VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

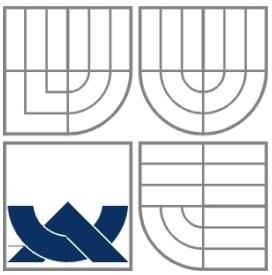
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

BEZPEČNÉ PROPOJENÍ POČÍTAČŮ

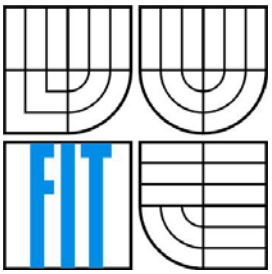
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE JAN WINTER
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

BEZPEČNÉ PROPOJENÍ POČÍTAČŮ

SECURE PC CONNECTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — JAN WINTER
AUTHOR

VEDOUCÍ PRÁCE — ING. JÁN KUBEK
SUPERVISOR

BRNO 2007

Zadání diplomové práce/5370/2006/xwinte00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Winter Jan**

Obor: Výpočetní technika a informatika

Téma: **Bezpečné propojení počítačů**

Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se výukovým kitem FIT, rozhraním USB a RS-485 (plně duplexní průmyslová sběrnice založená na 20mA smyčce).
2. Navrhněte konfiguraci pro FPGA kitu umožňující propojení dvou kitů pomocí RS-485.
3. Navrhněte jednoduchý software umožňující jednoúčelovou komunikaci dvou PC, ke kterým je přes USB připojen výukový kit.
4. Zabezpečte komunikaci po lince RS-485 proti rušení či chybám v přenosu.

Literatura:

- http://www.fit.vutbr.cz/kit

Při obhajobě semestrální části diplomového projektu je požadováno:

- Bez požadavků.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kubek Ján, Ing.**, UPSY FIT VUT

Datum zadání: 1. listopadu 2006

Datum odevzdání: 22. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Jan Winter**
Id studenta: 20500
Bytem: Náchodská 491, 549 32 Velké Poříčí
Narozen: 05. 02. 1981, Náchod
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: Bezpečné propojení počítačů
Vedoucí/školitel VŠKP: Kubek Ján, Ing.
Ústav: Ústav počítačových systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Cílem této diplomové práce je vytvořit sériové komunikační rozhraní pro výukový FITkit. Toto sériové rozhraní má být plně duplexní průmyslová sběrnice založená na 20mA smyčce a má umožňovat propojení dvou kitů pomocí RS-485. Úkolem je navrhnout jednoduchý software umožňující jednoúčelovou komunikaci dvou kitů a zabezpečit komunikaci po lince RS-485 proti rušení či chybám v přenosu.

Klíčová slova

FITkit, sériové rozhraní, sériová komunikace, VHDL, vysílač, přijímač, FPGA

Abstract

Aim of this master's thesis is the creation of serial communication interface for FITkit. This serial interface is a full duplex industrial bus based on 20mA loop circuit and it should allow connection of two FITkits by RS-485. Aim of this paper is to design simple software allowing communication of two FITkits and to secure this communication on RS-485 link against interference or transfer errors.

Keywords

FITkit, serial interface, serial communications, VHDL, transmitter, receiver, FPGA

Citace

Jan Winter: Bezpečné propojení počítačů, diplomová práce, Brno, FIT VUT v Brně, 2007

Bezpečné propojení počítačů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Jána Kubeka.
Další informace mi poskytl Ing. Zdeněk Vašíček.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Winter
22.5.2007

Poděkování

Děkuji Ing. Jánu Kubekovi za odborné vedení, rady a podněty, které mi během práce poskytoval. Dále děkuji Ing. Zdeňku Vašíčkovi za odborné rady při práci s platformou FITkit.

Obsah

1 Úvod

Tématem mé diplomové práce je bezpečné propojení počítačů. Pro realizaci bude použita vyúková platforma FITkit, která je na uvedené téma vhodná. Bude tedy nutné navrhnout konfiguraci pro FPGA umožňující propojení dvou kitů pomocí RS-485 (plně duplexní průmyslová sběrnice založená na 20mA smyčce), zabezpečit komunikaci po lince RS-485 proti rušení či chybám v přenosu a vytvoření převodníku na 20mA smyčku.

V první části se zaměřuji na teoretické znalosti, kdy jsem se seznámil s výukovou platformou FITkit, která byla vytvořena na Fakultě informačních technologii Vysokého učení technického v Brně. V dalších kapitolách se zabývám jednotlivými problémy nutnými pro realizaci propojení. Mezi ně patří například sériové rozhraní RS-232 a jeho modifikace, formát přenosu dat i správná volba kabelu.

Druhá část se věnuje návrhu řešení a následné implementaci sériové komunikace. Zde je detailně popsán návrh a implementace přijímače a vysílače komunikačního rozhraní RS-485, řešení proudové smyčky i způsob zadávání dat a jejich zobrazení po přijetí. Pro implementaci byl zvolen jazyk VHDL, jako celosvětově nejrozšířenější jazyk při návrhu hardwaru.

Cílem této práce je navrhnout způsob propojení dvou počitačů a to při splnění nároků, které jsou kladeny firmami (trhem) a zároveň aby byla aplikovatelnost v praxi a nedocházelo k chybným řešením.

Tato práce nenavazuje na můj ročníkový ani semestrální projekt.

2 FITkit

Platforma FITkit byla vytvořena na Fakultě informačních technologii Vysokého učení technického v Brně (dále jen FIT). Cílem platformy FITkit je navrhovat a prakticky realizovat nejen softwarové, ale také hardwarové projekty či dokonce celé aplikace.

Platforma FITkit umožňuje obsáhnout značnou část spektra znalostí a dovedností, které musí dnešní inženýr – informatik znát, aby byl schopen obstát na globálním trhu práce. Typickým příkladem využití informatiky v praxi jsou tzv. vestavěné systémy (anglicky Embedded Systems), které se v dnešní době uplatňují v běžném životě a jejichž význam ještě výrazně poroste. Jednoduše řečeno se jedná o veškerá zařízení, která v sobě mají nějakým způsobem vestavěn procesor (mobilní telefon, MP3 přehrávač, televizní přijímač atd.).

FITkit obsahuje výkonný mikrokontrolér s nízkým příkonem a řadu periferií. Důležitým aspektem je též využití pokročilého reprogramovatelného hardwaru na bázi hradlových polí FPGA (anglicky Field Programmable Gate Array) jenž lze, podobně jako software na počítači, neomezeně modifikovat pro různé účely dle potřeby – uživatel tedy nemusí znovu vytvářet nový hardware pro každou aplikaci.

Obrázek 1.1.

Při návrhu aplikací se využívá skutečnosti, že vlastnosti hardwaru se v dnešní době převážně popisují vhodným programovacím jazykem (např. VHDL), díky čemuž se návrh softwaru a hardwaru provádí do značné míry obdobně. Software po mikrokontrolér se tvoří v jazyce C a do spustitelné formy se překládá pomocí GNU překladače. Generování programovacích dat pro FPGA z popisu v jazyce VHDL probíhá zcela automaticky pomocí profesionálních návrhových systémů.

Iniciativa je koncipována jako open-source (pro software) a open-core (pro hardware), což znamená, že veškeré výsledky práce s platformou FITkit jsou přístupné na internetu ve zdrojové formě pro kohokoli. Cílem tohoto konceptu je nastartovat synergický efekt, kdy každý, kdo využije výsledky práce vytvořené uživateli platformy FITkit, bude je dále poskytovat ve zdrojové formě a umožní tím jejich použití pro všechny zájemce.

Cílem nasazení platformy na FIT je, aby každý student měl FITkit k dispozici a aby s ním mohl pracovat ve škole, doma i na kolejích. Důležité je, že FITkit je využíván ve výuce řady kurzů a studenta bude provázet po celou dobu studia napříč bakalářským i magisterským studijním programem. Znalosti, získané ve výuce, pak bude moci využít pro tvorbu ročníkových, semestrálních a diplomových prací. Výsledkem by měla být výrazná podpora výuky technologií hardware a

software výpočetních systémů s důrazem na praktické aplikační výstupy s tím, že výsledky budou zpřístupněné a využitelné pro co nejširší okruh zájemců nejen z řad studentů informatiky. [1]

2.1 Součásti platformy FITkit

- FPGA Spartan 3 XC3S50-4PQ208C (Xilinx)
- MCU MSP430F168 (Texas Instruments)
- USB-UART převodník FT2232C
- Audio IN / OUT
- konektory PS2
- konektor VGA
- konektor RS232
- DRAM 8x8mbit
- Klávesnice
- Řádkový LCD displej
- Pinheaders

2.2 FPGA (field programmable gate array)

Obvody typu FPGA jsou polovodičové zařízení obsahující funkční bloky a propojovací sítě. Programovatelné logické komponenty mohou být naprogramované tak, aby fungovaly jako základní logická hradla AND, OR, XOR, NOT, nebo více komplexní kombinační funkce jako je dekodér nebo jednoduché matematické funkce. Ve většině FPGA, tyto programové logické komponenty (nebo logické bloky v jazyce FPGA) také zahrnují paměťové elementy, které mohou být jednoduchými klopnými obvody nebo celými bloky paměti. Pomocí programovacího předpisu lze tyto bloky mezi sebou propojovat a vytvářet tak požadovanou funkci. FPGA se programuje technologií RAM, takže je možné přeprogramovat obvod během okamžiku. Nevýhodou je, že obvod je třeba automaticky programovat po připojení napájení, před startem aplikace. Z jiného pohledu to může být výhodou, protože FPGA (resp. jejich části) lze přeprogramovávat za běhu. To může vést k tomu, že dle konkrétních požadavků daných stavem aplikace a vnějších vstupů se může zařízení samo dle potřeby přeprogramovat. Tento trend přenáší rovinu řízení obecného procesoru softwarem do hardware, kdy vlastně jakýsi software (rozuměj programovací předpis pro FPGA), určuje to, co se bude v hardware odehrávat. Aby možností bylo více, i hardware může být (a často bývá) řízen programem ve smyslu software. Tzn. v FPGA je naprogramovaný procesor, specifický pro řízení dané úlohy. K dispozici má program, dle kterého se řídí. Zde je vidět obrovská flexibilita, jednak v tom, že je možné dynamicky měnit program pro řízení procesoru, ale také v tom, že v případě potřeby je možné

nahradit celý procesor. Ať již třeba za jiný optimalizovaný, či za pouhý stavový automat nebo jinou komponentu.

Obvody FPGA řady Spartan-3 od firmy Xilinx mají řešení s nízkou cenou a vysokým výkonem pro aplikace s velkým rozsahem výroby ve spotřební elektronice. Mezi hlavní rysy architektury Spartan-3 patří:

- Nová 90nm technologie výroby
- Možnost až 5 miliónů systémových hradel nebo 74880 logických buněk
- Nastavení hodinového signálů až 326MHz.
- Napájecí zdroje : 1.2V , 2.5V , 3.3V
- Možnost až 784 I/O pinů
- 622Mb/s přenosová rychlost pinu
- Podporuje až 17 úrovňových standardů
- Double Data Rate (DDR) podpora
- Integrovaná struktura pro konstrukci rychlých sčítaček a až 104 integrovaných násobiček
- JTAG port pro testování
- Blokové paměti RAM až 1872Kb a distribuované RAM až 520Kb
- Podporuje 4 Digital Clock Manager (DCM) umožňující frekvenční syntézu
- 8 globálních hodinových linek
- Plně podporuje vývojové prostředí Xilinx ISE

Architektura je složena z 5 základních funkčních/logických bloků.

- IOBs (Input/Output Blocks) řídí tok dat mezi I/O pinem a vnitřní logikou
- CLBs (Configurable Logic Blocks) obsahující LUTs (Look-Up Tables) na principu paměti RAM
- Block RAM umožňující ukládaní dat ve formátu 18Kb dual-port blocks
- násobičky umožňující vynásobení dvou 18-bitových čísel
- DCM (Digital Clock Manager) poskytuje autokalibraci, plně digitální řešení distribuce zpoždění, násobení, dělení a fázový posun hodinového signálu

Konfigurace obvodu FPGA typu Spartan-3 je programován naplněním konfiguračních dat do buněk statické paměti, které řídí všechny funkční a propojovací prvky. Před připojením napájecího napětí jsou konfigurační data uschována externě v paměti PROM nebo v jiném nevolatilním médiu na desce plošného spoje nebo mimo ni. Po připojení napájecího napětí jsou konfigurační data přepsána do FPGA, k čemuž může být použit některý z pěti módů: Master Paralel, Slave Paralel, Master Serial, Slave Serial a Boundary Scan (JTAG). Oba paralelní módy používají osmibitovou bránu SelectMAP™. Pro uložení konfiguračních dat jsou doporučeny levné paměti z rodiny Xilinx

Platform Flash PROM, zahrnující sériové konfigurační paměti XCF00S pro sériovou konfiguraci a paměti s větší hustotou z rodiny XCF00P pro paralelní nebo sériovou konfiguraci.

Možnosti vývodů I/O obvodů řady Spartan-3 dovolují použít 18 nesymetrických standardů a 8 diferenčních standardů. Mnoho standardů podporuje rys DCI, který využívá integrované zakončení k eliminaci nežádoucích odrazů signálu. [6]

Všechny uvedené parametry použitého FPGA mi k diplomové práci zcela vyhovují. Některé možnosti čipu, které poskytuje, plně nevyužiji.

2.3 Mikrokontrolér

Mikrokontrolér či mikropočítač [angl. microcontroller] je elektronická součástka nejčastěji v podobě integrovaného obvodu (čipu). Tuto součástku lze naprogramovat, aby prováděla určenou úlohu v elektronickém zapojení. Velké množství výrobců vyrábí různé typy mikropočítačů. S mikropočítači se dnes již setkáte v téměř každém elektronickém zařízení (např. v televizorech, mobilních telefonech, mp3 přehrávačích, automobilech, atd.).

Výhodou použití mikropočítačů je, že dokáží nahradit velké a drahé elektronické zapojení složené např. z diskrétních součástek a jednoduchých integrovaných obvodů. Mikropočítač sice v sobě ukrývá podstatně složitější elektronické zapojení, ale díky tomu, že se mikropočítače vyrábí ve velkém množství, se podařilo snížit jejich výrobní cenu na přijatelnou úroveň.

Mikropočítač je někdy nesprávně označován za procesor nebo mikroprocesor [angl. processor, microprocessor]. Procesor je uvnitř mikropočítače a podílí se na výpočtech. Mikropočítač na rozdíl od procesoru je plně soběstačná jednotka, obsahující množství dalších obvodů (např. programovou paměť, porty, atd.).

Mikrokotroler MSP430F168 z řady MSP430 má nízkou spotřebu a řadu nástrojů, které mají různé periferie zaměřené na odlišné aplikace. Architektura, která kombinuje pět modů nízké spotřeby, je optimalizována, aby prodloužila životnost baterie přenosných zařízení. Mikrokontroler obsahuje 16-bitový RISC procesor, 16-bitové registry a generátory konstant, které jsou navržené pro nejvyšší efektivitu kódu. Digitálně ovládaný oscilátor umožňuje přebuzení módu s nízkou spotřebou do aktivního módu za méně než 6 sekund. Mikrokontroler je konfigurovatelný dvěma stavy 16-bitovými časovači, rychlým 15-bitovým A/D převodníkem a dvojitým 12-bitovým D/A převodníkem, a jedním nebo dvěma univerzálním sériovým synchronním/asynchronním komunikačním rozhraním(USART), I2C, DMA, a 48 vstupy/výstupy.[7]

Tento mikropočítač bude nutné doplnit proudovým D/A převodníkem, který bude umožňovat aplikovat 20mA smyčku.

3 Sériová komunikace

Jako sériový přenos označujeme takový, kdy se signálové prvky téhož datového proudu předávají za sebou - sériově. Naopak při paralelním přenosu se určitý počet signálových prvků (např. bitů) přenáší současně. Při sériové komunikaci existují tři druhy přenosu, a to synchronní, asynchronní a arytmický přenos dat.

Synchronní přenos dat zabezpečuje přenos dat konstantní přenosovou rychlostí, neboť po vyslání synchronizačních znaků jsou data vysílána nepřetržitě, a proto se používá tehdy, kdy je možno zabezpečit přípravu dat zdrojem dat s odpovídající přenosovou rychlostí. Přenosová rychlost při synchronním přenosu dat musí být volena s ohledem na nejpomalejší zařízení na straně přijímače dat.

Při asynchronním přenosu dat se synchronizace mezi vysílačem a přijímačem zabezpečuje předáváním tzv. synchronizačních znaků nebo jejich posloupností. Tyto synchronizační znaky jsou doplňovány před vysílanou zprávu a zabezpečují synchronizování přijímače dat s vysílačem. Aby nehrozilo nebezpečí synchronizace přijímače při příjmu náhodné posloupnosti bitů vzniklé poruchou, je často nastavení této znakové synchronizace podmíněno příjmem synchronizačního znaku s dvojnásobnou délkou nebo dvojice synchronizačních značek vysílaných vysílačem bezprostředně po sobě. Protože generátory hodinových signálů vysílače a přijímače pracují s určitou tolerancí, je nutno při synchronním přenosu dat používat takové kódování znaků, které zabezpečí, že při přenosu libovolné zprávy bude docházet ke změnám úrovně signálu tak často, aby bylo zabezpečeno udržení synchronizace přijímače s vysílačem.

