ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra měření

Implementácia nekonvenčného radiča CAN v jazyku VHDL

Implementation of unconventional CAN controller in VHDL

Diplomová práce

Studijní program: Otevřená informatika
Studijní obor: Počítačové inženýrství

Vedoucí práce: doc. Ing. Jiří Novák, Ph.D.

Bc. Dušan Hamza

Praha 2013

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Dušan Hamza**

Studijní program: Otevřená informatika (magisterský)
Obor: Počítačové inženýrství

Název tématu: **Implementace nekonvenčního řadiče CAN v jazyce VHDL**

Pokyny pro vypracování:

1. Navrhněte a implementujte IP funkci řadiče sítě CAN s podporou nadstandardních funkcí pro analýzu sítě.
2. Kromě běžných funkcí řadiče musí implementace podporovat odesílání z front dle časové značky zprávy, doplnění časové značky k přijatým zprávám a zpětný příjem odeslaných zpráv s časovou značkou skutečného odeslání. Velikosti všech front, šířka údaje časové značky a typ (datová šířka) rozhraní k nadřazenému procesoru musí být parametrizovatelné.

Seznam odborné literatury:

[1] Pinker, J., Poupa, M.: Číslicové systémy a jazyk VHDL, BEN 2006, ISBN 978-8073001988
[2] Etschberger, K.: Controller Area Network, IXXAT Automation 2001, ISBN 978-3000073762

Vedoucí: Doc.Ing. Jiří Novák, Ph.D.

Platnost zadání: do konce letního semestru 2013/2014

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 1. 2013

Abstrakt

Cieľom práce „Implementace nekonvenčního řadiče CAN v jazyce VHDL“ je vytvorenie obvodu s IP funkciou radiča zbernice CAN s nadštandardnou funkčnosťou v jazyku VHDL. Okrem bežnej funkčnosti radiča zbernice CAN náš obvod implementuje frontu prijatých správ a štyri fronty pre správy, ktoré majú byť odoslané. Navyše všetkým prijatým správam priraďuje časové značky a pre všetky správy, ktoré majú byť vyslané, je možné určiť, v akom čase sa majú vyslať na zbernicu CAN.

Abstract

The goal of work “Implementation of unconventional CAN controller in VHDL language” is to design a circuit with IP CAN bus controller function with above-standard functionality in VHDL language. In addition to the functioning of standard CAN bus controller our circuit implements queue of incoming messages and four queues for messages to be sent. Furthermore it assigns timestamps to all received messages and for all messages to be sent it is possible to determine at what time these messages are to be sent to the CAN bus.

Kľúčové slová

Zbernica CAN, jazyk VHDL, časové značky, programovateľné hradlové pole, radič, automobilová elektronika

Keywords

Controller Area Network, VHDL language, timestamps, field-programmable gate array, controller, automotive electronics

Prehlásenie

Prehlasujem, že som túto prácu vypracoval sám pod vedením doc. Ing. Jiřího Nováka, Ph.D. len s pomocou jeho konzultácií, konzultácií s Ing. Petrom Záleským a s pomocou uvedenej literatúry a zdrojov a že existencia rovnakej, či obdobnej práce mi nie je známa.

V Prahe dňa 10.5.2013

……

Dušan Hamza

Poďakovanie

V prvom rade by som chcel poďakovať doc. Ing. Jiřímu Novákovi, Ph. D. za jeho profesionálny prístup, cenné rady k mojej práci a k tomu že ma priviedol k automobilovej elektronike. Ďalej Ing. Petrovi Záleskému za konzultácie, Ing. Ivanovi Fridrichovi za zapožičanie mikrokontrolérov na účely testovania, Štěpánovi Sýkorovi za jazykovú korektúru, Michalovi Juránkovi za jeho bezodnú trpezlivosť v dňoch vzniku tejto práce a všetkým obyvateľom bloku X na Kolejích Strahov za spríjemnenie pracovnej atmosféry.

Použitá symbolika

V celom dokumente je používaná jednotná symbolika: Anglické termíny sú písané *kurzívou*, názvy funkčných blokov sú písané **tučne**, názvy signálov ***tučnou kurzívou*** a stavy automatov VEĽKÝMI PÍSMENAMI.

Obsah

1. Úvod 9
   1.1. Dôvody vzniku 9
   1.2. Požiadavky 9
2. Popis zbernice 10
3. Štruktúra rámcov 10
   3.1. Štandardný dátový rámec: 11
      Identifikátor správy (11 bitov): 12
      Príznak žiadosti o dáta (dominantný = log. 0): 12
      Príznak rozšíreného rámca (dominantný = log. 0): 12
      Dĺžka dát (4 bity): 13
      Dátové pole (0 - 64 bitov): 13
      Kontrolný súčet CRC: 13
      CRC oddeľovač, Acknowlegde a ACK oddeľovač: 13
      Koniec rámca (*End of Frame* - 7 bitov): 13
   3.2. Rozšírené dátové rámce: 14
      Príznak rozšíreného rámca (1 bit): 14
   3.3. Žiadosti o dáta: 14
   3.4. Chybové rámce: 15
   3.5. Preťažovacie rámce: 16
4. *Bit-stuffing* a synchronizácia hodín 16
5. Chybové čítače 18
   5.1. Aktívne uzly (*error-active*) 19
   5.2. Pasívne uzly (*error passive*) 19
   5.3. Odpojený uzol (*bus-off*) 19
6. Použité fyzické médium 20
7. Implementácia radiča 22
   7.1. Baud-Rate Prescaler 22

7.2. Časovač ... 22
7.3. Príjmač bitov ... 23
7.4. CRC (Cyclic Redundancy Check) ... 25
7.5. Prijímač rámcov ... 26
7.6. Akceptančné filtre ... 30
7.7. Vysielač rámcov ... 32
7.8. FIFO ... 37
7.9. Registre ... 38
7.10. Potvrdzovač ... 40
7.11. Celkové zapojenie ... 41
8. Realizácia a testovanie obvodu ... 44
9. Záver ... 47
10. Tabuľky ... 48
10.1. Tabuľka rozdelenia 99-bitového signálu paket ... 48
10.2. Tabuľka rozdelenia užívateľsky nastaviteľných registrov ... 48
10.3. Podrobný popis registrov, združujúcich viac nastavení ... 50
11. Zoznam skratiek a značiek ... 50
12. Odkazy a referencie ... 51

1. Úvod

Moja práca "Implementácia nekonvenčného radiča CAN v jazyku VHDL" vznikla za účelom odľahčenia a zefektívnenia práce mikrokontroléra pri obsluhe CAN zbernice. Implementuje hardwarový radič zbernice CAN, ktorý však okrem štandardnej funkčnosti radičov bežne dostupných na trhu implementuje navyše frontu prijatých a odosielaných správ spolu s ich časovými značkami.

1.1. Dôvody vzniku

V osemdesiatych rokoch dvadsiateho storočia sa v automobiloch začalo používať čoraz viac elektronických zariadení. Dôvodom bolo najmä zvyšovanie bezpečnosti, spoľahlivosti a komfortu, čo so sebou prinášalo zvyšovanie zložitosti a počtu komponentov. V automobiloch dramaticky vzrástol počet a dĺžka drôtov, čo paradoxne začalo prinášať opačný efekt – nárast poruchovosti, potreba dedikovaných vodičov pre každý druh komunikácie a v neposlednom rade zvyšovanie celkovej hmotnosti automobilov. Toto všetko začalo tlačiť na výrobcov, aby celý tento systém zjednodušili, odľahčili na hmotnosti, zvýšili spoľahlivosť a životnosť elektronických komponentov. Výrobcovia potrebovali vyvinúť spôsob, akým budú jednotlivé komponenty v automobile medzi sebou komunikovať po jednej spoločnej zbernici.

Odpoveďou bol prvý štandard zbernice CAN od firmy Robert Bosch GmbH vyvinutý v rokoch 1983 – 1986. Prvý krát bol komerčne nasadený v automobile spoločnosti Bayerische Motoren Werke AG v modeli 850 Coupé v roku 1986. Tento automobil bol prevratný v mnohých oblastiach – prvýkrát v ňom bola použitá šesťstupňová manuálna prevodovka s motorom V12. Rovnako bol jeden z prvých, ktoré mali elektronický plynový pedál (*throttle-by-wire*) a boli navrhované s použitím CAD softvérov. Po prvý krát začali senzory, akčné členy a riadiace obvody komunikovať po dvoch drôtoch vysokou rýchlosťou. Odhaduje sa, že použitím zbernice CAN sa ušetrili dva kilometre drôtov, čo zodpovedá asi päťdesiat kilogramov hmotnosti a navyše sa ušetrila približne polovica konektorov.

1.2. Požiadavky

Aby bolo možné zaistiť bezpečnú a real-time komunikáciu medzi všetkými elektronickými komponentmi, určil automobilový priemysel uvedené požiadavky na komunikačný štandard:

- real-time
- čo najmenší počet vodičov
- schopnosť vysporiadať sa s nasledovnými poruchovými faktormi:
    - zlyhanie alebo odpojenie jedného alebo viacerých uzlov
    - elektromagnetické rušenie

- prerušenie jedného z vodičov
- pripojenie jedného z vodičov na kostru automobilu, kladný pól batérie, +5V

Uvedené technické požiadavky boli spracované a výsledkom bol prvý štandard zbernice CAN vyvinutý firmou Robert Bosch GmbH v rokoch 1983 - 1986. Využitie napokon našiel aj v iných oblastiach ako automobily, napríklad aj v oblasti priemyslovej automatizácie, niekde dokonca vytlačil dovtedy zaužívané štandardy ako SPI, I2C. Pre jeho real-timové vlastnosti siahli po protokole CAN dokonca niektorí výrobcovia medicínskych prístrojov.

2. Popis zbernice

Štandard CAN (ISO11898-1) umožňuje medzi sebou prepojiť až 30 zariadení bez potreby akéhokoľvek centrálneho uzla, čo znižuje riziko zlyhania práve tohto nadradeného prvku sústavy (siete). Jedná sa teda o zbernicu typu multi-master, kde všetky uzly sú si medzi sebou rovnocenné. Zbernica sa navyše dokáže vyrovnať so zlyhaním jedného, alebo aj viacerých uzlov.

Protokol CAN je založený na princípe posielania správ medzi uzlami po sériovej komunikačnej zbernici. Správy môžu byť vysielané každým uzlom vždy, pokiaľ je zbernica voľná (v kľudovom stave). Všetky správy sú doručované všetkým uzlom pripojeným k zbernici. Norma ISO11898-1 navyše vyžaduje, aby fyzické médium implementovalo logický súčin AND. V praxi to znamená, že ak vysiela čo len jeden uzol úroveň označenú ako *dominant* (v metalickom vedení log. 0, v optickom vedení "svieti"), môže ľubovoľný počet ďalších uzlov súčasne vysielať úroveň *recessive* (v metalickom vedení log. 1, v optickom vedení "nesvieti") a na zbernici sa aj tak v konečnom dôsledku objaví úroveň *dominant* (0 - svieti). Na začiatku vysielania každej správy uzol vyšle na zbernicu identifikačné číslo správy. V prípade, že naozaj nastane situácia, kedy chce v jednom okamihu vysielať viac uzlov, na zbernici sa objaví aj tak iba správa s najnižším identifikačným číslom. Z tohto vyplýva, že správy s vyššou prioritou majú nižšie identifikačné číslo, než správy s nižšou prioritou. Štandard CAN rovnako popisuje, že ihneď, ako vysielajúci uzol zaznamená, že na zbernici je v skutočnosti iná logická úroveň, než on sám vysiela, musí okamžite prerušiť svoje vysielanie. Svoju správu sa potom musí pokúsiť poslať ihneď, ako to bude možné (v najbližšom ďalšom kľudovom stave zbernice). Takto je zabezpečená tzv. arbitráž zbernice, čiže akási súťaž o právo vysielať.

3. Štruktúra rámcov

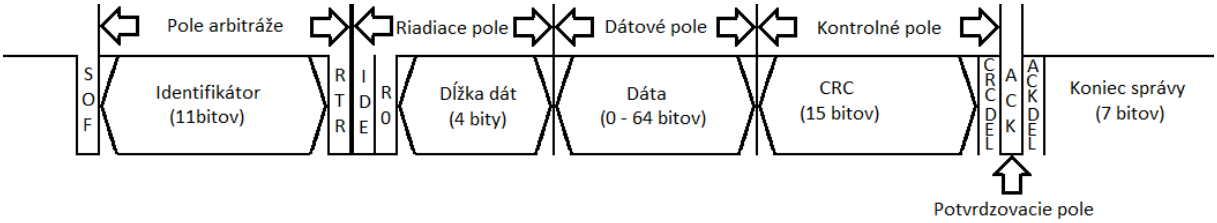
Jednotlivé správy (rámce) sú oddeľované medzirámcovými medzerami. Za medzirámcovú medzeru sa považujú aspoň 3 za sebou idúce recesívne (log. 1) bity. Navyše na konci každej správy je oddeľovač správ (7 recesívnych bitov – log. 1). Po uplynutí tohto času môžu všetky uzly začať znovu súťažiť o právo vysielať, pretože sa toto považuje za kľudový stav zbernice. Každá správa začína

jedným dominantným bitom (log. 0) označovaným v literatúre ako SOF (*Start of Frame*). Na začiatku správy je teda vždy dobežná hrana, ktorá spôsobí tvrdú synchronizáciu hodín (o nej podrobnejšie v kapitole Bit Stuffing a synchronizácia hodín) všetkých ostatných prijímačov a zapne v nich pravidlo vkladania bitov (*bit-stuffing*).

Rámce môžu mať 5 podôb:

- Normálny dátový rámec – *Standard Data Frame*
- Rozšírený dátový rámec – *Extended Data Frame*
- Žiadosť o dáta – *Remote Transmission Request (standard, extended)*
- Preťaženie – *Overload Frame*
- Chyba – *Error Frame*

3.1. Štandardný dátový rámec:



Začína štandardne jedným dominantným bitom SOF, ktorý zapína *bit stuffing*. Za ním nasleduje pole arbitráže, samotné dáta rámca (bytovo zarovnané), ďalej kontrolné a nakoniec potvrdzovacie pole.

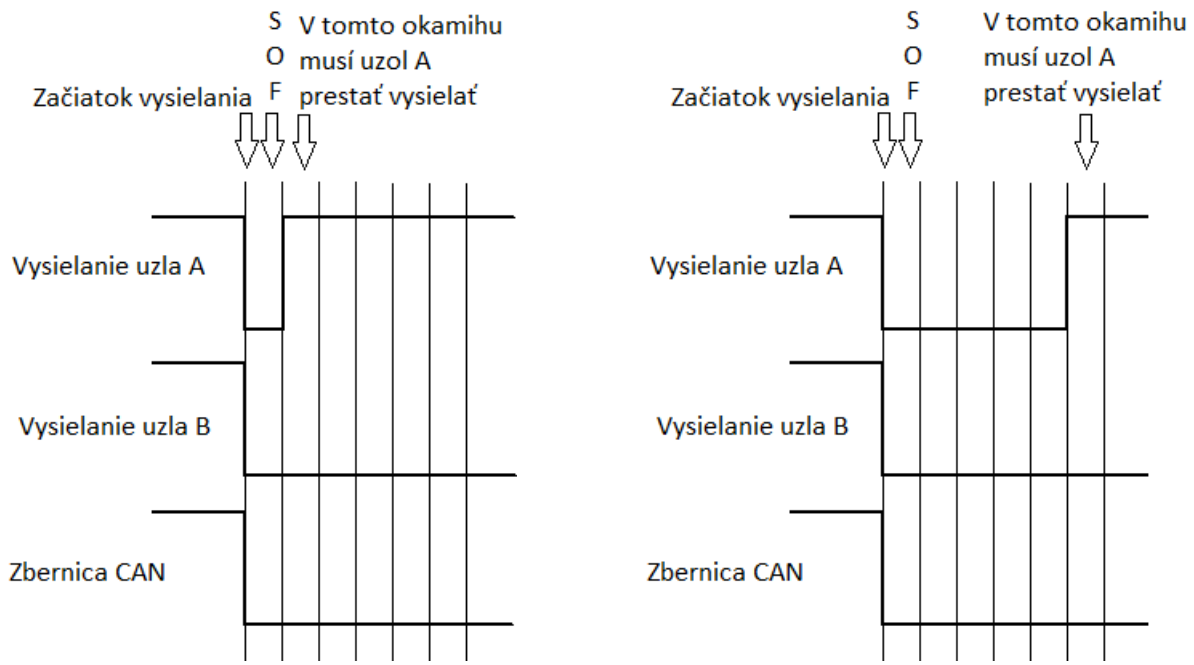
1. **Arbitrážne pole (12 bitov)**
   a) Identifikátor správy - ID - určuje jej prioritu - 11 bitov
   b) Žiadosť o dáta - *Remote Transmission Request* - 1 bit (dominantný = log. 0)
2. **Riadiace pole (6 bitov)**
   a) Rozšírený rámec – *Extended Frame* - 1 bit (dominantný = log. 0)
   b) Rezervovaný bit - pre prípadné budúce využitie - 1 bit (dominantný = log. 0)
   c) Dĺžka dát - počet bytov v dátovom poli rámca
3. **Dátové pole (0 – 64 bitov)**
   a) Bytovo zarovnané dáta rámca
4. **Kontrolné pole (16 bitov)**
   a) Kontrolný súčet CRC (ODKAZ) - 15 bitov
   b) Oddeľovač kontrolného súčtu - 1 bit (recesívny = log. 1)
5. **Potvrdzovacie pole (1 bit)**
   a) Potvrdenie - 1 bit (dominantný = log. 0)
6. **Koniec rámca a medzirámcová medzera (11 bitov)**
   a) Oddeľovač potvrdenia - 1 bit (recesívny = log. 1) - vypína *bit stuffing*

b) Koniec rámca – *End of Frame* - EOF - 7 bitov (recesívne = log. 1)

c) Medzirámcová medzera - 3 bity (recesívne = log. 1)

Identifikátor správy (11 bitov):

Počas jeho vysielania uzly zároveň súťažia o právo vysielať. Ak počas vysielania identifikátora ktorýkoľvek uzol zistí, že na zbernici je iná logická úroveň, než sám vysiela, musí okamžite prerušiť svoje vysielanie a vyčkať s vyslaním svojho rámca do najbližšieho kľudového stavu zbernice. Identifikátor je sekvencia 11 bitov. Ako príklad uvedieme nasledovné dve situácie - obidve ilustrujú situáciu, kedy chcú dva uzly vysielať súčasne.



Príznak žiadosti o dáta (dominantný = log. 0):

Počas vysielania tohto bitu stále platí, že ak vysielajúci uzol zaznamená na zbernici inú hodnotu, než sám vysiela, musí prerušiť svoje vysielanie a čakať, kým zbernica znova neprejde do kľudového stavu. V štandardnom, ako aj v rozšírenom rámci, má tento bit hodnotu dominant (log. 0). Ak však iný uzol vysiela žiadosť o dáta s rovnakým identifikátorom, tento bit je v recesívnej úrovni (log. 1). Ak teda začnú vysielať naraz dva uzly, uzol A vyšle rovnaký identifikátor ako uzol B, avšak uzol A posiela žiadosť o dáta a uzol B posiela práve tieto dáta, na zbernici sa objaví iba rámec s dátami a rámec so žiadosťou musí počkať. O žiadostiach o dáta podrobnejšie v kapitole žiadosť o dáta.

Príznak rozšíreného rámca (dominantný = log. 0):

V štandardnom dátovom rámci musí tento bit mať úroveň dominant. V prípade, že má tento bit úroveň recesívnu úroveň, jedná sa o tzv. rozšírený rámec.

Dĺžka dát (4 bity):

Tieto 4 bity informujú prijímač o dĺžke dát, ktoré budú nasledovať bezprostredne za týmito 4 bitmi. Udáva počet bytov v dátovom poli správy. Povolené hodnoty sú medzi 0 bytov až 8 bytov. Štvorbitové pole dĺžky dát však umožňuje nastaviť dĺžku dát až na 15 bytov. V prípade, že v prijímanej správe je informácia o počte bytov väčšia než 8, prijímač prijme aj tak iba prvých 8 bytov správy.

