ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

Řešení komunikace mezi systémem NX5030 firmy INTRONIX a sériově vyráběnými automaty PLC.

Praha 2004 Michal Ditrich

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne …… ……

Podpis

Poděkování

Na tomto místě bych rád poděkoval Ing. Miroslavu Růžičkovi za poskytnutí prostředků pro tvorbu této diplomové práce, Ing. Pavlu Burgetovi, Ing. Petru Černohorskému a Ing. Pavlu Píšovi za poskytnutí cenných rad a v neposlední řadě svému nejbližšímu okolí za podporu při studiu.

ANOTACE

Diplomová práce se zabývá modernizací řídicího systému pro brousicí automaty NX5030 firmy INTRONIX s.r.o. a je koncipována do dvou částí:

První se zabývá řešením komunikace mezi řídicím systémem NX5030 a průmyslově vyráběnými automaty PLC. Výsledkem je navržení sériového komunikačního protokolu a vytvoření programů pro PLC automaty zajišťující komunikaci.

Druhá se zabývá řadičem pro sběrnici CAN, který byl naprogramován v jazyce VHDL a implementován v hradlovém poli Spartan XC2S100 firmy Xilinx.

THE ANNOTATION

The diploma dissertation deals with modernization of grinder machine controller NX5030 system and is divided into two parts:

The first one is about communication between control system NX5030 and programmable logic controllers. The Result is serial communication protocol and programs for PLCs handling communication.

In the second part CAN controller is described written in VHDL language and verified in gate array Spartan XC2S100 made by XILINX.

Obsah

1. ROZBOR ZADÁNÍ

Diplomová práce, zadaná a vykonávaná ve firmě INTRONIX s.r.o, se zabývá modernizací řídicího systému pro brousicí automaty NX5030 a je rozčleněna do dvou částí. První část, zabývající se komunikací řídicího systému s PLC automaty, je již dokončena a inovované řídicí systémy se již prodávají zákazníkům. Druhá část diplomové práce se zabývá komunikací mezi řídicím systémem a Servopohony Servostar 600 firmy Danaher Motion Kollmorgen [1] . Povely pro servopohony je možno zadávat v základní verzi buď pomocí analogového signálu v rozsahu ±10V, nebo přes sběrnici CAN. První zmiňované řešení se používá dosud a druhé řešení bylo motivací pro druhou část mé diplomové práce.

1.1. POPIS ŘÍDICÍHO SYSTÉMU

Řídicí systém NX5030 je určen pro řízení brousicích strojů a sestává se ze tří částí:

1. Procesorové desky obsahující dva osmibitové mikroprocesory DS80C320, hradlové pole XILINX XC4005 a podpůrné obvody starající se o výpočet algoritmu řízení, ovládání výstupních servomotorů, zpracování údajů z inkrementálních čidel a komunikaci se zbylými dvěmi částmi ŘS.Řídicí systém řídí přísun brousicího kotouče (osa X) a posun brouseného předmětu (osa Z) pomocí servozesilovačů Servostar 600. Zpětná vazba je uzavřena přes enkodéry udávajícími polohu obou os. Dále systém přijímá signál od ručních koleček, kterými lze ručně pohybovat oběma osami stoje a umožňuje ovládat rychlost otáčení brousicího kotouče a brouseného válce.
2. Desky vstupů-výstupů poskytující binární výstupní signály a zpracovávající signály z binárních vstupů (koncové spínače a pod.) a čtyř analogových vstupů.
3. Desky pro komunikaci s obsluhou, která existuje ve dvou modifikacích: v levnější, obsahující dvouřádkový šestnáctisegmentový displej, čtyři tlačítka a přepínač a v komfortnější, která používá jako zobrazovací jednotku barevný LCD display s úhlopříčkou 6 palců viz obr. 2.

Všechny tři části spolu komunikují pomocí sériového protokolu, jako médium je použit optický kabel, jehož velikou výhodou je galvanické oddělení jednotlivých částí a minimální náchylnost na rušení, což umožňuje v případě rozlehlejších strojů rozmístit jednotlivé části ŘS na různá místa nejblíže připojovaným zařízením při minimálním riziku rušení. Na obr. 1 je zjednodušené schéma řízení brusky blokově zobrazující servosmyčky a připojení řídícího systému NX 5030 k technologii stroje.

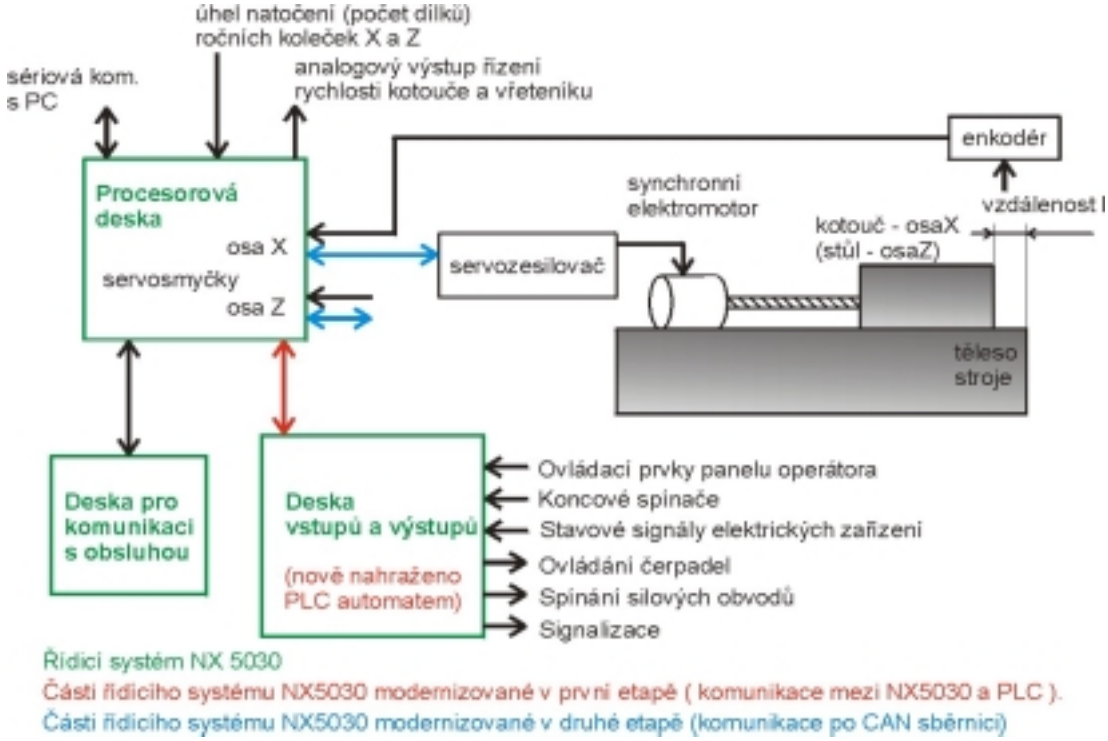
1.2. SÉRIOVÁ KOMUNIKACE

Úvodní část práce byla motivována možností nahradit druhou jmenovanou část řídicího systému – desku vstupů a výstupů PLC automatem, který by dále mohl obstarávat činnosti spojené s logickým řízením stroje, jako je obsluha koncových členů, kontrola hydraulických okruhů a pod. Z dostupných PLC stavebnic na trhu byly vybráni dva zástupci: automat TECOMAT 600 a SIMATIC S7-200. Pro tyto automaty bylo požadováno vytvoření univerzálního komunikačního protokolu a podpůrných programů.

1.3. ŘÍZENÍ SERVOPOHONŮ

Druhá etapa práce byla motivována potřebou modernizace způsobu řízení servopohonů. V současném systému je řízení prováděno analogovým signálem v rozsahu ±10V. Toto napětí je získáváno odfiltrováním střídavé složky PWM signálu, generovaném v hradlovém poli řídicího systému. V současné době je zákazníky jako servozesilovač výhradně používán typ SERVOSTAR 600. Ten provádí číslicové řízení servosmyčky a podporuje ovládání pomocí několika průmyslových sběrnicových standardů, mezi nimiž je i sběrnice CAN.

Požadavkem tedy je vytvořit řadič sběrnice CAN, který lze implementovat do hradlového pole v řídicím systému, umožňující komunikovat se servozesilovači, popřípadě dalšími zařízeními připojitelnými na tuto sběrnici, jako jsou například sledovací měřidla.



obr. 1 - Zjednodušené schéma řízení brusky

obr. 2 - Panel operátora

2. SÉRIOVÁ KOMUNIKACE MEZI NX5030 A PLC

Téma této části diplomové práce je náhrada části ŘS NX 5030- desky vstupů a výstupů průmyslově vyráběným programovatelným automatem. Za reprezentanty automatů byly zvoleny nejprve systém firmy TECO – Tecomat a posléze Siemens – Simatic.

2.1. VOLBA DRUHU AUTOMATU

Hlavní funkcí automatu je distribuce vstupů/výstupů blíže ke koncovým členům, automat sice vykonává základní řízení typu ošetření mezí pohybu koncovými spínači, řízení však není příliš složité, proto požadavek na výkonnost nebyl výrazný, hlavními požadavky na oba automaty byly počet binárních výstupů a počet binárních vstupů. Požadavkům vyhověly dva typy: Tecomat TC605 a Simatic S7 řady 200 CPU 226 s katalogovým číslem 6ES7 216-2BD22-0XB0.

Technické parametry automatů:

Automaty byly vybírány podle svých technických parametrů, aby vyhověly zadané úloze a cenové hladině.

Tecomat TC605:

Počet binárních vstupů: 12
Počet analogových vstupů: 4
Počet reléových výstupů: 8
Jeden sériový kanál s rozhraním RS-232 pro komunikaci s nadřazeným systémem (počítačem).
Volitelně další dva sériové kanály s rozhraními RS-232, RS-422, RS-485

Simatic SC 7-200 CPU 226:

Počet binárních vstupů: 24
Počet reléových výstupů: 16
Dva sériové kanály s rozhraním RS-485.

2.2. FYZICKÉ PŘIPOJENÍ

Zde bylo voleno podle možností, které programovatelné automaty nabízejí a to jsou: Programovatelný automat firmy TECO obsahuje jeden sériový kanál používající rozhraní RS-232 a dále volitelné dva kanály používající rozhraní RS-232, RS-422 nebo RS-485.

Automat firmy Siemens je v tomto skromnější, nabízí sice rovnou dva sériové kanály, oba však podporují pouze rozhraní RS-485.

Jedním z hlavních požadavků na celé řešení je jeho univerzálnost, proto bylo v tomto případě zvoleno rozhraní RS-485 jako způsob komunikace mezi ŘS a automatem. Kromě možnosti požití automatů obou výrobců má řešení další výhodu v možnosti připojení více zařízení než dvě na tuto sběrnici. Např. v sortimentu fy. Intronix s.r.o. existuje zařízení pro snímání údajů z indukčních snímačů komunikující také přes rozhraní RS-485, kterým je možné rozšířit řídicí systém brusky o měření průměru hřídele v čase broušení.

2.3. KOMUNIKAČNÍ MOŽNOSTI AUTOMATŮ

Požadavkem je vytvořit komunikaci mezi oběma zařízeními typu Master – Slave, kdy master, což je v tomto případě řídicí systém, vysílá v intervalu 20ms paket obsahující 3byte dat pro výstupy automatu a ten mu obratem pošle zpět zprávu obsahující 5 byte binárních vstupů.

Oba automaty umožňují použít sériové kanály v několika režimech, z nichž jeden je uživatelsky konfigurovatelný, v automatu TECO se nazývá režim UNI, v automatu SIMATIC režim FREEPORT, tyto režimy byly použity pro další návrh.

Při vytváření konečné podoby protokolu byla opět významná snaha o vytvoření protokolu univerzálního, který by se dal implementovat v obou automatech.

2.3.1.Komunikační vlastnosti automatu TECOMAT TC 605

Tento automat podporuje celkem šest režimů komunikace, z nichž pět pracuje se zabudovaným protokolem a režim UNI určený pro univerzální použití, pro něž se musí protokol nadefinovat. Tento režim byl použit pro další návrh.

Komunikační parametry podporované automatem:

Přenosová rychlost: 50-115200Bd.
Počet přenášených bitů v jednom bytu: 7 nebo 8
Parita: lichá/sudá/bez

Přijatá data jsou ukládána do přijímacího bufferu, je-li celá zpráva bezchybně přijata, je alternován indikační bit ve stavovém slově.

Data k odvysílání uložená ve výstupním bufferu jsou vyslána po alternaci bitu v řídicím slově vysílání.

Komunikační služby obecného uživatelského kanálu automatu TC605:

Detekce počátečního znaku zprávy
Detekce koncového znaku zprávy
Potvrzení bez dat
Adresa stanice
Kontrolní součet
Délka dat
Maximální délka zprávy
Opačná parita prvního znaku zprávy
Klid na lince

2.3.2.Komunikační vlastnosti automatu SIMATIC S7-200

Automat podporuje celkem tři zabudované protokoly přenosu a to: multipoint interface MPI, point-to-point interface PPI a Profibus. Kromě toho je dále možné v režimu FREEPORT mode způsob komunikace definovat. Tento režim byl také použit.

Komunikační parametry podporované automatem:

Přenosová rychlost: 1200-115200Bd.
Počet přenášených bitů v jednom bytu: 7 nebo 8
Parita: lichá/sudá/bez

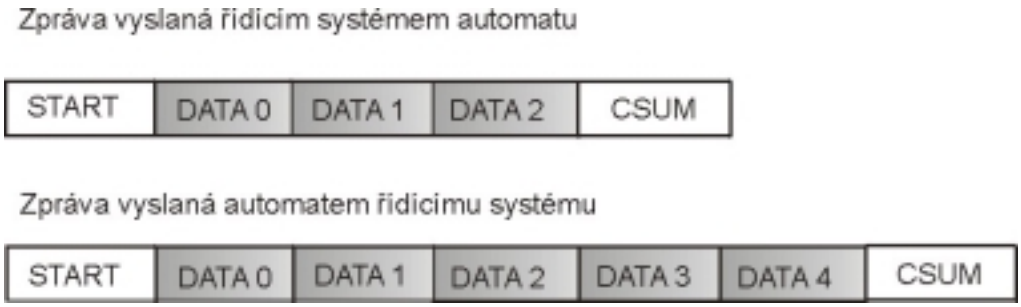
Pro komunikaci jsou vyhrazeny dvě instrukce RCV a XMT. Při příjmu je přijetí bezchybného paketu indikováno vygenerováním přerušení a podobně po zadání odeslání je konec činnosti indikován přerušením.

Komunikační služby obecného uživatelského kanálu automatu Simatic S7-200:

Klid na lince
Detekce počátečního znaku zprávy
Detekce koncového znaku zprávy
Časový interval mezi znaky
Časovač délky zprávy
Maximální počet znaků zprávy

2.4. KOMUNIKAČNÍ PROTOKOL:

Zprávy jsou mezi jednotlivými účastníky komunikace vysílané po bytech, první byte je vždy START s hodnotou AA(h), následují datové byty a zprávu zakončuje byte s kontrolním součtem, viz obr. 3.



obr. 3 - Komunikační protokol

Každý BYTE obsahuje start bit, osm datových bitů, paritu a jeden stop bit.

Pro identifikaci počátku zprávy je použito služeb detekce klidu na lince a detekce počátečního znaku, pro identifikaci konce je použita pevná délka zprávy. Byty typu koncový znak, adresa stanice, počet datových slov a pod. nebyly v tomto případě použity z důvodu maximální jednoduchosti a minimalizace přenosové doby zprávy.

Původně byl požadavek na kontrolní součet ve tvaru doplňku součtu všech předcházejících bytů do nuly z důvodu jednoduššího vyhodnocení, kdy bezchybně přijatý paket má součet všech bytů od prvního po CSUM bez přenosu řádů roven nule. Tuto funkci však ani jeden z automatů nepodporuje, SIMATIC nepodporuje kontrolní součet vůbec, TECOMAT ano, ale pouze v kladné formě. Ve výsledku je tedy použit kontrolní součet ve tvaru součtu všech byte předcházejících CSUM bez přenosu do vyšších řádů a v automatu SIMATIC byla tato funkce dopsána.

2.5. POMOCNÉ PROGRAMY

Pro testovací a servisní účely byly vytvořeny tři programy, z důvodu co nejmenších nároků na počítač byly napsány pod operační systém MS-DOS, lze je však použít i v prostředí MS-Windows 95,98.

2.5.1. Program KOMTEST pro PC simulující řídicí systém NX5030.

Program pracuje v prostředí MS-DOS a slouží pro simulaci řídicího systému při zkoušení jednotlivých komunikačních možností programovatelných automatů.

Při činnosti je řídící systém propojen s počítačem pomocí převodníku rozhraní RS485/RS232.Parametry typu přenosová rychlost v Baudech, počet datových bitů, počet stop bitů a přítomnost parity jsou zadávány při startu programu, při běhu jsou hodnoty těchto parametrů zobrazeny v pravém horním rohu.

Obrazovka, viz. Příloha 1, je rozdělena do dvou oken – dolní se týká odeslaného paketu do automatu a horní paketu přijatého.

Program při svém běhu cyklicky každých 20ms odvysílá paket k vyslání a vzápětí čeká na paket příchozí, přičemž platné jsou pouze pakety začínající platným start byte a nejsou-li kratší než zadaná velikost.

U paketu určenému k vyslání lze měnit jeho délku v rozsahu 2 -7 byte (0-5datových byte) a dále hodnoty start byte i datových byte, hodnoty se zadávají a jsou zobrazeny v hexadecimální soustavě. Hodnota posledního byte CSUM je dopočítána automaticky jako osmibitový součet všech předchozích byte bez přenosu řádu.

U přijímaného paketu lze měnit jeho očekávanou délku a hodnotu start byte, jednotlivá slova z přijatého paketu jsou potom vypsána na obrazovku, vypočtená hodnota CSUM je poté porovnána s posledním bytem z paketu a výsledek porovnání je opět vypsán na obrazovku.

Pro kontrolu správnosti komunikace mezi zařízeními jsou použity dva způsoby a to pokud není v době mezi dvěma odeslanými pakety žádný paket přijat, je vypsáno na obrazovku příslušné hlášení a dále je počítána procentuelní úspěšnost přenosu jako podíl počtu přijatých bezchybných paketů za dobu odeslání posledních 100paketů. Tento parametr je výhodný pro testování vyšších rychlostí a vlivu rušení a délky vodičů sběrnice na činnost zařízení.

2.5.2. Program SIMAUT simulující PLC automat

Program je napsán pod operační systém MS-DOS a je určen jako pomocný program psaní a odlaďování programů pro řídicí systém, kdy umožňuje simulovat automat PLC a s využitím skriptování umožňuje simulovat celou technologii.

Funkčně program nahrazuje činnost automatu, tj. po úspěšném přijetí zprávy zobrazí její obsah a okamžitě vyšle zprávu s daty z vysílacího zásobníku.

Obsah přijaté i vysílané zprávy je zobrazen na obrazovce, viz Příloha 2, jednotlivé bity lze pojmenovat pomocí textového souboru, který se programu předává při spuštění jako parametr.

2.5.2.1. SYNTAXE SOUBORU S POJMENOVÁNÍM VSTUPŮ A VÝSTUPŮ:

Části pojmenování vstupů předchází řádek s textem "In", předchozí řádky jsou ignorovány, řádky následující po "In" jsou po řadě přiřazeny vstupům, je-li jich méně, zbývající názvy budou prázdné.

Délka textu na řádce je omezena na 13 znaků, je-li text delší, text přesahující 13 znaků bude ignorován.

Část pojmenování výstupů předchází řádek s textem "Out", řádky následující po "Out" jsou po řadě přiřazeny výstupům, je li jich méně, zbývající názvy budou prázdné.

Délka textu je omezena na 13 znaků, je-li text delší, text přesahující 13 znaků bude ignorován.

Vysílaná data lze editovat a to buď přímo na obrazovce, kdy lze šipkou vybrat konkrétní bit a klávesou Enter změnit jeho hodnotu, nebo pomocí skriptu, který mění hodnoty ve výstupním zásobníku při běhu programu. Skript se zapisuje do textového souboru, jeho jméno se předává programu jako třetí nepovinný parametr.

2.5.2.2. SYNTAXE SKRIPTOVACIHO SOUBORU:

Skript slouží pro zadání hodnot, které budou v programu přiřazeny vstupům, dále označeno hodnota vstupu. Jednotlivé hodnoty mohou být odděleny čekáním po časový interval, čekáním na hodnotu výstupního bitu a dále podmíněným a nepodmíněným skokem.

Formát hodnoty vstupu:

Pět po sobě jdoucích hexadecimálních dvojčíslic, které mohou být odděleny mezerami, přípustné jsou znaky 0-9, A-F (a-f).

Vzor přípustných hodnot vstupu:

```
00 11 BB CC DD
0011223344
aa bb 00 11 22
```

Je li číslic na řádku více než 10, ostatní jsou ignorovány, je-li jich méně, je ignorován celý řádek.

Syntaxe skriptovacích příkazů:

Příkazy je možné psát celé velkými písmeny, celé malými písmeny nebo první velké písmeno a ostatní malé.

Přípustné formy příkazu delay, pro ostatní je zápis analogický:

```
DELAY
delay
Delay
```

Při své činnosti program načte ze skriptovacího souboru hodnotu vstupu a podmínku, po kterou bude tuto hodnotu prezentovat na svém výstupu, vyprší-li podmínka, načte další hodnotu vstupu a další podmínku, popřípadě skočí na jiné místo v programu, dojde-li na poslední hodnotu ve skriptu, použije ji, oznámí to uživateli a ukončí práci se skriptem.

Program po spuštění provede lexikální analýzu skriptovacího souboru, aby byly případné chyby ve skriptu odhaleny hned a ne až při běhu programu.

Komentáře je možno psát za znak %

2.5.2.3. POPIS PŘÍKAZŮ:

Program podporuje čtyři příkazy, pomocí kterých lze popsat činnosti probíhající při reálném řízení stroje. Syntaxe podporovaných příkazů je popsána v tomto odstavci.

