová rýchlosť	počet správ	počet chýb	chybovosť
250 kbit/s	11299	0	0,00%
500 kbit/s	12608	0	0,00%
1000 kbit/s	13875	0	0,00%

Tabuľka 9.1: Výsledky testovania dvoch inštancií mojej implementácie



(a) Snímka obrazovky bežiaceho testu



(b) Fotografia testovaných zariadení

Obr. 9.1: Testovanie komunikácie medzi dvoma FITkit-mi

9.1.2 Testovanie kompatibility

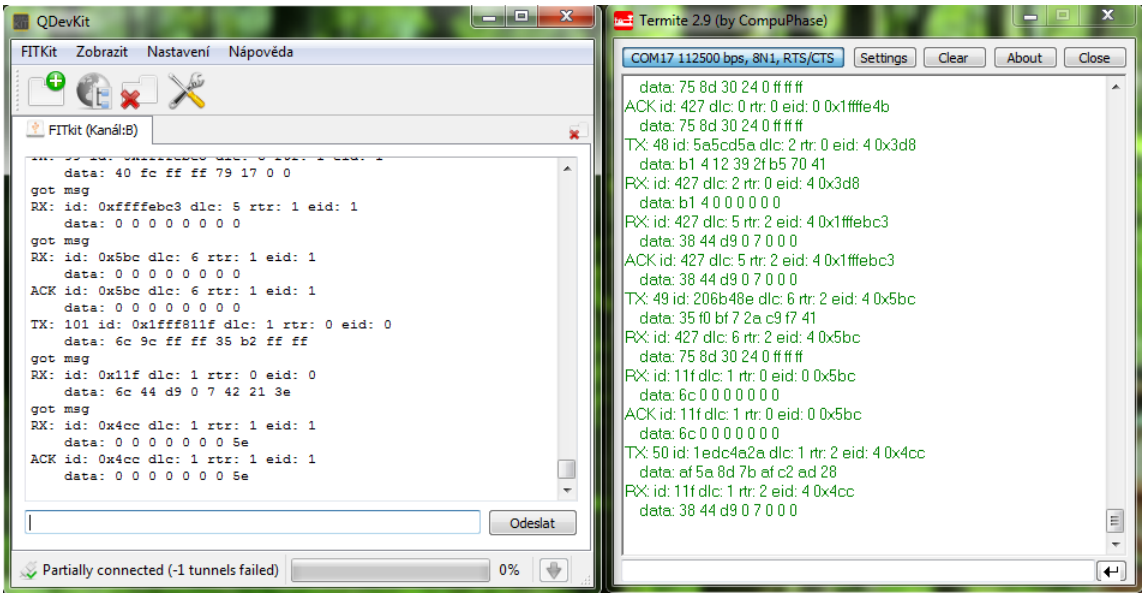
V ďalšom teste sa overovala kompatibilita mojej implementácie voči existujúcemu zariadeniu. Výsledky testov sú opäť zhrnuté v tabuľke.

prenosová rýchlosť	počet správ	počet chýb	chybovosť
250 kbit/s	12832	0	0,00%
500 kbit/s	14478	2	0,01%
1000 kbit/s	12767	4	0,03%

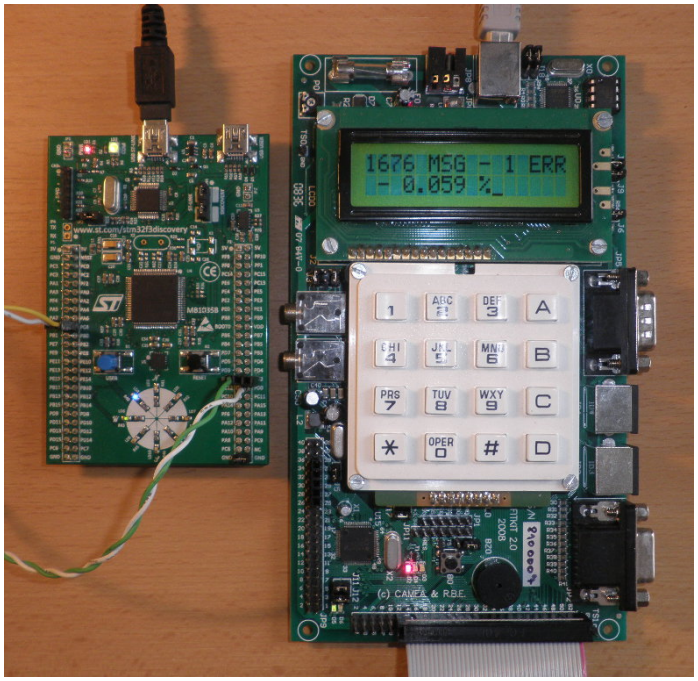
Tabuľka 9.2: Výsledky testovania voči existujúcemu zariadeniu