Arytmický přenos je kombinací synchronního a asynchronního přenosu. Je kompromisním řešením, kdy se spoléháme na to, že hodinový signál přijímající strany vydrží jít dostatečně přesně alespoň nějakou dobu. Data jsou pak vysílána ve skupinkách pevně dané velikosti (např. 8 bitů). Na začátek každé takové skupinky je umístěna speciální posloupnost, která umožní přijímající straně "seřídit si" jeho hodinový signál a pak samostatně určovat časy vzorkování přicházející posloupnosti bitů. V praxi je tato metoda velmi oblíbená a místo o skupinkách znaků se hovoří o znacích, které jsou při přenosu ohraničeny na začátku tzv. start bity a na konci stop bitem. Tento pojem pochází od znakových terminálů, které vysílaly opravdu za sebou jednotlivá písmenka, přičemž jejich hustota závisela na zručnosti operátora, který u terminálu seděl a nebylo proto možné předem říci, za jak dlouho přijde další znak. Odtud pojem arytmický, neboli postrádající rytmus. Arytmický přenos používá jak běžná sériová rozhraní PC, tak běžně používané modemy.

Dále je možno podle směru přenosu dat přenosovou linkou rozdělit přenos dat na jednosměrný a obousměrný. Při použití jednosměrných přenosových linek se ovšem velmi často používají jejich dvojice. Jedna linka zabezpečuje přenos dat jedním směrem, druhá pak směrem opačným. Při obousměrném přenosu dat je jedna přenosová linka využívána pro přenos dat oběma směry. Při

úplném obousměrném (plně duplexním) přenosu dat se mohou data přenášet oběma směry současně. Při tzv. poloduplexním provozu se data přenášejí buď jedním nebo druhým směrem, není zde možný současný přenos dat oběma směry. Poloduplexní přenos dat je jednodušší z hlediska řízení přenosu dat, neboť linkou je přenesen blok dat a další (resp. stejný) blok dat je přenášen až po potvrzení správného (resp. chybného) přijetí bloku dat ze strany příjemce. Při plně duplexním provozu se přenášejí bloky dat za sebou a je nutno je číslovat. Zprávy o správném (resp. chybném) přijetí bloku pak musí obsahovat toto číslo přenášeného bloku. Pro zabezpečení přenosu dat je nutné uspořádat jednotlivé přenášené informace do zprávy v jednoznačně definovaném formátu zprávy. [5]

3.1 Proudová smyčka

Možnou alternativou k rozhraní TIA/EIA 485 je rozhraní typu proudová smyčka. Historicky vzato se jedná vlastní o nejstarší sériové rozhraní vůbec a jeho počátky jsou spjaty s vývojem dálnopisné techniky v prvních desetiletích dvacátého století. Logické úrovně zde jsou vyjádřeny pomocí proudových signálů a v důsledku toho je přenos velmi odolný proti rušení, neboť energie rušivého pole nutná k tomu, aby se proud procházející smyčkou změnil natolik, aby došlo k chybě, je o několik řádů větší než energie potřebná ke změně napětí v signálovém vodiči, kterým protéká jen malý proud daný vstupním odporem přijímače (3-7 kΩ). Princip funkce proudové smyčky je velmi jednoduchý a je schematicky znázorněn na obr.3.1. Smyčka je buzena zdrojem proudu. Jím může ve složitějším případě být skutečně zdroj konstantního proudu schopný udržet ve smyčce stálou hodnotu proudu, která bude v určitém rozsahu odporů nezávislá na změnách odporu smyčky. Častěji to však bude pouze zdroj napětí doplněný sériovým odporem omezujícím proud smyčkou na vhodnou hodnotu. Sepnutí spínače vysílače umožní, aby proud obvodem procházel a odpovídá log.1, je-li spínač rozepnut proud neprochází a je tedy vysílána log.0. Ve skutečné realizaci se samozřejmě jedná o elektronické spínače realizované pomocí tranzistorů nikoliv o spínače mechanické. K tomu, aby mohl procházet dostatečně vysoký proud, aniž by to vyžadovalo zbytečně velké napájecí napětí smyčky, musí mít přijímače nízký vstupní odpor, nanejvýše ve stovkách Ohmů.

Zařízení, která poskytují proud k napájení smyčky označujeme jako zařízení aktivní, zatímco pasivní zařízení buď pouze přijímají nebo pokud zároveň obsahují i vysílač, mohou pouze propouštět nebo zabraňovat průchodu proudu, který je generován aktivním zařízením. Na rozdíl od aktivního zařízení, které smí být v každé smyčce pouze jedno, je možné pasivních zařízení zařadit v sérii za sebou do smyčky několik (součet jejich vstupních odporů ovšem nesmí překročit přípustné maximum) a proudová smyčka tak může zabezpečovat nejen dvoubodové spojení, ale umožňuje vytvořit i jednoduchou kruhovou síť. Pokud uvažujeme dvoubodové spojení, můžeme ještě rozlišit podle obr.3.3.2, plně duplexní propojení umožňující současnou komunikaci oběma směry a poloduplexní propojení, které umožňuje komunikaci buď jedním nebo druhým směrem ne však oběma zároveň. Z obrázku je také patrné, že spínač vysílače té strany, která právě přijímá, musí být v

sepnutém stavu, jinak by spojení nebylo možné. Doplněním tohoto poloduplexního propojení o další pasivní zařízení pak získáme již zmíněnou kruhovou síť.

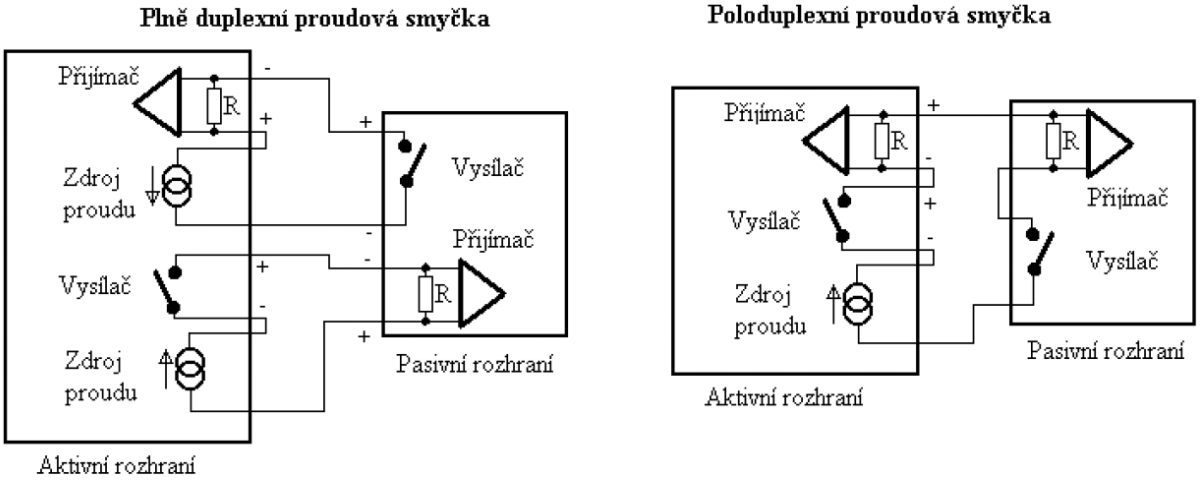
Pro proudovou smyčku neexistuje žádná všeobecně používaná norma, která by definovala její elektrické či mechanické parametry, ale zařízení jednotlivých výrobců se více či méně liší a jejich vzájemná kompatibilita je tak ještě problematičtější než u zařízení používajících rozhraní TIA/EIA 485. Jednotlivé varianty tohoto rozhraní se liší již velikostí proudu procházejícího smyčkou v sepnutém stavu při log.1. Z tohoto hlediska lze rozlišit proudovou smyčku 60 mA, 30 mA a 20 mA, přičemž v současné době bývá ovšem zdaleka nejčastěji užíváno 20 mA proudové smyčky. I zařízení používající 20 mA smyčku se však mohou více či méně lišit v konstrukčním provedení budičů, logických úrovních a v neposlední řadě i typem použitého konektoru. Na obecné rovině lze snad říci jen to, že logické úrovně nejsou definovány jenom konstatováním, že při log.0 neteče smyčkou nic a při log.1 teče 20 mA, ale v úvahu se bere skutečnost, že budiče proudové smyčky mají často dosti daleko k ideálnímu proudovému zdroji a proud v log.1 tedy jednak závisí na odporu vedení, a jednak nemusí být vzhledem k nedokonalostem polovodičových spínačů a přítomnosti rušení v log.0 zcela nulový. Obvykle se tedy v případě 20 mA smyčky pracuje s následujícími definicemi logických hodnot:

log. 0 - proud od 0 do 3 mA

log. 1 - proud od 14 do 20 mA

Někdy a zvláště při přenosu dat na větší vzdálenosti se používá také symetrická varianta proudové smyčky. V tomto případě jsou logické úrovně stanoveny tak, že proudy v log. 1 a v log. 0 mají tutéž velikost, ale opačný směr.

Vzdálenosti na něž je možné komunikovat a maximální přenosové rychlosti se u jednotlivých zařízení pracujících s proudovou smyčkou značně liší. Za maximální hodnoty lze pokládat v případě vzdálenosti jednotky km (výjimečně i desítky km) a v případě rychlostí 19200 bit/s (zřídka až 38400 bit/s i více), často se však lze technických datech setkat v případě vzdáleností i rychlostí s hodnotami podstatně menšími. Pro přenos dat po rozhraní typu proudová smyčka se je obvykle používá asynchronní přenos s obdobnou strukturou rámce jako u rozhraní TIA/EIA 232. Nejčastěji je ostatně zdrojem proudové smyčky výstup ze sériového portu pracujícího podle TIA/EIA 232, který byl převeden do proudových úrovní, aby byl schopný přenosu na delší vzdálenosti v zarušeném prostředí.



Obrázek 3.1 - Proudová smyčka

3.2 EIA/TIA 232

První varianta tohoto rozhraní byla americkou organizací EIA (Electronic Industries Association) definována již v roce 1962 pod označením RS 232 (RS obvykle chápáno jako zkratka z Recommended Standard, ve skutečnosti však z Radio Section). Později byla opakovaně revidována a významného rozšíření pak doznala především její třetí revize RS 232 C pocházející z roku 1969. Po ní následovaly další varianty (EIA 232 D, TIA/EIA 232 E). Jeho v současné době nejnovější verze je jakožto společné doporučení EIA a TIA (Telecommunications Industry Association) oficiálně uváděna pod názvem TIA/EIA 232 F. Vzhledem k tomu, že rozdíly mezi RS 232 C a pozdějšími variantami rozhraní nejsou příliš rozsáhlé a velké zažitosti staršího označení, je oficiální pojmenování používáno spíše jen výjimečně. V literatuře se proto o tomto rozhraní stále hovoří nejčastěji jako o rozhraní RS 232 C nebo prostě jen RS 232. Sériové rozhraní, které je v podstatných rysech shodné s TIA/EIA 232 F, je definováno rovněž mezinárodními doporučeními ITU-T pod označeními V.24 (definice datových a řídicích signálů) a V.28 (elektrické parametry).

Významnou výhodou tohoto rozhraní je především všeobecná dostupnost a rozšířenost daná tím, že je součástí téměř každého osobního počítače. Dále však již následují spíše nevýhody. Rozhraní umožňuje pouze komunikaci relativně nízkými rychlostmi a na krátké vzdálenosti. Podstatný nedostatek je dán i samotným elektrickým uspořádáním rozhraní, v němž země vysílače a přijímače jsou spojeny a pro určení logické úrovně na signálových vodičích je rozhodující napětí vůči této společné zemi. Jelikož zemní potenciály zařízení připojených k různým větvím síťového rozvodu se často liší, mohou zemním vodičem jednak protékat vyrovnávací proudy předem neodhadnutelné velikosti a jednak tento posuv zemního potenciálu mezi vysílačem a přijímačem může vést ke špatnému vyhodnocení logické hodnoty signálu a ve zvláště nepříznivém případě i ke zničení obvodů rozhraní. Toto uspořádání také činí přenos dat rozhraním málo odolným vůči rušení, neboť rušivé

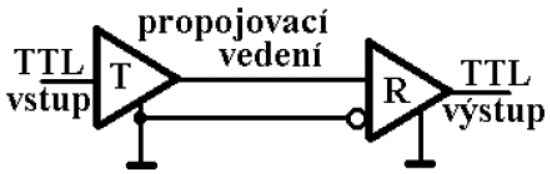
napětí, které se indukuje v signálovém vodiči a sčítá se s užitečným napětím, je vyhodnocováno vzhledem k relativně stálému zemnímu potenciálu může způsobit chybnou interpretaci logických úrovní signálu. Proti tomu je samozřejmě možné se alespoň částečně chránit použitím stíněných kabelů, bohužel však z důvodů uvedených výše dochází v tomto případě zároveň i ke zmenšení vzdálenosti, na níž je možné komunikovat. Použití v oblasti řídicí techniky je proto většinou omezeno na spíše laboratorní aplikace jako je připojení externích měřicích a vstupně výstupních modulů či měřicích přístrojů k osobnímu počítači nebo na propojení, které je jen na krátkou vzdálenost a má pouze nastavovací či diagnostický a nikoliv trvalý provozní charakter: Poměrně běžně je tak např. používáno k přenosu programů napsaných a přeložených na osobním počítači do programovatelného automatu. Jako propojení, které by mělo trvale obstát v podmínkách průmyslového provozu je však většinou nepoužitelné a je tedy třeba použít vhodnější rozhraní. Ta musí být v první řadě odolnější vůči rušení a schopná provozu na větší vzdálenosti. Často, byť již nikoliv vždy, je zároveň třeba, aby mohla pracovat také s vyššími přenosovými rychlostmi. [4]

3.3 EIA/TIA 422/485

S využitím proudové smyčky lze provozovat komunikaci na poměrně dlouhé vzdálenosti a s velmi dobrou odolností proti rušení. Nejvyšší dosažitelné rychlosti přenosu však stále zůstávají dosti nízké. Je-li proto požadována rychlejší komunikace, je třeba použít vhodnější rozhraní, která sice pracují s napěťovými signály, jsou však modifikována tak, aby eliminovala největší nedostatky rozhraní EIA/EIA 232. Řada problematických rysů tohoto rozhraní je spjata s tím, že země vysílače a přijímače jsou propojeny a logické úrovně jsou vyhodnocovány na základě velikosti napětí na signálovém vodiči vůči této společné zemi. Nabízí se tedy možnost pracovat namísto toho s diferenciálním signálem a vyhodnocovat napětí mezi dvěma signálovými vodiči. První možností, jak něco podobného učinit, je použít v zásadě obdobné zapojení vysílače jako u TIA/EIA 232, ale vyhodnocení provést pomocí přijímače s diferenciálními vstupy. Tak je tomu v případě rozhraní TIA/EIA 423 B znázorněného na obr.5.1. Zde již případný potenciálový rozdíl mezi zemí vysílače a zemí přijímače bezprostředně neohrožuje kvalitu přenosu, neboť pro určení logické úrovně na vstupu přijímače je podstatný rozdíl napětí mezi jeho dvěma vstupy a ne vůči zemi přijímače. Na druhé straně však potenciálový rozdíl mezi oběma zeměmi nesmí samozřejmě být příliš vysoký (ne více než jednotky voltů), jelikož v takovém případě by mohlo dojít k nesprávné funkci rozhraní nebo i ke zničení obvodů. Nejvyšší přenosová rychlost na tomto rozhraní je 100 kbit/s na vzdálenosti do 12 m. Při menších rychlostech lze ovšem přenosovou vzdálenost podstatně zvýšit až na asi 1200 m při 1 kbit/s.

Navíc také umožňuje nikoliv pouze dvoubodovou komunikaci, ale přijímačů může být na linku připojeno více (tzv. multidrop uspořádání). Přesto však není tohoto rozhraní využíváno příliš často. Příčinou je skutečnost, že místo tohoto poněkud hybridního rozhraní kombinujícího nesymetrický

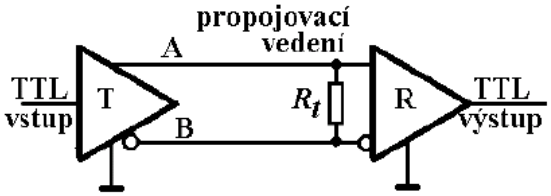
vysílač se symetrickým přijímačem, jsou častěji užívána plně symetrická rozhraní TIA/EIA-422 a TIA/EIA-485. [4]



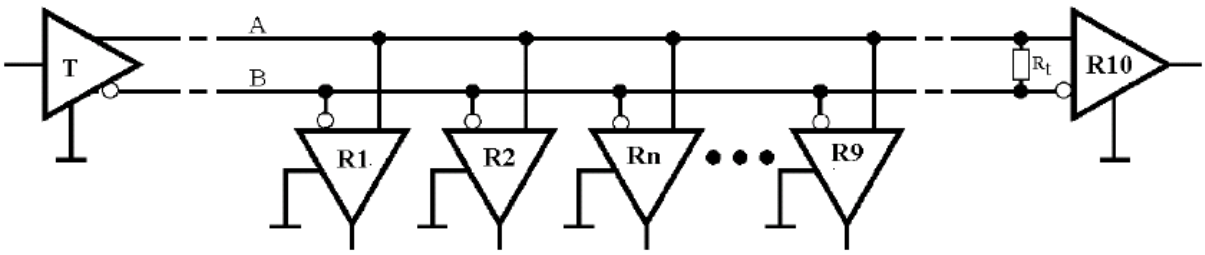
Obrázek 3.3.2 - Rozhraní TIA/EIA 423

3.3.1 Základní zapojení

Základní zapojení rozhraní TIA/EIA 422 je schematicky naznačeno na obr. 3.3.3 Vysílač i přijímač mají diferenciální uspořádání a logické úrovně jsou definovány pomocí rozdílu napětí mezi vodiči A a B. Jedné úrovni odpovídá kladná a druhé záporná napěťová diference mezi vodičem A a vodičem B. Napětí na obou vodičích U_A a U_B vůči zemi však obvykle zůstávají kladná stále, mění se pouze znaménko jejich rozdílu. Kromě nejjednoduššího zapojení podle obr. 3.3.3, které uskutečňuje dvoubodové spojení podobně jako tomu bylo u rozhraní TIA/EIA 232, je u rozhraní TIA/EIA 422 možné i již zmíněné multidrop uspořádání, kde je k jednomu vysílači připojeno až deset přijímačů nebo přesněji řečeno deset zátěží se vstupním odporem 4 kΩ. Je-li vstupní odpor přijímačů větší, je možné jich připojit i více. Toto uspořádání je schematicky znázorněno na obr. 3.3.4.