Formát příkazu Delay:

Příkaz obsahuje klíčové slovo DELAY a po něm číselný údaj v sekundách, po který bude program čekat, přičemž bude poskytovat předešlou hodnotu vstupu.

Vzor zápisu příkazu:

```
Delay 15                    % příkaz čeká po dobu 15sec.
```

Forma příkazu Wait until :

Příkaz obsahuje klíčová slova Wait until a po nich podmínku, na kterou se čeká, tj. program čeká do doby, než je podmínka splněna, je-li splněna ihned, je ignorována. Podmínka je ve tvaru Outx = 1, kde za x se zapíše číslo výstupu číslované od nuly a za rovnítkem může být 0 nebo 1. Místo klíčového slova Out, může být použit název výstupu shodný s názvem v souboru s pojmenování vstupů a výstupů.

Vzor zápisu příkazu:

```
Wait until Out6 = 1         % Čeká do doby, než hodnota šestého výstupu je
                            jedna.
Wait until KAC = 0          % Čeká do doby, než výstup pojmenovaný KAC
                            bude mít hodnotu nula.
                            % Pozor!!! Název KAC musí být definován
                            v souboru pojmenování vstupů a výstupů,
                            defaultně names.ini
```

Formát příkazu Jump:

Příkaz obsahuje klíčové slovo Jump a po něm návěští, které se musí vyskytovat ve skriptu, kde mu předchází znak ":". Přítomnost návěští je kontrolována. Program provede skok na návěští, kde očekává opět hodnotu vstupu nebo podmínku.

Vzor zápisu příkazu:

```
Jump Názevnávěští
                            % jinde v programu musí být řádek:
:Názevnávěští
```

Formát příkazu If Jump:

Příkaz obsahuje klíčové slovo If, poté podmínku ve formátu stejném jako u příkazu Wait until, dále klíčové slovo Jump a návěští, kam se provede skok programu. Program při tomto příkazu vyhodnotí podmínku a je-li splněna, provede skok na zadané návěští, není-li podmínka splněna, je příkaz ignorován.

Podmínka je ve tvaru Outx = 1, kde za x se zapíše číslo výstupu číslované od nuly a za rovnítkem může být 0 nebo 1. Místo klíčového slova Out, může být použit název výstupu shodný s názvem v souboru pojmenování vstupů a výstupů.

Po klíčovém slově Jump následuje návěští, které se musí vyskytovat ve skriptu, kde mu předchází znak ":". Přítomnost návěští je kontrolována. Program provede skok na návěští, kde očekává opět hodnotu vstupu nebo podmínku.

Vzor zápisu příkazu:

`If Out6 = 1 Jump konec`	% Pokud výstup s číslem šest číslovaný od nuly má hodnotu jedna, program provede skok na návěští "konec".
`If KAC = 0 Jump cyklus`	% Skok na návěští "cyklus" se provede pokud výstup pojmenovaný KAC bude mít hodnotu nula. % Pozor!!! Název KAC musí být definován v souboru pojmenování vstupů a výstupů, defaultně names.ini

2.5.2.4. VZOROVÝ PŘÍKLAD SKRIPTU:

`AA BB CC DD F1`	%Inicializační hodnota
`Delay 5`	%Čeká po dobu pěti sekund
`%Toto je komentář`	
`:cyklus`	%Návěští použité v příkazu skoku.
`AA BB CC DD F3`	%Následná hodnota vstupu
`delay 10`	%Čeká po dobu deseti sekund
`00 11 22 33 44`	
`Wait until KAC = 0`	%Načte hodnotu vstupu a čeká, než výstup pojmenovaný KAC bude mít hodnotu 0, tento název musí být definován v souboru pojmenování vstupů.
`if Out6 = 1 Jump konec`	%Podmíněný skok, pokud bude mít šestý výstup číslovaný od nuly hodnotu 1, provede program skok na návěští konec
`AA BB CC DD F4`	
`Delay 5`	%Čeká po dobu pěti sekund
`AA BB CC DD F5`	
`jump cyklus`	% nepodmíněný skok na návěští "cyklus"
`:konec`	

2.5.3. Program AGENT monitorující komunikaci

Program je napsán pod operační systém MS-DOS a je určen jako pomocný program pro monitorování komunikace mezi řídicím systémem a automatem PLC. Používá se především jako pomocný prostředek pro odstraňování problémů s komunikací při spouštění nových strojů.

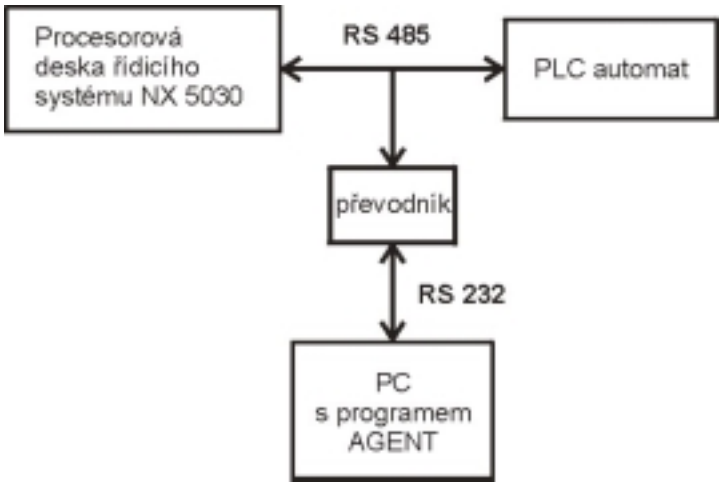
Na obrazovce, viz Příloha 3, jsou vypisovány jednotlivé bity zpráv od obou účastníků komunikace, každý z bitů lze pojmenovat v externím textovém souboru.

Dále jsou zobrazovány stavy komunikace a to:

- žádný ze systémů nekomunikuje,
- zprávy vysílá pouze master,
- komunikace v pořádku.

Připojení PC s programem AGENT ke sběrnici

Použitá sběrnice RS 485 [15] umožňuje připojení více než dvou zařízení ke sběrnici, PC s programem AGENT je připojeno k datovému kabelu mezi oběma účastníky komunikace pomocí dodávaného kabelu, viz obr. 4. Sériová linka RS 232, běžně používaná v osobních počítačích, je připojena ke kabelu pomocí převodníku mezi rozhraními RS 232 a RS 485.



obr. 4 - připojení PC s programem AGENT

2.6. VYTVOŘENÍ PROGRAMŮ PRO PROGRAMOVATELNÉ AUTOMATY ZAJIŠŤUJÍCÍ KOMUNIKACI

Pro oba dva typy programovatelných automatů byly vytvořeny programy, obstarávající komunikaci mezi nimi a řídicím systémem. Programy zajišťují konfiguraci a inicializaci příslušného sériového kanálu, zpracování dat z došlé zprávy a odvysílání zprávy s daty pro řídicí systém jako odpověď.

2.6.1. Tecomat

Pro programování automatu firmy TECO byl použit program Mosaic verze 1.1.

Parametry komunikačního kanálu se nastavují pomocí inicializační tabulky, pro snadnější práci Mosaic obsahuje formulář, viz Příloha 4, kde lze všechny požadované parametry komunikace nastavit a který sám zajistí převod údajů do tabulky.

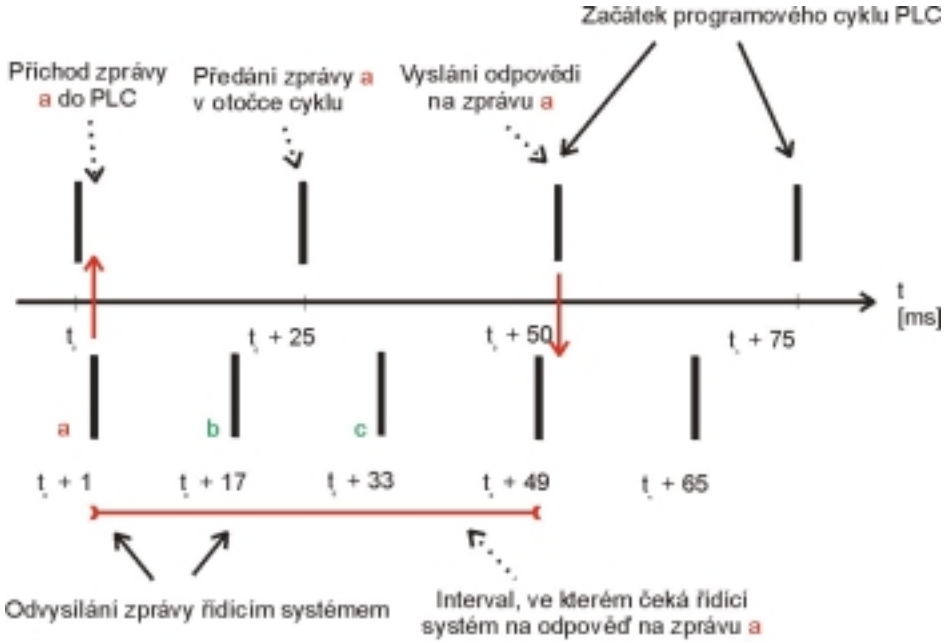
Příjem zprávy z komunikačního kanálu je v tomto automatu indikován dvěma způsoby a to alternací bitu ARC v bytu STAT [2] a vznikem přerušení, přičemž o druhé z možností se manuály od výrobce z neznámého důvodu zmiňují velice sporadicky, přestože tento způsob ošetření příjmu zprávy se nakonec ukázal jediný možný pro tuto úlohu.

Pří počátečních zkouškách byla přijatá zpráva indikována pomocí alternovaného bitu ARC vždy na začátku programového bloku PLC. Po zjištění příjmu zprávy byly zaktualizovány registry vstupních dat přijatými daty z komunikačního kanálu a naopak data

pro odvysílání byla nahrána do výstupního bufferu. Po tomto bloku následoval již samotný program v PLC.

Při úvodním testování měl vlastní program v PLC prakticky nulovou délku, komunikace fungovala bezproblémově a odezva mezi vysílanou zprávou a přijatou odpovědí ze strany řídicího systému se pohybovala mezi 5-7ms. Tento rozptyl je dán rozdílnou délkou programových cyklů v PLC Tecomat z důvodu procesů spouštěných časovou podmínkou, procesů spouštějících se v každém druhém, třetím... cyklu a přerušujících procesů. Protože rozestup zpráv vysílaných masterem byl 16ms, komunikace probíhala v pořádku.

Po nasazení do provozu se však ukázalo, že skutečná délka programového cyklu se pohybuje mezi 18-25ms. Při detekci přijaté zprávy pomocí vyhodnocení hodnoty bitu ARC se jeho hodnota aktualizuje vždy v otočce cyklu a taktéž požadavek na vysílání je akceptován vždy v otočce cyklu. To však znamená, že maximální doba odezvy systému na přijatou zprávu je rovna dvojnásobku nejdelší programové smyčky. To je v tomto případě Tmax = 50ms, což je více než trojnásobek rozestupu mezi zprávami vysílanými masterem a ten nahlásí chybu komunikace. Situace je znázorněna na obr. 5.



obr. 5 - Kolize na Tecomatu

Jako reakce na tuto situaci byl zvětšován rozestup mezi zprávami vysílanými masterem a to až na pokusnou maximální hodnotu 80ms, která už je nepoužitelná pro řízení stroje z důvodu velikého zpoždění. Při zvětšování rozestupu byl registrován pokles počtu chybových hlášení o špatné komunikaci, nedošlo však k jejich odstranění, protože se projevoval další efekt v důsledku použití sběrnice RS 485, kde vysílač i přijímač obou účastníků komunikují po stejných vodičích, a proto mohlo dojít k poškození zpráv v důsledku současného vysílání obou účastníků. Na tom má hlavní příčinu rozptyl mezi časy jednotlivých programových smyček PLC Tecomat.

Jako konečné bylo zvoleno ošetření příjmu zprávy v PLC pomocí přerušovacího procesu P45. Přerušovací proces je vhodný pouze pro nejnutnější a co nejkratší operace, aby příliš neprodlužoval délku programového cyklu PLC, to však pro tuto aplikaci plně vyhovuje. Přerušovací procedura stejně jako v minulém případě aktualizuje registry vstupních dat

přijatými daty z komunikačního kanálu a naopak data z výstupního registru nahraje do výstupního bufferu. Je zde však riziko, že zpráva do automatu přijde dvakrát v době jedné programové smyčky a proto pro odstranění možnosti vzniku nekonzistence dat jsou vstupní a výstupní registry pouze dočasné a k aktualizaci skutečných vstupních a výstupních registrů dojde vždy na začátku programové smyčky. Toto řešení má za následek, že pokud v jedné programové smyčce přijde více než jedna zpráva, tak se do další programové smyčky předá poslední z nich a všechny předchozí jsou ignorovány.

Celkově se tedy problém s komunikační odezvou přesunul na problém s aktualizací dat, v posledně popsaném řešení nejdelší doba odezvy aktualizovaných dat na jejich požadavek bude rovna času programové smyčky a rozestupu mezi dvěma zprávami vysílanými masterem.

Při ošetření komunikace pomocí přerušení je doba odezvy mezi vyslanou a přijatou zprávou masterem 2-3ms, což při rozestupu zpráv vysílaných masterem znemožňuje současné vysílání obou vysílačů a tím znehodnocení zpráv.

Operační systém PLC Tecomat se sám stará o vrácení původní hodnoty registrů a zásobníku po skončení přerušení, proto není třeba toto řešit uživatelsky.

2.6.2.Simatic

Struktura programu pro tento automat je obdobná jako u předešlého s tím, že pro detekci konce komunikační služby se používá přerušení a pro inicializaci komunikace je potřeba zapsat požadované údaje do oblasti řídicích registrů vymezené příslušnému sériovému kanálu.

2.7. ZÁVĚR PRVNÍ ETAPY :

Tato etapa je již ukončena, jejím výsledkem je inovovaný řídicí systém s firemním označení NX 5031, který neobsahuje desku vstupů a výstupů a místo ní má zabudované prostředky pro komunikaci s PLC automaty podle komunikačního protokolu popsaného v této práci. Při vývoji a implementaci řešení byly největší problémy s rychlostí PLC automatu TECOMAT TC600, který při zpracování komunikačních požadavků v programové smyčce nestíhal odpovídat na zprávy vysílané řídicím systémem. Jediné možné konečné řešení bylo obsluha komunikace pomocí přerušení.

3. NÁVRH ŘADIČE SBĚRNICE CAN

Motivací pro tuto část diplomové práce bylo navrhnout řadič sběrnice CAN implementovaný do hradlového pole firmy XILINX, který bude v co největší míře kompatibilní s již existujícími řadiči ve formě integrovaného obvodu, konkrétně typu SJA1000 firmy Philips [14] Tomuto zadání předcházel semestrální projekt z předmětu Návrhy automatizovaných zařízení, který se týkal ověření funkčnosti řadiče CAN, jehož popis a zdrojové kódy byly uveřejněny na internetové stránce viz. [4] Ověřit funkčnost tohoto řadiče se nepodařilo z důvodu v té době nekompletního kódu a nulové dokumentace k němu, stal se však inspirací pro tento projekt.

Řadič je napsán v programovacím jazyce VHDL [12] pro testovací účely byl implementován v hradlovém poli SPARTAN II XC2S100 firmy Xilinx.

3.1. POPIS SBĚRNICE CAN

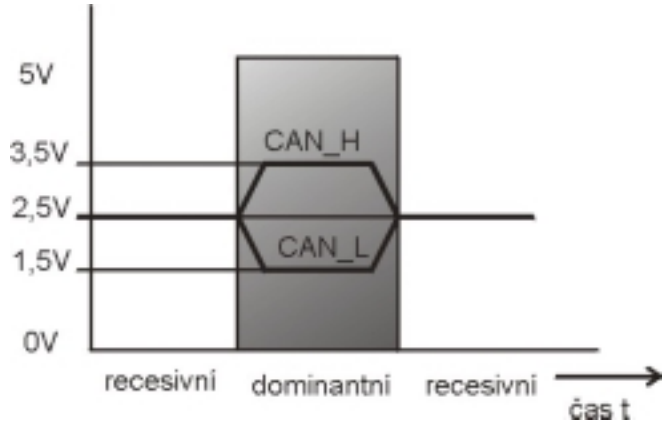
Sběrnice CAN (Controller Area Network) byla vyvinuta firmou Bosch a standardizována normou ISO 11898. Jedná se o sériový komunikační protokol, který má charakter broadcast sběrnice s prioritním rozhodováním o přístupu k médiu. Přenosová rychlost může být až 1Mbit/s, délka datového pole komunikačního rámce je od nuly do osmi bytů. Mezi hlavní výhody této sběrnice patří vysoká spolehlivost díky Hammingově vzdálenosti 6, krátká doba odezvy (nejkratší doba mezi dvěma zprávami je 47 μs) a rozsáhlá podpora ze strany výrobců.

3.2. POPIS ČINNOSTI ŘADIČE

V tomto odstavci je popsáno chování CAN řadiče při přijímání a vysílání. Je zde popsána struktura jednotlivých rámců a jejich použití, časování a s ním spojená synchronizace, mechanizmus vkládání bitů, chybové stavy, jejich ošetření a princip arbitráže.

3.2.1.Napěťové úrovně sběrnice CAN

Sběrnice CAN používá pro odlišení logických úrovní bitů dvou napěťových úrovní nazvaných dominantní a recesivní, viz obr. 6.

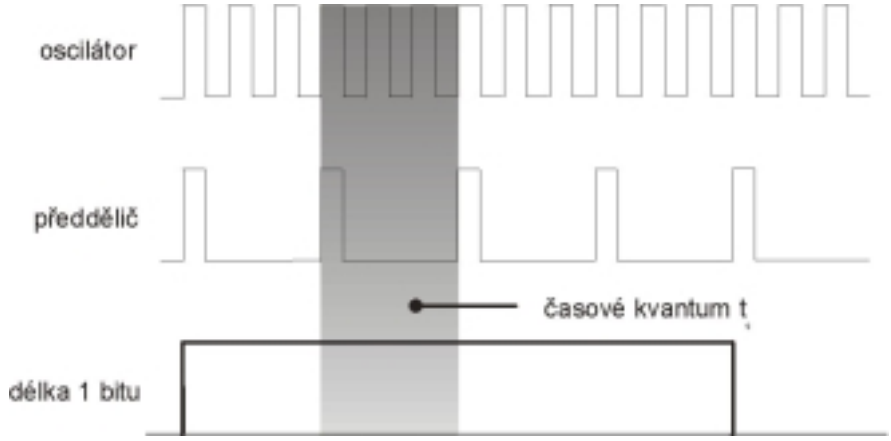


obr. 6 - Napěťové úrovně sběrnice CAN

Recesivní úroveň je definována nulovým rozdílem potenciálů mezi oběma vodiči CAN sběrnice, dominantní úroveň je definována napětím vodiče CAN_H 3,5 V proti zemnímu vodiči a napětím vodiče CAN_L 1,5 V proti zemnímu vodiči. Využití rozdělení úrovní na dominantní a recesivní je popsáno v odstavci 3.2.4.

3.2.2. Časování CAN řadiče

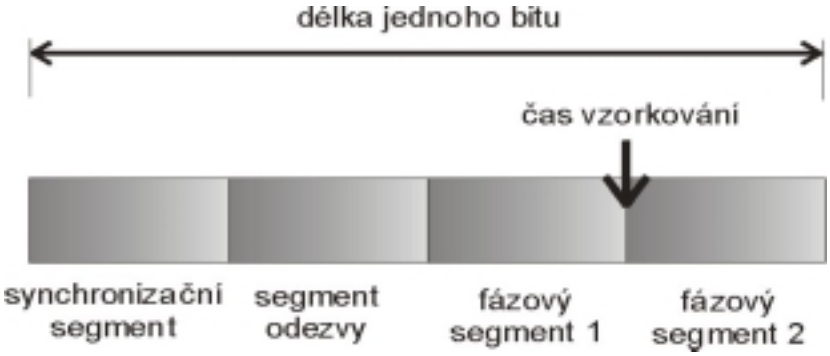
Obvod používá pro svou činnost hodinový signál, který zároveň slouží pro synchronizaci všech sekvenčních obvodů implementovaných v hradlovém poli. Tento signál je vydělen ve vstupním předděliči a výstupem je signál s periodou t_q, udávající délku jednoho časového kvanta (viz obr. 7). Perioda t_q je výchozí pro zadání baudové rychlosti, kterou bude řadič komunikovat. Dělící poměr předděliče je zadáván programově, a to prostřednictvím registru časování 0 – CAN adresa 6 (viz Tabulka 7).



obr. 7 - Časování CAN řadiče

Norma pro CAN sběrnici, viz [6] definuje délku jednoho bitu skládající se ze čtyř nepřekrývajících se segmentů, z nichž každý je složen z celočíselného počtu časových kvant t_q (viz obr. 8).

- Synchronizační segment (sync seg) je dlouhý $1t_q$ a slouží pro synchronizaci přijímačů s vysílačem na CAN sběrnici.
- Registr odezvy (prop. seg.) je programovatelný v délce 1 až 8 t_q a je použit pro kompenzaci časových zpoždění na sběrnici.
- Fázový segment 1 (phase seg. 1) je programovatelný v délce 1 až 8 t_q a je použit pro kompenzaci fázového rozdílu mezi přijímaným signálem a vnitřním časovačem. Segment může být prodloužen během resynchronizace.
- Fázový segment 2 (phase seg. 2) je roven maximální hodnotě z fázového segmentu 1 a dobou zpracování informace (information processing time). Taktéž je určen ke kompenzaci fázového rozdílu a může být zkrácen během resynchronizace.
- Doba zpracování informace je kratší nebo rovna dvěma časovým kvantům t_q.
- Celkový počet časových kvant t_q musí být v rozsahu 8 až 25.



obr. 8 - Časování jednoho bitu

Programovatelný čas vzorkování sběrnice umožňuje optimalizovat řadič pro konkrétní použití, pozdější vzorkování je výhodné pro rozsáhlé sběrnice s delší dobou odezvy, dřívější vzorkování dovoluje vzorkování signálu s pomalejšími vzestupnými a sestupnými hranami.