9.1.3 Vyhodnotenie testov

Pri prvom teste dosiahol radič vynikajúce výsledky – pri prenose nedošlo k výskytu žiadnej chyby. Z toho usudzujem, že radič je schopný prenosu aj pri vysokých rýchlostiach. Pri druhom teste došlo k malému zvýšeniu chybovosti. Pri maximálnej prenosovej rýchlosti bola chybovosť 0,03%. Toto hodnotím ako veľmi dobrý výsledok. Chyby môžu byť spôsobené z časti náhodným rušením, nakoľko pri teste bola použitá zbernica bez tienenia. Druhá príčina vzniku chýb môže byť nepresnosť v časovaní protokolu. Oba tieto zdroje chýb majú väčší efekt pri vyšších rýchlostiach – prechodná porucha alebo časovacia odchýlka sa stáva významnejšou s kratším časom prenosu jedného bitu. Z pohľadu protokolu CAN nie sú tieto množstvá chýb príliš významné, pretože protokol bol navrhnutý robustne. Potenciálne chybné správy sa pre istotu zneplatnia a odošlú znovu.



(a) Snímka obrazovky bežiaceho testu



(b) Fotografia testovaných zariadení

Obr. 9.2: Testovania kompatibility voči CAN radiču na doske Discovery

Kapitola 10

Záver

Cieľom práce bolo vytvoriť funkčný radič CAN kompatibilný s inými zariadeniami CAN pracujúcimi podľa štandardu. V prvej časti som čitateľa zoznámil s princípmi komunikácie na zbernici CAN. Na základe týchto poznatkov som v druhej časti analyzoval niektoré implementácie CAN radičov. Potom som navrhol štruktúru môjho radiča.

Najrozsiahlejšou časťou práce je samotná implementácia radiča. Tá vychádza z návrhu a opiera sa o pôvodnú špecifikáciu CAN a dostupnú literatúru. Implementovaný bol kompletný radič CAN 2.0B, ktorý podporuje všetky štandardizované funkcie: odosielanie a príjem rámcov o štandardnej a rozšírenej dĺžke, arbitrácia v prípade konfliktu na zbernici, príjem a zasielanie špeciálnych rámcov (vzdialený, chybový a preťažovací rámec).

Implementácia bola otestovaná pri bežných podmienkach na kompatibilitu voči komerčne dostupnému radiču. Tu je priestor na rozšírenie práce – dôkladnejšie testovanie implementácie špecificky zameranými jednotkovými testami, prípadne formálna verifikácia radiča voči špecifikácií. Výsledky testov hodnotím ako veľmi uspokojivé, radič je bez problémov použiteľný na komunikáciu na CAN zbernici.

Prínosom tejto práce je vytvorenie ďalšej periférie k školskej vývojovej doske FITkit ako aj vytvorenie vhodného aplikačného rozhrania – knižnice `fitkitcan`. Tá sprostredkúva všetky funkcie radiča užívateľskej aplikácií. Tu je možné nadviazať vytvorením konkrétnych aplikácií využívajúcich CAN radič, napríklad monitor CAN zbernice alebo nástroj na diagnostiku osobných automobilov.

Literatúra

[1] CAN in Automation (CiA): CAN history [online]. [cit. 2014-01-18].
URL http://www.can-cia.de/index.php?id=161

[2] CAN in Automation (CiA): CAN physical layer [online]. [cit. 2014-01-28].
URL http://www.can-cia.org/index.php?id=systemdesign-can-physicallayer

[3] Bosch, R.: *Can Specification Version 2.0*. Robert Bosch GmbH, Postfach 50, D-7000 Stuttgard 1, 1991.

[4] CAN in Automation: *CiA Draft Standard 102 Version 2.0: CAN Physical Layer for Industrial Applications*. 1994-04-20 [cit. 2014-01-29].

[5] Carvalho, F.; Jansch-Porto, I.; Freitas, E.; aj.: The TinyCAN: an optimized CAN controller IP for FPGA-based platforms. In *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, ročník 1, 2005, s. 4–374, doi:10.1109/ETFA.2005.1612547.

[6] Donchev, B.; Hristov, M.: Implementation of CAN controller with FPGA structures. In *CAD Systems in Microelectronics, 2003. CADSM 2003. Proceedings of the 7th International Conference. The Experience of Designing and Application of*, 2003, s. 577–580, doi:10.1109/CADSM.2003.1255163.

