VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

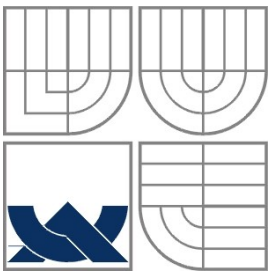
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PŘEVODNÍK MEZI SBĚRNICEMI I2C A RS232

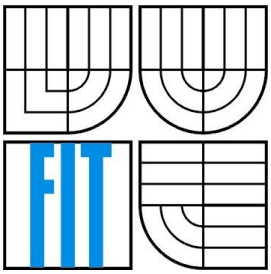
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Martin Votava
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PŘEVODNÍK MEZI SBĚRNICEMI I2C A RS232

I2C BUS TO RS232 CONVERTOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Martin Votava

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Filip Orság, Ph.D.

BRNO 2010

Abstrakt

Cílem této bakalářské práce je navrhnout a sestavit převodník mezi sběrnicemi RS-232 a I2C. Pomocí sběrnice I2C převodník ovládat bitové expandéry PCF8574.
Teoretická část popisuje sběrnice a expandér PCF8574. Dále se zabývá volbou vhodného mikrokontroléru a návrhem komunikačního protokolu.
Praktická část se věnuje tvorbě řídícího programu pro mikrokontrolér, vhodnému způsobu zpracování komunikačního protokolu a testováním převodníku. Také popisuje pomocné přípravky a programy pro testování.

Abstract

The main goal of this Bachelor's thesis is to design and create I2C bus to RS-232 convertor. Using I2C bus the convertor drives 8bit I/O expander.
In theoretical part buses I2C and RS-232 are described along with PCF8574. There is also mentioned selection appropriate microcontroller and design of comunication protocol.
Practical part is dedicated to creating control program for microcontroller, appropriate comunication protocol processing and the convertor testing. There are also described some auxiliary products and programs for testing.

Klíčová slova

RS232, I2C, převodník, komunikační protokol, PCF8574, ATMEL, AVR, ATmega8, WinAVR, AVRStudio

Keywords

RS232, I2C, convertor, comunication protocol, PCF8574, ATMEL, AVR, ATmega8, WinAVR, AVRStudio

Citace

Martin Votava: Převodník mezi sběrnicemi I2C a RS232, bakalářská práce, Brno, FIT VUT v Brně, 2010

Převodník mezi sběrnicemi RS232 a I2C

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Filipa Orsága, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Votava
14.5.2010

Poděkování

Rád bych poděkoval Ing. Filipu Orságovi, Ph.D. za poskytnuté rady a konzultaci.

Obsah

1 Úvod

S rozvojem elektroniky a jejím rozšiřováním do všech odvětví vzniká i potřeba přenášení signálů na dlouhé vzdálenosti. Ze dvou možných způsobů přenosu – sériového a paralelního – se jeví jako výhodnější sériový. Vyžaduje minimální počet vodičů, což je z ekonomického hlediska velmi důležitý faktor, a také není tak náchylný na elektromagnetické rušení. Další pozitivní vlastností je jednodušší připojení k řídícím prvkům (např. počítač). Převod mezi paralelním a sériovým formou přenosu se pak jeví jako velice důležitá část přenosu. Existuje velké množství obvodů, které převádějí paralelní data na sériová, avšak žádný z nich není schopen autonomně komunikovat pomocí některé ze standardizovaných sběrnic. Pro dosažení této vlastnosti je nutné použití mikrokontroléru s vhodným programem. A právě konstrukcí podobného převodníku se zabývá tato bakalářská práce.

Cílem této bakalářské práce je navrhnout, sestavit a oživit převodník, který bude schopen autonomní činnosti. Jeho úkolem má být detekce změn na vstupních pinech expandérů PCF8574 následovaná zasláním zprávy nadřazenému prvku nebo dalšímu expandéru ke zpracování. Dalším úkolem by mělo být zpracovávání příkazů přicházejících po RS-232 a jejich interpretace. Všechny součástky, ze kterých se bude převodník skládat, by měly být levné a lehce dostupné.

První kapitola této bakalářské práce je samotný úvod spolu se stanovením cílů. Druhá kapitola popisuje sběrnice RS-232 a I2C a také se zmiňuje o expandéru PCF8574. Ve třetí kapitole jsem se věnoval volbě vhodného mikrokontroléru. Čtvrtou kapitolu jsem věnoval návrhu vhodného komunikačního protokolu. Návrh a sestavení převodníku je zmíněno v kapitolách pět a šest. Pátá kapitola popisuje hardwarovou stránku, šestá software. Sedmá a osmá kapitola popisují pomocné přípravky, určené k testování převodníku a v deváté je popsáno vlastní testování. Shrnutí poznatků a závěr je kapitola desátá.

2 Popis sběrnic a expandéru

2.1 RS-232

Komunikační standard RS-232 (Recomended Standard) pochází z počátku šedesátých let. Byl navržen pro komunikaci dvou zařízení označovaných jako DTE (Data Terminal Equipment) a DCE (Data Circuit-terminating Equipment). Jako zařízení DTE byl většinou označován počítač či jiné řídící zařízení, DCE byl obvykle modem.

Při návrhu protokolu byl kladen důraz především na vysokou odolnost proti rušení a možnost vytvořit spojení mezi koncovými zařízeními na dlouhou vzdálenost. Proto jsou napěťové úrovně odlišné od běžné 5V logiky. Logická 1, označována jako marking state (značka), je přenášena zápornými hodnotami napětí, logická nula se nazývá space state (mezera) a přísluší jí kladné hodnoty napětí. Povolené úrovně napětí podle [1] jsou v tabulce 2.1.

Datové a řídící signály		
Úroveň	Vysílač	Přijímač
Log. 0	5V až 15V	3V až 25V
Log. 1	-5V až -15V	-3V až -25V

Tabulka 2.1: Napěťové úrovně RS-232

Přenos dat podle standardu RS-232 probíhá asynchronně s pevnou předem stanovenou přenosovou rychlostí. Každý byte začíná start-bitem, který má opačnou hodnotu oproti klidové úrovni tzn. log. 0, následují datové bity přenášené od LSB po MSB, případně i paritní bit, a ukončuje jej stop-bit, který je shodný s klidovou sekvencí. Sekvence je zobrazena na obrázku 2.1.

Obrázek 2.1: Časový diagram RS-232

Před zahájením komunikace je nutné nastavit parametry přenosu shodně na obou stranách. Nastavitelnými parametry jsou přenosová rychlost, počet datových bitů, použitá parita a počet stop-bitů. Možný počet datových bitů je v rozsahu 5 až 8 bitů, přičemž běžně se používají pouze

7 a především 8 bitové režimy. Standard určuje 5 základních režimů parity. Prvním režimem je úplné vynechání paritního bitu. Další dva mají hodnotu paritního bitu nastavenou napevno bez ohledu na přenášená data, buďto na značku nebo mezeru. Poslední dva režimy určují hodnotu paritního bitu podle právě odesílaného bytu. Lichá parita nastavuje bit na takovou hodnotu, aby byl celkový počet přenášených logických 1 v datové a paritní části lichou hodnotou. Sudá parita má naopak počet jedniček sudý. Počet stop-bitů je možné nastavit na 1, 1.5 nebo 2 bity. Vyšší počet stop-bitů byl používán především u pomalejších zařízení, aby jim poskytl dostatečnou časovou mezeru pro přesun právě přijaté hodnoty z posuvného registru do jiného bufferu. Jinak by totiž mohlo docházet ke ztrátám dat při příjmu.

Pro vkládání čekacích stavů definuje sběrnice několik řídících vodičů, které signalizují druhé straně stav zařízení a případnou připravenost k příjmu/vysílání dat. Mimo to sběrnice obsahuje i několik informačních vodičů. Jejich popis a význam podle [1] je zobrazen v tabulce 2.2. Literatura [9] uvadí u některých vodičů trochu jiný význam. Toto je způsobeno postupným vývojem rozhraní a přechodem od half-duplex komunikace směrem k full-duplex. Proto pominula potřeba definovat směr přenosu a na některé vodiče se začalo pohlížet jinak.

Název	Pin (Can 9)	Směr	Popis
CD	1	Vstup	Detekce nosného kmitočtu na telefonní lince
RXD	2	Vstup	Příjem dat ze zařízení DCE do DTE (terminál)
TXD	3	Výstup	Vysílání dat ze zařízení DTE do DCE
DTR	4	Výstup	DTE signalizuje, že je připraven komunikovat
GND	5		Signálová zem
DSR	6	Vstup	DCE signalizuje, že je připraven komunikovat
RTS	7	Výstup	DTE signalizuje, že je komunikační cesta volná
CTS	8	Vstup	DCE signalizuje, že je komunikační cesta volná
RI	9	Vstup	Detekce vyzvánění na telefonní lince

Tabulka 2.2: Přehled signálů RS-232

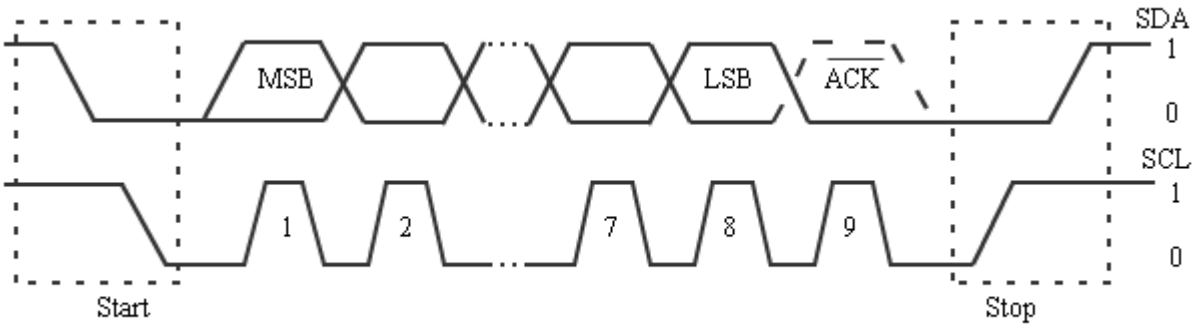
Ačkoliv byl protokol původně navržen pro komunikaci řídícího zařízení s řízeným, časem se objevila potřeba spojit i zařízení na stejné úrovni. Zde se vyskytl především problém na hardwarové úrovni. Proto se postupně objevily tři různé verze kabelů. Kabely sloužící k propojení zařízení DTE s DCE se nazývají „Straight Cable“ a jejich vývody jsou propojeny 1:1. Pro propojení zařízení DTE s DTE slouží kabel „Null-Modem“, který má zkřížené páry vodičů a některé nejsou propojeny vůbec. Stejně zapojený pouze s opačnými konektory je kabel „Tail Circuit“ sloužící k propojení DCE s DCE. Při propojování zařízení je navíc nutné dbát na shodný zemnící potenciál, případně použít galvanicky oddělený kabel.