Norma umožňuje sloučení registru odezvy a fázového registru 1 do jediného registru, jak to bylo provedeno i v tomto případě.

Pro programování časovacích segmentů řadiče slouží registr časování 1 (CAN adresa 7 - Tabulka 8). Zde jsou zadány hodnoty registrů Tseg1, který vznikl sloučením registru odezvy a fázového registru 1 a Tseg2 odpovídající fázovému registru 2. Podrobný popis registrů časování je uveden v odstavci 3.5.6.

V tabulce 1 jsou popsány normou doporučované rychlosti přenosu dat spolu s příslušnými délkami časových kvant t_q a časy vzorkování.

Tabulka 1: Přenosové rychlosti sběrnice CAN

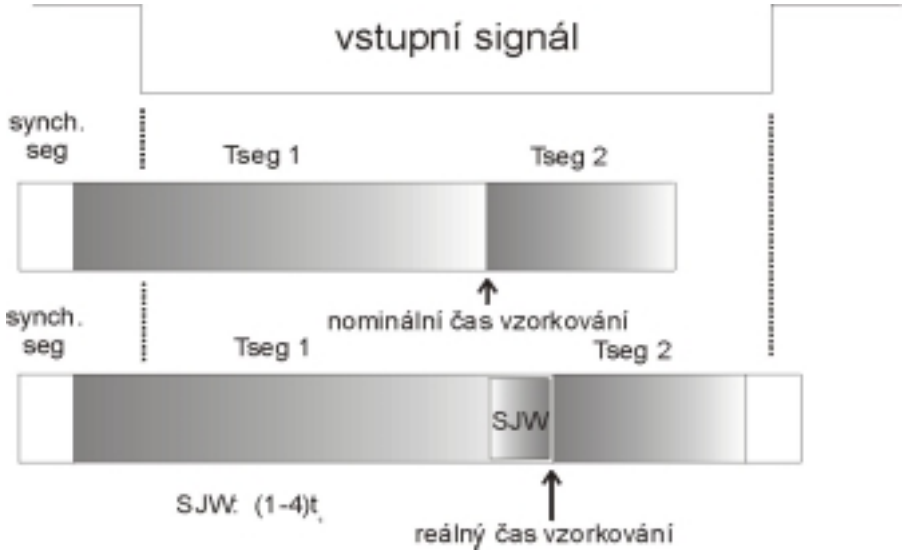
Přenosová rychlost	**Nominální doba jednoho bitu t_b**	**Počet časových kvant t_q v jednom bitu**	**Délka časového kvanta t_q**	**Poloha času vzorkování**
1 Mbit/s	1 μs	8	125 ns	6 t_q (750 ns)
800 kbit/s	1,25 μs	10	125 ns	8 t_q (1 μs)
500 kbit/s	2 μs	16	125 ns	14 t_q (1,75 μs)
250 kbit/s	4 μs	16	250 ns	14 t_q (3,5 μs)
125 kbit/s	8 μs	16	500 ns	14 t_q (7 μs)
50 kbit/s	20 μs	16	1,25 μs	14 t_q (17,5 μs)
20 kbit/s	50 μs	16	3,125 μs	14 t_q (43,75 μs)
10 kbit/s	100 μs	16	6,25 μs	14 t_q (87,5 μs)

3.2.3.Resynchronizace

S časováním úzce souvisí proces resynchronizace, který umožňuje synchronizaci přijímačů na CAN sběrnici s vysílaným signálem při každé změně úrovně příchozího signálu.

Přijímač je synchronizován s vysílačem při začátku vysílání zprávy, nejdelší možná kompletní zpráva může být až 131 bitů dlouhá, přičemž pro správnou činnost řadiče je třeba zajistit fázový rozdíl mezi interním oscilátorem a přijímaným signálem menší než jedno časové kvantum t_q v každém bitu zprávy, což znamená přesnost oscilátoru lepší než $\Delta \leq \frac{1}{16*131} = 0.477$ ‰. Při předpokládaném použití CAN řadičů obsahujících místo krystalového oscilátoru jeho levnější variantu používající syntézu, je v normě CAN sběrnice[6] definován proces resynchronizace a v popisovaném řadiči je implementován následujícím způsobem.

Příchod hrany vstupního signálu se vždy očekává v průběhu synchronizačního segmentu. Pokud hrana přijde déle, znamená to, že oscilátor vysílače pracuje s větší periodou, než oscilátor přijímače. Fázový rozdíl je kompenzován prodloužením segmentu Tseg1 (viz obr. 9). Doba, o kterou je segment prodloužen, je shora omezena proměnnou SJW, která je zadána programově při konfiguraci řadiče a udává celočíselný násobek časového kvanta t_q.



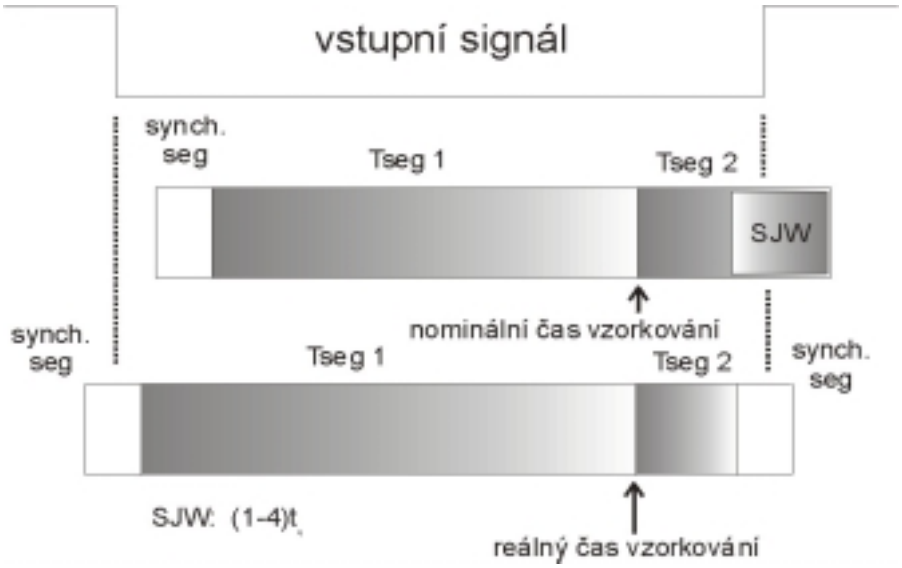
obr. 9 - Resynchronizace 1

Přijde-li hrana vstupního signálu dříve, než v synchronizačním segmentu, indikuje tím situaci, kdy oscilátor vysílače pracuje s menší periodou, než oscilátor přijímače. Fázový rozdíl je odstraněn zkrácením segmentu Tseg2(viz obr. 10). Doba, o kterou se segment zkracuje je celočíselným násobkem časového kvanta t_q a je v rozsahu 1 až SJW.

Celý proces resynchronizace by nebyl účinný, pokud by se přenášela zpráva obsahující dlouhé úseky shodné logické úrovně. Tomu je však zamezeno jednak konstrukcí rámce, který obsahuje úseky s pevně danou logickou úrovní, tak i mechanizmem vkládání bitů (bit stuffing), který zajišťuje, že pokud je ve zprávě pět po sobě jdoucích bitů se shodnou logickou úrovní, tak další odvysílaný bit bude mít úroveň opačnou, více viz odstavec 3.2.4. Je tedy zaručeno, že přijímač se bude synchronizovat maximálně každých pět bitů, z toho plyne požadavek na přesnost hodinového kmitočtu:

$$\Delta \leq \frac{1}{16*5} = 1.25\% .$$

Tato chyba je maximální možná a ve skutečnosti je lepší použít přesnější oscilátor, je to ale zároveň přesnost běžnými prostředky lehce dosažitelná, např. umožňuje použití krystalového rezonátoru místo krystalu pro rychlosti menší než 125kBd.



obr. 10 - Resynchronizace 2

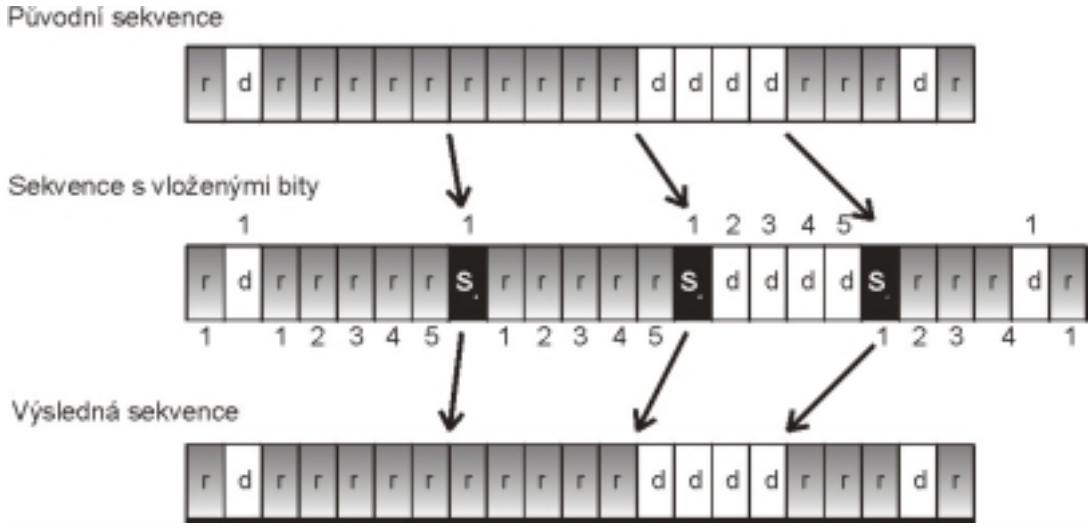
3.2.4. Vkládání bitů

CAN sběrnice používá pro reprezentaci bitů NRZ (Non Return to Zero - obr. 11) kódování, kdy hodnota bitu je reprezentována úrovní na sběrnici, není proto zaručeno, že se v každém bitu nachází náběžná a sestupná hrana.

obr. 11 - Kódování bitů

Obě logické úrovně bitů nejsou z definice rovnocenné, jsou-li všechny vysílače připojené ke sběrnici v log. 1, sběrnice bude také v log. 1, tato úroveň se nazývá recesivní (recessive bit level). Je-li však mezi vysílači alespoň jeden, který má na svém výstupu logickou nulu, také sběrnice bude v log. nule. Tato úroveň se nazývá dominantní (dominant bit level). Celá sběrnice má tedy charakter logické funkce AND. Zapojení více budičů na sběrnici je definováno v [6] blokové schéma je uvedeno např. v [10]

Při použitém způsobu kódování bitových úrovní je nebezpečí ztráty synchronizace mezi přijímačem a vysílačem při přenosu delších sekvencí stejné úrovně. Sběrnice CAN používá mechanismus vkládání bitů (bit stuffing) pro odstranění tohoto rizika. Obsahuje-li zpráva, která se má přenášet, sekvenci pěti nebo více po sobě jdoucích bitů stejné úrovně, je za tuto pětici vložen bit úrovně opačné, sloužící pro synchronizaci. Bit je vložen nezávisle na hodnotě bitu následujícího a je započítáván do počtu bitů rozhodujících pro další vložený bit, viz obr. 12.



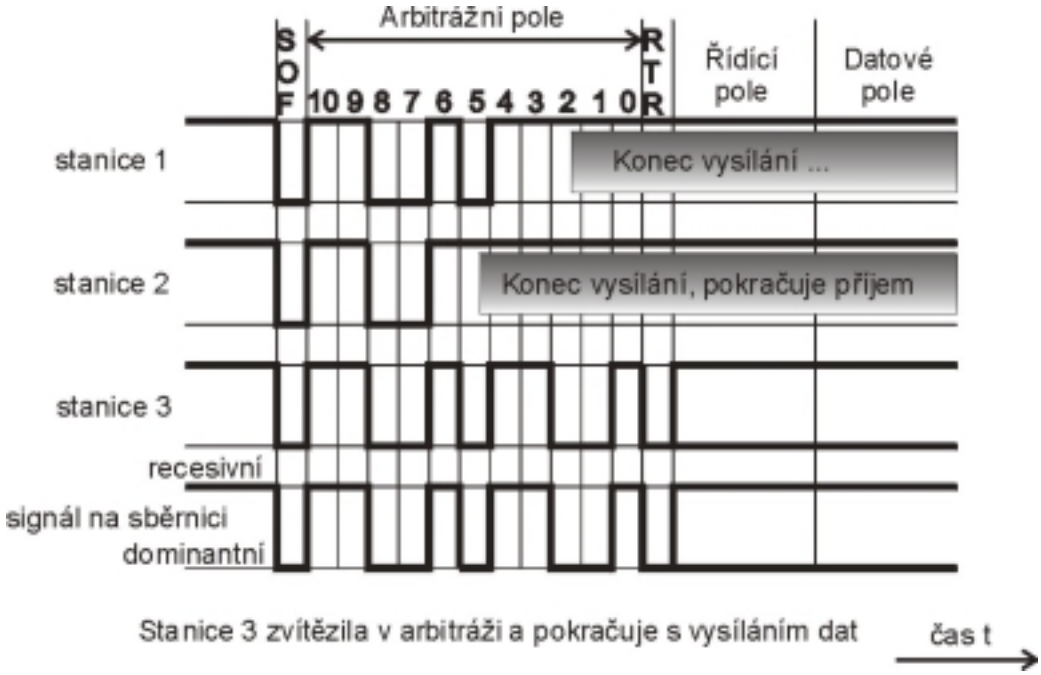
obr. 12 - Vkládání bitů

Oblast, kde se vkládají bity do zprávy, zahrnuje začátek zprávy, arbitrážní, řídicí a datové pole a kontrolní součet. Další z využití mechanismu vkládání bitů je indikace a globalizace chyb, detekuje-li jedna za stanic chybu, oznámí ji ostatním stanicím pomocí vyslání sekvence šesti dominantních bitů, které přepíší vysílanou zprávu, dojde k porušení pravidla vkládání bitů a zpráva bude ve všech přijímačích stornována a posléze opakována. Viz odstavec 3.2.9.

3.2.5.Řízení přístupu na sběrnici – Arbitráž

Protokol sběrnice CAN umožňuje současný přístup na sběrnici různým stanicím (node). Pokud na sběrnici přistupují současně dvě nebo více stanic, je zapotřebí arbitráž. Metoda použitá u CAN sběrnice je nedestruktivní a nazývá se nedestruktivní bitové rozhodování podle priority (Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority CSMA/CD + AMP). Priorita zprávy je zakódována v identifikátoru zprávy, čím nižší číslo představuje, tím větší má zpráva prioritu.

Je-li sběrnice ve stavu čekání, kterákoli ze stanic může zahájit vysílání. Každá z nich poté čte zpět ze sběrnice logickou úroveň bitu a porovnává ji s úrovní, kterou sama vyslala. Sběrnice má charakter logického součinu, tzn. dominantní úroveň přepíše úroveň recesivní a stanice, která recesivní úroveň vyslala přestává v tomto okamžiku vysílat.

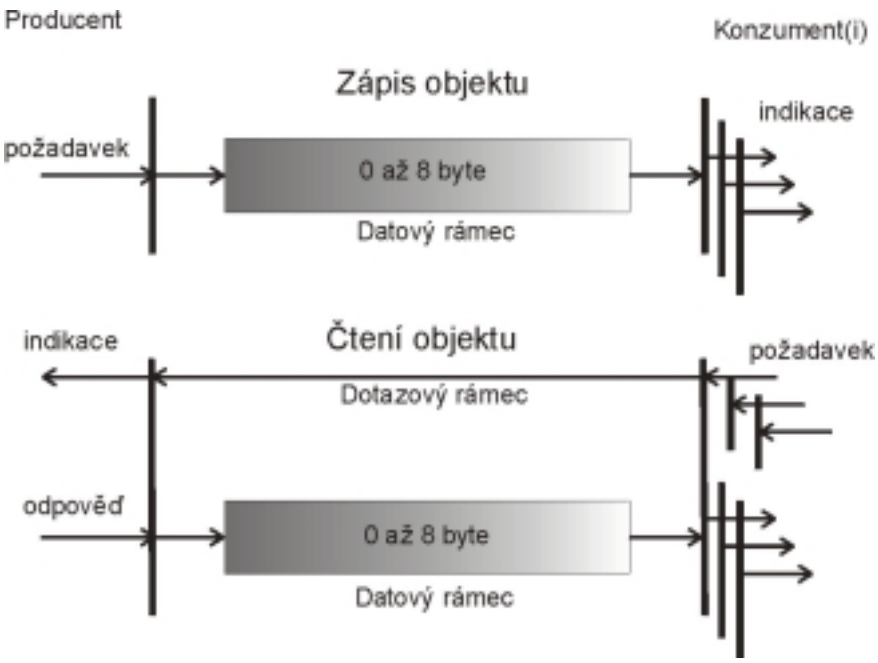


obr. 13 - Princip arbitráže

Začnou-li vysílat dvě nebo více stanic ve stejný okamžik, všechny vysílají data a zároveň je porovnávají se signálem ze sběrnice. Pokud se tato data rovnají, kolize nevzniká a stanice pokračují v činnosti. V případě, že stanice vyslala recesivní bit, ale přijala bit dominantní (stanice 2 na obr. 13), začne namísto zprávy vysílat pouze recesivní úroveň a po zbytek zprávy se chová jako přijímač. Stanice 2 v tomto případě prohrála arbitráž kvůli vyšší binární hodnotě identifikátoru než stanice 3. Tímto postupem je zaručeno, že při současném začátku vysílání vyhrává arbitráž zpráva s nejvyšší prioritou (zde stanice 3) , stanice, které arbitráž prohrály (zde 1 a 2) odvysílají své zprávy ihned po dokončení zprávy právě vysílané.

3.2.6.Komunikační služby

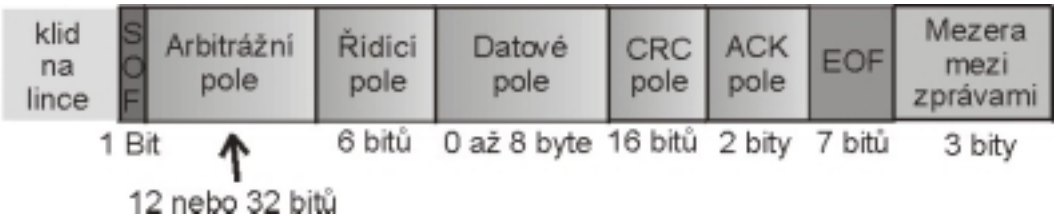
Protokol sběrnice CAN poskytuje dva druhy komunikačních služeb. Služba zápisu Objektu (Write Object Service) vysílá datový rámec (Data Frame) od jedné stanice (producenta) k druhé (konzument). Tato služba předpokládá, že existuje stanice, která očekává tuto zprávu. Druhou komunikační službou je požadavek na konkrétní zprávu. Tato služba čtení objektu (Read Object Service) je iniciována jedním nebo více konzumenty. Rámec vyslaný těmito stanicemi se nazývá dotazový rámec (Remote Frame). Stanice, která vlastní požadované informace, vyšle jako odpověď odpovídající datový rámec. Použití těchto služeb je demonstrováno na obr. 14.



obr. 14 - Komunikační služby

3.2.7.Datový rámec

Datový rámec (viz obr. 15) je vyslán CAN stanicí, pokud ta chce vyslat data nebo pokud jsou data požadována jinou stanicí. V jednom rámci lze vyslat maximálně 8 bytů dat. Rámec začíná polem SOF (Start Of Frame – začátek zprávy). Toto pole je dlouhé jeden bit, který je vždy vysílán jako dominantní a slouží k synchronizaci všech přijímačů s vysílačem při začátku vysílání. Po začátku následuje Arbitrážní pole (Arbitration field), ve kterém je zakódován druh zprávy a její priorita. Následuje řídicí pole (Control field), které specifikuje hlavně délku zprávy, další v řadě je CRC pole (Cyklic Redundancy Check – cyklická redundantní kontrola), sloužící k detekci případných přenosových chyb. V ACK poli (Acknowledgement – Potvrzení) vysílá vysílač recesivní úroveň a jakákoli stanice, která přijme zprávu bezchybně potvrdí příjem vysláním dominantní úrovně. Recesivní bity v poli EOF (End Of Frame – konec rámce) zakončují rámec. Mezi dvěmi zprávami musí být mezera dlouhá tři bity.



obr. 15 - Datový rámec

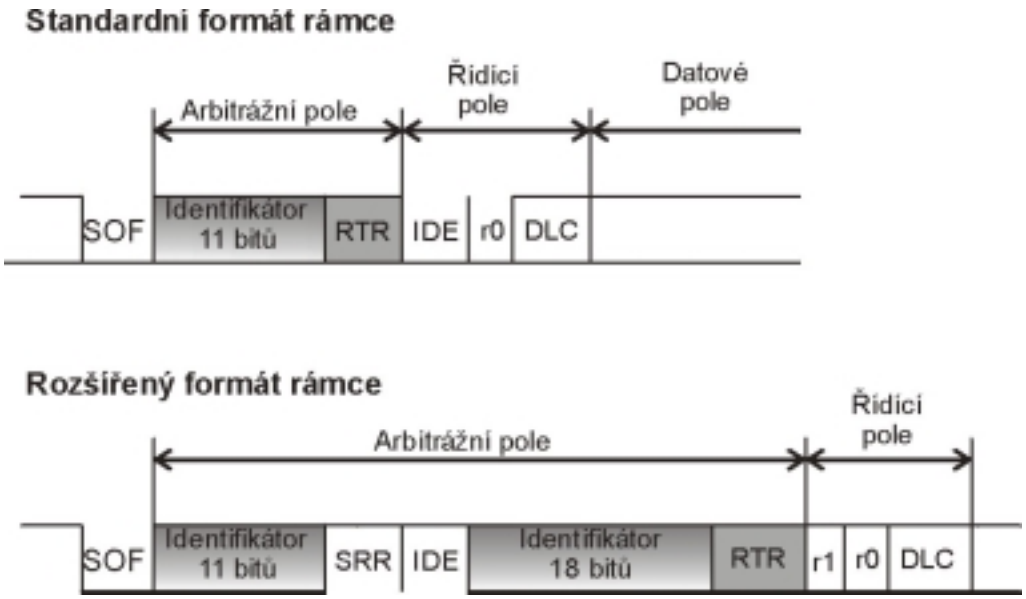
3.2.7.1.ZAČÁTEK RÁMCE – SOF

Začátek rámce (Start Of Frame – SOF) je dlouhý 1 bit a je vždy vysílán jako dominantní. Toto pole slouží k synchronizaci stanic přijímajících signál se stanicí která jej vysílá. Synchronizace nastává při přechodu signálu na sběrnici z recesivní do dominantní úrovně při klidu na lince (Bus idle) nebo na konci mezery mezi zprávami (Intermission). V průběhu příjmu zprávy je aktivní proces resynchronizace blíže popsaný v odstavci 3.2.3.