[7] Holota, R.: CAN Bus Analyser with Utilisation of FPGAs. In *Applied Electronics 2003*, Plzeň, Západočeská univerzita, 2003, ISBN 80-7082-951-6, s. 83–86.

[8] ISO/IEC 7498-1: *Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model*. 1996.

[9] Microchip Technology Inc.: *MCP2551 High-Speed CAN Transceiver*. 2010 [cit. 2014-05-02].
URL http://ww1.microchip.com/downloads/en/DeviceDoc/21667f.pdf

[10] Richards, P.: A CAN Physical Layer Discussion. 2002.
URL http://ww1.microchip.com/downloads/en/AppNotes/00228a.pdf

[11] STMicroelectronics: *RM0316 Reference manual - STM32F302xx, STM32F303xx and STM32F313xx advanced ARM-based 32-bit MCUs*. 2014 [cit. 2014-03-31].
URL http://www.ti.com/lit/pdf/slab034

[12] Texas Instruments: *MSP430^{TM} Ultra-Low-Power Microcontrollers*. 2014 [cit. 2014-01-28].
URL http://www.ti.com/lit/pdf/slab034

[13] Vašíček, Z.: Komunikační systém - FITkit [online]. 2009-09-17 [cit. 2014-03-12].
URL
http://merlin.fit.vutbr.cz/FITkit/docs/firmware/fpga_interconect.html

[14] Vašíček, Z.: Hardware - FITKit [online]. 2012 [cit. 2014-01-18].
URL http://merlin.fit.vutbr.cz/FITkit/hardware.html

[15] Voss, W.: *A Comprehensible Guide to Controller Area Network*. Copperhill Media Corporation, druhé vydání, 2005, ISBN 0976511606.

[16] WWW stránky: Introduction to Wired-OR Outputs and Open-Collector Circuits [online]. 2012-12-05 [cit. 2014-01-18].
URL http://www.ni.com/white-paper/3544/en/

Zoznam skratiek

- *D* - dominantný bit
- *r* - recesívny bit
- ACK - potvrdenie príjmu (*Acknowledgment*)
- BRAM - bloková RAM (*Block RAM*)
- CAN - *Controller Area Network*
- CE - povolenie hodinového signálu (*Clock Enable*)
- CiA - organizácia *Can in Automation*
- CLK - hodinový signál (*Clock*)
- CRC - cyklický redundantný kontrolný súčet (*Cyclic Redundant Checksum*)
- DLC - kód dĺžky správy (*Data Length Code*)
- EOF - koniec rámca (*End Of Frame*)
- FIFO - „prvý dnu, prvý von“ (*First In First Out*)
- FPGA - programovateľné hradlové pole (*Field Programmable Gate Array*)
- GPIO - vstup/výstup na obecné použitie (*General Purpose Input Output*)
- IFS - medzirámcová pauza (*Interframe Space*)
- IRQ - žiadosť o prerušenie (*Interrupt Request*)
- ISO - Medzinárodná organizácia pre štandardizáciu (*International Organization for Standardization*)
- LCD - displej z tekutých kryštálov (*Liquid Crystal Display*)
- MCU - mikrokontrolér
- RAM - pamäť s náhodným prístupom (*Random Access Memory*)
- RTR - žiadosť o zaslanie rámca (*Remote Transmission Request*)
- SOF - začiatok rámca (*Start of Frame*)
- SPI - rozhranie *Standard Peripheral Interface*
- TTL - logika typu tranzistor - tranzistor (*Transistor Transistor Logic*)
- VHDL - jazyk na popis hardwaru (*VHSIC Hardware Description Language*)