2.2 I2C

Sběrnice I2C pochází z počátků 80. let z dílen NXP Semiconductors [2]. Jedná se o jednoduchou obousměrnou dvouvodičovou sběrnici. Jelikož je sběrnice synchronní, jeden z vodičů, označováný SCL, slouží k synchronizace, po druhém, pojmenovaném SDA, jsou přenášena data. Obě linky jsou vybaveny pull-up rezistory a každé zařízení na ně připojené je v provedení open-drain, neboli s otevřeným kolektorem u bipolární technologie. Tím je zajištěno, že kterékoliv zařízení může v libovolném okamžiku ovlivnit hodnotu na sběrnici bez poškození jiného prvku.

Na sběrnici je připojeno minimálně jedno zařízení typu Master a minimálně jedno typu Slave. Zařízení typu Master řídí komunikaci po sběrnici a generuje hodinový signál. Jedná se většinou o mikrokontrolér. Slave je většinou nějaké periferní zařízení a reaguje na příkazy a data od Master.

V klidové úrovni jsou oba vodiče v logické 1. Komunikaci zahajuje Master tzv. start-sekvencí. Během vysoké úrovně SCL stáhne Master SDA do nízké úrovně a následně i SCL. Poté vyšle sedmibitovou adresu od MSB po LSB, osmým bitem určí směr další komunikace (log. 0 – zápis do Slave, log. 1 – čtení ze Slave). Během devátého hodinového pulsu ponechá Master vodič SDA ve vysoké úrovni. Rozpoznalo-li nějaké Slave zařízení, že je adresováno, a je schopné komunikovat, stáhne SDA do nízké úrovně. Tento bit se nazývá ACK (Acknowledge) a následuje za každým bytem. Je-li ACK v nízké úrovni, značí to, že je zařízení schopno komunikovat, a Master pokračuje, v opačném případě ukončí komunikaci. Bit ACK je součástí každého přenášeného bytu. Komunikace je ukončena vysláním stop-sekvence. Během vysoké úrovně SCL uvolní vodič SDA. Celá sekvence je zobrazena na obrázku 2.2.



Obrázek 2.2: Časový diagram sběrnice I2C

2.3 PCF 8574

Obvod PCF 8574 je 8-bitový expandér řízený po sběrnici I2C. Obvod je vybaven 8 kvazi-obousměrnými piny. Kvazi-obousměrný pin je takový pin, který může být provozován jako vstupní

i výstupní pin zároveň aniž by byl předem určen směr tohoto pinu. Z tohoto důvodu je každý pin konstruován jako výstup s otevřeným kolektorem, který je vybaven pull-up odporem. Toto řešení má velikou nevýhodu. Za určitých okolností může být pin, který je uvažován jako výstupní a ve vysoké logické úrovní, přetažen nízkoimpedančním vstupem následujícího obvodu do nízké úrovně a tím k chybné interpretaci stavu pinu. Toto řešení také neumožňuje, aby byl vstupní pin ve stavu vysoké impedance, vždy se projeví vliv pull-up rezistoru.

Pro komunikaci s řídícím mikrokontrolérem je obvod vybaven sběrnicí I2C. Pro připojení více zařízení na jednu sběrnici jsou vyvedeny nejnižší tři bity adresy obvodu na piny jeho pouzdra a umožňují volbu adresy. Tímto způsobem lze na jednu sběrnici připojit až osm zařízení. Dalších osm zařízení je pak možné připojit ve formě obvodů PCF8574A, které mají pouze pozměněnou adresu. Vzhledem k hardwarovému provedení portů vyžaduje řízení pouze dvě operace – zápis do registru ovládajícího výstupní tranzistory a čtení stavu pinů. Vlastní komunikace je tedy jednoduchá. Nejprve dojde k naadresování obvodu a poté pouze k zapsání nebo přečtení hodnoty.

Vzhledem k principům sběrnice není možné, aby obvod sám o sobě kontaktoval nadřazené zařízení, detekoval-li změnu vstupů. Pro takovouto situaci je obvod vybaven přerušovacím výstupem. Přechodem do nízké úrovně je signalizováno, že došlo ke změně vstupních hodnot a je vhodný zásah z řídící strany. Vývod přejde zpět do vysoké úrovně až po té, co je daný obvod naadresován a je přečtena nová hodnota. Hardwarové provedení je shodné jako u ostatních vývodů sběrnice, tj. výstup s otevřeným kolektorem. Díky tomu je možné spojit výstupy všech obvodů „montážním součinem“.

3 Volba mikrokontroléru

V dnešní době existují na trhu tři nejznámější výrobci mikrokontrolerů: Microchip, Atmel a Motorola. Mikrokotroléry prvních dvou výrobců jsou založeny na architektuře Harward, naproti tomu třetí na architektuře von Neumann. Zadáním je omezen výběr pouze na firmu Atmel. Atmel nabízí výběr mezi 8-bitovými až 32-bitovými mikrokontroléry označované jako ATiny, ATmega a ATXmega. Také vyrábí různé verze mikrokontrolérů postavených na jádře 8051. Při výběru jsem zohledňoval cenu, dostupnost a také hardware, kterým je mikrokontorlér vybaven. Jako nejvýhodnější jsem nakonec určit rodinu AVR ATmega.

3.1 Rodina AVR ATmega

Mikrokontroléry rodiny ATmega jsou 8-bitové jednočipy založené na architektuře RISC (Reduced Instruction Set Computer). I tak má programátor k dispozici 131 instrukcí a 3 základní adresovací režimy – přímá hodnota, přímá adresa a nepřímá adresa. Jádro všech mikrokontrolérů je totožné, jednotlivé verze se liší pouze pouzdrem, velikostí pamětí a integrovaným hardwarem.

3.2 ATmega8-20PU

Mikrokontroler AVR Atmega8 je osazen do dvacetipinového pouzdra DIP (Dual Inline Package). Oproti zbývajícím provedení v pouzdrech TQFP a MLF mu schází pouze dva vstupy A/D převodníku. Mimo jiné obsahuje jeden 16-bitový a dva 8-bitové čítače, komunikační jednotky USART, SPI a TWI, analogově-digitální převodník a analogový komparátor. Podrobný popis mikrokontroléru vychází z [3] a částečně z [6].

3.2.1 Jádro mikrokontroléru

Mikrokontrolér je postaven na jádru RISC. Tato jádra jsou specifická vysokým množstvím pracovních registrů, redukovanou instrukční sadou a rychlým zpracováním instrukcí (většina v jednom hodinovém cyklu). Jádro obsahuje registrové pole skládající se z 32 všobecně použitelných registrů 8-bitové šířky. Každý z těchto registrů může být jak zdrojovým tak cílovým operandem libovolné aritmetickologické operace s vyjímkou instrukcí obsahujících zdrojový operand zadaný ve formě přímé hodnoty. V takovémto případě je nutné, aby cílovým operandem byl jeden z horních 16 registrů, tzn. registr r16 až r31. Pro techniku nepřímého adresování jsou vyhrazeny tři dvojice registrů (r27:r26, r29:r28 a r31:r30), které jsou pojmenovány jako registry X,Y,Z a slouží

jako ukazatele do paměťového prostoru. Tyto ukazatele nejsou rovnocenné. Pouze u ukazatelů Y a Z lze použít techniky post-inkrementace, pre-dekrementace a přičtení offsetu. A pouze ukazatel Z může být použit pro určení cíle v paměti programu pro načítání a ukládání do této paměti, případně pro nepodmíněné absolutní skoky či volání podprogramů.

3.2.2 Časování

Pro generování hodinového signálu pro jádro mikrokontroléru je možné zvolit jeden ze 4 nezávislých zdrojů. Prvním z nich je integrovaný RC oscilátor. Oscilátor generuje základní signál o frekvenci 8MHz ale pomocí děličky lze vybrat i nižší hodnoty a to 4MHz, 2MHz nebo 1MHz. Pro zvýšení přesnosti RC oscilátoru obsahuje řídící logika kalibrační registr, jehož hodnotu lze upravovat. Každý obvod má však od výrobce zvolenou a nastavenou nejoptimálnější hodnotu a většinou není nutné ji již dále opravovat. Je-li použit tento způsob časování, je možné využít piny XTAL1 a XTAL2 jako všeobecně použitelné vstupy/výstup portu PB.

Druhým zdrojem, který lze použít, je externí zdroj hodinového signálu. Tento se připojuje na pin XTAL1 a jeho hodnota musí být v rozmezí stanoveném výrobcem. Pin XTAL2 je opět možné použít jako všeobecně použitelný pin.

Zbylé dva režimy již vyžadují oba vývody. Jsou to režimy, při kterých je použit buďto externí krystal v kombinaci se zabudovaným oscilátorem, případně místo krystalu keramický rezonátor. Při použití těchto dvou režimů vyžaduje mikrokontorlér, aby došlo k přibližné specifikaci hodinového kmitočtu příslušných součástek při programování obvodu. V opačném případě nemusí fungovat časování korektně. Volba hodinového signálu se provádí během programování speciálními bity, tzv. propojkami.

3.2.3 Paměťový prostor

Rodina ATmega má adresový prostor datové paměti rozdělen do tří částí. Na nejnižší adresy jsou namapovány pracovní registry. Pracovní registry lze adresovat dvěmi způsoby. Jsou-li adresovány pomocí jejich názvů, přístup k nim je velice rychlý a lze je používat k aritmeticko-logickým operacím. Jsou-li však adresovány jako paměťové buňky, přístup je pomalejší. Následuje 64 speciálních funkčních registrů, které slouží k ovládání integrovaného hardware. Tyto registry jsou namapovány na adresy 0x0020 až 0x005F a lze je adresovat také dvěma způsoby. Instrukcemi *In* a *Out* nebo jako paměťové buňky. Až na adrese 0x0060 začíná operační paměť. Přístup k této paměti trvá dva hodinové cykly a je možný pouze pomocí instrukcí *Load* a *Store*.

3.2.4 Přerušovací systém

Přerušení umožňuje pozastavení právě vykonávaného programu a obsloužit požadavek, který vyžaduje neodkladný zásah. Při přechodu do obslužné rutiny přerušení je nutné uložit informace o stavu důležitých registrů, aby bylo možné později pokračovat v přerušené činnosti. U mikrokontrolérů řady ATmega se jedná minimálně o obsah registru PC (programový čítač instrukcí), stavový registr SREG a všechny registry, jejichž hodnota může být změněna.

K povolení přerušení je nutné provést dva kroky. V první řadě je nutné nastavit bit I v registru SREG, který ovlivňuje všechny zdroje přerušení. Každý zdroj přerušení má ještě svůj vlastní ovládací bit v některém z funkčních registrů. I ten musí být nastaven, aby jádro zaregistrovalo žádost o přerušení.

Po detekci žádosti jádro uloží pouze obsah registru PC, který obsahuje návratovou adresu, a zakáže další přerušení vynulováním bitu I. Zálohování ostatních registrů musí zařídit obsluha přerušení sama. Adresy pro obslužné rutiny jsou pevně stanoveny výrobcem a jsou umístěny na počáteční adresy paměti programu. Každému zdroji požadavku na přerušení odpovídá jedna položka. Na tuto adresu se typicky ukládá instrukce skoku do obslužné rutiny. Přesné adresy jednotlivých vektorů lze nalézt v [3].

