ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA APLIKOVANÉ ELEKTRONIKY A TELEKOMUNIKACÍ

DIPLOMOVÁ PRÁCE

Jan Hutr

2013

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jan HUTR**
Osobní číslo:	**E12N0087P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Dopravní elektroinženýrství a autoelektronika**
Název tématu:	**Monitor průmyslové sběrnice AIBus-2**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Prostudujte protokol průmyslové sběrnice AIBus-2.
2. Vytvořte pro Windows řídicí a monitorovací aplikaci umožňující sledování komunikace na průmyslové sběrnici AIBus-2.
3. V aplikaci implementujte logování průběhu komunikace na sběrnici do souborů a dále parametrizovatelnou filtraci dat na sběrnici.
4. Realizaci v práci velmi podrobně popište.

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

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

Vedoucí diplomové práce:	**Doc. Ing. Martin Poupa, Ph.D.** Katedra aplikované elektroniky a telekomunikací
Konzultant diplomové práce:	**Ing. Jan Krpálek** TEDIA s.r.o. Plzeň
Datum zadání diplomové práce:	**15. října 2012**
Termín odevzdání diplomové práce:	**9. května 2013**

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

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 15. října 2012

Abstrakt

Cílem práce je navrhnout software pro monitorování a ukládání zpráv přenášených pomocí protokolu AIBus-2. Dále navrhnout aplikaci pro PC, která umožní jejich prohlížení a další zpracování.

Klíčová slova

AIBus-2, RS-485, VHDL, CRC, WatchDog, Master, Slave

Abstract

The aim of this diploma thesis is to design software for monitoring and storing messages transmitted using the protocol AIBus-2 and to design an application for the PC that allows viewing and further processing of the stored data.

Key Words

AIBus-2, RS-485, VHDL, CRC, WatchDog, Master, Slave

Prohlášení:

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

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

…….

V Plzni dne 9.5.2013 Jan Hutr

Poděkování:

Rád bych poděkoval Ing. Janu Krpálkovi, za příležitost účastnit se zajímavé práce a podporu. Dále pak vedoucímu práce Doc. Ing. Martinu Poupovi, Ph.D..

Obsah

1. Úvod

Cílem této práce je navrhnout řešení pro modul společnosti TEDIA® spol. s r. o. takové, aby bylo možné zaznamenávat a vyhodnocovat komunikaci probíhající prostřednictvím jejich protokolu AIBus-2. Modul by měl sledovat provoz na sběrnici a v případě, že se v komunikaci vyskytne chyba, tuto chybu identifikovat a upozornit na ni. Snahou je zaměřit se zejména na kontrolu ochranných prvků protokolu jako je parita a CRC kódování, ale taky na časování sběrnice. Výstupní hodnoty budou ukládány na paměťová médium pomocí standardních modulů společnosti TEDIA® spol. s r. o.. Odtud budou přenesena do počítačové aplikace, kde budou data zobrazena a vyhodnocena.

2. Protokol AIBus-2

Protokol AIBus-2 je navržen s ohledem na jednoduchou a účelnou komunikaci na střední vzdálenost okolo 1 km a přenosovou rychlostí řádově x1 MB/ s [3]. Topologie sítě odpovídá modelu Master – Slave. Bezpečnost komunikace zajišťují paritní bity určené směrem komunikace, 16 bitové CRC, pevná délka zprávy, zpětné potvrzení platnosti výzvy spolu s dalšími diagnostickými informacemi a robustní konstrukce fyzické vrstvy. V následující části bude popsána struktura protokolu v modelu ISO-OSI a způsob komunikace v konfiguraci Master – Slave

.

2.1. Model ISO-OSI

Mezinárodní organizací pro standardizaci ISO (International Organization for Standardization) byl roku 1984 vypracován referenční model ISO-OSI (Open System Interconnection) a přijat jako mezinárodní norma ISO 7498 [2]. Jednalo se o snahu sjednotit systém komunikace napříč spektrem výrobců. Model je složen ze sedmi vrstev, které je možné vidět v tabulce níže.

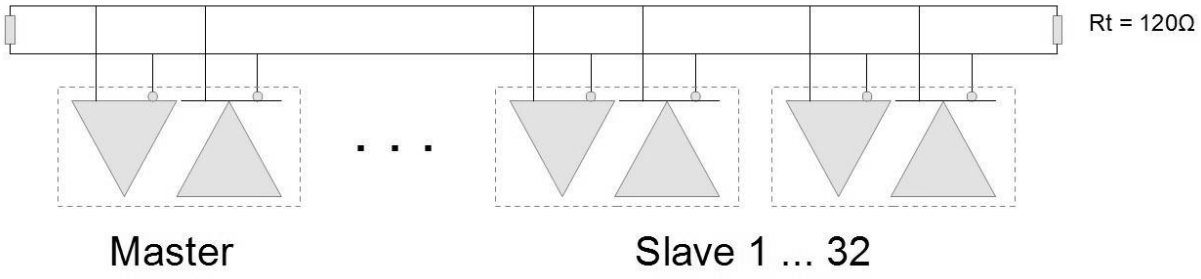
Tabulka 1: Model ISO-OSI

vrstva	popis
fyzická vrstva	Definice fyzických vlastností sběrnice, napěťových úrovní, konektorů a jiné
linková vrstva	Uspořádání dat do rámců, detekce chyb
síťová vrstva	Definice směrování a adresování paketů
transportní vrstva	Zajištění bezpečnosti přenosu dat
relační vrstva	Organizace a synchronizace rozhraní sítě
prezentační vrstva	Úprava do jednotné datové struktury
aplikační vrstva	Zpřístupnění systému přenosu aplikaci

2.2. Pozice AIBus-2 v modelu ISO-OSI

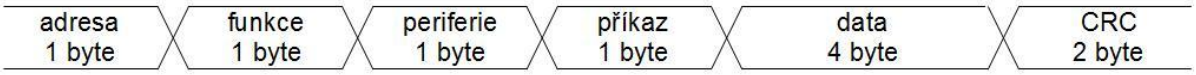
Protokol AIBus-2 zaujímá v modelu ISO-OSI linkovou a částečně fyzickou vrstvu. Ostatní vrstvy nejsou protokolem definovány [1].

