Technická univerzita v Liberci

Fakulta Mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie

Studijní obor: 1802R007 – Informační technologie

Realizace jednoduchého uzlu RS485 s protokolem MODBUS

Bakalářská práce

Autor: Michal Štrick

Vedoucí práce: Ing. Josef Grosman

V Liberci 13. 5. 2012

Prohlášení

Byl jsem plně seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména §60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiju-li bakalářskou práci, nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce.

Datum:

Podpis:

Poděkování

Chtěl bych poděkovat všem, kteří se jakýmkoliv způsobem podíleli na vzniku mé práce. v první řadě bych chtěl poděkovat vedoucímu své bakalářské práce Ing. Josefovi Grosmanovi za rady, trpělivost a svůj čas. Nemalý dík patří i mé rodině, která mě podporovala fyzicky i psychicky a pečovali o mé pohodlí při práci.

V Liberci dne 13. 5. 2012

Abstrakt

Tato bakalářská práce se zabývá návrhem, realizací a otestováním jednoduchého uzlu sběrnice RS485 s použitím protokolu MODBUS s následujícími vlastnostmi: možnost změny adresy uzlu, měření fyzikální veličiny, možnost nastavení režimu protokolu. První část práce se zabývá možnostmi návrhu obecného uzlu s procesorem řady 8051. v další části práce je již proveden konkrétní návrh zapojení uzlu, následně je popsáno, jak proběhla jeho realizace a nakonec jsou popsány úseky kódů programu, který byl použit pro otestování výsledného uzlu.

Klíčová slova:

uzel RS485, MODBUS protokol, 8051, snímání fyzikální veličiny

Abstract

This thesis deals with design, implementation and testing of simple node for serial bus RS485 using MODBUS protocol with the following properties: possibility of changing node address, measurement of physical quantity, possibility of mode setting protocol. The first part deals with general design options of the node with a processor 8051 series. In the following part of the work is made a specific design of node connection, then there is described how was its realization made and the end describes the sections of code that was used to test final node.

Keywords:

Node of RS485, MODBUS protocol, 8051, sensing of physical quantities

Obsah

Seznam obrázků

1 Úvod

V dnešní době se velmi rozmáhá použití mikroprocesorů pro nejrůznější úlohy. Kladou se na ně různé požadavky, jako např. na spotřebu, výkon, integrované periferie, odolnost proti vnějším vlivům apod. Často se také požaduje možná komunikace se stolním počítačem či mezi více mikroprocesory různých výrobců a typů na určitou vzdálenost. k tomu se může použít nejrůznější technologie. Každá je vhodná pro jiné prostředí. Nejčastěji se používají sběrnice, ať už sériové či paralelní, s jedním masterem nebo s více, polo duplexní či plně duplexní.

Úkolem první části této bakalářské práce je navrhnout a realizovat jednoduchý uzel na sériové sběrnici RS485 s mikroprocesorem řady 8051 pracující v linkové vrstvě jako SLAVE, v aplikační vrstvě jako SERVER. Jelikož tato sběrnice nemá definován formát zpráv, bude proto použit otevřený protokol MODBUS, který řeší tento problém. Výsledný uzel bude měřit vybranou fyzikální veličinu, bude mít možnost změny adresy a možnost změny nastavení režimu ASCII a RTU.

V druhé části práce se funkčnost výsledného uzlu ověří pomocí výukových přípravků s mikroprocesory AT89C51CC03 řady 8051, s funkcí MODBUS MASTER. Program jak pro master s tímto výukovým přípravkem, tak pro slave s realizovaným uzlem, bude psán v jazyce C ve zkušební verzi vývojového prostředí uVision od firmy Keil.

2 Cíle práce

Cílem této práce je navrhnout jednoduché zařízení pro studijní účely, které bude schopno komunikovat po sériové sběrnici RS485 ve funkci slave a s možností nastavení režimu protokolu MODBUS, který tento přípravek bude používat. Dále musí poskytovat možnost změny adresy a musí mít připojen senzor fyzikální veličiny. Celý obvod bude řídit procesor řady 8051. Tento navržený přípravek se zrealizuje a jeho funkčnost se otestuje.

3 Technické prostředky

3.1 RS-485

Sériová sběrnice RS-485 se používá především v průmyslovém prostředí. Tento standart má možnost mít až 32 uzlů na sběrnici bez použití opakovačů. Je zde jeden master, který řídí veškerou komunikaci na sběrnici, a jeden nebo více slavů, každý s unikátní adresou. Sběrnici tvoří dva vodiče, často označované jako a a B. Logické úrovně jsou reprezentovány rozdílem napětí mezi těmito vodiči, což je výhodné zejména kvůli zvýšené odolnosti proti rušení (rušivý signál se indukuje na obou vodičích současně a rozdílem dvou stejně zkreslených signálů se toto zkreslení eliminuje).

Tato dvouvodičová verze je však pouze jednosměrná, tudíž je potřeba přepínat mezi vysíláním a příjmem. Tento problém řeší čtyřvodičová verze, která je plně duplexní, čili obousměrná. Nevýhodou je však vyšší počet vodičů. Na delší vzdálenosti je také nutné propojit i uzemnění všech zařízení mezi sebou, což také zvýší celkový počet vodičů o jeden. Současně s tím se i na konce vedení musejí připojit zakončovače neboli terminátory, které zabraňují odrazům signálu na konci vedení.

Data se přenáší po 7 nebo 8 bitech. Pokud je sběrnice v klidovém stavu, vyšle se start bit, následují data, poté volitelně paritní bit, jeden či více stop bitů a nakonec opětné uvedení do klidového stavu. Standart RS-485 však nemá definován formát zprávy. Je tedy nutné použít nějaký protokol, který tuto chybějící část nahradí. Mimo to nemá tento standart definovány ani konektory, tudíž při použití nějakého vhodného konektoru se musí dát pozor na správné zapojení vodičů a a B (a případně i uzemnění).

3.2 Protokol MODBUS pro RS-485

Modbus je otevřený protokol pro komunikaci mezi zařízeními na principu klient-server na principu předávání zpráv. Tento protokol není přímo spjatý s jednou sítí či sběrnicí. Následuje popis protokolu Modbus používaný na sběrnici RS-485.

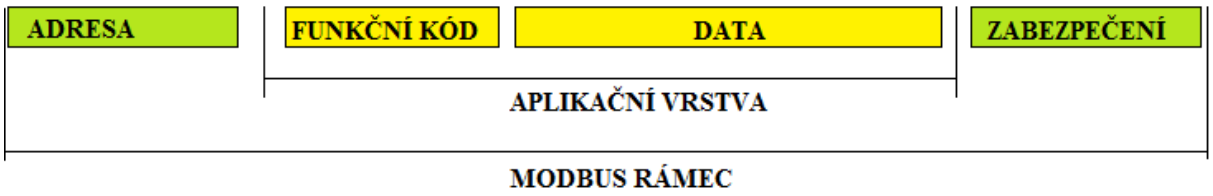
3.2.1 Základní popis

Modbus pro RS-485 funguje na linkové vrstvě OSI modelu na principu master-slave, v aplikační vrstvě potom jako klient-server. Na sběrnici se musí nacházet právě jeden master a jeden až 247 slavů, každý s unikátní adresou v rozmezí 1 až 247. Samotný master nemá žádnou adresu a inicializuje veškerou komunikaci na sběrnici. Slave nesmí vysílat na sběrnici

bez přijetí požadavku od masteru a nemůže ani komunikovat s jiným slave zařízením. Každý slave přijme zprávu pouze jemu určenou, tedy pokud je adresa příjemce shodná s adresou slavu, nebo pokud se jedná o broadcast zprávu (adresa příjemce je 0). Při broadcast zprávě však nesmí slave vyslat odpověď. Tuto zprávu totiž dostanou všechna slave zařízení, takže by začala vysílat všechna najednou a na sběrnici by došlo ke kolizi.

3.2.2 Jednotka datového protokolu (PDU)

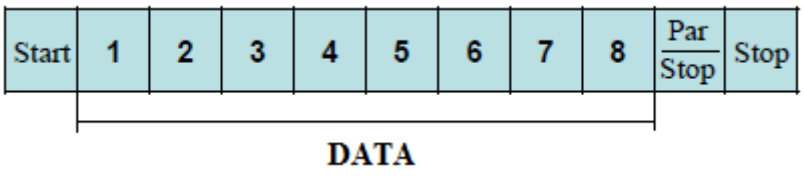
Datová jednotka je v aplikační vrstvě složena z funkčního kódu a dat. Funkční kód udává, jaká operace se má s danými daty vykonat. Jedná se například o čtení jednoho bitu nebo zápis skupiny registrů. Linková vrstva dále přidá na začátek rámce adresu slavu a na konec rámce přidá zabezpečení. Pokud slave odpovídá na dotaz od mastera, potom do části s adresou umístí svojí adresu, aby tak master poznal, od kterého slavu přišla odpověď. Zabezpečení je rozdílné pro jednotlivé režimy. RTU režim (binární režim) používá CRC zabezpečení, naproti tomu ASCII (znakový) režim používá LRC zabezpečení (kontrolní součet). Celý rámec viz Obrázek 3.1.



Obrázek 3.1 MODBUS PDU

3.2.3 RTU mód

RTU mód je binární režim posílání zpráv. Vysílání rámce může začít, pokud je sběrnice v klidovém stavu alespoň po dobu, za kterou by se vyslalo 3,5 znaku. v tomto režimu je 8 bitový byte odeslán jako jeden znak. Formát jednoho znaku viz Obrázek 3.2.



Obrázek 3.2 Formát znaku

Celý rámec má délku omezenou na maximálně 256 bytů. z toho je 1 byte adresa, 1 byte funkční kód, N bytů dat, 2 byty zabezpečení CRC. z toho vyplývá, že maximální délka dat je 252 bytů. Formát celého rámce včetně uvedených délek jednotlivých bloků a délek nutného klidového stavu na sběrnici viz Obrázek 3.3.

ZAČÁTEK	ADRESA SLAVU	FUNKČNÍ KÓD	DATA	CRC	KONEC
>=3,5 ZNAKU	1 byte	1 byte	0 až 252 bytů	2 byty	>=3,5 ZNAKU
	MAX 256 bytů				

Obrázek 3.3 Formát RTU rámce

Vysílání rámce se ukončí uvedením sběrnice do klidového stavu po dobu minimálně 3,5 znaku. Klidová doba mezi vysíláním jednotlivých znaků nesmí přesáhnout 1,5 znaku.

Zabezpečení CRC se počítá z adresy, funkčního kódu a dat. RTU režim je oproti ASCII režimu rychlejší, proto je vhodný pro průmyslové sběrnice.

3.2.4 ASCII mód

ASCII mód je znakový režim posílání zpráv. Vysílání rámce začíná odesláním znaku dvojtečky s ASCII hodnotou 3A hexadecimálně (58 desítkově). Jeden 8 bitový byte se odešle jako 2 znaky. Konec rámce se určí vysláním dvojice řídících znaků CR a LF (hodnoty 10 a 13 desítkově). Formát jednoho znaku viz Obrázek 3.2. Každý 8 bitový byte, který se má odeslat, se rozdělí v půlce na dvě čísla po 4 bitech. Každé takové 4 bitové číslo se převede na osmibitový znak, který odpovídá tomuto číslu. Pokud tedy chci odeslat například číslo 75 (binárně 0100 1011 v 8 bitech), tak se toto číslo rozdělí po 4 bitech na číslo 4 (0100 binárně) a číslo 11 (1011 binárně). v hexadecimálním tvaru máme tedy čísla 4 a A. To se převede pomocí ASCII tabulky na hodnoty znaků. Ve výsledku se tedy přenese číslo 52, které odpovídá ASCII hodnotě číslice 4, a číslo 65, které odpovídá znaku A.