3.2.7.2. ARBITRÁŽNÍ POLE

Délka identifikátoru ve standardním formátu rámce je 11 bitů a odpovídá základnímu identifikátoru v rozšířeném formátu. Identifikátor je následován RTR bitem. RTR od sebe odděluje datové rámce od dotazových (viz. 3.2.6) a zde musí být vysílán dominantní. V případě dotazového rámce musí být RTR bit recesivní. Základní identifikátor je následován bitem IDE (Identifier Extension – rozšíření identifikátoru), který je vysílán dominantní ve standardním formátu a recesivní v rozšířeném formátu identifikátoru. Standardní formát tedy zvítězí v případě kolize, pokud jsou současně vysílány standardní a rozšířený formát s jinak stejným identifikátorem.

Rozšířený formát se skládá ze dvou částí – základního identifikátoru o délce 11 bitů a rozšířeného identifikátoru o délce 18 bitů. Bit SRR(Substitute Remote Request – náhrada za dotazový rámec) je substitucí za bit RTR v případě standardního formátu a je vysílán recesivní. Pokud je tento bit změněn a přijímač ho přijme jako dominantní, nebude to považovat za chybovou událost, ale tato hodnota bude rozhodující pro vkládání bitů a arbitráž. Chyba to nemůže být z toho důvodu, že tento bit je vysílán dříve než IDE bit a proto přijímač neví, zda přijímá standardní nebo rozšířený formát zprávy, o tom rozhoduje pouze hodnota bitu IDE. Arbitrážní pole je zobrazeno na obr. 16.



obr. 16 - Arbitrážní pole

3.2.7.3. ŘÍDICÍ POLE

Složení řídicího pole (Control Field - obr. 17) je podobné pro standardní i rozšířený formát zprávy. Ve standardním formátu zahrnuje délku datového pole (Data Length Code - DLC), IDE bit vysílaný jako dominantní a rezervovaný bit r0 také vysílaný dominantní. Řídicí pole rozšířeného formátu obsahuje stejné DLC pole a dva rezervované bity r1 a r0. Rezervované bity jsou vysílány dominantní, avšak přijímač akceptuje obě úrovně v jakýchkoli kombinacích

obr. 17 - Řídicí pole

Počet bytů, který je přenášen v datovém poli, je indikován v řídicím poli pomocí DLC kódu, který je 4 bity dlouhý. DLC kód v rozsahu 0-7 odpovídá přímo délce dat, všechny ostatní kombinace odpovídají délce datového pole 8 bytů.

Tabulka 2: Počet datových bytů v datovém poli

Počet datových bytů	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d/r	d/r	d/r

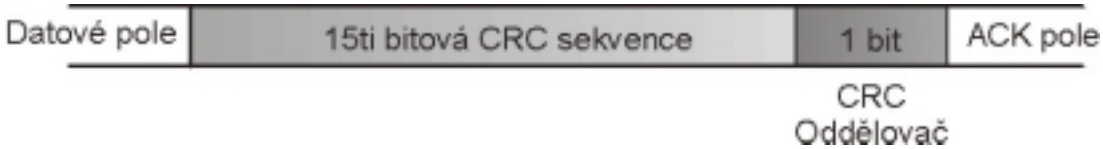
3.2.7.4. DATOVÉ POLE

Datové pole, viz obr. 18, má volitelnou délku od 0 do osmi bytů, skutečný počet bytů přenášený ve zprávě je určen hodnotou DLC řídicího pole. V některých případech je požadavek vyslat rámec neobsahující žádná data, například při nutnosti indikovat událost, kterou lze popsat vhodným identifikátorem.

obr. 18 - Datové pole

3.2.7.5. CRC POLE

K zabezpečení celé zprávy slouží CRC pole (Cyklic Redundancy Check – Cyklická redundantní kontrola, viz obr. 19). Pole se skládá z patnáctibitové CRC sekvence a jednobitového oddělovače, který je vysílán vždy jako recesivní. CRC kód je zvolen tak, aby co nejlépe vyhovoval pro zprávy kratší než 127 bitů, a zabezpečuje zprávy s Hammingovou vzdáleností 6. To znamená, že pět bitových chyb libovolně rozmístěných ve zprávě lze tímto způsobem detekovat. Dále je možné detekovat shlukové chyby až do délky 15 bitů.



obr. 19 - CRC pole

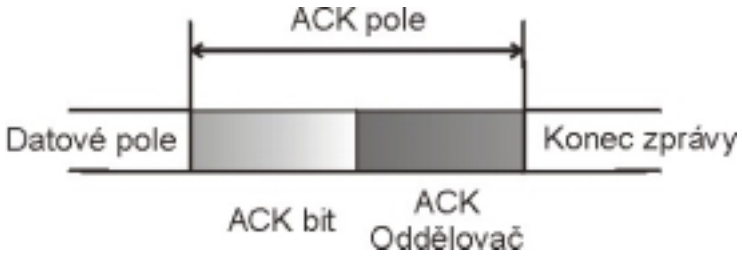
Vysílač i přijímač počítají CRC sekvenci obdobným způsobem podle následujícího postupu:

1. Zpráva, tj. všechny pole předešlé CRC poli, je považována za polynom, který je vydělen generujícím polynomem: $x^{15} + x^{14} + x^{10} + x^{8} + x^{7} + x^{4} + x^{3} + 1$.
2. Zbytek po dělení je CRC sekvence, která je vysílána spolu se zprávou v CRC poli.
3. Přijímač dělí celou zprávu včetně CRC sekvence generujícím polynomem.

CRC kód má tu vlastnost, že pokud je proveden výpočet CRC sekvence celé zprávy plus CRC pole, tak na konci CRC pole je výpočetní registr vynulován. Tato vlastnost usnadňuje implementaci řadiče – není třeba kontrolovat celou sekvenci bit po bitu, stačí pouze ověřit nulovost všech bitů výpočetního registru na konci CRC pole. Není-li registr nulový, je indikována chyba kontroly a přijímač vyšle chybový rámec jako požadavek na opětovné odvysílání zprávy.

3.2.7.6.ACK POLE

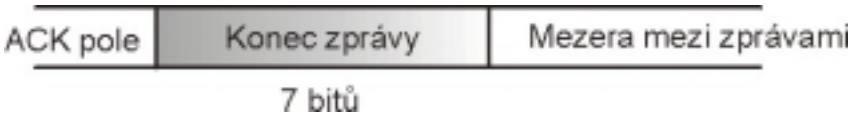
Potvrzovací pole (Acknowledge field – ACK viz. obr. 20) je dlouhé dva bity a tvoří jej ACK bit a ACK oddělovač. Vysílač zprávy vysílá oba dva bity jako recesivní. Přijímač, který přijal celou zprávu bez chyb, potvrdí příjem vysláním dominantní úrovně na sběrnici v době ACK bitu. Detekuje-li vysílač v době ACK slotu dominantní úroveň, dozví se tímto způsobem, že alespoň jeden přijímač přijal zprávu korektně.



obr. 20 - ACK pole

3.2.7.7.KONEC ZPRÁVY

Datový i dotazový rámec jsou zakončeny sekvencí sedmi po sobě jdoucích recesivních bitů. V případě vzniku chyby CRC kontroly je odvysílán chybový rámec (viz odstavec 3.2.9.3), pro jeho případnou detekci slouží právě konec zprávy, viz. obr. 21.



obr. 21 - Konec zprávy

3.2.8.Dotazový rámec

Cílová stanice může vyslat požadavek na data ze zdrojové stanice prostřednictvím dotazového rámce viz. obr. 22 s identifikátorem, který souhlasí s identifikátorem požadovaného datového rámce. Příslušná zdrojová stanice vyšle datový rámec jako odpověď.

obr. 22 - Dotazový rámec

Mezi datovým a dotazovým rámcem jsou dva rozdíly, za prvé bit RTR je vysílán dominantní v datovém rámci a recesivní v dotazovém a za druhé dotazový rámec neobsahuje datové pole. V případě, že začnou být vysílány dotazový i datový rámec se stejným identifikátorem současně, datový rámec vyhraje arbitráž kvůli dominantnímu RTR bitu a stanice, která vysílala dotazový rámec dostane data okamžitě.

3.2.9. Ošetření chyb

Sběrnice CAN obsahuje na fyzické úrovni mnoho prostředků pro zajištění bezchybného přenosu signálu po sběrnici, jako jsou například definice napěťových úrovní, resynchronizace, doporučená přenosová média a jejich délky. Přesto může dojít buď nedodržením těchto pravidel, nebo vznikem dodatečného rušení, k poškození přenášené zprávy. Tato zpráva, pokud by byla akceptovaná, by vedla k nekonzistenci dat mezi vysílačem a přijímačem a k chybné reakci přijímače na povel od vysílající stanice. Aby se těmto situacím zabránilo, musí přijímače i vysílače reagovat na vznik chybové zprávy tím, že bude stornována a při nejbližší možné příležitosti opět opakována. Tímto způsobem je zajištěno, že chybová zpráva nebude přijímačem akceptována. Vyvstává zde však další riziko, že pokud by došlo k poškození jedné ze stanic, která by nebyla schopna svou vinou přijmout korektní zprávu, tak by trvale blokovala provoz na sběrnici stornováním přijatých zpráv. Jako řešení je použit mechanizmus povolení přístupu na sběrnici, kdy se podle čítače chybně odeslaných a přijatých zpráv uděluje stanici povolení k úplnému přístupu na sběrnici, případně pouze čtení nebo odpojení od sběrnice.

3.2.9.1. GLOBALIZACE LOKÁLNÍCH CHYB

Na obr. 23 je ilustrační příklad, kdy přijímač číslo 2 detekuje lokální chybu a odvysílá chybový rámec. Při šestém bitu chybového rámce všechny ostatní stanice na sběrnici detekují porušení pravidla vkládání bitů a začnou všechny vysílat chybový rámec. Po oddělovači chybového rámce s délkou 8 bitů a mezeře mezi zprávami s délkou 3 bity se vysílač znovu pokusí odvysílat poškozenou zprávu.

obr. 23 - Globalizace lokálních chyb

3.2.9.2. OŠETŘENÍ CHYB

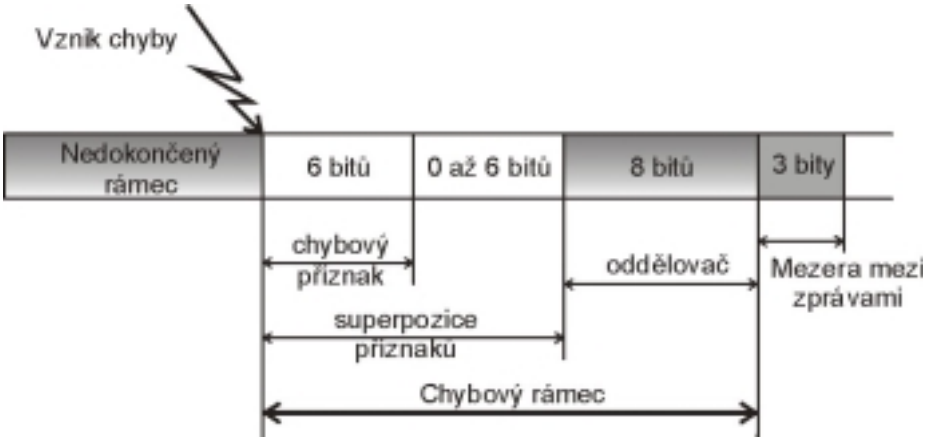
Při vzniku chyby na sběrnici CAN jsou prováděny následující kroky:

1. Byla detekována lokální nebo globální chyba
2. Je odvysílán chybový rámec, který dá chybu na vědomí všem ostatním stanicím na sběrnici.
3. V případě lokální chyby chybový rámec vyprodukuje další, překrývající se chybový rámec následovaný oddělovačem chybového rámce s délkou 8 bitů.
4. Zpráva je zlikvidována každou stanicí
5. Čítače chybových zpráv každé stanice zvýší svou hodnotu o jedna.
6. Vysílání zprávy bude automaticky zopakováno.

Pomocí mechanizmu chyb je garantována konzistence dat na sběrnici, pokud žádná ze stanic není v pasivním chybovém stavu nebo odpojená od sběrnice. V případě lokální chyby je tato globalizována vysláním chybového rámce obsahujícího šest bitů stejné úrovně, který způsobí porušení pravidla vkládání bitů. Po skončení chybového rámce vysílá stanice znovu zrušenou zprávu. Pokud žádná zpráva s vyšší prioritou nepřevezme řízení na sběrnici, bude zrušená zpráva odvysílána nejdříve za dobu 23 bitů.

3.2.9.3. AKTIVNÍ CHYBOVÝ RÁMEC

Aktivní chybový rámec (Active Error Frame viz. obr. 24) je generován, pokud stanice detekuje chybu v přijímané zprávě. Rámec se skládá z chybového příznaku a chybového oddělovače. Dominantní bity chybového příznaku přepíší poškozený rámec a zapříčiní jeho opětovné odvysílání. Protože je možné, že chyba byla lokální, obsahuje chybový rámec recesivní oddělovač o délce 8 bitů, po jehož skončení a uplynutí mezery mezi zprávami o délce tří bitů, může vysílač znovu zahájit vysílání.



obr. 24 - Aktivní chybový rámec

3.2.9.4. CHYBA VKLÁDÁNÍ BITŮ

Pole datového i dotazového rámce, ve kterých jsou bity vkládány, jsou zobrazeny na obr. 25. Ostatní pole stejně jako chybový rámec mají pevný formát a nejsou kódovány vkládáním bitů.

Pokud stanice detekuje šest po sobě jdoucích bitů stejné úrovně mezi začátkem zprávy a CRC polem, znamená to, že došlo k porušení pravidel vkládání bitů a jako reakci stanice vygeneruje chybový rámec. Mechanizmus vkládání bitů je podrobně popsán v odstavci 3.2.4.

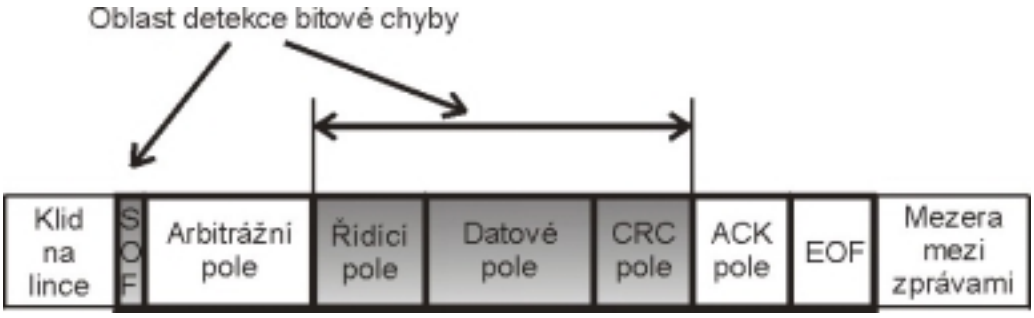
obr. 25 - Oblast vkládání bitů

3.2.9.5. BITOVÁ CHYBA

Oblasti, ve kterých se výskyt bitové chyby kontroluje, jsou zobrazeny na obr. 26.

Vysílače i přijímače připojené k CAN sběrnici provádějí kontrolu bitové chyby (Bit Error). Bitová chyba nastane v případě, že vysílač vyšle dominantní úroveň bitu a detekuje recesivní nebo vyšle recesivní úroveň a detekuje dominantní. Jako reakce na chybový stav

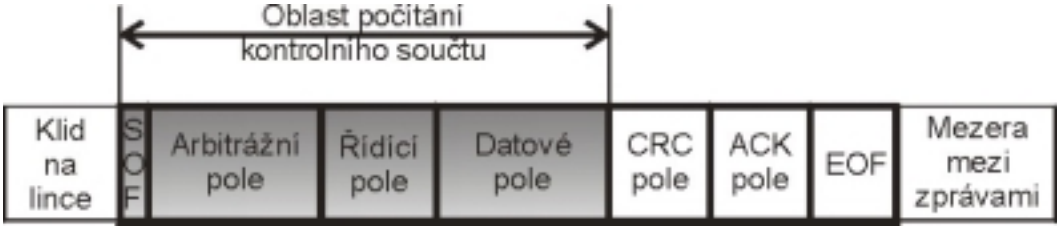
bude vyslán chybový rámec iniciující opakování zprávy, ve které byla detekována chyba. Je-li v průběhu arbitrážního a potvrzovacího pole detekován dominantní bit místo recesivního, není to považováno za chybu, z důvodu funkčnosti funkcí arbitráže a potvrzení popsaných v odstavcích 3.2.5 a 3.2.7.6.



obr. 26 - Oblast detekce bitové chyby

3.2.9.6. CHYBA KONTROLNÍHO SOUČTU

Kontrolní součet je počítán vysílačem od začátku zprávy do konce datového pole (viz obr. 27), po té je odvysílán jako patnáctibitová sekvence v CRC poli. Přijímač počítá CRC kód ve stejných polích jako vysílač a v CRC poli kontroluje přijatou sekvenci s vypočtenou. Pokud se sekvence od sebe liší, je následně vygenerován chybový rámec. Kontrolní součet je zde použit pouze pro detekci chyb a ne pro jejich opravu.



obr. 27 - Kontrolní součet

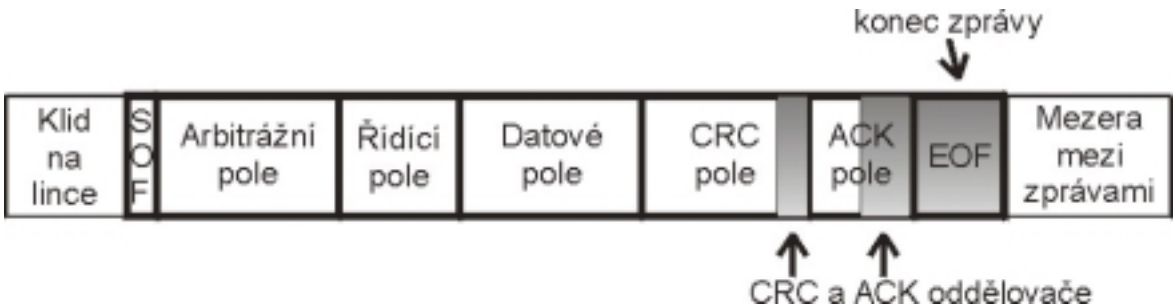
3.2.9.7. CHYBA POTVRZENÍ ZPRÁVY

V průběhu ACK bitu (viz obr. 28) vysílač vysílá recesivní úroveň, ale očekává, že bude detekovat úroveň dominantní, indikující, že alespoň jeden přijímač na sběrnici přijal bezchybně zprávu. Nestane-li se tak, nastala chyba potvrzení zprávy a vysílač při dalším bitu začne vysílat chybový rámec.

obr. 28 - Chyba potvrzení zprávy

3.2.9.8. CHYBA PEVNÝCH BITŮ RÁMCE

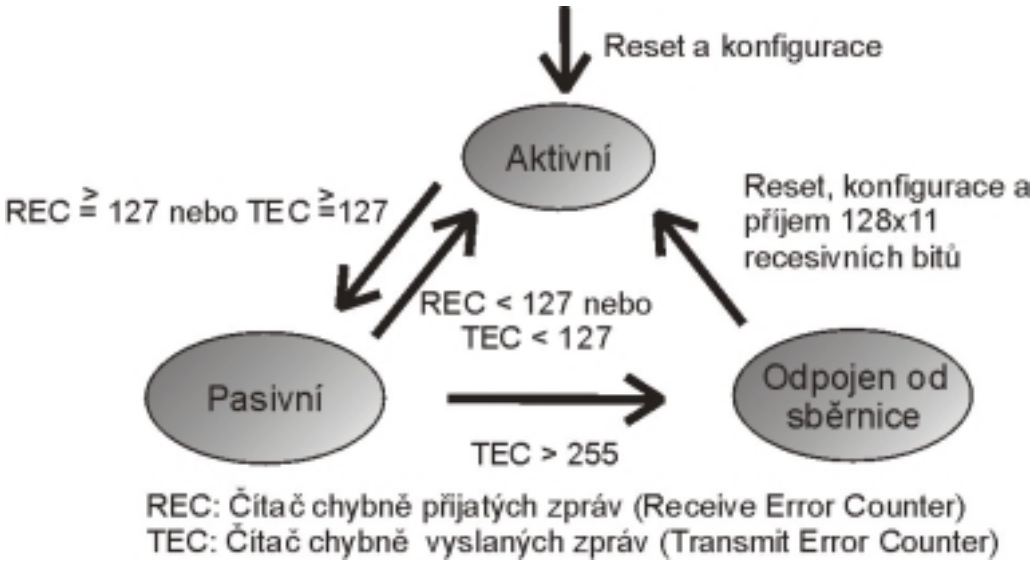
V datovém i dotazovém rámci jsou segmenty, uvnitř nichž mají bity vždy recesivní hodnotu, je-li však detekována hodnota dominantní, došlo k chybovému stavu, která bude ošetřena vysláním chybového rámce. Segmenty, ve kterých může být tato chyba detekována jsou: CRC oddělovač, ACK oddělovač a konec zprávy (viz obr. 29)



obr. 29 - Chyba pevných bitů rámce

3.2.9.9.CHYBOVÉ STAVY

Aby mohl řadič odlišit dočasné a trvalé poruchy, obsahuje dva čítače: Čítač chybně přijatých zpráv (Receive Error Counter – REC) a čítač chybně vyslaných zpráv (Transmit Error Counter – TEC). Čítače v případě chybové zprávy zvětšují svou hodnotu o jedna. Pokud je zpráva přijata/odeslána v pořádku, hodnota příslušného čítače je o jedna snížena. V závislosti na hodnotě čítačů mění řadič svůj stav. Počáteční stav je aktivní, ve kterém může vysílat aktivní chybové rámce. V případě, že počet chybných zpráv převažuje nad správnými a hodnota jednoho z čítačů je větší nebo rovna 127, přejde řadič do pasivního stavu, ve kterém smí vysílat pouze pasivní chybové rámce. V případě velkého výskytu chyb nebo poruše stanice přejde řadič do třetího stavu, kdy je odpojen od sběrnice. Z tohoto stavu je možný přechod do stavu aktivního za použití hardwarového nebo softwarového resetu. Byl-li reset softwarový, řadič bude čekat než přijme ze sběrnice 128x11 po sobě jdoucích recesivních bitů, než bude moci odeslat zprávu. Celá situace je zobrazena na obr. 30.