3.2.5 Vstupně výstupní porty

Pro komunikaci s okolím je mikrokontrolér vybaven množstvím plně obousměrných vývodů rozdělených do skupin označovaných jako porty. Zpravidla obsahuje port 8 pinů, ale ne vždy je toto dodrženo. Vzhledem k počtu vývodů pouzdra má mikrokontrolér pouze dva plnohodnotné porty pojmenované PB a PD a jeden obsahující pouze šest pinů označený jako PC. Navíc nejvyšší pin je využíván jako resetovací vstup.

Každý port je řízen třemi registry. Registr DDRx určuje směr jednotlivých pinů, kdy log. 1 označuje výstupní pin, log. 0 vstupní. Význam registru PORTx závisí na směru příslušného pinu. Je-li pin nastaven jako výstupní, pak se pomocí registru PORTx určuje hodnota pin, u vstupního pinu určuje, bude-li připojen pull-up odpor. Funkce registru PINx nezávisí na směru a vždy odráží hodnotu přečtenou z portu. Zde je nutné poznamenat, že kvůli synchronizaci vzniká určité zpoždění mezí tím, kdy je hodnota na port zapsána a kdy se projeví změna v registru PINx. Přenesení změny přes synchronizační obvod trvá jeden hodinový takt.

Některé piny jsou vybaveny alternativními speciálními funkcemi. To znamená U velké části pak dojde k odpojení těchto vývodů od řídících registrů portu a jsou ovládány pouze příslušným hardwarem.

3.2.6 Čítač / časovač

ATmega8 obsahuje dva 8-bitové a jeden 16-bitový čítač. Hlavní činností čítače je počítání impulsů pocházejících z jádra mikrokontroléru nebo externího zdroje. Obecně umí čítače pracovat v několika různých režimech.

Čítač 0 je jednoduchý 8-bitový čítač umožňující pouze opakující se vzestupné čítání. Při přetečení umožňuje generovat přerušení. Jako zdroj hodinového signálu může být výstup předděličky taktovacího signálu jádra, případně vzestupná nebo sestupná hrana vnějšího zdroje.

K ovládání čítače slouží registr TCCR0. Jediné jeho 3 bity vybírají zdroj signálu, případně zastavují čítač. K indikaci přetečení čítače slouží jeden společný registr pro všechny čítače TIFR. Povolení přerušení od těchto příznaků se provádí v registru TIMSK. Bity odpovídají 1:1 příznakům v registru TIFR.

Čítač 1 je 16-bitový a oproti čítači 0 má více funkčních režimů. Kromě základního režimu prostého čítání je vybaven komparátory pro generování PWM nebo pro záchyt vnějších událostí. Generování PWM ani záchyt vnějších událostí nejsou předmětem této bakalářské práce, proto jejich popis zde chybí.

Rozšířením oproti čítači 0 je mimo jiné režim CTC (Clear Timer on Compare), ve kterém je hodnota volně běžícího čítače porovnávána s registrem OCR1A případně ICT a v případě shody je hodnota volně běžícího čítače vynulována. Tento režim umožňuje přesnější rozlišení délky generovaných časových úseků. Také je vybaven jednotkami pro generování PWM signálu.

Čítač 2 je z velké části shodný s čítačem 0. Navíc má pouze jednotku komparátoru pro generování PWM a také možnost asynchronní činnosti. Pro asynchronní činnost je nutné připojit externí krystal k vývodům oscilátoru, které jsou shodné s generátorem hodinového signálu pro jádro. Proto musí jádro využívat integrovaný RC oscilátor, aby bylo možné tento režim zvolit.

3.2.7 EEPROM

Paměť EEPROM je řešena formou rozšiřujícího hardware, proto neexistuje žádná instrukce pro komunikaci s touto pamětí. Ovládání se provádí pomocí tří registrů; řídícího, datového a adresového. Většina mikrokontrolérů řady ATmega má velikost EEPROM 512B, proto je datový registr 16 bitový, ve skutečnosti je ovšem využito pouze 9 bitů.

Čtení dat z paměti je prováděno nastavením bitu EERE v řídícím registru. Do datového registru je přenesen byte z adresy udané adresovým registrem. Čtení trvá 4 hodinové takty a na tuto dobu je jádro mikrokontroléru pozastaveno.

Zápis je mírně komplikovanější a to z důvodu ochrany obsahu paměti. Aby nemohlo dojít k náhodnému přepsání obsahu paměti, je nutné vykonat speciální posloupnost v určitém časovém

okamžiku. Nejdříve je nutné vyčkat na dokončení předchozího zápisu, což je signalizováno vynulováním bitu EEWE. Následně je nutné zapsat hodnoty do adresového a datového registru. Zápis se provede sekvencí nastavení bitu EEMWE (globální povolení zápisu) a během dalších 4 hodinových taktů i bitu EEWE.

3.2.8 Watchdog

Obvod watchdog je realizován jako jednoduchý čítač s nezávislým RC oscilátorem. Jeho úkolem je resetování mikrokontroléru po určitém, předem nastaveném časovém úseku, nebyl-li včas vynulován. Pro vynulování čítače existuje speciální instrukce. Takto se dá zabránit zacyklení nebo uváznutí způsobeného především externím hardware. Je-li chyba již v návrhu, pak tento obvod postrádá smysl.

3.2.9 USART

Jednotka USART umožňuje synchronní a asynchronní přenos dat po sériové lince. Princip asynchronní komunikace je shodný s RS-232 až na napěťové úrovně logických stavů. Jednotka je vybavena vlastním generátorem, který při příjmu vkládá do přenášeného signálu synchronizaci. Frekvence tohoto generátoru je 16 krát vyšší než vlastní přenosová rychlost. Hodnota sběrnice je skenována v průběhu 7,8 a 9 vzorku a výsledná hodnota je aritmetickým průměrem. Generátor umožňuje také běh ve zrychleném režimu, kdy je frekvence pouze osminásobná.

Jednotka je ovládána třemi kontrolními registry, registrem pro volbu přenosové rychlosti a jedním datovým. Datový registr je společný pro příjem i pro vysílání a je dvojitě bufferovaný, tzn. dokáže uschovat až dvě hodnoty pro každý směr. Vysílání je zahájeno okamžikem zápisu do tohoto registru. Jednotka umožňuje přenost 5 až 9 datových bitů a volbu sudé případně liché parity. Zároveň při příjmu je schopná samostatně kontrolovat tento paritní bit a také hodnotu stop-bitu. Výsledky porovnání předává pomocí bitů v registru UCSRC. V tomto registru je také bit signalizující ztrátu znaku na příjmu kvůli přeplnění přijímacích bufferů. Registr UCSRA obsahuje několik bitů, které signalizují dokončení jednotlivých operací. Každý z těchto bitů umožňuje generovat přerušení. Bit RXC signalizuje dokončení příjmu, tzn. že vstupní buffer obsahuje alespoň jeden nepřečtený znak. Naopak dokončení vysílání signalizuje bit TXC. Posledním z trojice bitů je UDRE. Tento bit je nastaven v okamžiku, kdy je alespoň jedna pozice ve vysílacím bufferu volná. Pouze v tuto dobu je možné bezpečně zapisovat nová data do registru UDR.

3.2.10 TWI

Jednotka TWI umožňuje komunikaci po jednoduché dvouvodičové sběrnice shodné s protokolem sběrnice I2C. Jednotka umí pracovat v režimu Master i Slave a podporuje i sběrnici s více zařízeními

typu Master a proto je vybavena modulem pro detekci ztráty arbitráže. Ten porovnáváním vysílané hodnoty a hodnoty na sběrnici určuje, zda-li je komunikace rušena jinou vysílací stranou a případně se jednotka vzdá sběrnice.

Ovládání jednotky se provádí pomocí nastavování bitů v řídícím registru. Dokončení požadované operace je signalizováno nastavením bitu TWINT, který může být i zdrojem přerušení. Vynulování tohoto bitu, které se provádí zapsáním log. 1 na jeho místo, odstartuje další operaci na sběrnice. Jednotka zároveň pomocí 5 stavových bitů informuje o současném stavu přenosu po sběrnici a potvrzování odeslaných dat druhou stranou. Pro stanovení přenosové rychlosti se slouží předdělička a hodnota registru TWBR.

4 Komunikační protokol

Literatura [8] definuje komunikační protokol jako formální popis množiny pravidel, které určují syntaxi a význam jednotlivých zpráv během komunikace. Protokol musí být před zahájením komunikace definován a musí mu rozumět všechny strany, které se komunikace účastní.

4.1 Požadavky na komunikační protokol

Komunikační protokol musí být schopen zajistit přenos následujících zpráv

- Stav vstupů expandérů PCF8574
- Požadavky na zaslání informací o stavu expandérů
- Nastavení důležitých parametrů převodníku
    - Přenosová rychlost
    - Kmitočet oscilátoru
    - Perioda autonomního zasílání a skenování stavu expandérů
    - Směr jednotlivých vývodů
    - Výchozí hodnoty po resetu
- Reset mikrokontroléru

Jelikož RS-232 umožňuje pouze přenos bytů, nikoliv stavů po sběrnici, je nutné zabalit vlastní přenášená data do paketů. Pro označení začátku a konce je třeba vyhradit minimálně dvě speciální hodnoty. Další hodnotu je třeba vyhradit pro přenos informace, že na příslušné adrese není žádný expandér připojen. Na přenos stavu expandéru je třeba dalších 256 různých hodnot. V součtu je tedy třeba přenášet minimálně 259 různých hodnot. Pro přenos 259 různých hodnot je třeba 9 bitů. Ačkoliv většina mikrokontrolérů tuto variatu umožňuje, není specifikována v RS-232 a ani PC tyto přenosy nepodporuje. Problém je možno vyřešit dvěma způsoby. První by vedl k použití 16 bitové proměnné pro uložení hodnoty, která by pak byla přenášena nadvakrát. Tento způsob by komplikoval zpracování na straně mikrokontroléru, který je 8 bitový a všechny operace by byly zdlouhavější. Navíc by se zde projevil problém v případě, kdy by zařízení nebyla synchronizována a mohlo by tak dojít k špatné interpretaci, zda-li je byte horní či dolní část proměnné. Navíc je tento způsob náročnější na množství přenášených a ukládaných dat. Druhým způsobem je naopak komplikovanější na zpracování ale přenášené množství příliš nezvyšuje. Vyberou se hodnoty, které se v komunikaci často nevyskytují a těm je určen speciální význam. Dále je vybrána další hodnota, která ruší speciální význam hodnoty, která po ní následuje. Je-li v průběhu komunikace nutné přenést hodnotu, která by jinak měla speciální význam, předřadí se jí právě tato rušící hodnota. Je to obdobný princip jako v

programovacím jazyce C tzv. escape sekvence. V implementaci komunikačního protokolu jsem zvolil tento druhý způsob.