Celková délka rámce bez znaku zahajujícího komunikaci a včetně znaků pro indikaci konce rámce je 512 bytů. z toho tvoří 2 byty adresa, 2 byty funkční kód, N bytů pro data, 2 byty pro kontrolní součet (LRC) a 2 byty pro označení konce rámce. Data tak mají maximální délku 2 * 252 bytů. Formát celého rámce včetně uvedených délek jednotlivých bloků viz Obrázek 3.4

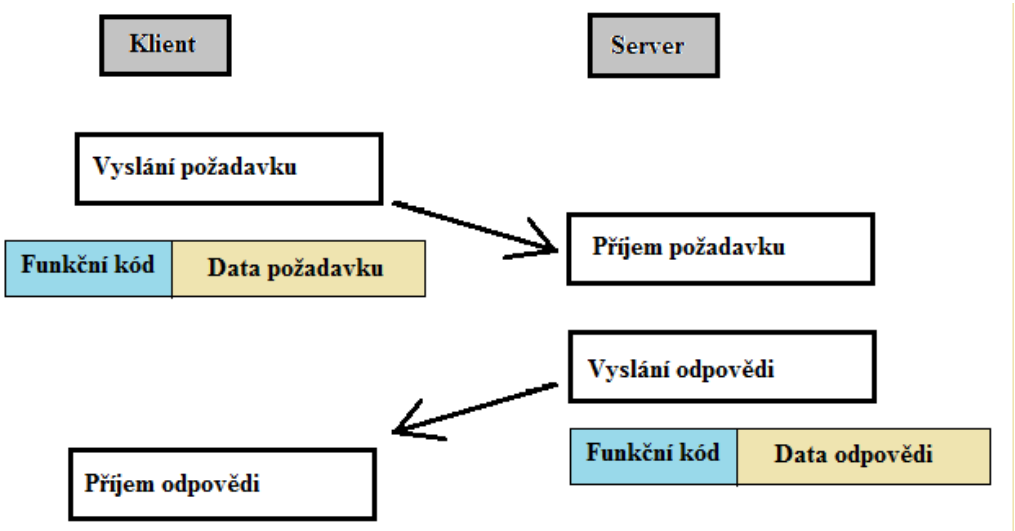
ZAČÁTEK	ADRESA SLAVU	FUNKČNÍ KÓD	DATA	LRC	KONEC
1 znak : 3Ah	2 znaky	2 znaky	0 až 2x252 znaků	2 znaky	2 znaky CR, LF 0Dh, 0Ah

Obrázek 3.4 Formát ASCII rámce

Kontrolní součet LRC se vypočítá z adresy, funkčního kódu a dat. ASCII režim je oproti RTU režimu pomalejší, protože musí přenášet dvojnásobek dat, mezi vysláním dvou znaků ale může být delší mezera než v RTU.

3.2.5 Aplikační vrstva protokolu MODBUS

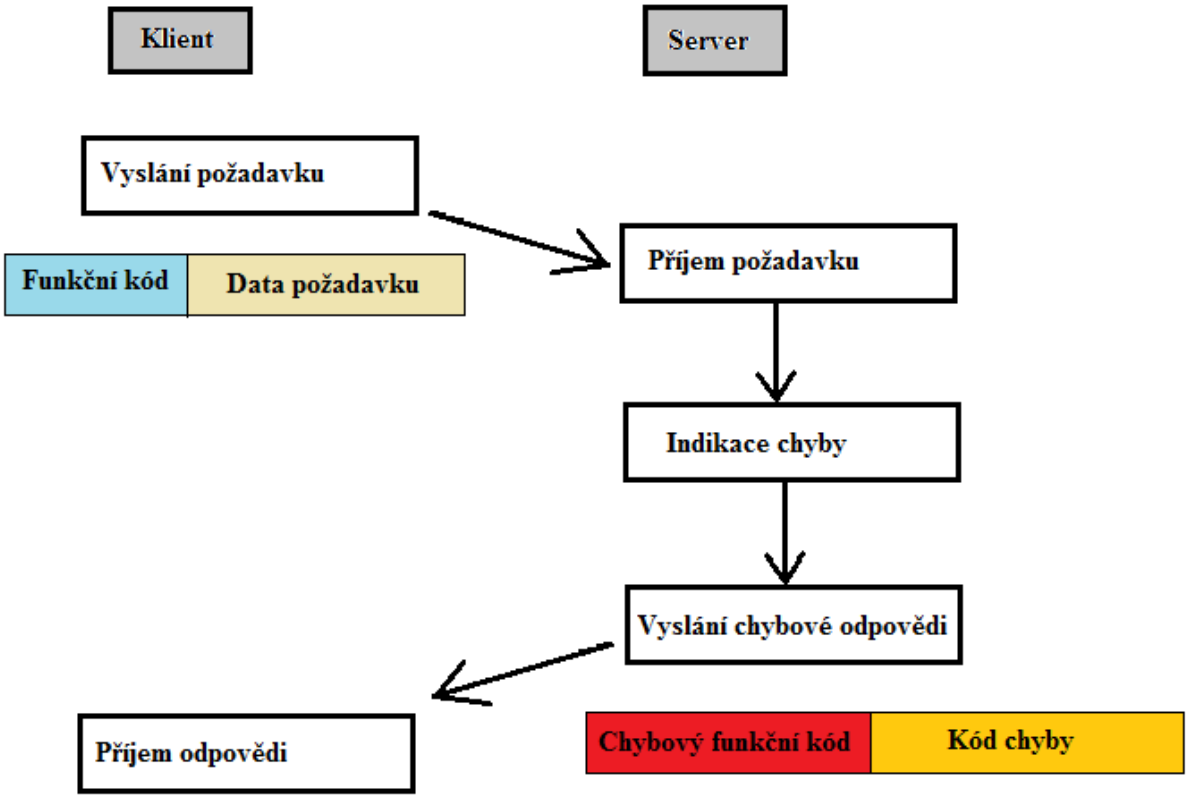
V aplikační vrstvě funguje protokol MODBUS na principu klient – server. Princip normální komunikace (když nedojde k chybě) viz Obrázek 3.5.



Obrázek 3.5 MODBUS komunikace v aplikační vrstvě

Pokud slave indikuje chybu, odešle zpět chybovou zprávu s chybovým funkčním kódem v části funkčního kódu a číslem chyby v datové části (viz Obrázek 3.6). Chybový funkční kód se vypočítá přičtením čísla 128 (hexadecimálně 80) k funkčnímu kódu požadavku. v aplikační vrstvě může dojít ke třem druhům chyb:

- Kód chyby 01, Neznámý kód funkce (Slave nezná požadovanou funkci)
- Kód chyby 02, Neznámý objekt (Slave nezná požadovanou adresu objektu)
- Kód chyby 03, Chyba dat objektu (Zapisovaná data jsou mimo rozsah)



Obrázek 3.6 Indikace chyby v aplikační vrstvě

3.2.6 Hlavičky metod MODBUS knihovny v jazyce C

Implementace tohoto protokolu v jazyce C je realizována několika funkcemi pro oba režimy. Veškeré funkce pracují nad bufferem, který je zde reprezentován ukazatelem na pole prvků datového typu byte (resp. char).

Znakový režim je složen z následujících funkcí:

- Funkce pro převod mezi znaky a binárními hodnotami:

```
byte AHex(byte c);
byte HexAsc(byte b);
```

- Pomocné funkce pro operace s daty:

```
byte WrWord(word val,byte *bf);
word RdWord(byte *bf);
byte MbRdByte(byte *bf);
word MbRdWord(byte *bf);
byte MbWrByte(byte b,byte *bf);
byte MbWrWord(word w,byte *bf);
```

- Funkce pro master a funkce pro slave:

```
byte MbRd(byte adr,byte fce,word reg,word val,byte *bf);
```

```
byte MbWrOne(byte adr,byte fce,word reg,word val,byte *bf);
byte MbWr(byte adr,byte fce,word reg,word nbr,byte *vals,byte *bf);
byte MbAnsWr(byte adr,byte fce,word reg,word val,byte *bf);
byte MbAnsRd(byte adr, byte fce, byte bytes, byte *vals,byte *bf);
byte MbAnsErr(byte adr,byte fce,byte er,byte *bf);
```

- Funkce pro zabezpečení LRC a funkce pro zapsání znaků CR a LF:

```
byte MbLrc(byte *bf,byte len);
byte MbWrEoT(byte *bf);
```

Binární režim je složen z těchto funkcí:

- Pomocné funkce pro operace s daty:

```
byte WrWord(word val,byte *bf); word RdWord(byte *bf);
word MrtuRdCrc(byte *bf);
byte MrtuWrCrc(word crc,byte *bf );
```

- Funkce pro master a funkce pro slave:

```
byte MrtuWr(byte adr,byte fce,word reg,word nbr,byte *vals,byte *bf);
byte MrtuWrOne(byte adr,byte fce,word reg,word val,byte *bf);
byte MrtuRd(byte adr,byte fce,word reg,word val,byte *bf);
byte MrtuAnsErr(byte adr,byte fce,byte er,byte *bf);
byte MrtuAnsRd(byte adr,byte fce,byte reg,byte *vals,byte *bf);
byte MrtuAnsWr(byte adr,byte fce,word reg,word val,byte *bf);
```

- Funkce zabezpečení CRC:

```
word MrtuCrc(byte *bf, byte len);
```

Pomocí všech těchto funkcí se dá napsat program pro komunikaci po sběrnici, který bude používat protokol MODBUS.

4 Návrh řešení

4.1 Napájení obvodu

Pro funkčnost obvodu musí být každá součástka (která to vyžaduje) připojena na napájecí napětí. Toto napětí může být pro každou součástku odlišné. Nejčastěji se používá napětí TTL logiky 5V nebo napětí 3,3V. Je vhodné používat součástky se stejným napájecím napětím. Zdroj napájecí napětí může být například z připojení k USB. v takovém případě je nutno mít obvod stále připojen k počítači. Další možností mohou být baterie, které se však časem vybijí a poté je nutno je vyměnit. Také je zde možnost připojení přímo ze sítě 230V střídavých, samozřejmě s použitím nějakého převodníku na stejnosměrné napětí požadované velikosti (například 5V).

4.2 Procesor

Hlavní součástí výsledného uzlu je procesor. Řídí veškerou funkcionalitu obvodu. Proto je důležité vybrat správný procesor, který splňuje všechny požadavky na něj kladeny.

4.2.1 Požadavky na procesor

Jelikož úkolem uzlu je snímat fyzikální veličinu, bude muset vybraný procesor obsahovat převodník z analogového signálu na digitální (A/D převodník). v případě snímání z více analogových senzorů by měl procesor mít A/D převodník na více vstupech (pinech), nebo mít jeden A/D převodník s multiplexovanými (přepínanými) vstupy. v případě použití digitálního senzoru však požadavek na A/D převodník odpadá.

Pro připojení ke sběrnici RS485 musí mít procesor rozhranní sériové sběrnice, které se pak může snadno pomocí externího převodníku připojit přímo na RS485.

Pro implementaci režimů RTU a ASCII protokolu MODBUS a komunikaci po sběrnici jsou potřeba 2 čítače/časovače. Jeden je zapotřebí na určení komunikační rychlosti sběrnice. Jsou i jiné možnosti pro určení rychlosti sběrnice, jako například režim, kdy je místo čítače použita vydělená frekvence oscilátoru. To však není tolik flexibilní, proto jsem do požadavků zahrnul pro tuto funkci čítač/časovač. Druhý čítač/časovač je pouze pro režim RTU. Jeho úkolem je měřit časové rozpětí mezi jednotlivými znaky zprávy a indikovat konec zprávy.

4.2.2 AT89C51CC01

Tento procesor od firmy Atmel splňuje všechny požadavky. Má 10bitový A/D převodník s 8 multiplexovanými vstupy, jednu sériovou sběrnici UART a tři čítače/časovače. Má i sběrnici CAN, která je však pro tento návrh poněkud zbytečná. Piny pro připojení na tuto sběrnici však lze využít i jako obyčejné vstupně-výstupní piny.

4.2.3 DS89C450

Procesor DS89C450 od firmy Maxim má dva sériové kanály UART, což by zjednodušilo ladění programu, pokud by byl procesor programován přímo na desce. Jedna sběrnice by se tak využila pro programování procesoru a druhá sběrnice pro připojení k RS485. Tento procesor však nemá A/D převodník, proto by v případě jeho použití musel být použit digitální senzor fyzikální veličiny. DS89C450 také obsahuje tři 16bitové čítače/časovače.

4.2.4 AT89C51ED2

Tento procesor je opět od firmy Atmel a má jednu sběrnici UART a tři čítače/časovače. Stejně jako DS89C450 nemá A/D převodník, tudíž by v případě jeho použití musel být použit digitální senzor fyzikální veličiny.

4.3 Naprogramování procesoru

Pro funkčnost uzlu se do procesoru musí nahrát program. To se může provést dvěma způsoby, a to přímo na desce bez nutnosti další manipulace s procesorem, nebo na programovací desce, s tím, že se procesor bude muset přendávat.

4.4 Volba adresy uzlu

Dalším požadavkem na výsledný obvod je možnost změny adresy. Ta se může měnit buď za běhu, nebo pouze po resetu. Tyto způsoby se realizují až v programu, stejně jako ošetření její platnosti (slave nemůže mít například adresu 255). Jelikož je adresa osmibitová, může být pro její volbu použito 8 přepínačů (switchů). Pokud by byly zapojeny přímo na piny procesoru, bylo by potřeba 8 takových pinů. Pro snížení počtu připojených pinů lze použit multiplexor, který redukuje počet připojených vývodů na piny procesoru na polovinu (tedy na 4). Jedná se o 1 datový vodič a 3 adresové. Jedinou nevýhodou je komplikovanější čtení adresy v programu. Další možností by mohlo být maticové zapojení přepínačů, které by sice vyžadovalo 6 připojených vývodů, ale poskytovalo by o něco snadnější čtení stavů přepínačů.

Pro volbu adresy by se mohla použít i maticová klávesnice 4x4. Pro zadání adresy by se tak použila tlačítka s číslicemi 0 až 9.