Dátové pole (0 - 64 bitov):

Obsahuje samotné dáta správy. Tieto dáta musia byť bytovo zarovnané. Norma ISO11898-1 neurčuje endianitu týchto dát, čo v praxi znamená, že je na programátorovi, aké poradie jednotlivých bytov si zvolí.

Kontrolný súčet CRC:

Je vypočítaný z celej správy (vrátane SOF) bez bitov doplnených pravidlom vkladania bitov (*bit-stuffing*). Znamená to, že každý bit (vrátane SOF) rámca je spracovaný CRC obvodom až po koniec dátového poľa a porovnaný s CRC poľom na konci správy. V prípade, že sa tieto dve hodnoty líšia, je toto považované za chybu. Štandard CAN predpisuje, aby sa CRC počítalo podľa nasledovného polynómu:

$$\mathbf{x^{15} + x^{14} + x^{10} + x^{8} + x^{7} + x^{4} + x^{3} + x^{0}}$$

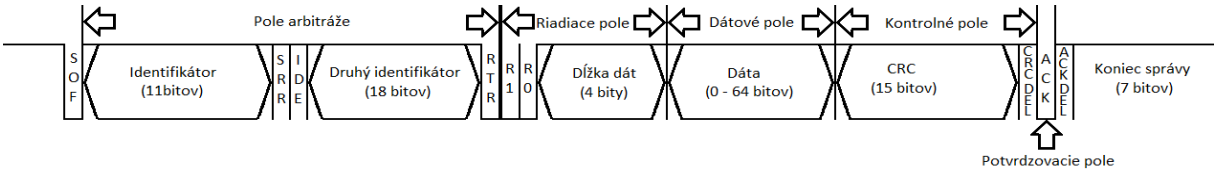
CRC oddeľovač, Acknowlegde a ACK oddeľovač:

Táto skupina bitov má funkciu potvrdzovania a kontroly správnosti prenesenia správy. CRC DEL a ACK DEL musia byť recesívne (log. 1), aby vysielač spoznal, že zbernicu neblokuje žiadny iný uzol, napríklad chybovým rámcom, prípadne z dôvodu inej chyby zbernice. V čase ACK vysielač vysiela recesívny bit (log. 1) a každý uzol, ktorý porozumel vysielanej správe, vyšle na zbernicu dominantný bit (log. 0). V prípade, že vysielač vyšle správu a nikto ju nepotvrdí dominantným bitom (log. 0), je toto považované za chybu.

Koniec rámca (*End of Frame* - 7 bitov):

Rovnako ako CRC DEL a ACK DEL, tak aj skupina siedmich bitov EOF musí byť striktne recesívna (log. 1), v opačnom prípade je toto považované za chybu. Za touto skupinou bitov nasleduje ešte medzirámcová medzera. Keďže EOF je spolu s medzirámcovou medzerou je recesívna (log. 1), všetky ostatné uzly toto považujú za voľnú zbernicu a teda pre seba aj právo vysielať.

3.2. Rozšírené dátové rámce:



Zhodujú sa so štandardnými rámcami vo všetkých poliach, okrem arbitrážneho. Toto pole v nich začína štandardným, 11 bitovým, identifikátorom. Za ním však nasleduje bit SRR (*substitute remote request*), ktorý by mal byť vyslaný v recesívnej úrovni (log. 1). Ak je však tento bit prijatý ako dominantný (log. 0 - napríklad kvôli chybe zbernice), nespôsobí tento stav chybu prijímača. Avšak túto chybu neignoruje *bit-stuffing* ani arbitráž zbernice.

Príznak rozšíreného rámca (1 bit):

Tento bit určuje, že rámec je rozšírený a nie štandardný. Rozhoduje však aj o význame predchádzajúceho bitu (SRR). Z tohto vyplýva, že prijímač až do okamihu prijatia tohto bitu nevie určiť význam predošlého bitu.

Hodnoty bitu:

Úroveň bitu	Druh rámca	Význam predchádzajúceho bitu
Dominantná	Štandardný	RTR
Recesívna	Rozšírený	SRR

V prípade, že bit EXT bol vyslaný v recesívnej úrovni (jedná sa o rozšírený rámec), všetky uzly naďalej súťažia o právo vysielať. Doba súťaženia (arbitráže) sa tak predlžuje o ďalších 18 bitov rozšíreného identifikátora správy. Navyše, rovnako, ako v štandardnom dátovom rámci, je aj v rozšírenom dátovom rámci arbitrážne pole ukončované ešte jedným bitom RTR.

3.3. Žiadosti o dáta:

Remote Transmission Request - vo väčšine aplikácii využívajúcich zbernicu CAN touto zbernicou komunikujú najrôznejšie senzory, akčné členy a riadiace mikrokontroléry. Tieto svoje dáta vysielajú zväčša autonómne, bez vonkajšieho príkazu. Napriek tomu ma zbernica CAN mechanizmus, ktorý dokáže získať dáta v prípade, že je to z nejakého dôvodu nutné. Táto žiadosť sa nazýva *Remote Transmission Request* a rovnako, ako dátové rámce, môže mať aj táto žiadosť 2 podoby - základnú a rozšírenú. V mnohom sa zhoduje s dátovými rámcami, či už sa jedná o jednu, alebo druhú variantu. Žiadosť o dáta s určitým identifikátorom má presne rovnaký identifikátor, ako žiadané dáta. Rámec žiadosti má však 2 zásadné rozdiely oproti nim: Dĺžka dát je vždy nula bytov, čo znamená, že DLC

(dĺžka dát) je “0000“ a dátové pole rámca chýba úplne. Druhý podstatnejší rozdiel je však v úrovni, akou je vyslaný RTR bit umiestnený na konci arbitrážneho poľa. Spomínaný bit je totiž vysielaný v recesívnej úrovni (log. 1). A toto je dôvod, prečo ak sa na zbernici náhodou objavia dáta a žiadosť o práve tieto dáta, na poslednú chvíľu (až v okamihu vysielania posledného bitu arbitrážneho poľa – RTR bit) zvíťazí v súťaži o právo vysielať uzol vysielajúci práve tieto dáta a uzol vysielajúci žiadosť prehrá. Týmto pádom prestane vysielať a prijme rovno žiadané dáta.

3.4. Chybové rámce:

V prípade, že ktorýkoľvek uzol zistí jednu, alebo viac z nasledovných chýb, okamžite (počas vysielania rámca - bez čakania koniec rámca) vygeneruje chybový rámec (*Error Frame*). Možné príčiny chýb na zbernici CAN:

- Chyba povinných bitov (ACK DEL, CRC DEL, EOF) - musia byť striktne recesívne (log. 1)
- Chyba CRC
- Chyba pravidla vkladania bitov (*bit-stuffing error*)

Právo každého uzla generovať chybové rámce určujú jeho dva vnútorné čítače chýb - odosielacích a prijímacích. O týchto čítačoch podrobnejšie v kapitole chybové čítače. Tieto čítače má každý uzol vlastné, preto každý uzol môže generovať, resp. negenerovať iný typ chyby. Každý uzol si sám na základe stavu týchto čítačov určuje, či vôbec, a aké chybové rámce vysiela, v prípade, že zistí chybu. Môžu byť v jednom z troch stavov:

- Aktívne - môžu vysielať aktívne chybové rámce
- Pasívne - môžu vysielať pasívne chybové rámce
- Odpojené - nemôžu vysielať chybové rámce

Keďže sa chybové rámce skladajú z minimálne šiestich bitov rovnakej úrovne, ich vyslanie spôsobí vo všetkých ostatných uzloch chybu pravidla vkladania bitov (*bit-stuffing*). Ostatné uzly preto vygenerujú svoje chybové rámce, ako tzv. chybovú ozvenu (*Error Echo flag*), zbernica sa vráti do kľudového stavu a uzol, ktorého vysielanie bolo prerušené sa pokúsi svoju správu poslať opätovne.

Aktívne chybové rámce sa skladajú zo šiestich až dvanástich dominantných bitov (log. 0). To znamená, že bez ohľadu na vysielanie sa zbernica ocitne v dominantnom stave (log. 0) po dobu minimálne šiestich bitov. Chybový rámec je potom ukončený ôsmimi recesívnymi bitmi (log. 1), za ktorými nasledujú ďalšie 3 recesívne bity (log. 1), označované ako medzirámcová medzera, prípadne môže byť miesto tejto medzery nasledovaný preťažovacím rámcom.

Pasívne chybové rámce sa skladajú z 14 recesívnych bitov (log. 1). Ak totiž uzol často prerušuje vysielanie svojimi aktívnymi rámcami, sám prejde do pasívneho chybového stavu, aby

nerušil ďalšiu komunikáciu na zbernici. V podstate jeho chybové vysielanie na zbernici zaniká. Jedinou výnimkou, kedy môže pasívny uzol upozorniť ostatné uzly na chybu je, ak počas vysielania svojho rámca náhle vyšle pasívny chybový rámec. V takomto prípade musí čakať na 6 za sebou idúcich recesívnych bitov (log. 1), ktoré väčšinou ešte predchádza chybová ozvena (6 - 12 dominantných bitov – log. 0) ostatných uzlov ako reakcia na práve vyslaný chybový rámec.

Odpojené uzly majú vypnuté výstupné budiče, takže ani pri zistení chyby negenerujú chybové rámce. Do tohto stavu uzol prechádza v prípade, že vygeneroval príliš veľké množstvo chýb.

3.5. Preťažovacie rámce:

V prípade, že ktorýkoľvek uzol zbernice po prijatí rámca vie, že prípadnú ďalšiu správu nebude schopný prijať ihneď, vyšle tzv. preťažovací rámec (*overload frame*), čím narúša recesívny kľud na zbernici (log. 1) považovaný za medzirámcovú medzeru. Za týmito dominantnými (log. 0) bitmi potom nasleduje 7 recesívnych bitov (log. 1) slúžiacich na oddelenie od nasledovného vysielania. Tým môže byť buď medzirámcová medzera (3 recesívne bity – log. 1) nasledovaná iným rámcom, alebo ďalším preťažovacím rámcom. Takéto rámce však môže uzol vyslať maximálne dva za sebou. Keď ostatné uzly zistia, že na zbernici sa objavil takýto preťažovací rámec, vyšlú ho rovnako aj ony svoj preťažovací rámec. Chybové rámce môžu byť vyslané iba počas vysielania iných rámcov, naproti tomu preťažovacie rámce (aj keď majú rovnakú štruktúru) môžu byť vysielané iba počas medzirámcovej medzery. To v praxi znamená po odvysielaní oddeľovača chyby, oddeľovača preťaženia, konca rámca, alebo v medzirámcovej medzere. Tu však platí jedno obmedzenie, a síce, že počas vysielania prvých dvoch bitov medzirámcovej medzery je dominantný bit (log. 0) skutočne považovaný za začiatok preťažovacieho rámca. Ak je však prvý dominantný (log. 0) bit vyslaný počas tretieho bitu medzirámcovej medzery, ostatné prijímače toto považujú už za začiatok ďalšieho dátového rámca (SOF).

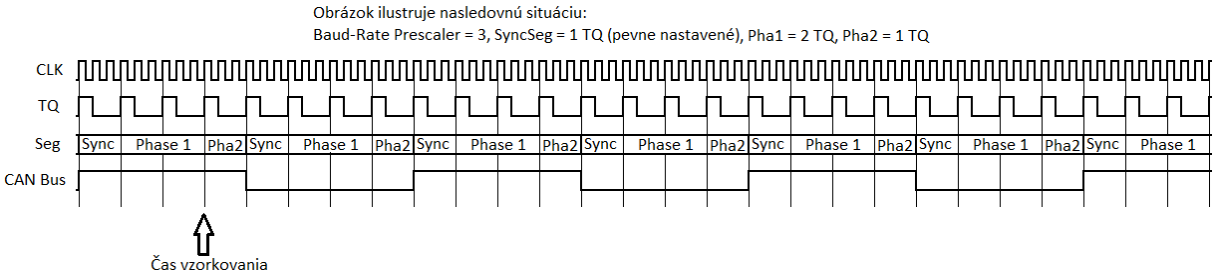
V dnešnej dobe sa však preťažovacie rámce na CAN zbernici prakticky nevyskytuje. Hlavným dôvodom je najmä pokrok v technológii dnešných radičov, ktoré preto ani tento druh rámca nevysielajú. Musia ho však tolerovať a svoje prípadné vysielanie pozdržať.

4. *Bit-stuffing* a synchronizácia hodín

Každý uzol si musí sám synchronizovať svoje hodiny. Z tohto dôvodu sa zaviedla do štandardu CAN synchronizácia hodín prijímača a to vždy na dobežnú hranu (pri prechode zbernice z recesívneho stavu do dominantného, čiže zo stavu log. 1 do stavu log. 0). Navyše, aby sa predišlo tomu, že sa hodiny jednotlivých uzlov navzájom rozsynchronizujú, vysielač vždy po piatich vyslaných bitoch rovnakej polarity vloží do komunikácie jeden bit s opačnou polaritou. Ostatné uzly ho ignorujú, iba si na ňom znovu zladia svoje hodiny dohromady. Pri zvážení tohto veľkého rizika vzniku chýb v rušnom prostredí automobilu vznikol systém časovania bitov, rozdeľujúci čas vysielania každého

bitu na tri časové úseky (segmenty). Ich dĺžku je možné nastaviť v každom radiči CAN zbernice. A práve správnou voľbou dĺžky trvania spomenutých časových segmentov dokážeme celú sieť nastaviť na určené použitie pre konkrétnu aplikáciu. Programátor môže ovplyvniť rýchlosť zbernice a čas, v ktorom sa vzorkujú jednotlivé bity.

V každom radiči je obsiahnutý tzv. *Baud-Rate Prescaler*. V tomto obvode je možné nastaviť, koľko taktov vnútorného oscilátora má trvať jedno časové kvantum (*Time Quantum* – TQ). V radiči je ďalej možné nastaviť jednotlivé segmenty trvania jedného bitu podľa uvedeného obrázku:



Dĺžka Sync segmentu je pevne nastavená na 1 časové kvantum. Nastaviť sa však dá, z koľkých časových kvánt sa skladajú segmenty Phase 1 a Phase 2 a to, či má prijímač vzorkovať každý bit len raz (na konci segmentu Phase 1), alebo trikrát (posledné 3 časové kvantá segmentu Phase1). Ak je v prijímači zapnutá voľba trojnásobného vzorkovania bitov, potom sa výsledná navzorkovaná hodnota zbernice vyhodnocuje majoritnou (väčšinovou) logikou. Navyše je možné nastaviť tzv. SJW (*Synchronization Jump Width* - maximálna šírka synchronizačného skoku), čo znamená maximálny počet časových kvánt, o ktoré sa môžu resynchronizovať hodiny prijímača. Samotná synchronizácia hodín sa potom môže diať dvomi spôsobmi, jedine však pri dobežnej hrane (pri prechode zbernice z recesívneho stavu do dominantného, čiže zo stavu log. 1 do stavu log. 0):

- Ak príde dobežná hrana počas trvania segmentu Phase1, predĺži sa tento časový úsek (maximálne však o SJW časových kvánt) a prijímač čaká dlhšie na navzorkovanie príslušného bitu.
- Ak však príde dobežná hrana počas trvania segmentu Phase2, skráti sa tento segment (opäť však maximálne o SJW časových kvánt), aby obvod mohol začať prijímať nasledujúci bit.

Doplňujúce požiadavky na nastavenie registrov časových kvánt:

SEG1 ≥ PS2 ≥ SJW

SEG1 ≥ čas propagácie

5. Chybové čítače

Nahradenie technológie dedikovaných vodičov pre každý typ komunikácie sériovou zbernicou je už vo svojej podstate spojené s problémom blokovania celého systému jediným chybným uzlom. Navyše k tomuto prispieva čiastočne aj samotný spôsob signalizácie chýb zbernice CAN. Keby chybný uzol mohol neustále generovať aktívne chybové rámce (6 dominantných bitov – log. 0), znemožnil by akýkoľvek iný prenos na zbernici. Aby sa predišlo uvedenému scenáru, má zbernica CAN mechanizmus rozpoznania chybného uzla a jeho následného odpojenia od zbernice. Tento mechanizmus musí mať každý uzol vlastný, čiže každý uzol si sám určuje, či to náhodou nie je práve on, kto je chybný. Štandard CAN predpisuje, aby na tento účel mal každý uzol vlastné čítače chýb – jeden počíta chyby pri odosielaní, druhý pri prijímaní. Ich hodnoty sú zvyšované buď o 1 alebo 8 s každým chybne označeným príjmom resp. odoslaním rámca a znižované o 1 s každým úspešne prijatým resp. odoslaným rámcom. V prípade chyby je hodnota zvýšenia závislá na tom, či je uzol prvý, kto zistil aktuálnu chybu, alebo nie. Spomenutý postup môže viesť k značnému zvyšovaniu chybových čítačov v krátkom časovom úseku, avšak počty chýb sú opätovne znižované počas úspešnej dlhšie trvajúcej bezchybnej prevádzky. Tieto chybové čítače sú teda akýmsi ukazovateľom chybovosti prenosov na zbernici.

Ak uzol zaznamená chybu, ktorú zaznamenal ako jediný z celej zbernice, vygeneruje chybový príznak (buď aktívny, alebo pasívny, každopádne však 6 bitový), za ktorým vyšle ešte 8 recesívnych bitov. Týmto vyvolá v ostatných uzloch chybu *bit-stuffing*-u a prinúti ich vygenerovať vlastné chybové rámce. Po vyslaní vlastného chybového príznaku, v prvom bite vysielania jeho následného recesívneho oddeľovača zisťuje, či je zbernica v dominantnej (log. 0), či recesívnej (log. 1) úrovni. Ak je v recesívnej (log. 1) úrovni, tak všetky uzly už ukončili vysielanie svojich chybových rámcov, čiže v okamihu zistenia chyby ju zistili aj ostatné uzly. Ak je však naďalej v dominantnej (log. 0) úrovni, znamená to, že ostatné uzly nezistili pôvodnú chybu, ale reagujú však na nami vyslaný chybový rámec chybovou ozvenou. Z tohto uzol rozozná, či pôvodnú chybu zaznamenal iba on (bol prvý, kto ju zaznamenal), alebo ju zachytili aj ostatné uzly. Ak uzol zaznamená chybu, zvýši príslušný čítač o 1. Ak však po odvysielaní chybového rámca zistí, že bol prvý, kto túto chybu zaznamenal, zvýši príslušný čítač o ďalších 8. Týmto sa zabezpečí, že ak uzol opakovane zisťuje chyby ako prvý, začne sám seba považovať za chybný. V prípade, že aspoň jeden z čítačov narastie nad hodnotu 127, uzol sa sám prepne do pasívneho chybového režimu, čím zastaví svoje aktívne chybové vysielanie a tým rušenie zbernice a všetkých ostatných uzlov. Ak navyše čítač chýb odosielania dovŕši hodnotu 255, uzol prejde až do stavu „*bus-off*“, čím sa zamedzí akejkoľvek ďalšej interakcii so zbernicou (vysielanie akýchkoľvek rámcov a potvrdzovacích bitov).