4.2 Vrstvový model protokolu

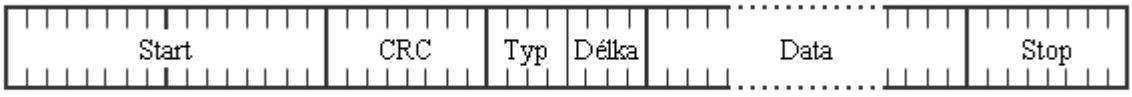
Komunikační protokol je rozdělen na několik vrstev. Každá vrstva virtuálně komunikuje s odpovídající vrstvou druhé strany. Nejvyšší vrstva přebírá parametry od programu, nejnižší pak odesílá data pomocí hardware. Vrstvy zapouzdřují data z vyšších vrstev, přidají potřebné hlavičky a předají paket ke zpracování nižší vrstvě. Tento způsob zpřehledňuje komunikační protokoly a zjednodušuje používání protokolu.

4.2.1 Fyzická vrstva

Nejnižší vrstvou je vrstva fyzická. Na této vrstvě jsou přenášena data po bitech. Fyzická vrstva je definována právě standardem RS-232. O zpracování dat na této nejnižší úrovni se stará samotný hardware. Ten umožňuje základní detekci chyb při přenosu a obstarává serializaci dat. Data jsou vyšším vrstvám předávána automaticky po jejich přijetí.

4.2.2 Linková vrstva

Nad fyzickou vrstvou pracuje vrstva, která jsem pojmenoval linkovou. Jejím úkolem je rozdělovat přenášená data do jednotlivých paketů. Struktura paketu je zobrazena na obrázku 4.1. Každý paket začíná dvojicí start-bytů. Dva byty jsem zvolil především z důvodu jednoduché a správné detekce začátku paketu, byla-li předchozí hodnota chybně přijata. V situaci, kdy má být přenášena uvnitř paketu hodnota start-bytu, je jí předřazena hodnota rušící speciální význam. Avšak bude-li právě tato hodnota chybně přijata, není přijímací algoritmus schopen detekovat, zda šlo opravdu pouze o přenos jedné hodnoty, nebo se jednalo o začátek nového paketu a ztracená hodnota značila pouze konec předcházejícího paketu. Po startovací značce je přenesena hodnota CRC. Dále následují vlastní přenášená data. Celý paket je ukončen jednobytovou hodnotou stop. Hodnoty jednotlivých speciálních bytů jsou v tabulce 4.1.



Obrázek 4.1: Paket linkové vrstvy

Hodnota	Popis
0x2829	Šesnáctibitová hodnota označující začátek paketu
0x2A	Následující hodnota postrádá speciálního významu
0x2B	Příslušný expandér není k převodníku připojen
0x2C	Chybně přijatá data
0x2D	Osmibitová hodnota ukončující paket

Tab. 4.1: Speciální hodnoty linkové vrtvy

4.2.2.1 Výpočet CRC

Kontrolní součet slouží k ověření správnosti přenášených dat. Pro výpočet CRC existuje velké množství algoritmů, jejihž výsledkem jsou různě široké hodnoty. Ale ani sebelepší výpočet CRC nemůže zaručit 100% přesnost přenosu. Za určitých okolností pak může dojít k označení i chybně přeneseného paketu za korektní. Při volbě algoritmu se spoléhá na předpoklad, podle kterého se chyby vyskytují ve shlucích. Jako výpočet CRC jsem zvolil výpočet obdobný paritě. Hodnota CRC je výsledkem operace XOR přes všechny přenášené byty s vyjímkou start a stop bytů

4.2.3 Aplikační vrstva

Na aplikační vrstvě jsou přenášena vlastní data určená ke zpracování. První byte určuje typ paketu a počet bytů paketem přenášených. Horní čtyři bity určují typ, dolní čtyři délku paketu. Do délky se nezapočítávají hodnoty, které nepřenáší užitečnou informaci. Přehled typů paketů lze nalézt v tabulce 4.2. Na přiloženém CD je dokument s příklady.

Požadavek na zaslání stavu převodníků má nepovinné dva parametry. Je-li zaslán bez parametrů, převodník odpoví zasláním stavu všech expandérů. Jsou-li zadány oba parametry, převodník zašle pouze stav expandérů, jejichž adresy náleží v uzavřeném intervalu tvořeném parametry žádosti. Převodník odpovídá hodnotami, které byly načteny během poslední aktualizace dat. Touto technikou se snižuje doba odezvy na požadavek i za cenu možné chyby.

Zasílání stavu expandérů je možné pomocí dvou typů paketů. V prvním případě se jedná o neadresovaný paket, takže první odeslaná hodnota je informace o stavu převodníku s adresou 0. Druhou možností je adresovaný paket. Zde je první hodnotou v paketu adresa prvního převodníku a následují hodnoty. U obou typů je počet přenášených hodnot stanoven v hlavičce paketu. Nemusí se tedy přenášet informace o všech expandérech, ale například pouze o jednom. Tyto možnosti jsou v komunikačním protokolu zahrnuty kvůli snížení počtu přenášených dat i paketů. Tuto techniku převodník používá pouze v případě odpovědi na požadavek.

Typ	Číslo	Parametry	Popis
Data	0	<addr><data>	Data začínající od adresy <addr>
	1		Data začínající od adresy 0
Požadavek	2		Požadavek na zaslání dat všech expandérů
		<addr0><addr1>	Požadavek na zaslání dat expandérů v rozmezí adres
Nastavení parametrů retězci	6	<typ><data_s>	Zápis nové hodnoty nastavení
	7	<typ>	Čtení hodnoty současného nastavení
	8	<typ><data_s>	Odpověď na čtení
			typ = 'X' Nastavení hodnoty frekvence oscilátoru
			typ = 'B' Nastavení hodnoty přenosové rychlosti
			typ = 'S' Nastavení periody obnovování stavu expandérů
			typ = 'R' Nastavení periody zasílání stavu po RS-232
Nastavení parametrů binárně	10	<typ><data>	Zápis nových hodnot směrů expandérů
	11	<typ>	Čtení hodnoty současného směru expandérů
	12	<typ><data>	Odpověď na čtení
			typ = 'D' Nastavení výchozího směru převodníků
			typ = 'V'
Sněr	13	<addr><data>	Okamžitá změna směru expandérů od <addr>
Restart	14		Restart převodníku, načtení nového nastavení

<data> Binární data
<data_s> Data v řetězcovém/textovém formátu
<addr> Adresa expandéru – v rozsahu <0,7>

Tab.4.2: Aplikační protokol

Pro nastavování parametrů převodníku jsou potřeba 3 různé typy paketů. První typ je paket provádějící nastavení. První byte určuje, který z parametrů bude změněn, následující pakety udávají hodnotu. Hodnota je přenášena jako řetězec číslic. Druhý typ označuje výzvu zaslání aktuální hodnoty, třetí pak odpověď s hodnotou. Formát přenášených dat je shodný jako u nastavení.
Stejným způsobem se nastavuje i směr převodníků a výchozí hodnoty po startu mikrokontroléru. Vzhledem k omezení délky paketu aplikační vrstvy na 15 znaků, není možné zasílat tyto hodnoty jako řetězec ani v hexadecimálním tvaru, a proto jsou použity vlastní příkazy. Data jsou přenášena v binárním tvaru. U nastavování směru značí logická 0 výstupní pin, logická 1 vstupní. Stejně jako v předchozím případě se změny projeví až po restartu převodníku. Změna směru je však operace, která se může vyskytnout i v průběhu běžného provozu. Proto aplikační protokol obsahuje i příkaz, který provede změnu směru pinů expandéru okamžitě. Tato změna se projeví v okamžiku, kdy dojde ke zpracování paketu na aplikační úrovni.

Příkaz pro reset mikrokontroléru slouží k načtení nového nastavení. Nepředpokládám, že by bylo nutné tento příkaz používat i k zotavení mikrokotroléru z chybového stavu,

jelikož pravděpodobně mikrokontrolér v tomto stavu již nebude reagovat na žádné příchozí pakety a místo toho zareaguje watchdog.

4.3 Vlastnosti protokolu

Protokol je navržen tak, aby přenášel data pokud možno s minimálním počtem redundantních informací. Správnost informace je zaručena pomocí osmibitové hodnoty kontrolního součtu. Tato hodnota je schopná v části případů úspěšně detekovat chybu a paket je vyřazen ze zpracování. Protokol však neposkytuje žádný nástroj pro zajištění spolehlivého přenosu dat. Místo toho doporučuji zasílání paketů častěji. Aby nedocházelo k zahlcování a ztrátě dat přetečením vstupních bufferů, používá protokol hardwarový handshaking. K řízení komunikace slouží převážně vodiče RTS a CTS.

5 Hardware

Při návrhu hardwarové stránky mé práce jsem se zaměřil především na jednoduchou testovatelnost výsledného projektu. Výsledkem je několik oddělených modulů, které jsou navzájem pospojovány kabely a je možné je jednoduše vytvářet různé kombinace. Hlavním modulem je vlastní převodník. K tomu je možné pomocí 5 žilového vodiče připojovat jednotlivé expandéry PCF8574. Výstupem modulu expandéru je 8 datových vodičů. Tyto vodiče je dále možné připojit ke vstupním nebo výstupním modulům. Schéma i plošný spoj každého modulu je uložen na přiloženém CD.

Modul převodníku lze rozdělit do tří částí; stabilizace napájení, úprava úrovní RS-232 na TTL a vlastní mikrokontrolér. Schéma převodníku je součástí přílohy k této bakalářské práci.

Pro napájení převodníku jsem se rozhodl použít 5V. Samozřejmě lze zakoupit všechny součástky i v provedení pro nižší napájecí napětí. Pro stabilizaci napětí jsem použil integrovaný stabilizátor 7805. Tento obvod jsem zapojil podle doporučení výrobce [11], tj. s blokovacími kondenzátory na vstupu i výstupu. Vstupní napětí může být v rozsahu 7V až 35V. Při větších proudových odběrech je vhodné opatřit stabilizátor odpovídajícím chladičem.

Nejpoužívanější převodník mezi úrovněmi TTL a RS-232 je obvod firmy Maxim MAX232. Tento obvod se dočkal mnoha různých provedení, které se od sebe liší především velikostí externích kondenzátorů, počtem párů budičů a napájecím napětím. Obvod se skládá z násobičky a invertoru napětí a výstupních budičů. Ke své činnosti obvod vyžaduje pouze 4 elektrolytické kondenzátory. Běžně dostupné obvody řady MAX232 obsahují 2 budiče pro každý směr. Pro převod dat a plného handshaking jsou třeba páry 3. Tento problém jsem vyřešil zapojením dvojice signálů DTR-DSR do zpětné smyčky a pro handshaking použít pouze CTS-RTS. Jako konektor sběrnice RS-232 jsem použil konektor, kterým jsou vybavovány zařízení DTE. Lze pak vždy pomocí kabelu typu Null-modem spojit jako počítač s převodníkem, tak dva převodníky navzájem.