Fyzická vrstva protokolu AIBus-2 vychází z vlastností sběrnice RS-485, tedy topologie sítě Master-Slave. Topologie Multimaster je v zásadě možná, ale protokol ji nezmiňuje. Sběrnice je buzena budiči s diferenciálními vstupy pro přijímač a výstupy pro vysílač. Data jsou přenášena po kroucené dvoulince, která při správné terminaci dosahuje dobré spolehlivosti a kvality přenosu. Využití diferenciálních vstupů a výstupu umožňuje nezávislost na zemním potenciálu a dobrou odolnost proti vnějšímu rušení. Podle [3] je možné připojit na sběrnici až 32 vysílačů, přijímačů, nebo jejich kombinací tak, jak je vidět na obrázku níže. Terminace sběrnice je zde znázorněna 120Ω odpory. Další možnosti terminace nejsou v této práci zmíněny.

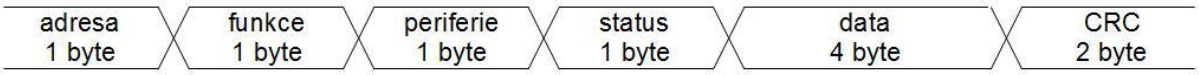


Obrázek 1: Blokové schéma sběrnice RS-485

Linková vrstva protokolu AIBus-2 v modelu ISO-OSI popisuje složení paketů, které se liší pouze podle toho, zda se jedná o výzvu (směr Master -> Slave), nebo odpověď (směr Slave -> Master). Složení paketu je možné vidět na obrázku dále. Jednotlivé byty jsou dále doplněny o start bit, paritní bit a stop bit. Pro výzvu je počítána parita sudá, pro opověď pak lichá. Z hlediska linkové vrstvy je možné připojit k Master až 253 jednotek Slave. Omezení je dáno dostupným rozsahem 255 adres. Z toho jsou dále adresy „0“ a „255“ vyhrazeny.



Obrázek 2: Složení paketu ve směru výzvy



Obrázek 3: Složení paketu ve směru odpovědi

Tabulka 2: Popis jednotlivých částí paketu. (Tabulka převzata z [1, st 1-7])

položka	popis
adresa	Identifikace koncového zařízení (Slave). Dvě adresy jsou vyčleněny: „0“ pro inicializaci jednotky a „255“ pro předávání společných příkazů. Při odpovědi udává odesílatele.
funkce	Udává operaci, která bude provedena s daty na příslušném periferním obvodu. Při odpovědi má totožnou hodnotu.
periferie	Parametr specifikující funkční obvod jednotky Slave, kterému přísluší funkce a data. Při odpovědi má totožnou hodnotu.
příkaz	Ovládání vyhrazených funkcí bez ohledu na prováděnou funkci. Pouze u výzvy, u odpovědi nahrazen statusem.
status	Přenos významných informací bez ohledu na prováděnou funkci, nebo periferii. Např.: chybové stavy jednotky. Pouze u odpovědi.
data	Přenos vlastní informace pro funkci a periferii. Podle zvolené funkce může být obsah nepodstatný v obou směrech komunikace.
CRC16	Ochranný kód určení k zabezpečení předchozích znaků zprávy v obou směrech komunikace.

2.3. Přímý příkaz, status registr a CRC

Přímý příkaz je přenášen v každé zprávě a slouží k ovládání prioritních funkcí jednotky[1]. Jeho struktura je vidět na obrázku dole.

Tabulka 3: Přímý příkaz (tabulka převzata z [1, str. 1-8])

7	6	5	4	3	2	1	0
rezerva	rezerva	rezerva	rezerva	rezerva	rezerva	rezerva	SMPL

Tabulka 4: Definice přímého příkazu (tabulka převzata z [1, str. 1-8])

položka	popis
rezerva	Rezerva pro budoucí využití
SMPL	Příkaz k zavzorkování vstupní jednotky. (určeno pro synchronní záznam více vstupních hodnot)

Status registr navrací základní chybové a provozní informace. Těchto nejdůležitějších 8 bitů je výňatek z 32bitového registru a je přenášen v každé odpovědi.[1] Struktura registru je vidět na obrázku dole.

Tabulka 5: Struktura status registru (tabulka převzata z [1, str 1-9])

7	6	5	4	3	2	1	0
rezerva	ERR24	RST	PWR	rezerva	PRF	FN	SMPL

Tabulka 6: Definice status registru (tabulka převzata z [1, str 1-9])

položka	popis
rezerva	Rezerva pro budoucí využití
ERR24	Alespoň jeden příznak z 24 bitového registru je aktivní.
RST	Stav po signálu reset
PWR	Stav po výpadku napájecího napětí
PRF	Funkce požadovala obsluhu neexistující periferie
FN	Požadovaná funkce nebyla provedena
SMPL	Signalizace zavzorkovaných dat.

CRC16 (Cyclic redundancy check), neboli cyklický redundantní součet je metoda, díky níž je možné odhalit chybu vzniklou při přenosu dat způsobenou rušením komunikace, nebo jiným zásahem. Důležitý je generující polynom, využívaný při výpočtu. Jeho hodnota a velikost přímo určuje pravděpodobnost odhalení chyby. To znamená, že čím delší paket, tím delší by měl být i CRC generující polynom. V případě protokolu AIBus-2 byl zvolen generující polynom CRC16: $A001_H$. Příklad výpočtu je uveden v příloze dokumentu [1].