5.1. Aktívne uzly (*error-active*)

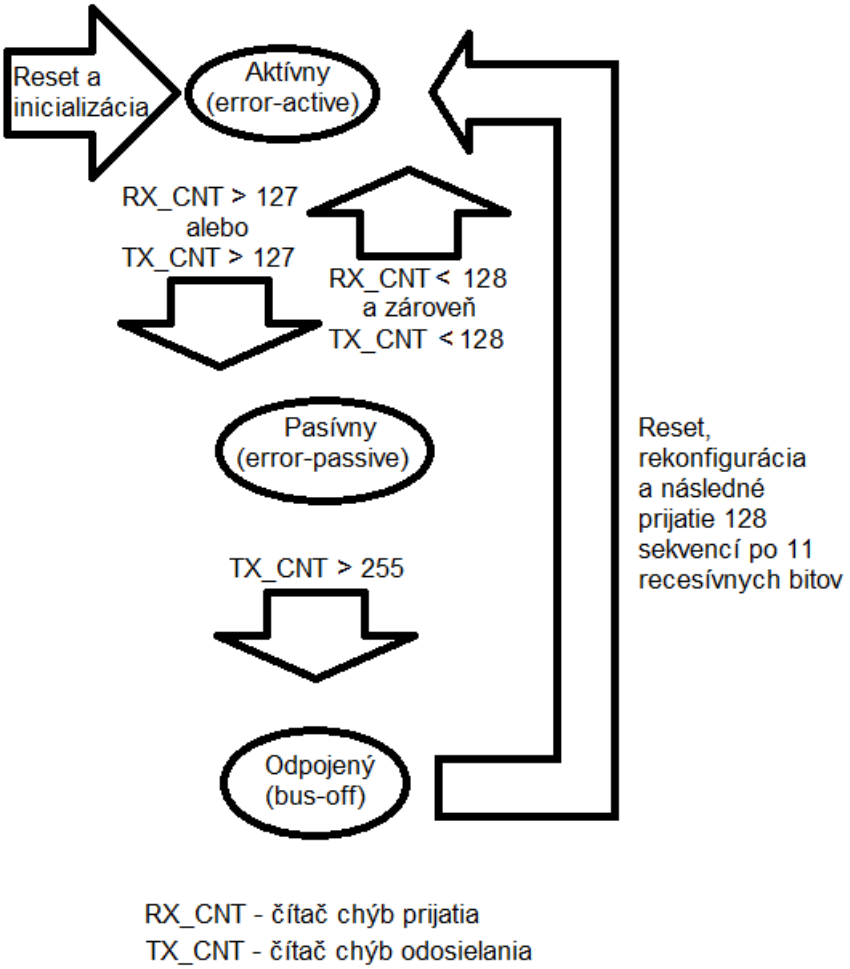
sa podieľajú na komunikácii po zbernici a posielajú aktívne chybové rámce v prípade zistenia chyby. Týmto aktívne rámce „zničia“ rámce, ktoré práve vysielajú tým, že počas ich vysielania začnú vysielať aktívny chybový rámec (6 dominantných bitov – log. 0). V ostatných uzloch týmto svojím vysielaním porušia pravidlo vkladania bitov (*bit-stuffing*) a teda znemožnia im prijať tento porušený rámec.

5.2. Pasívne uzly (*error passive*)

Do tohto chybového stavu prechádzajú uzly, ktoré už nahromadili relatívne dosť chýb pri príjme, či vysielaní a tým zaznamenali značne zvýšenú chybovosť zbernice v poslednom čase. Takýto uzol je stále schopný plnej komunikácie, avšak prípadné zistené chyby už nesmie ohlasovať aktívnym chybovým rámcom (6 dominantných bitov + 8 recesívnych bitov = 6x log. 0 + 8x log. 1). Zistené chyby ohlasuje pasívnym chybovým rámcom (6 recesívnych bitov nasledovaných 8bitovým recesívnym oddeľovačom = 14x log. 1), ktorý nezasahuje do komunikácie medzi inými uzlami na zbernici. Navyše pasívny uzol musí čakať ďalších 8 bitov pred tým, ako sa znova pokúsi poslať svoj rámec, ktorý bol označený ako poškodený. Ak teda uzol sám seba označil ako chybný, nemôže naďalej rušiť ostatné vysielaním dominantných aktívnych chybových rámcov.

5.3. Odpojené uzly (*bus-off*)

Nesmie nijak ovplyvňovať zbernicu. Znamená to teda, že nemôže vysielať žiadne rámce, potvrdzovacie bity, chybové ani preťažovacie rámce. Na konkrétnej implementácii závisí navyše to, či takýto uzol vôbec naďalej prijíma dáta.



Uvedený obrázok znázorňuje stavový diagram prechodov medzi jednotlivými chybovými stavmi. Po resete je uzol v aktívnom stave. Ak čo len jeden z dvoch chybových čítačov presiahne hodnotu 127, uzol prechádza do pasívneho stavu. Ak však na druhej strane obidva čítače klesnú pod 128, stane sa uzol znova aktívnym. Odpojeným (*bus-off*) sa uzol stáva, ak čítač jeho odosielacích chýb dovŕšil hodnotu 255. Z tohto stavu môže nadobudnúť aktívny stav jedine v prípade návratu obidvoch chybových čítačov do nuly, resetu, rekonfigurácie. Po tomto všetkom musí navyše čakať, kým na sa zbernici objaví 128 krát postupnosť za sebou nasledujúcich jedenástich recesívnych bitov (koncov rámca). Toto pravidlo zabezpečí, že aj po resete musí uzol počkať na úspešné prenesienie až 128 nepoškodených rámcov pred opätovným zapojením sa do komunikácie.

6. Použité fyzické médium

Samotný protokol CAN (ISO11898-1), ktorý definuje len linkovú vrstvu nestačí. Sú v ňom obsiahnuté totiž iba základné požiadavky na samotnú fyzickú vrstvu. Navyše kvôli vzájomnej kompatibilite radičov od rôznych výrobcov vznikla norma ISO11898-2. Obsahuje detailné požiadavky na fyzickú vrstvu. Nájdeme v nej, že pre fyzickú implementáciu zbernice máme použiť buď optické vedenie (dnes ešte stále príliš drahé riešenie), alebo lacnejšie elektrické vedenie. Pri použití optického

vedenia je dominantná úroveň reprezentovaná ako "svieti" a recesívna ako "nesvieti". V našej práci sa však pre uvedené dôvody nebudeme zaoberať optickým vedením. Predmetom záujmu tejto práce je elektrické vedenie, pretože práve toto riešenie je v praxi využívané vo veľkom počte dnešných automobilov a iných zariadení.

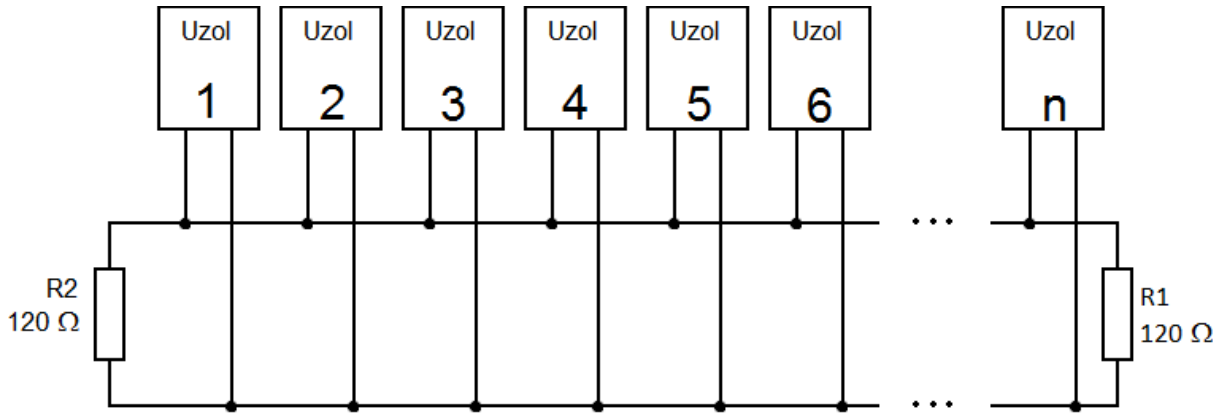
Ide o dvojvodičové krútené vedenie (jeden krútený pár), kde jeho dva vodiče nesú názvy CAN_LOW a CAN_HIGH. Hlavné dôvody využitia tohto riešenia sú hneď dva - vyššia odolnosť voči elektromagnetickému rušeniu a vyššia spoľahlivosť (jeden z vodičov môže zlyhať). Norma ďalej definuje napäťové úrovne v týchto drôtoch nasledovne (hodnoty sú vo voltoch):

Stav	CAN_LOW	CAN_HIGH	Rozdiel napätí (V_{Dif})
Dominantný	1,5	3,5	2
Recesívny	2,5	2,5	0

Prehľad maximálnych dĺžok vedenia:

Rýchosť zbernice (kbit/s)	Maximálna dĺžka vedenia (m)
1000	40
500	112
300	200
100	640
50	1340
20	2600

Norma navyše definuje, že na obidvoch koncoch (kvôli eliminácii odrazov na vedení) musí byť zakončovací rezistor s odporom 120 Ohmov. Na jednej zbernici môže byť zapojených až 30 zariadení. Nasledovný obrázok ilustruje spôsob zapojenia elektrického vedenia zbernice a spôsob pripájania jednotlivých uzlov k nej.



7. Implementácia radiča

7.1. Baud-Rate Prescaler

Tento obvod je delička hodinového vnútorného oscilátora. Pri použití nášho vývojového kitu kmitá kryštál vnútorného oscilátora na frekvencií 50 MHz. Keďže zbernica CAN v praxi dosahuje nižšie rýchlosti, (125 kbit/s – 1 Mbit/s) je často krát nutné túto frekvenciu vydeliť. Číslo, ktorým obvod **Baud-Rate Prescaler** delí hodinovú frekvenciu sa dá nastaviť v užívateľsky nastaviteľných registroch. Keďže je dátová šírka tejto hodnoty 16 bitov, musí byť rozdelená do dvoch registrov (BRPH a BRPL). BRPH sa nachádza v nastavovacích registroch na adrese 5, BRPL na adrese 6. Každé časové kvantum potom trvá o jeden takt hodín viac, než nastavená hodnota. Ak teda chceme, aby jedno časové kvantum trvalo napríklad 10 taktov vnútorného oscilátora, nastavíme do registra BRP hodnotu 9, čomu zodpovedá nasledovné nastavenie: BRPH - „0000 0000“ BRPL „0000 1001“

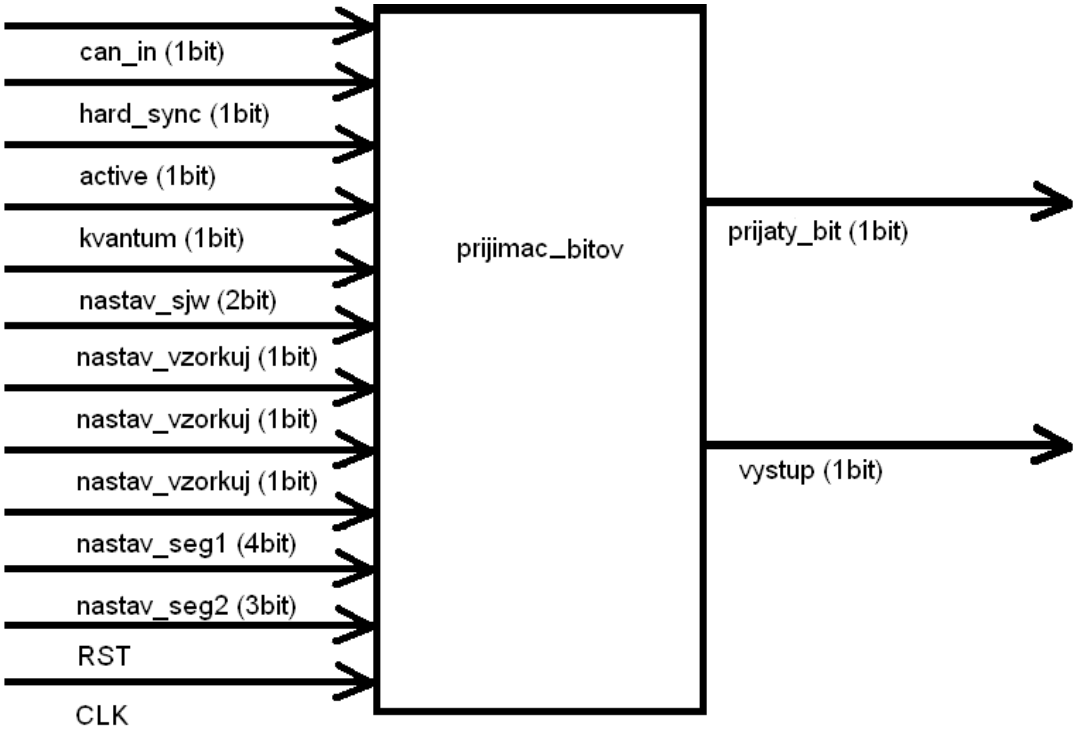
Baud-rate prescaler obsahuje jeden výstupný register kvantum_outreg a jeden vnútorný register counter. Riadi sa signálmi ***RST (reset)***, ***CLK (hodiny)***, ďalej signálom ***active*** a 16 bitovým nastavením ***nastav_brp***. Jediným jeho výstupom je signál ***kvantum***. Vnútorný register kvantum_outreg slúži len preregistrovanie výstupu ***kvantum***. Ak je vstup ***active*** aktívny (v úrovni log. 1), obvod s každým hodinovým taktom pripočíta k registru *counter* hodnotu 1. V prípade, že sa register *counter* rovná žiadanej hodnote (nastavenej užívateľom v registri BRP), obvod miesto zvýšenia registru *counter* o 1 zapíše do neho hodnotu 0 a do registra kvantum_outreg zapíše hodnotu log. 1. Pri akejkoľvek inej hodnote čítača *counter* sa do kvantum_outreg zapisuje hodnota log. 0. Vo vyššej vrstve musí byť zabezpečené pripojenie k výstupom registra BRPH a BRPL na vstup ***nastav_brp*** a ***active*** na ***active*** vstup obvodu **Baud-Rate Prescaler**. Výstup ***kvantum*** je využívaný obvodmi (modulmi) **prijmac_bitov** a **vysielac_ramcov**, ktoré si pomocou neho synchronizujú svoj príjem/vysielanie na zbernicu CAN.

7.2. Časovač

Jedinou funkciou modulu **casovac** je určovať čas celému obvodu. Čas je využívaný modulom **vysielac** (aby vedel, či už prišiel čas odoslania tej, ktorej správy) a modulom **prijimacia_fronta** (pri prijatí správy ukladá čas jej prijatia spolu so správou)

Implementácia obvodu je triviálna - keďže vnútorný oscilátor taktuje na frekvencii 50 MHz (perióda hodín je tým pádom 20 ns) a čas má byť určovaný v µs, obvod počíta vždy od 0 do 49, kedy vynuluje svoje počítadlo taktov oscilátora a čas zväčší o 1 µs. Výstup 32-bitového registra je priamo privedený na výstup ***cas***.

7.3. Príjmač bitov



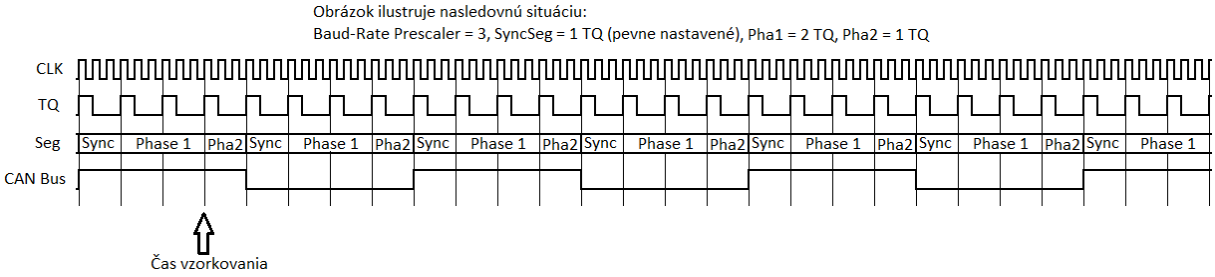
Slúži na prijímanie jednotlivých bitov zo zbernice CAN. Synchronizuje si svoje hodiny na každú dobežnú hranu dvomi spôsobmi: Ak prijíma začiatok rámca (dominantný bit SOF), svoje hodiny nasynchronizuje na prichádzajúce vysielanie. Ak však príjme prvý bit, svoje hodiny môže len resynchronizovať, aj to iba na dobežnú hranu (pri prechode zbernice z recesívneho stavu do dominantného, čiže zo stavu log. 1 do stavu log. 0), navyše maximálne o SJW (*Synchronisation Jump Width*). Po prijatí každého bitu toto oznámi výstupným signálom ***prijaty_bit***. Vtedy sa logická úroveň prijatého bitu objaví na výstupe ***vystup***.

Každý bit sa skladá z nasledovných časových úsekov:

- *Synchronization segment* - jeho dĺžka je pevne nastavená na 1 časové kvantum
- *Phase 1 segment* - nastaviteľný 4 bitovým vstupom nastav_seg1 - segment trvá vždy 1 jedno časové kvantum dlhšie, než nastavený počet, ten keďže je 4 bitový, sa dá

nastaviť v intervaloch od 0 do 15, z čoho vyplýva, že samotný *phase 1 segment* môže trvať od 1 do 16 časových kvánt.

- *Phase 2 segment* - trvá o 1 časové kvantum dlhšie, než 3 bitové číslo privedené na vstup nastav_seg2. Nastaviteľné hodnoty sú od 0 do 7, čiže časový segment 2 môže trvať od 1 do 8 časových kvánt.



Prijímac_bitov má ešte jednu nastaviteľnú voľbu, a síce ***nastav_vzorkuj3x***. V prípade, že je táto voľba vypnutá (na vstup ***nastav_vzorkuj3x*** je privedená úroveň log. 0), platí prvý z vyššie uvedených obrázkov. Znamená to, že vzorkovanie logickej úrovne prijímaného bitu prebieha iba raz a to v poslednom časovom kvante segmentu 1. Ak je voľba ***nastav_vzorkuj3x*** v úrovni log. 1, **prijimac_bitov** odoberie 3 vzorky zbernice pre každý bit a síce v posledných 3 časových kvantách prvého časového segmentu. Výslednú hodnotu prijímaného bitu potom určí podľa toho, ktorá logická úroveň prevažuje v týchto troch vzorkách. Je nutné podotknúť, že pokiaľ užívateľ zvolí možnosť trojitého vzorkovania, musí tiež zvoliť dĺžku trvania prvého časového segmentu aspoň 3 časové kvantá (***nastav_seg1*** $\geq$ 2). ***Nastav_sjw*** určuje, o koľko časových kvánt sa maximálne môžu resynchronizovať hodiny každého bitu. Znova platí, že skutočný maximálny počet časových kvánt je o 1 vyšší, teda v rozmedzí od 1 do 4 časových kvánt.

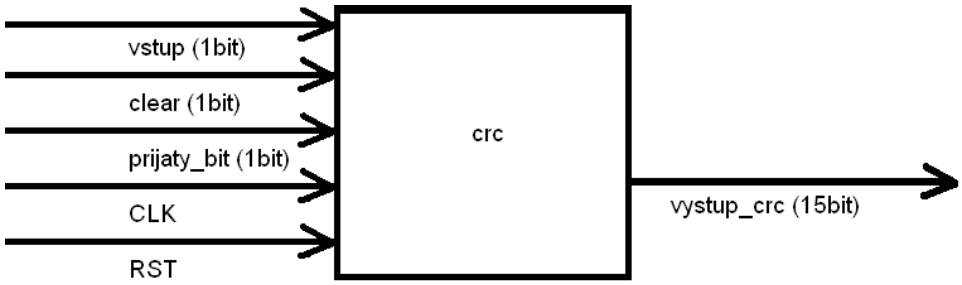
Prijímac_bitov je stavový automat s tromi stavmi:

- SYNC
- PHASE1
- PHASE2

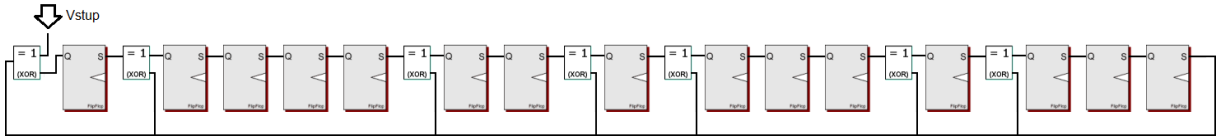
Ak je zapnutá tvrdá synchronizácia, jednoducho počíta časové kvantá, ak napočíta koniec príslušného časového segmentu, zmení svoj stav a počíta od nuly ďalší segment. Ak navyše napočíta posledné (posledné 3) časové kvantum segmentu 1, navzorkuje logickú úroveň zbernice. Ak v akomkoľvek čase príde dobežná hrana, automat okamžite prejde do stavu PHASE1 a poznačí si, že daný bit už bol synchronizovaný. Pri vypnutí tvrdej synchronizácie automat rovnako prechádza medzi týmito tromi stavmi, avšak stavy PHASE1 a PHASE2 majú navyše každý po 2 podstavy (začiatok a koniec). Keď príde dobežná hrana na začiatku stavu PHASE1, čítač časových kvánt sa vynuluje, čím sa predĺži trvanie stavu PHASE1. Ak dobežná hrana príde na konci stavu PHASE1, čítač časových kvánt sa namiesto vynulovania, zvýši o sjw, čím zabezpečí garantovaný posuv hodín prijímača bitov.