Mikrokontrolér ATmega prakticky nevyžaduje žádné externí součástky a vystačí si ve většině případů s integrovaným RC oscilátorem. Pro vyšší přenosové rychlosti by však rychlostí ani stabilitou nemusel dostačovat a proto je na plošném spoji místo pro krystal a blokovací kondenzátory (tyto součástky mají v názvu hvězdičku). Případně je možné připojit i externí zdroj. Vývody jednotky USART jsou spojeny s obvodem MAX, stejně tak dva signály pro řízení toku. Vedle mikrokontroléru je umístěno tlačítko pro reset. Nad mikrokontrolérem je vyveden 5 pinový konektor pro připojení expandérů a 7 pinový konektor pro připojení programátoru.

Expandér PCF8574 také nepotřebuje žádné externí součástky. Na modulu jsou vyvedeny dva konektory pro průběžné připojení k řídícímu modulu. 10 pinový konektor slouží k vyvedení datových

pinů expandéru a napájení. Poslední lišta slouží k nastavování adresy pomocí zkratovacích propojek, tzv. jumperů. Je-li propojka zkratována, je na příslušném bitu adresy logická 0.

Pro signalizaci stavu bitů slouží výstupní modul. Jelikož PCF8574 obsahuje v kladné větvi budičů pouze pull-up rezistory, nemůže poskytnout dostatečný proud pro rozsvícení diody. Nejjednodušší řešením by bylo otečením směru diod a rozsvěcovat je v logické 0, avšak toto řešení se mi zdá poněkud matoucí navíc má PCF8574 i proudová omezení a mohlo by dojít k jeho spálení. Raději jsem mezi expandér a diody zařadil 8 bitový budič, který již dostatečný proud poskytne.

Jako vstupní modul slouží prostý modul dip-spínačů. Vzhledem k výše uvedené architektuře expandérů s pull-up rezistory značí sepnutý spínač logickou 0. Při delší vzdálenosti mezi moduly je vhodné osadit v blízkosti dip spínačů externí pull-up rezistory.

6 Řídící program

Jazyk C umožňuje programátorovi napsat program pouze s omezenou znalostí cílové architektury. Avšak ani na této úrovni nejsou programy 100% přenosné mezi architekturami především kvůli rozdílným názvům konfiguračních registrů. Ani v rámci rodiny ATmega není zdrojový kód přenosný zcela bezezměn, což je zapříčiněno rozdílným hardware a jeho ovládáním. Na druhou stranu programátor nemusí mít znalost instrukční sady mikrokontroléru.
Program v jazyce C určený pro mikrokontorléry se většinou sestává ze dvou částí; konfigurace a nekonečná smyčka. V konfigurační části je prováděno nastavení hardware a inicializace proměnných. Nekonečná smyčka zajišťuje vlastní činnost programu. Většinou se používá cyklus *while* nebo *for* s prázdnou ukončovací podmínkou. Opuštění této smyčky většinou znamená problém, jelikož program může začít náhodně vykonávat posloupnost příkazů, která je v paměti na následujících adresách.

Pro návrh a překlad programu jsem zvolil nástroj AVRStudio od výrobce mikrokontrolérů. Jako překladač se pak přímo nabízel opensource projekt WinAVR. Ten se skládá z GNU GCC překladače, knihoven definujících jednotlivé mikrokontrélery a usnadňujících práci programátorovi. AVRStudio také dokáže spolupracovat s několika programátory.

Program během své činnosti musí kontrolovat několik požadavků od různých zdrojů. K tomuto účelu existují dvě techniky – pooling a přerušení. Pooling je způsob, při kterém program neustále testuje, zdali se nevyskytl požadavek na obsluhu. Tato technika znemožňuje efektivní zpracování jakékoliv další operace. Také hrozí nebezpečí, bude-li požadavek trvat pouze krátkou dobu, že jej program nestihne detekovat včas. Přerušení oproti poolingu umožňuje efektivní provádění operace na pozadí ale trpí jinými problémy. Především může dojít k „vyhladovění“. Každá žádost o přerušení má svojí prioritu. Bude-li žádost o přerušení s vysokou prioritou generována příliš často, nemusí dojít k obsluze nižších žádostí.

Pro návrh programu převodníku jsem použil kombinaci obou způsobů. Všechny požadavky jsou detekovány pomocí přerušení, ale pouze u některých dojde ke skutečnému zpracování. U ostatních dojde pouze k překopírování žádosti do speciální proměnné a bude obslouženo až v okamžiku, kdy procesor nebude vytížen zpracováváním jiných operací.

6.1 Hlavní vlákno

Po resetu mikrokontroléru proběhne nejprve inicializace proměnných, což je sekvence, kterou generuje sám překladač bez přičinění programátora. Program uchovává několik konfiguračních

parametrů, které jsou důležité pro správnou funkci. Nejdůležitějším je údaj uchovávající frekvenci hodinového signálu mikrokontroléru. Z tohoto údaje se pak určují další, hodnoty, jako předvolba pro časovou základnu, či dělící poměr pro sériovou linku a sběrnici I2C. Další ukládanou hodnot je přenosová rychlost po sběrnici RS-232, výchozí směr jednotlivých expandérů a intervaly, ve kterých má být prováděn periodický zápis a čtení z expandérů. Poslední hodnotou je kontrolní číslo. Toto číslo je nastaveno na konstantu v okamžiku zápisu platných dat do paměti EEPROM. Z důvodů prevence poškození této buňky, která by byla častým terčem zápisů, jejichž počet je v EEPROM omezen, je hodnota fyzicky zapsaná pouze pokud nebyla tato hodnota dříve nastavena.

Každý z konfiguračních parametrů je možné nastavit dvěma způsoby. První, složitější, je pomocí řídícího programu s využitím komunikačního protokolu. Tento způsob vyžaduje počítač s nakonfigurovaným softwarem schopným zpracovávat komunikační protokol. Avšak software zpracovávající komunikační protokol přiložený k této bakalářské práci jsem navrhl pouze pro operační systémy Windows, implementoval jsem i další způsob konfigurace. Tento druhý způsob vyžaduje pouze zařízení vybavené sběrnicí RS-232 a textový komunikační program – například hyperterminál.

Program převodníku po startu nastaví jednotku USART na přenosovou rychlost 19200 Bd s 8 datovými bity, bez parity a s jedním stop-bitem a naslouchá na sběrnici. Přenosová rychlost se vypočítává z nastavené frekvence hodinového signálu v paměti EEPROM. Není-li v paměti nalezena značka korektního zápisu hodnot, pak se defaultně předpokládá kmitočet 8MHz. Obdrží-li do 10 vteřin posloupnost dvou znaků CR (ASCII 13) přerušenou maximálně znakem LF (ASCII 10), přepne se do textového komunikačního režimu. Na obrazovku terminálu budou vypsány základní informace a menu s možnými příkazy. Pomocí těchto příkazů je pak možné jednoduše nastavit všechny důležité parametry.

Po ukončení nastavování, případně neobdrží-li v daném časovém intervalu požadovanou posloupnost, pokračuje program ve startování. Program se pokusí načíst konstanty z EEPROM. Platnost dat v paměti EEPROM udává kontrolní součet. Není-li nalezena očekávaná hodnota, jsou použity defaultní hodnoty. Ty jsou uvedeny v tabulce TAB. Po načtení všech hodnot se provede výpočet předvoleb pro jednotlivý hardware. Toto je jediné místo, kde jsem použil instrukce dělení, jelikož jejich softwarová implementace je jak časově, tak paměťově náročná.

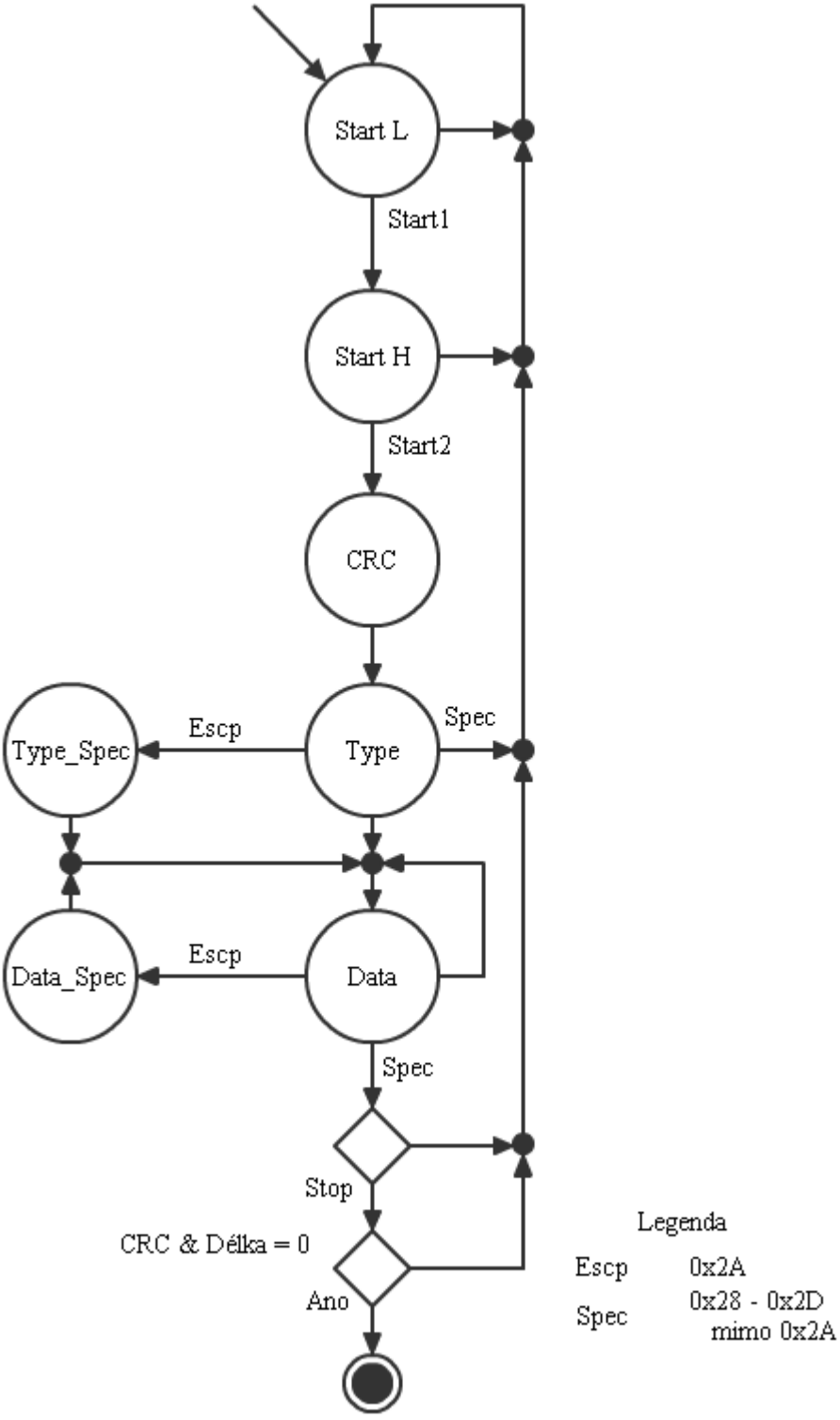
Po inicializaci hardwarových částí mikrokontroléru vstupuje program do hlavní smyčky. V této smyčce jsou postupně testovány příznaky žádostí o obsluhu jednotlivých částí programu. Všechny žádosti mají stejnou prioritu a jsou testovány popořadě. Nemůže tedy dojít k situacím, aby byla některá ze žádostí vyhladověna jinou. Žádosti také na sebe logicky navazují, takže po dekódování aplikační části komunikačního protokolu je proveden zápis do expandérů. Bude-li aplikační protokol provádět zápis do těchto expandérů, nepotřebuje obsahovat implementaci zápisu do expandérů, pouze

upraví hodnoty v paměti a nastaví bit, kterým se v následujícím kroku hlavní smyčky okamžitě provede zápis. Postupně se provádí testování příjmu po sériové lince, zápis do expandérů, pravidelné čtení z expandérů a mimořádné čtení z expandérů. Odesílání po sériové lince je implementováno v každé části programu, kde by se mohl tento požadavek vyskytnout.