Príloha A

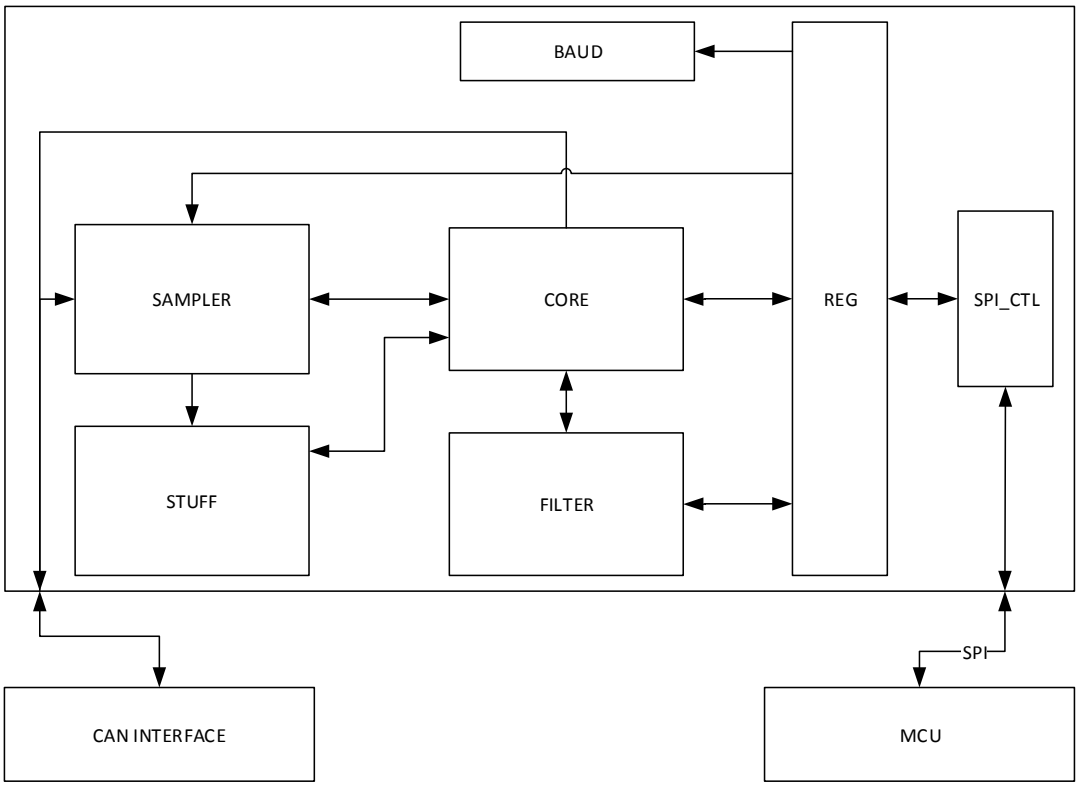
Obsah CD

Priložený nosič CD obsahuje nasledovnú adresárovú štruktúru:

- /CAN – priečinok projektu CAN radiča pre QDevKit
  - /build – zostavené binárne súbory pre nahratie do FITkitu
  - /fpga – zdrojové kódy pre FPGA v jazyku VHDL
  - /mcu – zdrojové kódy pre mikrokontrolér v jazyku C/C++
- /doc – elektronická verzia tohoto dokumentu
- /pcb – projekt dosky plošných spojov pre KiCad
  - /gerber – export dosky plošných spojov vo formáte GERBER
- /tex – zdrojové kódy technickej správy vo formáte LaTeX
- README – textový súbor popisujúci obsah disku CD