Detekovaná dobežná hrana na začiatku stavu PHASE2 spôsobí, že k čítaču časových kvánt sa nepripočíta 1, ako by sa pripočítavala za normálnych okolností, ale pripočíta sa k nemu SJW, čím sa unáhli skončenie aktuálneho bitu a začiatok bitu ďalšieho. Posledný prípad resynchronizácie môže nastať, ak prijímač bitov detekuje dobežnú hranu na konci segmentu 2 (v stave PHASE2). Ak nastane uvedená situácia, okamžite ukončí PHASE2 a prejde do stavu PHASE1 nasledovného bitu. Ku ktorejkoľvek synchronizácii prijímača bitov (aj tvrdej aj resynchronizácii) je nutné podotknúť, že každý bit je možné synchronizovať maximálne raz. V čase odoberania poslednej vzorky zbernice CAN automat navyše na výstup ***prijaty_bit*** indikuje príznak prijatia bitu. Výstup ***vystup*** obvodu je určovaný podľa toho, či je zapnuté trojnásobné vzorkovanie. V prípade, že nie, objaví sa na ňom hodnota poslednej vzorky. Pokiaľ áno, je na výstup ***vystup*** zaradený multiplexor, ktorý má nadefinovanú každú kombináciu troch vzoriek a na ich základe vyberá výstupnú hodnotu podľa toho, aká logická úroveň vo vzorkách prevažuje.

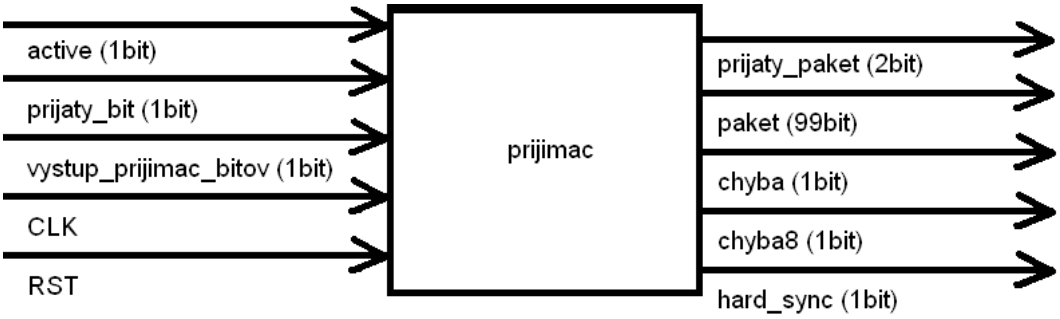
7.4. CRC (Cyclic Redundancy Check)



Modul je využívaný rovnako modulom **vysielac**, tak aj modulom **prijimac**, avšak každý z týchto modulov v sebe obsahuje vlastný **CRC** modul, z čoho plynie, že sa v našom obvode tieto obvody nachádzajú dva. Slúži na výpočet kontrolného súčtu správ pre overenie správnosti prijatia správy. Modul **vysielac_ramcov** počas vysielania správy postupne počíta kontrolný súčet vysielanej správy a tento pripojí na jej koniec. Modul **prijimac_ramcov** ho využíva na to, aby overil správnosť prijatej správy porovnaním jej kontrolného poľa s hodnotou vypočítanou týmto podmodulom. Modul počíta zo správy 15-bitovú sekvenciu spôsobom uvedeným na obrázku. Ak je aktívny vstupný signál ***RST (reset)***, alebo ***clear (zmazanie)***, do všetkých registrov sa zapíše hodnota log. 0. Pokiaľ bol nejaký bit prijatý, v špeciálnom 15-bitovom registri sa jednotlivé bity posunú o 1. Avšak tu platí, že nie všetky bity sa posúvajú nezmenené, niektoré z týchto registrov majú na vstupe hradlo XOR. Jeho prvým vstupom je vždy predošlý bit posuvného registra a druhým vstupom je výstup hradla XOR vstupu a posledného bitu registra. Výstupom tohto obvodu je 15-bitová brána ***vystup_crc***, ktorej každý bit je pripojený na príslušný pamäťový register.



7.5. Prijímač rámcov



Rovnako, ako všetky ostatné obvody, pracuje iba vtedy, ak na jeho vstup ***active*** je z príslušného nastavovacieho registra privedená úroveň log. 1. Slúži na rozpoznávanie jednotlivých rámcov zo zbernice CAN a to dátových, vzdialených (oboje štandardné aj rozšírené), chybových a prípadnú detekciu chýb v prichádzajúcej komunikácií. Chyby spozná, indikuje vyššej vrstve, avšak ich spracovanie prenecháva vyšším vrstvám. Vyššej vrstve oznamuje ich prijatie výstupnými signálmi ***prijaty_paket***, ***chyba***, ***chyba8***. Jeho nižšiu vrstvu tvoria obvody **CRC** a **prijimac_bitov**. Im oznamuje svoje príkazy pomocou signálov ***hard_sync*** (**prijimac_bitov**), ***clear*** a ***prijaty_bit*** (**CRC**). Vždy, keď prijme rámec (vzdialený, či dátový), oznámi túto skutočnosť signálom ***prijaty_paket***. Vtedy sú platné dáta na jeho výstupe ***paket*** (99 bit) vo formáte uvedenom v tabuľke 10.1. Toto oznámenie od neho prichádza v čase a*cknowledge* bitu rámca, čiže je zároveň využívaný modulom **potvrdzovac**. Ten sa na základe chybových čítačov rozhodne, či rámec potvrdí, alebo (ak je uzol v chybovom stave *bus-off*, alebo je vypnuté potvrdzovanie rámcov − *silent mode*) nie. Zároveň tento signál spôsobí v čítači chýb prijímača jeho dekrementáciu (pokiaľ nie je 0). Ak v prichádzajúcej komunikácií zistí chybu, vyvolá buď signál ***chyba***, alebo ***chyba8*** v závislosti od toho, či sa má čítač chýb prijímača inkrementovať o 1 alebo o 8. ***Chyba8*** je vyvolaná napríklad, ak náš uzol bol prvý, ktorý detekoval chybu, alebo chybový príznak trvá dlhšie, než 14 bitov. Jeho vstupmi sú okrem ***hodín (CLK)***, ***resetu (RST)*** a signálu ***active*** navyše ešte vstup zo zbernice CAN (***can_in***) a nastavovacie vstupy pre nižšiu vrstvu (**prijimac_bitov**). Sám však nemá žiadne nastavovacie vstupy z nastavovacích registrov.

Vo vnútri obvodu **prijimac** sa nachádzajú 2 moduly - **CRC** a **prijimac_bitov**. Nastavovacie vstupy pre nižšiu vrstvu sú všetky pripojené na nastavovacie vstupy modulu prijímač bitov, spolu so signálmi ***active***, ***RST (reset)***, ***CLK (clock)***. Vstupný signál ***hard_sync*** modulu **prijimac_bitov** je pripojený na vlastný register v module **prijimac**, ktorý do neho podľa svojej potreby ukladá svoje príkazy. Obdobná situácia je pri vstupe modulu **CRC** (***clear*** a ***prijaty_bit***), kde každý je pripojený na výstup vlastného registra, určeného na ovládanie tohto obvodu. Vstup **CRC** modulu je priamo pripojený na výstup modulu **prijimac_bitov**, avšak jeho vstup ***prijaty_bit*** nie je priamo pripojený na rovnomenný výstup, ale ovládaný zvlášť. Vlastné registre majú v prijímači rámcov aj všetky výstupy (***prijaty_paket***, ***chyba***, ***chyba8***, ***paket***). Ďalšími vnútornými pamäťovými registrami modulu **prijimac** sú *bit_stuffing_active* (1 bit), crc_paket (15 bitov), pocitadlo (7 bitov), stav_automatu (5 bitov),

predosle_bity (6 bitov). Keď je obvod zapnutý (nastavovací vstup *active* = log. 1), reaguje na signál ***prijaty_bit*** z modulu **prijimac_bitov**. V každom čase, kedy je tento signál aktívny sa uplatňuje nasledovný princíp stavového automatu. Ak je neaktívny *bit-stuffing*, rozhoduje sa obvod čisto podľa aktuálneho stavu automatu, prípadne ešte počítadla a hodnoty prijatého bitu. Ak je však *bit-stuffing* aktívny (v čase prijímania SOF až CRC_DEL), rozhoduje sa obvod prioritnejšie podľa toho, či neprišlo za sebou 5 rovnakých bitov, až potom podľa spomínaných ostatných kritérií. Ak prišlo za sebou 5 rovnakých bitov a je zapnutý *bit-stufing*, vložený bit sa automaticky zahodí. Ak však aj šiesty bit má rovnakú logickú úroveň, automat vyhlási chybu a prejde do stavu CHYBA. Ak je *bit-stuffing* zapnutý, modul **CRC** rovnako neprijíma *bit-stuffing*-ové bity. V prípade detekcie akejkoľvek chyby v prijímanom rámci, automat vyhlási chybu a prejde do stavu CHYBA.

Stavový automat má 17 stavov. Pri každom prechode do iného stavu vynuluje počítadlo.

Stavy automatu dosiahnuteľné iba pri vypnutom *bit-stuffing*-u:

SOF: Čaká, kým príde dominantný bit (log. 0) - keď príde, zapne *bit-stuffing*, vypne tvrdú synchronizáciu, vynuluje všetky vnútorné registre, pošle mazací príkaz (***clear***) modulu **CRC**, zmení stav automatu ID.

ACK: V tomto stave automat oznámi vyššej vrstve prípadné úspešné prijatie rámca signálom ***prijaty_paket***. Vyššia vrstva tento príznak využije na to, aby na zbernicu vyslala dominantný (log. 0) bit, ako potvrdenie úspešne prijatého rámca (iba ak uzol nie je v chybovom stave *bus-off*, čiže odpojený zo zbernice). Zároveň zníži (pokiaľ nie je 0) chybový čítač prijímača a oznámi, že dáta na 99bitovom výstupe ***paket*** sú platné. Avšak signál ***prijaty_paket*** je vyslaný iba v prípade, že kontrolný 15 bitový súčet vypočítaný modulom **CRC** z doterajších bitov aktuálneho rámca (od SOF po koniec dátového poľa, resp. DLC ak sa tento rovná 0) je rovný kontrolnému súčtu na konci správy (uloženému v registri crc_paket). **Prijimac** prejde do stavu ACK_DEL.

ACK_DEL: Činnosť automatu je obdobná, ako v stave CRC_DEL. Zisťuje, či je zbernica po prenesení dominantného (log. 0) potvrdzovacieho bitu opäť v recesívnom (log. 1) stave a mení stav automatu na EOF.

EOF: Trvá sedem bitov. V každom z nich automat kontroluje, či je zbernica v recesívnom stave (log. 1). Navyše počas prvého bitu znova skontroluje, či sa vypočítaný kontrolný súčet zhoduje s prijatým na konci aktuálne prijímaného rámca. Ak nie, vyhlási chybu signálom chyba a prejde do stavu CHYBA. Ak všetko prebehlo v poriadku, zmení stav automatu na INTERFR a zapne tvrdú synchronizáciu (výstup ***hard_sync*** nastaví do úrovne log. 1).

INTERFR: Medzirámcová medzera by mala trvať 3 bity, počas ktorých je zbernica v recesívnom stave (log. 1). Avšak napr. kvôli skráteniu tretieho bitu hrubou synchronizáciou hodín prijímača bitov je prípadný tretí dominantný bit (log. 0) považovaný za SOF ďalšieho rámca.

CHYBA: Do tohto stavu prejde automat prijímač rámcov vždy, keď zistí chybu v prijímanom rámci (chyba formátu rámca a chyba *bit-stufing*-u). Pri prechode do tohto stavu automat vždy vypne kontrolu *bit-stuffing*-u a aktivuje výstupný príznak chyby, čiže aktivuje na dobu jedného hodinového taktu výstupný signál ***chyba***. Keď tento signál zachytí modul **registre**, zvýši počítadlo chýb prijímača o 1. V tomto stave automat čaká na doznenie chybového príznaku (či už aktívneho, alebo pasívneho), čiže 6 bitov. Potom prejde do stavu CHYBA_ODDEL.

CHYBA_ODDEL: V prvom bite oddeľovača chyby automat zisťuje, či je zbernica v dominantnom stave (log. 0). Ak by totiž zbernica bola v dominantnej úrovni, v praxi by to znamenalo, že iný uzol vysiela svoju chybovú ozvenu. A práve toto by znamenalo, že ľubovoľný iný uzol nezistil chybu, ktorú náš uzol detekoval a teda to, že náš uzol bol prvý, kto chybu zistil a iné uzly iba reagujú na náš chybový príznak. A práve v tomto prípade by sme museli zvýšiť naše počítadlo chýb prijímača o 8. Navyše s každým ôsmym detekovaným dominantným bitom (log. 0) v oddeľovači chyby sa zvyšuje počítadlo chýb prijímača vždy o ďalších 8.

Zapracovaním týchto pravidiel vzniklo správanie sa automatu v stave CHYBA_ODDEL. Automat v prvom bite (počítadlo = 0) chybového oddeľovača kontroluje, či je zbernica v dominantnom stave (log. 0). Pokiaľ áno, aktivuje výstupný príznak ***chyba8*** (modul **registre** vtedy zvýši počítadlo chýb prijímača o 8). S každým ďalším detekovaným dominantným bitom (log. 0) zvýši svoje vnútorné počítadlo o 1, s jednou výnimkou, a síce keď sa počítadlo rovná 6. Vtedy počítadlo vynuluje, čím sa znova dostane do stavu, v ktorom v prípade detekcie dominantnej hodnoty (log. 0) na zbernici aktivuje výstupný príznak chyba8. S detekciou každého recesívneho bitu (log. 1) počítadlo vždy vynuluje a porovná posledných 7 prijatých bitov so sekvenciou „111 1111“ (7 jednotiek). Pokiaľ teda automat príjme ôsmy recesívny bit (log. 1), prejde do stavu INTERFR, čiže medzirámcovej medzery. Tento ôsmy recesívny bit oznamuje, že sa zbernica úspešne zotavila z chyby a je pripravená na ďalšiu normálnu komunikáciu. Po uplynutí medzirámcovej medzery sa môže uzol, ktorého vysielanie bolo prerušené chybovým rámcom, znova pokúsiť vyslať svoj rámec na zbernicu.

Stavy automatu dosiahnuteľné iba pri zapnutom *bit-stuffing*-u:

ID: Prijíma postupne 11 bitov základného identifikátora tak, že ich postupne posúva v registri paket_outreg (10 - 0 bit). S každým prijatým bitom pripočítava k počítadlu 1, okrem situácie, kedy prijme posledný 11-ty bit identifikátora (počítadlo = 10). Vtedy počítadlo znuluje a zmení stav automatu na RTR.

RTR: prijme RTR bit, zapíše ho do registra paket_outreg (11-ty bit) a zmení stav automatu na EXT.

EXT: prijme *extended* bit, uloží ho do registra paket_outreg (30-ty bit). Ak má práve prijímaný bit hodnotu dominant (log. 0), zmení stav automatu na RESERVED. Prijímač v takomto prípade pokračuje prijímaním štandardného rámca. Príjem recesívneho bitu (log. 1) spôsobí prechod do stavu EXT_ID.

EXT_ID: prijme 18 bitov rozšíreného identifikátora. Prijímané bity ukladá do registra paket_outreg (29 až 12-ty bit) ich postupným posúvaním v ňom. Počítadlo počíta od 0 do 17 (18 stavov počítadla). Ak sa toto počítadlo nerovná 17, pripočíta k nemu 1, ak áno znuluje sa a stav automatu sa zmení na EXT_RTR.

EXT_RTR: prijme RTR bit rozšíreného rámca, prepíše ním hodnotu 30-teho bitu v registri paket_outreg, ktorú pred ním automat zapísal v stave RTR, keď ešte nevedel, že tento bit nie je RTR ale SRR. Stav automatu zároveň zmení na EXT_RESERVED.

EXT_RESERVED: preskočí 2 rezervované bity v rozšírenom rámci. Ak je počítadlo 0, pripočíta k nemu 1, ak už je 1, znuluje ho a zmení stav automatu na DLC.

RESERVED: preskočí 1 rezervovaný bit v štandardnom rámci, vynuluje počítadlo a zmení stav automatu na DLC.

DLC: stav, ktorom sa opäť stretávajú štandardné rámce s rozšírenými. Postupným posúvaním bitov v paket_outreg (34 až 31-ty bit) načíta 4-bitový kód dĺžky dát rámca (počtu bytov, ktoré budú nasledovať v dátovom poli správy). Počítadlo, keďže DLC má 4 bity, počíta od 0 do 3. Keď počítadlo napočíta do 3, vynuluje sa namiesto ďalšieho zvýšenia o 1 a obvod rozhoduje, do akého stavu ďalej prejde. Pri nulovej dĺžke dát (DLC = „0000“) automat zmení svoj stav na CRC. Ak je však dĺžka nenulová, prejde do stavu DATA.

DATA: Ak je dĺžka dát medzi „0001“ až „1000“ (1 až 8) prijme automat v tomto stave uvedený počet bytov. Docieli toho tak, že hodnotu DLC bitov bitovo posunie doľava o 3 bity (vynásobí 8) a od výsledného čísla odpočíta 1. Tým dostane číslo, po ktoré má počítať počítadlo. Pokiaľ je dĺžka dát väčšia než „1000“ (8), automat prijme aj tak iba 8 bytov. Dátové pole správy je postupne posuvným spôsobom ukladané do registru paket_outreg (98 až 35-ty bit). Automat po prijatí celého dátového poľa správy prechádza do stavu CRC a vynuluje počítadlo.

CRC: Na rozdiel od všetkých doterajších stavov, tento prijaté dáta nezapisuje do registru paket_outreg, ale na tento účel má prijímač rámcov vlastný posuvný register crc_paket (15 bitový). Ďalším zásadným rozdielom oproti všetkým doterajším stavom je, že už neohlasuje prijaté bity modulu CRC, pretože správnosť tohto poľa sa už nekontroluje, práve naopak všetkých

predchádzajúcich polí. Keď počítadlo napočíta do 14 (15-ty bit), automat prejde do stavu CRC_DEL a počítadlo vynuluje.