4.5 Vyvedení RS485

Protože standart RS-485 nemá definované konektory, lze použít téměř jakýkoliv. Vzhledem k tomu, že výsledný obvod bude muset být schopen komunikovat s výukovými přípravky, které jsou již hotové a mají již připájený konektor na vyvedení této sběrnice, je nejvhodnější použít právě tento konektor. Jedná se o 3 pinový konektor se zámkem.

4.6 Senzor fyzikální veličiny

Výsledný výrobek má snímat fyzikální veličinu. To může být například teplota, tlak, vlhkost vzduchu, intenzita světla apod. Tyto senzory se vyrábějí v mnoha pouzdrech, nejčastěji však v pouzdrech DIL (dual-in-line, viz Obrázek 4.1), nebo v pouzdře TO-92 (pouzdro se třemi vývody, často používaný i pro tranzistory, viz Obrázek 4.2). Senzory také mohou být analogové, kdy je snímaná veličina reprezentována na výstupu senzoru velikostí napětí či proudu, nebo digitální.

Nejčastěji používaným senzorem v podobných obecnějších aplikacích bývá senzor teploty. Proto v následujících podkapitolách popíšu několik vybraných senzorů teploty, analogových i digitálních.

4.6.1 Analogový senzor teploty

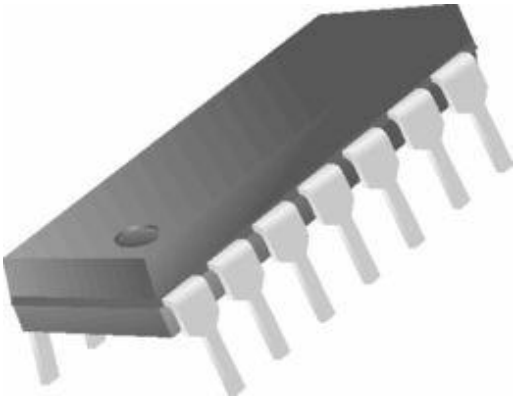
Mezi analogové teplotní senzory patří například TMP36. Jedná se o nízkonapěťový analogový senzor s přesností ±2°C. Na výstupu pak generuje napětí v rozmezí 2,7V až 5,5V, kde změna teploty o 1°C je reprezentována změnou výstupního napětí o 0,1V. První nevýhodou takovýchto senzorů je nutnost použití AD převodníku, pokud chceme hodnotu teploty dále digitálně zpracovávat. Druhou nevýhodou je jeho generování šumu, což nám znepříjemní další práci.

4.6.2 Digitální senzory teploty

Digitální senzory teploty mají různé metody, jak reprezentovat naměřenou hodnotu teploty na svém výstupu. Je zde například senzor TMP04, který na svém výstupu generuje digitální pulzy o frekvenci 35Hz. Výsledná hodnota teploty se poté určí z poměru délky logické 1 a logické 0 v jedné periodě výstupního signálu.

Další digitální teploměry mohou být TC74, který se připojuje pomocí sběrnice I2C, či senzor DS18B20, který se připojuje pro změnu přes sběrnici 1-wire.

Digitální senzory odstraňují nevýhody analogových senzorů, tedy nutnost použití AD převodníku a generování šumu.

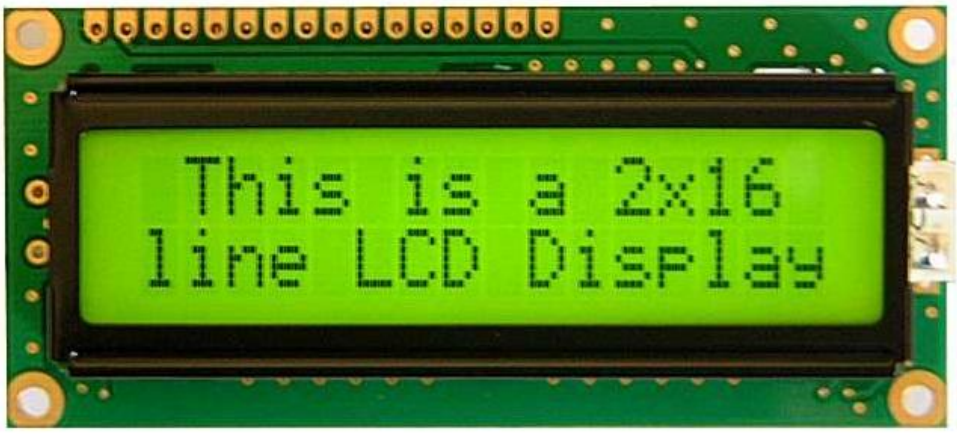


Obrázek 4.1 Příklad pouzdra DIL. Zdroj: [3]

Obrázek 4.2 Pouzdro TO-92. Zdroj: [4]

4.7 Ostatní periferie a možnosti

Výsledný výrobek může dále obsahovat celou řadu dalších periferií, které například umožní lepší možnosti pro ladění programu pro tento slave. Může se jednat o LED diody, LCD displej apod. v případě použití více LED diod je lepší použít obvod, který zredukuje počet pinů nutných k připojení diod. Může se jednat například o sériovo-paralelní posuvný registr. LCD displej by mohl být použit obyčejný 16x2 (16 sloupců a dva řádky) s podsvícením (viz Obrázek 4.3)



Obrázek 4.3 LCD displej 16x2. Zdroj: [5]

4.8 Usazení součástek

Pro správnou funkčnost obvodu musejí být všechny součástky usazeny na nějaké propojovací pole. Zde je opět několik možností, jak toto realizovat. v první řadě připadá v úvahu deska plošných spojů, ať už jednostranná nebo oboustranná. Nevýhodou desky plošných spojů je velice nízká flexibilita (po výrobě se téměř nedá změnit). To znamená, že při chybě v návrhu schématu se po opravení chyby musí většinou vyrobit deska nová.

Další možností je nepájivé pole. To odstraňuje nedostatek nízké flexibility. Oproti desce plošných spojů je však velice nepřehledné a navíc v případě vypadnutí jednoho propojovacího drátku se těžko zjišťuje, kde tento drátek chybí. Nepájivé pole také omezuje použité součástky. Nelze například použít SMD pouzdra (viz Obrázek 4.4) či pouzdra s více než dvěma vývody ve vertikálním i horizontálním směru (například pouzdra PLCC).

Obrázek 4.4 příklad SMD pouzdra Zdroj: [6]

Při použití procesoru řady (A)T89C51xD2 se může pro usazení součástek použít univerzální vývojový prostředek ED-2 Kit (viz Obrázek 4.5). Ten kombinuje výhody nepájivého pole (dostatečnou flexibilitu) a desky plošných spojů (součástky jsou připájeny

k desce). Veškerá propojení jsou realizována propojovacími drátky připájenými k vývodům součástek.

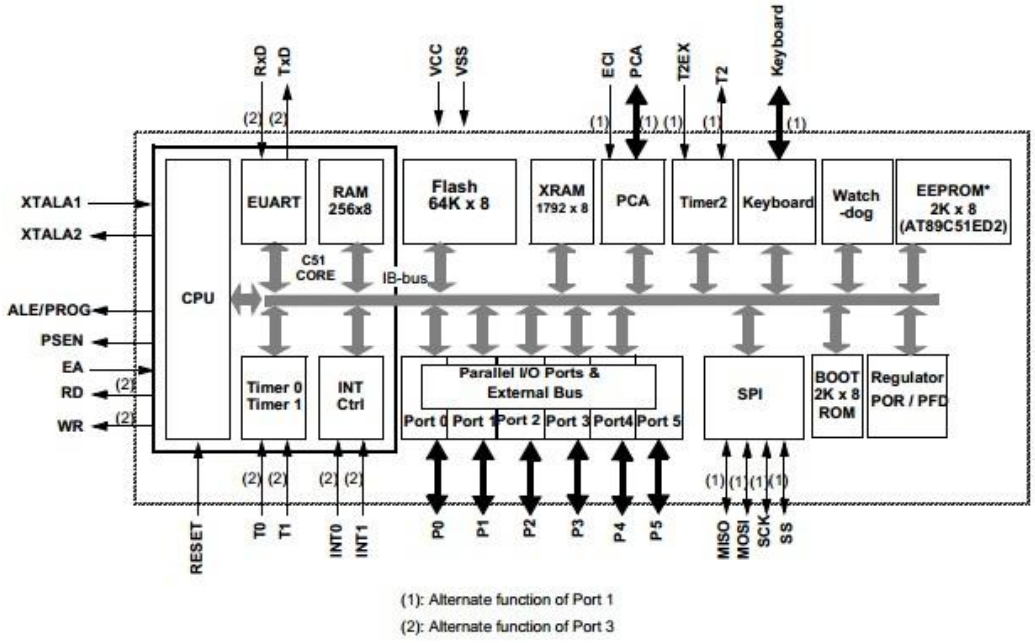
Obrázek 4.5 ED-2 Kit, Zdroj: [5]

5 Sestavení schématu obvodu

Při realizaci zadání se nejprve musí navrhnout schéma, podle kterého se následně sestrojí výsledný prostředek.

5.1 Zvolený procesor

Pro účely této práce jsem se rozhodl použít procesor AT89C51ED2 od firmy Atmel v pouzdře DIL40 (pouzdro DIL se 40 piny). Jak je vidět na blokovém schématu procesoru (viz Obrázek 5.1), AT89C51ED2 nemá A/D převodník. Pro měření fyzikální veličiny se tak bude muset použít digitální senzor, který využije PCA (programovatelné čítačové pole) modul v procesoru (na portu 1).



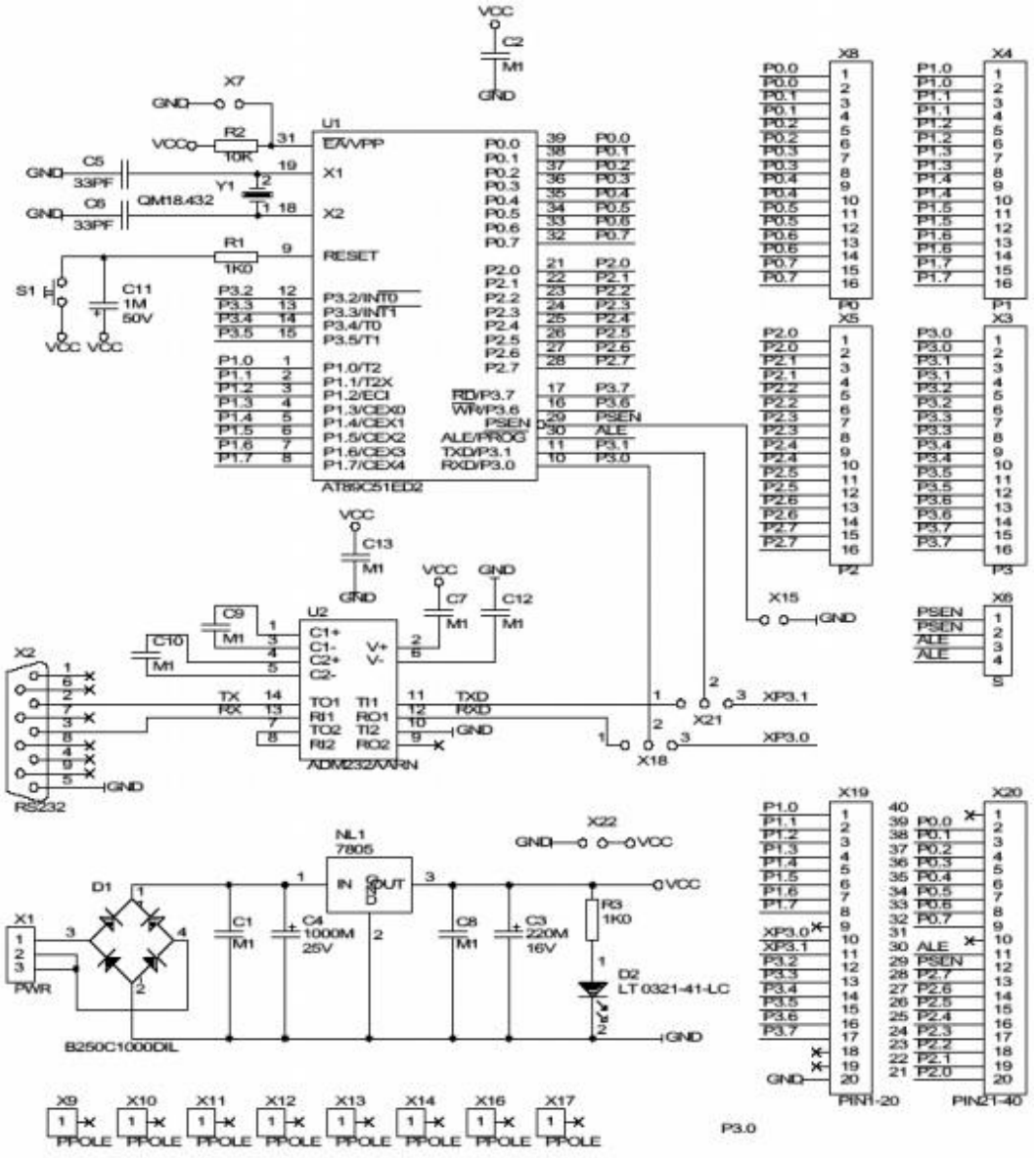
Obrázek 5.1 Blokové schéma procesoru AT89C51ED2, Zdroj: [8]

Tento procesor potřebuje k vykonání strojového cyklu díky funkci X2 pouze 6 hodinových cyklů. To zdvojnásobí výkon procesoru při zachování stejné frekvence krystalu. Kvůli zpětné kompatibilitě s C51 lze tuto funkci softwarově vypnout.