6.2 Zpracování komunikačního protokolu

Pro příjem a vysílání dat po sériové lince je využito přerušení. Pro každý směr přenosu je vytvořena fronta. Při příjmu program ověří, přijala-li jednotka USART byte korektně a data jsou případně uložena do bufferu nebo jsou nahrazena slovem signalizujícím chybu. Také je testována zbývající kapacita fronty. Pokud tato hodnota klesne pod kritickou mez, program vynuluje CTS a signalizuje tak druhé straně svoji neschopnost dalšího přijmu. Po přijetí celého paketu je nastaven bit stavové proměnné signalizující příjem paketu. Tento bit je také nastaven v případě přílišného zaplnění přijímacího bufferu nebo detekce chyby při příjmu. Následně je v hlavní smyčce provedeno dekódování a zpracování. Zpracování paketu probíhá pomocí jednoduchého stavového atuomatu zobrazeného na obrázku 6.1. Je-li paket korektní, program zpracuje aplikační část paketu.

Vysílání probíhá také pomocí přerušení. Po dokončení odesílání bytu program vyzvedne další z fronty a odešle jej. Během této akce je také testován příznak povolující odesílání, který kopíruje hodnotu signálu RTS. Pro odeslání nového paketu je nutné jej nakopírovat do fronty a zahájit vysílání. Je-li vysílač jednotky USART aktivní, což je signalizováno bitem ve stavové proměnné, není nutné provádět žádnou akci. V opačném případě je nutné vysílání zahájit přenesením prvního bytu do registru UDR. Při vysílání je také testován příznak povolení vysílání. Je-li zakázáno druhou stranou další vysílání, je odesílání pozdrženo až do doby, kdy je druhá strana schopna data opět přijímat. Změna signálu RTS je detekována pomocí přerušení a je následně překopírována do stavové proměnné. V okamžiku povolení vysílání je také, byla-li již komunikace pozastavena, znovu restartován přenos uložením nové hodnoty ze začátku fronty do registru UDR.



Obr.6.1: Doporučené zpracování komunikačního protokolu

6.3 Fronta

Fronta je abstraktní datový typ využívající mechanizmus FIFO (First In, First Out). Fronta má dva konce, které se nazývají head nebo začátek a tail nebo konec. Začátek fronty je taková pozice, ze které budou odebírány prvky, na konec se nové prvky přidávají. Nad frontou jsou typicky prováděny 3 základní operace – *Put*, *Get*, *Empty*. Někdy jsou tyto operace nazývány stejně jako u zásobníku – *Push*, *Pop* – a toto značení využívám i já. K této základní trojici operací jsem přidal i čtvrtou, kterou jsem nazval *Full*. Tato funkce vrací počet zbývajících volných položek, které jsou ještě využitelné pro uložení dat do fronty.

6.3.1 Implementace pomocí rotačního pole

Nejjednodušší způsob implementace fronty je pomocí tzv. rotačního pole. Tento způsob využívá obyčejné pole a dva ukazatele do něj. Oba ukazatelé se v poli pohybují stejným směrem, například jsou oba dva inkrementovány a v případě přetečení mimo pole jsou resetovány na začátek pole. Jeden z ukazatelů značí začátek fronty, druhý pak konec. Začátek obvykle ukazuje první obsazené pole, konec je pak za posledním uloženým, tedy na prvním volném. Ukazují-li oba ukazatele na stejný prvek v poli, pak je fronta prázdná. Naopak plná je fronta v okamžiku, kdy konec je o jeden prvek za začátkem. Aby bylo možné korektně odlišit prázdnou frontu od plná, je při tomto řešení nedostatečně využit veškerý paměťový prostor a vždy jeden prvek zde zůstane nevyužitý.

Vlastní implementace v jazyce C je poměrně jednoduchá. Vytvořil jsem si nový datový typ obsahující vlastní pole pro data a dva ukazatele do něj. Velikost pole je definována konstantou a je pro všechny fronty stejná. K udržení ukazatelů v rozmezí pole se většinou používá funkce modulo (zbytek po dělení). Tato operace stejně jako dělení není implementována v základní instrukční sadě mikrokontroléru ATmega. Operaci modulo jsem nahradil operací logického součinu. Po každé inkrementaci jsou zamaskovány bity, které by mohly způsobit, že index bude mimo oblast pole. Proto musí být velikost pole mocninou dvojky. Bude-li velikost pole 32, pak možné indexy jsou v intervalu 0 až 31, tzn. využívá se pouze nejnižších 5 bitů.

6.4 I2C

Komunikace po I2C kvůli svému návrhu přímo vyžaduje pooling. Tímto omezením nevznikají žádné komplikace, jelikož veškerou komunikaci zahajuje mikrokontrolér. Proto program vždy bude testovat pouze dokončení operace, kterou sám zahájil.

Zápis dat do převodníků probíhá jednoduše. Program nejprve vyšle adresu a pak data. V průběhu vysílání samozřejmě testuje, je-li příslušný převodník připojen a odpovídá-li.

Bohužel sběrnice I2C neumožňuje kontrolu přenášených dat, pouze zda-li nějaká data slave přijal. Dojde-li ke změně hodnoty některého bitu vinou rušení, neexistuje žádný možný způsob detekce a to ani přečtením zapsané hodnoty kvůli architektuře expandérů. Jediné rozumné řešení spočívá v periodicky opakovaném zapisování hodnot do expandéru. Vznikne-li chyba při přenosu, bude dalším zápisem snad opravena.

Čtení je nutné rozdělit na dva případy. V první případě jsou data čtena v periodických intervalech, aby byla zajištěna platnost dat v paměti mikrokontroléru. Také je pravidelně testováno připojení nových expandérů. Převodníky jsou adresovány vzestupně podle adres. Nastala-li nějaká změna, je po kompletním přečtení všech expandérů odeslán paket s novými daty. Naopak pro zjištění, který z expandéru vyvolal požadavek na přerušení, existuje tabulka pořadí adresace. V tabulce je na prvním místě prvek s nejvyšší prioritou. Je-li detekována změna čtené hodnoty na určitém převodníku, je posunut v tabulce o jednu pozici směrem k vyšším prioritám. Tímto způsobem je zajištěna co nejrychlejší odezva na často opakující se změny určitých bitů. Čtení se provádí postupně pro všechny převodníky, dokud trvá požadavek na přerušení, maximálně však třikrát každý převodník, aby nedošlo k zacyklení. Jakmile je zjištěna změna, je nová hodnota odeslána.

6.5 Časová základna

Jako časová základna slouží 16 bitový čítač/časovač 1. Ten, narozdíl od ostatních, umožňuje provoz v režimu CTC a zároveň nastavení dostatečně dlouhé periody. Dělící poměr časovače je nastaven v průběhu inicializace v závislosti na datech v EEPROM na 1 ms. Intervaly mezi jednotlivými operacemi jsou pak udávány jako násobky tohoto intervalu. Po každém přetečení časovače jsou dekrementovány proměnné jednotlivých akcí a případně nastaveny bity ve stavových proměnných signalizující start. Interval se od tohoto okamžiku začíná počítat nanovo bez jakékoliv kontroly, zdali a kdy byla akce provedena.

6.6 Watchdog

Vlastní program jsem navrhnul a vytvořil tak, aby byl odolný vůči možnosti zacyklení. Existuje však vyjímka, vůči kterým není možná 100% ochrana. Touto vyjímkou je komunikační sběrnice I2C. I když komunikaci řídí program mikrokontroléru, sběrnice umožňuje, aby si Slave zařízení mohlo vynutit vložení čekacích stavů. A právě zde může vzniknout zaseknutí se. Pravdou zůstává, že riziko by bylo vyšší, kdyby Slave zařízením byl nějaký mikroprocesor řízený programem, ale i u čistě hardwarového zařízení může tento problém vzniknout.

Vlastní ovládání obvodu watchdog jsou použita makra projektu WinAVR. Hlavičkový soubor těchto maker a funkcí je ve složce \util\watchdog.h. Tento soubor obsahuje jako funkce pro nastavení watchdogu na určitý interval, tak i makro pro resetování hodnoty čítače Watchdogu.

6.7 Ukládání konstant

Ukládání dlouhých řetězcových konstant je na architekturách von Neumann jednoduché a efektivní. Konstanta se uloží do zdrojového programu a jelikož jsou data a program adresovány do stejné paměti, není jejich načtení složité. Bohužel na architekturách Harvard je tento způsob nemožný. Překladače nejsou koncipovány tak, aby si samovolně načítali konstanty z paměti programu pomocí speciálních instrukcí. Proto tuto variantu obcházejí vytvořením globálních proměnných, do kterých tento řetězec v rámci inicializace proměnných načtou. Řetězec pak zabírá místo dvakrát, jednou v paměti programu, podruhé v paměti dat. Také je v paměti dat uložen po celou dobu a je tedy náchylnější na nechtěné přepsání.

Řešení této komplikace nabízí opět projekt WinAVR. Makra nazvaná PGM (ProGram Memory) umožňují ukládání konstant do paměti programu a pak jejich načtení pomocí jednoduché funkce do paměti dat. Sice i zde je nutné před použitím řetězce jeho překopírování do paměti dat, ale celkové využití paměti se mnohonásobně zvyšuje. Není totiž nutné vymezovat pro každý řetězec vlastní pole a také existuje jedna univerzální funkce pro překopírování dat. Nevýhodou tohoto řešení je vyšší časová náročnost, protože řetězce jsou z paměti programu načítány častěji.

7 Pomocné přípravky

7.1 USB-RS232

U většiny moderních počítačových sestav a u notebooků již úplně chybí port RS-232. Jeho absenci lze řešit několika způsoby. Nejjednodušší je využití specializovaného obvodu, který dekóduje protokol USB. Existuje několik výrobců z nichž je pro tento úkol nejvhodnější firma FTDI. Firma vyrábí různé převodníky mezi jednotlivými typy sběrnic. Mezi nimi je i RS-232. Ovladače, které jsou k jednotlivým obvodům dodávány, emulují v počítači nový sériový port. Přistup k tomuto portu je identický jako přístup ke klasickým portům integrovaným na základní desce počítače. Avšak po hardwarové stránce je činnost portu rozdílná.