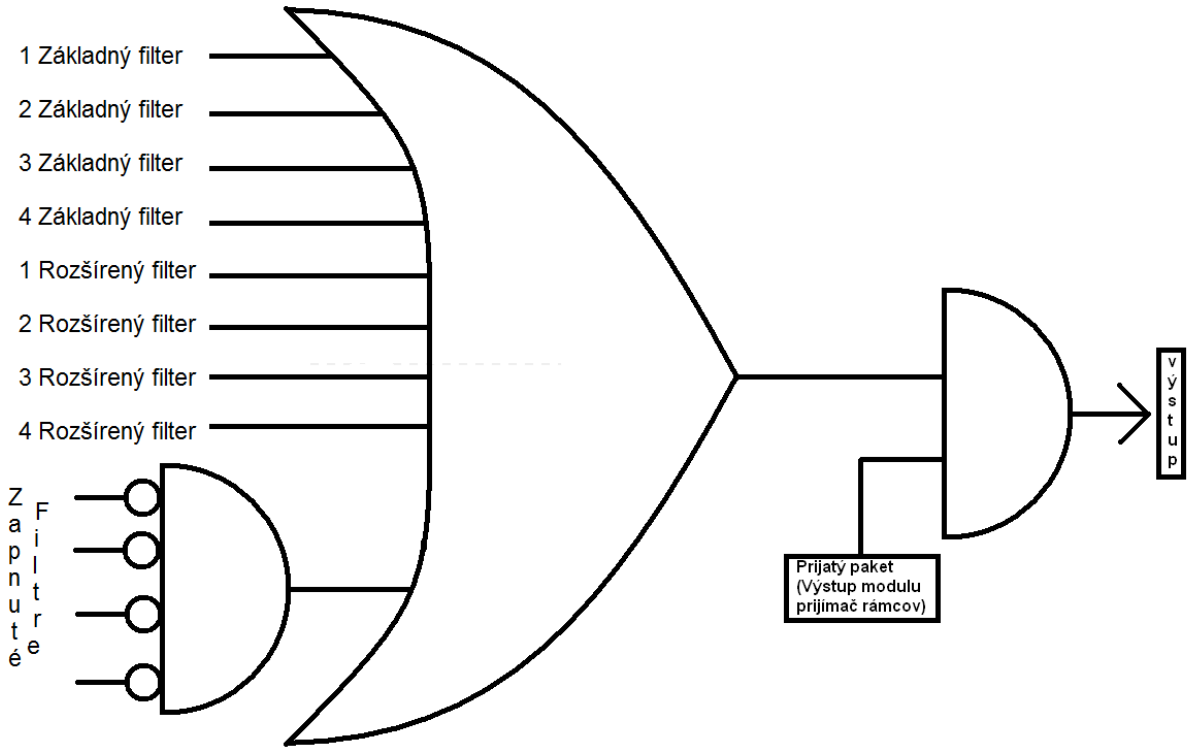
CRC_DEL: Automat v tomto stave zisťuje, či je zbernica v recesívnom stave po dobu 1 bitu. Pokiaľ áno, vynuluje počítadlo a zmení stav automatu na ACK a vypne *bit-stuffing*.

7.6. Akceptančné filtre

Náš obvod obsahuje 4 akceptančné filtre pre filtrovanie prijatých správ. Každý z nich sa dá samostatne konfigurovať a to nasledovnými parametrami nastaviteľnými v užívateľsky nastaviteľných registroch (11 - 42). Zapnutie, resp. vypnutie, určenie, či sa jedná o filter pre rozšírené, či základné rámce a 11 resp. 29 bitové kódy a masky každého filtra sú v 8 bajtovom užívateľsky nastaviteľnom registri. Ak sú všetky filtre vypnuté, automat prijme každý rámec, ktorý sa objaví na zbernici. Ak je však čo i len jeden filter zapnutý, prijme automat iba tie rámce, ktoré prejdú cez zapnuté filtre. Prijímač však potvrdí zbernici CAN každú prijatú správu bez ohľadu na to, či prejde filtrami. Filtrovanie správ prebieha na základe porovnávania jednotlivých bitov v identifikátoroch prijatých správ. V maske každého filtra sa dá pre každý bit nastaviť, či sa má porovnávať (log. 1) alebo nie

(log. 0). Do kódu filtra potom užívateľ pre tieto bity nastaví hodnotu, ktorú si praje prijímať. Ostatné bity v kóde filtra sa potom ignorujú, bez ohľadu na ich nastavenú hodnotu. Filter potom prijme iba správy, ktorých identifikátor sa zhoduje vo zvolených bitoch s kódom filtra. Koncové určenie toho, či je správa prijímačom akceptovaná napokon prebieha podľa toho, či správa prešla aspoň jedným zo zapnutých akceptančných filtrov.

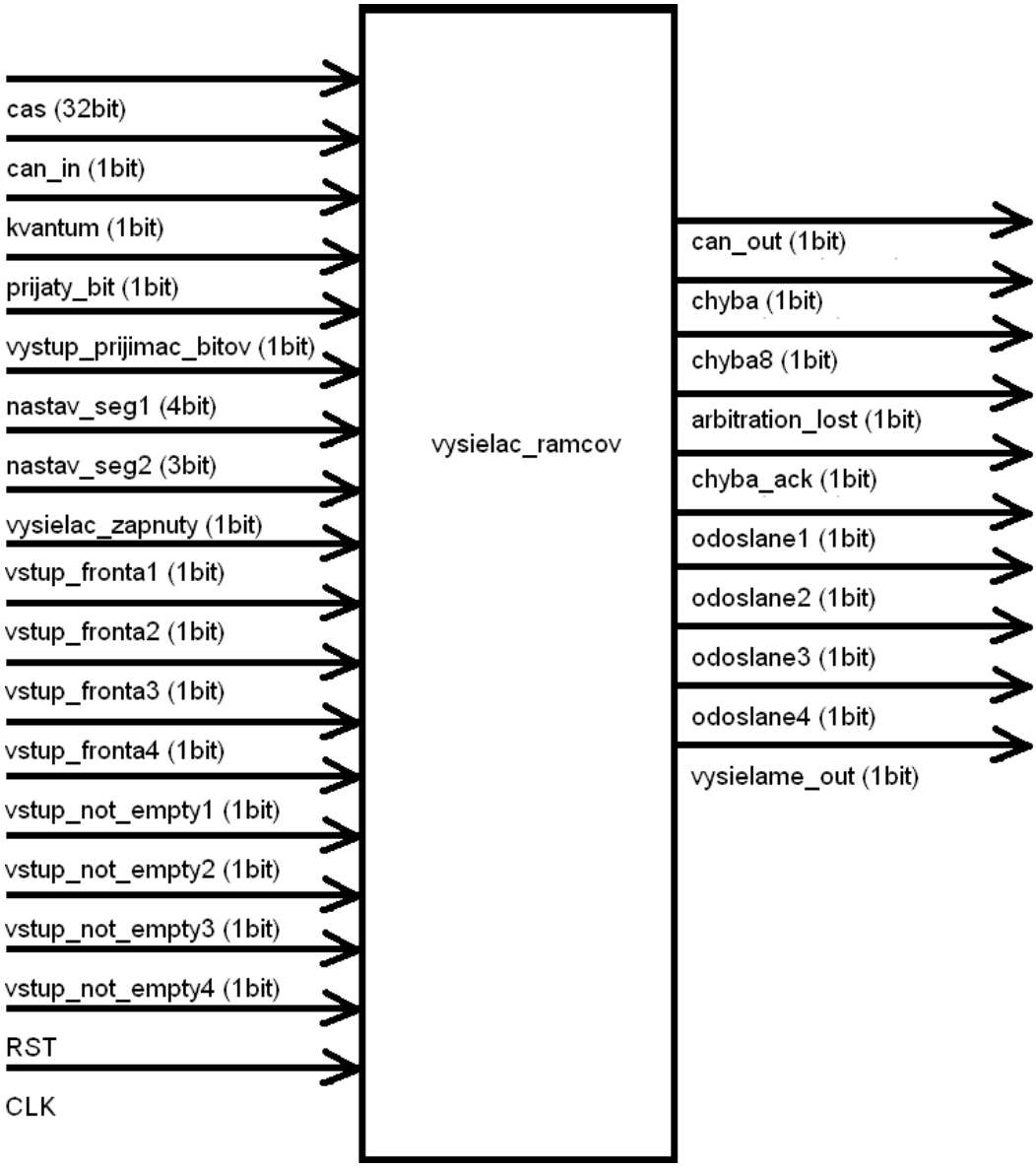
Tento modul neobsahuje žiadne pamäťové registre, jedná sa čisto o kombinačnú sieť hradiel a multiplexorov. Na výstupe tohto obvodu (***prijaty_paket_vystup***) je hradlo AND, čiže výstup je aktívny iba v prípade, že sú obidva jeho vstupy pozitívne. Prvým jeho vstupom je vstupný signál ***prijaty_paket***, čo zabezpečuje prvú podmienku aktivity výstupu – môže byť aktívny iba v prípade, že bol nejaký rámec skutočne prijatý. Na druhý vstup koncového AND hradla je pripojený výstup veľkej kombinačne siete samotných filtrov popísaných v odseku nižšie.



Jedná sa o 9-vstupové hradlo typu OR. Prvým jeho vstupom je výsledok podmienky vyhodnocujúcej, či nie sú všetky filtre vypnuté. V takom prípade je výstup vždy aktívny, čiže cez filtre prejde každý rámec. Ak je však čo i len jeden filter zapnutý, tento vstup je neaktívny. Jedná sa o 4-vstupové hradlo typu AND, ktorého každý vstup je negovaný a je na neho privedený bit zapnutia každého filtra. Ostatných 8 vstupov koncového 9-vstupového hradla OR na konci filtrov sú výstupy štyroch základných a štyroch rozšírených filtrov. Tieto obsahujú na svojom konci hradlo vyhodnocujúce podmienku, či je prijatý rámec základný, alebo rozšírený a konkrétny filter má akceptovať práve daný typ rámca. V každom filtri je pre každý bit podmienka (hradlo) XNOR porovnávajúce ho s príslušným bitom akceptančného kódu filtra. Za ním je navyše hradlo OR, ktorého

druhým vstupom je negácia zapnutia porovnávania daného bitu, čo znamená, že bit je vždy vyhodnotený ako akceptovaný, pokiaľ je filtrovanie daného bitu vypnuté. Posledným článkom každého filtra je 11 resp. 29 vstupové AND hradlo určujúce, či boli všetky bity akceptované a rámec je takého typu, pre aký je filter nastavený.

7.7. Vysielač rámcov



Ak je zbernica voľná, môže každý uzol zbernice začať vysielať. Každý uzol iniciujúci svoje vysielanie pošle na zbernicu dominantný bit SOF (log. 0) oznamujúci začiatok vysielania. Tu však môžu vznikať rôzne časové predstihy a oneskorenia spôsobené rôznym časovaním jednotlivých uzlov, rôznou dobou propagácie (šírenia informácií) medzi všetkými uzlami, a faktom, že zbernica neprenáša hodiny samostatným vodičom. Za voľnú zbernicu sa síce považuje recesívny stav zbernice (zbernica v úrovni log. 1) po dobu prenosu minimálne jedenástich bitov, avšak z uvedených dôvodov desynchronizácie zbernice je tolerovaný začiatok vysielania už po desiatich recesívnych (log. 1)

bitoch. Vtedy môže každý uzol oznámiť začiatok svojho vysielania jedným dominantným (log. 0) bitom SOF. Keďže vyslaním tohto bitu začínajú uzly súťažiť o právo vysielať, je veľmi dôležité, aby bolo vysielanie jednotlivých arbitračných bitov synchronizované u všetkých uzlov pokúšajúcich sa vysielať.

A práve tu môžu vznikať problémy, preto po uplynutí desiatich za sebou idúcich recesívnych (log. 1) bitov na zbernici začne po dobu trvania 1 bitu (jedenásteho) v každom okamihu nábežnej hrany hodín zisťovať, či zbernica nie je v dominantnom (log. 0) stave, čiže či už niekto iný nezačal vysielať v predstihu. Ak ani tento jedenásty bit medzirámcovej medzery nikto nezačal vysielať, znamená to, že môžeme začať my. Ak však začal predsa len niekto vysielať v predstihu, pre nás to znamená, že s ním už v predstihu musíme začať súťažiť o právo vysielať. V obidvoch prípadoch (niekto iný začal vysielať v predstihu, alebo nikto iný nezačal vysielať po dobu jedenástich bitov) to pre nás znamená, že musíme zistiť, či chceme aj my na zbernicu niečo poslať. Ak nechceme poslať nič, čakáme v stave pripravenosti na pokyn užívateľa o tom, že chce niečo poslať, alebo naopak, ak začne niekto (či už v predstihu, alebo nie) iný posielať nejaký rámec, musíme počkať. Kým jeho vysielanie skončí. Ak však v okamihu, kedy môžeme začať vysielať zistíme, že máme čo poslať, začneme vysielať. Toto svoje rozhodnutie oznámime najprv vyslaním jedného dominantného bitu (SOF – log. 0) a potom vysielaním arbitrážneho poľa. Počas vysielania arbitráže po vyslaní každého bitu spätne kontrolujeme zbernicu, či je na nej naozaj taká logická úroveň, akú skutočne vysiela. Ak je všetko v poriadku, pokračujeme vo svojom vysielaní. Ak však vysielame recesívny bit (log. 1) a zbernica je v dominantnom stave (log. 0), znamená to pre nás nutnosť okamžitého prerušenia vysielania, pretože nám týmto iný uzol signalizuje, že vysiela rámec s vyššou prioritou. Takýto stav navyše oznámime užívateľovi prerušovacím signálom ***arbitration_lost***. Najhorší prípad však nastane, ak vysielame dominantný bit (log. 0) a zbernica je napriek tomu v recesívnom stave (log. 1). V takomto prípade rovnako musíme prerušiť svoje vysielanie a vyslať chybový rámec, okrem toho musíme navyše zvýšiť chybové počítadlo vysielača o 8, pretože toto značí fatálnu chybu zbernice. Pri úspešnom odoslaní všetkých bitov arbitrážneho poľa môžeme prejsť k samotnému posielaniu správy, čiže riadiaceho poľa, prípadného dátového poľa, CRC a nakoniec potvrdzovacieho poľa s ukončovačom vysielania. Naďalej aj po úspešnom odvysielaní arbitrážneho poľa však platí podmienka kontrolovania každého bitu. Zmenia sa však pravidlá pri detekcii iného bitu na zbernici, než aký vysielame. V akomkoľvek prípade detekcie iného bitu musíme okamžite zvýšiť počítadlo chýb vysielača o osem, prerušiť vysielanie chybovým rámcom a po zotavení zbernice z chybového stavu sa pokúsiť vyslať rámec znova. Ale ak všetko dobre prejde a my pošleme celý rámec až po *CRC delimiter* správne, vysielame recesívny bit (log. 1) *acknowledge* ako potvrdzovací bit. Počas vysielania tohto bitu my síce vysielame recesívny bit (log. 1), ale očakávame zbernicu v dominantnom stave (log. 0). Je to z dôvodu potvrdzovania nášho rámca inými uzlami, ktoré vyslaním dominantného bitu (log. 0) prebijú nami vyslaný recesívny bit (log. 1) a tým nám oznámia úspešnosť prijatia nami

vysielaného rámca. Ak sme naozaj v tomto bite detekovali dominantný stav zbernice (log. 0), pokračujeme vo vysielaní. Ak však detekujeme recesívny stav zbernice (log. 1) znamená to pre nás, že náš rámec nik neprijal. Vtedy začneme vysielať chybový rámec a uplatníme nasledovné pravidlo zvyšovania počítadla chýb vysielača: V prípade, že sme v stave *error-passive* (jedno z chybových počítadiel je väčšie ako 127), nezvýšime ho, v opačnom prípade ho zvýšime o 8. Ak náš rámec čo i len jeden uzol potvrdil, pokračujeme vo vysielaní ďalších ôsmych recesívnych bitov (log. 1) ako ukončovača správy (EOF). Počas vysielania EOF kontrolujeme, či naše vysielanie neprerušil iný uzol dominantným bitom (log. 0). Pretože toto by znamenalo, že ktorýkoľvek uzol zaznamenal chybu (napríklad CRC) ktoréhokoľvek z prijímačov. V takomto výnimočnom prípade vyšleme aj my chybový rámec (ako ozvenu na vzdialenú chybu). V štandardnom prípade, kedy je vysielanie všetkých ôsmych recesívnych bitov (log. 1) úspešné, indikujeme vyššej vrstve signál vyslania. Na neho reaguje napríklad počítadlo chýb vysielača a fronta odoslaných správ a automat znova prejde do stavu INTERFR, avšak je nastavené, aby čakal už iba 3 bity na ďalšie právo vysielať, pretože 8 recesívnych bitov (log. 1) sa na zbernici už objavilo. Každý vyslaný bit, okrem *bit-stuffing*-ových bitov, je zároveň posielaný do modulu CRC.

Automat štartuje v stave INTERFR. V tomto stave automat čaká na 10 respektíve 2 recesívne bity (log. 1), zároveň nuluje modul **CRC** signálom ***clear***. V prípade, že príde recesívny bit (log. 1), automat zväčší počítadlo bitov o 1 (ak už nie je 10), alebo ho vynuluje a prejde do stavu CAKANIE. S prijatím dominantného bitu (log. 0) vynuluje počítadlo bitov. Tento postup zabezpečuje, že automat detekuje 10 za sebou idúcich recesívnych bitov (log. 1), čo znamená, že zbernica je voľná a teda my máme právo vysielať.

CAKANIE – v tomto stave, ako je uvedené vyššie, sa už môže stať, že niekto iný začne vysielať svoj rámec v predstihu a náš uzol zbernice v prípade, že chce tiež niečo poslať, musí okamžite začať súťažiť s týmto uzlom (uzlami) o právo vysielať. Ak však nenastane takýto prípad (nik nezačne vysielať v predstihu – v jedenástom bite, ktorý mal byť ešte recesívny – log. 1), znamená to pre náš uzol, že môže začať vysielať (ak má čo). Tento postup je zabezpečený tak, že automat v stave CAKANIE pri každej nábežnej hrane hodín (signálu ***CLK***) zisťuje stav zbernice. Pri dominantnom stave (log. 0) začne okamžite vysielať aj on (ak má čo) svoj dominantný bit SOF (log. 0) a prejde do stavu ARBITRAZ. Ak nemá čo poslať, prejde do stavu INTERFR, čím začne znova čakať na nasledujúce právo vysielať. V opačnom prípade (zbernica je recesívna – log. 1) automat počíta časové kvantá dovtedy, kým z nich nenapočíta dĺžku celého bitu (jedenásteho recesívneho – log. 1). Ak je zbernica po celú túto dobu recesívna (log. 1), znova máme právo vysielať, ak máme čo. Ak však nemáme čo poslať ani v tento okamih, zostávame v stave pripravenosti (počítadlo časových kvánt si zachováva svoju hodnotu). Neustále zisťujeme, či nenastala jedna z dvoch uvedených situácií: Ak zbernica prešla do dominantného stavu (log. 0) a my stále nemáme čo poslať, prejdeme do stavu INTERFR, čím začneme znova čakať na ďalšie právo vysielať. Ak však zrazu máme čo poslať a stále

nikto nezačal vysielať, začneme s vysielaním (prejdeme do stavu ARBITRAZ). Fakt, či máme čo poslať zisťujeme tak, že sa pozeráme do jednotlivých štyroch odosielacích front prioritným spôsobom. S každou nábežnou hranou hodín zisťujeme, či je prvá fronta zapnutá, niečo v nej je a aktuálny čas je väčší, alebo rovný času, kedy má správa byť odoslaná. Ak neplatia súčasne tieto tri podmienky, nazrieme rovnakým spôsobom do druhej fronty. Ak ani z nej nemáme čo poslať, nazrieme do tretej, prípadne ešte do štvrtej fronty. Ak narazíme v ktorejkoľvek fronte na rámec, ktorý je vhodné v túto chvíľu vyslať, zostavíme do posuvného registra vysielany_paket celý tento rámec a aktivujeme výstup. Zároveň si do dvojbitového registra aktivna_fronta poznačíme, z ktorej fronty sme rámec vybrali, aby sme po jeho prípadnom úspešnom odoslaní vedeli, z ktorej fronty máme jeden rámec zahodiť. Vtedy sa na výstupe ***can_out*** objaví vždy posledný bit posuvného registra vysielany_paket, inak je výstup vždy v recesívnej úrovni (log. 1), aby sme nijak neovplyvňovali zbernicu.