2.4. Časování sběrnice

Podle [1] je možné měřit následující časové intervaly:

- prodleva mezi jednotlivými znaky výzvy a odpovědi
- prodleva mezi dokončením výzvy a začátkem odpovědi
- prodleva mezi dokončením odpovědi a schopností jednotky přijímat nová data

Prodleva mezi jednotlivými znaky určuje kontinuitu přenosu. Je to časový interval mezi ukončením přenosu stop bitem a začátkem přenosu start bitem mezi jednotlivými slovy. Za chybu se podle [1, str 1-9] považuje doba potřebná pro přenos dvou bitů. Tato hodnota je shodná jak u jednotky Master, tak u jednotky Slave.

Prodleva mezi dokončením výzvy a začátkem odpovědi je čas potřebný pro zpracování dat jednotkou a přípravu odpovědi. Tato hodnota se podle [1, str 1-9] může pohybovat v rozmezí 1 až 20 ms v závislosti na použité jednotce a obsluhovaných periferiích a je uváděna v návodech jednotlivých jednotek.

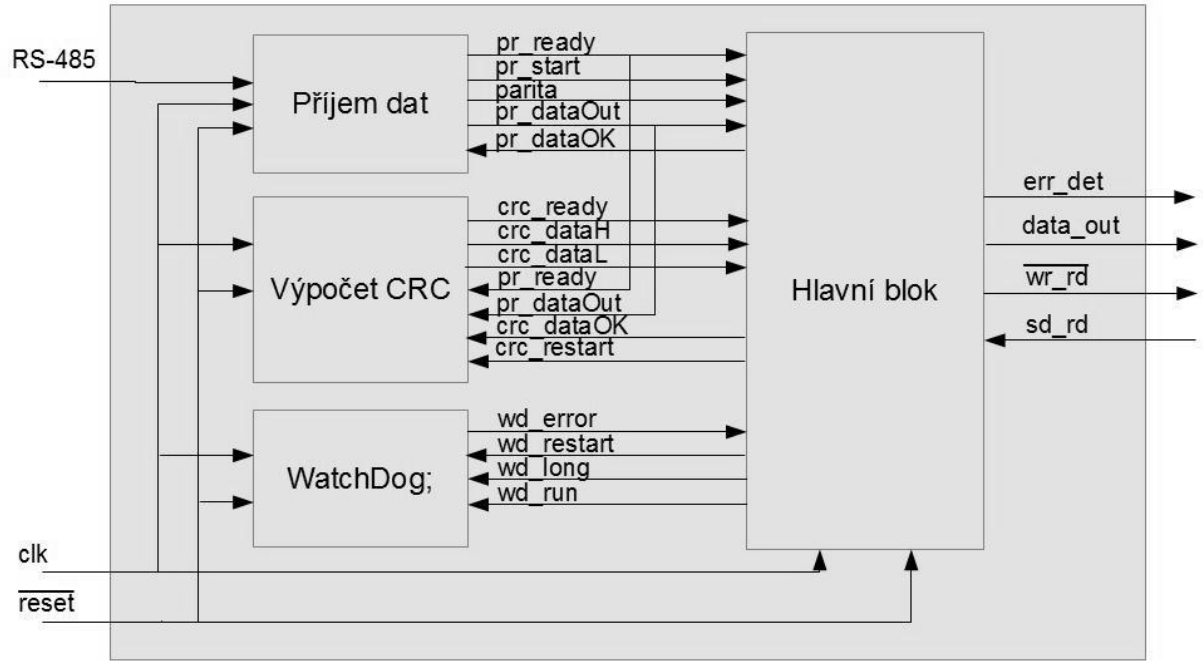
Prodleva mezi dokončením odpovědi a schopností jednotky přijímat nová data tato hodnota je závislá na jednotce Slave a její schopnosti zotavit se. Ta se podle [1, str 1-10] pohybuje u naprosté většiny jednotek kolem 0 a je tedy zanedbávána.

3. Jednotka logování dat

Původní myšlenka, logovat průběh na PC v reálném čase a následně ukládat hodnoty do souboru, se ukázala pro společnost TEDIA® spol. s r. o. jako nešťastná. Vhodnějším řešení je přesunout logování do jednoduššího modulu, který bude data nezávisle zpracovávat. Až v případě odhalení chyby budou data zobrazena na PC. Tak je možné sbírat data dlouhodobě a zároveň snížit nároky na spotřebu energie i prostor. Celá hardwarová část bude realizována standardními moduly společnosti TEDIA® spol. s r. o.. Program je napsán v jazyce VHDL.

3.1. Blokové schéma

Na blokovém schématu níže, jsou vidět vstupní a výstupní signály. Jednotlivé bloky odpovídají komponentám, které takto spojené tvoří entitu. Vstupními signály entity jsou *reset, clk, RS-485* a *wr_rd*. Výstupní signály jsou *err_det, sd_rd* a *data_out*. Popis jednotlivých signálů a komponent následuje níže.



Obrázek 4: Blokové schéma navrženého obvodu ve VHDL

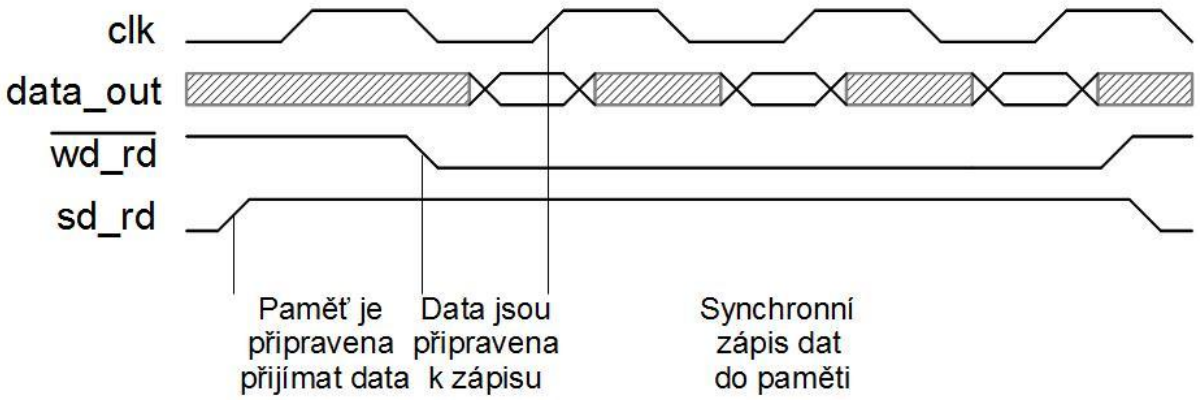
Hodinový signál **clk** je distribuován mezi jednotlivými architekturami a je zároveň použit jako vzorkovací kmitočet. Modulační rychlost je dána 115 200 Bd. Každý bit je vzorkován 5 krát, to znamená, že fázový závěs hodinového signálu je nastaven na frekvenci 576 kHz. Změně této hodnoty se věnuje část popisující architekturu *Příjem dat*. Tento hodinový signál musí být zaveden také do paměti FIFO, jako hodinový signál pro synchronní zápis dat.