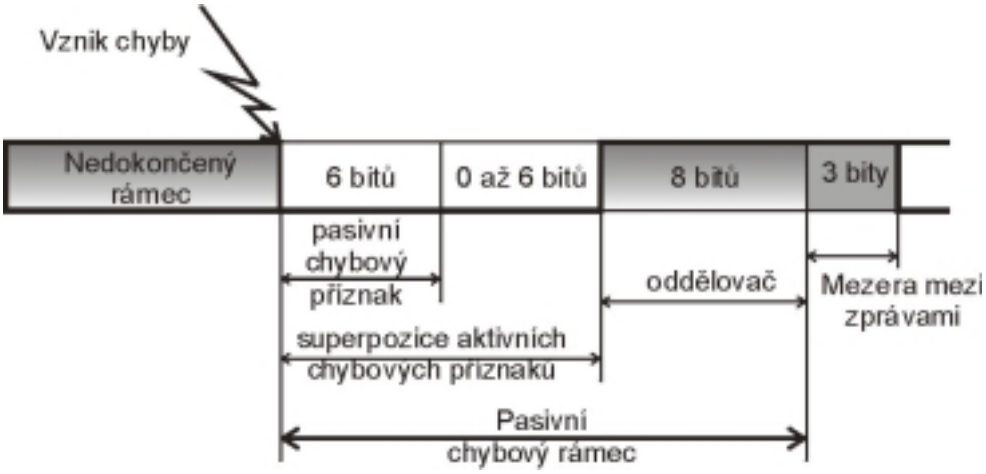


obr. 30 - Chybové stavy

3.2.9.10.PASIVNÍ CHYBOVÝ RÁMEC

Pasivní chybový rámec (Passive Error Frame viz. obr. 31) je vysílán řadičem jako reakce na chybovou událost, pokud je v pasivním stavu. Analogicky k aktivnímu chybovému rámci vysílanému v aktivním stavu. Tyto dva chybové rámce jsou podobné, odlišnost mezi nimi je ta, že chybový příznak je u aktivního rámce vysílán v dominantní úrovni, zatímco u pasivního rámce v recesivní úrovni. Účel pasivního chybového rámce je omezit zatížení sběrnice v případě, že ve stanici vznikla porucha a ta jako jediná není schopna přijímat zprávy a proto by častým vysílání aktivních chybových rámců omezovala komunikaci ostatních stanic na sběrnici.

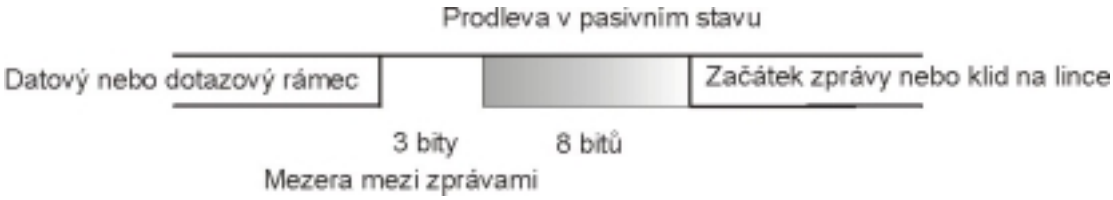
Pokud je řadič v pasivním stavu, nemůže přerušit zprávu vysílanou jinou stanicí kvůli recesivní hodnotě chybového příznaku, může však přerušit vysílání své vlastní zprávy.



obr. 31 - Pasivní chybový rámec

3.2.9.11. PRODLEVA PŘED VYSÍLÁNÍM V PASIVNÍM STAVU

Dalším prostředkem, jak omezit zatížení sběrnice při vzniku poruchy v jedné ze stanic, je zařazení prodlevy před vysíláním (Suspend Passive Transmission viz. obr. 32), pokud je řadič v pasivním stavu. V tom případě musí řadič čekat po každém konci zprávy mimo mezery mezi zprávami o délce 3 bity navíc po dobu prodlevy o délce 8 bitů. Tímto způsobem je umožněno odvysílat zprávu stanici s nižší prioritou, která by jinak prohrála arbitráž.



obr. 32 - Prodleva před vysíláním

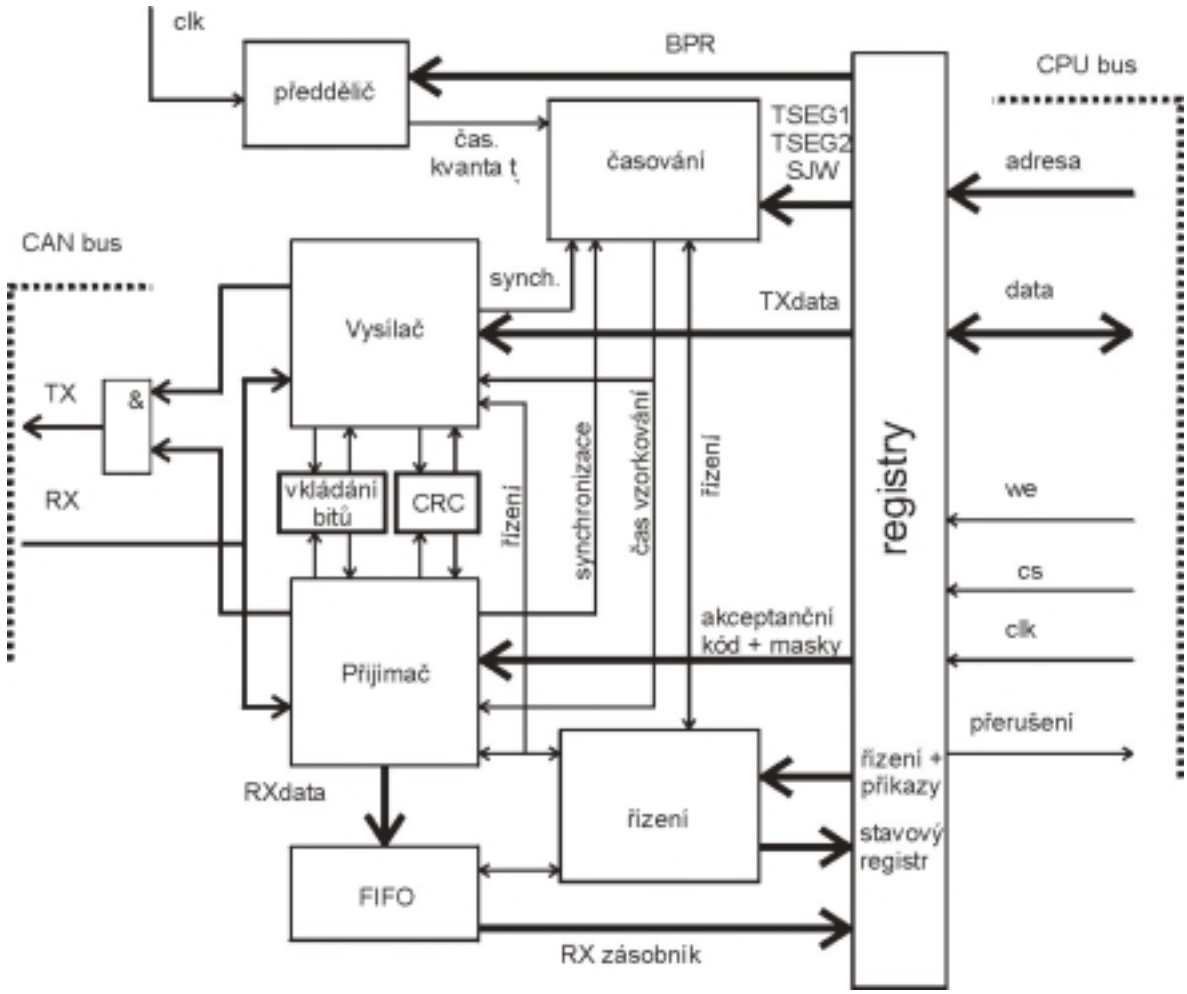
3.3. STRUKTURA ŘADIČE

V následujícím odstavci je popsána struktura řadiče v podobě modulů, které pracují paralelně a komunikují spolu. Tato filozofie návrhu je umožněna díky použití hradlového pole a programovacího jazyka VHDL, který programování paralelních obvodů podporuje.

3.3.1. Blokové schéma

Blokové schéma CAN řadiče je zobrazeno na obr. 33. S okolím řadič komunikuje pomocí tří rozhraní: První je globální vstup hodinového signálu, který je použit pro synchronizaci všech sekvenčních logických obvodů a dále vstupuje do předděliče, který jej vydělí hodnotou registru BPR, výsledný signál má periodu jednoho časového kvanta t_q. Druhé rozhraní je určeno k připojení na sběrnici CAN a obsahuje signál TX, který je výstupem řadiče na sběrnici a signál RX, pomocí kterého řadič snímá úroveň na sběrnici. Pomocí posledního rozhraní komunikuje řadič s nadřízeným mikroprocesorem. Komunikace probíhá po osmi vstupně/výstupních datových vodičích a osmi vstupních adresových vodičích, funkce ostatních řídicích signálů lze měnit podle konkrétní sběrnice, na kterou je řadič připojen, např. CPU 8051, sběrnice PC104.

Bloky zakreslené na schématu zhruba odpovídají jednotlivým programovým modulům, schéma je však značně zjednodušeno, kvůli přehlednosti byla vynechána většina řídicích a synchronizačních signálů, které byly symbolicky zahrnuty do signálů s názvem řízení.



obr. 33 - Blokové schéma

Rozhraní ze strany CPU je výstupem programového bloku registrů. Tento blok se stará o konverzi dat a řídicích příkazů mezi vnitřními bloky řadiče a nadřazeným mikroprocesorem. Mikroprocesor se v tomto případě nemusí zajímat celou strukturou CAN rámce, je pouze nucen nahrát data identifikátoru a datového pole do příslušných registrů a zapsat log. 1 do registru příkazu CMR.0. Tím řadiči oznámí požadavek na vyslání zprávy, který se již o zpracování zprávy postará a o výsledku upozorní nastavením bitů stavového registru. Podobná je situace i při příjmu, např. paměť FIFO není pro CPU přístupná, je viditelné vždy jen okno s poslední přijatou zprávou na místě přijímacího zásobníku a hodnota čítače přijatých zpráv. Význam a struktura a umístění jednotlivých registrů jsou popsány v odstavci 3.5.

O odvysílání zprávy se stará blok s příslušným názvem. Jeho vstupy jsou signály časování a data pro vysílanou zprávu. Blok se stará o konstrukci celého rámce zprávy, používá přitom blok CRC, který průběžně počítá kontrolní součet, a blok starající se o vkládání bitů do vysílané zprávy.

Přijímač neustále monitoruje dění na sběrnici a detekuje-li začátek zprávy, začne jí zpracovávat. Stará se při tom o kompletní dekódování zprávy, jeho výstupy jsou data z identifikátoru zprávy a datového pole, které umístí na následující pozici ve FIFO paměti, pokud není zaplněna.

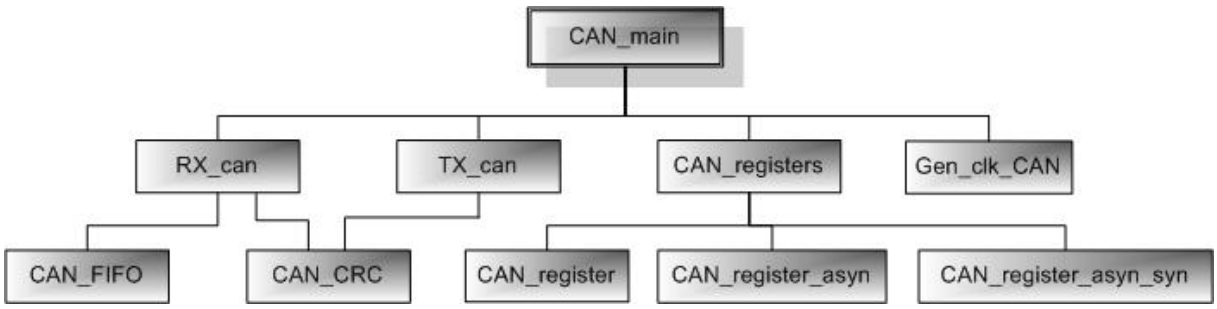
Blok časování se stará o tvorbu signálů, sloužících pro vzorkování vstupního signálu a signálů oddělujících od sebe jednotlivé bity zprávy, zajišťuje také synchronizaci se vstupním signálem při začátku příjmu a resynchronizaci v průběhu příjmu.

Blok řízení se stará o spolupráci jednotlivých modulů a rozděluje mezi přijímač a vysílač právo přístupu na sběrnici. Například pokud vysílač začne vysílat, přijímač začne současně s ním a v průběhu arbitrážního pole se podle výsledku arbitráže rozhodne, který z nich bude pokračovat.

Blok FIFO je realizován pamětí, do které vysílací blok zapisuje přijaté zprávy, a blok registrů umožňuje mikroprocesoru tyto zprávy číst. Logika spolu s čítačem zpráv v paměti obstarává činnost celého bloku.

3.3.2. Rozdělení zdrojového kódu do modulů

Zdrojový kód je rozdělen do modulů, které svojí funkcí zhruba odpovídají blokům popsaným v předešlém odstavci, hierarchie modulů je zobrazena na obr. 34.



obr. 34 - Programové moduly

Moduly RX_can, TX_can a Gen_clk_can jsou umístěny v jednom souboru can.vhd, ostatní moduly jsou umístěny v souborech se shodnými jmény. Modul CAN_main obstarává funkci bloku řízení a modul Gen_clk_CAN funkci předděliče. V modulu CAN_registers jsou vytvořeny registry řadiče, pro jednotlivé registry jsou použity moduly CAN_register, CAN_register_asyn a CAN_register_syn lišící se způsobem nulování registru. Vzorový zkrácený zdrojový kód programového modulu CAN_FIFO je uveden v odstavci 3.3.6, kde je popsána jeho činnost.

3.3.3. Blok Řízení

Skládá se ze tří hlavních stavů: *čekání* (idle), *vysílání*(transmitting), *přijímání*(receiving) a je schematicky zobrazen na obr. 35.

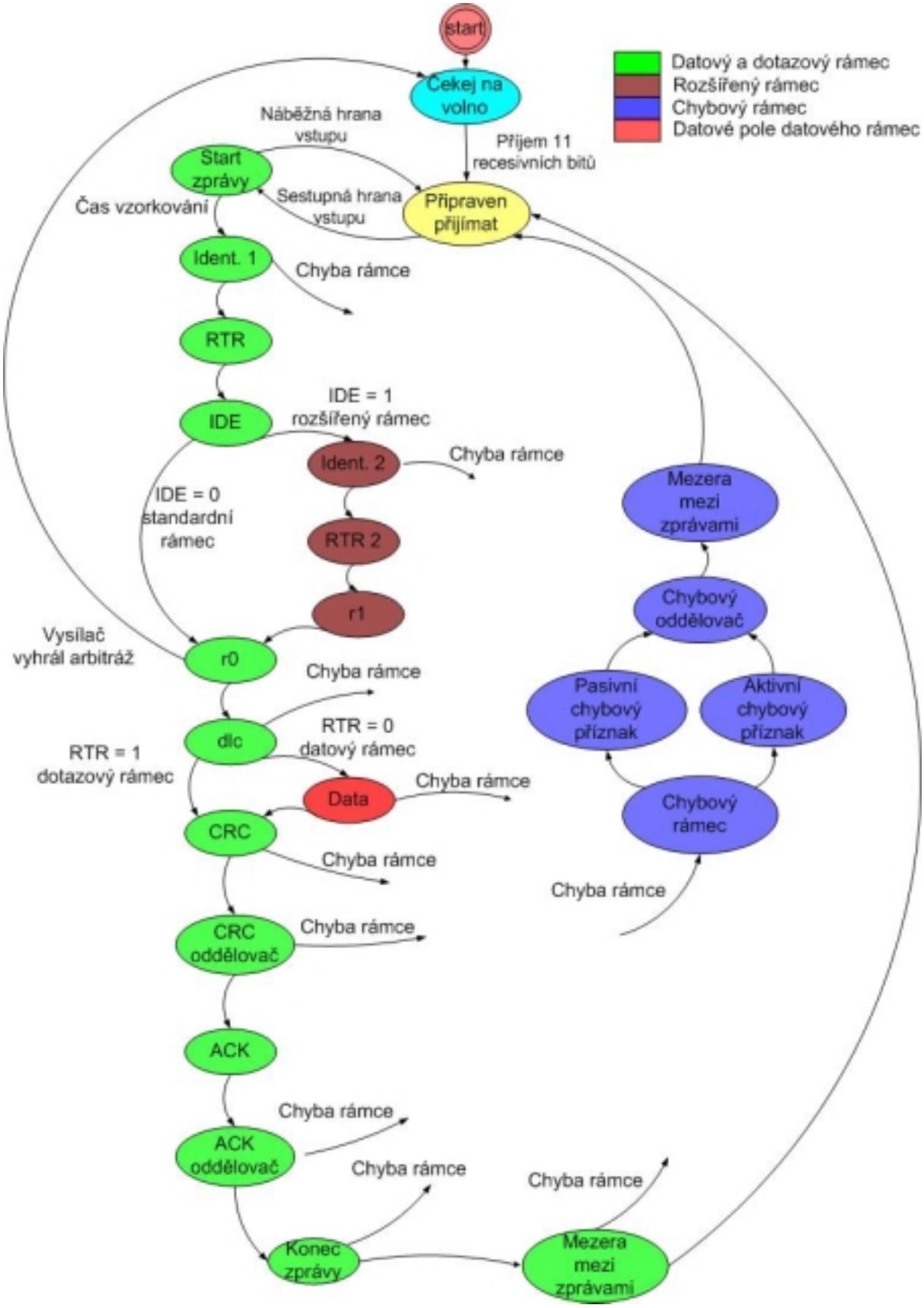
Po resetu je automaticky nastaven *servisní režim* řadiče, který blokuje přechod hlavního stavového stroje do stavu *čekání*. V tuto chvíli lze programovat registry týkající se časování řadiče, řadič je však od CAN sběrnice odpojen. Po přechodu ze *servisního* do *operačního režimu* řadič čeká, dokud se na sběrnici neobjeví 11 po sobě jdoucích recesivních bitů, značících konec zprávy a možnost odvysílání zprávy nové. Přijde-li tato sekvence, přejde řadič do stavu *čekání*, který odpovídá klidu na sběrnici. Je-li programově zadán příkaz k *vysílání*, přejde automat do tohoto stavu a zůstane v něm, buď dokud neodvysílá celou zprávu nebo dokud neztratí přístup na sběrnici vlivem prohrané arbitráže, v tom případě automaticky přechází do stavu *přijímání*. Další možností přechodu do stavu *příjmu* je detekce začátku nové zprávy na sběrnici, přechod zpět do stavu *čekání* je možný pouze po příjmu celé zprávy, popřípadě detekci chyby ve zprávě.

obr. 35 - Hlavní stavový stroj

3.3.4.Blok Příjmu

Chování tohoto bloku je popsáno jeho stavovým strojem na obr. 36. Základním stavem přijímacího bloku je stav *připraven přijímat*, který odpovídá klidu na sběrnici. Po zapnutí nebo resetu řadiče se iniciuje stav *start*, přechod do stavu *připraven přijímat* je podmíněn příjmem sekvence jedenácti po sobě jdoucích recesivních bitů, které oddělují jednotlivé zprávy na sběrnici CAN. Tímto je zajištěno, že řadič začne přijímat vždy na začátku zprávy a ne v jejím průběhu. Stav *připraven přijímat* je navržen jako cílový pro všechny možnosti vývoje algoritmu a navíc žádný jiný stav není blokován stavem na sběrnici. Vzorkování úrovně na sběrnici tedy slouží pouze pro větvení algoritmu, ale přechod mezi stavy je vždy iniciován časovacími signály a čítači přijatých bitů od začátku zprávy. Tato filozofie návrhu zamezuje možnosti, aby řadič z důvodu chybné zprávy zůstal čekat v jakémkoli jiném stavu než *připraven přijímat*. Je-li detekována jakákoli chyba v přijímané zprávě, je ošetřena vysláním příslušného chybového rámce. Druhy chyb nejsou na obrázku kvůli přehlednosti zobrazeny, mezi stavy *Ident. 1* až *CRC* oddělovač je kontrolována chyba vkládání bitů a je-li detekována, způsobí přechod do *stavu chybový rámec*, ostatní chyby jsou popsány přímo pro konkrétní stavy.

Ze stavu *připraven přijímat* je iniciován přechod do stavu *start zprávy*, při změně úrovně na sběrnici z recesivní na dominantní. V tomto stavu řadič čeká do příchodu signálu indikujícího čas vzorkování, kdy přechází do stavu *Ident. 1*. Změní-li se však úroveň na sběrnici zpět na recesivní před příchodem vzorkovacího signálu, vrátí se automat do stavu *připraven přijímat*. Toto slouží k odfiltrování krátkých rušivých signálů na sběrnici.



obr. 36- Stavový stroj příjmu

Stav *Ident. 1* trvá po dobu jedenácti bitů a je v něm přijímán identifikátor standardního rámce, popřípadě základní identifikátor rozšířeného rámce.