Príloha B

Bloková schéma radiča CAN

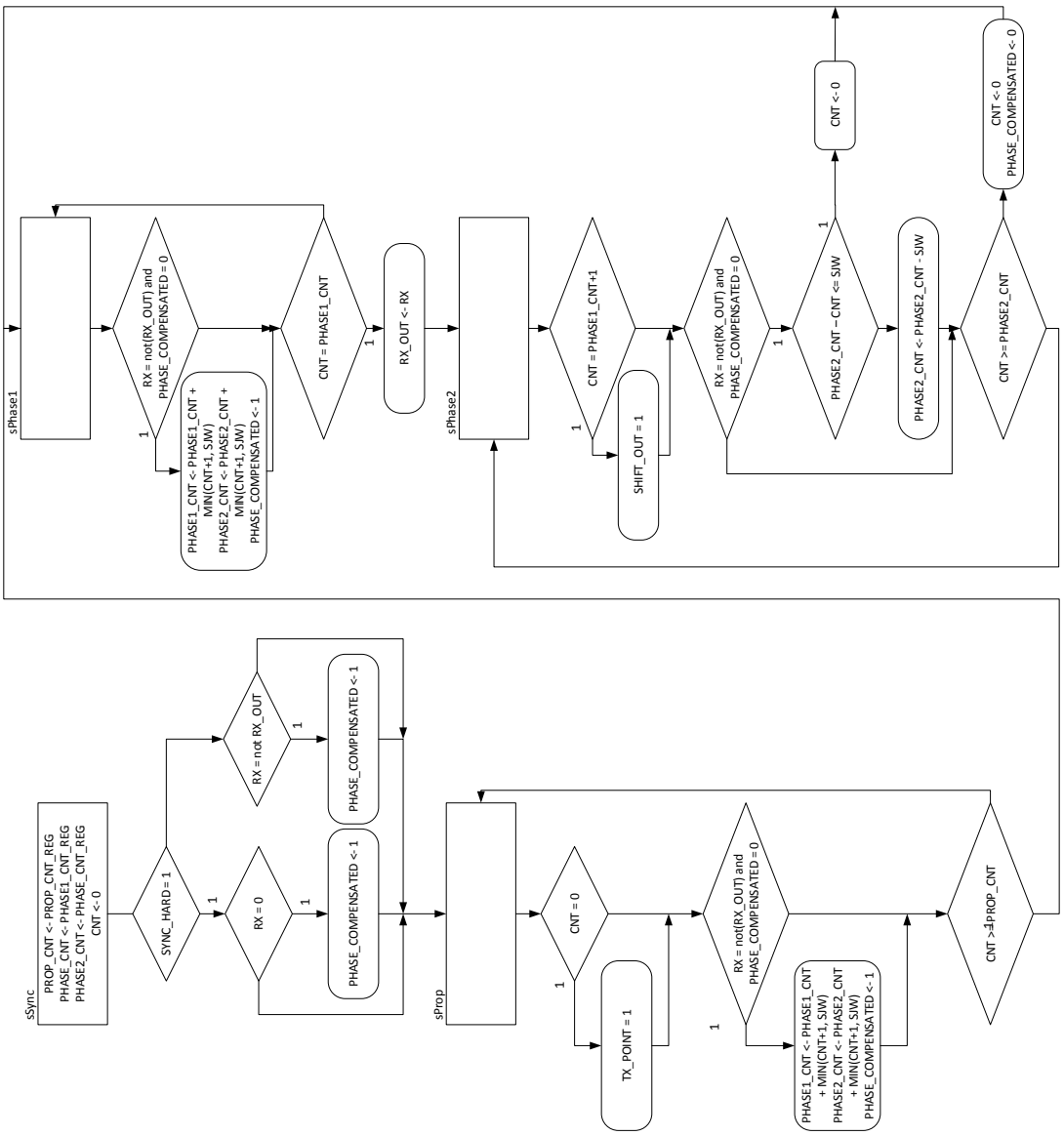


Obr. B.1: Prepojenie blokov radiča CAN: `BAUD` - zdroj hodinového signálu podľa prenosovej rýchlosti, `CAN INTERFACE` - zorhranie pre zbernicu CAN, `CORE` - jadro radiča, `FILTER` - blok akceptačných filtrov, `MCU` - mikrokontrolér, `REG` - sada registrov, `SAMPLER` - jednotka vzorkovania zbernice, `SPI_CTL` - radič SPI rozhrania, `STUFF` - jednotka bit stuffin-u