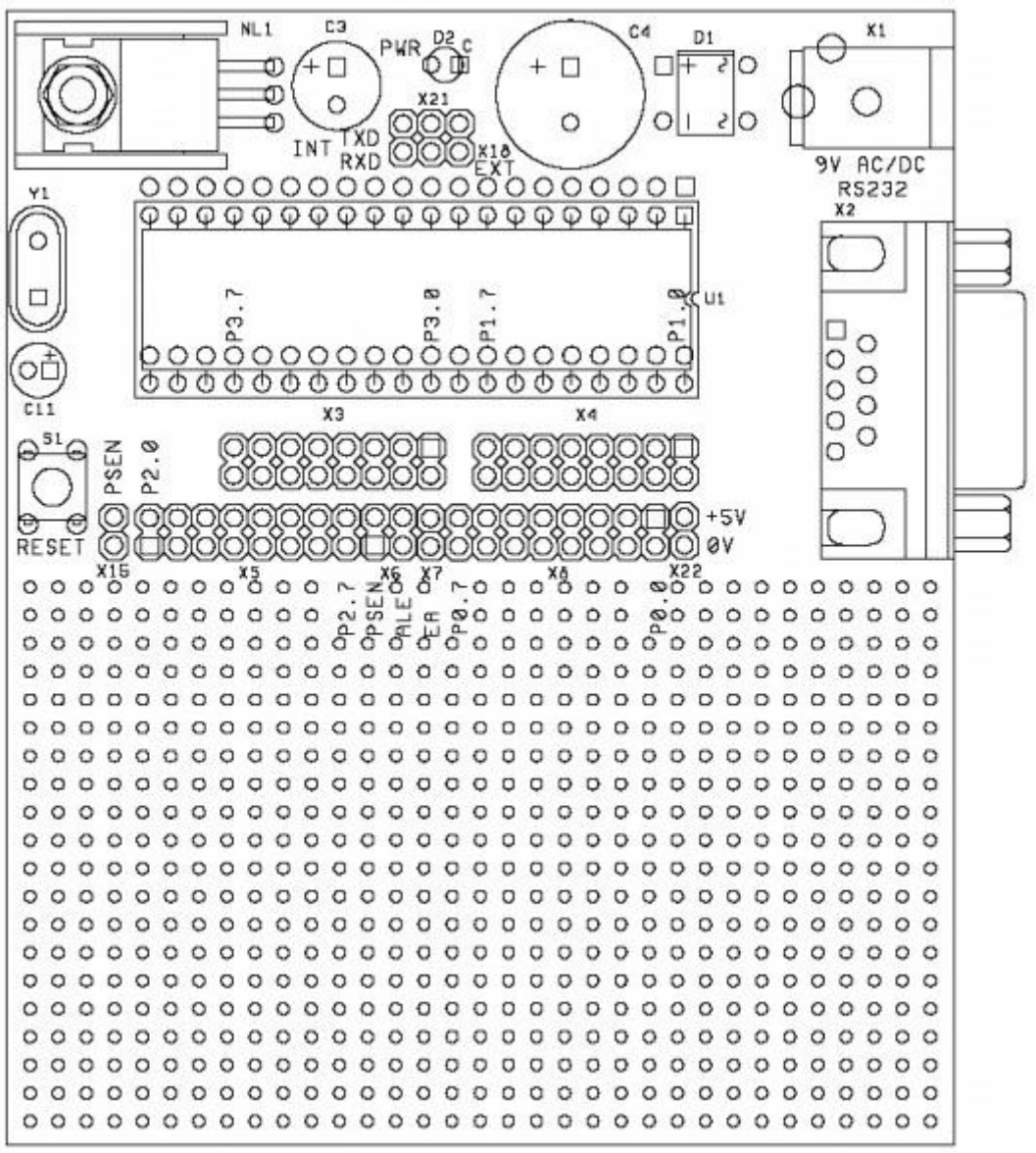
Systém přerušení procesoru poskytuje čtyř-úrovňovou prioritu přerušení. Zdroje přerušení mohou být 3 čítače/časovače, UART sběrnice, PCA, SPI sběrnice, klávesnice a 2 piny pro externí přerušení.

5.2 Usazení součástek

Jelikož ve výsledném prostředku bude použit procesor AT89C51ED2, lze použít univerzální vývojovou desku ED-2 Kit. Tato deska zajistí mimo jiné připojení resetu procesoru, připojení oscilátoru o frekvenci 11,0592 MHz, možnost propojení s počítačem přes sběrnici RS232 a nahrání programu z počítače do procesoru (přes toto propojení). Deska je napájena pomocí síťového adaptéru 230/7-9V s kladným pólem vyvedeným na prostřední kolík. Na samotné desce je pak 5V stabilizátor napětí. Schéma desky viz Obrázek 5.2 (na tomto schématu je krystal o frekvenci 18,432 MHz), osazovací výkres desky viz Obrázek 5.3.



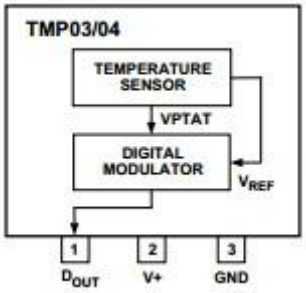
Obrázek 5.2 Schéma ED-2 Kitu. Zdroj: [7]



Obrázek 5.3 Osazovací výkres ED-2 Kitu. Zdroj: [7]

5.3 Měření fyzikální veličiny

Pro tento přípravek jsem jako fyzikální veličinu, kterou obvod bude měřit, zvolil teplotu s jednotkami °C. Původně jsem plánoval použití analogového teplotního senzoru TMP36. Tento senzor však způsobuje šum, který se musí složitě přes různé filtry odstranit. Vybraný procesor navíc nemá A/D převodník, proto bylo nutno použít digitální senzor, jako například TMP04FT9 v pouzdře TO-92, který jsem se nakonec rozhodl použít. Tento senzor má přesnost ±1,5°C pro teploty od −25°C do +100°C. Dle specifikací může měřit teploty od −40°C do +100°C. Blokové schéma a pořadí vyvedených vývodů viz Obrázek 5.4.



Obrázek 5.4 Blokový diagram senzoru teploty TMP04FT9. Zdroj: [9]

Tento teploměr generuje na výstupu obdélníkové pulzy, z jejichž délky se určí teplota okolí. Tyto pulzy se generují na frekvenci 35Hz, což je dostatečně nízká frekvence na dostatečně vysokou přesnost určení výsledné teploty. Příklad generovaných pulzů viz Obrázek 5.5, vzorce pro výpočet teploty viz vzorce níže. Pro tento senzor tedy bude využit PCA modul na procesoru. Ten se nachází na portu 1, na pinech 3 až 7. Zapojení teploměru (realizováno v programu Eagle) viz Obrázek 5.6.

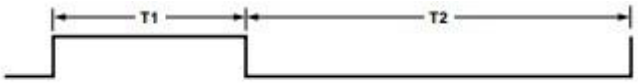


Figure 2. TMP03/TMP04 Output Format

Obrázek 5.5 Příklad generovaných pulzů na výstupu TMP04FT9. Zdroj: [9]

$$T(°\mathrm{C}) = 235 - \left(\frac{400 * T_1}{T_2}\right)$$

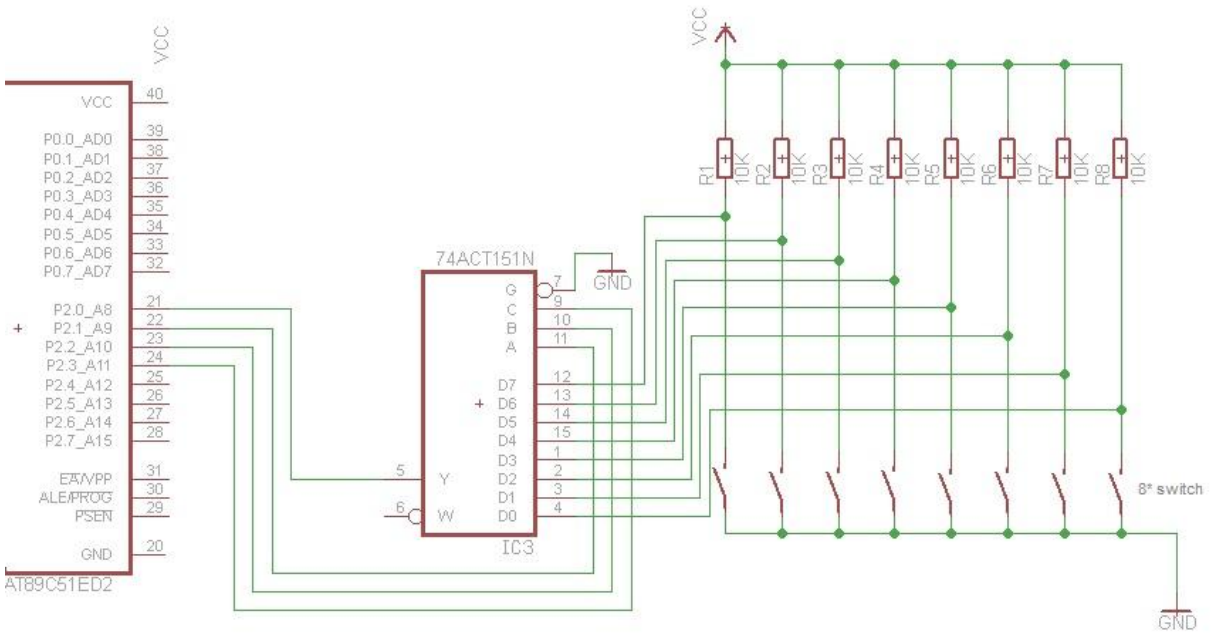
$$T(°\mathrm{F}) = 455 - \left(\frac{720 * T_1}{T_2}\right)$$

Obrázek 5.6 Zapojení teploměru TMP04FT9

5.4 Volba adresy uzlu

Pro volbu adresy jsem zvolil 8 posuvných přepínačů (konkrétně 1 sadu s osmi posuvnými přepínači, v schématu je to však realizováno jako 8 samostatných přepínačů). k procesoru jsem je připojil přes multiplexor (integrovaný obvod 74151) s 8 datovými vstupy, 3 adresovými vstupy a jedním datovým výstupem. Tento multiplexor po zadání adresy na

adresových vodičích má průměrné zpoždění 35ns, než se na výstupu objeví hodnota vstupu na dané adrese. Schéma připojení přepínačů k procesoru přes multiplexor viz Obrázek 5.7 (schéma vytvořeno v programu Eagle). Adresa je tedy zadávána v binární podobě (dvojkové soustavě).



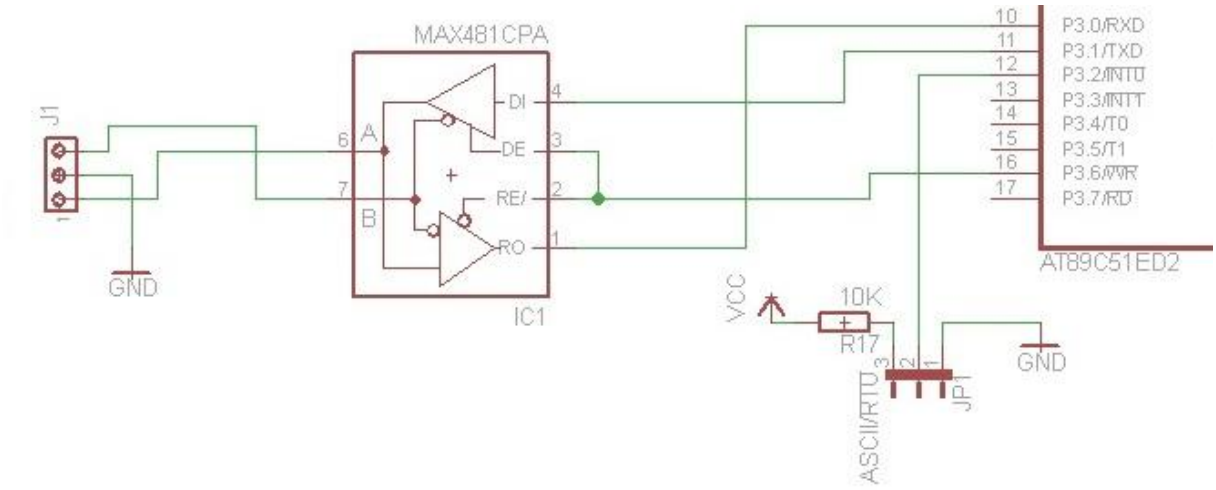
Obrázek 5.7 Schéma připojení přepínačů k procesoru

5.5 Zapojení RS485 a volba režimu přenosu dat

Jelikož vybraný procesor nemá přímo sběrnici RS485, ale pouze sběrnici UART, je nutné použít převodník. Já jsem zvolil převodník z UART na RS485 MAX481 od firmy Maxim. Tento převodník má vyvedený jeden pár vodičů pro RS485, tudíž se jedná o převodník na dvouvodičovou verzi RS485 (poloduplexní verze, lze pouze zapisovat na sběrnici, nebo pouze číst ze sběrnice, nikoliv obojí najednou). k procesoru tak musí být mimo datových vodičů také připojeno ovládání směru komunikace (čtení/zápis). Na konektoru pro vyvedení RS485 je připojena i zem (GND).