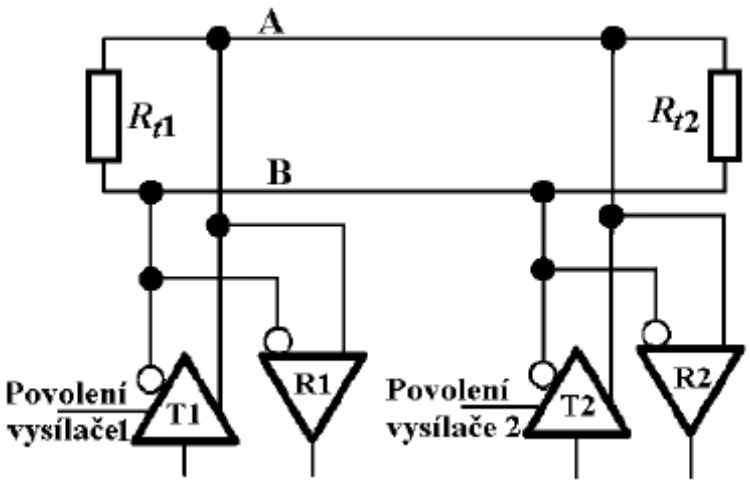


Obrázek 3.3.3 - Rozhraní TIA/EIA 422



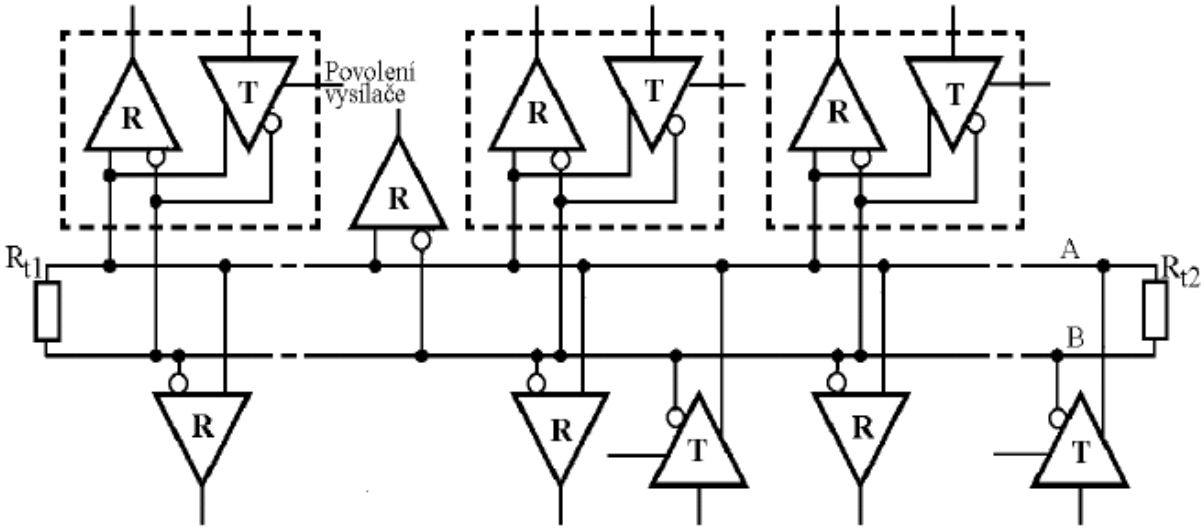
Obrázek 3.3.4 - Rozhraní TIA/EIA 422 v multidrop uspořádání

Zde mohou být vysílaná data sice zároveň přijímána několika přijímači, jde však stále o přenos dat pouze jedním směrem. Měla-li by být dvě zařízení schopna komunikovat spolu obousměrně, bylo by nutné použít dvě nezávislá vedení, každé vedoucí opačným směrem, což by vyžadovalo celkem čtyři vodiče. Při požadavku na vzájemnou komunikaci většího množství zařízení by pak komplikovanost propojení dále rostla. Rozhraní TIA/EIA 422 tedy neumožňuje obousměrný přenos dat po jednom vedení ani vytvoření sériové sběrnice. Pro realizaci těchto struktur je třeba použít rozhraní TIA/EIA 485. Toto rozhraní je rozhraní TIA/EIA 422 velmi podobné. Je rovněž symetrické s diferenciálním uspořádáním přijímače i vysílače. Lze jej tedy využít i k realizaci sériové komunikace podle obr. 3.3.3 a 3.3.4. Vysílače však navíc mohou být uvedeny do třetího stavu, v němž jsou od vedení odpojeny, což umožňuje realizovat sériovou sběrnici. Kromě toho existují mezi oběma rozhraními i jisté odlišnosti v elektrických parametrech, které budou v následujícím výkladu postupně zmíněny.



Obrázek 3.3.5 - Rozhraní TIA/EIA 485 v zapojení umožňujícím poloduplexní obousměrnou komunikaci

Základní zapojení TIA/EIA 485 je uvedeno na obr. 3.3.5. Podle toho, zda je povolen vysílač T1 nebo T2, může přenos dat probíhat jedním nebo druhým směrem. Pro správnou funkci zapojení musí samozřejmě být zajištěno, že oba vysílače nebudou povoleny zároveň. Uvedené zapojení lze snadno zobecnit a přidáním dalších přijímačů a vysílačů je možné realizovat nikoliv jen obousměrnou poloduplexní linku ale sériovou sběrnici. Její zapojení je naznačeno na obr. 3.3.6. Jak je uvedeno na tomto obrázku, na sběrnici mohou být vedle zařízení schopných data jak přijímat tak vysílat také zařízení, která data pouze přijímají popř. pouze vysílají. [4]

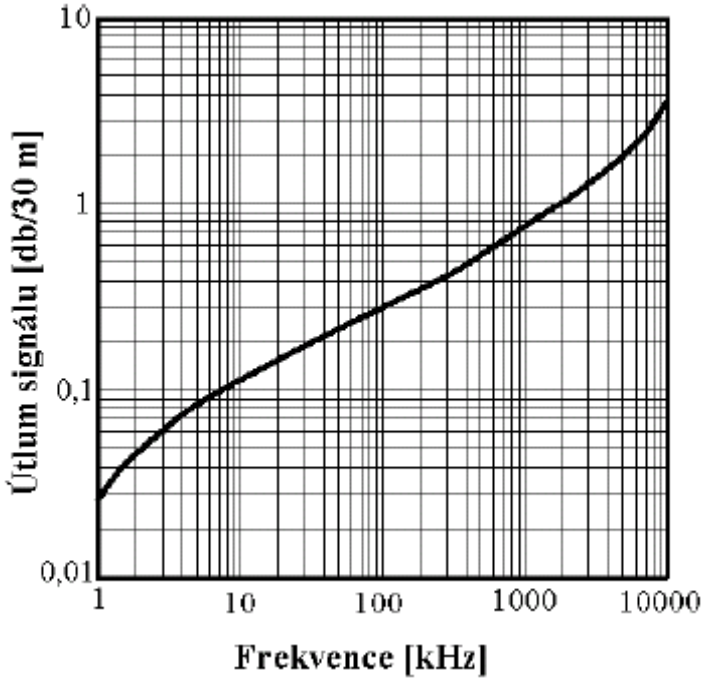


Obrázek 3.3.6 - Sériová sběrnice realizovaná pomocí rozhraní TIA/EIA 485

3.3.2 Elektrické parametry

Ačkoliv jsou obě rozhraní velmi podobná, v některých elektrických parametrech se přesto mírně liší. Vysílače rozhraní TIA/EIA 422 musí být dimenzovány tak, aby při plné zátěži (deset přijímačů se vstupním odporem 4 kΩ a zakončovací odpor 100 Ω) bylo na jejich výstupech rozdílové napětí alespoň ±2 V, maximální rozdílové napětí pak smí být ±10 V. Přitom však nesmí hodnoty napětí U_{OA} a U_{OB} proti zemi na výstupu vysílače přesáhnout ±6 V. U rozhraní TIA/EIA 485 musí být při plné zátěži na výstupu rozdílové napětí alespoň ±1,5 V. Tuto plnou zátěž představují dva zakončovací odpory s doporučenou hodnotou 120 Ω a k tomu až 32 přijímačů se vstupním odporem 12 kΩ. Jedná se tedy o větší zatížení než u rozhraní 422. Použitím přijímačů s větším vstupním odporem lze počet zařízení připojitelných k rozhraní zvětšit (běžně dostupné jsou obvody, které zatěžují sběrnici jednou osminou jednotkové zátěže tj. vstupním odporem Jak již bylo zmíněno, rozhraní TIA/EIA 422 i 485 jsou určena pro podstatně větší rychlosti přenosu než rozhraní předchozí. Jejich maximální přenosová rychlost není normou přesně stanovena, vychází však z vlastností používaných budičů a z požadavku na zachování určitého poměru mezi dobou vysílání jednoho bitu a trváním náběžných a sestupných hran impulsů. Obvykle proto bývá udáváno, že maximální přenosová rychlost je 10 Mbit/s. S touto rychlostí lze však komunikovat pouze na krátké vzdálenosti, na nichž se ještě výrazněji neuplatňuje vliv přenosového vedení. S rostoucí vzdáleností se pak začíná negativně projevovat frekvenčně závislý útlum signálu. Orientační graf závislosti útlumu na frekvenci pro kroucený dvoudrát je zakreslen na obr. 3.3.7. Podstatné přitom nejsou ani tak konkrétní hodnoty (převzaté v tomto případě z Texas Instruments, 1998a), protože ty se u jednotlivých typů kabelů liší a je třeba vyjít vždy z údajů výrobce kabelu použitého v dané aplikaci, ale skutečnost, že tento útlum s frekvencí poměrně výrazně narůstá (důležitou roli v tom hraje mj. tzv. skin efekt, kdy proud vyšších frekvencí prochází pouze na

povrchu a nikoliv celým průřezem vodiče a jeho odpor proto ve srovnání se stejnosměrným odporem roste). Při větších rychlostech přenosu tak již při relativně nevelkých délkách kabelu přesáhne přijatelnou míru.



Obrázek 3.3.7 - Průběh závislosti útlumu kabelu na frekvenci

V určení toho, co bude ještě přijatelné existuje samozřejmě určitý prostor. Jednoznačně platí pouze to, že zeslabení nemůže být větší než takové, které minimální zaručený rozdílový signál na výstupu vysílače (±2 resp. ±1,5 V) zeslabí na minimální požadovaný rozdílový signál na vstupu přijímače (±200 mV). Takové zeslabení by ovšem naprosto eliminovalo jakoukoli šumovou imunitu. Z tohoto důvodu je třeba přípustný útlum volit menší. Obvykle se počítá, že signál smí být zeslaben na polovinu či v nejhorším případě na třetinu (tj. o 6 resp. o 10 dB). Při práci s údaji o útlumu kabelu je také třeba vzít v úvahu, že souvislost mezi přenosovou rychlostí a frekvencí signálu na vedení, není nikterak jednoduše přehledná. Na rozhraní 422 i 485 jsou většinou posílána přímo vysílaná binární data (tj. jedná se o NRZ kódování a na lince je buď kladné či záporné rozdílové napětí) a základní frekvence vysílaného signálu je tak v průměru rovna polovině přenosové rychlosti. Signál je to ovšem obdélníkový nikoliv harmonický a frekvenční spektrum jeho hran sahá mnohem výše než tato základní frekvence. Pro velmi hrubý odhad útlumu je proto třeba pracovat s minimálně dvojnásobkem této frekvence (tj. frekvence signálu rovná se přenosová rychlost). Frekvenčně závislý útlum navíc způsobuje i zkreslení signálu, neboť jeho jednotlivé harmonické složky jsou přeneseny s různým útlumem. Kromě toho se projevuje i další zkreslení způsobené tím, že i rychlost šíření signálu po vedení je frekvenčně závislá. Jestliže se pak jednotlivé harmonické šíří po lince různou rychlostí a linka je dostatečně dlouhá, aby se časy, v nichž jednotlivé složky dorazí na její konec, výrazněji lišily, je to, co vznikne součtem těchto různě zpožděných harmonických, velmi nepodobné tomu, co bylo

vysláno, a důsledkem je další výrazné zkreslení průběhu signálu a z toho plynoucí chyby přenosu. Omezení na délku kabelů při dané přenosové rychlosti je proto ještě výraznější než by odpovídalo útlumu kabelu. Pro stanovení maximální vzdálenosti, na níž lze pracovat alespoň při malých přenosových rychlostech, jsou pak důležité i stejnosměrné parametry vedení. V propojovacích kabelech bývá nejčastěji používán měděný vodič rozměru standardizovaného jako 24 AWG American Wire Gauge). Jeho průměr je 0,511 mm, jedná-li se o plný vodič, popř. 0,61 mm, jde-li o lanko, a stejnosměrný odpor cca 84 Ω popř. 76Ω/km. Z důvodů, které budou vysvětleny níže, je vedení ukončováno odporem R_t, jehož hodnota je u rozhraní TIA/EIA 422 obvykle 100 Ω a u rozhraní TIA/EIA 485 120 Ω. Tento odpor vytváří se sériovým odporem vedení napěťový dělič určující hodnotu napětí na vstupu přijímače v ustáleném stavu. Nemá-li být signál zeslaben na více než třetinu hodnoty na výstupu vysílače, neměl by odpor jednoho drátu vedení převyšovat velikost tohoto ukončovacího odporu. Z toho pak vychází maximální doporučená délka vedení zhruba takto 1200 m. Kombinací těchto vlivů pak lze získat obvyklá orientační pravidla udávající vztah mezi přenosovou rychlostí a vzdáleností, na níž lze přenášet data. Typický průběh je zakreslen na obr. 3.4.8.