Ve stavu *RTR* je vzorkována hodnota tohoto bitu, která indikuje, zda se jedná o datový nebo dotazový rámec.

Hodnota bitu vzorkovaného ve stavu *IDE* rozlišuje typ přijímaného rámce. Je-li IDE = 0, je přijímán standardní rámec a následující stav bude *r0*. Je-li hodnota bitu opačná, je přijímán rozšířený rámec a následující stav bude *Ident.2*.

Ve stavu *Ident. 2* je přijímáno rozšíření identifikátoru rozšířeného rámce o délce 18 bitů.

Bit *RTR 2* vzorkovaný ve stejnojmenném stavu nahrazuje RTR bit standardního rámce.

Hodnoty bitů vzorkovaných ve stavech *r1* a *r0* se předpokládají dominantní, opačná hodnota je však také akceptována a nezpůsobí iniciaci chybového stavu. Ve stavu *r0* se navíc kontroluje hodnota signálu od vysílacího bloku informujícího o stavu arbitráže. Prohrál-li vysílač arbitráž, bude přijímač pokračovat v činnosti a následující stav bude *DLC*, v opačném případě musí přijímač počkat na začátek následující zprávy ve stavu *čekej na volno*.

Stav *DLC* má délku 4 bity, hodnota těchto bitů rozhoduje o délce datového pole a následném stavu. Je-li DLC = 0 nebo bit RTR = 1, bude následovat stav *CRC*, ve všech ostatních případech bude následovat stav *Data.*

Délka stavu *Data* v bitech je dána osminásobkem hodnoty registru DLC.

Na konci CRC stavu je kontrolován nulový obsah registru počítajícího CRC kód, značící bezchybné přijetí zprávy. Není-li registr roven nule, následuje přechod do chybového stavu.

CRC oddělovač vzorkovaný ve stejnojmenném stavu musí mít recesivní hodnotu, pokud je detekována dominantní, bude iniciován přechod do chybového stavu.

Ve stavu *ACK* je vysílána na sběrnici dominantní úroveň, pokud nebyla do této doby detekována žádná chyba.

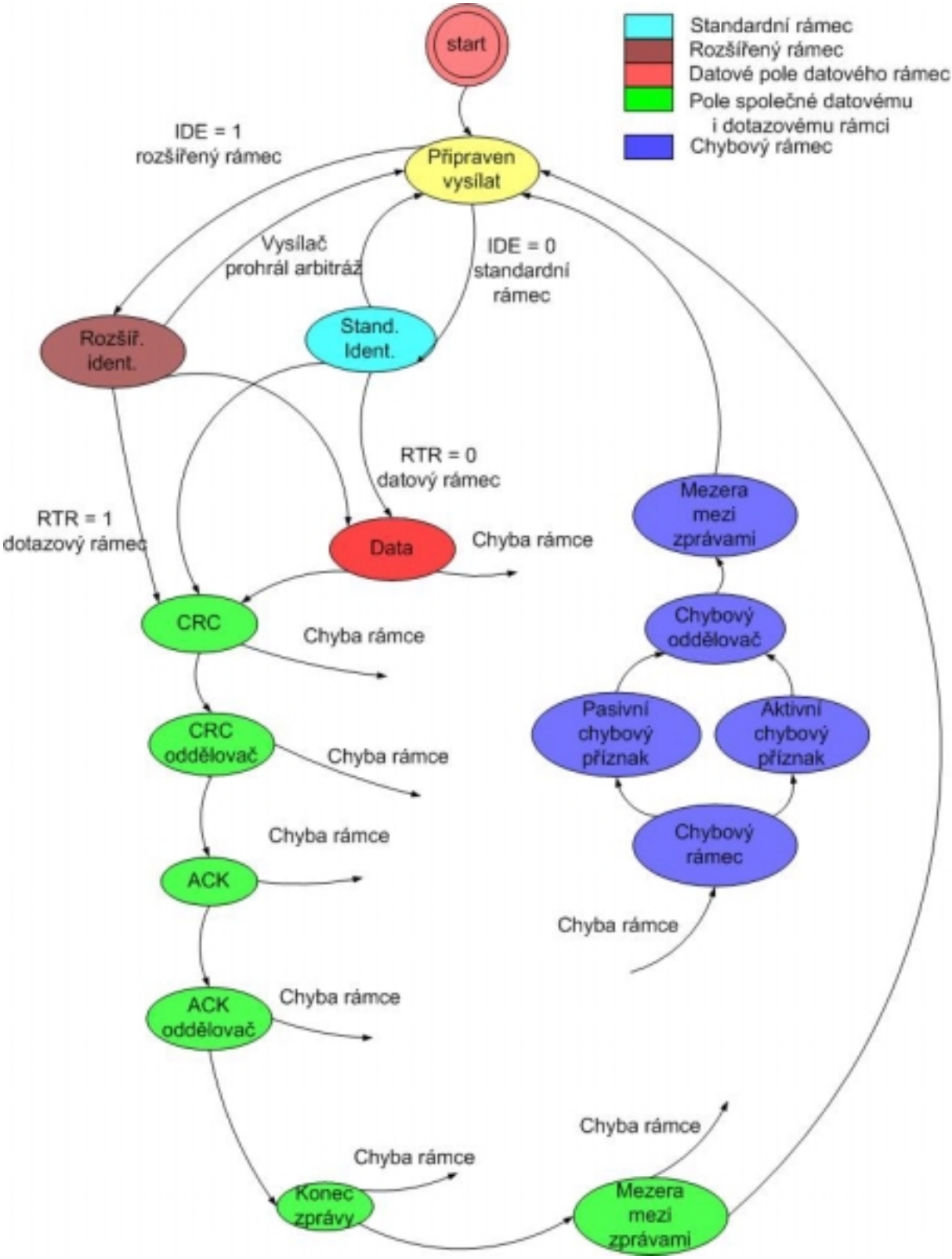
Navazující stavy *ACK Oddělovač*, *Konec zprávy* a *Mezera mezi zprávami* mají délku 1, 8 a 3 bity a musí být v jejich průběhu vzorkována recesivní úroveň sběrnice. Stane-li se tak, je při začátku posledního bitu stavu *Mezera mezi zprávami* indikován úspěšný příjem zprávy do řídicí části řadiče.

Odvysílání *chybového rámce* začíná stejnojmenným stavem, následující stav je vybírán podle hodnoty čítačů chybně přijatých a odeslaných zpráv. Je-li hodnota obou čítačů menší nebo rovna 127, následující stav bude *Aktivní chybový příznak,* v opačném případě bude následovat stav *Pasivní chybový příznak*. Minimální délka každého z těchto stavů je 6 bitů a prodlužuje se po dobu trvání dominantní úrovně na sběrnici.

Ve stavech *Chybový oddělovač* a *Mezera mezi zprávami* je vysílána recesivní úroveň na sběrnici o délce 8 a 3 bity. Následující stav je opět výchozí *Připraven přijímat*.

3.3.5. Blok Vysílání

Tento blok se stará o odvysílání zprávy na CAN sběrnici, jeho stavový stroj je na obr. 37. Hlavní stav je zde *Připraven vysílat*, do něhož se řadič dostane po resetu a po každém odvysílání zprávy, stejně jako u stavového stroje přijímače i zde jsou přechody ze všech ostatních stavů řízeny časovacími signály a čítači. Podmínkou pro přechod do *chybového stavu*, kdy je ukončeno vysílání a je odvysílán chybový rámec, může být pouze detekce dominantní úrovně na sběrnici u bitu, který byl odvysílán jako recesivní. Tento stav indikuje porušení vysílané zprávy a ta bude stornována. Tato situace se však netýká stavů *Stand. ident*, *Rozšíř. ident*. a *ACK*, viz další text. Stavový stroj vysílače také neobsahuje stavy zajišťující detekci mezery mezi zprávami. Zamezení počátku vysílání v době, kdy je na sběrnici vysílána jiná zpráva je řešeno pomocí nadřazeného hlavního stavového stroje, který informaci o přítomnosti zprávy nebo klidu na sběrnici získává z přijímacího bloku.



obr. 37 - Stavový stroj vysílače

Na rozdíl od přijímacího bloku zde lze data pro pole začátek zprávy až řídicí pole připravit před začátkem vysílání a celou tuto sekvenci vyslat najednou. Po příchodu signálu z řídicího bloku zahajujícího vysílání přejde stavový stroj do jednoho ze stavů *Stand. Ident.* nebo *Rozšíř. Ident.* a to podle toho, zda je požadováno vyslání zprávy se standardním nebo rozšířeným identifikátorem. Z těchto stavů není možný přechod do stavu indikujícího chybu. Pokud je zde detekována dominantní úroveň na sběrnici a vyslána byla recesivní, znamená to, že vysílač prohrál arbitráž, vrací se do stavu připraven vysílat a o odvysílání zprávy se pokusí ihned po skončení právě odvysílané zprávy. Dalšími možnými následníky stavů *Stand. Ident.* a *Rozšíř. Ident.* jsou stavy *Data* a *CRC*. Stav *Data* bude vybrán v případě, že je požadováno vyslat datový rámec, v případě dotazového rámce je stav *Data* přeskočen a následníkem je přímo stav *CRC*.

Ve stavu *CRC* o délce 15 bitů je vysílán CRC kód vypočtený stejnojmenným modulem, následný stav *CRC oddělovač* má délku 1 bit a v této době je vysílán recesivní signál.

Ve stavu *ACK* je vysílána recesivní úroveň, ale na rozdíl od všech ostatních stavů je očekávána úroveň dominantní, značící, že alespoň jeden z přijímačů zprávu přijal v pořádku. Opačný případ je vyhodnocen jako chyba.

O odvysílání zbylé části zprávy se starají tři následné stavy *ACK oddělovač*, *Konec zprávy* a *Mezera mezi zprávami* s délkami 1, 8 a 3 bity. Tyto všechny bity jsou vysílány jako recesivní a jsou-li detekovány se stejnou hodnotou, je na konci stavu *Mezera mezi zprávami* vysílání ukončeno a řídicí blok dostane signál o úspěšném odvysílání zprávy.

Stavy starající se o ošetření vzniku chyby a odvysílání chybového rámce mají totožný význam jako u přijímacího bloku.

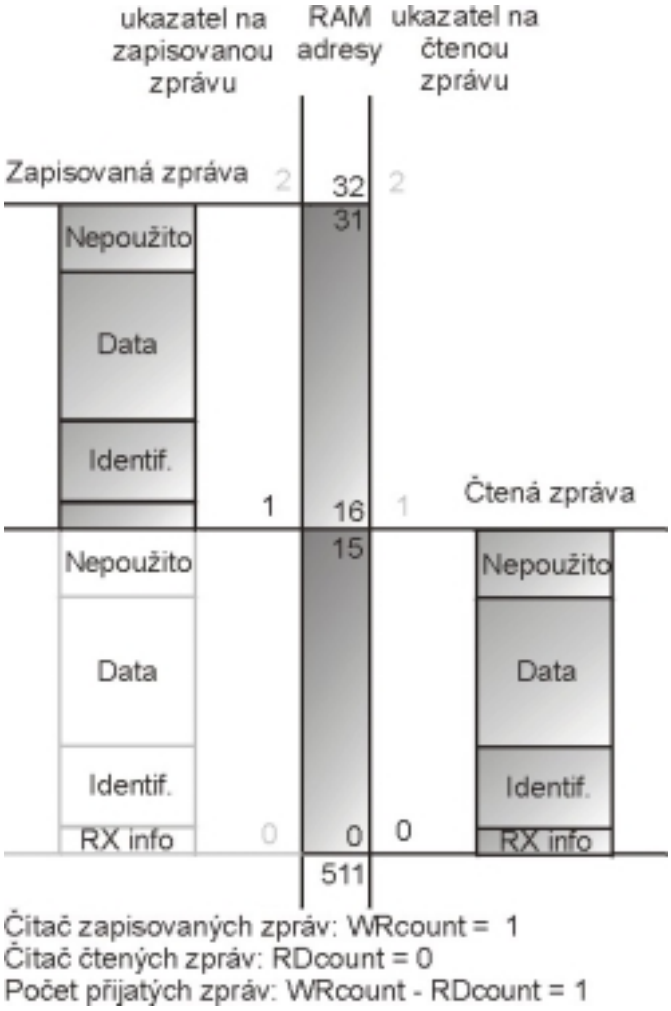
3.3.6. FIFO

Tento modul využívá vnitřní dvouportovou blokovou paměť RAM [11] o velikosti 4kb, která je implementována na čipu hradlového pole. Každý z portů paměti má nezávisle volitelný počet datových a tomu odpovídajících adresových signálů v rozsahu: data: 1 až 15, adresa 12 až 8. pro tento účel byly použity oba porty s osmibitovou datovou sběrnicí a odpovídající devítibitovou adresovou sběrnicí. Paměť v tomto případě umožňuje uchovávat 512 bytů dat. Počet bytů každé přijaté zprávy je proměnlivý a to v rozsahu od tří do třinácti. Minimální počet tři byty stačí pro popis zprávy standardní délky bez datového pole, maximální hodnota 13 je třeba, byla-li přijata zpráva rozšířeného formátu s plným počtem datových bytů. Pro uchovávání zpráv ve FIFO paměti tedy bylo zvoleno okno velikosti 16, které umožňuje skladovat v paměti maximálně 32 zpráv (viz obr. 38). V okně sice jsou byty, které nejsou nikdy využity, celočíselný počet zpráv v paměti však velmi zjednodušuje okolní logické obvody a možnost uchovat v paměti 32 zpráv najednou je pro použití v automatizaci více než dostatečná.

Vstupní adresa bloku je čtyřbitová a udává číslo bytu v zásobníku, tato adresa je použita jako spodní část adresy pro zápis do FIFO, příslušná horní část je tvořena ukazatelem na zapisovanou zprávu.

Při čtení zpráv ze zásobníku je použit stejný mechanismus, kde ukazatel zapisovaných zpráv je nahrazen ukazatelem na čtenou zprávu. Oba tyto ukazatele jsou tvořeny čtyřbitovými nereverzibilními čítači s inicializační hodnotou 0, které se inkrementují při zápisu zprávy/uvolnění zásobníku. Hodnota čítače přijatých zpráv je vypočtena jako rozdíl ukazatele na zapisovanou zprávu a ukazatele na čtenou zprávu.

Další logické obvody tohoto bloku poskytují informaci do stavového registru o přítomnosti zprávy v zásobníku (bit SR.0) a o zaplnění celého zásobníku (bit SR.1), v tomto případě také zamezí nahrání další zprávy do zásobníku.



obr. 38 - FIFO

Zdrojový kód modulu CAN_FIFO

Pro demonstraci rozdělení zdrojového kódu do jednotlivých modulů je zde uveden zkrácený zdrojový kód programového modulu CAN_FIFO. Zdrojové kódy jsou napsány v programovacím jazyce VHDL, na začátku každého modulu je uveden seznam použitých knihoven, dále následuje definice rozhraní pro připojení modulu do modulu nadřazeného. Zde jsou definovány signály včetně jejich šířky a rozlišení na vstupní, výstupní a vstupně-výstupní. Následují funkční popis modulu, na jehož začátku jsou případné definice rozhraní vnořených modulů a deklarace proměnných.

Definice jazyka VHDL[12] připouští zápis příkazů v sekvenčním a paralelním prostředí. Příkladem paralelního prostředí je přiřazení:

```
data_out <= data_o when reset_mode = '0' else X"FF";
```

Porovnávání hodnoty signálu reset_mode bude probíhat neustále a hodnota signálu data_out bude měněna ihned se zpožděním daným průchodem signálu danou logickou funkcí.

Příklad sekvenčního prostředí poskytuje úsek nazvaný write pointer. Zde je toto prostředí použito v operační části procesu, příkazy jsou vykonávány sekvenčně jako v jiných procedurálních jazycích. Sekvenční prostředí je použito u stavových strojů a při synchronním přiřazování hodnot signálům. V tomto konkrétním případě je proces spouštěn při každé náběžné hraně hodinového signálu, po té, je-li aktivní signál rst, je ukazatel Wr_pointer vynulován, v opačném případě je ukazatel inkrementován, je-li k tomu popud a není-li zásobník plný.

V rámci funkčního popisu je uvedeno mapování signálů případných vnořených modulů.

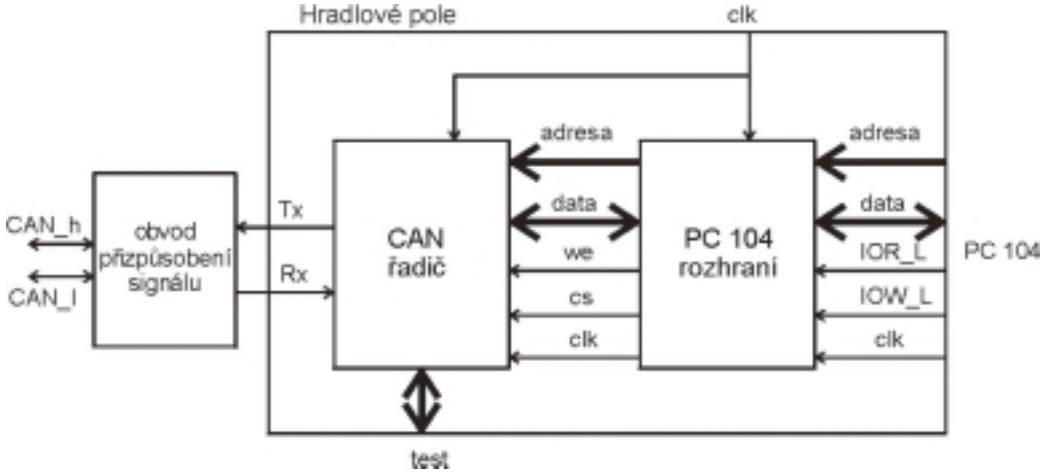
```
--pouzite knihovny
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
-- definice rozhrani pro pripojeni modulu do nadrazeneho modulu
entity fifo_can is
Port(clk              : in std_logic;            --Globalni hodinovy vstup
     rst              : in std_logic;            --Resetovaci vstup
     wr               : in std_logic;            --Zapis byte do FIFO
     wr_done          : in std_logic;            --Zprava zapsana
     read             : in std_logic;            --Cteni bytu z fifo
     data_in          : in std_logic_vector(7 downto 0);    --Vstupni data
     addr_w           : in std_logic_vector(3 downto 0);    --Adr. pro zapis
     addr_r           : in std_logic_vector(7 downto 0);    --Adr. pro cteni
     reset_mode       : in std_logic;            --Operacni/aplikacni rezim
     release_buffer   : in std_logic;            --Uvolni zpravu
     message_inside   : out std_logic;           --=1 je-li zprava uvnitr
     data_out         : out std_logic_vector(7 downto 0);   --Vystupni data
     test_fifo        : out std_logic_vector(16 downto 0)); --Test. vyst.

end fifo_can;

--funkcni popis bloku FIFO
architecture fifo_can_arch of fifo_can is

----- Komponenta dvouportove RAM
component RAMB4_S8_S8
--zde je umístěna definice vstupních a výstupních signálů RAM paměti
end component;

-- deklarace pouzitych promennych
  signal Mess_cnt    :std_logic_vector(4 downto 0); --pocet zprav ve FIFO
  signal Addr1       :std_logic_vector(8 downto 0); --adresa zapisu do RAM
  signal Release_old :std_logic;

begin
--mapovani modulu RAMfifo
  RAMfifo : RAMB4_S8_S8
  port map (data_in, "00000000", Wr,Read, Wr, '0', rst, rst, clk, clk,
addr1, addr2, open, data_o);

  Addr1 <= Wr_pointer&Addr_w;
  Addr2 <= Rd_pointer&(Addr_r(3 downto 0));
  data_out <= data_o when reset_mode = '0' else X"FF";

  --- Write pointer
  process(clk)
  begin
  if (Clk = '1' and clk'event) then
      if (Rst = '1' ) then Wr_pointer <="00000";
     elsif(Wr_done='1'and Mess_cnt /="11111")then Wr_pointer<=Wr_pointer+1;
      end if;
  end if;
  end process;

  --Message counter
  Mess_cnt <= Wr_pointer - Rd_pointer;
  Message_inside <= '1' when Mess_cnt /= "00000" else '0';
end fifo_can_arch;
```

3.4. IMPLEMENTACE ŘADIČE

Pro testování funkce řadiče byl použit vývojový kit firmy Insight electronics typ DS-KIT-2S100-EURO. Tento kit obsahuje hradlové pole Spartan II XC2S100, zdroj hodinového kmitočtu, konfigurační paměť EEPROM a obvody napájení. Hradlové pole má vyvedené všechny své vstupy a výstupy na konektory, dále lze k němu volitelně připojit; seriovou linku RS 232, dvousegmentový displej, čtyři vstupní tlačítka a tři LED diody. Pro testovací účely bylo třeba vytvořit rozhraní, které by umožnilo nadřazenému mikroprocesoru přístup do registrů řadiče a tím jeho řízení. Toto rozhraní bylo vytvořeno ve dvou modifikacích: pro přístup k řadiči přes sériovou linku a sběrnici PC 104.

3.4.1.Řízení řadiče přes sběrnici PC 104

Tato šestnáctibitová sběrnice je hojně používána v průmyslových počítačích a je obdobou dnes už řídce používané sběrnice ISA. Vzhledem k šířce datové sběrnice řadiče je zde použita pouze poloviční osmibitová verze sběrnice. Blokové schéma viz obr. 39.



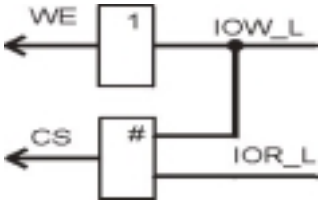
obr. 39 - Řízení řadiče přes sběrnici PC104

Řadič je umístěn v hradlovém poli spolu s rozhraním ke sběrnici PC 104, celý obvod komunikuje s okolím pomocí signálů, které lze rozdělit do čtyř skupin:

1. vstup hodinového signálu,
2. sběrnice PC 104,
3. sběrnice CAN,
4. testovací výstupy.