Pro zvolení režimu přenosu dat, tedy binární (RTU) či znakový (ASCII), poslouží obyčejný jumper se třemi vývody, z nichž prostřední je připojen k procesoru.

Celé schéma zapojení těchto částí viz Obrázek 5.8 (schéma vytvořeno v programu Eagle).



Obrázek 5.8 Schéma vyvedení RS485 a obvodu pro volbu režimu přenosu dat

5.6 Další pomocné periferie

Se zapojením součástek, které jsou zmíněné výše, by již mohl být obvod hotov tak, aby splňoval zadání. Já jsem se přesto rozhodl přidat ještě několik částí, které umožní lepší práci s odlaďováním programu na procesoru.

5.6.1 LED diody

Součástka, která se na podobné úlohy vždy vyplatí zahrnout do návrhu, je LED dioda, tedy svítící dioda. Může sloužit jak k indikaci chyb, tak k indikaci normálního běhu programu. Když jich je více, dají se využít jako celek. Proto jsem se rozhodl použít 8 červených diod s průměrem 3mm. Pro jejich zapojení k procesoru jsem se rozhodl použít sériovo-paralelní posuvný registr s 5 vstupy a 8 výstupy. Má ještě jeden výstup navíc, který slouží ke zřetězení několika posuvných registrů za sebe. Tento vývod však v mé aplikaci zůstane nezapojen.

Prvním vstupem je povolení dat (anglicky Output Enable, na schématu označeno písmenem G). Tento vstup je aktivní na logické úrovni 0. Jakmile se tento vstup nastaví do logické 1, výstupy registru se nastaví do třetího stavu (stav vysoké impedance).

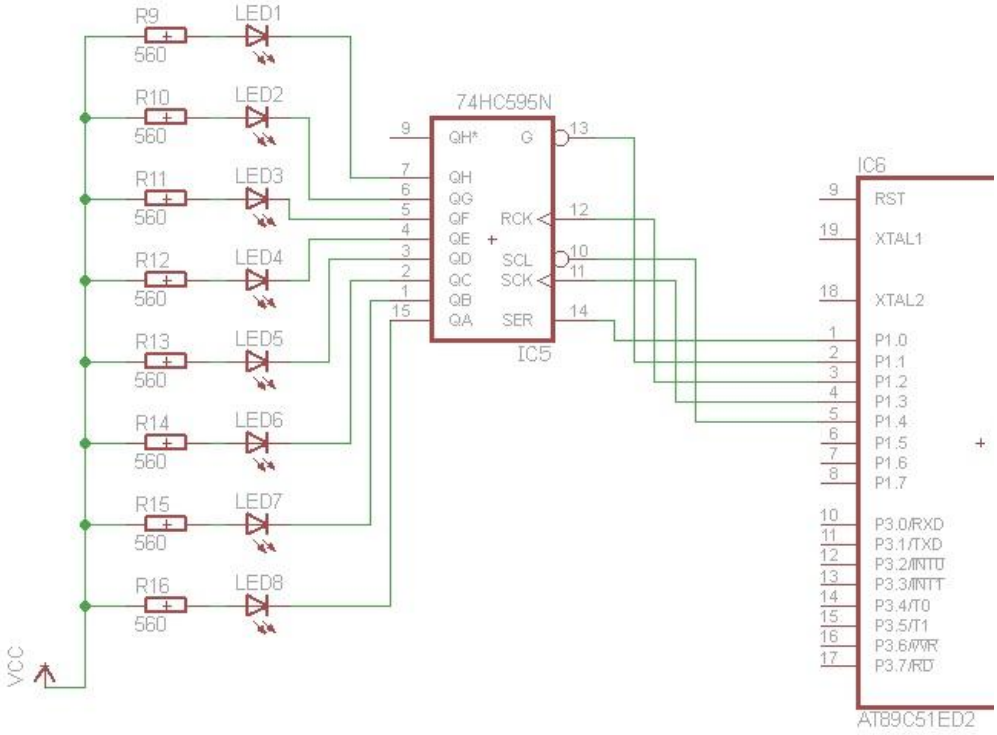
Dalším vstupem, který je také aktivní v logické 0, je sériové mazání (anglicky serial clear, na schématu označeno zkratkou SCL). Před každým nahráním nových dat do registru je vhodné stará data vymazat pomocí tohoto vstupu.

Posledním vstupem, který je aktivní na hladinu, jsou data (na schématu označeno jako SER). Zde se nastaví logická úroveň, jakou chceme zadat do registru.

Další vstup je aktivní na náběžnou hranu signálu. Jedná se o sériové hodiny (anglicky serial clock, na schématu vyznačeno zkratkou SCK). Vždy, když se na tomto vstupu objeví náběžná hrana, posuvný registr odrotuje (posune) data o jeden prvek doleva. Postupně se tak dá tedy naplnit všech 8 bitů posuvného registru.

Poslední vstup posuvného registru je na schématu označen jako RCK a při náběžné hraně zobrazí data, která jsou v posuvném registru, na všechny výstupy.

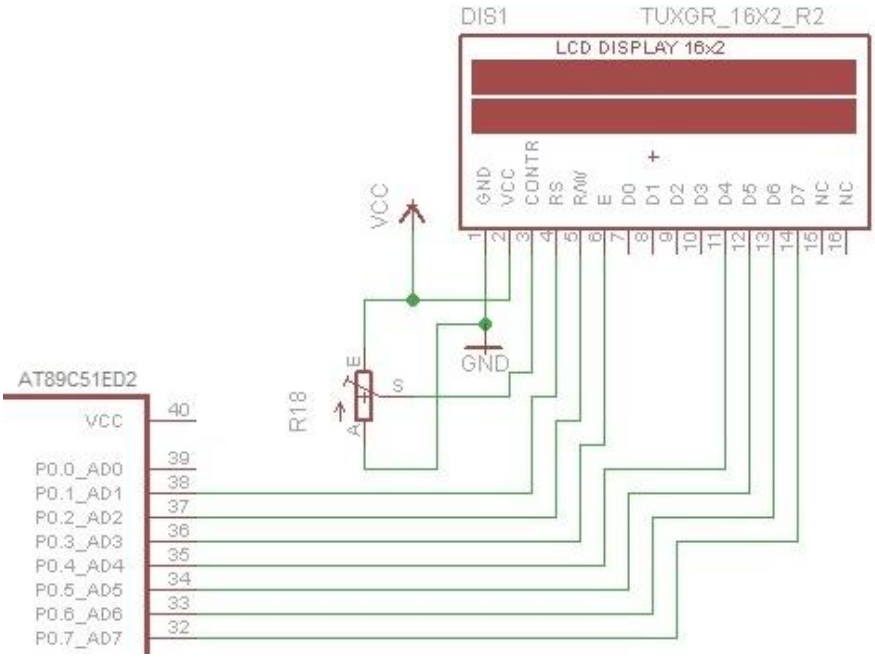
Pro zapojení osmi diod je tak třeba v případě použití posuvného registru zapojit pouze 5 vývodů na piny procesoru. Schéma zapojení viz Obrázek 5.9 (toto schéma bylo nakresleno v programu Eagle).



Obrázek 5.9 Zapojení LED přes posuvný registr k procesoru.

5.6.2 LCD displej

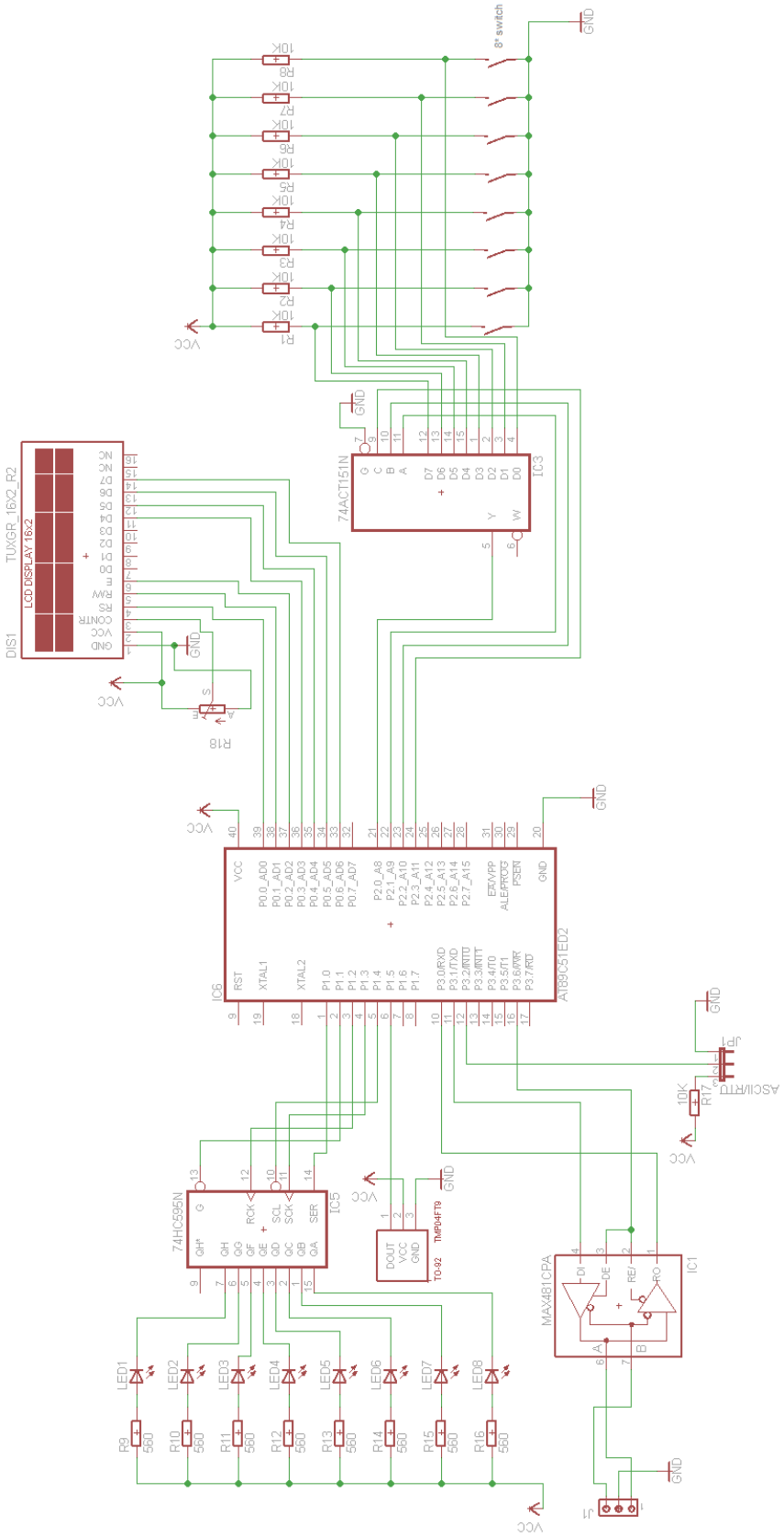
Další součástka, kterou jsem se rozhodl použít, je LCD displej 16x2 (viz Obrázek 4.3). Může se na něj například vypisovat stav spojení na sběrnici či aktuální teplota získaná z použitého teploměru. Tento displej však vyžaduje zapojení 7 vývodů k procesoru (3 řídící a 4 datové), které není vhodné dále redukovat (dal by se opět použít sériovo-paralelní posuvný registr, avšak to by ještě více zhoršilo přehlednost a obtížnost naprogramování v procesoru). Připojení LCD k procesoru viz Obrázek 5.10 (schéma bylo nakresleno v programu Eagle).



Obrázek 5.10 Schéma připojení LCD k procesoru

5.7 Výsledné schéma

Výsledné schéma (viz Obrázek 5.11) je sestaveno ze součástek zmíněných v celé této kapitole, avšak nezahrnul jsem do něj části, které jsou již hotové na univerzální vývojové desce ED2-kitu, kterou pro konečnou realizaci obvodu použiji. Jedná se tedy o připojení krystalu, resetu procesoru, napájení a obvodů pro naprogramování procesoru. (jejich zapojení na vývojové desce viz Obrázek 5.2



Obrázek 5.11 Výsledné schéma uzlu

6 Realizace schématu

Výsledné schéma bylo nutné realizovat do fyzického obvodu. Většinu součástek jsem dostal k dispozici od univerzity, integrované obvody však bylo nutné dokoupit. Jejich cena se však pohybuje pouze v řádech desítek korun.