Signál **reset** je tak, jako signál *clk*, distribuován do všech bloků architektury. Jeho úkolem je přivést zařízení do pohotovostního stavu. Je nutné, aby reset proběhl po každém startu.

Signál **RS-485** je připojen k driveru sběrnice. Diferenciální výstupy driveru jsou permanentně nastaveny na vysokou impedanci. Není tedy třeba jejich ovládání a v návrhu nejsou uvažovány.

Signál **err_det** je nastaven do „1“, pokud je detekována chyba. Umožňuje tak upozornit na nestandardní chování na sběrnici a odstartovat vyčítání paměti FIFO do souboru.

Funkce signálů **sd_rd** a **wr_rd** je patrná z obrázku dole. Společně tvoří synchronní ovládání paměti FIFO. Signál *sd_rd* nemusí být využit, pak je nutné jej připojit na permanentní „1“.



Obrázek 5: Zápis dat do paměti FIFO

Data_out je signálový vektor 8 bitů, tedy byte. Celkem je v každé zprávě odesláno 22 bytů. První byte udává, zda se jedná o stavové pole, nebo o získaná data a vzájemnou příslušnost. Následuje 10 datových bytů. Po nich opět určující byte a 10 stavových bytů. Bližší vysvětlení v popisu *Hlavního bloku*.

3.2. Architektura Příjem dat

Architektura Příjem dat vzorkuje prostřednictvím driveru stav na sběrnici. Vzorkovací frekvence je dána hodinovým signálem. Ten má takovou frekvenci, aby při modulační rychlosti 115 200 Bd odebral právě 5 vzorků. V tomto bloku jsou data upravována do jednotlivých bytů a je zde počítána parita. Po detekci start bitu je signál *pr_start* nastaven na „1“ a spustí se příjem bytu. Architektura detekuje příchozí bit 3 možnými způsoby:

- změnou stavu před očekávaným okamžikem – v takovém případě je bit zapsán a architektura se synchronizuje se sběrnicí
- vypršením stanovené doby – taková možnost nastává, pokud je přenášeno více stejných bitů. Bit je zapsán a architektura vyčkává na změnu stavu, aby bylo možné ji synchronizovat.
- změnou po očekávané chvíli – protože bit už je zapsaný, dojde pouze k synchronizaci

Parita je určena spočítáním liché parity, bez ohledu na směr přenosu (výzva má paritu sudou a naopak), včetně paritního bitu. Výsledkem je tedy „0“, pokud je parita sudá a data jsou v pořádku, bez ohledu na původní hodnotu paritního bitu. Pokud je očekávána sudá parita a výsledkem je „1“, došlo při přenosu k chybě.

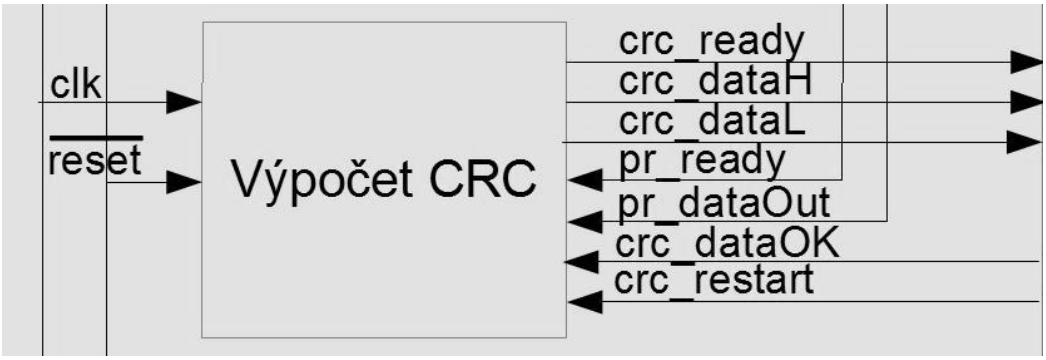
Pokud je byte kompletní, je signál *start* nastaven na „0“ a signál *pr_ready* na „1“. Tak zůstane, dokud není vynulován vstupním signálem *pr_dataOK*. Po dokončení následujícího přenosu jsou data přepsána bez ohledu na to, zda byla přečtena nebo ne. Protože ale na přečtení je poskytnut čas 55 hodinových cyklů a čtení přijatých dat má vysokou prioritu, není pravděpodobné, že by taková situace nastala. Pokud je sběrnice v klidovém stavu (trvalá „1“) a výstupní data jsou přečtená, jsou *pr_start* i *pr_ready* nastaveny na „0“.