ARBITRAZ – s každým časovým kvantom zvyšujeme počítadlo kvánt o 1, pri napočítaní celého bitu počítadlo kvánt vynulujeme a posunieme posuvný register vysielany_paket o 1 bit, čím vyšleme na zbernicu ďalší bit. Je to z dôvodu uvedeného vyššie a síce, že pokiaľ je výstup ***can_out*** aktívny, je naň privedený posledný bit posuvného registra vysielany_paket. Zároveň s každým bitom prijatým modulom **prijimac_bitov** zisťujeme, či je tento prijatý bit zhodný s bitom, ktorý posiela náš uzol na zbernicu. Ak vysielame recesívny bit (log. 1) a na zbernici sa objavil bit dominantný (log. 0), okamžite vypíname svoj výstup a prechádzame do stavu INTERFR, čiže čakáme na ďalšiu príležitosť k vyslaniu rámca. Znamená to, že počas vysielania identifikátora našej správy sa na zbernici objavil identifikátor s vyššou prioritou a teda sme prehrali v súťaži o právo vysielať. Túto skutočnosť navyše oznámime vyššej vrstve signálom ***arbitration_lost***. Ak vysielame dominantný bit (log. 0) a zbernica je aj napriek tomu v recesívnom stave (log. 1), znamená to fatálnu chybu zbernice, pri ktorej rovnako prerušíme vysielanie rámca, avšak vyšleme chybový rámec a zvýšime počítadlo chýb vysielača o 8. V prípade, že sa počítadlo rovná počtu arbitrážnych bitov (podľa toho, či je rámec rozšírený, alebo základný) a teda v žiadnom bite nedošlo k chybe, znamená to teda, že sme vyhrali súťaž o právo vysielať, môžeme teda začať s vysielaním riadiaceho poľa rámca, čiže môžeme prejsť do stavu CONTROL. V tomto stave navyše kontrolujeme či sme nevyslali za sebou 5 bitov s rovnakou polaritou. Ak by nastal takýto prípad, musíme vložiť do komunikácie jeden bit opačnej polarity a až po odvysielaní tohto bitu pokračuje v ďalšom vysielaní.

CONTROL – automat v tomto stave odošle 6 bitov riadiaceho poľa rámca. Navyše rovnako ako v arbitrážnom poli kontroluje pravidlo vkladania bitov (*bit-stuffing*), čím sa počet reálne odoslaných bitov môže zvýšiť. Dĺžku každého bitu určuje počítaním časových kvánt. Každý bit je dlhý *time_segment* 1 + *time_segment* 2 + 2 časových kvánt. Pri napočítaní celého bitu posunieme posuvný register vysielany_paket o jeden bit, čím zabezpečíme vyslanie ďalšieho bitu. Zároveň vždy kontrolujeme, či sa každý bit úspešne vyslal na zbernicu. V prípade chyby bitu prerušíme vysielanie rámca chybovým rámcom a zvýšime počítadlo chýb vysielača o 8. V prípade úspešného vyslania

všetkých bitov riadiaceho poľa rámca sa rozhodujeme, či zmeníme stav na DATA, alebo rovno na CRC, podľa toho, či rámec nejaké dáta obsahuje (DLC sa nerovná 0). V prípade, že meníme stav na CRC, skopírujeme CRC vypočítané zo všetkých doteraz vyslaných bitov (okrem *bit-stuffing*-ových) do posuvného registra vysielany_paket a jeho zvyšok doplníme jednotkami, ktoré budeme vysielať ako ukončovanie rámca.

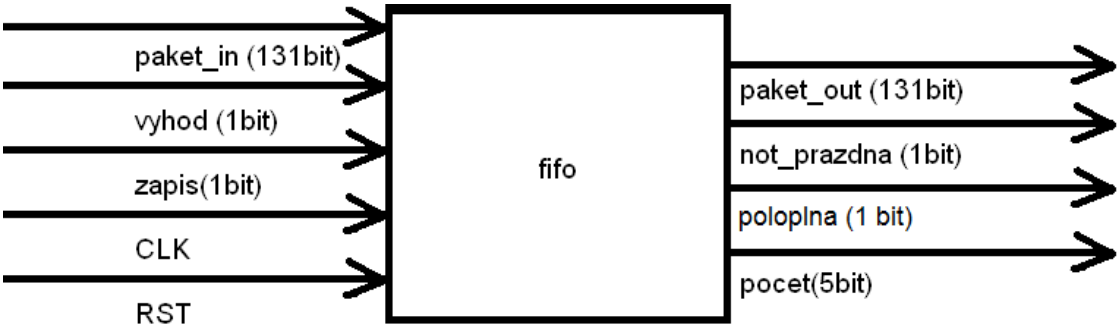
DATA – tento stav sa správa presne rovnako, ako predchádzajúci stav CONTROL - počíta časové kvantá a keď ich napočíta *time_segment* 1 + *time_segment* 2 + 2, posunie posuvný register vysielany_paket o jeden bit. Zároveň s každým odoslaným bitom kontroluje, či sa vyslal úspešne. V prípade, že nie, vyšle chybový rámec a zvýši počítadlo chýb vysielača o 8. Počet bitov, ktoré sa v tomto poli majú vyslať určí tak, že DLC posunie (*left shift*) o 3 bity doľava, čím ho vlastne vynásobí ôsmimi. Počítadlo v skutočnosti počíta od 0 do tohto čísla zmenšeného o 1. V prípade, že úspešne vyšleme všetky dátové bity, prejdeme do stavu CRC, skopírujeme do posuvného registra vysielany_paket výstup modulu **CRC** a zvyšok doplníme jednotkami.

CRC – automat v tomto stave vyšle na zbernicu 15 bitov kontrolného súčtu (CRC) a 1 recesívny bit (*CRC Delimiter* – log. 1). Spôsob posielania a kontroly bitov je zhodný s predošlými stavmi (COTROL a DATA). Rovnako je zhodná aj reakcia na prípadnú chybu bitu (vyslanie chybového rámca a zvýšenie počítadla chýb vysielača o 8). Po úspešnom odoslaní všetkých bitov kontrolného súčtu prechádza do stavu ACK.

ACK – automat v tomto stave vysiela recesívny bit (log. 1). Na rozdiel od predchádzajúcich stavov však nekontroluje či je zbernica zhodná s vysielaným bitom, ale naopak očakáva dominantnú úroveň (log. 0). V prípade, že je zbernica v recesívnom stave (log. 1), znamená to, že žiadny uzol nepotvrdil naše vysielanie, čo oznámime signálom chyby potvrdenia (***chyba_ack***) a vyslaním chybového rámca, avšak až po uplynutí dvoch bitov. Navyše ak je náš uzol v aktívnom chybovom stave (*error-active*), zvýši svoje počítadlo chýb vysielača o 8. Ak je však v pasívnom chybovom stave (*error-passive*), nezvyšuje počítadlo chýb vysielača, aby nedošlo k odpojeniu nášho uzla zo zbernice v dôsledku chýbajúcich potvrdení rámcov vysielaných naším uzlom. V prípade úspešného prijatia potvrdzovacieho dominantného bitu (log. 0) prechádza do stavu EOF.

EOF – vysiela 8 recesívnych bitov (log. 1) konca rámca. S každým bitom samozrejme kontroluje, či je zbernica v recesívnom stave (log. 1). V opačnom prípade by to mohlo znamenať napríklad chybu CRC signalizovanú iným uzlom. V prípade úspešného vyslania všetkých ôsmych bitov nevynuluje počítadlo a prejde do stavu INTERFR, čím docieli, že automat v stave INTERFR bude čakať iba na dva recesívne bity namiesto desiatich na právo vysielať, pretože 8 sa ich už na zbernici objavilo. Navyše po úspešnom odoslaní celého rámca automat túto skutočnosť oznámi jedným z výstupných signálov ***odoslane1*** - ***odoslane4*** podľa toho, z ktorej fronty pochádzal vysielaný rámec, čiže podľa stavu registra aktivna_fronta.

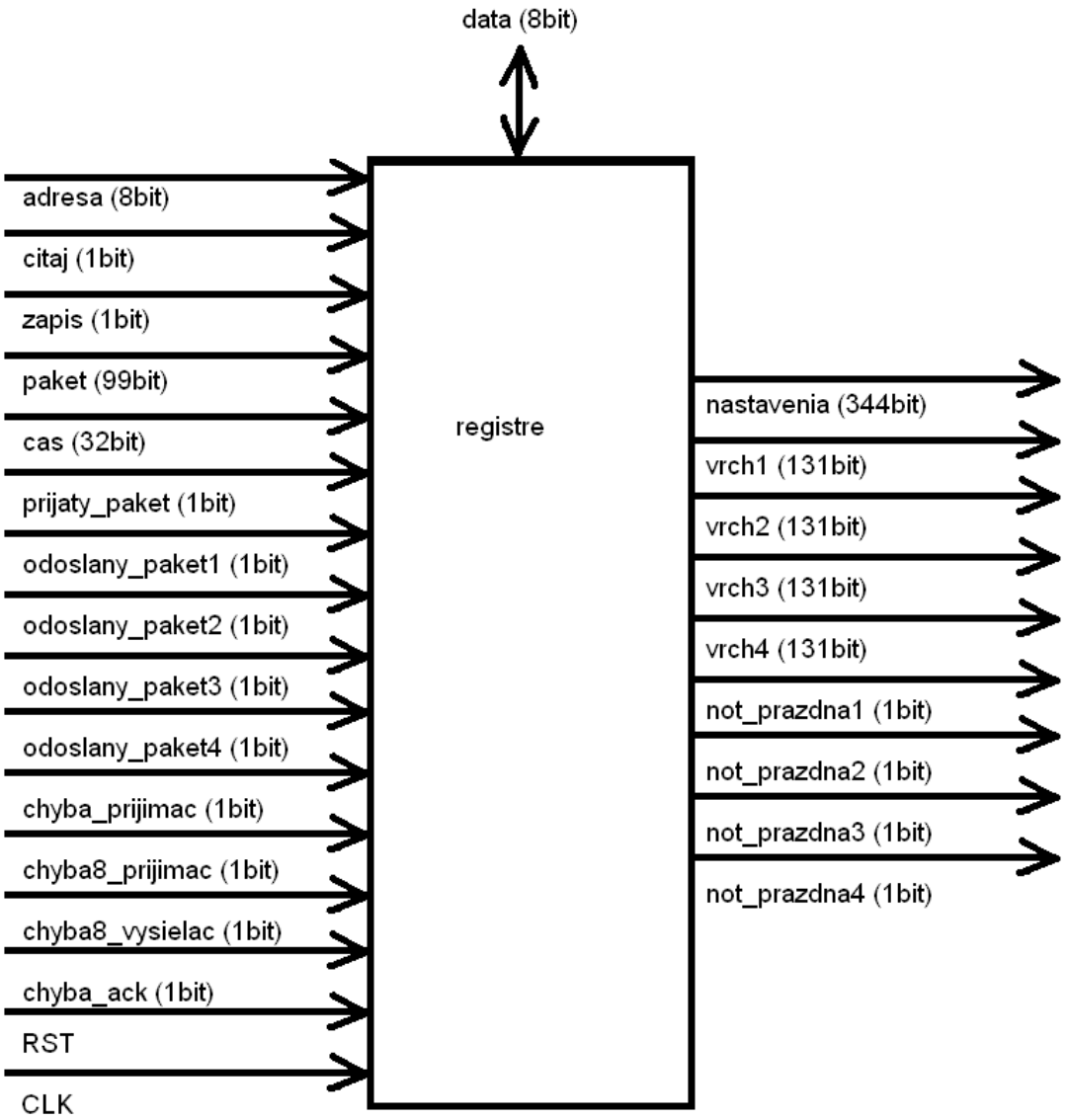
7.8. FIFO



Označenie tohto modulu vyplýva z anglického popisu jeho funkcie: *First-in, First-out* - prvý dnu, prvý von. Jedná sa o fronty správ, pri ktorých je vždy spolu so správou uložená jej časová značka. Na výstup ***paket_out*** je vždy privedená prvá správa uložená vo fronte (vrch fronty), pokiaľ fronta nie je prázdna. Ďalšími výstupmi sú negovaný výstup ***not_prazdna***, oznamujúci, či je fronta prázdna, aktívny v logickej 0 a výstup ***pocet*** (5 bitový) oznamujúci celkový počet správ v nej uložených. Okrem toho ešte obsahuje výstupný signál ***poloplna***, ktorý (ak je to povolené) užívateľovi posiela prerušenie o tom, že je polovica fronty správ plná. Je to zavedené z dôvodu, aby užívateľ vedel s dostatočným predstihom reagovať na túto skutočnosť a prípade ju stihol vyprázdniť ešte pred tým, než sa úplne naplní. Každá z týchto front sa ovláda signálmi ***vyhod*** a ***zapis***. Signál ***vyhod*** spôsobí vyhodenie vrchnej správy z fronty, signál ***zapis*** naopak na spodok fronty (pokiaľ nie je plná), pridá 131 bitovú správu (99 bitov rámec a 32 bitov časová značka) privedené na vstup ***paket_in***.

Fronta je implementovaná s použitím blokových RAM pamätí čipe hradlového poľa. Obsahuje 3 vnútorné registre - hlava, hlava1 a chvost. V registroch hlava a chvost sú uložené aktuálne adresy hlavy a chvosta fronty. V registri hlava1 je uložená vždy hodnota o 1 vyššia, než v registri hlava, čo sa potom využíva pri porovnávaní plnosti fronty. Koncový multiplexor na výstup ***paket_out*** vyberá vždy tú správu, ktorá je uložená v pamäti na adrese, danej registrom chvost. Každé pridanie správy (vstupný signál ***zapis***) spôsobí zápis vstupnej správy (***paket_in***) do pamäti na adresu danú hodnotou registru hlava (pokiaľ fronta nie je plná) a následné zvýšenie hodnoty tohto registru o 1. Naopak každé vyhodenie správy z fronty spôsobí zvýšenie hodnoty uloženej v registri chvost o 1, čím sa zmení hodnota výstupu ***paket_out*** (pokiaľ fronta nie je prázdna).

7.9. Registre



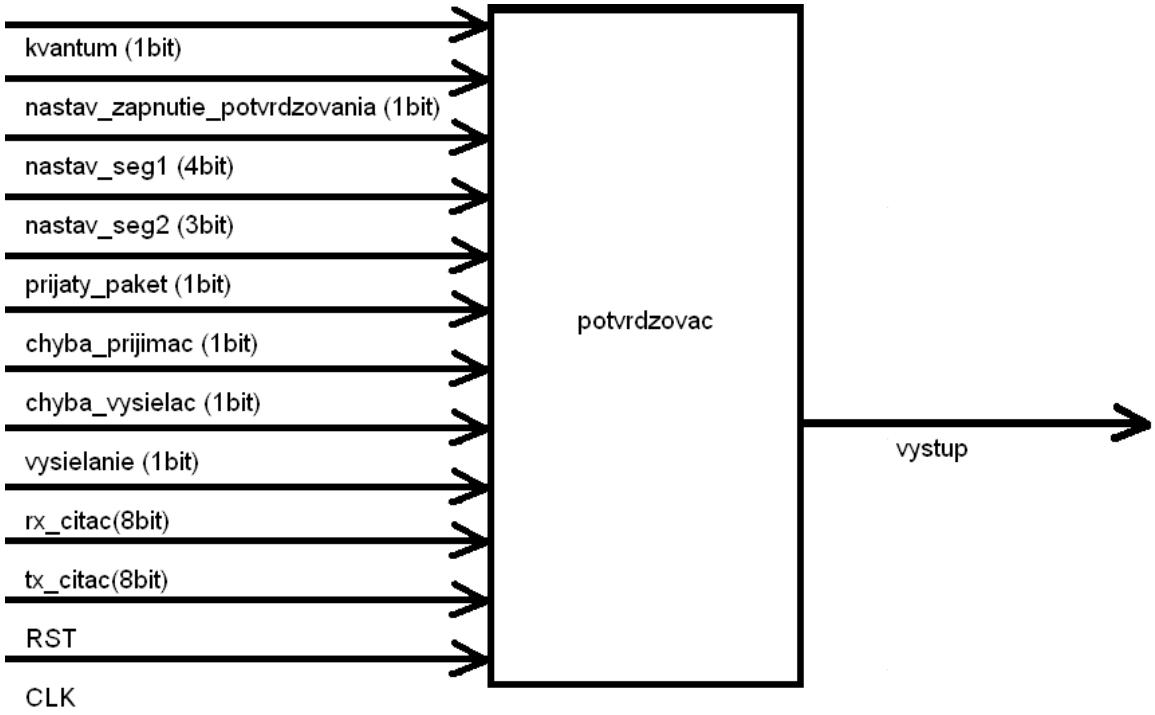
Tento modul zabezpečuje kompletnú interakciu s užívateľom, čo obnáša nie len všetky nastavovacie registre vrátane chybových počítadiel, ale aj užívateľskú komunikáciu s frontami správ (4 odosielacie a 1 prijímacia). Keďže práve tento modul obsahuje chybové počítadlá (vysielacie aj prijímacie), informuje nižšie vrstvy o chybovom stave, v ktorom sa celý obvod aktuálne nachádza. Zabezpečuje ich zvyšovanie a znižovanie pri konkrétnych udalostiach (úspešný, či chybný príjem správy, alebo jej odoslanie). Pri komunikácií užívateľa s bežnými nastaviteľnými registrami má však užívateľ pri zápise nižšiu prioritu, než radič zbernice, preto pri užívateľskom prepisovaní chybových čítačov je vždy po zápise nutné toto nastavenie po sebe skontrolovať opätovným vyčítaním, pretože môže dôjsť k udalosti, kedy užívateľ zapíše do chybových počítadiel hodnotu a v tomto okamihu prebieha ich prepis radičom. **V tomto prípade je užívateľov príkaz zápisu ignorovaný!** Ako bolo vyššie spomenuté, táto výnimka sa týka iba chybových čítačov a registra prerušení (registre 1, 9 a 10), ostatné nastaviteľné registre (0, 2-8, 11-42) radič nikdy neprepisuje, takže sa nemôže stať, že by bol užívateľský príkaz zápisu ignorovaný. Avšak sa neodporúča meniť (prepisovať) akékoľvek

užívateľsky nastaviteľné registre (0 - 42) v stave *active* (6-ty bit registra číslo 2 - CMDR sa rovná 1), pretože by to mohlo viesť k fatálnym následkom. Registre s číslami vyššími, než 42 majú špeciálne funkcie: 43 - 60 sa dajú iba čítať a je v nich obsiahnutá správa na vrchole čítacej fronty spolu s jej časovou značkou a počtom správ v tejto prijímacej fronte. Ďalšie 4 skupiny po 18 registroch sprístupňujú užívateľovi jednotlivé odosielacie fronty a sú rozdelené vždy takto: posledný osemnásty register sa dá iba čítať a jeho dolných 5 bitov obsahuje počet správ vo fronte. Naopak do prvých 17 registrov môže užívateľ iba zapisovať, pričom zápis posledného registra vždy spôsobí samotný zápis správy z týchto šestnástich registrov do konkrétnej odosielacej fronty. Rozhranie tohto modulu je zároveň užívateľským rozhraním. Má 3 signály, 8-bitovú adresu a 8-bitovú obojsmernú dátovú bránu. Táto dátová brána, keďže je obojsmerná, je vo vysokoimpedančnom stave vždy, pokiaľ užívateľ neaktivuje signál ***citaj***. Vtedy (a len ak je adresa relevantná) sa na tejto bráne objavia dáta uložené v príslušnom registri. Pokiaľ navyše dôjde k prečítaniu interrupt status registra (byte č. 1), tento sa pri zostupnej hrane (vypnutí signálu ***citaj***) vynuluje. Rovnako s prečítaním posledného bajtu správy z prijímacej fronty sa táto správa z fronty vyhodí pri dobežnej hrane signálu ***citaj*** (byte č. 59).