Graf je rozčleněn na tři části. Část jedna odpovídá oblasti, kde se ještě vliv zkreslení a útlum způsobované vedením výrazněji neprojevují a je tedy možné do asi 12 m pracovat s maximální rychlostí cca 10 Mbit/s. Druhá část pak popisuje chování rozhraní v té oblasti, kdy již je třeba v důsledku ztrát a zkreslení s rostoucí délkou vedení snižovat přenosovou rychlost. Třetí část pak odpovídá stavu, kdy délka vedení již narazila i na čistě stejnosměrná omezení a nelze ji již dále zvyšovat ani při ještě klesající přenosové rychlosti. Závislost uvedenou na obrázku je rovněž možné vyjádřit v podobě orientačního pravidla. Rychlost přenosu [bit/s] x Délka vedení [m] $\leq 1{,}2 \cdot 10^8$ (4.5)

Přitom ovšem musí být respektována obě horní omezení tj. jednak rychlost do cca 10 Mbit/s a především maximální délka vedení do 1200 m. Jelikož při stanovení této maximální vzdálenosti již hraje podstatnou roli i stejnosměrné chování přenosové linky se zakončovacím odporem, není možné pracovat např. s rychlostí 1 kbit/s na vzdálenost 100 km, přestože by vztah 4.5 byl formálně vzato splněn. Zároveň však je třeba zdůraznit, že jak je ostatně patrné i z předchozího výkladu, tato závislost je orientační a není ničím, co by norma přímo předepisovala, ale odpovídá nejběžněji používanému kabelu (kroucený dvoudrát 24 AWG, s kapacitou 16 pF/ft tj. cca 50 pF/m) a obvyklým budičům rozhraní. Jsou-li použity vysílače konstruované na vyšší rychlosti a kvalitnější kabely je možné podstatně zvětšit především oblast 1 grafu tj. délku vedení, na níž lze pracovat s maximální rychlostí. Tak např. průmyslová sběrnice Profibus DP, o níž bude řeč v dalším výkladu, používá rozhraní TIA/EIA 485 A pro komunikaci rychlostí 12 Mbit/s až do vzdálenosti 100 m. [4]

3.3.3 Porovnání

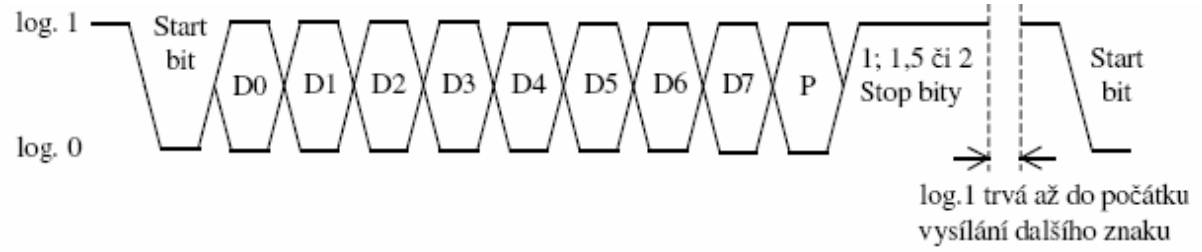
Jelikož vlastnosti rozhraní TIA/EIA 422 a 485 byly uvedeny postupně a jejich vyhledávání v textu by občas mohlo být zbytečně zdlouhavé, jsou v následující tabulce shrnuty ty nejdůležitější parametry rozhraní. Z této tabulky je také zřejmé do jaké míry jsou obě rozhraní vzájemně slučitelná. V podstatě lze říci, že zapojíme-li do obvodu, který byl navržen za předpokladu, že budou použity obvody splňující požadavky TIA/EIA 422 , obvody splňující TIA/EIA 485 nenastanou žádné problémy. Jediným parametrem, v němž se budiče rozhraní 485 na první pohled zdají být horší, je minimální rozdílové napětí na výstupu vysílače při plné zátěži. Je však třeba si uvědomit, že tato zátěž je podstatně vyšší než u rozhraní 422 a technické specifikace budičů rozhraní 485 proto obvykle zajišťují, že budou-li zatíženy pouze plnou zátěží odpovídající 422 poskytnou na svém výstupu rovněž alespoň 2V. Opačná zaměnitelnost však je velmi problematická až nemožná, neboť obvody splňující pouze TIA/EIA 422 nejsou uzpůsobeny pro sběrnicový provoz a mají menší zatížitelnost stejně jako menší přípustný rozsah vstupního souhlasného napětí. [4]

Parametr	TIA/EIA 422	TIA/EIA 485
Počet přijímačů a vysilačů	1 vysílač, 10 přijímačů	32 vysílačů, 32 přijímačů
Maximální délka vedení	1200m	1200m
Nejvyšší přenosová rychlost	10Mbit/s	10Mbit/s
Rozdílové napětí na vstupu vysílače	$2V \leq\leq 10V$	$1,5V \leq\leq 6V$
Zátěž vysílače(vstupy přijímačů v to nepočítaje)	$\geq 100\Omega$	$\geq 60\Omega$
Proudové omezení na výstupu vysílače	150mA při zkratu na zem	150mA při zkratu na zem 250mA při zkratu na -7V nebo +12V
Vstupní odpor vysílače	$\geq 4\Omega$	$\geq 12\Omega$
Citlivost přijímače	±200mV	±200mV
Největší přípustné souhlasné napětí na vstupu přijímače	±7V	-7V až +12V

Tabulka 3.1 - Nejdůležitější parametry RS-422 a RS-485

3.4 Formát přenosu dat

Toto rozhraní používají variantu asynchronního přenosu, při níž jsou nezávislé hodiny přijímače a vysílače vždy znovu synchronizovány při vysílání každého datového slova pomocí zvláštních synchronizačních bitů tzv. start a stop bitů. Princip tohoto způsobu přenosu je znázorněn na obr. 5.3. V klidovém stavu, kdy není vysíláno nic, je na přenosové lince logická hodnota 1. Stav přenosové linky je přijímačem periodicky vzorkován s frekvencí, která je celočíselným násobkem přenosové rychlosti (obvykle je 16x nebo 64x vyšší). Pokud přijímač zjistí, že došlo ke změně stavu z log.1 do log.0, interpretuje to jako počátek start bitu, počká po dobu odpovídající polovině doby, která je při zvolené přenosové rychlosti vyhrazena na přenos jednoho bitu a stav linky otestuje znovu. Pokud zjistí, že se vrátil do log.1, znamená to, že předchozí změna byla pouze náhodným šumem a nikoliv skutečným start bitem a přijímač proto začne znovu pravidelně vzorkovat stav linky jako předtím a čeká na nový přechod z log.1 do log.0. Jestliže však se signál po uplynutí poloviny bitové periody stále rovná log.0, jedná se zřejmě o skutečný start bit a přijímač začne testovat přicházející signál vždy po uplynutí jedné bitové periody. Tedy tak, aby k testování stavu linky došlo vždy zhruba v polovině vysílání každého jednotlivého bitu. Tímto způsobem jsou postupně načteny hodnoty jednotlivých vysílaných datových bitů a na závěr je pak jednou či dvakrát testována hodnota stop bitu a podle výsledku tohoto testu se určí, zda datové slovo bylo přeneseno nebo došlo k tzv. chybě rámce (tzn. stop bity byly skutečně nějakým způsobem porušeny nebo jsou přijímač i vysílač nakonfigurovány na jiný počet stop bitů).



Obrázek 3.4.8 - Přenos jednoho datového slova

Celé vysílané slovo je tak zahrnuto do rámce, který začíná jedním nulovým start bitem a zakončují jej volitelně buď jeden, jeden a půl nebo dva jedničkové stop bity. Varianta jeden a půl stop bitu přitom fakticky znamená, že log.1 se na lince objeví po dobu odpovídající jeden a půl násobku času, který je při zvolené přenosové rychlosti vymezen na vyslání jednoho bitu. Hlavním důvodem proč jsou stop bity vysílány je poskytnout přijímacímu zařízení čas, aby se mohlo připravit k přijetí dalšího slova.

Používání většího počtu stop bitů než jeden má proto smysl jen u zvláště pomalých zařízení jako jsou např. elektromechanické dálnopisy. Samo vysílané slovo může volitelně obsahovat 5 až 8 datových bitů a k tomu jeden paritní bit. Paritní bit přitom může být nastaven jedním z následujících způsobů:

a) **sudá parita** – bit je nastaven tak, aby celkový počet jedničkových bitů ve vysílaném slově včetně paritního bitu bylo sudé číslo

b) **lichá parita** – bit je nastaven tak, aby celkový počet jedničkových bitů ve vysílaném slově včetně paritního bitu bylo liché číslo

c) **nulová parita** (space parity)– paritní bit je vždy v log.0

d) **jedničková parita** (mark parity) – paritní bit je vždy v log.1

e) **žádná parita** - paritní bit se nepoužívá

Skutečný význam mají pouze varianty a) a b), které představují nejjednodušší formu zabezpečení přenosu dat. Dojde-li právě u jednoho bitu k chybě přenosu tzn. log.0 bude přijata jako log. 1 nebo obráceně, počet jedniček se změní podle nastaveného druhu parity ze sudého na lichý nebo opačně a tato situace bude vyhodnocena jako chyba parity. Toto jednoduché zabezpečovací schéma však již není schopno odhalit dvě chyby v přenosu, neboť v tomto případě se parita nezmění. Podobně také nedokáže rozpoznat, zda došlo jen k jedné nebo k většímu lichému počtu chyb a samozřejmě není také schopno chyby opravit. Má-li přenos proběhnout správně, je třeba nejen, aby přenosové rychlosti přijímače i vysílače byly nastaveny na stejnou hodnotu, ale také musí oba mít nastaven stejný počet datových bitů, stop bitů a stejnou paritu.

Při popsaném způsobu sériového přenosu dat je počátek vysílání datového slova skutečně asynchronní událostí, která může nastat kdykoliv bez vazby na jakýkoliv synchronizační signál. V rámci jednoho vysílaného rámce však již přenos probíhá synchronně. Celý postup je proto možné chápat jako zvláštní kombinaci synchronního a asynchronního přenosu označován jako tzv. arytmický přenos. Daleko častěji však je označován prostě jako asynchronní. [4]

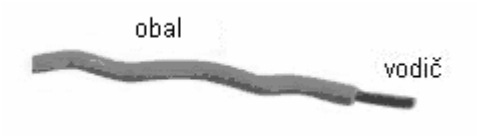
4 Druhy kabelů

V této kapitole se zabývám problematikou volby použití kabelu. V současné době je k dispozici celá řada kabelů . Detailněji popisuji kabely, vhodné pro použití v oblasti datového (komunikačního) spojení a jsou také běžně užívány v praxi.

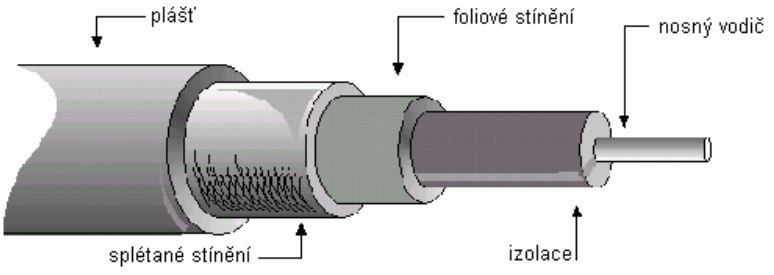
Kabel je jeden nebo více vodičů spojených dohromady. Vodiče mohou být holé nebo chráněné izolací. Kabel nemusí obsahovat jen elektrické vodiče, ale i jiná media, např. optický kabel. Kabely jsou nejčastěji používané k přenosu elektrické energie a optických signálů. Podle způsobu užití mohou být kabely ještě upraveny. Mohou být točené (jako telefonní šňůra) vázané, pletené, jednotlivé vodiče mohou být zkroucený do tvaru šroubovice takzvaná kroucená dvoulinka a podobně. Kabely mohou být stíněny vodivou vrstvou tzv. stíněný kabel. [3]

Mezi jednotlivé druhy patří například:

- obyčejný kabel

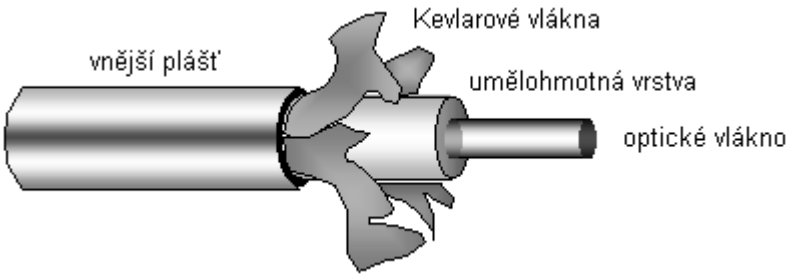


- koaxiální kabel



- kroucená dvojlinka

- optický kabel



4.1 Kroucená dvojlinka (twisted pair)

Kroucená dvoulinka je druh kabelu, který je používán v telefonních a počítačových sítích. Kroucená dvoulinka je tvořena dvěma vodiči (párem vodičů), které jsou po své délce pravidelným způsobem zkrouceny (anglicky: twisted, odsud také twisted pair). Oba vodiče jsou v zásadě rovnocenné (i v tom smyslu, že žádný z nich není spojován se zemí či s kostrou), a kroucená dvoulinka proto patří mezi tzv. symetrická vedení. Signál, přenášený po kroucené dvoulince, je vyjádřen rozdílem potenciálů obou vodičů. Signál je přenášen jako rozdíl mezi těmito dvěma signály, což způsobuje menší náchylnost k rušení a útlumu. Kroucená dvojlinka se vyskytuje ve dvou formách. Stíněná označována STP (shielded twisted pair) a nestíněná označovaná UTP(unshielded twisted pair). [3]

Výhody kroucené dvojlinky jsou snadné připojování jednotlivých zařízení, možno využít i pro telefonní (popř. jiné) rozvody, STP má velmi dobrou ochranu proti rušení, snadná instalace a nízká cena.

Nevýhody kroucené dvojlinky jsou, že STP je silný a obtížně se s ním pracuje, UTP je citlivější na šum než koaxiální kabel, UTP signály nemohou bez regenerace (zesílení a čištění) být přenášeny na větší vzdálenost (ve srovnání s jinými typy kabelů) a realizování pouze pro dvojbodové spojení.

4.2 Koaxiální kabel

Koaxiální kabel je používán k přenosu hlasu i dat (telefonování, internet, vedení vysílacích nebo přijímacích antén, satelitní přijímače). Je nejdéle používaným kabelem na propojení počítačové sítě. Podobně jako kroucená dvoulinka je i koaxiální kabel tvořen dvojicí vodičů, zato ale v jiném vzájemném uspořádání. Jeden z vodičů je středový (ve středu kabelu), zatímco druhý vodič je součástí vodivého opláštění, které obaluje izolační vrstvu kolem středového vodiče. Tento druhý vodič má současně i dobrý stínící účinek (stínění znamená zabránění tomu, aby se signály těchto vodičů navzájem rušily). Rychlost přenášených dat je 10 Mbitů za sekundu. [3]

Mezi výhody koaxiálního kabelu patří velká odolnost proti vnějšímu elektromagnetickému rušení. Vykazuje poměrně dobré parametry při frekvencích pod 1 GHz. Další výhody koaxiálního kabelu je relativně snadná instalace a přiměřená cena.

Nevýhoda koaxiálního kabelu je náchylnost k poškození. Také má horší vlastnosti v oblasti stínění v rozmezí 20KHz – 6MHz. Tento kabel nelze použít v sítích Token-Ring.

4.3 Optický kabel

Optické kabely jsou určené pro vysokorychlostní přenos internetu. Obsahují optická vlákna**,** přes které se data přenáší. Viditelné světlo se šíří velmi rychle a proto umožňuje velmi rychlý přenos dat. Přenášená číslicová data v optických kabelech můžeme reprezentovat pomocí světelných impulsů - přítomnost impulsu může představovat např. logickou 1, zatímco jeho nepřítomnost logickou 0. Pro praktickou realizaci potřebujeme ovšem celý optický přenosový systém, složený ze zdroje, přenosového média a přijímače. Zdroj neboli vysílač převádí elektrický signál na světelný a vysílá jej do vlákna. Vysílač musí obsahovat světelný zdroj například LED nebo laser. Přijímač se potom skládá z fotodetektoru, který převádí optický signál do elektrického tvaru, zesilovače, který zesiluje signál a převádí jej do tvaru připraveného pro zpracování, a procesoru, který reprodukuje původní signál. [3]

Výhoda optického kabelu je dosažení velké přenosové rychlosti. Kromě velké přenosové rychlosti je další velkou výhodou optických vláken jejich naprostá necitlivost vůči elektromagnetickému rušení (což je velmi důležité např. v průmyslových aplikacích). Mezi další výhody patří také velká bezpečnost proti odposlechu, malý průměr a malá hmotnost optických kabelů.

Nevýhoda optických kabelů je v poněkud problematickém spojování jednotlivých vláken. V současné době jsou technologie jejich lepení či svařování v praxi dostatečně zvládnuty. Optická vlákna jsou velmi citlivá na mechanické namáhání a ohyby.

5 Návrh sériového rozhraní

Tato kapitola se zabývá vlastním návrhem sériového rozhraní. Popisuji zde postup při návrhu sériového komunikačního rozhraní jako je vysílač a přijímač, převodu na proudovou smyčku a také způsob zadávání dat pro odesílání přes navržené sériové komunikační rozhraní.

5.1 Volba parametru a způsobu přenosu

Volba přenášených bitů

Pro přenos jsem si zvolil osm datových bitů. Tato velikost 1 Bytu mi pokryje základní ASCII tabulku. To znamená, že budu schopen přenést jakýkoliv znak z ASCII tabulky. Zvolený počet datových bitů bylo nutné rozšířit jednak o start bit, který upozorňuje přijímač na začínající přenos, a také je do přenosu zahrnut stop bit a paritní bit. Paritní bit je pro kontrolu přenosu. Stop bit určuje přijímači, že přenos je u konce.

Asynchronní přenos

Při asynchronním přenosu dat se synchronizace mezi vysílačem a přijímačem zabezpečuje předávání synchronizačního znaku. Tento synchronizační znak je označován jako start bit. Tento start bit je poslán před daty a zabezpečuje synchronizování přijímače dat s vysílačem.

Poloduplexní a fullduplexní přenos

Při zapojení přijímače na jedné straně a vysílače na druhé straně bude toto propojení fungovat v poloduplexním režimu. To znamená, že data jsou přenášena pouze jedním směrem po komunikační lince. Při zapojení vysílače a přijímače na obouch stranách může probíhat komunikace v plněduplexním režimu. Znamená to, že data se přenáší oběma směry po jedné komunikační lince.

Volba parity

Pro kontrolu úspěšného přenosu datových bitů jsem si zvolil použití parity. Nastavení paritního bitu lze využít sudou nebo lichou paritu. Z těchto dvou možností jsem vybral sudou paritu, přestože z hlediska kontroly přenosu nehraje volba parity žádnou roli. Liší se pouze v tom, jestli přenesené bity s hodnotou logické jedničky budou doplňovány do lichého nebo sudého počtu**.**

Kabel

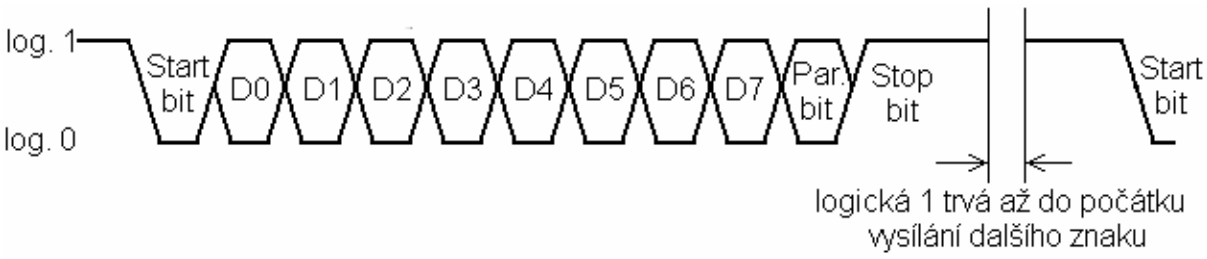
K propojení mezi vysílačem a přijímačem jsem si volil mezi koaxiálním kabelem, optickým kabelem a kroucenou dvojlinkou. Koaxiální kabel se ukázal pro moji potřebu jako příliš robustní a další jeho nevýhodou je velká náchylnost na porušení. Použití optického kabelu by bylo velice nákladné a zbytečně komplikované, protože k jeho použití je potřeba navíc pořídit vysílač do optického kabelu a poté přijímač. Na optický kabel bych navíc nemohl použít přenos využívající 20mA smyčku. Nakonec jsem tedy zvolil kroucenou dvojlinku a to z důvodu levného pořízení a snadného použití.