Obrázek 6: Architektura Příjem dat

3.3. Architektura Výpočet CRC

Po detekování příchodu nových dat je výpočet resetován pomocí signálu *crc_restart*. Tím je blok výpočtu uveden do klidového stavu. Jakmile je signál *pr_ready* jsou z vektoru *pr_dataOut* přečtena data a vložena do vnitřního registru. Následně proběhne výpočet tak, jak je uvedeno v příloze [1]. Vstupní data jsou exkluzivně sečtena s hodnotou $FFFF_H$. Poté je provedena rotace vpravo a vygenerován příznak. Pokud je příznak nulový, je provedena další rotace vpravo, dokud není příznak roven „1“. Následně je výsledek exkluzivně sečten s generujícím polynomem ve tvaru $A001_H$ a opět se opakuje rotace vpravo, dokud nebude příznak roven „1“. Tato část je vykonávána cyklicky, dokud není vygenerováno 8 příznaků. Výsledkem operace je obsah registru původních vstupních dat.

Výsledná data jsou rozdělena do výstupních vektorů *crc_dataL* pro 0 až 7 bit registru a *crc_dataH* pro 8 až 15 bit. Také je nastaven signál *crc_ready* na hodnotu „1“. Ten je nulován po restartování, nebo po příchodu nových dat. Signál *pr_dataOK* je pozůstatkem starší verze, který nebyl odstraněn.



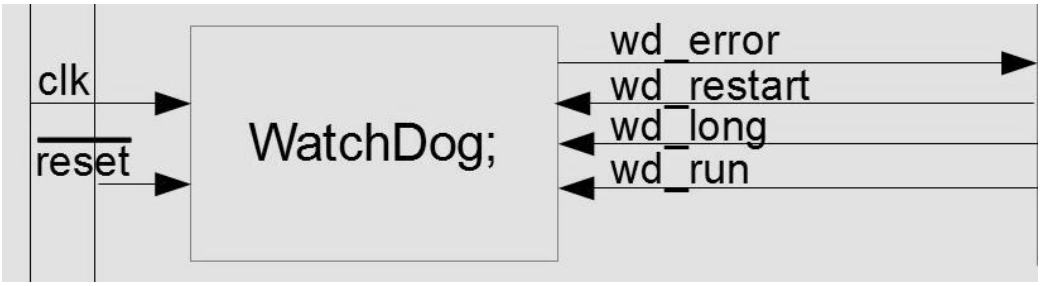
Obrázek 7: Architektura Výpočet CRC

3.4. Architektura WatchDog

V této část je podle nastavení signálu *wd_long* sledována doba, která uplyne od příchodu prvního start bitu do dokončení zprávy, nebo od ukončení přenosu do nového zahájení. Na začátku je přenosu je architektura po resetu. Po nastavení *wd_run* je aktivován čítač řízený hodinovým signálem *clk*. Stav čítače je kontrolován s pevně stanovenou hodnotou, určenou signálem *wd_long*. Je-li nastaven na „1“, pak je hodnota čítače porovnávána s číslem 11520,

což odpovídá zpoždění 20 ms. V opačném případě je nastaven na hodnotu 560, což odpovídá přenesené zprávě sed zpožděním 2 bitů.

Pokud je čas překročen, je čítač zastaven a signál *wd_error* nastaven na „1“. Tak zůstane, dokud nebude nastaven signál *wd_restart*.



Obrázek 8: Architektura WatchDog

3.5. Architektura Hlavní blok

Tento blok je nejdůležitější části celé entity. Zajištuje sběr dat z okolních architektur, jejich zpracování a vyhodnocení. Po resetu se blok nachází ve fázi *stand_by* a vyčkává na příchod start bitu. Ten je deklarován nastavením signálu *pr_start* na „1“ blokem Příjem dat. Následuje nastavení signálu *wd_long* na „0“ a signálů *wd_restart, wd_run* a *crc_restart* na „1“. Restartovací signály jsou při další vzestupné hraně hodinového signálu *clk* vynulovány. Fáze je nastavena na *call*.

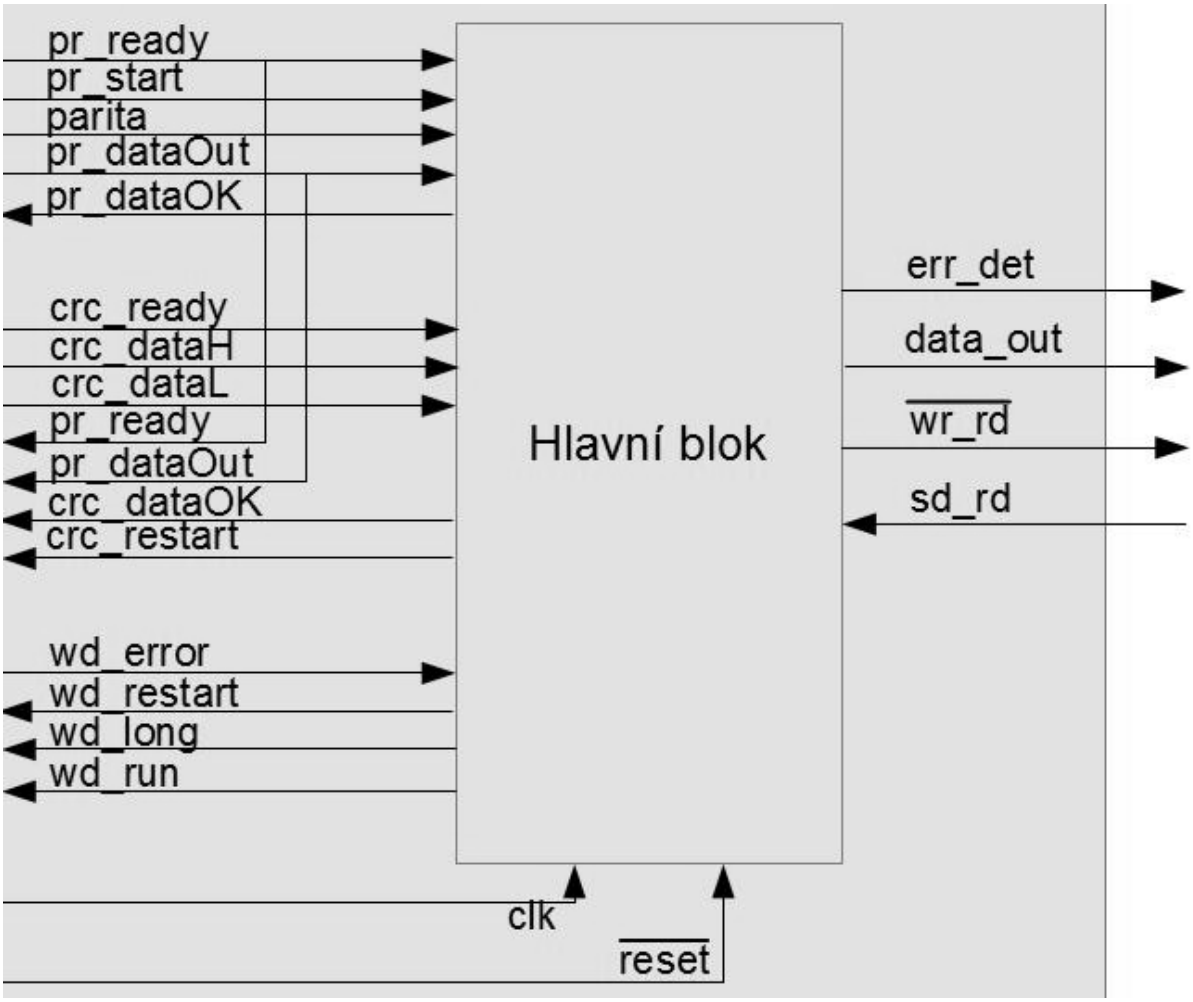
Ve fázi *call* jsou počítány příchozí byty. Nejprve je zkontrolována parita, poté jsou data uložená do zásobníku příchozích dat. Pokud je identifikována špatná parita, není příjem přerušen. Tato událost je pouze zaznamenána do status zásobníku. V opačném případě je uložena informace o bezchybných datech. Po příchodu 8 a tedy posledního datového bytu, je očekáváno spočtení CRC. To se projeví nastavením signálu *crc_ready* na „1“. Data z vektorů *crc_dataL* a *crc_dataH* jsou uložená do registrů a fáze je nastavena na *waiting1*.