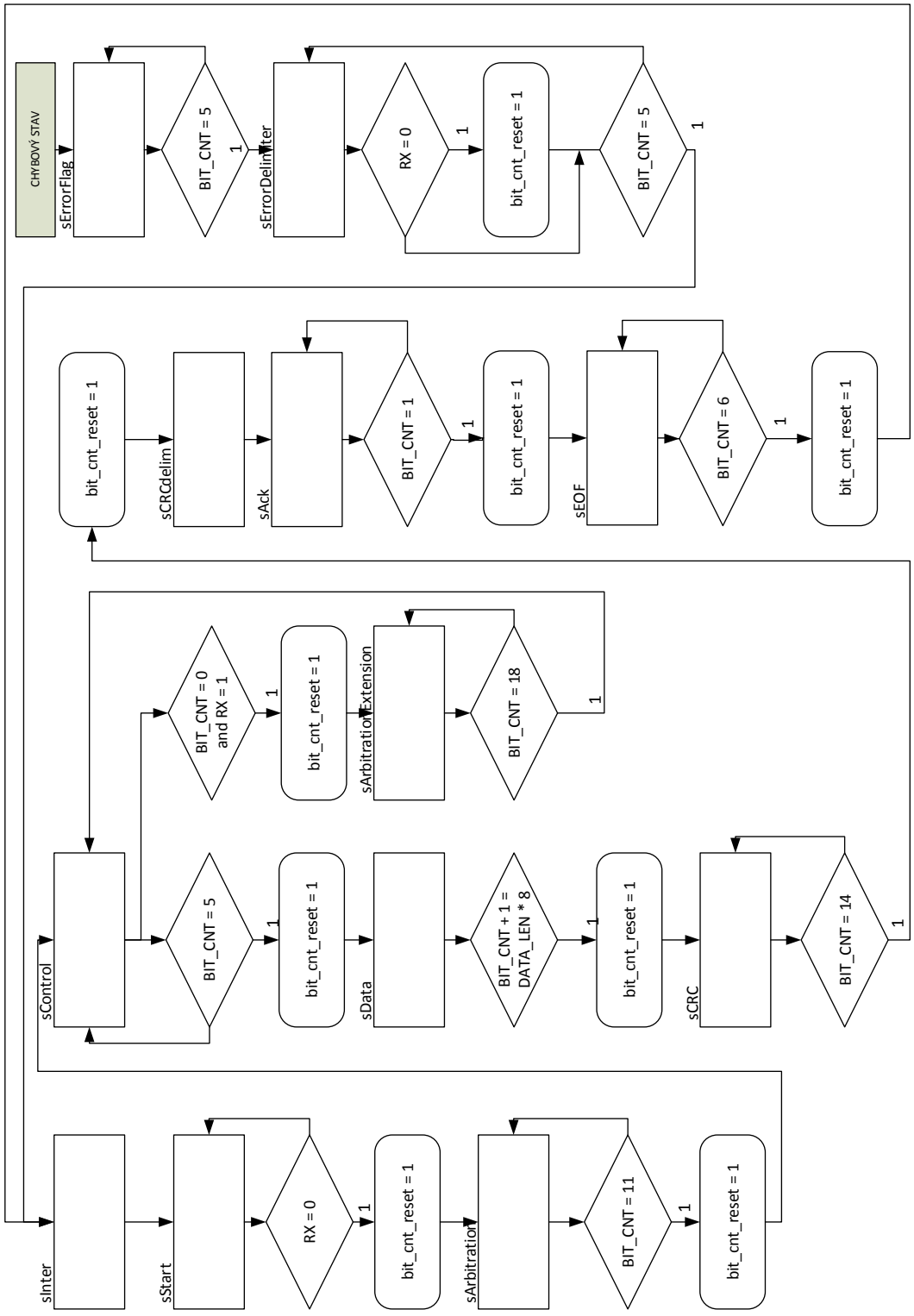
Príloha C

ASM diagramy stavových automatov

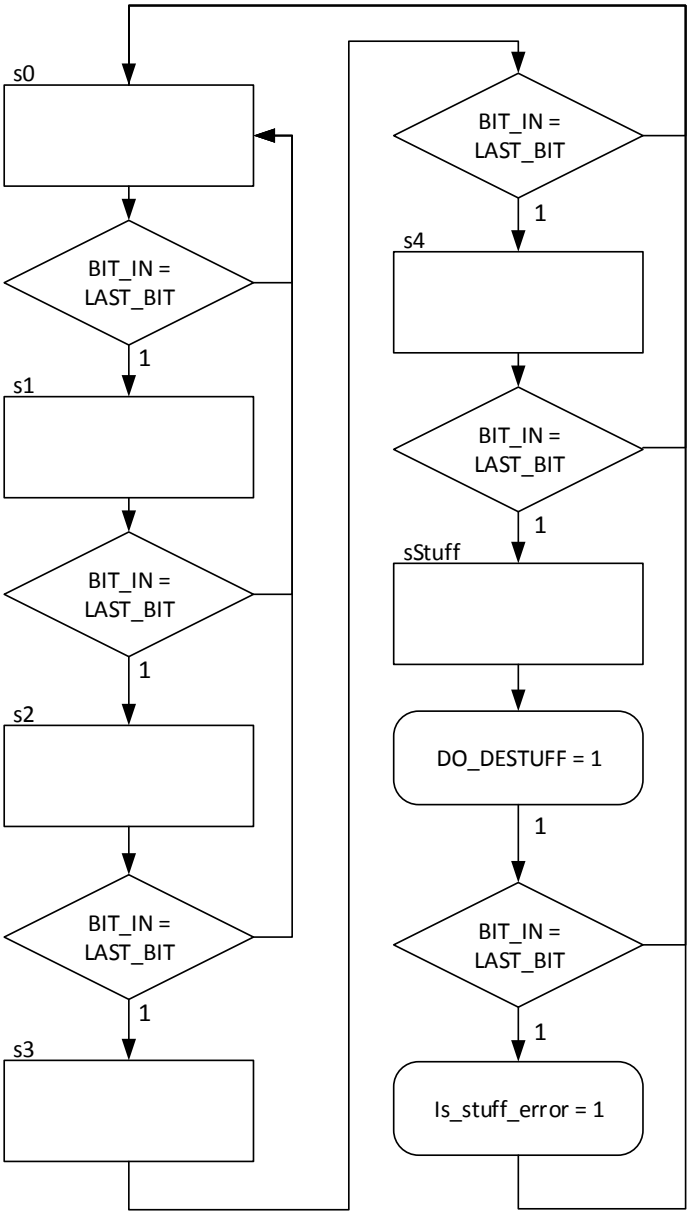
C.1 Jednotka vzorkovania zbernice



C.2 Stavová logika jadra CAN



C.3 Stavová logika bit stuffing-u



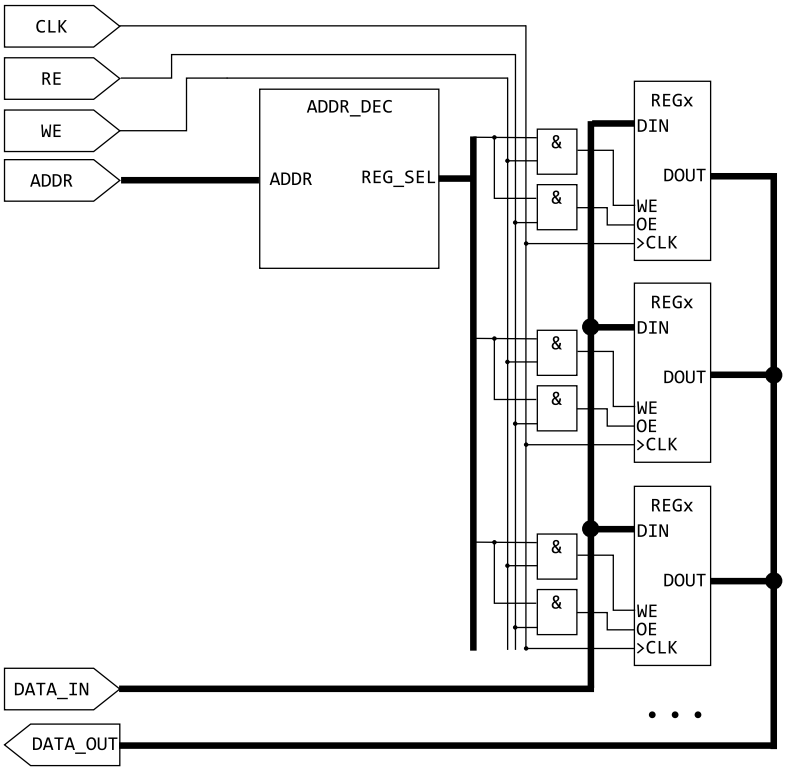
Príloha D

Organizácia sady registrov

Bity v registroch označené ako `R` sú určené na čítanie, operácia zápisu nie je definovaná. Bity označené `W` sú určené na zápis, operácia čítania nie je definovaná. Bity označené `R/W` sú určené na čítanie aj zápis.

offset	0000xxxx	0001xxxx	0010xxxx	0011xxxx
xxxx0000	RXMSGIDL	TXMSGIDEH	TPHASE2	F1MSGIDELM
xxxx0001	RXMSGIDH	TXCTRL	TSJW	F1MSGIDEHM
xxxx0010	RXMSGIDEL	TXDATA0	F0MSGIDL	REC
xxxx0011	RXMSGIDEH	TXDATA1	F0MSGIDH	TEC