5.2 Návrh sériového komunikačního rozhraní

V této kapitole popisuji postup návrhu sériového komunikačního rozhraní. Celý návrh je rozdělen na tři části. V první části se zabývám způsobem popisem komunikace mezi vysílačem a přijímačem. Další dvě části jsou věnovány návrhu vysílače a přijímače.

5.2.1 Popis komunikace mezi vysílačem a přijímačem

Na začátku každého přenosu bude vyslán první bit, který nazývám start bit. Tento bit je nastaven na logickou „0“, protože na komunikační lince po dobu klidu je stále logická „1“ a má za úkol oznamovat přijímači začátek přenosu. Po odeslání start bitu následuje odeslání datových bitů, které zastupují přenášenou hodnotu, v mém případě vždy osm. Tento počet jsem si zvolil vzhledem k tomu, že na osmi bitech lze přenést každý základní znak ASCII tabulky. Po přenesení posledního osmého datového bitu bude následovat paritní bit. Paritní bit představuje základní kontrolu, že přenos datových bitů proběhl úspěšně. Z nabízených možností jsem se rozhodl využít pro kontrolu sudou paritu, která se mění během přenosu datových bitů tak, aby počet přenesých datových bitů v logické „1“ byl doplňován na sudý počet. Po ukončení přenosu přijímač zkontroluje, že součet datových bitů v logické „1“ a paritního bitu je roven sudému číslu. Po paritním bitu následuje poslední bit přenosu zvaný stop bit, který má hodnotu logické „1“. Tento bit upozorňuje, že přenos je u konce. Po ukončení přenosu čeká vysílač na odpověď přijímače, zda přenos proběhl správně. Průběh odeslání jednoho datového cyklu je uveden na obrázku 5.2.1.

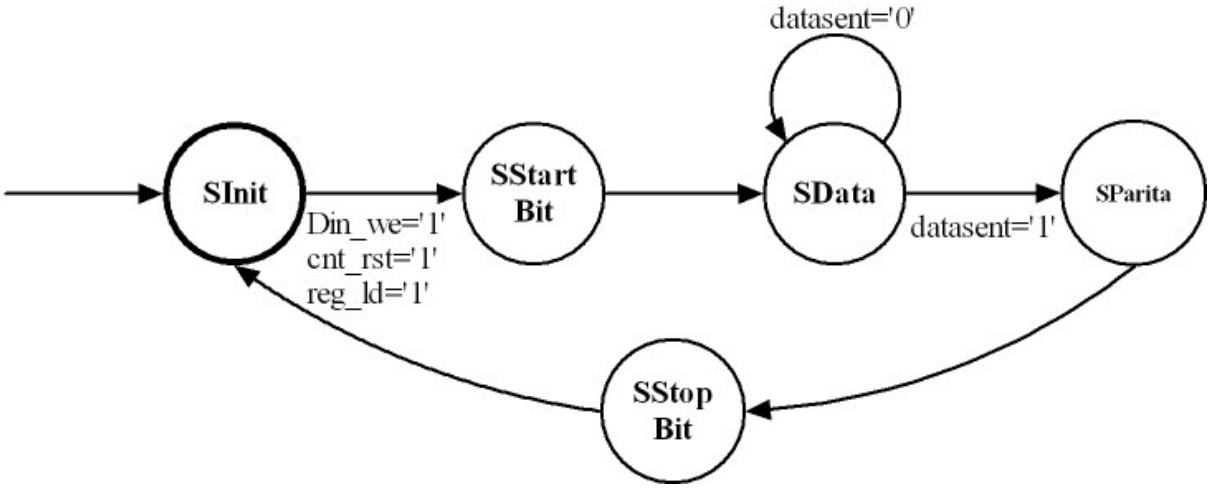


Obrázek 5.2.1 – Průběh odesílání

5.2.2 Návrh vysílače a popis jeho vlastností

Navržený výsílač se bude skládat z několika částí. Jednou částí je posuvný osmibitový registr, do kterého se ukládají data určená k odeslání. Další část vysílače je Moorův automat, který bude řídit proces odesílání a jeho definice je podstatná pro správnou posloupnost odesílaných dat. Poslední částí vysílače je multiplexer, který je nastavován pomocí automatu. Do vysílače bude vstupovat signál s daty, které má vysílač odeslat a signál určující platnost těchto dat. Z vysílače pak bude vést signál, který oznamuje, že vysílač je právě zaneprázdněn vysíláním a signál po kterém jsou posílána data přijímači.

Jak je již uvedeno, je pro funkčnost vysílače nutný správný návrh Moorova konečného automatu. Pro svůj úkol jsem nadefinoval pětistavový Moorův konečný automat viz obrázek 5.2.2.



Obrázek 5.2.2 – stavový automat vysílače

Popis jednotlivých stavů a jejich činností:

Stav SInit (počátečný a zároveň koncový)

Pokud se automat nachází v tomto stavu, znamená to, že automat čeká na potvrzení platnosti dat, které jsou na vstupu do vysílače. V případě, že jsou data na vstupu do vysílače platná, nastaví se signál, který povoluje zápis do posuvného registru. Zároveň se nastaví i signál informující o stavu vysílače, který, pokud se tento signál nachází v logické nule, je ve stavu čekání na data. V opačném případě je ve stavu odesílání dat. Při splnění podmínky se automat převede do následujícího stavu, který je označován SStartBit.

Stav SStartBit

Jestliže je automat v tomto stavu, znamená to, že data na vstupu byla potvrzena a může se začít s odesílaním dat. Tehdy se na sériovou linku nastaví logická nula, která reprezentuje start bit namísto dosavadní logické jedničky. Start bit upozorňuje přijímač na začátek přenosu. Ještě než se automat převede do dalšího stavu (SData), musí se restartovat čítač odeslaných bitů.

Stav SData

Automat se nachazí v tomto stavu tehdy, když se mají začít odesílat data uložená v posuvném registru nebo všechny data z registru nebyla doposud odeslána. Datový bit, který se nachází v nejnižším bitu posuvného registru, je právě odesílán. Po každém odeslání bitu se inkrementuje čítač a nasune se další datový bit na nejnižší bit registru. Pokud není hodnota čítače rovna sedmi odeslaných bitům, znamená to, že data se stále odesílají. Jakmile je na čítači počet odeslaných bitů roven sedmi, musí se automat připravit na další stav (SParita). Kdyby nebyl proveden tento krok před odesláním posledního bitu, došlo by k opětovnému odeslání prvního již odeslaného datového bitu.

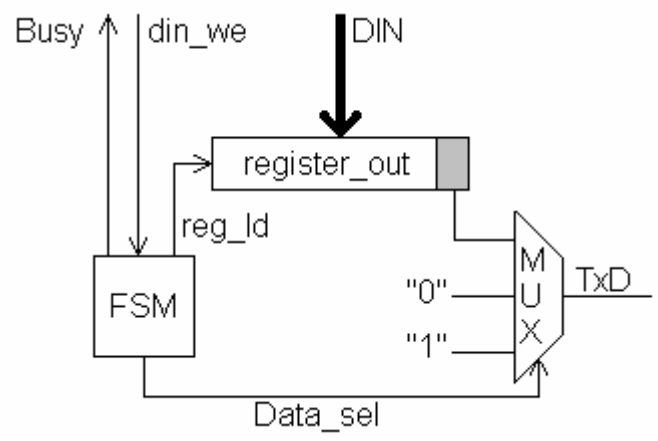
Stav SParita

Stav SParita je stav, který zabezpečuje odeslání kontrolního bitu nazvaný paritní bit. Tento paritní bit se v průběhu odesílání datových bitů mění a to tak, aby se počet odeslaných bitů doplnil do sudého počtu. Po odeslání tohoto kontrolního bitu se automat dostane do posledního stavu (SStopBit).

Stav SStopBit

Pokud je automat v tomto stavu, zamená to, že data uložená v posuvném registru a paritní bit byla úspěšně odeslána. Může se tedy ukončit přenost a to nastavením seriové linky do hodnoty, která zastupuje stav nečinosti. Zároveň se nastaví signál informující o stavu vysílače na hodnotu logická nula a automat se dostane do počátečního stavu označovaný (SInit).

Definováním automatu, který je součástí vysílače potřebného pro návrh sériového komunikačního rozhraní, je problém odesílání dat vyřešen. Schéma vysílače je znázorněno na obrázku 5.2.3

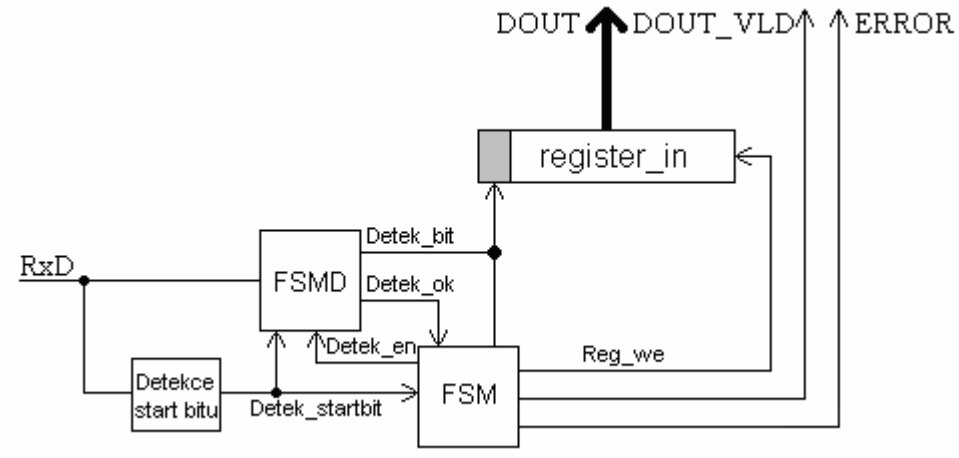


Obrázek. 5.2.3 – Schéma navrženého vysílače

5.2.3 Návrh přijímače a jeho vlastností

Navržený přijímač se skládá z několika částí. Jednou z nich je devítibitový registr pro uložení příchozích dat s kontrolním paritním bitem. Další částí přijímače je obvod detekující začátek přenosu, přičemž informací o době nečinosti je na vstupu přijímače logická jednička. Pokud se změní vstup na logickou nulu, značí to předpokladáný začatek přenosu (start bit). Dále přijímač obsahuje dva Moorovy automaty. Jeden z těchto automatů bude ovládat celý proces přijímání dat a ovládání přijímače, zatím co druhý automat obsluhuje detekci vstupního signálu. Jeho výstupem je bit detekovaný na vstupu a signál určující platnost detekování bitu.

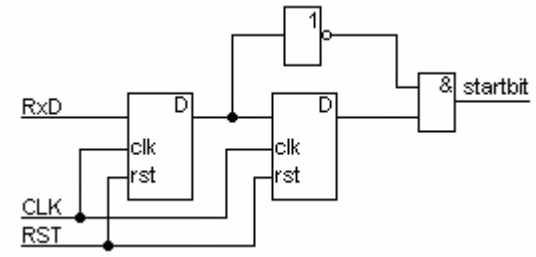
Vstupem do přijímače je pouze příchozí sériová linka. Na výstupech z přijímače najdeme data, která byla přijata, potvrzení platnosti přijatých dat a signál určující chybu, která mohla být způsobena přenosem. Schéma zapojení jednotlivých částí přijímače je uvedno na obrázku 5.2.4.



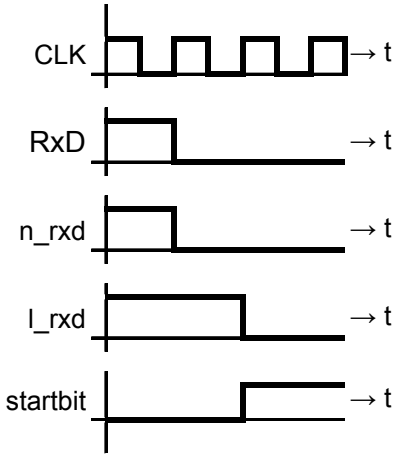
Obrázek 5.2.4 - Vnitřní zapojení přijímače

Princip detekce start bitu

V době nečinnosti je na vstupu přijímače nastavena logická jednička. Detekování start bitu je v zásadě zjištění sestupné hrany na vstupu, jinak řečeno změna z logické jedničky do logické nuly. Princip této detekce je velice jednoduchý. Detekování sestupné hrany je založeno na porovnání aktuální hodnoty na vstupu s předcházející uloženou hodnotou. Pokud je aktuální hodnota v logické nule, přičemž předcházející je v logické jedničce, značí to detekování sestupné hrany. Obvod pro detekci sestupné hrany na vstupu přijímače je uveden na obrázku 5.2.5 a časový průběh tohoto obvodu je znázorněn na obrázku 5.2.6.



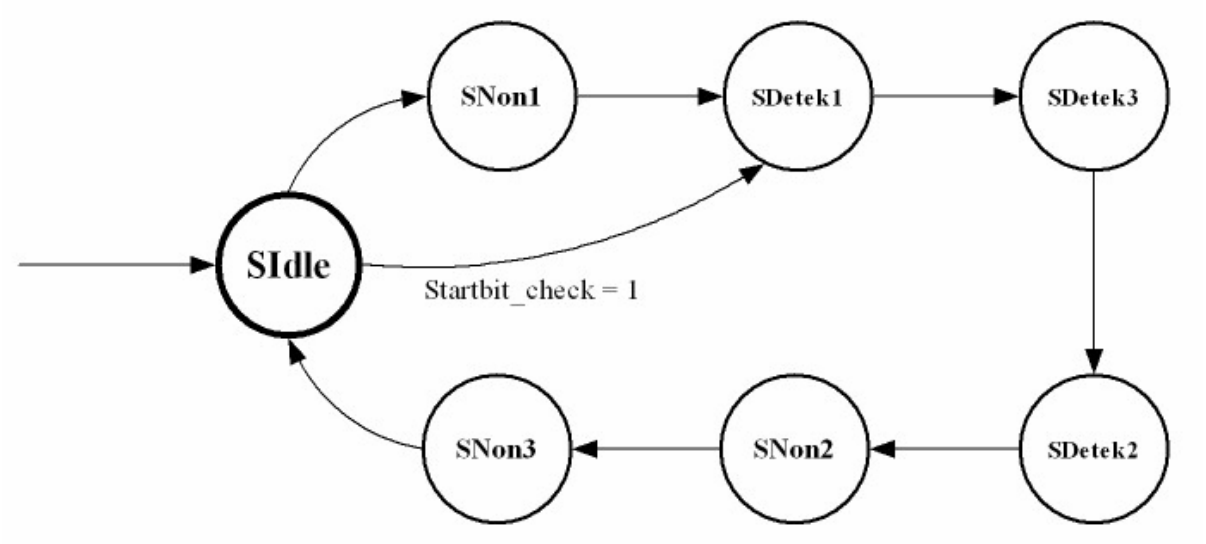
Obrázek 5.2.5 – Schéma zapojení obvodu pro detekci start bitu



Obrázek 5.2.6 - Časový průběh detekce a vyhodnocení start_bitu

Princip detekce bitu na vstupu přijímače

Pokud byla detekována sestupná hrana nebo je vysílač ve stavu přijímání, je zapojen do tohoto procesu automat pro detekci bitu a to i z důvodu toho, že detekovaná sestupná hrana nemusí znamenat pouze jen začátek přenosu (start bit), ale i chybu způsobenou na sériové lince, Stavový diagram automatu pro detekci bitu je znázorněn na obrázku 5.2.7.



Obrázek 5.2.7 – Automat na detekci bitu na vstupu přijímače

Stavy automatu na detekci bitu a popis jejich činností:

Stav SIdle (počátečný a zároveň koncový)

Pokud se automat nachází v tomto stavu znaméná to, že nebyl aktivován nebo je v počátečním stavu, kde začíná detekce bitu. Jestliže dojde k aktivaci pro ověření start bitu, přejde automat do stavu SDetek1. V opačném případě automat přejde do následujícího stavu SNon1. Přeskočením stavu SNon1 při ověřování start bitu předcházím zpoždění při detekci dalších bitů.

Stav SNon1

Tento stav nijak nezasahuje do detekce bitu, protože v tomto stavu se nezjišťuje hodnota na vstupu přijímače. V dalším kroku se automat nastaví na SDetek1.

Stav SDetek1

Stav SDetek1 uloží aktuální hodnotu ze vstupu přijímače a přejde do dalšího stavu (SDetek2).

Stav SDetek2

Automat v tomto stavu uloží aktuální hodnotu ze vstupu přijímače a přejde do dalšího stavu (SDetek3).