V této fázi je očekáván příchod CRC bytů z bloku Příjem dat. Poté jsou bloky Výpočet CRC a WatchDog resetovány tak, jako ve fázi *stand_by*. Watchdog je dále nastaven na delší čas pro zpracování odpovědi nastavením signálu *wd_long* na „1“. Tím je definován časový interval pro odpověď na 20 ms. Tento čas by měl stačit i nejpomalejším jednotkám na odeslání

odpovědi. Zároveň je vyvolána funkce *clean* pro přípravu přijatých dat k odeslání do paměti FIFO. To znamená, že data se seřadí do pořadí, v jakém byla přijata, než byla uložena do kruhového zásobníku. Dále je nastavena fáze *waiting2*.

Toto je poslední fáze, ve které se pracuje s daty výzvy Master -> Slave. Jsou zde porovnány spočítané CRC byty s těmi přijatými. Výsledek je uložen do status vektoru. Počet přijatých bytů je vynulován a je povoleno odeslat data. Fáze je nastavena na *answer*.

Příjem odpovědi je z velké části stejný jako příjem výzvy. Je očekáván příchod 8 bytů, poté je ve fázi *waiting3* vypočítáno CRC. To je porovnáno s přijatými daty a data jsou odeslána do paměti. Bloky WatchDog a Výpočet CRC jsou resetovány a WatchDog je navíc zastaven nastavením signálu *wd_run* na „0“. Fáze *waiting4* se vrací do *stand_by*.



Obrázek 9: Architektura Hlavní blok

3.1. Detekce, zpracování a interpretace chyby

Jak už bylo zmíněno v předchozí kapitole, Hlavní blok je vybaven dvěma kruhovými zásobníky, do kterých se ukládají přijatá data a jejich status. Status může být následující:

Tabulka 7: Status přijatého bytu

status	číselná reprezentace	význam
MS_dataOK	$C1_H$	Tento byte směru M–S je v pořádku
MS_CRCErr	31_H	Tento byte ve směru M-S neodpovídal spočtenému CRC
MS_WDErr	11_H	V tomto bytu došlo k přetečení času ve směru M-S.
MS_PaErr	21_H	Parita tohoto bytu ve směru M-S neodpovídala
SM_dataOK	$C3_H$	Tento byte směru S-M je v pořádku
SM_CRCErr	33_H	Tento byte ve směru S-M neodpovídal spočtenému CRC
SM_WDErr	13_H	V tomto bytu došlo k přetečení času ve směru S-M.
SM_PaErr	23_H	Parita tohoto bytu ve směru S-M neodpovídala

Z předchozího je patrné, že systém dokáže detekovat 3 základní chyby a to:

- chyba parity
- chyba CRC
- překročení vyhrazeného času

První chybou, která může být odhalena, je chyba parity. Jak už bylo napsáno dříve, u přijatého bytu je počítána lichá parita, včetně paritního bitu. To znamená „0“ na výstupu v případě správně určené sudé parity a naopak „1“ v případě parity liché. Pokud je tedy v případě výzvy spočítána parita „1“, jedná se zjevně o chybu. Zde není sledování ukončeno, ale tento fakt je zaznamenán ve formě status bytu, je aktivován chybový výstup *err_det* a je na něho upozorněno při zpracování v PC.

Další možností je chyba CRC. Může se jednat o více násobnou chybu, která nebyla paritou detekována, ale i naopak. V takovém případě bude patrné, v kterém bytu chyba nastala a v jakém směru. Příklad takové chyby je znázorněn v tabulce dále. Je možné vidět, že chyba nastala v 7. bytu a ze stejného důvodu si neodpovídají ani CRC byty.