7.1.1 USB

Sběrnice USB je moderní sériovou sběrnicí. Komunikaci na sběrnici řídí vždy aktivní prvek, tj. počítač nebo hub. Pro každé zařízení je vyhrazeno přenosové pásmo. V závislosti na přenosovém pásmu hostitelské zařízení testuje stav funkční jednotky a případně čte její data. Proto na sběrnici vznikají mírná zpoždění, která můžou mít vážný vliv na chod některých zařízení. Mezi ně může částečně i patřit převod RS-232 pomocí USB.

7.1.2 Obvod FT232RL

Tento obvod je jádrem většiny RS-232 adaptérů. Existuje v několika různých verzích, které jsou odlišné především stupněm integrace a počtem převáděných kanálů. Obvod generuje všechny signály sběrnice RS-232 avšak pouze na napěťových úrovních TTL, případně nižších. Pro převod je nutné použít případně shodný převodník napěťových úrovní jaký je uveden výše. Mimo to obvod umožňuje vyvedení některých řídících signálů, které již s RS-232 nesouvisí. Jedná se například o signály pro indikaci příjmu a vysílání pomocí LED.

Z principu komunikace po sběrnici USB je jasné, že obvod musí být vybaven vyrovnávací pamětí a komunikaci po sériové lince řídit sám. Obvod odesílá a přijímá data z/do bufferů a až během synchronizace s PC je přenáší do počítače. Toto způsobuje určité zpoždění v komunikaci. Také dochází k rozsynchronizování signálů v režimu přímého přístupu k portu. Obvod však disponuje režimem zvaným Bit Bang. Tento režim, na rozdíl od běžné komunikace, umožňuje přímé řízení jednotlivých signálů. Cenou za tento přístup je požadavek širšího přenosového pásma a také nižší propustnost obvodu.

7.2 Programátor

Pro naprogramování mikrokontroléru je programátor nezbytný. Existují dva různé způsoby, kterými lze programování řešit. Buďto pořízením drahého profesionálního programátoru vybaveného různými funkcemi s podporou pro ladění programu nebo postavením si vlastního jednoduchého programátoru. Vzhledem k faktu, že mikrokontrolér ATmega8 nedisponuje rozhraním pro podporu ladění na čipu (JTAG), a ceně běžně prodávaných programátorů jsem se rozhodl, že vystačím s jednoduchým programátorem a veškeré případné ladění budu provádět simulací pomocí vývojového nástroje.

Pro nahrání programu do mikrokontroléru existují dva různé způsoby. První se provádí paralelním způsobem. Tento režim se používal v začátcích programování a vyžaduje propojení velkého počtu vodičů. Výhodou tohoto režimu však nadále zůstává fakt, že tento způsob nejde nijak zakázat a funguje vždy. V dnešní době je již vhodný především na „opravu“ mikrokontroléru, pokud došlo k chybnému nastavení sériovým programováním a není možné provést opravu sériově.

Sériové programování naopak vyžaduje pouze několik málo vodičů. Data jsou do mikrokontroléru přenášena pomocí SPI rozhraní. SPI umožňuje sériový obousměrný synchronní přenos dat. Hodiny generuje řídící obvod, v tomto případě programátor, a data jsou vždy přenášena s vysokou úrovní hodinového signálu. Kromě 3 vodičů pro SPI je nutné ještě připojit resetovací vývod a napájecí napětí. Programovaný mikrokontrolér také musí mít aktivní jednotku časovaní. Je-li propojkami nastavena některá z variant vyžadující externí hodinový signál případně externí krystal, musí být tyto požadavky splněny.

Velká část sériových programátorů využívá sériového nebo paralerního portu počítače a pracuje s přímým přístupem k portu. Nejjednodušší programátor, který jsem našel, využíval paralelní port a skládá se z konektoru a 5 ochranných rezistorů. Řízení programování pak provádí speciální software, který generuje SPI signály na těchto portech. Takovýto přístup úspěšně fungoval na starších verzích operačních systémů. U moderních operačních systémů je režim přímého přístupu na port zakázán a je třeba využívat pomocné knihovny, které ne vždy fungují.

Bohužel žádný nový notebook ani žádná moderní počítačová sestava sériový ani paralelní port neobsahují a jejich emulace není pro použití na programátoru vhodná. Proto jsem se poohlédl po programátoru, který by bylo možné připojit přímo do USB nebo alespoň přes převodník na sériový port. Zvolil jsem open-source projekt AvrUsb500v2, zmíněný v literatuře [13]. Jedná se o programátor postavený na mikrokontroléru ATmega8. Programátor vychází ze svého času velmi známého vývojového kitu firmy Atmel, který byl označován jako STK500. Kromě vývojového kitu mohl být také používán jako programátor. Pro spojení s počítačem využíval RS-232 a speciální komunikační protokol. Podpora tohoto kitu je zachována i v současných verzích vývojových nástrojů ATMEL.

Programátor tedy obsahuje převodník USB – TTL a mikrokontrolér, který bude vykonávat příkazy zasílané počítačem po sériové lince. Zde ovšem vzniká problém, často označovaný jako „vejce-slepice“. Pro oživení programátoru potřebujeme naprogramovat mikrokontrolér. Naštěstí i zde existuje řešení. Obvod FT232RL, který slouží jako převodník mezi USB a TTL, umožňuje provoz v Bit Bang režimu. Využitím tohoto režimu a zbývajících volných vodičů linky RS-232 lze vytovořit jednoduchý programátor, kterým lze, ač pomalejší rychlostí, naprogramovat mikrokontrolér. Autor AvrUsb500v2 doporučuje použití jím vytvořené knihovny a operačního systému Linux. Tato cesta však je v různých diskuzích označována jako příliš pracná a často nefunkční, takže jsem se poohlédl po jiném řešení. V literatuře [12] autor doporučuje použití programu AVRdude spolu s jeho grafickou nadstavbu. Touto cestou se mi povedlo úspěšně oživit programátor bez sebemenších komplikací.

8 Demonstrační program

Pro demonstraci činnosti jsem navrhl a vytvořil jednoduchý program. Program je vytvořen pomocí programovacího jazyku Object Pascal. K návrhu jsem použil integrované grafické vývojové rozhraní vytvořené firmou Borland nazývané Delphi. V současnosti odprodala firma Borland tento produkt Embarcadero technologies. Pro komunikace se sériovou linkou jsem použil komponentu pojmenovanou serialng [literatura]. Komponenta je volně šířitelná pro bezplatně vyvíjené programy. Programátor pouze používá funkce pro odeslání a přijetí bytů, resp řetězců.

Program se skládá z dvou částí. První část reaguje na akce uživatele. Jelikož se jedná o objektově orientovaný programovací jazyk, nevyžaduje opakované testování, ale obslužné rutiny jsou spouštěny automaticky po zaslání zprávy operačním systémem. Reakcí na změnu je zaslání paketu podle komunikačního protokolu. K detekci nově přijatých dat je použit časovač. Ten periodických spouští obslužnou rutinu. Ta testuje počet přijatých bytů a jsou-li k dispozici nové, uloží je do pole a pak se pokouší se dekódovat paket. Data pak následně vizualizuje výsledky.

Program umožňuje dva základní režimy činnosti. V prvním režimu vizualizuje hodnoty na vstupech expandérů a zároveň zasílá hodnoty zvolené uživatelem na výstup expandérů. Současně může uživatel ovlivňovat směr jednotlivých pinů a tak i jejich hodnotu. Druhý režim umožňuje testovat reakce na podněty zasílané komunikačním protokolem. Uživatel pomocí několika textových polí ovlivňuje obsah paketu zasílaného po RS-232. Zároveň je obsah každého přijatého paketu zobrazen v textovém poli a uživatel si může prohlížet historii přijatých zpráv včetně těch, které byly přijaty v okamžicích, kdy nebyl tento režim aktivní.

9 Testování převodníku

Převodník jsem testoval ve dvou etapách. První částí bylo připojení převodníku k počítači a testování správnosti odezvy na změnu vstupu. K převodníku jsem připojil expandér spolu s diodami. Současně jsem zakázal automatické obnovování a zasílání dat a testoval jsem, zdali jsou všechny změny přenášeny a výsledná data se zobrazují korektně. Zjistil jsem, že několika případech změna nebyla detekována. Při dalším zkoumání jsem došel k závěru, že se jedná o případy, kdy dojde chybě během přenosu a paket je vyřazen ze zpracování.

Druhou částí testování bylo přibližné měření doby odezvy. Doba odezvy převodníku je ovlivněna několika faktory, a proto ji nelze změřit přesně. Tento údaj ovlivňuje počet paketů vyčkávajících na zpracování ve frontách, aktuální činnost převodníku, přenosové rychlosti RS-232 a také pořadí expandéru, tj. s kolika předcházejícími expandéry bude muset převodník komunikovat. Měření jsem prováděl pomocí dvou převodníků. Na jednom jsem jsem generoval náhodné změny a měřil jsem dobu, za kterou se změna projevila na druhém převodníku. Zároveň jsem na obou převodnících pomocí dalších expandérů generoval náhodné změny, aby měření odpovídala reálnému pracovnímu vytížení převodníků. Měření jsem prováděl při přenosové rychlosti 19200 Bd. Při této přenosové rychlosti dosáhl převodník přibližné doby odezvy 10 ms. Tato doba byla podle předpokladu proměnná v závislosti na pořadí expandéru v prioritní tabulce. Při zvýšení přenosové rychlosti na 115200 Bd se tato změna podle očekávání snížila na pouhé 2 ms. Další zvyšování přenosové rychlosti již nemělo na dobu odezvy přílišné změny. To je způsobeno tím, že není možné zvyšovat přenosovou rychlost po I2C. Také jsem již dosáhl maximální pracovní frekvence pro použitý převodník MAX232. Je nutné zopakovat, že tyto hodnoty byly naměřeny experimentálně při náhodném zatížení a výsledky dalších měření se můžou velmi lišit.

10 Závěr

Takto vytvořený převodník splňuje požadované cíle. Je schopen autonomní komunikace a spolupráce jak s řídící jednotkou, tak vytvořit spojení se shodným převodníkem a při vhodné konfiguraci spolupracovat za účelem vytvoření inteligentního multiplexu. Při jeho tvorbě jsem se seznámil se způsobem návrhu programu pro mikrokontroléry a s možnostmi, které poskytují různá vývojová prostředí. Také jsem si ověřil, jak je složité vytvořit návrh kvatního plošného spoje.

Vytvořený převodník klade na přenášený signál určitá omezení. Jedná se o minimální dobu, která musí uplynout mezi dvěma změnami signálu. Omezení zde kladou především přenosové rychlosti sběrnic. Zatímco rychlost přenosu po RS-232 lze zvyšovat, i když na úkor maximální dosažitelné vzdálenosti a odolnosti proti rušení, I2C má přenosovou rychlost konstantní a to 100 kbit/s. Jednoduše pak lze odvodit, že maximální rychlost přenášeného signálu při použití jediného převodníku může být teoreticky 5 kHz. Bohužel tohoto teoretického maxima převodník nebude pravděpodobně schopen nikdy dosáhnout.