Stav SDetek3

V tomto stavu se porovnávají detekované hodnoty uložené ve stavech SDetek1 a SDetek2 s aktuální hodnotou na vstupu. Pokud se tyto hodnoty nerovnají, znamená to, že došlo k chybě při detekci a nastaví se signál informující o vyskytlé chybě. Hodnota detekovaného bitu se následně nastaví na aktualní hodnotu vstupu a signál informující o ukončené detekci na logickou jedničku. Automat přejde se do následujícího stavu SNon2

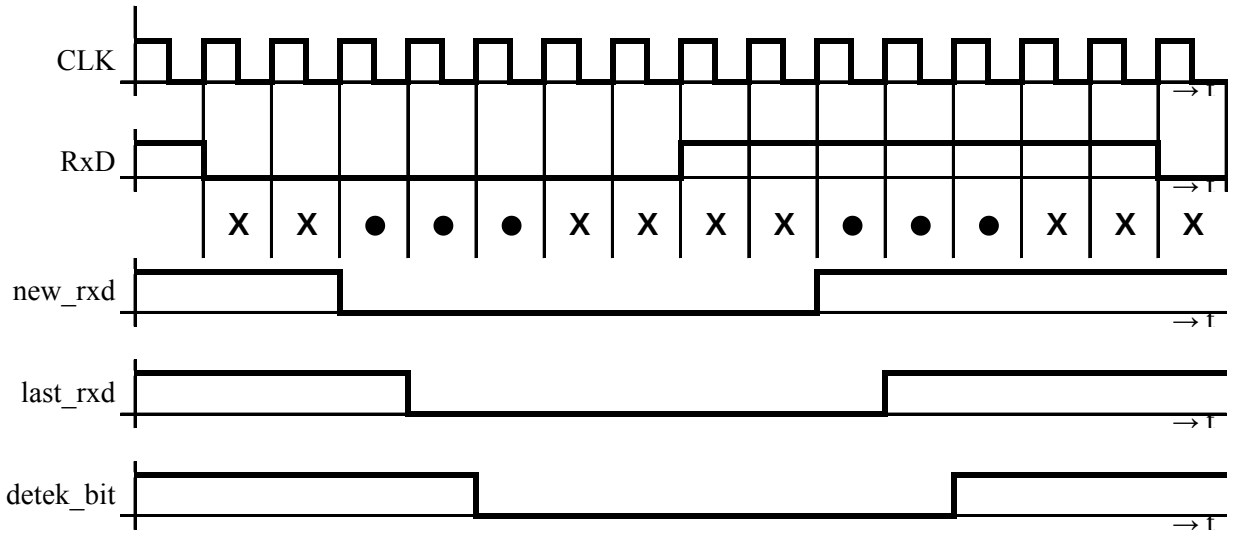
Stav SNon2

Tento stav nijak nezasahuje do detekce bitu, protože v tomto stavu se nezjišťuje hodnota na vstupu přijímače. V tomto stavu se nastaví další stav automatu (SNon3) a ponechá signál informující o stavu detekce na logické jedničce.

Stav SNon3

Tento stav nijak nezasahuje do detekce bitu, protože v tomto stavu se nezjišťuje hodnota na vstupu přijímače. V tomto stavu se nastaví další stav automatu (SIdle) a ponechá signál informující o stavu detekce na logické jedničce.

Stavy SNonx (kde x = 1,2,3) a stav SIdle jsou stavy automatu, ve kterých se nezjišťuje stav vstupu přijímače. Tyto stavy by měli pokrýt popřípadnou pozvolnou změnu vstupního signálu. Při detekci bitu dochází k menšímu zpoždění, který by neměl mít vliv na správnost přijímání. Příklad detekce a vyhodnocování vstupu přijímače je uveden na obr. 5.2.8.



Obrázek 5.2.8 – Princip detekce příchozího signálu

Legenda:

• - detekce příchozího signálu

x – nedělá nic

Stavy automatu pro ovládání příjmu bitu a popis jejich činností:

Stav SInit (počáteční a zároveň koncový stav)

Pokud je automat v tomto stavu, čeká se na detekci start bitu, aby mohlo dojít k zahájení přijímání dat. Pokud je detekován start bit, automat přejde do následujícího stavu SStartBit a aktivuje automat na detekci bitu, který tento start bit ověří.

Stav SStartBit

V tomto stavu automat čeká na ověření start bitu. Pokud automat pro detekci detekoval bit a má hodnotu logické nuly, jedná se o start bit. Jestliže je start bit ověřen, automat přejde do následujícího stavu SData, nastaví signál pro reset čítače, registru a erroru, a společně s těmito signály nastaví i signál povolující zápis do posuvného registru. V opačném případě je automat vrácen do počátečního stavu SInit.

Stav SData

Pokud se automat nachází v tomto stavu, znamená to, že se mají začít přijímat data. Data, které detekuje automat na detekci bitů, se ukládají do posuvného registru po jednotlivých bitech. Přijatý bit je uložen v nevyšším bitu posuvného registru. Po každém přijatém bitu se zvýší čítač přijatých bitů. Jestliže je na čítači hodnota menší než osm, znamená to, že se data teprve přijímají. Ale pokud má čítač hodnotu nastavenou na počet přijatých bitů rovnu osmi, znamená to, že se musí nastavovat signály potřebné pro převedení automatu do dalšího stavu. Tato odlišnost od vysílače, kde se nastavovalo o jeden bit dříve, je způsobena rychlejší činnosti přijímače než vysílače. Uložením posledního bitu se nastaví stav SParita a signál pro případnou chybu.

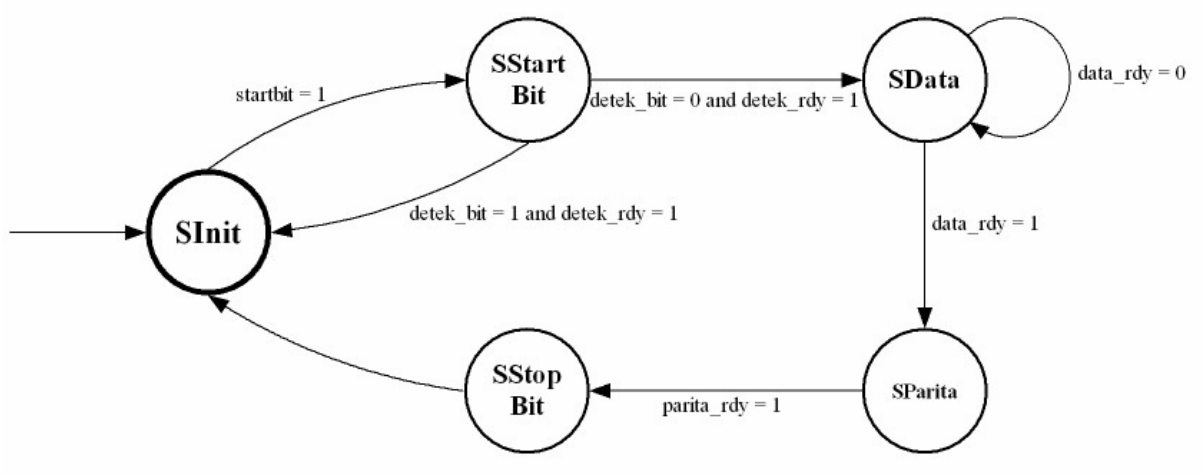
Stav SParita

V tomto stavu se uloží paritní bit do posuvného registru. Tento uložený paritní bit se porovná s vypočítaným paritním bitem zjištěný při přijímání jednotlivých bitů. Pokud se obě parity shodují, automat se převede do následujícího stavu SStopBit. Připadě neshody, to znamená, že při přenosu došlo k chybě a data jsou sice uložena v registru, ale nebudou zpřístupněna a automat po skončení přijímání oznámí chybu při přenosu.

Stav SStopBit

Stav StopBit signalizuje, že přijímač přijmul všechny data. V tomto stavu vyhodnotí zjištěné chyby při přenosu, a to jak chyby při nesprávné detekci bitu ze vstupu přijímače tak i při kontrole paritního bitu. Automat se vždy posléze nastaví do počátečního stavu SInit a pokud nenastala chyby ,povolí přečtení dat z registru. V opačném případě data nebudou povolena a nastaví se signál znázorňující chybu při přijímání.

Schéma automatu je znázorněno na obrázku 5.2.9.

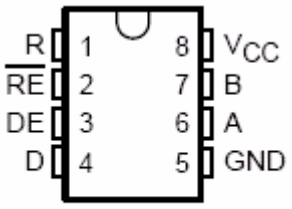


Obrázek 5.2.9. – stavový automat přijímače

5.3 Převod na proudovou smyčku

Na platformě FITkit jsou v mikrokontroleru umístěny převodníky D/A. Tyto převodníky nejsou dostačující z hlediska proudového i rychlostního převodu. Z těchto důvodů jsem musel navrhnout převodník z napětí na proudovou smyčku.

Jednou z možností, jak sestrojit převodník z napětí na proud, je koupit si v GM electronic diferenciální transceiver SN65176B (SN75176B).

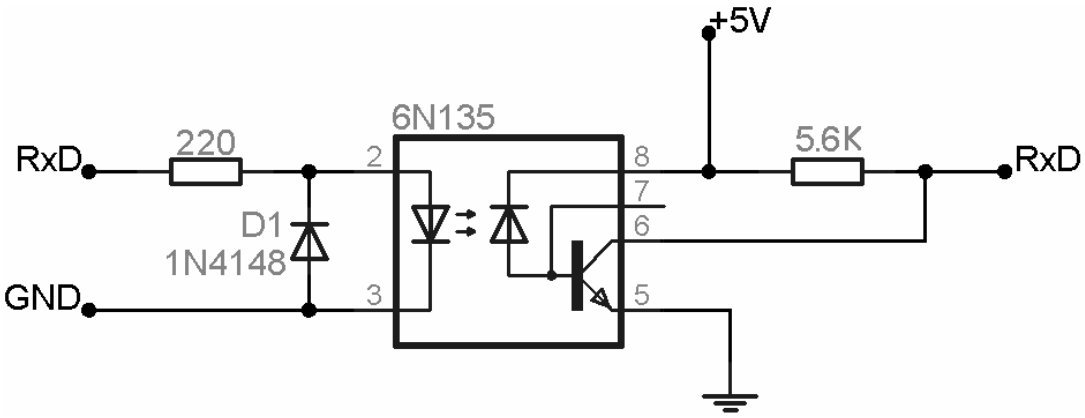


Obrázek 5.3.10

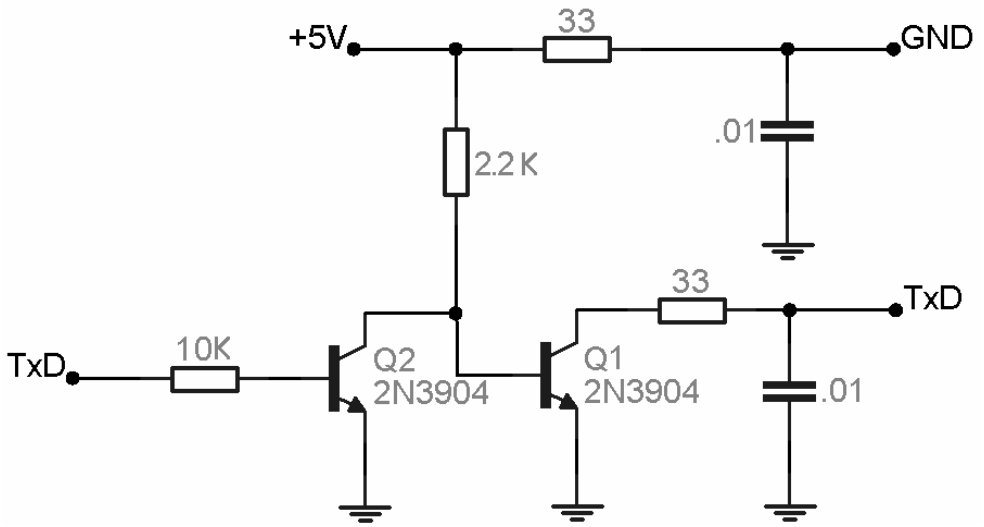
SN65176B a SN75176B diferenciálně sběrnicové přijímače/vysílače jsou integrované obvody navržené pro obojsměrnou komunikaci na vícebodových komunikačních sběrnicových linkách. Jsou navržené pro vyvážené komunikační linky a splňují ANSI standardy TIA/EIA-422-B a TIA/EIA-485-A, jako i ITU doporučení V.11 a X.27. SN65176B a SN75176B kombinují třístavový diferenciální linkový vysílač a linkový přijímač s diferenciálním vstupem. Oba prvky jsou napájené ze společného 5V zdroje. Vysílač a přijímač mohou být nastavené na aktivitu při vysokém nebo nízkém napětí, respektive mohou být externě spojené, aby fungovali jako kontrola řízení. Diferenciální výstupy vysílače a vstupy přijímače jsou interně připojené tak, aby vytvořili diferenciální vstupně/výstupní (I/O) sběrnicové porty, které jsou navržené, tak aby minimalizovali nahrávaní na sběrnici, když je vysílač vypnutý, anebo je VCC=0. Tyto porty obsahují širokou kladnou a zápornou škálu nastavení napětí, čímž je zařízení vhodné pro aplikace vyžadující skupinovou přípojku. Vysílač je navržený až na 60 mA pokles zdrojového proudu. Taktéž má kladné a záporné proudové omezení a teplotní vypínač na ochranu před chybami linky. Teplotní vypnutí je nastavené na hranici přibližně 150 °C.

Přijímač je nastavený na minimální impedanci 12K (ASI OHMOV), a citlivost vstupu má ±200 mV. typická hystereze vstupu je 50 mV. SN65176B a SN75176B může byt použit v komunikačních aplikacích využívajících SN75172 a SN75174 čtyřnásobné diferenciální linkové vysílače, SN75173 a SN75175 čtyřnásobné diferenciální linkové přijímače. [9]

Druhá možnost je sestavit si tento převodník z jednotlivých součástek (rezistor, kondensátor a diod). Toto zapojení je dostupné na http://www.cryogenius.com/hardware/sbmidi/, jen musíme dávat pozor, abychom správně zvolili velikost odporu rezistoru na straně přijímače jinak by mohla shořet světelná dioda v optočlenu. Zapojení převodníku pro přijímač je na obrázku 5.3.11 a zapojení převodníku pro vysílač je na obrázku. 5.3.12



Obrázek. 5.3.11 – Převodník na straně přijímače



Obrázek 5.3.12 - Převodník na straně vysílače

5.4 Zadávání a zobrazení dat

Prostředkem, kterým se budou zadávat data pro odeslání, je zvolena klávesnice umístěná na platformě FITkit. Na ní jsou umístěna tlačítka s označením 0-9, A, B, C, D, *, #. Po stisknutí tlačítka se vygeneruje jeho ASCII kód a odešle se zároveň do navrženého vysílače i na display, který je také součástí platformy FITkit. Display poté příslušný ASCII znak zobrazí.

U komponenty klávesnice je potřebné čtení pouze jednou za nějaký čas na základě stisknutí tlačítka. Aby se z řadiče nemusel stále číst stav a tím zjišťovat, zda jsou k dispozici data, využívá se systému přerušení. Systém přerušení se skládá z dvou částí a to z řadiče přerušení uvnitř *FPGA*, který zpracovává požadavky o přerušení od jednotlivých komponent uvnitř FPGA a na základě toho generuje žádost o přerušení pro *MCU*. Obsluha přerušení v *MCU* si při zpracování žádosti o přerušení přečte pomocí SPI vektor přerušení z řadiče uvnitř *FPGA* a podle jeho hodnoty zjistí zdroj žádosti. Podle stavu bitů tohoto vektoru se volají příslušné obslužné rutiny. Aby přerušení fungovalo tak jak má, je nutné na starší verzi platformě FITkit propojit pin JP10(5) (signál do MCU) s pinem JP9(26) (signál z FPGA).[1]

Pro zobrazení přijatého znaku, který přijímač příjmul, je použit display. Pro propojení mezi přijímačem a displayem není třeba propojení pinu na platformě FITkit. Po přijetí znaku se na display zobrazí přijatý znak, nebo chyba, která byla způsobena přenosem.

6 Implementace

V této kapitole je uveden postup při implementaci jednotlivých komponent pro komunikační rozhraní. Je zde také vysvětleno, co je to jazyk VHDL. K teoretickému ověření, zda implementované komponenty jsou správně implementovány, bude sloužit simulační program ModelSim XE. K syntéze pro FPGA je použit program Xiling ISE 9.1i. Všechny zmiňované programy pro úspěšnou realizaci jsou uvedeny na stránkách věnovaných platformě FITkit http://merlin.fit.vutbr.cz/FITkit/.