6.1 Pájení součástek k desce

Některé součástky lze bez problémů připájet přímo k desce. Jedná se zejména o rezistory a diody. Naopak součástky jako teplotní senzor, integrované obvody či LCD displej je lepší nasazovat do patic. Proto je lepší připájet k desce pouze tyto patice. Nejen že tak vznikne možnost znovupoužití těchto součástek (snadno se vyjmou z obvodu), navíc tak nemůže dojít k jejich zničení teplem při jejich dlouhém pájení k desce.

Pro samotné pájení jsem použil hrotovou pájku, která poskytuje dostatečnou přesnost pro připájení všech použitých součástek a patic.

6.2 Propojení součástek na desce

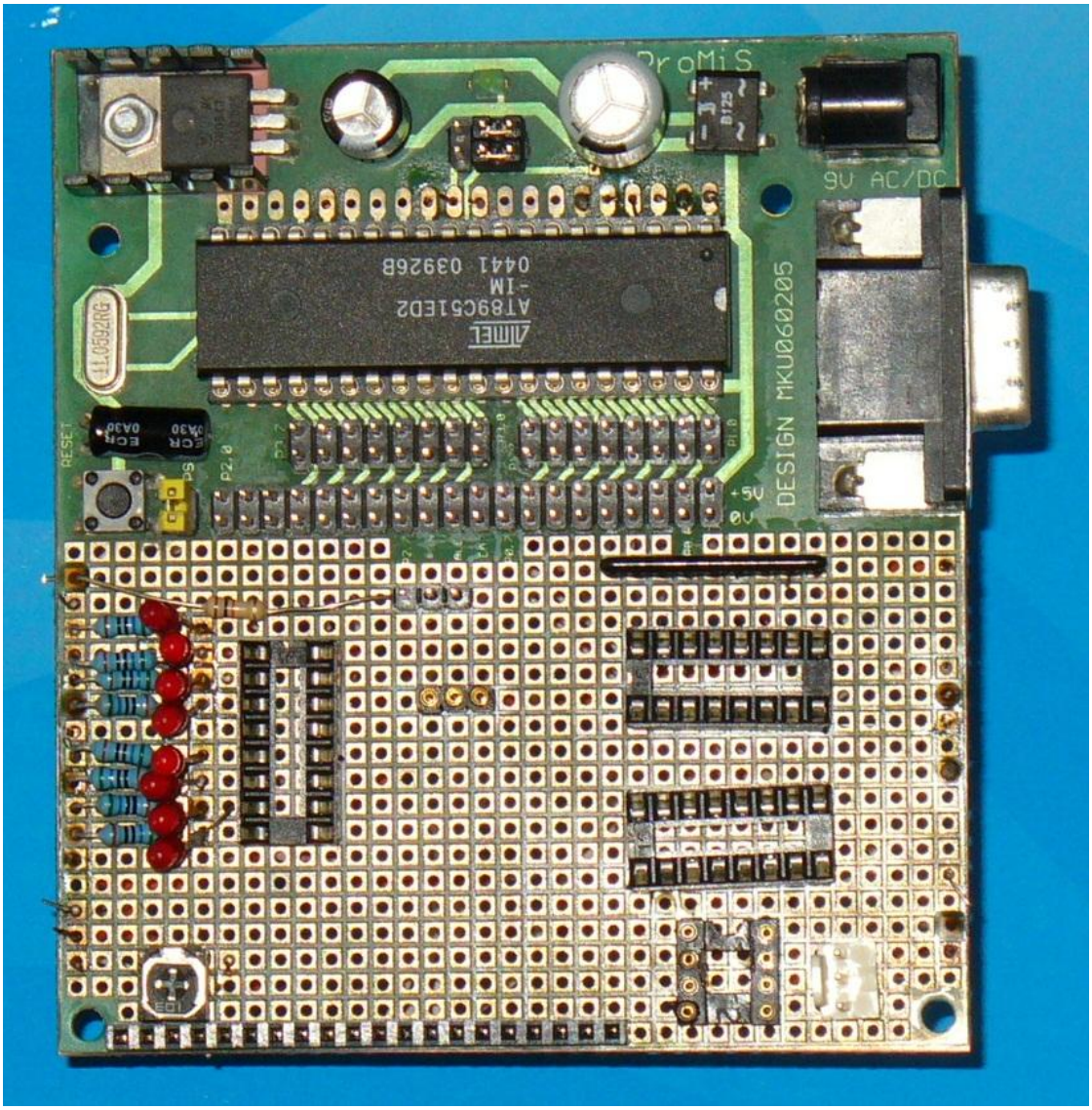
Použitá univerzální vývojová deska ED2-kit sama o sobě neposkytuje žádnou možnost propojení jednotlivých součástek, proto je k tomuto účelu nutno použít propojovacích izolovaných vodičů, jako například zvonkový drát (viz Obrázek 6.1), který byl pro tyto účely použit. Tento způsob propojení je však nepřehledný a vytváří na desce opticky nehezké klubko propojovacích vodičů. Pro větší aplikace je použití této metody propojení téměř nemožné.

Obrázek 6.1 Zvonkový drát

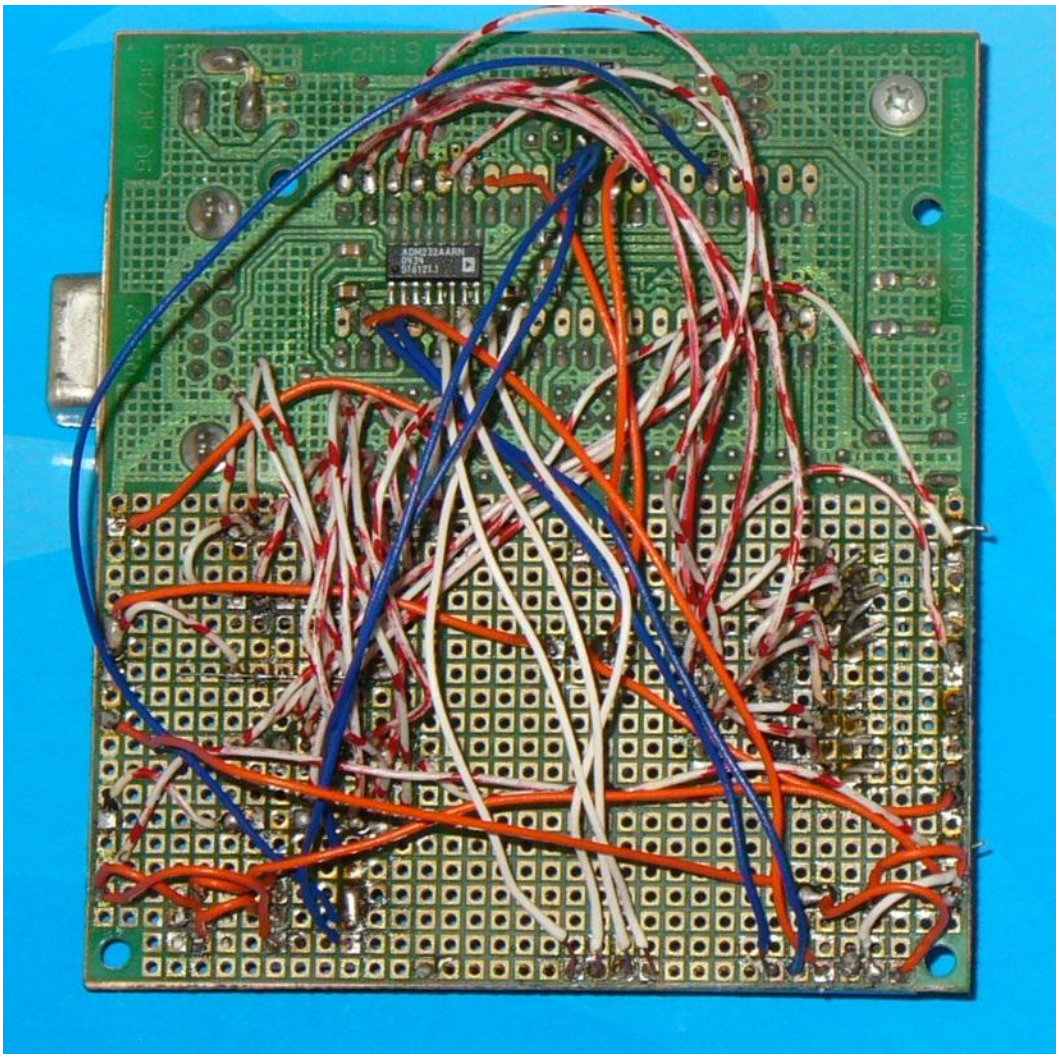
6.3 Výsledný výrobek

Výsledná podoba výrobku (pohled shora) bez součástek usazených do patic znázorňuje Obrázek 6.2. Na pohledu zdola (viz Obrázek 6.3) je vidět množství propojovacích kabelů

mezi jednotlivými součástkami a procesorem. Uzel s usazenými součástkami v paticích viz Obrázek 8.1.



Obrázek 6.2 Výsledný výrobek bez usazených součástek do patic, pohled shora



Obrázek 6.3 Výsledný výrobek bez usazených součástek do patic, pohled zdola

7 Realizace programu pro procesor

Klíčovou součástí celého přípravku je procesor. Bez nahraného funkčního programu je však i tento procesor zbytečný. Proto je třeba napsat a nahrát do procesoru program, který zařídí ve slavu:

- Načtení adresy slavu z přepínačů
- Zjištění typu přenosu (binární či znakový)
- Zjištění aktuální teploty z teplotního senzoru
- Zajištění správné komunikace po sběrnici

Případné rozšiřující funkce mohou být:

- Ovládání LED
- Ovládání LCD displeje

Také se však musí realizovat program do přípravku, který bude v roli master a bude ovládat sběrnici, na které bude připojen výše zmíněný slave. Jako master bude použit školní výukový přípravek s procesorem AT89C51CC03. Jeho úkol bude podstatně jednodušší:

- Pravidelné žádání a zpracovávání hodnoty teploty od slavu přes sběrnici
- Zobrazení získané hodnoty teploty (displej či ledbar)

Pro tento přípravek navíc také již existuje knihovna pro ovládání LCD displeje, což práci s ním ještě zjednoduší.

Jako jazyk, ve kterém se programy budou psát, jsem zvolil jazyk C, jako vývojové prostředí jsem zvolil program uVision od firmy Keil. Jedná se však o placený program a univerzita nemá k tomuto programu zakoupené licence (ani ho nikterak nepoužívá), takže jsem musel použít zkušební verzi, která má jistá omezení. Pro mé účely však bude plně stačit. Pro nahrání programu do procesoru je použit program Flip od firmy Atmel. Školní výukový přípravek se připojuje k počítači přes USB rozhranní, mnou vyrobený přípravek na desce ED2-kitu se k počítači připojuje přes sériové rozhranní RS232 (možno použít i převodník USB-RS232).

7.1 Program pro slave

Jako prvním se budu zabývat programem pro výrobek, který je předmětem této práce a který je na sběrnici v roli slavu. Celý kód obsahuje Příloha A.

7.1.1 Inicializace

Jako první se musí provést inicializace procesoru a jeho periferií. k tomu poslouží procedura nazvaná init. Dojde zde k následujícím operacím:

- Načtení hodnoty adresy slavu z přepínačů
- Inicializace používaných globálních proměnných do výchozího stavu
- Detekce režimu, v kterém bude procesor komunikovat po sběrnici (znakový či binární)
- Nastavení speciálně funkčních registrů
- Inicializace externích periferií (LCD, LED)

Kód této procedury by tedy mohl vypadat přibližně takto:

```
if(P3_2==0)rezim=re_rtu;
else rezim=re_ascii;
ixr=0;
ixt=0;
if(rezim==re_ascii){
  //IEN0 - EA=1, EC=1, ET2=0, ES=1, ET1=0, EX1=0, ET0=0, EX0=0
  IEN0=0xD0;
}else{
  //IEN0 - EA=1, EC=1, ET2=0, ES=1, ET1=1, EX1=0, ET0=0, EX0=0
  IEN0=0xD8;
}
//SMOD v PCON =1
PCON=0x80|(PCON&0x7F);
//TMOD - T1: GATE=0, C/T' = 0, M1=0, M0=1 || T0: GATE=0, C/T' = 0,
//M1=1, M0=0
TMOD = 0x12;
TH0=T0_PRELOAD;
//TCON - TF1=0, TR1=0, TF0=0, TR0=1, IE1=0, IT1=0, IE0=0, IT0=0
TCON=0x10;
//Nastavení rychlosti přes Baud Rate Generator
BRL=SET_BRL;
//BDRCON - X, X, X, BRR=1, TBCK=1, RBCK=1, SPD=0, SRC=0
BDRCON=0x1C;
//SCON - SM0=0, SM1=1, SM2=0, REN=1, TB8=0, RB8=0, TI=0, RI=0
SCON=0x50;
RS485_SMER=RS485_PRIJEM;
```

```
//cmod: CIDL=0, wdte=0, x, x, x, cps1=0, cps0=0, ecf=0
CMOD=0x00|(CMOD&0x38);
//CCON: CF=0, CR=1, X, CCF4=0, CCF3=0, CCF2=0, CCF1=0, CCF0=1
CCON=0x41|(CCON&0x20);
//ccapm2: X, ECOM3=0, CAPP3=1, CAPN3=1, MAT3=0, TOG3=0, PWM3=0,
ECCF3=1
CCAPM2=0x31|(CCAPM2&0x80);
led_inicializace();
nacti_adresu();
```

7.1.2 Odeslání bufferu na sběrnici

Sestavování odpovědi i přijímání zprávy se provádí přes proměnnou ve funkci bufferu, Jedná se o prosté pole znaků (osmibitová proměnná), které se postupně plní a když je v něm uložena celá zpráva, dojde k jejímu dalšímu zpracování. v případě odesílání zprávy je kód jednoduchý, avšak mírně se liší pro oba režimy. o odeslání zprávy se stará metoda SendBuf, jejíž vstupní parametry jsou buffer a jeho délka.