Tabulka 8: Příklad získaných dat a jejich statusu

bit	**9**	**8**	**7**	**6**	**5**
data	$A1_H$	$A2_H$	$A3_H$	$A4_H$	$A5_H$
status	MS_dataOK	MS_dataOK	MS_PaErr	MS_dataOK	MS_dataOK
bit	**4**	**3**	**2**	**1**	**0**
data	$A6_H$	$A7_H$	$A8_H$	$A9_H$	AA_H
status	MS_dataOK	MS_dataOK	MS_dataOK	MS_CRCErr	MS_CRCErr

Poslední možností je překročení vyhrazeného času. To způsobí ukončení sledování komunikace a resetování celého systému. Chyba je zaznamenána pomocí patřičného statusu a následující data jsou nahrazeny hodnotou FF_H. Výsledkem mohou být zdánlivě nesmyslné chyby, které je ale potřeba správně interpretovat. Pokud například něco způsobí zpoždění odesílatele, nikoli ukončení jeho vysílání, může se taková skutečnost projevit jako 2 zdánlivě nesouvisející chybné zprávy. Jako příklad uvádím následující dvě tabulky.

Je zde vidět, že první chyba nastala už v 7. bytu. Sledování tedy bylo ukončeno a chybějící data byla nahrazena. Systém byl resetován. Pak ale jsou zachycena zbývající data, která ale nejsou očekávána. Jsou mylně považována za opačný směr a je jich méně, než je očekáváno. Znovu tedy vyprší čas a sledování je opět ukončena. Přesto data nejsou ztracena a příčina selhání komunikace je zřejmá.

Tabulka 9: Příliš dlouhá odpověd

bit	**9**	**8**	**7**	**6**	**5**
data	$A1_H$	$A2_H$	$A3_H$	FF_H	FF_H
status	SM_dataOK	SM_dataOK	SM_dataOK	SM_WDErr	FF_H
bit	**4**	**3**	**2**	**1**	**0**
data	FF_H	FF_H	FF_H	FF_H	FF_H
status	FF_H	FF_H	FF_H	FF_H	FF_H

Tabulka 10: Neočekávaná a tedy špatně interpretovaná data

bit	**9**	**8**	**7**	**6**	**5**
data	$A1_H$	$A2_H$	$A3_H$	$A4_H$	$A5_H$
status	MS_PaErr	MS_PaErr	MS_PaErr	MS_PaErr	MS_PaErr
bit	**4**	**3**	**2**	**1**	**0**
data	$A6_H$	FF_H	FF_H	FF_H	FF_H
status	MS_PaErr	MS_WDErr	FF_H	FF_H	FF_H

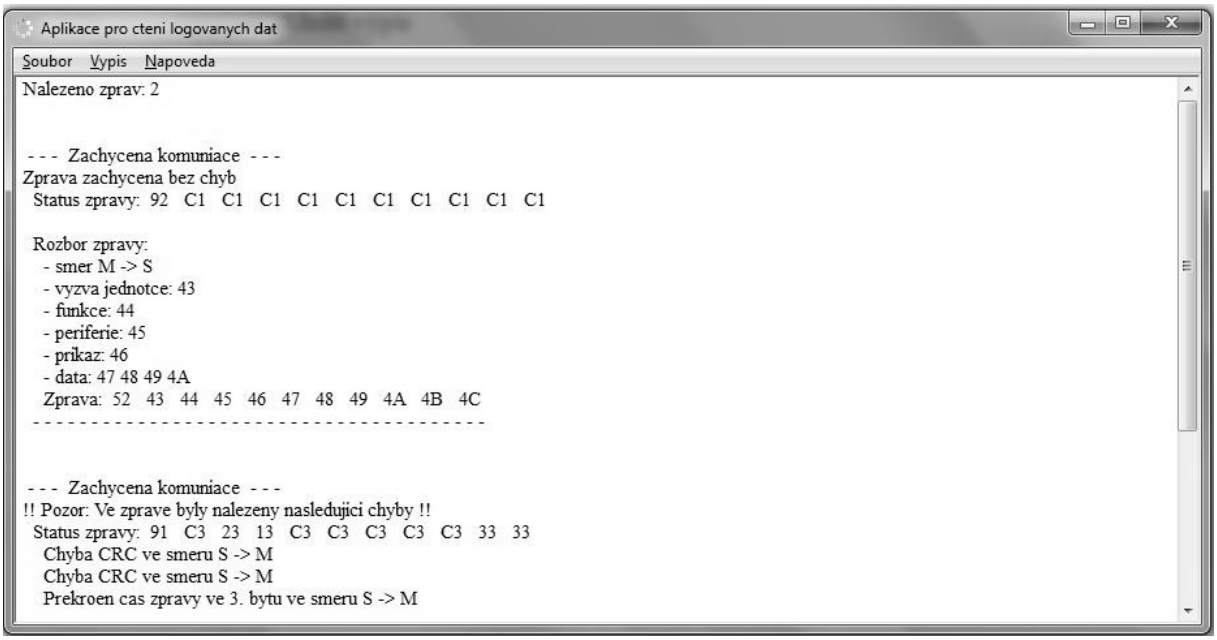
4. Aplikace pro Windows

Z důvodů, objasněných v předchozí kapitole, neprobíhá logování v PC, ale v modulu. Zde jsou data ukládána do paměti typu FIFO a podle potřeby vyčítána do souboru na paměť FLASH. Tento soubor je pak čten aplikací, která synchronizuje a dekóduje data. Aplikace je napsána v programu Delphi7.

4.1. Uživatelské prostředí programu

V základním okně je možné najít prostor pro výpis dat a nabídku funkcí. Ta je rozdělena následovně:

- Soubor
  - Otevřít a zpracovat
  - Otevřít a vypsat
  - Uložit výpis
  - Konec
- Výpis
  - Font
  - Filtrace dat
- Nápověda
  - Master
  - Slave
  - Statut registr
  - Funkce



Obrázek 10: Rozklíčované informace pokusných dat

Položka **Otevřít a zpracovat** otevře soubor s logovanými daty a zahájí synchronizaci. To znamená, že soubor je postupně procházen a je hledán začátek přenosu zprávy, který má určité náležitosti. Už v modulu byla každé datové i status zprávě přidána synchronizační značka. Tyto synchronizační značky musejí být v přesné vzdálenosti od sebe a musejí být shodné jak pro datovou, tak stavovou zprávu.