6.1 Jazyk VHDL

VHDL je jazyk pro popis hardwaru (VHSIC HDL – Very High Speed Integrated Circuit Hardware Description Language). Tento jazyk je standardizován organizací IEEE, což zaručuje širokou kompatibilitu a jednotnost. VHDL má široké výrazové možnosti. Jeho prostřednictvím lze jednak popisovat hardwarové prvky, dalšími jazykovými konstrukcemi se dají zapisovat obecné algoritmy, provádět testování hardware, jeho simulaci apod. VHDL kód lze tedy zhruba rozlišit na dva základní typy, kód syntetizovatelný a kód simulační. Syntézou rozumíme převod VHDL kódu do binárního předpisu, který je určen k naprogramování reálného hardware. Je však nutné podotknout, že existuje pouze omezená podmnožina jazykových konstrukcí, které je možné syntetizovat. Tato množina není pevná, závisí na překladači, na syntezátoru a také na tom, které konstrukce a jakým způsobem je schopen reflektovat hardware. Existují ustálené typické zápisy či konstrukce hardwarových prvků, které syntezátory bezpečně poznají. [3]

6.2 Postup při implementaci

Před samotnou implementací a návrhem jsem se musel nejprve seznámit se sériovým komunikačním rozhraním i s jeho variantami. Mezi tyto varianty patří RS-232, RS-422, RS-485 a proudová smyčka. Dále jsem musel získat informace o způsobu odesílání dat po sériové lince, počtu odesílaných bitů včetně volby velikosti dat a možnými variantami zabezpečení přenosu po zmiňované lince. Dalším krokem byl postup při implementaci. Celou implementaci jsem rozdělil do několika částí. V první části jsem se zabýval návrhem vysílače a poté přijímače. V další části jsem řešil problematiku zadávání dat pro odeslání a jejich zobrazení po přijetí. A nakonec jsem navrhnul a realizoval převodník na proudovou smyčku a naopak.

6.2.1 Implementace vysílače

Při návrhu vysílače jsem se zabýval způsobem odesílání jednotlivých bitů. Jedna z možnosti, která však nakonec nebyla použita, je využití dvou registrů ve vysílači. Jeden registr by byl na uložení vstupních dat, kdy po zjištění jejich parity se data uloží do výstupního registru, který bude obsahovat start bit, paritní bit a stop bit. Odesílání jednotlivých bitů by bylo řešené posuvným registrem. Počet odeslaných bitů by počítal čítač. Druhou možností, která byla i implementována, je využít k odesílání bitů automat. Tento automat bude řídit celý proces odesílání a proto bylo nutné si zvolit správný počet stavů, které tento automat bude mít. Postup při implementaci byl takový, že v prvním stavu (SInit) automat čeká na data a potvrzení jejich platnosti. Po splnění této podmínky se uloží data do posuvného registru a přejde se do dalšího stavu, kde by se začne s odesíláním dat. Prvním odeslaným bitem musí být vždy start bit, který bude upozorňovat přijímač na začátek přenosu, proto se tento stav nazývá SStartBit. Po jeho odeslání přijdou na řadu data uložena v registru. Odeslání těchto dat bude mít na starosti další stav automatu (SData). V tomto stavu se postupně odesílají data z posuvného registru tak, že se postupně nasouvají do nejnižšího bitu v registru, odkud se i odesílají. Pro zjištění počtu odeslaných bitů slouží čítač. Po odeslání těchto bitů se bude následovat stav SParita, kdy je odeslán kontrolní bit zvaný parita zaručující správnost přenosu. Tento paritní bit bude doplňovat počet odeslaných logických jedniček na sudý počet. Po tomto kontrolním bitu je odeslán stop bit, proto jsem nazval tento stav SStopBit. Stop bit upozorňuje přijímač na ukončení přenosu. Po odeslání stop bitu se automat vrací do prvního stavu. V klidovém stavu, kdy vysílač nevysílá, je na výstupu hodnota reprezentující logickou jedničkou. Ukázka části automatu a jeho logiky pro určení následujícího stavu:

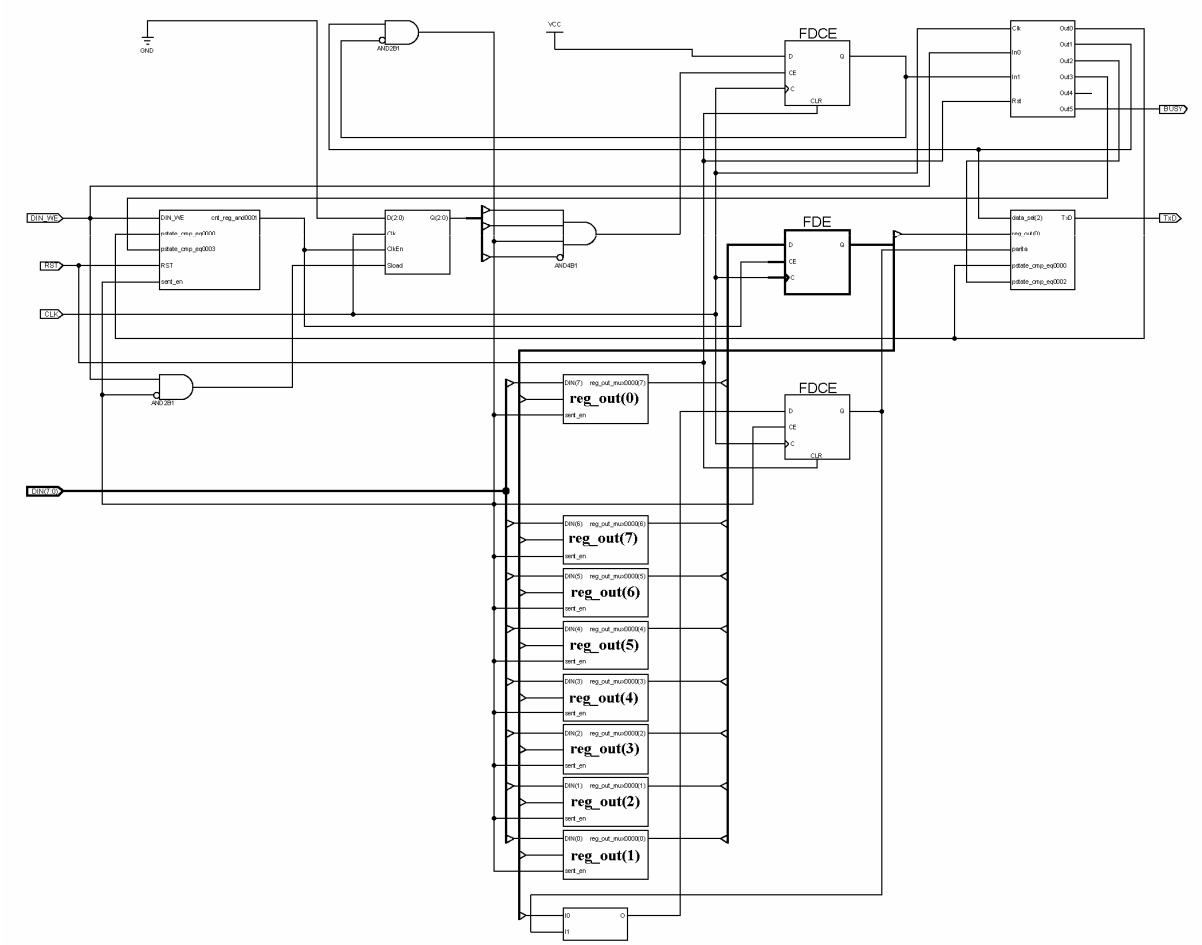
```
nstate <= SInit;
case pstate is
    when SInit =>
          …
          if din_we = '1' then
                nstate <= SStartBit;
                …
    when SStartBit =>
          nstate <= SData;
          …
    when SData =>
          if datasent = '1' then
                nstate <= SParita;
                …
```

```
                elsif datasent = '0' then
                    nstate <= SData;
                    …
                end if;
            when SParita =>
                nstate <= SStopBit;
            when SStopBit =>
                nstate <= SInit;
        end case;
```

Vysílač bude tedy obsahovat automat, jeden osmibitový posuvný registr, čítač odeslaných dat a multiplexer. Velikost zvoleného registru jsem zvolil proto, protože velikost dat osmi bitů by měla postačit pro znak z ASCII tabulky. Čítač odeslaných dat bude počitat počet odeslaných bitů. Multiplexer bude ovládán automatem a podle stavu automatu se bude nastavovat jeho činnost. Na multiplexer je připojen výstup z posuvného registru, logická jednička a logická nula. Do vysílače budou vstupovat data a signál potvrzující jejich platnost. Z vysílače bude vycházet sériová linka, po které se data budou odesílat, a signál ukazující zaneprázdněnost vysílače.
Ukázka definovaného vysílače v jazyku VHDL:

```
entity rs485_transmitter is
port(
   CLK : in std_logic;
   RST : in std_logic;
-- vstupy do vysilace
   DIN : in std_logic_vector(7 downto 0);-- data do vysilace
   DIN_WE : in std_logic;                -- platnost dat
-- vystupy z vysilace
   BUSY : out std_logic;                 -- vysilac zaneprazden
   TxD : out std_logic                   -- linka z vysilace
);
end rs485_transmitter;
```

V programu Xiling ISE 9.1i jsem vygeneroval zapojení mého navrženého vysílače, které je uvedeno na obrázku 6.2.1.



Obrázek 6.2.1 – RTL(register transfer logic) schéma navrženého vysílače

Abych mohl ověřit funkčnost vysílače, musím doimplementovat, jakým způsobem budu zadávat data, které chci odeslat. Nabízelo se zde několik možností. Jedna z možností bylo zadávat data na PC a posílat je přes USB do platformy FITkit, kde by byli předány vysílači. Další varianta bylo využití konektoru PS/2 a připojit k němu klávesnici nebo použít klávesnici umístěnou na platformě FITkit. Použití klávesnice mi přidalo nejlepší z hlediska využití platformy FITkit i přesto, že je zde omezení volby odesílaných znaků. Lze pouze odesílat znaky 0-9, A,B,C,D, * a #. Abych věděl, jaký znak posílám, využil jsem display, který je součástí této platformy. Tento display umí zobrazit základní znaky z ASCII (číslice, malá a velká písmena, speciální znaky) a to včetně čínských znaků.

Výchozí aplikace (soubor top_level.vhd) pro ovládání klávesnice pro FPGA ve VHDL je dostupná na internetových stránkách platformy FITkit a to včetně ovládání. Do této aplikace, která má již naimplementovanou klávesnici, jsem doplnil komponenty display a sériové rozhraní i s jejich řadiči, přes které jednotlivé komponenty spolupracují. Největším uskálím bylo správné propojení

jednotlivých signálů komponent. Výstup ze sériového rozhraní je připojen na PINHEADERS(JP10) s číslem 1. Také se musela upravit překladová dávka MAKEFILE.

Po doimplementování výchozí aplikace ve VHDL bylo nutné upravit program pro mikrokontroler určený pro ovládání klávesnice (soubor app_key_int.c) a to tak, že jsem přidal knihovnu pro display (display.h) a do funkce `void FPGA_keyboard_4x4_idle()` jsem doplnil funkci pro poslání znaku do displaye `send_char_display(ch)` a funkci pro posílaní znaku do sériového rozhraní `FPGA_SPI_RW_A8_D8(SPI_FPGA_ENABLE_WRITE, 0x70, ch)`. V hlavní funkci `void main()` jsem doplnil inicializaci displaye.

Pro ověření správnosti navržení a implementace vysílače jsem si jeho funkčnost odsimuloval v programu ModelSim XE a v příloze této diplomové práce je výstup ze simulace. Pro syntézu celého zapojení vysílače, klávesnice a displaye jsem použil program ISE 9.1i, podle kterého jsem určil kolik a jaké jednotky navržený vysílač bude potřebovat.

Výsledky syntézy:

```
Number of Slices:              143 out of 768  18%
Number of Slice Flip Flops:    192 out of 1536 12%
Number of 4 input LUTs:        228 out of 1536 14%
Number of IOs:                 124
Number of bonded IOBs:         122 out of 124  98%
Number of GCLKs:               1 out of 8      12%
```

6.2.2 Implementace přijímače

Před vlastním návrhem přijímače jsem se seznámil s různými možnostmi realizace přijímače. Jednou z možností jak přijímat data, která jsou posílána od vysílače je, že od začátku zahájení přijímání se všechny data (start bit, 8 bitů dat, parita, stop bit) přijmou do vstupního registru a poté se vyhodnotí. Tato varianta by však vedla k velkému zpoždění na straně přijímače. Další z možností, kterou jsem i zvolil je, že přijímač bude obsahovat vstupní detektor, automat na detekci bitu, automat na ovládání přijímání, devíti bitový posuvný registr a čítač. Vstupní detektor je určen k detekování start bitu a posuvný registr má velikost devíti bitů z důvodu ukládání dat i s příchozím paritním bitem. Abych věděl kolik dat jsem již příjmul, bude mi k tomuto účelu sloužit čítač přijatých dat. Přijímač bude mít dále vstupní signál od vysílače, jeho výstup s daty, signál informující o platnosti těchto dat a výstup, který informuje o výskytu chyby při přijímání. Přijímač v jazyce VHDL je definován takto:

```
entity rs485_receiver is
    port(
        CLK : in std_logic;
        RST : in std_logic;
    -- vystupy prijimace
         -- vystup z registru prijimace
        DOUT : out std_logic_vector(7 downto 0);
         -- signal informujici platnost dat
        DOUT_VLD : out std_logic;
         -- signal informujici o chybe během prijimani
        ERROR : out std_logic;
    -- vstup prijimace
         -- vstup prijimanych dat
        RxD : in std_logic
    );
end rs485_receiver;
```

Největším problémem při návrhu přijímače byl způsob detekovaní příchozích bitů. Řešením je detekovat vstupní signál několikrát rychleji než rychlost změny na vstupním signálu. Pokud by se detekovalo pomalu, mohlo by dojít k nesprávnému přijetí poslaných dat.

Vstupní detektor start bitu zjišťuje, kdy pravděpodobně začíná přenos od vysílače. Na vstupu vysílače v případě, že se nepřenáší data, je hodnota reprezentující logickou jedničku. Detekce start bitu je detekce sestupné hrany na vstupu. Detekce sestupné hrany je řešena vyhodnocováním aktuálního stavu a předchozího stavu a to pomocí dvou klopných obvodů typu D, které se většinou používají jako jednoduchá paměť. Realizace obvodu v jazyce VHDL je:

```
…
if startbit_en = '1' then
   n_rxd <= RxD; -- prirazeni akutalni hodnoty
   l_rxd <= n_rxd; -- predchazejici hodnota
   if (l_rxd = '1' and  n_rxd = '0') then
      …
   end if;
else
   …
end if;
```

Detekci příchozího bitu má na starosti automat na detekci příchozích bitů. Princip této detekce je založen na rychlosti synchronizačních hodin. V mém případě rychlost synchronizační hodin je sedmkrát rychlejší než synchronizační hodiny vysílače. To znamená, že při detekci jednoho bitu se spustí sedmkrát synchronizační hodiny přijímače, zatím co synchronizační hodiny vysílače pouze jednou. Tento automat je aktivován v případě, že je detekována sestupná hrana značící začátek přenosu, nebo probíhá příjem. Při návrhu tohoto automat jsem postupoval tak, že spouštění tohoto automatu řídí automat pro řízení přijímání. Dokud signál od automatu řídící přijímání není nastaven, je automat v počátečním stavu SIdle. Pokud byl signál nastaven, znamená to, že byla detekována sestupná hrana na vstupním signálu. Je-li detekován pravděpodobně start bit, je na tomto automatu ověřit, zda jde o start bit nebo nějakou chybu způsobenou rušením.

Ověřování start bitu probíhá tak, že přeskočením následujícího stavu (SNon1) se eliminuje zpoždění způsobené detekcí sestupné hrany. Tedy z počátečního stavu automat přejde do stavu SDetek1. V tomto stavu si uloží aktuální stav ze vstupu přijímače. V následujícím stavu SDetek2 po SDetek1 se uloží aktuální stav a také předcházející stav. Po tomto stavu následuje stav SDetek3, ve kterém se porovnává aktuální stav s uloženými hodnotami. Pokud jsou uložené hodnoty s aktuální hodnotou stejné, nastaví se výsledek detekování na tuto hodnotu. V opačném případě se nastaví signál oznamující, že se vyskytla chyba. Po tomto vyhodnocení následují stavy, ve kterých se nic neděje. Tyto stavy jsou zde zahrnuty, protože jsme na předpokládané změně signálu. Pokud by se signál změnil a šlo by o pozvolnou změnu signálu, dokážeme tuto pozvolnost eliminovat.