Na začátku této funkce se musí vypnout povolení přerušení a musí se přepnout směr komunikace na příjem:

```
ES=0;
RS485_SMER=RS485_VYSILAT;
```

Následně se podle režimu odvysílají data:

```
while(len--)
{
      if(rezim==re_ascii)SBUF=*bf++ | 0x80;
      else SBUF=*bf++;
      while(!TI);
      TI=0;
}
```

Nakonec již pouze stačí přepnout směr komunikace na příjem a opět povolit přerušení od sériové sběrnice:

```
RS485_SMER=RS485_PRIJEM;
ES=1;
```

7.1.3 Načtení adresy slavu

Načítání adresy se provádí přes přepínače, které jsou k procesoru zapojeny přes multiplexor. Stačí tak pouze osmkrát nastavit adresové vstupy multiplexoru a následně přečíst hodnotu na výstupu multiplexoru. Načtení prvního bitu může vypadat takto:

```
MUX_A=0;
MUX_B=0;
MUX_C=0;
if(MUX_Y)ADR_S=ADR_S+1;
```

Nakonec je třeba ještě provést validitu adresy, protože nemůže nabývat hodnoty 0 a hodnot vyšších než 247.

```
if(ADR_S==0)ADR_S=1;
else if(ADR_S>247)ADR_S=247;
```

7.1.4 Čekání 3,5 znaku

Čekání po délku doby, za kterou by se odvysílalo 3,5 znaku, slouží pouze pro binární režim. Jedná se o krátkou proceduru, která k této činnosti používá čítač/časovač 0. Vnitřní kód této procedury:

```
TH1=(word)T1_3_5>>8;
TL1=(byte)T1_3_5;
TF1=0;
TR1=1;//spuštění časovače
while(!TF1);
TF1=0;
TR1=0;//vypnutí časovače
```

7.1.5 Obsluha přerušení od sériové linky

Když sběrnice přijme jeden znak, vyvolá se přerušení. v jeho obsluze poté dojde ke zpracování tohoto znaku. Zpracování závisí na režimu. Pro binární režim je kód pro zpracování kratší:

```
RI=0;
TR1=0;
bfin[ixr++]=SBUF;
TH1=(word)T1_3_5 >> 8;
TL1=(byte)T1_3_5;
TF1=0;
TR1=1;
```

Dojde zde k pozastavení časovače T1, který je určen k indikaci konce zprávy. Následně se do bufferu vloží přijatý znak, přenastaví se registry časovače T1 a opět se spustí.

Pro znakový režim se nejdříve načte znak do proměnné. Dále se tento znak testuje, zda se nejedná o znak začínající zprávu (znak dvojtečky), či se nejedná o znak ukončující zprávu

(další řádek). Pokud to žádný z těchto znaků není, přidá se do bufferu. Pokud se jedná o znak ukončující zprávu, dojde ke zpracování této zprávy. Kód:

```
RI=0;
byteIn=SBUF&0x7F;
if(byteIn==':')
{
      ixr=0;
      fmsg=1;
}else if(fmsg) ixr++;
bfin[ixr]=byteIn;
if(fmsg && byteIn=='\n')
{ //zpracování zprávy
```

7.1.6 Zpracování zprávy a příprava odpovědi

Zpracování přijaté zprávy procesorem je opět závislé na režimu, avšak logika je stále stejná. Nejdříve se zkontroluje adresa slavu, pro který je zpráva určena. Pokud souhlasí s nastavenou adresou, zkontroluje se zabezpečení zprávy. Poté následuje pouze kontrola vyžadovaných dat (zda má slave vyžadovaná data).

Binární režim provádí toto zpracování v obsluze přerušení od časovače 1. Kód zpracování přijaté zprávy:

```
if((bfin[0]==ADR_S))
{
 if(MrtuCrc(bfin,ixr-2)==MrtuRdCrc(bfin+ixr-2))
 {
   er=0;
   switch(kod_r=bfin[1])
   {
    case FCE_RREG:
       if((reg=(bfin[2]<<8)+bfin[3])!=REG_WR) er=2;
       else if((pocet=(bfin[4]<<8)+bfin[5])!=1) er=3;
       teplota=235-((400*pca_high)/pca_low);
       registr[0]=(byte)teplota;
        if(er==0)ixt=MrtuAnsRd(ADR_S,kod_r, 1,registr,bfout);
break;
    default:
        er=1;
        break;
  }//následuje zpracování odpovědi
```

Pokud došlo k nějaké chybě, vytvoří se chybová odpověď. Nakonec se připojí zabezpečení.

```
if(er){
  ixt=MrtuAnsErr(ADR_S,kod_r|0x80,er,bfout);
}
ixt+=MrtuWrCrc(MrtuCrc(bfout,ixt),bfout+ixt);
SendBuf(bfout,ixt);
```

Podobně je tomu u režimu ASCII, který používá pouze jiné funkce knihovny protokolu MODBUS. Proto považuji za zbytečné vkládat do tohoto textu logikou shodný kód.

7.1.7 Přerušení od PCA modulu

K zjištění teploty na senzoru se využívá PCA modul procesoru, ke kterému je senzor připojen. Generuje přerušení vždy, když se změní hodnota na jeho vstupu. v obsluze tohoto přerušení stačí tedy pouze přečíst hodnotu doby, po jakou byla na vstupu daná logická úroveň:

```
CL=0;
CH=0;
CCF2=0;
CF=0;
if(P1_5==0){
  pca_high=(word)(CCAP2L+(CCAP2H<<8));
}else{
  pca_low=(word)(CCAP2L+(CCAP2H<<8));
}
```

7.2 Program pro master

Úkolem zařízení, které je na sběrnici ve funkci master, je v pravidelných intervalech odesílat na sběrnici dotaz na aktuální teplotu a po přijetí zprávy s touto informací zobrazit získanou hodnotu teploty na displeji. Celý zdrojový kód obsahuje Příloha B pro režim ASCII a Příloha C pro režim RTU.

7.2.1 Inicializace

Jako první se opět musí provést inicializace programu, kdy se nastaví globální proměnné a speciálně funkční registry procesoru na požadované hodnoty. Jedná se především o nastavení povolení přerušení, nastavení sériové sběrnice a čítačů/časovačů. Zkrácená verze procedury *init* (bez nastavování globálních proměnných):

```
//IE - EA=1, X, X, ES=1, ASCII:ET1=0 RTU:ET1=1, EX1=0, ET0=1, EX0=0
```

```
IE=0x92;
//SMOD v PCON =1
PCON=0x80|(PCON&0x7F);
//TMOD - T1: GATE=0, C/T' = 0, M1=0, M0=1 |||| T0: GATE=0, C/T' = 0,
//M1=0, M0=1
TMOD = 0x11;
//TCON - TF1=0, TR1=0, TF0=0, TR0=0, IE1=0, IT1=0, IE0=0, IT0=0
TCON=0x00;
//[RCAP2H, RCAP2L]=SET_SERIAL_SPEED;
RCAP2H=(word)SET_SERIAL_SPEED>>8;
RCAP2L=(byte)SET_SERIAL_SPEED;
//T2CON - TF2=0, EXF2=0, RCLK=1, TCLK=1, EXEN2=0, TR2=1, C/T2'=0,
//CP/RL2'=0
T2CON=0x34;
//SCON - SM0=0, SM1=1, SM2=0, REN=1, TB8=0, RB8=0, TI=0, RI=0
SCON=0x50;
```

7.2.2 Ovládání LCD displeje

Pro výukový přípravek, který používám pro realizaci zařízení master na sběrnici, již existuje volně dostupná knihovna pro ovládání LCD displeje (zdroj: [10]), kterou jsem také použil.

7.2.3 Příjem a zpracování odpovědi

Příjem a zpracování odpovědi od slavu je velice podobné jako příjem požadavku u slavu, viz kapitoly 7.1.4, 7.1.5 a 7.1.6.

Zpracování dat odpovědi je prosté, je zde kontrola detekující chybovou odpověď a v případě nechybové odpovědi je hodnota přenesená v datové části (jedná se o hodnotu teploty) vypsána na LCD displej.

8 Vyhodnocení řešení

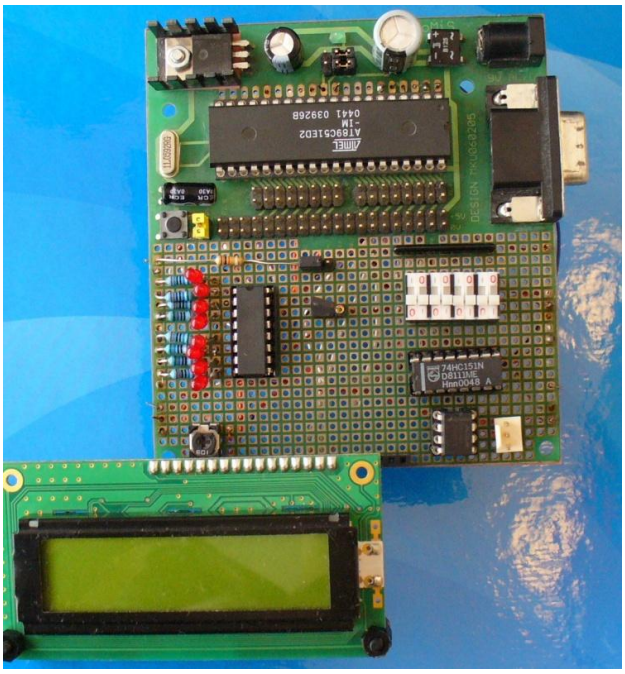
Výsledný uzel byl úspěšně realizován a otestován s pomocí školních výukových přípravků, které byly použity jako řídící prvek na sběrnici (master). Po dokončení realizace jsem také obvod důkladně proměřil, zda někde nevzniká zkrat a zda nevznikl při pájení nevodivý studený spoj. Při proměření jsem nenarazil na žádný problém.

Při realizaci vznikl malý problém, když jsem usazoval do schématu patici pro LCD displej. Předpokládal jsem, že číslování pinů na použitém LCD bude od 1 do 16. Ve skutečnosti jsou zde však nejdříve piny 15 a 16, poté následují piny 1 až 14. z tohoto důvodu nemá displej připojeny piny číslo 15 a 16, které slouží k podsvícení.

Další problém vznikl při ladění programu. Na schématu ED2-kitu od firmy Promis Liberec je zakreslený krystal o frekvenci 18,432 MHz, avšak na desce, se kterou jsem pracoval, je usazen krystal o frekvenci 11,0592 MHz. Tohoto omylu jsem si však naštěstí všiml včas.

Práce zahrnuje i upravenou knihovnu pro ovládání LCD displeje (původně [10]). Jednalo se pouze přizpůsobení knihovny jinému zapojení vstupů displeje a jiné frekvenci procesoru.

Výsledné zapojení včetně usazených součástek viz Obrázek 8.1.



Obrázek 8.1 Výsledný uzel včetně usazených součástek v paticích, pohled shora.

9 Shrnutí práce

V této práci bylo provedeno seznámení se základními technickými prostředky pro návrh uzlu na sběrnici RS485 s protokolem MODBUS. Dále byly prezentovány různé možnosti návrhu, ze kterých se následně sestavil výsledný návrh s požadovanými vlastnostmi (možnost změny adresy, možnost volby režimu protokolu MODBUS, snímání fyzikální veličiny). Po sestavení kompletního návrhu uzlu došlo k jeho realizaci a následnému úspěšnému otestování s pomocí výukových přípravků s procesory AT89C51CC03 s funkcí MODBUS master. Program pro otestování byl psán v jazyce C, ve zkušební verzi programu uVision od firmy Keil.