xxxx0100	RXCTRL	TXDATA2	F0MSGIDEL	-
xxxx0101	RXDATA0	TXDATA3	F0MSGIDEH	-
xxxx0110	RXDATA1	TXDATA4	F0MSGIDLM	-
xxxx0111	RXDATA2	TXDATA5	F0MSGIDHM	-
xxxx1000	RXDATA3	TXDATA6	F0MSGIDELM	-
xxxx1001	RXDATA4	TXDATA7	F0MSGIDEHM	-
xxxx1010	RXDATA5	STATUSMASK	F1MSGIDL	-
xxxx1011	RXDATA6	STATUS	F1MSGIDH	-
xxxx1100	RXDATA7	ERROR	F1MSGIDEL	-
xxxx1101	TXMSGIDL	BAUD	F1MSGIDEH	-
xxxx1110	TXMSGIDH	TPROP	F1MSGIDLM	-
xxxx1111	TXMSGIDEL	TPHASE1	F1MSGIDHM	-

Tabuľka D.1: Mapa registrov



Obr. D.1: Spôsob zapojenia registrov

D.1 CAN_STATUS

Na modifikáciu jednotlivých bitov registra slúži maskovací register CAN_STATUSMASK.

bit	7	6	5	4	3	2	1	0
význam	ERR	RXMSG	-	LOOPB	HISPEED	SLEEP	SUSPEND	TXMSG
funckia	R	R	-	R/W	R/W	R/W	R/W	R/W