Z hlediska spolehlivosti se dá převodník označit jako spolehlivý. Během testování nedošlo k žádnému zaseknutí či zacyklení. I kdyby k takovéto situaci došlo, integrovaný watchdog mikrokontrolér resetuje a ten je během několika sekund znovu připraven spolupracovat. Bohužel vzhledem k principu inicializace dat, která je provedena dříve, než je předáno řízení uživatelskému programu, není možné pokračovat v činnosti tam, kde došlo k přerušení a všechna data jsou ztracena. Také v případě chyby při přenosu dochází ke ztrátě paketu. Vzhledem k paměťovým nárokům jsem se rozhodnul neimplementovat žádné mechanismy pro zajištění přenesení paketu. Nakonec jsem však zjistil, že by bylo možné implementovat nějakou techniku, která by toto zajistila. Z výsledků testů však usuzuji, že to není možné, protože jsem zaznamenal minimální počet rušení na sběrnici. Také jsem zjistil, že vhodnou úpravou hardwaru by bylo možné odstranit určité problémy v prvních okamžicích po připojení napájení. Pro úpravu hodnot napájení používá obvod MAX nábojovou pumpu. Ta vyžaduje kvalitní kondenzátory. Bohužel běžně používané elektrolitické kondenzátory nejsou pro přenos velkého množství údajů dostatečně kvalitní.

Jako další vylepšení je vhodné zvážit potřebu spolehlivého přenosu informace. Tato potřeba bude odlišná aplikaci od aplikace. V první řadě je nutné zvažovat míru rušení. RS-232 je navržen tak, aby odolával relativně vysokému rušení. Míra odolnosti záleží hlavně na vzdálenosti a přenosové rychlosti. Na delší vzdálenosti bych doporučoval spíše použití protokolu RS485 nebo RS422, která přenášejí data ve formě diferenciálního signálu. V případě, že by byla do komunikace vnášena relativně vysoká míra rušení, bylo by nutné buďto zavést metody pro zajištění spolehlivého přenosu, zvýšit počet opravných bytů nebo jednoduše snížit periodu mezi jednotlivými odesílání stavů.

Bohužel žádné řešení není stoprocentní, jelikož může dojít i k chybě při přenosu po I2C a zde není možné implementovat žádný korekční mechanizmus.

Také považuji za vhodné do budoucna používat kvalitnější kondenzátory (především tantalové) pro násobič napětí v obvodu MAX232. To by mělo vyloučit případné chyby na sběrnici vinou nedostatečného náboje na kondenzátorech.

V neposlední řadě je možné upravit program takovým způsobem, aby neprobíhala při startu inicializace ihned, ale až po prověření, zdali nejde pouze o watchdog reset. Tím by byla zajištěna možnost pokračovat v přerušené činnosti s posledními daty nebo maximálně se ztrátou posledního paketu.

I přes tyto některé nedostatky považuji převodník za kvalitní nástroj pro kontrolu velkého množství vodičů pomocí počítače případně pro přenos velkého množství dat na vyšší vzdálenosti.

Literatura

[1] OLMR, Vít. HW server [online]. 12.12.2005 [cit. 2010-05-11]. HW server představuje - Sériová linka RS-232. Dostupné z WWW: <http://hw.cz/rs-232>.

[2] HIMPE, Vince. Embedded Systems Academy [online]. c2010 [cit. 2010-05-11]. I2C (Inter- Integrated Circuit) Bus Technical Overview and Frequently Asked Questions. Dostupné z WWW:<http://www.esacademy.com/en/library/technical-articles-and-documents/miscellaneous/i2c-bus.html>.

[3] Datasheet : Atmega8A. : Atmel, 2009. 307 s. Dostupné z WWW: <http://www.atmel.cz/dyn/resources/prod_documents/doc8159.pdf>.

[4] Datasheet : PCF8574. Philips Semiconductors, 2002. 24 s. Dostupné z WWW: <http://www.nxp.com/documents/data_sheet/PCF8574.pdf>.

[5] MANN, Burkhard. C pro mikrokontroléry : ANSI-C, kompilátory C, spojovací programy - linkery, práce s ATMEL AVR a MSC-51, příklady programování v jazyce C, nástroje pro programování, tipy a triky. Vyd. 1. Praha : BEN - technická literatura, 2003. 279 s. ISBN 80-7300-077-6.

[6] MATOUŠEK, David. Práce s mikrokontroléry ATMEL ATmega16. 1. vyd. Praha : BEN - technická literatura, 2006. 319 s. ISBN 80-7300-174-8.

[7] VÁCLAV, Kadlec. Delphi : Hotová řešení. 1. vyd. Brno : Computers Press, 2003. 293 s. ISBN 80-251-0017-0.

[8] Communication protocol In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 24.12.2002, 12.5.2010 [cit. 2010-05-13]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Communication_protocol>.

[9] GOOK, Michael. Hardwarová rozhraní : Průvodce programátora. Brno : Computer Press, a.s., 2006. 456 s. ISBN 80-251-1019-2.

[10] DomIS [online]. 2005 [cit. 2010-05-12]. SerialNG. Dostupné z WWW: <http://www.domis.de/cms/index.php?module=ContentExpress&func=display&ceid=7>.

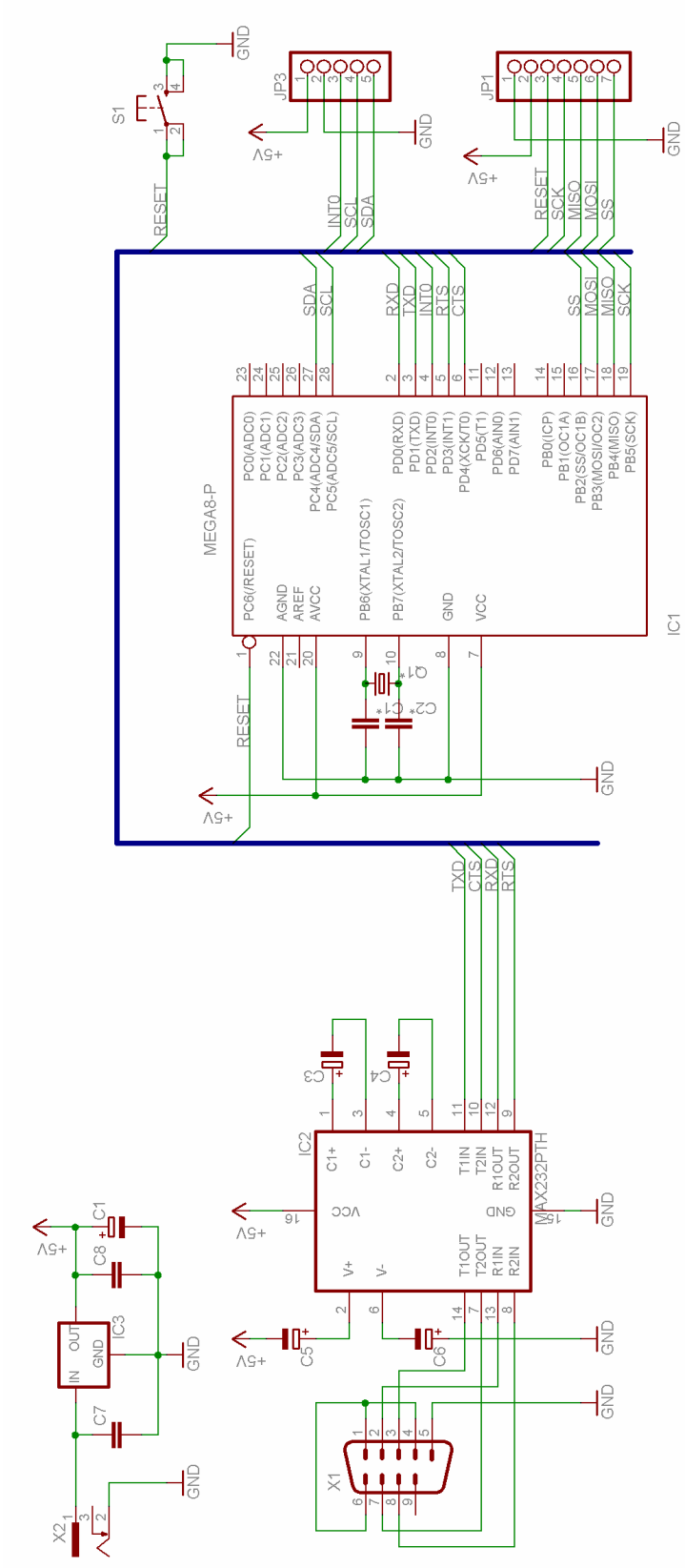
[11] Datasheet: 7805. Dostupný z WWW: <http://www.datasheetcatalog.org/datasheets/228/390068_DS.pdf>.

[12] WAGNER, Vlastimil. Wagnerovi.cz [online]. 27.1.2009 [cit. 2010-05-10]. AVRUSB500v2 - programátor AVR na USB. Dostupné z WWW: <http://www.wagnerovi.cz/view.php?nazevclanku=avrusb500v2-programator-avr-na-usb&cisloclanku=2009010001>.

[13] SOCHER, Guido. Tuxgraphics.org [online]. 31.5.2007 [cit. 2010-05-10]. AvrUsb500v2 -- an open source Atmel AVR Programmer, stk500 V2 compatible, with USB interface. Dostupné z WWW: <http://tuxgraphics.org/electronics/200705/article07052.shtml>.

Seznam příloh

Příloha 1


GND
S1
JP3
JP1
+5V
RESET
INT0
SCL
SDA
SCK
MISO
MOSI
SS
RXD
TXD
RTS
CTS
MEGA8-P
IC1
PC0(ADC0)
PC1(ADC1)
PC2(ADC2)
PC3(ADC3)
PC4(ADC4/SDA)
PC5(ADC5/SCL)
PC6(/RESET)
PD0(RXD)
PD1(TXD)
PD2(INT0)
PD3(INT1)
PD4(XCK/T0)
PD5(T1)
PD6(AIN0)
PD7(AIN1)
PB0(ICP)
PB1(OC1A)
PB2(SS/OC1B)
PB3(MOSI/OC2)
PB4(MISO)
PB5(SCK)
PB6(XTAL1/TOSC1)
PB7(XTAL2/TOSC2)
AGND
AREF
AVCC
VCC
C1
C2
C3
C4
C5
C6
C7
C8
IC2
MAX232PTH
C1+
C1-
C2+
C2-
T1IN
T2IN
R1OUT
R2OUT
V+
V-
T1OUT
T2OUT
R1IN
R2IN
IC3
IN
OUT
X1
X2

Příloha 2

- *source*
  - Složka obsahuje zdrojové soubory a projektové soubory pro AVRStudio
- *board*
  - Složka se schématy a návrhy plošných spojů
- *test*
  - Složka obsahuje programy pro testování převodníku
- *example*
  - Složka se ukázkami použití komunikačního protokolu
- *documentation*
  - Složka s dokumentací zdrojových kodu
- *thesis*
  - Složka s elektronickou verzí bakalářské práce
- *foto*
  - Složka s fotkami výsledného převodníku