3.4.1.1.PC 104 ROZHRANÍ

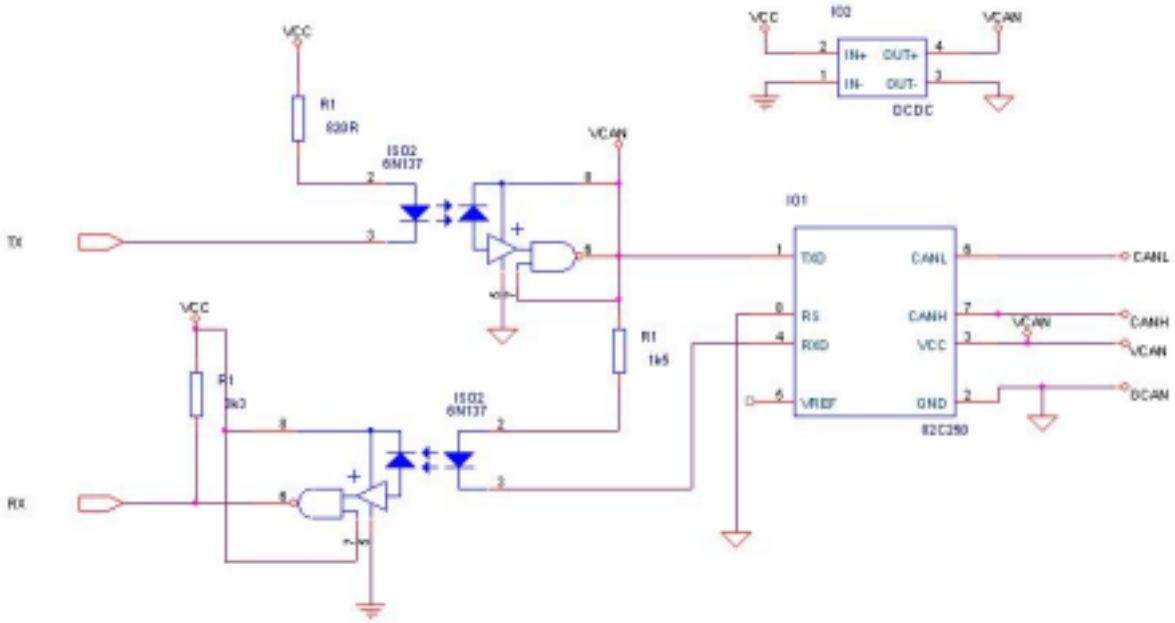
Sběrnice PC 104 je podobná sběrnici použité u řadiče CAN, liší se pouze významem řídících signálů. Řadič CAN používá řídící signály kompatibilní s mikroprocesorem 8051[15] . Požadavek na čtení (resp. zápis) je dán polaritou řídícího signálu WE, pro zápis je v log. 1 (resp. log. 0). Požadovaná činnost je provedena, pokud je signál CS v log. 1 a nastala-li náběžná hrana hodinového signálu. Sběrnice PC 104 používá řídící signály IOR_L signalizující požadavek na čtení a IOW_L signalizující požadavek na zápis. Oba dva signály jsou aktivní v log. nule. Zapojení bloku PC 104 rozhraní transformujícího řídící signály je na obr. 40.



obr. 40 - PC 104 rozhraní

3.4.1.2.OBVOD PŘIZPŮSOBENÍ ÚROVNĚ SIGNÁLU

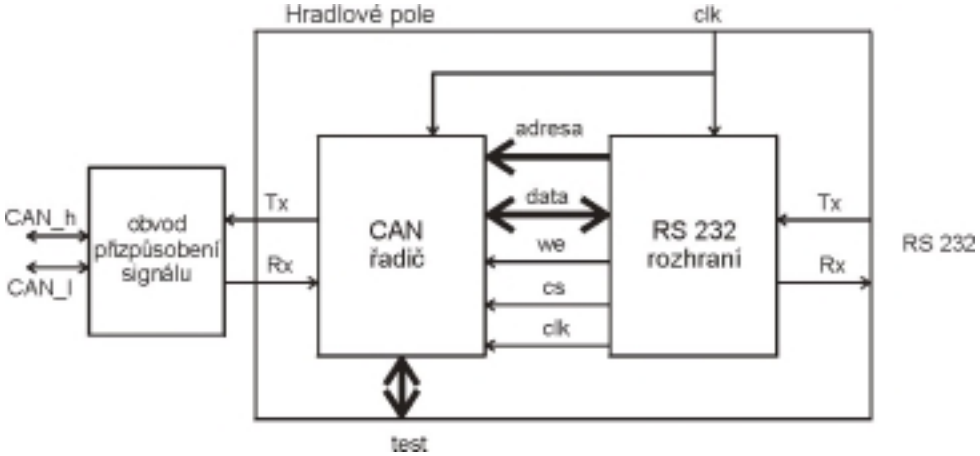
Výstupní signály CAN řadiče jsou připojeny na sběrnici pomocí obvodu, jehož schéma je na obr. 41. Hlavní funkcí obvodu je galvanické oddělení elektronických obvodů s hradlovým polem od sběrnice CAN. Oddělení je realizováno pomocí dvou optočlenů 6N137. Jako budič sběrnice je použit obvod 82C250[16] firmy Philips.



obr. 41 - Zapojení obvodu pro galvanické oddělení od CAN sběrnice

3.4.2.Řízení řadiče přes rozhraní RS 232

Jako alternativa k rozhraní PC 104 byl vytvořen modul umožňující řídit řadič přes sériovou linku RS 232, viz obr. 42.



obr. 42 – Řízení řadiče přes rozhraní RS 232

Tento modul umožňuje zápis a čtení dat z registrů CAN řadiče přes sériovou linku pomocí komunikačního protokolu, viz obr. 43. Modul reaguje na dva druhy přijímaných zpráv:

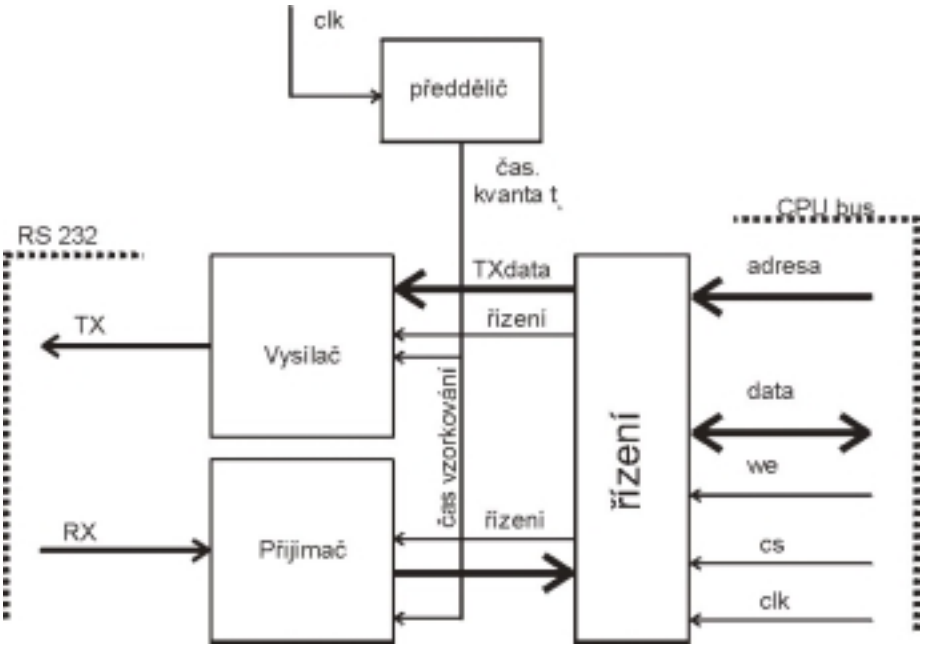
Zprávu začínající bytem AA, který je následován byty označujícími adresu, data a kontrolní součet. Po úspěšném přijetí této zprávy bude datový byte zapsán na adresu ADR_W registru CAN řadiče. Kontrolní součet této zprávy je počítán jako součet všech předchozích bytů bez přenosu do vyššího řádu.

Druhá možná zpráva začíná bytem A5 a je následována bytem ADR_R majícím význam adresy. Po přijetí této zprávy je přečten obsah CAN registru na adrese ADR_R a jeho obsah je vyslán zpět ve zprávě, která začíná bytem AA, následuje jej adresa ADR_R a zakončuje čtený byte DATA_R.

obr. 43 - Komunikace přes rozhraní RS 232

3.4.2.1. STRUKTURA OBVODU TVOŘÍCÍHO ROZHRANÍ

Tento obvod slouží pro dekódování zpráv přijatých po sériové lince a jejich transformaci na zápis dat do registrů CAN řadiče a dále pro čtení dat z požadované adresy a jejich vyslání. Využívá k tomu dvou modulů zajišťujících vysílání resp. přijímání jednoho byte, viz obr. 44.



obr. 44 - Blokové schéma obvodu pro řízení CAN řadiče přes sériovou linku

Komunikace probíhá v konfiguraci 1 start bit, 8 datových bitů, sudá parita a jeden stop bit. Přijímač je neustále připojen na sériovou linku a přijme-li kompletní byte, předá jej řídicímu bloku. Ten ve stavu čekání akceptuje pouze dva znaky a to AA nebo A5 udávající začátek příslušné zprávy. Po přijetí znaku AA je očekáván příjem dalších tří znaků, z nichž poslední je kontrolní součet, jehož hodnota se porovnává s vypočtenou, a jsou-li shodné, jsou vyslány signály na sběrnici umožňující zápis datového bytu na přijatou adresu.

Po přijetí znaku A5 je očekáván příjem dalšího znaku majícího význam adresy ADR_R. Tato událost iniciuje čtení dat z adresy ADR_R a jejich zpětné odeslání. Odesílání je realizováno pomocí samostatného bloku, proto lze v době vysílání zároveň přijímat další řídící zprávu.

3.5. POPIS REGISTRŮ ŘADIČE

Řadič je možno připojit do vstupně výstupního adresového prostoru nadřazeného mikroprocesoru, jeho činnost je poté řízena pomocí přístupu do jeho registrů. Adresový rozsah řadiče je 0-31, na sběrnici je připojen osmi adresovými vodiči, tři nejvýznamnější bity nejsou dekódovány. Umístění registrů je kvůli kompatibilitě zvoleno shodné jako u řadiče SJA1000 [14] . Tento řadič může pracovat v různých režimech lišících se i mapováním registrů, za vzor byl vybrán peliCAN režim ve verzi pro rozšířené rámce zpráv.

3.5.1.Názvy a umístění registrů v I/O prostoru řadiče

V následující tabulce jsou uvedeny názvy registrů a jejich umístění v adresovém prostoru. Pokud je řadič připojen všemi osmi adresovými vodiči, budou se registry vždy po třiceti dvou opakovat v celém adresovém prostoru.

Tabulka 3: Umístění registrů v I/O prostoru řadiče

Adresa	Operační režim		Servisní režim	
	Čtení	**zápis**	**Čtení**	**Zápis**
0	Režim (mode)	Režim	Režim	Režim
1	(00H)	Příkaz(command)	(00H)	Příkaz
2	Stav (state)	-	Stav	-
3	00h - rezerva	-	00h - rezerva	-
4	00h - rezerva	-	00h - rezerva	-
5	00h - rezerva	-	00h - rezerva	-
6	Registr časování 0[1]	-	Registr časování 0	Registr časování 0
7	Registr časování 1	-	Registr časování 1	Registr časování 1
8	00h - rezerva	-	00h - rezerva	-
9	00h - rezerva	-	00h - rezerva	-
10	00h - rezerva	-	00h - rezerva	-
11	00h - rezerva	-	00h - rezerva	-
12	00h - rezerva	-	00h - rezerva	-
13	00h - rezerva	-	00h - rezerva	-
14	00h - rezerva	-	00h - rezerva	-
15	00h - rezerva	-	00h - rezerva	-
16	RX frame information [2]	TX frame information [1]	00h - rezerva	-

[1] Bus timing 0
[2] RX frame information = Informace o přijatém rámci

17	RX identifikátor 1	TX identifikátor 1	00h - rezerva	-
18	RX identifikátor 2	TX identifikátor 2	00h – rezerva	-
19	RX identifikátor 3	TX identifikátor 3	00h – rezerva	-
20	RX identifikátor 4	TX identifikátor 4	00h – rezerva	-
21	RX data 1	TX data 1	00h – rezerva	-
22	RX data 2	TX data 2	00h – rezerva	-
23	RX data 3	TX data 3	00h - rezerva	-
24	RX data 4	TX data 4	00h - rezerva	-
25	RX data 5	TX data 5	00h - rezerva	-
26	RX data 6	TX data 6	00h - rezerva	-
27	RX data 7	TX data 7	00h - rezerva	-
28	RX data 8	TX data 8	00h - rezerva	-
29	00h - rezerva	-	00h - rezerva	-
30	00h - rezerva	-	00h - rezerva	-
31	00h - rezerva	-	00h - rezerva	-

3.5.2.Registr režimu (MODE REGISTER)– adresa 0

Slouží ke změně chování řadiče, jednotlivé bity mohou být nastaveny nebo vynulovány prostřednictvím CPU, pro kterou se registr jeví jako paměť umožňující čtení i zápis. Rezervované bity jsou čteny jako logická nula.

Tabulka 4: Interpretace bitů registru režimu (MOD), CAN adresa 0

bit	symbol	jméno	hodnota	význam
MOD.7 –MOD.1	-	rezerva	-	-
MOD.0	RM	Servisní režim (Reset mode)	1	Servisní režim – detekce tohoto bitu způsobí konec vysílání/přijímání a přechod do servisního režimu
			0	Operační režim (Operating mode)

Po hardwarovém resetu je hodnota bitu RM vždy 1 (Servisní režim), po nastavení komunikačních parametrů uvede řadič do operačního režimu nadřízený mikroprocesor změnou hodnoty bitu RM do nuly. Při příchodu požadavku na změnu režimu na operační čeká CAN řadič na výskyt jedenácti po sobě jdoucích recesivních bitů na CAN sběrnici. Poté přejde do operačního režimu a je připraven přijímat a vysílat zprávy.

3.5.3.Registr příkazů (COMMAND REGISTER)– CAN adresa1

Registr příkazů slouží k zadávání povelů řadiči CAN ze strany CPU, je umožněn pouze zápis, při požadavku na čtení je vrácena hodnota logická 0.

Tabulka 5: Interpretace bitů registru příkazů (CMR), CAN adresa 1

bit	**symbol**	**jméno**	**hodnota**	**význam**
CMR.7 –CMR.4	-	rezerva	-	-
CMR.3	CDO	Vynuluj příznak přetečení [1] (Clear Data Overrun)	1	Příznak přetečení vynulován
			0	- (žádná akce)

[1] TX frame information = Informace o vysílaném rámci

CMR.2	RRB	Uvolni přijímací zásobník [2] (Release Receive Buffer)	1	Přijímací registr reprezentován oblastí v přijímací FIFO paměti je uvolněn
			0	- (žádná akce)
CMR.1	AT	Stornuj vysílání [3] (Abort Transmission)	1	Pokud ještě nezačalo, vysílání aktuální zprávy je zrušeno
			0	- (žádná akce)
CMR.0	TR	Požadavek na vysílání [4] (Transmission Request)	1	Požadavek na vysílání
			0	- (žádná akce)

Poznámky:

1. Příkaz je použit pro vynulování příznaku přetečení, které je indikováno stavovým registrem.
2. Po přečtení obsahu přijímacích registrů může CPU uvolnit tento prostor ve FIFO paměti zapsáním log 1 do tohoto bitu.
3. Příkaz stornuj vysílání lze použít, pokud CPU požaduje stornovat svůj požadavek na vysílání zprávy nebo pokud je potřeba odvysílat jinou zprávu s větší prioritou. Zpráva, která se již začala vysílat není zastavena. Při potřebě zjistit, jestli byla originální zpráva odvysílána, je třeba sledovat hodnotu bitu vysílání dokončeno (SR.3).
4. Pokud byla nastavena hodnota bitu požadavek na vysílání (CMR.0) do log. 1, není možné vysílání přerušit nastavením tohoto bitu do log. 0. Vysílání lze přerušit pouze nastavením bitu Stornuj vysílání (CMR.1) do log.1.

3.5.4.Stavový registr (STATUS REGISTER)– adresa 2

Položky stavového registru představují hodnoty jednotlivých stavů CAN řadiče, hodnoty registru lze pouze číst, případný zápis nebude proveden.

Tabulka 6: Interpretace bitů stavového registru (SR), CAN adresa 2

bit	symbol	jméno	hodnota	význam
SR.7 –SR.6	-	rezerva	-	-
SR.5	TS	Stav vysílání [1] (Transmit Status)	1	Vysílání – zpráva je vysílána řadičem
			0	Čekání – momentálně není vysílána žádná zpráva
SR.4	RS	Stav přijímání [1] (Receive Status)	1	Přijímání – zpráva je přijímána řadičem
			0	Čekání – momentálně není přijímána žádná zpráva
SR.3	TCS	Vysílání dokončeno [2] (Transmission Complete Status)	1	Dokončeno – poslední požadovaná zpráva byla odvysílána
			0	Nedokončeno – poslední požadovaná zpráva se stále vysílá
SR.2	TBS	Stav vysílacího zásobníku [3] (Transmit Buffer Status)	1	Uvolněn – CPU může zapisovat do tohoto zásobníku
			0	Zamčen – CPU nesmí zapisovat do vysílacího zásobníku, poslední požadovaná zpráva se stále vysílá

SR.1	DOS	Příznak přetečení [4] (Data Overrun status)	1	Přetečení – přijatá zpráva byla ztracena protože přijímací FIFO paměť je zaplněna
			0	Od posledního vynulování tohoto bitu nedošlo k přetečení zásobníku
SR.0	RBS	Stav přijímacího registru [5] (Receive Buffer Status)	1	Plný – jedna nebo více zpráv jsou dostupné v přijímací FIFO
			0	Prázdný – žádná zpráva není k dispozici

Poznámky:

1. Pokud jsou oba stavové bity vysílání i příjmu v log. 0 není na CAN sběrnici žádná aktivita.
2. Hodnota bitu vysílání dokončeno je nastavena do logické 0 po uvedení bitu požadavek na vysílání do logické 1. Bit vysílání dokončeno zůstane v logické nule do doby úspěšného odeslání zprávy.
3. Pokud se CPU pokusí zapisovat do vysílacího zásobníku v případě, že hodnota tohoto bitu je logická nula, hodnota zapisovaného byte nebude akceptována a tato situace nebude jinak indikována.
4. Příznak přetečení je nastaven v případě, že čítač přijatých zpráv dosahuje své maximální hodnoty 31, v tomto případě každá korektně přijatá zpráva bude ztracena. Příznak přetečení lze vynulovat pomocí bitu CMR.3 příkazového registru.
5. Po přečtení zprávy umístěné ve FIFO a uvolnění tohoto paměťového místa pomocí příkazu CMR.2 příkazového registru je tento bit vynulován, pokud je ve FIFO přítomna další zpráva, bit je opět nastaven do log. 1.

3.5.5.Registr časování 0 (Bus Timing Register 0) – CAN adresa 6

Obsah tohoto registru definuje hodnoty předděliče a synchronizačního skoku. Registr je přístupný pouze v servisním režimu, v operačním režimu je možné ho pouze číst.

Tabulka 7: Interpretace bitů registru časování 0 (BTR 0), CAN adresa 6

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
SJW.1	SJW.0	BRP.5	BRP.4	BRP.3	BRP.2	BRP.1	BRP.0

3.5.5.1. PŘEDDĚLIČ (BAUD RATE PRESCALER - BPR)

Perioda systémových hodin CAN řadiče t_{scl} je programovatelná a je výchozí pro časování bitů. Pro výpočet periody t_{scl}, která se rovná délce jednoho časového kvanta, platí následující vztah:

$$t_q = t_{IN}.(32 \times BRP.5 + 16 \times BRP.4 + 8 \times BRP.3 + 4 \times BRP.2 + 2 \times BRP.1 + BRP.0),$$

kde t_{IN} je perioda vstupního signálu $t_{IN} = \frac{1}{f_{IN}}$.

3.5.5.2.SYNCHRONIZAČNÍ SKOK (SYNCHRONIZATION JUMP WIDTH - SJW).

Pro kompenzaci fázového rozdílu oscilátorů v řadičích CAN účastnících se komunikace se přijímač synchronizuje se vstupním signálem při každé změně jeho úrovně. Synchronizační

skok udává hodnotu v časových kvantech t_q, o kterou se při synchronizaci může zkrátit/prodloužit čas jednoho bitu.

$$t_{SJW} = t_q.(2 \times SJW.1 + SJW.0 + 1)$$

3.5.6. Registr časování 1 (Bus Timing Register 1) – CAN adresa 7

Obsah tohoto registru pomocí proměnných časový segment 1 (Time Segment 1) a časový segment 2 (Time Segment 2) definuje délku periody jednoho bitu a dobu, kdy se vzorkuje vstupní signál. Registr je přístupný pouze v servisním režimu, v operačním režimu je možné ho pouze číst.

Tabulka 8: Interpretace bitů registru časování 1 (BTR 1), CAN adresa 7

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
-	TSEG2.2	TSEG2.1	TSEG2.0	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0

Bit 7 registru časování 1 není použit, lze do něj zapisovat i číst, zapsaná hodnota však nemá žádný význam.

Časové segmenty 1 a 2 (Tseg1 a Tseg2) definují délku jednoho bitu v počtu časových kvant t_q, význam těchto registrů je popsán v odstavci 3.2.2.

Délky jednotlivých segmentů jako násobky časového kvanta t_q:

$$t_{SYNCSEG} = 1 \times t_q,$$

$$t_{TSEG1} = t_q.(8 \times TSEG1.3 + 4 \times TSEG1.2 + 2 \times TSEG1.1 + TSEG1.0 + 1),$$

$$t_{TSEG2} = t_q.(4 \times TSEG2.2 + 2 \times TSEG2.1 + TSEG2.0 + 1).$$

Délka jednoho bitu: $T = t_{SYNCSEG} + t_{TSEG1} + t_{TSEG2}$.