Při detekci dalších bitů, po ověření start bitu, pracuje automat v následující posloupnosti. Na začátku ve dvou stavech, ve kterých se nic neděje. Tyto dva stavy mají stejný význam, jako poslední stavy u detekce start bitu. Po těchto stavech se automat dostane do stavu SDetek1. Od této doby automat pracuje stejně, jako při ověřování start bitu. Ukázka logiky následujících stavů při detekci bitů v jazyku VHDL:

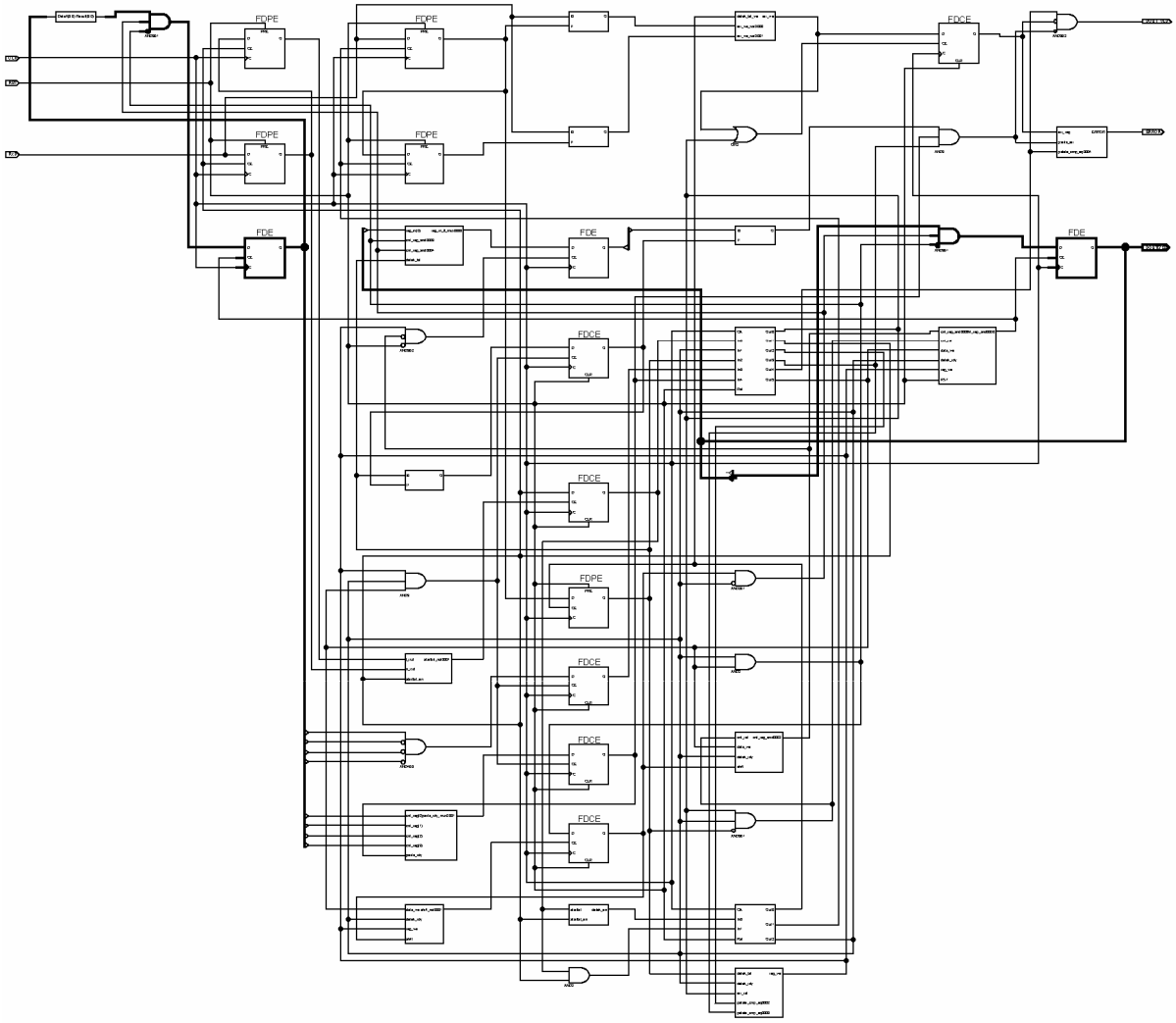
```
case astate is
         when SIdle =>
            if startbit_check = '1' then
               fstate <= SDetek1;
            else
               fstate <= SNon1;
            end if;
         when SNon1=>
            fstate <= SDetek1;
         when SDetek1 =>
            fstate <= SDetek2;
            …
```

```
        when SDetek2 =>
            fstate <= SDetek3;
            …
        when SDetek3 =>
            fstate <= SNon2;
        when SNon2 =>
            fstate <= SNon3;
        when SNon3 =>
            fstate <= SIdle;
        when others =>
             null;
    end case;
```

Při implementaci přijímače jsem si nejprve navrhnul automat, který bude celý proces přijímání ovládat. Nejprve jsem si definoval, co bude automat dělat v počátečním stavu (SInit). V tomto stavu mi automat povoluje detekci start bitu a čeká až detektor detekuje sestupnou hranu na vstupu přijímače. Po úspěšné detekci sestupné hrany se automat dostane do druhého stavu SStartBit a povolí automat, který má na starosti detekci start bitu. V tomto stavu čeká na ověření, zda je to start bit nebo ne. Pokud se opravdu detekoval start bit, začíná přenos a automat přejde do následujícího stavu a tím je stav SData a nastaví potřebné signály. Tyto signály slouží k resetu čítače a registru. V opačném případě se automat vrací do počátečního stavu. Ve stavu SData se začnou ukládat data do posuvného registru. Po každé detekci bitu se jednotlivé bity ukládají do tohoto registru. Abych věděl kolik bitů už mám uložených, slouží mi k tomuto účelu čítač. Při každém uložení bitu se čítač inkrementuje. Dokud není na čítači hodnota osm, znamená to, že data nejsou ještě uložena. Pokud je na čítači hodnota osm jsou data přijata a přejde se do dalšího stavu (SParita). V tomto stavu se mi uloží paritní bit a porovná se s vypočítaným paritním bitem při přijímání dat. Pokud se oba paritní bity shodují, přejde automat do dalšího stavu pojmenovaném SStopBit. Pokud se nerovnají, nastaví se signál signalizující chybu a přejde se také do stavu SStopBit. V tomto stavu se zjišťuje zda nedošlo během přenosu k nějaké chybě a to buď během detekce bitů nebo při porovnání parity. V těchto připadech je nastaven signál ERROR. Pokud není chyba zjištěna, je nastaven signál DOUT_VLD, které povolují přečtení dat z výstupu DOUT. Ukázka logiky následujících stavů při řízení přijímání v jazyku VHDL:

```
case pstate is
      when SInit =>
         …
         if startbit = '1' then
            …
            nstate <= SStartBit;
         end if;
      when SStartBit =>
         if detek_rdy = '1' then
            if detek_bit = '0' then
               …
               nstate <= SData;
             end if;
         else
            nstate <= SStartBit;
         end if;
      when SData =>
         …
         if data_rdy = '1' then
            nstate <= SParita;
         else
            nstate <= SData;
         end if;
      when SParita =>
         …
         if parita_rdy='1' then
            nstate <= SStopBit;
            …
            end if;
         else
            nstate <= SParita;
         end if;
      when SStopBit =>
          nstate <= SInit;
      when others =>
          null;
   end case;
```

V programu Xiling ISE 9.1i jsem si vygeneroval zapojení mého navrženého přijímače, které je uvedeno na obrázku 6.2.2.



Obrázek 6.2.2 – RTL(register transfer logic) schéma navrženého přijímače

Pro zobrazení přijatých dat jsem si zvolil, stejně jako v případě vysílače, display umístěný na platformě FITkit. Jako výchozí aplikaci pro propojení přijímače a displaye, jsem použil implementované rozhraní pro display(soubor top_level.vhd), který je dostupné na internetových stránkách. Do této aplikace bylo nutné doimplemetovat komponentu přijímače, řadič a přerušení, přes které budou komunikovat s displayem. Jelikož vysílač pracuje na 7MHz a přijímač musí pracovat 7x rychleji, aby mohl vzorkovat vstupní signál, doplní se do implemetováné aplikace generátor hodin. Tento generátor bude nastaven na rychlost 50MHz, které vyhuvuje uveden podmínce. Další úpravy byli pro mikrokontroler v souboru app_key_int.c, který byl součásti implementované klávesnice. Využitím této aplikace docílím již implementovaného přerušení. Do funkce USER_FPGA_interrupt se doplnila funkce, která předává display přijmuté znaky ze sériového rozhraní a funkce pro přijetí znaku ze sériového rozhraní.

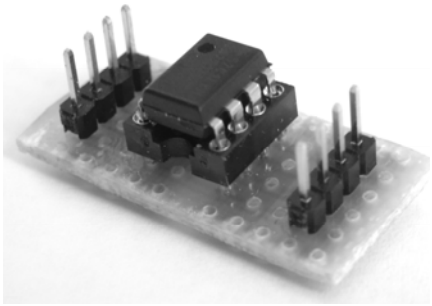
Pro teoretické ověření správnosti navržení a implementace jsem si funkčnost přijímače odsimuloval v programu ModelSim XE. Zprávný průběh přijímče je ověřen v progrmau ModelSim XE. Výstup ze simulace je přiložen jako příloha této diplomové práce. Pro syntézu celého zapojení přijímače a displaye jsem použil program ISE 9.1i, podle kterého jsem určil kolik a jaké jednotky navržený přijímač bude potřebovat.

Výsledky syntézy:

```
Number of Slices:              93  out of   768    12%
Number of Slice Flip Flops:    127  out of  1536   8%
Number of 4 input LUTs:        146  out of  1536   9%
Number of IOs:                 124
Number of bonded IOBs:         118  out of  124    95%
Number of GCLKs:               1  out of    8       12%
```

6.2.3 Proudová smyčka

Pro realizování proudové smyčky jsem si vybral použít diferenciální transceiver SN75176B dosahující hodnoty proudové smyčky až 60mA. Tento obvod nepotřebuje použit další komponenty. Napájení proudové smyčky se připojí na FITkitu na Xport J10 číslo pin J10(1) a uzemnění na pin J10 (3 nebo 4). Data z vysílače jsou vyvedeny s pin J10 (6) a data do přijímače jsou přivedena na tentýž pin J10(6).



Obrázek 6.2.3 – Převodník na 20mA smyčku

7 Závěr

Tématem této diplomové práce bylo vytvoření bezpečného propojení počítačů. Tato problematika zahrnuje celou řadu variant řešení a některá se již i běžně využívají. Cílem této práce bylo propojit dva počítače pomocí sériového komunikačního rozhraní RS-485 s požadavkem na odolnost vůči chybám a rušení při přenosu. Pro realizaci jsem zvolil výukovou platformu FITkit, která je používána na Fakultě informačních technologií a zároveň odbornou veřejností právě pro účely modelování a ověření funkčnosti návrhů.

Diplomová práce zahrnuje popis zařízení FITkit, které bylo použito pro realizaci navrženého řešení. Je zde detailně vysvětlena problematika sériové komunikace, a vzhledem k požadavkům na odolnost přenosu jsem uvedl typy a vlastnosti kabelů používaných pro propojení počítačů. Dále jsem se zaměřil na podrobný popis jednotlivých kroků návrhu řešení a způsobu implementace. Mnou navržené a implementované sériové rozhraní umožňuje dosažení maximální přenosové rychlostí 7Mbit/s. Tato hranice je určena rychlostí FPGA, která je 7MHz. Odolnost proti rušení je zajištěna 20mA smyčkou, která vykazuje dobré vlastnosti a splňuje kladené požadavky. Pro ověření správnosti přenosu je při komunikaci využita funkce odesílání kontrolního bitu. Navrhnutá konfigurace pro FPGA čip umožňuje jednosměrnou komunikaci.

Diplomová práce mi umožnila nahlédnout do problematiky procesu od návrhu až po vlastní implementaci pro konkrétní čip FPGA a získal jsem tak zkušenosti, jak postupovat při návrhu a vytváření hardwaru. Při postupu jednotlivými fázemi jsem se setkával s nejrůznějšími komplikacemi, jako například u platformy FITkit, které bylo nutné řešit s ohledem na další kroky procesu implementace. Jako stěžejní a zároveň pro mne nejnáročnější úkol bych označil dosažení úspěšné syntézy pro daný čip FPGA.

Tuto diplomovou práci je možné ještě dále rozšířit například spojením vysílače a přijímače do jednoho typu, takzvaného transceiveru. Takové řešení by vyžadovalo spojení automatů pro přijímání a odesílání. Dále by bylo možné změnit způsob zadávání dat pro odesílání, tedy využít možnosti připojit klávesnici přes PS/2 nebo USB a nebo rozšířit vlastnosti stávající klávesnice modulu FITkit tak, aby bylo možné ji použít jako například klávesnici u mobilního telefonu. V neposlední řadě je zde možnost doplnění o volbu rychlosti odesílání.

Projekt nenavazuje na ročníkový ani semestrální projekt

Literatura

[1] FUČÍK, Otto: *FITkit*. [online]. [cit. 16.5.2007].
Dostupná z WWW < http://merlin.fit.vutbr.cz/FITkit/ >

[2] KOCOUREK, Petr. *Přenos informace*. [online]. [cit. 16.5.2007].
Dostupný z WWW: < http://measure.feld.cvut.cz/groups/edu/pi/site_fieldbus.doc >

[3] *WIKIPEDIA* [online]. 2002 [cit. 2007-05-17]. Dostupný z WWW: < en.wikipedia.org >

[4] HLAVA, Jaroslav. *Prostředky automatického řízení II: analogové a číslicové regulátory, elektrické pohony, průmyslové komunikační systémy*, Vydavatelství ČVUT Fakulta strojní, listopad 2000

[5] PLHAL, Pavel. *Přenos dat*. [online]. [cit. 16.5.2007].
Dostupný z WWW < http://web.quick.cz/plhal/prendat1.htm >

[6] XILINX. *Spartan-3 FPGA Family: Complete Data Sheet*. [online]. [cit. 16.5.2007].
Dostupný z WWW < http://merlin.fit.vutbr.cz/FITkit/doc/hardware/datasheet/Spartan-3.pdf >

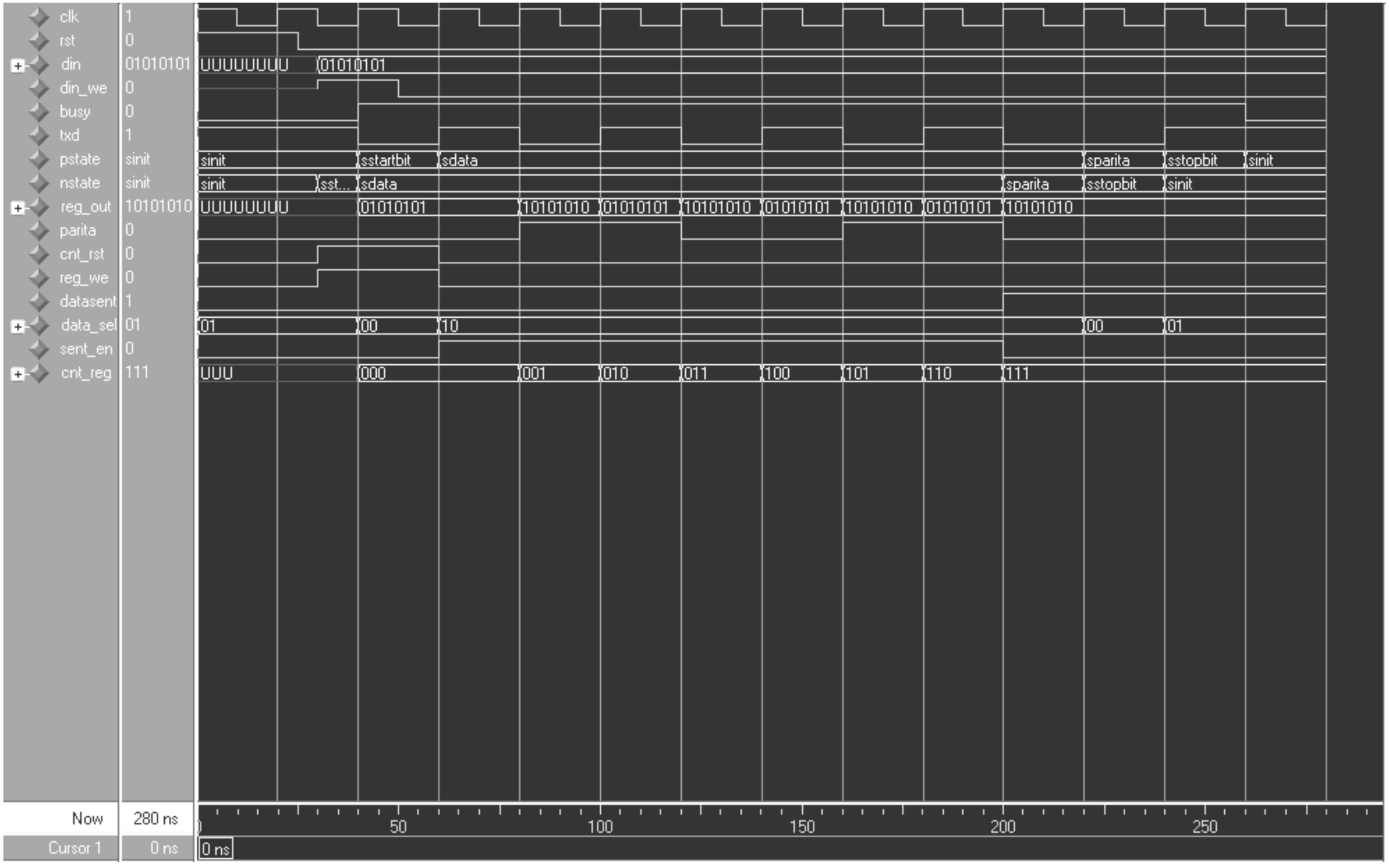
[7] TEXAS INSTRUMENTS. MSP430x15x, MSP430x16x, MSP430x161x, MIXED SIGNAL MICROCONTROLLER [online]. [cit. 16.5.2007].
Dostupný z WWW < http://focus.ti.com/docs/prod/folders/print/msp430f168.html >

[8] DOUŠA, Jiří. *Jazyk VHDL*, Vydavatelství ČVUT, 2003, ISBN 80-01-02670-1

[9] SN75176B DIFFERENTIAL BUS TRANSCEIVERS [online]. 1985 [cit. 2007-05-20]. Dostupný z WWW: < http://groups.engr.oregonstate.edu/ieee/store/datasheets/sn75176b.pdf >

Seznam příloh

Příloha A – Výstup simulace vysílače



Příloha B – Výstup simulace přijímače