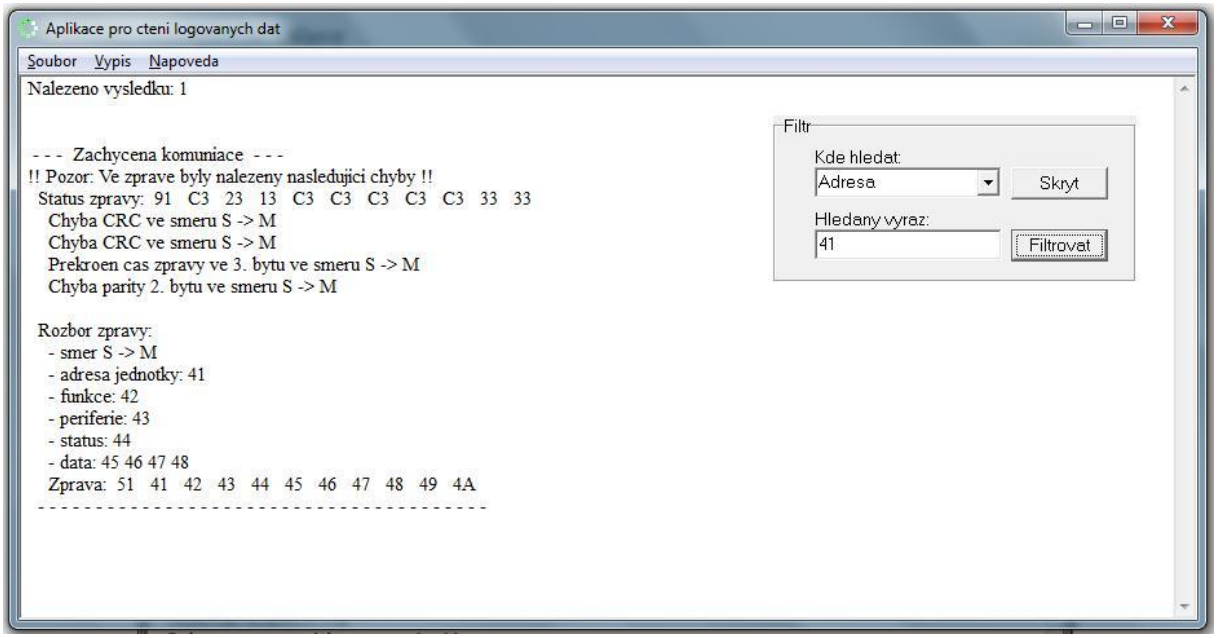
Otevřít a vypsat slouží pro otevírání dříve uložených reportů pomocí funkce **Uložit výpis**. Do reportů je možné dopisovat poznámky a ukládat je. Následné vyvolání již zpracovaných dat může usnadnit práci.

Položka **Konec** ukončí aplikaci.

Pomocí funkce **Font** lze nastavit font výpisu

Filtrace dat otevře panel, na kterém je možné zadat bázi, kde vyhledávání probíhat a také hledaný výraz. Tlačítko Filtrovat pak vyčistí okno s výpis a nahradí jej filtrovaným bsahem.

V položce **Nápověda** je možné najít složení zpráv, status registru a funkcí.



Obrázek 11: Ukázka filtrování dat

5. Závěr:

Přes počáteční nesnáze se podařilo navrhnout funkční program pro programovatelný obvod FPGA Altera Cyclon1. Program je složen z několika samostatných architektur, které si vzájemně vyměňují data a tvoří tak funkční celek. Obvod je schopen sledovat pohyb dat na sběrnici, vyhodnocovat jejich korektnost a ukládat nejen přenesená data, ale i jejich status v rámci zprávy. Pomocí počítačové aplikace je možné si data prostudovat a jasně identifikovat chyby, které během komunikace nastaly. Data jsou jasně prezentovány a je možné je filtrovat, procházet, editovat a ukládat. Na skutečnou aplikaci a testování v reálných podmínkách se však nedošlo.

6. Literatura

[1] TEDIA® spol. s r. o., 2012. AIBus-2 specifikace komunikačního protokolu [online]. Plzeň: TEDIA® spol. s r. o. [cit. 2.9.2012] Dostupné z: http://www.tedia.cz/download/files/aibus2.pdf

[2] Ikaros, redakce. Mezinárodní organizace ISO zvyšuje kvalitu odborné komunikace. Ikaros [online]. 2003, roč. 7, č. 13 [cit. 10.04.2013]. Dostupné z: http://www.ikaros.cz/node/1537. urn:nbn:cz:ik-001537. ISSN 1212-5075.

[3] http://www.hw.cz/teorie-a-praxe/dokumentace/rs-485-422.html [online] [cit.18.4.201]

[4] PINKER, Jiří a POUPA, Martin. *Číslicové systémy a jazyk VHDL*. 1. vyd. Praha: BEN - technická literatura, 2006. 349 s. ISBN 80-7300-198-5.

[5] VAHID, Frank a LYSECKY, Roman. *VHDL for digital design*. Hoboken: John Wiley & Sons, ©2007. xvi, 166 s. ISBN 978-0-470-05263-1.

[6] KRÁL, Jiří. *Řešené příklady ve VHDL: hradlová pole FPGA pro začátečníky*. 1. vyd. Praha: BEN - technická literatura, 2010. 127 s. ISBN 978-80-7300-257-2.

[7] ŠŤASTNÝ, Jakub. *FPGA prakticky: realizace číslicových systémů pro programovatelná hradlová pole*. 1. vyd. Praha: BEN - technická literatura, 2010. 199 s. ISBN 978-80-7300-261-9.