Hlavná pamäť modulu **registre** je realizovaná, rovnako ako fifo fronty rámcov, pomocou blokových RAM pamätí. Obsahuje 42 8-bitových pamätí. Ich všetky výstupy sú priamo privedené na výstup modulu ***nastavenia***, ktorými sa riadia všetky ďalšie moduly obvodu. Na výstupe obvodu (brána ***data***) je multiplexor, ktorý na výstup privádza nasledovné hodnoty: vysokoimpedačný stav, pokiaľ výstupný signál ***citaj*** nie je aktívny (log. 0), prípadne sa užívateľ snaží čítať byte, ktorý nemôže čítať, alebo neexistuje. V prípade, že adresa privedená užívateľom na adresný vstup ***adresa*** obvodu je menšia ako 43 a vstupný signál ***citaj*** je aktívny, je na dátovú bránu privedený výstup príslušného bytu poľa nastavovacích registrov. Je nutné podotknúť, že pokiaľ užívateľ vyčítava register 1 (*interrupt status register*) ihneď po deaktivovaní signálu ***citaj*** užívateľom sa tento byte v pamäti nuluje (príznaky prerušenia sa rušia vyčítaním). Pri aktivite signálu ***citaj*** s adresou nastavenou užívateľom na číslo 43 a vyššou je obvodu zrejmé, že užívateľ chce čítať niektorú informáciu z prijímacích, či zapisovacích front. A keďže čítať v tejto adresnej oblasti je povolené iba frontu prijatých správ a u konkrétnych zapisovacích front iba počty správ v nich, na vstupno-výstupnú dátovú bránu sú konkrétne dáta privedené iba v prípade, že adresa označuje platnú položku. V opačnom prípade je vstupno-výstupná dátová brána vo vysokoimpedančnom stave. Tento vysokoimpedančný stav je použitý preto, že ak chceme do nášho obvodu zapisovať, používame k tomu rovnakú vstupno-výstupnú dátovú bránu, avšak s dvomi rozdielmi: aktivujeme namiesto vstupného signálu ***citaj*** signál ***zapis*** a namiesto nášho očakávania dát ich naopak obvod očakáva od nás. Je nutné poznamenať 3 poznámky: Dáta sa zapisujú s nábežnou hranou signálu ***zapis***, čiže je nutné, aby dáta spolu s adresou boli na príslušné piny obvodu privedené najmenej o 20 nanosekúnd skôr (pri použití vnútorného oscilátora 50 MHz), než dôjde k aktivácií signálu ***zapis***. Navyše rovnako, ako pri čítaní, aj pri zápise adries nad 42 sú niektoré obmedzenia: nie všetky adresy nad 42 sú zapisovateľné, sú to iba vysielacie fronty (bez bajtu

oznamujúceho počet správ v konkrétnej fronte). Zapísanie na iné (neplatné) miesto nemá nijaký efekt. Spomeňme ešte podstatný fakt: pokiaľ chce užívateľ prečítať správu z vrchu fronty prijatých správ, musí ju čítať sekvenčne po bytoch, pretože prečítanie posledného bytu správy spôsobí jej zmazanie z fronty. Rovnako pri zápise do ktorejkoľvek zo štyroch zapisovacích front je nutné jednotlivé byty správy a časovej značky zapisovať postupne (sekvenčne), pretože zápis posledného bytu spôsobí zapísanie celej prenášanej správy do príslušnej zapisovacej fronty.

7.10. Potvrdzovač



Úlohou tohto obvodu je zabezpečovať vysielanie všetkých potvrdení (*acknowledge*) prijatých rámcov (ak nie je obvod v stave *silent mode*) a všetkých chybových rámcov. Obvod teda v každom čase potrebuje poznať 2 parametre - či nie je náš obvod v *silent mode*, kedy je zakázané potvrdzovanie prijatých rámcov a v ktorom chybovom stave sa náš obvod nachádza. Navyše samozrejme potrebuje mať na svoje vstupy privedené signály ***prijaty_paket***, ***chyba_prijimac*** a ***chyba_vysielac***. S príkazom ***prijaty_paket*** skontroluje, či náš uzol nie je v *silent mode, bus-off* chybovom stave a iba ak žiadna z týchto podmienok neplatí, vyšle na zbernicu jeden dominantný bit (log. 0), ako potvrdenie rámca, ktorý modul **prijimac** nášho obvodu práve prijal. S ktorýmkoľvek chybovým príkazom obdobne skontroluje, či nie je v *silent mode* a ani v *bus-off* chybovom stave. Ak má na to teda právo, vyšle 6 dominantných bitov, ako aktívny chybový rámec.

Obvod je implementovaný ako stavový automat s 5 stavmi:

CAKANIE: Len v prípade, že nie je aktívny *silent mode* a ani obvod nie je v chybovom stave *bus-off*, automat v tomto stave reaguje na vstupné signály ***prijaty_paket***, ***chyba_prijimac***

a ***chyba_vysielac***. S aktiváciou vstupného signálu ***prijaty_paket*** prechádza do stavu DOBEH, s aktiváciou ktoréhokoľvek z dvoch chybových vstupov prechádza do stavu DOBEH_CHYBY.

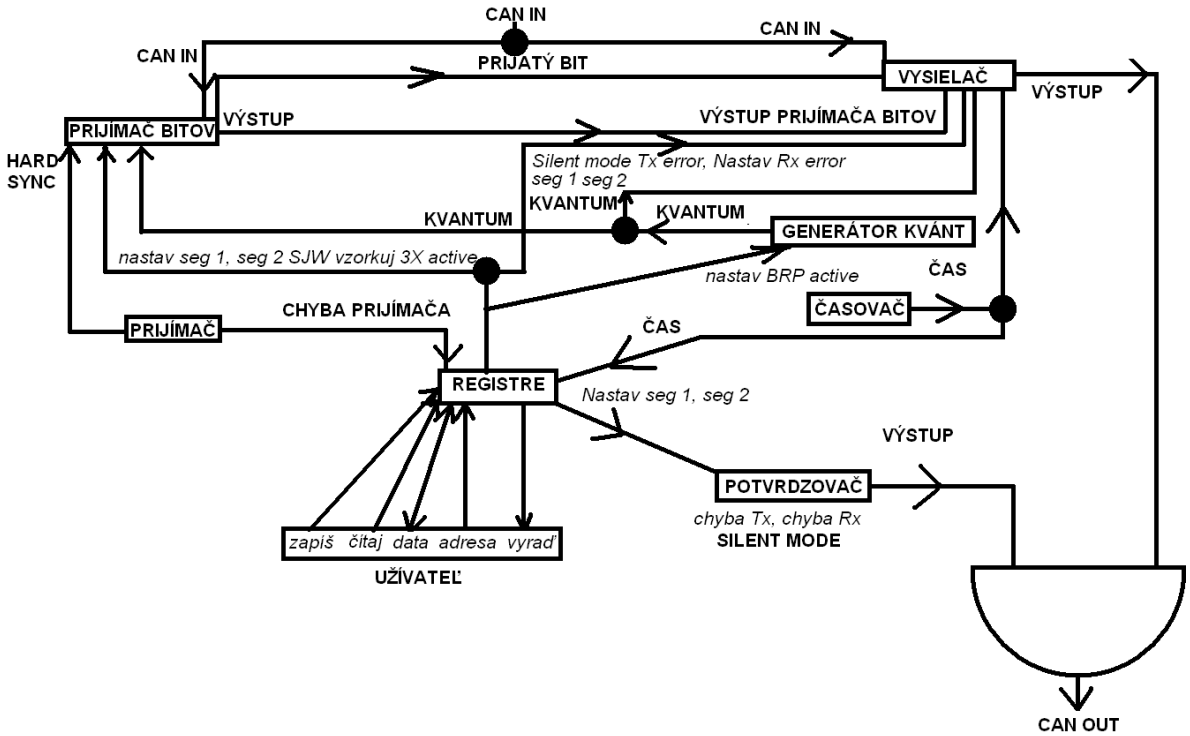
V stave DOBEH automat čaká, kým na zbernici odoznie bit *crc_delimiter*, aby mohol začať svoje vysielanie potvrdzovacieho bitu. Keď automat napočíta príslušný počet časových kvánt (seg2), prechádza do stavu VYSIELANIE.

DOBEH_CHYBA: Je to rovnaký stav, ibaže automat pri napočítaní správneho počtu časových kvánt (seg2) prechádza do stavu VYSIELANIE_CHYBA.

VYSIELANIE: Automat v tomto stave vysiela na zbernicu dominantnú hodnotu (log. 0) a počíta časové kvantá. S napočítaním hodnoty seg1 + seg2 + 2 mení výstup naspäť do recesívnej úrovne (log. 1) a vracia sa do stavu CAKANIE.

VYSIELANIE_CHYBA: Rovnako, ako v stave VYSIELANIE automat na zbernicu vysiela dominantnú úroveň (log. 0), avšak po napočítaní seg1 + seg2 + 2 časových kvánt pripočíta k počítadlu bitov číslo 1 a vynuluje počítadlo časových kvánt. Až keď je toto počítadlo bitov rovné 5 (6 vyslaných bitov), prepína výstup naspäť do recesívnej úrovne (log. 1) a vracia sa do stavu CAKANIE.

7.11. Celkové zapojenie



Modul zastrešujúci všetky doteraz spomínané moduly nesie názov **zastresenie**. Všetky moduly, <u>okrem akceptančných filtrov</u> (tie sú implementované ako kombinačný obvod – nepotrebujú hodiny) majú rozvedené spoločné signály ***CLK (hodiny)*** **a** ***RST (reset*** **-** pripojený na tlačidlo 0 na

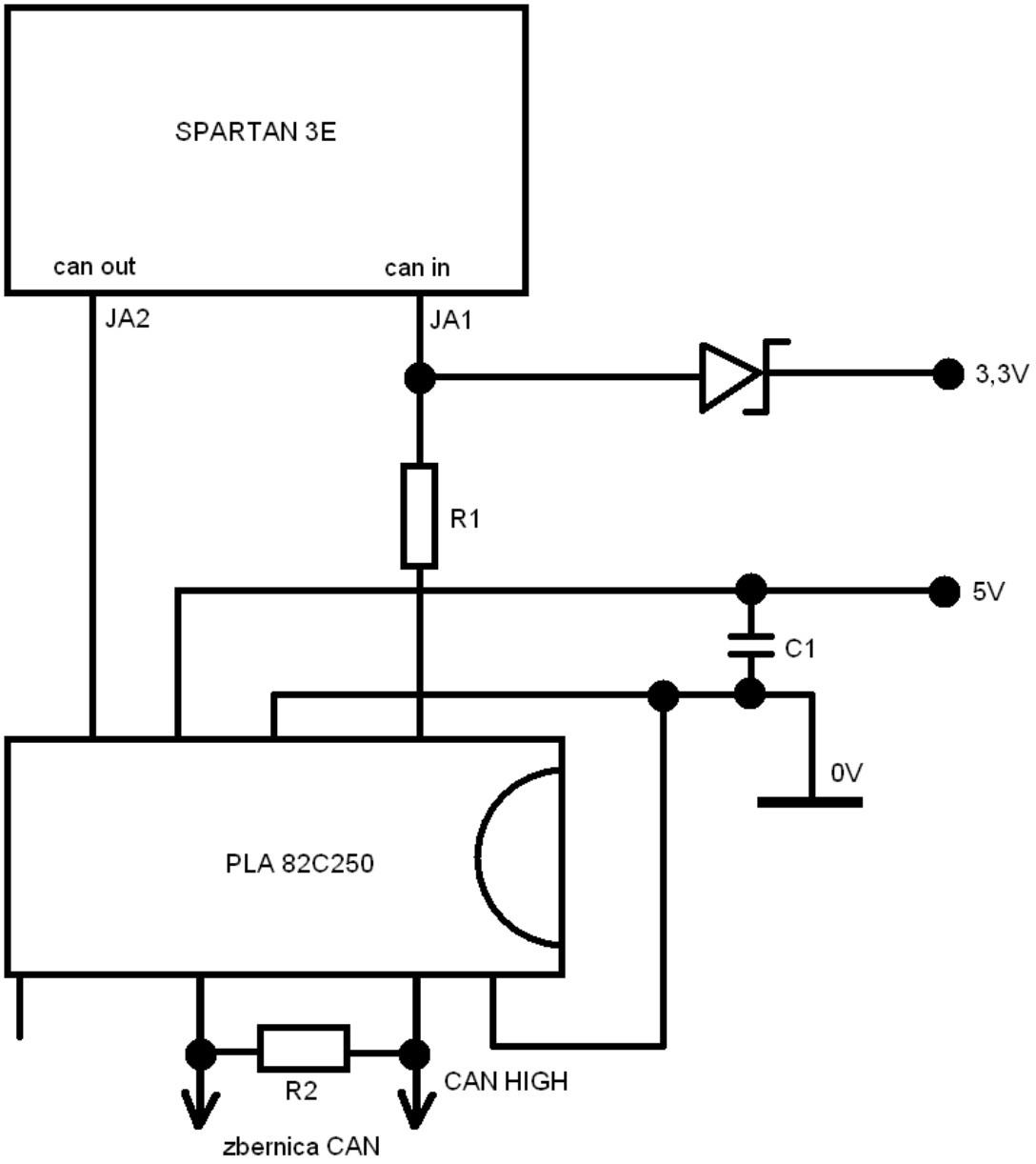
vývojovom kite). Tieto 2 signály nie sú zakreslené v obrázku celkového zapojenia pre jeho lepšiu prehľadnosť.

Prijímač je realizovaný modulmi ***prijimac_bitov*** a ***prijimac***. Ako iste vyplýva z názvu modulu **prijimac_bitov**, tento modul zabezpečuje rozpoznávanie a prijímanie jednotlivých bitov zo zbernice. K tomu potrebuje však vedieť od modulu **prijimac**, či bity, ktoré prijíma sú bity rámca (iba mäkká synchronizácia), alebo bity medzirámcovej medzery (očakáva tvrdú synchronizáciu). Táto komunikácia je zabezpečená prepojením výstupu obvodu **prijimac** s názvom ***hard_sync*** na rovnomenný vstup modulu **prijimac_bitov**. Keď modul **prijimac_bitov** prijme bit, túto skutočnosť oznámi prijímaču signálom ***prijaty_bit***. Zároveň logickú hodnotu tohto prijatého bitu pošle prijímaču pomocou pripojenia ***vystup***. Z modulu **generator_kvant** je naň privedený signál ***kvantum***, ktorým modul **generator_kvant** oznamuje modulu **prijimac_bitov**, že prešlo ďalšie časové kvantum. **Prijimac_bitov** má 4 nastavovacie vstupy (***seg1, seg2, sjw, sampluj 3x***) a **generator_kvant** má jeden nastavovací vstup (***brp***). **Prijimac** má okrem troch uvedených pripojení s modulom **prijimac_bitov** (***hard_sync***, ***prijaty_bit*** a ***vystup***) 4 výstupy pripojené k modulu **registre** (***paket***, ***chyba***, ***chyba8***, ***prijaty_paket***). Keď je nejaký rámec úspešne prijatý, oznámi modul **prijimac** túto skutočnosť signálom ***prijaty_paket*** a na 99-bitovom výstupe ***paket*** sa tento rámec vtedy objaví. Modul **registre** na túto udalosť zareaguje tým, že ho spolu s aktuálnym časom pridá do prijímacej fronty a (ak je to možné) zníži počítadlo chýb prijímača. Naopak s neúspešným prijatím rámca sa aktivuje signál ***chyba*** pripojený s prijímača do registra (zvýši v registroch počítadlo chýb prijímača o 1), s každou hrubou chybou prijímača túto skutočnosť dá **prijimac** na vedomie **registrom** signálom ***chyba8***, čím v nich spôsobí zvýšenie chybového počítadla prijímača o 8. **Generator_kvant** má svoj jediný výstup ***kvantum*** pripojený k modulu **prijimac_bitov**, **vysielac** a **potvrdzovac**. Nastavovací vstup ***brp*** má samozrejme pripojený z modulu **registre** k príslušným vstupným bitom jeho vstupných nastavení. Vysielač na svoju činnosť vyžaduje poznanie svojho chybového čítača privedeného z nastavovacieho výstupu modulu **registre**, aby vedel, či nie je v chybovom stave *bus-off* a teda nemá nič vysielať. Ďalej vyžaduje vstup zo zbernice ***can_in***, ktorý využíva na to, aby pred vyslaním každého rámca zisťoval, či niekto nezačal vysielať v predstihu a aby spätne kontroloval s vysielaním, či toto jeho vysielanie naozaj na zbernici je. Potrebuje poznať, či náš uzol zbernice nie je v *silent mode*, na čo mu slúži vstupný signál ***silent_mode***. No a nakoniec potrebuje ešte poznať 4 skupiny vstupov ktorými sú privedené 4 správy z vrcholov odosielacích front s indikátormi ich prázdnych stavov týchto front. Výstupom obvodu je pochopiteľne signál ***vystup***, privedený na prvý vstup výstupného AND hradla. Jeho druhým vstupom je výstup modulu **potvrdzovac**. Svojimi ďalšími výstupmi ***chyba*** (indikácia chyby pri vysielaní), ***chyba8*** (príznak hrubej chyby vysielania), ***arbitration_lost*** (informácia o prehranej súťaži o právo vysielať), ***chyba_ack*** (nepotvrdenie nami vyslaného rámca), ***odoslane1*** až ***odoslane4*** (oznam o úspešnom odoslaní rámca z konkrétnej fronty) sa pripája k modulu **registre**. Nakoniec má ešte jeden výstup ***vysielame***, ktorý je pripojený k modulu **potvrdzovac** s dôvodu toho,

aby modul **potvrdzovac** nepotvrdzoval rámce, ktoré sa práve vysielajú modulom **vysielac** nášho uzla zbernice. **Potvrdzovac** má okrem tohto vstupu ešte 3 vstupy riadiace jeho činnosť: ***chyba_prijimac***, ***chyba_vysielac*** (príkaz pre vyslanie chybového rámca) a ***prijaty_paket*** (príkaz pre poslanie potvrdzovacieho bitu). Signál ***kvantum*** z modulu **generator_kvant** mu určuje, kedy uplynulo ďalšie časové kvantum. Signál ***silent_mode*** z príslušného bitu v module **registre** informuje o užívateľovom nastavení *silent mode* nášho uzla. Nakoniec aj tento obvod potrebuje k svojej činnosti nastavovacie vstupy privedené z nastavovacích výstupov modulu **registre** a síce ***seg1*** a ***seg2*** a čítač chýb vysielača (***tx_counter***), aby vedel, koľko časových kvánt má jeden bit, ktorý eventuálne bude treba vysielať a aby vedel, či nie je v pasívnom chybovom stave. **Casovac** má jediný výstup ***cas***, ktorý má 32 bitov a informuje prijímaciu frontu a vysielač o aktuálnom čase - počte µs uplynulých od spustenia obvodu. Filtre majú na vstupe signál ***prijaty_paket*** privedený z rovnomenného výstupu modulu **prijimac**. Všetky ostatné vstupy sú nastavovacie (***maska***, ***ID***, zapnutie pre každý zo štyroch filtrov a určenie, či sa jedná o filter akceptujúci rozšírené, či základné rámce). Jediným výstupom tohto modulu je ***prijaty_paket_vystup*** privedený k modulu **registre**, ktoré na jeho základe pridáva, či nepridáva práve prijatý rámec do prijímacej fronty.

Mimo samotného hradlového poľa obsahujúceho celú logiku obvodu sa obvod nezaobíde ešte bez jednej podstatnej veci a tou je prevodník napäťových úrovní z 3,3 Voltovej CMOS na CAN napäťovú úroveň a naspäť, aby naše hradlové pole bolo schopné komunikovať so samotnou CAN zbernicou. Toto zabezpečuje obvod, z ktorého srdcom je integrovaný obvod PCA 82C 250. K nemu je potrebné pridať ďalšie súčiastky podľa nasledovnej schémy.