3.5.7. Vysílací zásobník

Vysílací zásobník má délku 13 byte a je umístěn v adresové části CAN řadiče mezi adresami 16 a 28.

Zásobník je rozdělen do tří částí, první nese informace o vysílané zprávě jako formát zprávy a počet datových byte, ve druhé jsou umístěny identifikátory a ve třetí data pro vysílanou zprávu.

Tabulka 9:

Členění vysílacího zásobníku

CAN adresa	význam
16	Inf. o vysílané zprávě
17	TX identifikátor 1
18	TX identifikátor 2
19	TX identifikátor 3
20	TX identifikátor 4
21	TX data byte 1
22	TX data byte 2
23	TX data byte 3

24	TX data byte 4
25	TX data byte 5
26	TX data byte 6
27	TX data byte 7
28	TX data byte 8

Tabulka 10: Informace o vysílané zprávě (TX frame information), CAN adresa 16

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
FF [1]	RTR [2]	X [3]	X [3]	DLC.3 [4]	DLC.2	DLC.1	DLC.0

Poznámky:

1. Formát rámce (Frame format FF) FF=1, pokud je požadavek na odvysílání rozšířeného rámce a FF=0, pokud se má vyslat standardní rámec.
2. Požadavek na vzdálené vysílání (Remote transmission request RTR). Pokud je RTR=1, je vyslán dotazový rámec s nulovým počtem datových bytů bez ohledu na požadovanou délku dat DLC.
3. Nepoužito, do registru lze zapisovat, hodnota však nemá žádný význam.
4. Počet datových byte (Data Length Code DLC) udává, kolik datových byte bude ve zprávě odvysíláno.

Tabulka 11: TX identifikátor 1, CAN adresa 17 [1]

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21

Tabulka 12: TX identifikátor 2, CAN adresa 18 [1]

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13

Tabulka 13: TX identifikátor 3, CAN adresa 19 [1]

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5

Tabulka 14: TX identifikátor 4, CAN adresa 20 [1]

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ID.4	ID.3	ID.2	ID.1	ID.0	X	X	X

Tabulka 15: Význam bitů formát rámce (FF) a požadavek na vzdálené vysílání (RTR)

BIT	Hodnota	význam
FF	1	Požadavek na vyslání rozšířeného rámce (EFF)
	0	Požadavek na vyslání standardního rámce (SFF)
RTR	1	Požadavek na dotazový rámec (Remote transmission request RTR), vysílaný rámec nebude obsahovat datové pole
	0	Bude odvysílán datový rámec

[1] ID.X znamená bit X identifikátoru

3.5.7.1. POČET DATOVÝCH BYTE (DATA LENGTH CODE DLC)

Počet datových bytů ve vysílané zprávě je kódován pomocí DLC bitů. Je-li požadavek na vyslání dotazového rámce, je počet datových bytů automaticky vynulován. Přesto je nutné zapsat správnou informaci do DLC registru, aby se zabránilo chybám na sběrnici, pokud dva vysílače současně vysílají dotazový rámec se stejným identifikátorem.

Rozsah dat N může být v rozsahu 0 až 8 byte a je kódován pomocí vztahu:

$N = 8 \times DLC.3 + 4 \times DLC.2 + 2 \times DLC.1 + DLC.0$.

Z důvodů kompatibility by N nemělo být větší než 8, pokud bude taková hodnota detekována, bude odvysíláno pouze 8 byte.

3.5.7.2. IDENTIFIKÁTOR (IDENTIFIER ID)

Ve standardním formátu zprávy (SFF) je identifikátor 11 bitů dlouhý. Pokud je požadavek tuto zprávu vysílat (FF = 0), bude odvysílán identifikátor sestávající se z bitů ID.28 až ID.18 a ostatní bity identifikátoru budou ignorovány. Rozšířený formát zprávy (EFF) obsahuje 29 bitů v identifikátoru, v jeho případě se odvysílá všech 29 bitů z registrů CAN 17 až CAN 20. Bit ID.28 je nejvýznamnější a bude odvysílán jako první. Identifikátor se používá pro filtraci zpráv pomocí akceptačního filtru. Také udává prioritu zprávy při přístupu na sběrnici při arbitrážním procesu. Čím má identifikátor nižší hodnotu, tím má větší prioritu při přístupu na sběrnici.

3.5.8. Přijímací zásobník

Přijímací zásobník má délku 13 byte a je umístěn v adresové části CAN řadiče mezi adresami 16 a 28, kde je přístupný pouze pro čtení.

Počet, význam i umístění jednotlivých registrů přijímacího zásobníku jsou obdobné jako u zásobníku vysílacího.

Ve skutečnosti tvoří přijímací zásobník okno zobrazující úsek z přijímací FIFO paměti, kam se ukládají přijaté zprávy. Po přečtení všech registrů, kterých se týkala přijatá zpráva, je možné přesunout okno ve FIFO pomocí zápisu log. 1 do bitu CMR.2 příkazového registru, čímž se okno přesune na další přijatou zprávu, pokud byla skutečně přijata.

Tabulka 16: Členění přijímacího zásobníku

CAN adresa	význam
16	Inf. o přijaté zprávě
17	RX identifikátor 1
18	RX identifikátor 2
19	RX identifikátor 3
20	RX identifikátor 4
21	RX data byte 1
22	RX data byte 2
23	RX data byte 3
24	RX data byte 4
25	RX data byte 5
26	RX data byte 6
27	RX data byte 7
28	RX data byte 8

Tabulka 17: Informace o přijaté zprávě (RX frame information), CAN adresa 16

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
FF [1]	RTR [2]	X [3]	X [3]	DLC.3 [4]	DLC.2	DLC.1	DLC.0

Poznámky:

1. Formát rámce (Frame format FF) FF=1, pokud je požadavek na odvysílání rozšířeného rámce a FF=0, pokud se má vyslat standardní rámec.
2. Požadavek na vzdálené vysílání (Remote transmission request RTR). Pokud je RTR=1, je vyslán dotazový rámec s nulovým počtem datových bytů bez ohledu na požadovanou délku dat DLC.
3. Nepoužito, do registru lze zapisovat, hodnota však nemá žádný význam.
4. Počet datových byte (Data Length Code DLC) udává, kolik datových byte bude ve zprávě odvysíláno.

Tabulka 18: RX identifikátor 1, CAN adresa 17 [1]

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21

Tabulka 19: RX identifikátor 2, CAN adresa 18 [1]

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13

Tabulka 20: RX identifikátor 3, CAN adresa 19 [1]

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5

Tabulka 21: RX identifikátor 4, CAN adresa 20 [1]

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ID.4	ID.3	ID.2	ID.1	ID.0	X	X	X

3.5.9. Čítač přijatých zpráv (RX message counter - RMC)– adresa 29

Hodnota v tomto registru odpovídá počtu přijatých zpráv, které jsou ve FIFO paměti. Čítač je inkrementován při každé korektně přijaté zprávě a dekrementován při každém uvolnění přijímacího registru (bit CMR.2 přijímacího registru). Hodnotu tohoto registru je možné pouze číst.

tabulka 22: Čítač přijatých zpráv (RMC), CAN adresa 29

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
(0)	(0)	(0)	RMC.4	RMC.3	RMC.2	RMC.1	RMC.0

[1] ID.X znamená bit X identifikátoru

3.6. ZKUŠENOSTI S OŽIVOVÁNÍM ŘADIČE

Jazyk VHDL má přísnou syntaxi, jejíž dodržování je první předpoklad vyvarování se problémů s oživováním jednotlivých modulů. Vzhledem k tomu, že zdrojový kód tohoto jazyka je překladačem přeložen na zapojení elementárních logických členů, tvořících logický obvod vykonávající požadovanou funkci, je zde třeba respektovat další zásady na rozdíl od běžných procedurálních jazyků. Typicky je třeba si uvědomit šířku signálu použitého pro řízení: pro přechod stavového stroje z jednoho stavu do druhého je potřeba signál dlouhý minimálně jednu periodu hodinového signálu, zatímco pro inkrementaci čítače musí být signál široký právě jednu periodu hodinového signálu. Nedodržení tohoto pravidla vede k chybné funkci navrženého obvodu a odhalení tohoto omylu je často obtížné.

Problémy při syntéze přijímacího bloku:

Stavový stroj přijímacího bloku CAN řadiče, jak je popsáno v odstavci 3.3.4, je navržen tak, aby se z jakéhokoli stavu sám vracel do stavu připraven přijímat. Přesto při testování jedné z pracovních verzí řadiče došlo k situaci, kdy byl klid na sběrnici a uběhl dostatečně dlouhý čas od poslední zprávy, stavový stroj však evidentně nebyl ve stavu připraven přijímat. Byly změřeny hodnoty signálů indikujících přítomnost stav. stroje ve všech stavech a ani jeden z nich nebyl aktivní. Zápis stavového stroje obsahuje povinnou položku when others, která slouží k ošetření situací, pokud se stavový stroj dostane do stavu, jež není uveden v předchozím výčtu. Tyto příznaky jsou však protichůdné. Situace byla vyřešena zavedením hlídacího obvodu watch dog, který obsahuje čítač bitů od začátku zprávy nulovaný ve stavu připraven přijímat. Je-li hodnota tohoto čítače větší než nejdelší možná zpráva, je iniciován přechod do stavu čekej na volno. Tato úprava odstranila popisovaný problém a bylo detekováno působení watch dog obvodu průměrně u každé dvacáté zprávy. Zajímavé bylo zjištění, že u další verze řadiče již k aktivaci hlídacího obvodu nedocházelo vůbec, i když se změny kódu stavového stroje přímo netýkaly. Pravděpodobná příčina popisovaného chování tedy byla špatná syntéza obvodu překladačem, kdy mohlo docházet k nesprávnému dekódování stavových proměnných vlivem zpoždění průchodu signálu logickými obvody. Hlídací obvod byl však kvůli dalším testům v modulu ponechán.

4. ZÁVĚR

Tato diplomová práce nabízí řešení modernizace řídicího systému NX 5030, ve své první části popisuje způsob, jak nahradit jeho část pro připojení binárních vstupů a výstupů pomocí PLC automatu. Popisované řešení bylo implementováno do inovovaného systému NX 5031, který je nyní ve výrobním sortimentu firmy Intronix s.r.o. Z tohoto důvodu obsahuje popisované řešení zkušenosti s uváděním do provozu a s vlastní činností inovovaného systému.

V druhé části práce je popsána následná inovace řídicího systému týkající se možnosti komunikovat po sběrnici CAN. Tato etapa není v současné době úplně dokončena, práce se proto zabývá převážně návrhem řadiče CAN sběrnice, který je pro ni klíčový. Řadič byl napsán v programovacím jazyce VHDL ve vývojovém prostředí XILINX ISE 5.1 a pro testovací účely byl implementován v hradlovém poli Spartan XC2S100 firmy Xilinx. Činnost řadiče byla ověřena připojením na sběrnici spolu se zásuvnou deskou do PC s řadičem SJA1000. Při vývoji programu byly jednotlivé programové bloky zkoušeny v simulačním programu ModelSim v5.6a, jejich spolupráce byla zkoušena přímo v hradlovém poli při komunikaci s reálnou CAN stanicí.

Při návrhu byla snaha vytvořit řadič svojí činností se co nejvíce blížící chování popsanému v normě, umístění a význam registrů byl volen téměř identický s běžně používaným řadičem SJA 1000 firmy Philips. Rozdíly jsou dány pouze absencí některých registrů a jednodušším mapováním. Řadič SJA1000 používá kvůli zpětné kompatibilitě několik režimů, kdy jsou jeho registry pokaždé mapovány rozdílným způsobem. V navrženém řadiči bylo zvoleno mapování, které je obecné pro všechny formáty přenášené zprávy.

Řadič je navržen, aby byl schopen dekódovat a vysílat všechny druhy zpráv definovaných CAN protokolem, při reálné aplikaci to ale často není nutné. Např. servopohony řady SERVOSTAR 600 nepodporují přenos zpráv pomocí rozšířených rámců, pro sběrnice s jedním Masterem zase rychle ubývá na významu definice aktivních a pasivních chybových stavů CAN řadiče. Nejen v těchto případech je výhodou implementace řadiče v hradlovém poli a možnost úpravy jeho zdrojového kódu.

5. LITERATURA

[1] *Servostar 600* [online].
Poslední revize 05/2003 [cit. 1.10.2003]
⟨http://www.danahermotion.de/englisch/dss6.htm⟩.

[2] *Sériová komunikace programovatelných automatů tecomat a regulátorů tecoreg* 8. vydání – říjen 2000
Firemní literatura Teco a.s. dokument číslo TXV 001 06.01

[3] *S7-200 Programmable Controller System Manual* [online].
Poslední revize 04/2002 [cit. 8.3.2002]
⟨http://www.siemens.cz/extra/AD/micro/S7200N_e.pdf ⟩.

[4] *Project: CAN Protocol Controller* [online].
Poslední revize 26.11.2003 [cit. 30.11.2003]
⟨ http://www.opencores.org/projects/can/⟩.

[5] *Průmyslové sítě* [online].
Poslední revize ???? [cit. 10.10.2003]
⟨ http://dce.felk.cvut.cz/hanzalek/_private/DRS/prum_site1.pdf⟩.

[6] *CAN Specification 2.0 Part A* [online].
Poslední revize 21.12.2002 [cit. 15.10.2003]
⟨http://www.can-cia.de/download/specification?268 ⟩.

[7] *CAN Specification 2.0 Part B* [online].
Poslední revize 21.12.2002 [cit. 15.10.2003]
⟨ http://www.can-cia.de/download/specification?269 ⟩.

[8] *CAN Specification 2.0, Addendum* [online].
Poslední revize 21.12.2002 [cit. 15.10.2003]
⟨ http://www.can-cia.de/download/specification?270 ⟩.

[9] *CAN Physical layer* [online].
[cit. 20.10.2003]
⟨ http://dce.felk.cvut.cz/hanzalek/_private/DRS/canphy.pdf ⟩.

[10] *CAN Data link layer* [online].
[cit. 20.10.2003]
⟨ http://dce.felk.cvut.cz/hanzalek/_private/DRS/candll.pdf⟩.

[11] Spartan-II 2.5V FPGA Family: Functional Description [online].
Poslední revize 3.9.2003 [cit. 6.11.2003]
⟨http://www.xilinx.com/bvdocs/publications/ds001_2.pdf ⟩.

[12] Douša, J. *Jazyk VHDL* Praha: Vydavatelství ČVUT, 2003 76 s. ISBN 80-01-02670-1.

[13] Bajer, J.; Hanzálek, Z.; Šusta, R.; *Logické systémy pro řízení* Praha: Vydavatelství ČVUT, 2000 269 s. ISBN 80-01-02147-5.

[14] *SJA1000 Data Sheet* [online].
Poslední revize 4.1.2000 [cit. 12.11.2003]
⟨ http://www-us.semiconductors.philips.com/acrobat/datasheets/SJA1000_3.pdf⟩.

[15] Vedral, J.; Fisher, J.; *Elektronické obvody pro měřící techniku* Praha: Vydavatelství ČVUT, 1999 340 s. ISBN 80-01-01950-0.

[16] *PCA82C250 CAN controller interface* [online].
Poslední revize 13.1.2000 [cit. 20.12.2003]
⟨ http://www.semiconductors.philips.com/acrobat/datasheets/PCA82C250_5.pdf⟩.

6. OBSAH PŘILOŽENÉHO CD

Obsah kořenového adresáře přiloženého CD:

/AGENT	Spustitelná verze spolu se zdrojovými kódy programu AGENT.
/KOMTEST	Spustitelná verze spolu se zdrojovými kódy programu KOMTEST.
/SIMOUT	Spustitelná verze spolu se zdrojovými kódy programu SIMOUT.
/doc	Elektronická podoba diplomové práce
/src	Zde jsou vytvořené zdrojové kódy řadiče CAN
/OXICAN	Spustitelná verze spolu se zdrojovými kódy programu OXICAN.

7. PŘÍLOHY

```
TECO + Simatic Comunication                                  COM1 57600Bd n 8 1
  Prijaty paket
    Start byte
    Byte number 1
    Byte number 2        !!! ZARIZENI NEPRIPOJENO !!!
    Byte number 3        !!!        NEBO          !!!
    Byte number 4        !!!  SPATNE KOMUNIKUJE   !!!
    Csum

                                                Delka paketu: 6
                                                Start byte: CA

  Odeslany paket
 CA  Start byte                Nova hodnota  start  bytu:
 FF  Byte number 1
 00  Byte number 2
 FF  Csum
                                                Delka paketu: 4
                                                Uspesnost  prenosu: 0.00%
 _
                                                                   ESC = Exit
Help: ↔ send/receive; ↑↓ vyber parametru; 0-9 zmena parametru
```

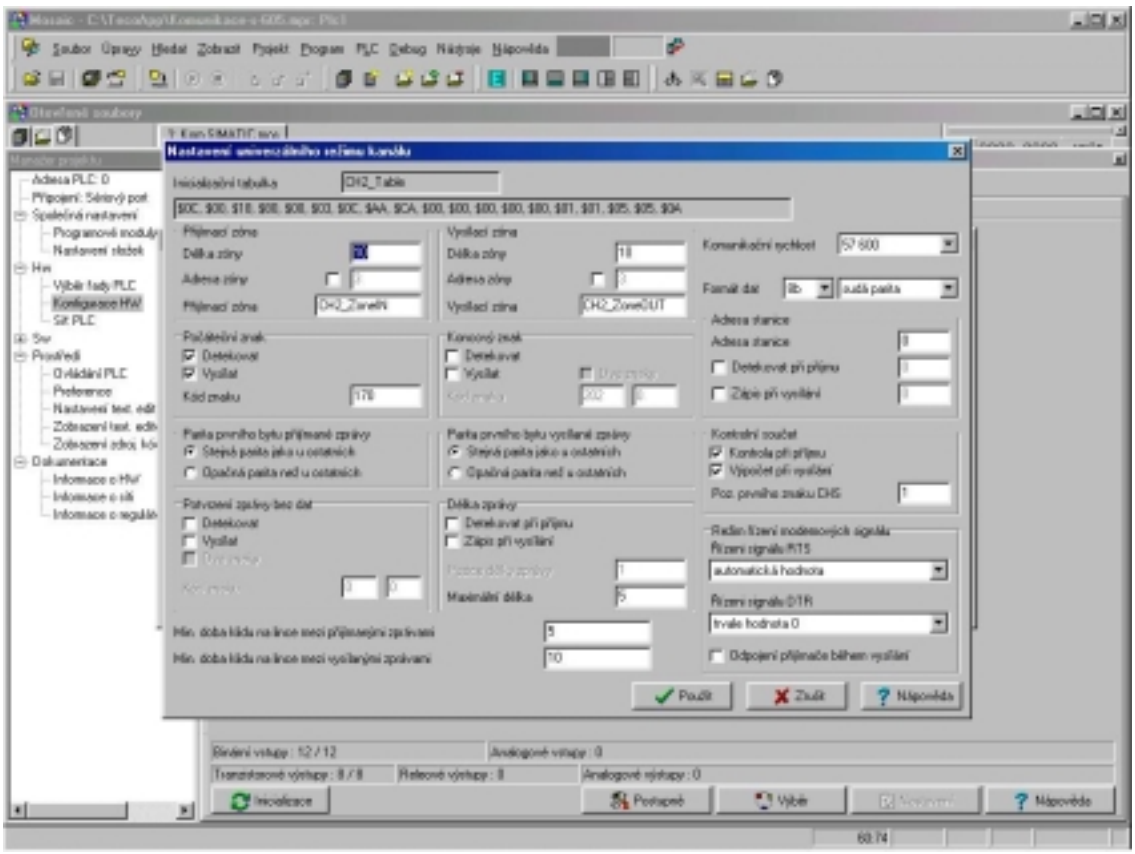
Příloha 1 - obrazovka programu KOMTEST

```
Automat simulation                                      COM 2 57 600Bd 8 1 e
  Prijaty paket
 AA 00 01 0A 0B     Datova cast: 00 01 0A          Csum: 0B Csum OK      _
 Datova binarne: 00000000 10000000 01010000

  0 X/Z                    1 pohyb osy X            0
  0 meridlo vpred          0 pohyb osy Z            1
  0 KAC                    0 konec cyklu            0
  0 LV                     0 PPH                    1
  0 PV                     0 cyklus probiha         0
  0 kryt VB                0 mimo narazky           0
  0 ZPH                    0 10x 20x                0
  0 orovnani               0 novy                   0
                          Pruchody: 492 1
  Odeslany paket
 AA AA BB CC 00 00 31
 Datova cast: AA BB CC 00 00
 Datova cast binarne: 01010101 11011101 00110011 00000000 00000000
                      ↑
 Nazvy bitu:
 prvni
                Cekam po dobu: 2
                                                                   ESC = Exit
Help: ↔ Home End: pohyb kurzoru;  Enter: alternace bitu
```

Příloha 2 - obrazovka programu SIMAUT

```
Serial comunication monitor
NX5031
Received paket in HEX: 41 0  a  4b
1 X/Z                    . osaX move              .
. meridlo vprd           . osaZ move              1
. KAC                    . urychl X               .
. leva  uvr              . BV vpredu              1
. prava uvr              . AC probiha             .
. maucta                 . 0/1 zap/podel          .
1 BV vzadu               . 10x/20x                .
. orovnavani             .                        ._

PLC
Received paket in HEX: 0  0  41 0  a  4b
. meridlo vpr  . sane vpredu  1 predpol BV   . 10x/20x      .
. zap stolu    . POB          . zadnpol BV   . rychl vzad   1
. orov rychl   . vnitrni brou . stul vlevo   . leva  naraz  .
. urychl Z     . HLP vpred    . stul vpravo  . prava naraz  1
. X/Z          . HLP vzad     . blok RK      . zvednuti nar .
. maucta       . HLP vlevo    . 1. imp       .              .
. aut. orovnan . HLP vpravo   1 2. imp       .              .
. sane v AC    . rychl vpred  . 3. imp       .              .
```

Příloha 3 - obrazovka programu AGENT



Příloha 4 - Formulář pro nastavení komunikačních parametrů