- TXMSG - zápis `1` spôsobí odoslanie správy
- SUSPEND - `1` - zastavenie funkcie radiča, `0` - normálna funkcia
- SLEEP - `1` - režim zníženého odberu transcieveru, `0` - normálna funkcia
- HISPEED - `1` - vysokorýchlostný režim transcieveru, `0` - normálny režim
- LOOPB - `1` - režim spätnej sľučky, `0` - normálny režim
- RXMSG - `1` - bola prijatá správa, `0` - nebola prijatá správa
- ERR - `1` - nastal chybový stav, `0` - nenastala žiadna chyba

D.2 CAN_RXMSGIDx

Nasledovné registre slúžia na uloženie identifikátora prijatej správy

- `RXMLSGIDL` - 8 najmenej významných bitov identifikátora správy
- `RXMLSGIDH`<0:2> - 3 najvýznamnejšie bity identifikátora správy
- `RXMLSGIDH`<3:7> - bity 11 - 15 rozšíreného identifikátora správy
- `RXMLSGIDEL`<0:7> - bity 16 - 23 rozšíreného identifikátora správy
- `RXMLSGIDEL`<0:5> - bity 24 - 29 rozšíreného identifikátora správy

D.3 CAN_RXCTRL

Register riadiaceho poľa správy

bit	7	6	5	4	3	2	1	0
význam	-	RTR	IDE	-	DLC3	DLC2	DLC1	DLC0
funckia	-	R	R	-	R	R	R	R

- DLC<0:3> - kód dĺžky správy
- IDE - `1` - správa s rozšíreným identifikátorom, `0` - správa so štandardným identifikátorom
- RTR - `1` - vzdialený rámec, `0` - dátový rámec

D.4 CAN_RXDATAx

Dátová časť prijatej správy je uložená v registroch `CAN_RXDATA0` až `CAN_RXDATA7` podľa dĺžky správy. Dáta sú zarovnané tak, že najnižší byte správy je uložený v registri `CAN_RXDATA0`.

D.5 CAN_TXMSGIDx, CAN_TXCTRL, CAN_TXDATAx

Registre na uloženie odchádzajúcej správy majú rovnakú štruktúru ako registre prijatej správy, v ich názve sa však nachádza predpona `TX` označujúca odchádzajúcu správu. Dátová časť správy musí byť zarovnaná tak, že najvyšší byte správy sa bude nachádzať v registri `CAN_TXDATA7`.

D.6 CAN_ERROR

Tento register obsahuje chybovú slabiku. Jednotlivé bity označujú prítomnosť zodpovedajúcej chyby.

- ESTUFF - chyba bit stuffing-u
- EFORM - porušenie formátu správy

bit	7	6	5	4	3	2	1	0
význam funckia	- -	EOFL R	EBUSOFF R	EPASV R	ECRC R	EACK R	EFORM R	ESTUFF R

- EACK - prijatie správy nebolo potvrdené
- ECRC - chyba kontrolného súčtu
- EPASV - zariadenie je chybovo-pasívne
- EBUSOFF - zariadenie je odpojené od zbernice
- EOFL - pretečenie príjmového buffera, správa bola zahodená

D.7 Registre časovania

- `CAN_BAUD` - nastavenie preddeličky
- `CAN_TPROP` - počet časových kvánt od začiatku bitu po koniec propagačného segmentu
- `CAN_TPHASE1` - počet časových kvánt od začiatku bitu po koniec prvého fázového segmentu
- `CAN_TPHASE2` - počet časových kvánt od začiatku bitu po koniec druhého fázového segmentu
- `CAN_TSJW` - maximálny počet časových kvánt, o ktorý sa posunie fáza pri synchronizácií

Symbolová rýchlosť sa vypočíta ako:

$$baudrate = \frac{f_{CLK}}{(\texttt{CAN\_BAUD} + 1)(\texttt{CAN\_TPHASE2} + 2)}$$

D.8 Registre filtrov FxMSGIDx

Filter sa skladá z registrov identifikátora `FxMSGIDL`, `FxMSGIDH`, `FxMSGIDEL`, `FxMSGIDEH` a zodpovedajúcej akceptačnej masky `FxMSGIDLM`, `FxMSGIDHM`, `FxMSGIDELM` a `FxMSGIDEHM`. Dostupné sú filter `F0` a `F1`.

D.9 Počítadlá chýb

- `CAN_REC` - počítadlo chýb príjmu
- `CAN_TEX` - počítadlo chýb odosielania

Zápis do týchto registrov spôsobí vynulovanie oboch počítadiel.