D1 - Schottkyho dióda
C1 - 100 nF
R1 - 1 kΩ
R2 - 120 Ω



8. Realizácia a testovanie obvodu

Náš obvod je zostavený s použitím vývojového kitu NEXYS2 od firmy Digilent s jadrom v podobe programovateľného hradlového poľa Spartan 3E 1200 od firmy XILINX. Tento kit má 3 možnosti napájania – batéria 3,3 V, USB káblom (5 V), alebo zo sieťového adaptéra (5 – 12 V). Ku kitu musí však byť pripojený ešte obvod PCA 82C 250 ako prevodník napäťových úrovní medzi TTL a CAN, ktorý si vyžaduje ďalšie napájanie 3,3 V privedené z kitu NEXYS 2, navyše 5 V, ktoré musia byť privedené z ďalšieho vlastného zdroja napájacieho napätia, ktorý nie ej súčasťou tejto práce.

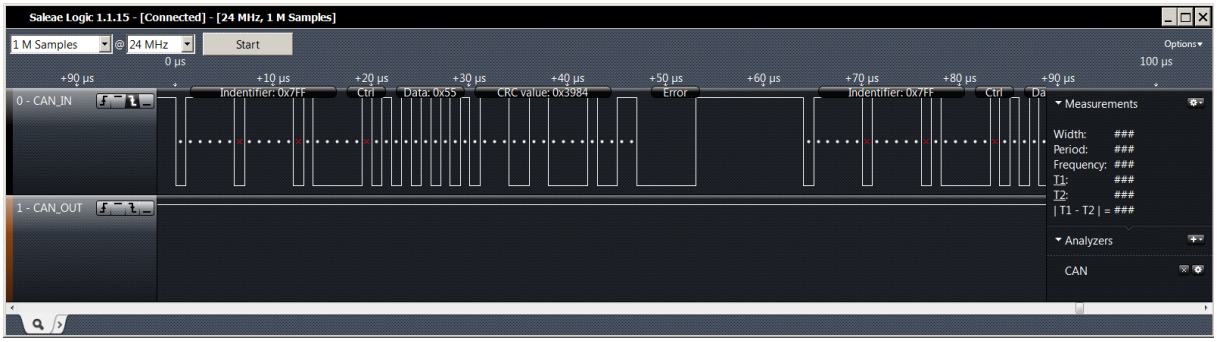
Testovanie obvodu bolo realizované simulátorom ISim od firmy XILINX a komunikáciou s dvomi nezávislými radičmi zbernice CAN. Prvým je PC karta s radičom CAN 80C527 od firmy Intel a druhým je vývojový kit s mikrokontrolérom ARM Cotrex M4 s CAN radičom integrovaným ako

periféria procesora. K overeniu funkčnosti sme navyše využili logický analyzátor SALAE, z ktorého pochádzajú aj screenshoty zobrazené nižšie. Ako nadradený mikrokontrolér bol vyžitý vývojový kit s mikrokontrolérom ARM Cortex M3, ktorý simuloval vyčítavanie a zápis z front.

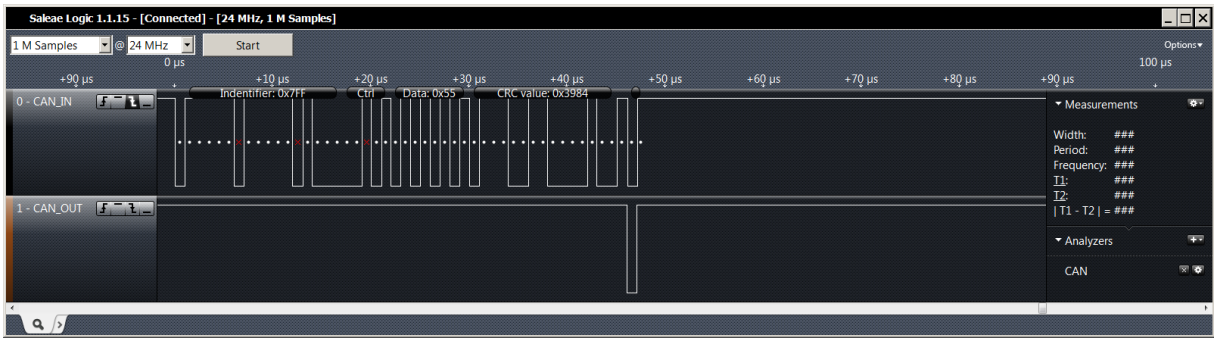
Prijímač s prijímacou frontou bol testovaný aj na vyčítavanie z prázdej fronty. Výsledky testovania hovoria, že fronta nie je nijak ovplyvnená príkazom čítania ani zahodenia rámca v prípade, že je prázdna. Každý rámec, ktorý na zbernicu vyšle iný uzol (v prípade, že prejde akceptančnými filtrami), alebo rámec vyslaný naším uzlom je pridaný do prijímacej fronty, avšak s výnimkou, že táto fronta nie je plná. Fakt, že fronta nie je prázdna, rovnako ako fakt, že je plná je užívateľovi oznámený prostredníctvom príslušného prerušenia, ktoré sa dá vypnúť. Navyše s vyčítaním každého rámca z prijímacej fronty sa tento rámec z fronty zahodí, čiže po vyčítaní rámca sa zníži počet rámcov vo fronte, čo bolo overované vyčítavaním počtu správ pred a po vyčítaní rámca. Zároveň s ďalším vyčítaním rámca z prijímacej fronty je na jej vrchu už ďalší prijatý rámec.

Ďalší test overoval funkčnosť vysielania potvrdzovania (*silent mode*). Výsledky tohto testu sú v tejto práci zdokumentované nasledovnými screenshotmi:

Screenshot znázorňujúci odozvu nášho uzla na prichádzajúci rámec v *silent mode* - rámec nie je potvrdený, a preto vzdialený uzol vysiela najprv chybový rámec a potom sa dáta snaží poslať znova.

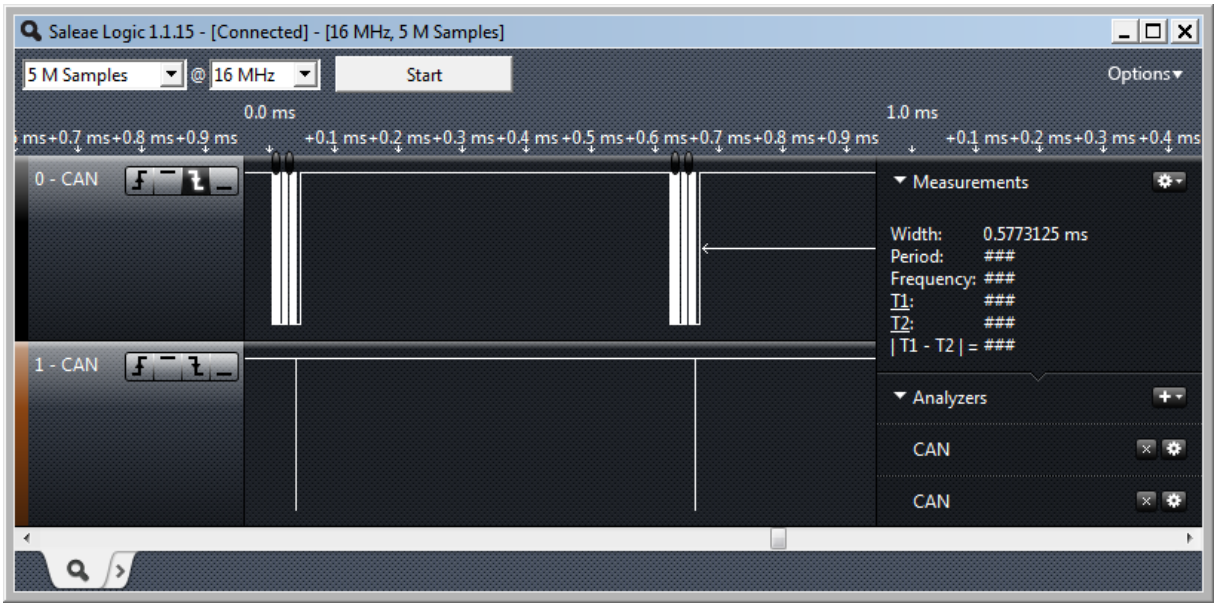


Screenshot znázorňujúci opačnú situáciu, kedy náš uzol nie je v *silent mode* a teda má povolené potvrdzovať prichádzajúce rámce – ako je z obrázka vidieť, rámec vzdialený uzol vyšle iba raz, pretože dostane od nášho uzla potvrdenie o jeho prijatí.

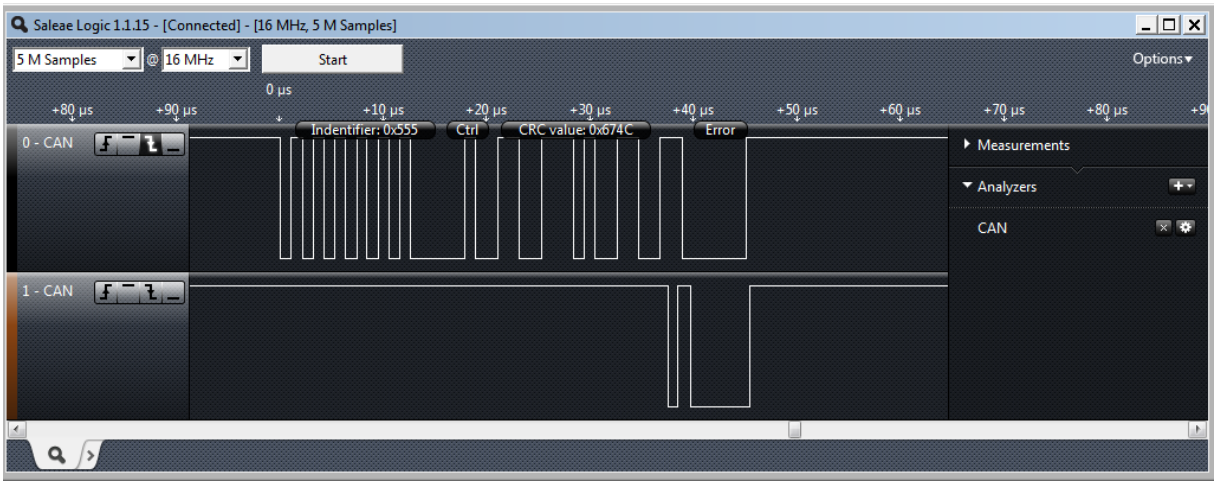


Tretí test overoval funkčnosť odosielania správ nastavených časoch. Nasledovný screenshot dokladá jeho výsledok – dva rámce, ktoré majú byť vyslané v časoch, ktoré sa líšia od seba viac, než bežná medzirámcová medzera boli naozaj odoslané v daných časoch.

Screenshot znázorňujúci vysielanie rámcov v nastavených časoch, nie ihneď za sebou:



Štvrtý test – screenshot znázorňujúci chybovú ozvenu nášho uzla – odozvu na chybový rámec vyslaný iným uzlom:



9. Záver

Podarilo sa mi implementovať IP funkciu štandardného radiča zbernice CAN. Táto štandardná funkcia zahŕňa nami úspešne implementované funkcie prijímania a odosielania správ, reagovanie na chybové stavy zbernice a prechádzanie medzi chybovými stavmi nášho radiča.

Ďalej sa mi podarilo implementovať všetky požadované nadštandardné funkcie: Prijaté správy sa ukladajú do prijímacej fronty spolu s ich časovými značkami, informujúcimi nadradený procesor o čase ich prijatia. Do tejto fronty sa úspešne ukladajú aj správy odoslané naším radičom spolu s časom naozajstného odoslania. Obvod rovnako úspešne implementuje funkciu odosielania správ v ich nastavených časoch a to z viacerých odosielacích front.

Všetky tieto funkcie som úspešne overil testami , o ktorých je bližšie popísané v kapitole 8 – realizácia a testovanie obvodu.

10. Tabuľky

10.1. Tabuľka rozdelenia 99-bitového signálu paket

10 d 0	id zakladne
11	extended
29 d 12	id ext
30	rtr
34 d 31	dlc length
98 d 35	data

	koncový bit	začiatočný bit
data 1	42	35
data 2	50	43
data 3	58	51
data 4	66	59
data 5	74	67
data 6	82	75
data 7	90	83
data 8	98	91

10.2. Tabuľka rozdelenia užívateľsky nastaviteľných registrov

Adresa	**Názov**	**Skratka**	**Popis**
0	Interrupt control register	INTCR	Zapínanie a vypínanie jednotlivých prerušovacích signálov
1	Interrupt status register	INTSR	Flagy jednotlivých prerušení
2	Comamnd register	CMDR	ACTIVE,
3	Receive count register	RXCNT	Počet správ v prijímacej fronte
4	Transmit count register	TXCNT	Počet správ vo vysielacej fronte
5	Baud rate prescaler High	BRPH	Delička pre generovanie hodinového signálu pre CAN, horných 8 bitov
6	Baud rate prescaler Low	BRPL	Delička pre generovanie hodinového signálu pre CAN, dolných 8 bitov
7	Can control register 1	CCR1	synchronization jump with, Silent mode - povolenie potvrdzovania a vypnutie vysielača
8	Can control register 2	CCR2	time segment1, time segment2, sampling 3x
9	Transmit error counter	TECNT	čítač chýb pri odosielaní
10	Receive error counter	RECNT	čítač chýb pri prijímaní
11,12,13,14	Acceptance mask 1	ACTM	Akceptančná maska, zapnutie filtra a rozhodnutie, či sa jedná o rozšírené ID,

			alebo štandardné
15,16,17,18	Acceptance ID 1	ACID	Akceptančné ID
19,20,21,22	Acceptance mask 2	ACTM	Akceptančná maska, zapnutie filtra a rozhodnutie, či sa jedná o rozšírené ID, alebo štandardné
23,24,25,26	Acceptance ID 2	ACID	Akceptančné ID
27,28,29,30	Acceptance mask 3	ACTM	Akceptančná maska, zapnutie filtra a rozhodnutie, či sa jedná o rozšírené ID, alebo štandardné
31,32,33,34	Acceptance ID 3	ACID	Akceptančné ID
35,36,37,38	Acceptance mask 4	ACTM	Akceptančná maska, zapnutie filtra a rozhodnutie, či sa jedná o rozšírené ID, alebo štandardné
39,40,41,42	Acceptance ID 4	ACID	Akceptančné ID

43, 44, 45, 46	Prijímacia fronta ID, rtr, ext
47	počet bytov správy (0 - 8)
48 - 55	Dáta rámca prijímacej fronty
56, 57, 58, 59	čas prijatia správy
60	počet rámcov vo fronte
61, 62, 63, 64	1. vysielacia fronta ID, rtr, ext
65	počet bytov správy (0 - 8)
66 - 73	dáta rámca odosielanej správy
74, 75, 76, 77	čas odoslania správy
78	počet rámcov vo fronte
79, 80, 81, 82	2. vysielacia fronta ID, rtr, ext
83	
84 - 91	dáta rámca odosielanej správy
92, 93, 94, 95	čas odoslania správy
96	počet rámcov vo fronte
97, 98, 99, 100	3. vysielacia fronta ID, rtr, ext
101	
102 - 109	dáta rámca odosielanej správy
110, 111, 112, 113	čas odoslania správy
114	počet rámcov vo fronte
115, 116, 117, 118	4. vysielacia fronta ID, rtr, ext
119	
120 - 127	dáta rámca odosielanej správy
128, 129, 130, 131	čas odoslania správy
132	počet rámcov vo fronte

10.3. Podrobný popis registrov, združujúcich viac nastavení

Pokiaľ bit nie je v tabuľke uvedený, je rezervovaný. Jeho zápisom sa nič nemení.

Adresa	Skratka	Bity	
0	**INTCR**	**7 - 0**	**Zapnutie / vypnutie jednotlivých prerušení**
	7	Bit 7	Počítadlo chýb vysielača plné
	6	Bit 6	Počítadlo chýb prijímača plné
	5	Bit 5	Chyba prijímania rámca
	4	Bit 4	Chyba vysielania rámca
	3	Bit 3	Prechod do pasívneho chybového stavu
	2	Bit 2	Prehratá súťaž o právo vysielať
	1	Bit 1	Naplnenie polovice fronty výstupu
	0	Bit 0	Naplnenie polovice fronty vstupu
1	**INTSR**	**15 - 8**	**Príznaky jednotlivých prerušení**
		Obdobné, ako v registri 0 - INTCR	
2	**CMDR**	**23 - 16**	
	22	Bit 6	Active 0 - neaktívny, 1 - aktívny
	21	Bit 5	Povolenie rozšírených rámcov - 0 iba základné, 1 aj rozšírené
7	**CCR1**	**63 - 56**	
	63 - 62	Bit 7, 6	SJW - synchronization jump width
	61	Bit 5	Vypínanie výstupu 0 - vysielač vyputý, 1 - vysielač zapnutý
	60	Bit 4	Silent mode - zapnutie potvrdzovania 1 - potvrdzovanie zapnuté 0 - potvrdzovanie vypnuté
8	**CCR2**	**71 - 64**	
	71	Bit 7	Trojnásobné vzorkovanie bitu 0 - jedna vzorka, 1 - 3 vzorky
	70 - 68	Bit 6,5,4	time segment 2
	67 - 64	Bit 3,2,1,0	time segment 1

11. Zoznam skratiek a značiek

ACK - *Acknowledge* - potvrdenie správnosti prijatia

ACK_DEL - *Acknowledge delimiter* - oddeľovač potvrdenia správnosti prijatia

BRP - *Baud-rate prescaler* - delička vstupného hodinového signálu

CAD - *Computer-aided design* - druh softvéru vyvinutý na návrh zväčša strojárenských výrobkov

CAN - *Controller Area Network* - Zbernica, používaná najčastejšie pre komunikáciu senzorov, akčných členov a mikrokontrolérov, zväčša v automobiloch

CRC - *Cyclic redundancy check* - kontrolný súčet, používaný pre kontrolu správnosti prenesených údajov

CRC_DEL - *Cyclic redundancy check delimiter* - Oddeľovač kontrolného súčtu

DLC - *data length code* - štvorbitový kód oznamujúci, koľko je v rámci bytov dát

EOF - *End of frame* - koniec rámca, jedná sa o 7 bitov vysielaných v recesívnej úrovni

EXT - *Extended* - bit indikujúci, či je rámec rozšírený, alebo štandardný

EXT_ID - rozšírený 18-bitový identifikátor rámca

I2C - *Inter-Integrated Circuit* - rozhranie používané ku komunikácii medzi integrovanými obvodmi

ID - Identifikátor

INTERFR - *Inter-frame space* - medzirámcová medzera

LOG - logický

SEG1 - *Time segment 1* - prvý časový segment trvania bitu

SEG2 - *Time segment 2* - druhý časový segment trvania bitu

SJW - *Synchronisation jump width* - maximálny počet kvánt, o ktoré sa môžu hodiny prijímača resynchronizovať

SOF - *Start of frame* - začiatok rámca - 1 dominantný bit, čiže vysielaný úrovni v logická 0

SPI - *Serial Peripheral Interface* - rozhranie používané ku komunikácii medzi integrovanými obvodmi

SRR - *Substitute remote request* - Bit nahrádzajúci bit príznaku žiadosti o dáta v rozšírenom rámci

TQ - *Time Quantum* - časové kvantum

RTR - *Remote transmission request* - žiadosť o dáta zo vzdialeného uzla zbernice

VHDL - *VHSIC Hardware Description Language* - jazyk popisujúci hardware, používaný na návrh zákazníckych digitálnych integrovaných obvodov

12. Odkazy a referencie

Etchberger, K.: Controller Area Network, IXXAT Automation 2001, ISBN 978-300-0073762

http://www.ti.com/lit/an/sloa101a/sloa101a.pdf 13. 2. 2013

http://rs232-rs485.blogspot.cz/2009/11/can-bus-message-frames-standard-data.html 13.2.2013

http://www.port.de/pdf/CAN_Indroduction_I+ME.pdf 13.2.2013

http://www.softing.com/home/en/industrial-automation/products/can-bus/more-can-bus/data-frame/arbitration-field.php 3.4.2013

http://canbuskit.com/what.php 28.3.2013

http://ww1.microchip.com/downloads/en/AppNotes/00754.pdf 4.4.2013

http://ghsi.de/CRC/index.php?Polynom=1100010110011001&Message=100 29.4.2013