10 Seznam použité literatury

[1] GROSMAN, Josef. ŘPS – MODBUS [online] Fakulta mechatroniky, informatiky a mezioborových studií, Technická univerzita v Liberci, 270 stran. [cit. 2012-05-13]. URL: (www.fm.tul.cz/esf0247/download_file.php/P03_MODBUS.pdf?id=233)

[2] GROSMAN, Josef. ŘPS – Úlohy MODBUS [online] Fakulta mechatroniky, informatiky a mezioborových studií, Technická univerzita v Liberci, 177 stran. [cit. 2012-05-13]. URL: (www.fm.tul.cz/esf0247/download_file.php/ULOHY_MODBUS.pdf?id=487)

[3] Octopart. Obrázek DIL pouzdra [online]. URL: (sigma.octopart.com/472285/image/Analog-Devices-AD594AQ.gif)

[4] Talonix. Obrázek TO-92 pouzdra [online]. URL: (www.talonix.com/images/TO-92.gif)

[5] SKPang. Obrázek displeje 16x2 [online]. URL: (www.skpang.co.uk/catalog/images/modtronix/display/lcd162b-yhy.jpg)

[6] Digikey. Obrázek SMD pouzdra [online]. URL: (media.digikey.com/photos/Lite%20On%20Photos/160-16-SMD.jpg)

[7] Promis Liberec. Obrázek ED-2 kitu [online]. URL: (www.promislbc.cz/images/ed2.png)

[8] Datasheet catalog. Dokumentace k procesoru AT89C51ED2 [online]. URL: (www.datasheetcatalog.org/datasheet/atmel/doc4235.pdf)

[9] Datasheet catalog. Dokumentace k teplotnímu senzoru TMP04FT9 [online]. URL: (www.datasheetcatalog.org/datasheet/analogdevices/TMP04FS.pdf)

[10] Ceba, z hw.cz. Zdrojový kód pro obsluhu LCD displeje v jazyce C [online]. URL: (http://www.fm.tul.cz/cip/download/cviceni.c)

11 Přílohy

Příloha A Zdrojový kód programu pro vytvořený uzel RS485

```
#include "modbus.h"
#include "AT89C51ED2.h"
#include "typy.h"
#define FCE_RBIT 0x01
#define FCE_WBIT 0x05
#define FCE_RREG 0x03
#define FCE_WREG 0x06
#define FCE_WBITS 0x0F
#define FCE_WREGS 0x10
#define BIT_RD 0
#define BIT_WR 0
#define BITS_WR 0
#define REG_RD 0
#define REG_WR 0
#define REGS_WR 0
#define LB_DATA P1_0
#define LB_SCK P1_3
#define LB_SCL P1_4
#define LB_RCK P1_2
#define LB_OE P1_1
#define MUX_A P2_3
#define MUX_B P2_2
#define MUX_C P2_1
#define MUX_Y P2_0
#define RS485_SMER P3_6                    //1 - vysilani, 0 - prijem
#define RS485_PRIJEM 0
#define RS485_VYSILAT 1
#define T0_PRELOAD 253
#define SET_BRL 250      //19200Bd
#define T1_3_5 61839     //19200Bd

byte xdata bfin[512];
byte xdata bfout[512];
byte ixt, ixr;            //pocet bytu k prijmu/odeslani
word pocet;
enum {re_rtu, re_ascii} rezim;
```

```
word val;
byte byteIn;
byte bity[2];
word registr[1];
byte fmsg;
byte lrc;
byte kod_r;
byte er;
word reg;
byte ADR_S;
long pca_low, pca_high;
int teplota;
void SendBuf(byte *bf,byte len);
void cekej3_5(void);
void init(void);
void serial(void);
void nacti_adresu(void);
void main (void) {
 init();
 while (1){
       if(TF0)TF0=0;
 }
}
void init(){
  if(P3_2==0)rezim=re_rtu;
  else rezim=re_ascii;
  ixr=0;
  ixt=0;
  if(rezim==re_ascii){
  //IEN0 - EA=1, EC=1, ET2=0, ES=1, ET1=0, EX1=0, ET0=0, EX0=0
  IEN0=0xD0;
  }else{
 //IEN0 - EA=1, EC=1, ET2=0, ES=1, ET1=1, EX1=0, ET0=0, EX0=0
  IEN0=0xD8;
  }
 //SMOD v PCON =1
  PCON=0x80|(PCON&0x7F);
  //TMOD - T1: GATE=0, C/T' = 0, M1=0, M0=1 |||| T0: GATE=0, C/T' = 0,
M1=1, M0=0
       TMOD = 0x12;
 TH0=T0_PRELOAD;
```

```
  //TCON - TF1=0, TR1=0, TF0=0, TR0=1, IE1=0, IT1=0, IE0=0, IT0=0
  TCON=0x10;

 //Set Serial Speed pres Baud Rate Generator
 BRL=SET_BRL;
 //BDRCON - X, X, X, BRR=1, TBCK=1, RBCK=1, SPD=0, SRC=0
 BDRCON=0x1C;
 //SCON - SM0=0, SM1=1, SM2=0, REN=1, TB8=0, RB8=0, TI=0, RI=0
 SCON=0x50;

 RS485_SMER=RS485_PRIJEM;
 //pca
 //cmod: CIDL=0, wdte=0, x, x, x, cps1=0, cps0=0, ecf=0
 CMOD=0x00|(CMOD&0x38);
 //CCON: CF=0, CR=1, X, CCF4=0, CCF3=0, CCF2=0, CCF1=0, CCF0=1
 CCON=0x41|(CCON&0x20);
 //ccapm2: X, ECOM3=0, CAPP3=1, CAPN3=1, MAT3=0, TOG3=0, PWM3=0,
ECCF3=1
 CCAPM2=0x31|(CCAPM2&0x80);

 led_inicializace();
 nacti_adresu();
}
void SendBuf(byte *bf,byte len){
 ES=0;
 RS485_SMER=RS485_VYSILAT;
 if(rezim==re_ascii){
 while(len--)
 {
       //odvysílání jednoho znaku
       SBUF=*bf++ | 0x80;
       while(!TI);
       TI=0;
 }
 }else{
 //cyklus pro odvysílání celého bufferu
 while(len--)
 {
       //odvysílání dvou najednou znaků
       SBUF=*bf++;
       while(!TI);
```

```
      TI=0;
 }
 }
 RS485_SMER=RS485_PRIJEM;
 //zapnutí povolení přerušení od UART
 ES=1;
}
void cekej3_5(void){
 TH1=(word)T1_3_5>>8;
 TL1=(byte)T1_3_5;
 TF1=0;
 TR1=1;
 while(!TF1);
 TF1=0;
 TR1=0;
 return;
}
void nacti_adresu(void){
 byte zpomal=0;
 ADR_S=0;
 MUX_A=0;
 MUX_B=0;
 MUX_C=0;
 zpomal++;
 if(MUX_Y)ADR_S=ADR_S+1;
 MUX_A=1;
 MUX_B=0;
 MUX_C=0;
 zpomal++;
 if(MUX_Y)ADR_S=ADR_S+2;
 MUX_A=0;
 MUX_B=1;
 MUX_C=0;
 zpomal++;
 if(MUX_Y)ADR_S=ADR_S+4;
 MUX_A=1;
 MUX_B=1;
 MUX_C=0;
 zpomal++;
 if(MUX_Y)ADR_S=ADR_S+8;
```

```
 MUX_A=0;
 MUX_B=0;
 MUX_C=1;
 zpomal++;
 if(MUX_Y)ADR_S=ADR_S+16;
 MUX_A=1;
 MUX_B=0;
 MUX_C=1;
 zpomal++;
 if(MUX_Y)ADR_S=ADR_S+32;
 MUX_A=0;
 MUX_B=1;
 MUX_C=1;
 zpomal++;
 if(MUX_Y)ADR_S=ADR_S+64;
 MUX_A=1;
 MUX_B=1;
 MUX_C=1;
 zpomal++;
 if(MUX_Y)ADR_S=ADR_S+128;
 //kontrola validity adresy
 if(ADR_S==0)ADR_S=1;
 else if(ADR_S>247)ADR_S=247;
 return;
}
void serial(void) interrupt 4{
 if(rezim==re_rtu){
       RI=0;
       TR1=0;
       bfin[ixr++]=SBUF;
       TH1=(word)T1_3_5 >> 8;
       TL1=(byte)T1_3_5 ;
       TF1=0;
       TR1=1;
 }else{
       RI=0;
       byteIn=SBUF&0x7F;
       if(byteIn==':')
       {
```

```
            ixr=0;
            fmsg=1;
        }else if(fmsg) ixr++;
        bfin[ixr]=byteIn;
        if(fmsg && byteIn=='\n')
        {
            fmsg=0;
            if((MbLrc(bfin+1,ixr-4)==(lrc=MbRdByte(bfin+ixr-3)))&&
(MbRdByte(bfin+1)==ADR_S))
            {
            kod_r=MbRdByte(bfin+3);
            er=0;
            switch(kod_r)
            {
            case FCE_RREG:
            if((reg=MbRdWord(bfin+5))!=REG_RD) er=2;
            if(pca_low!=0)teplota=235-((pca_high)/(pca_low/400));
            else teplota=0;
            registr[0]=(byte)teplota;
            if(er==0)ixt=MbAnsRd(ADR_S,kod_r, 1,registr,bfout);
            break;

            default:
            er=1;
            break;
            }
            if(er!=0) ixt=MbAnsErr(ADR_S,kod_r|0x80,er,bfout);
            lrc=MbLrc(bfout+1,ixt-1);
            ixt+=MbWrByte(lrc,bfout+ixt);
            ixt+=MbWrEoT(bfout+ixt);
            SendBuf(bfout,ixt);
            }
        }
 }
}
void timer1(void) interrupt 3
{
 //jen pro RTU rezim
 TR1=0;
 TF1=0;
 ixt=0;
```

```
 if((bfin[0]==ADR_S))
 {
       if(MrtuCrc(bfin,ixr-2)==MrtuRdCrc(bfin+ixr-2))
       {
       er=0;
       switch(kod_r=bfin[1])
       {
       case FCE_RREG:
             if((reg=(bfin[2]<<8)+bfin[3])!=REG_WR) er=2;
             else if((pocet=(bfin[4]<<8)+bfin[5])!=1) er=3;
             if(pca_low!=0)
             teplota=2350-(int)(4000*pca_high/pca_low);
             else teplota=0;
             registr[0]=(word)teplota;
             if(er==0)ixt=MrtuAnsRd(ADR_S,kod_r, 1,registr,bfout);
             break;
             default:
             er=1;
             break;
             }
             P4_4=0;
             if(er)
             {
             ixt=MrtuAnsErr(ADR_S,kod_r|0x80,er,bfout);
             }
             ixt+=MrtuWrCrc(MrtuCrc(bfout,ixt),bfout+ixt);
             SendBuf(bfout,ixt);
       }
 }
 ixr=0;
 return;
}
void pca_modul(void)interrupt 6
{
 CL=0;
 CH=0;
 CCF2=0;
 CF=0;
 if(P1_5==0){
       pca_high=(long)(CCAP2L+(CCAP2H<<8));
 }else{
```

```
        pca_low=(long)(CCAP2L+(CCAP2H<<8));
  }
  return;
}
```

Příloha B Zdrojový kód programu pro výukový přípravek, režim ASCII

```
#include "modbus.h"
#include "AT89C51CC03.h"
#include "typy.h"
#include "LCD.h"
#include "klavesnice.h"
#define FCE_RBIT 0x01
#define FCE_WBIT 0x05
#define FCE_RREG 0x03
#define FCE_WREG 0x06
#define FCE_WBITS 0x0F
#define FCE_WREGS 0x10
#define BIT_RD 0
#define BIT_WR 0
#define BITS_WR 0
#define REG_RD 0
#define REG_WR 0
#define REGS_WR 0
#define INTERVAL 20
#define TIMEOUT 40
#define RS485_SMER P3_7 //1 - vysilani, 0 - prijem
#define RS485_PRIJEM 0
#define RS485_VYSILAT 1
#define MAX_SLAVE 4
#define MAX_MENU 5
#define SET_SERIAL_SPEED 65471   //19200Bd
byte xdata bfin[512];
byte xdata bfout[512];
word adr_slave;
byte cnt_ticks;
byte itx, irx;
word pocet;
word val;
byte byteIn;
byte kod_r;
enum {stKlid,stCekani,stPrijem} stav;
void SendBuf(byte *bf,byte len);
void init();
void serial(void);
```

```
void timer0(void);
void main (void) {
 init();
 TR0=1;
 while (1);
}
void init(){
  TMOD=0;
  //IE - EA=1, X, X, ES=1, ET1=0, EX1=0, ET0=1, EX0=0
  IE=0x92;
  //SMOD v PCON =1
  PCON=0x80|(PCON&0x7F);
  //TMOD - T1: GATE=0, C/T' = 0, M1=0, M0=1 |||| T0: GATE=0, C/T' = 0,
M1=0, M0=1
  TMOD = 0x11;
  //TCON - TF1=0, TR1=0, TF0=0, TR0=0, IE1=0, IT1=0, IE0=0, IT0=0
  TCON=0x00;
  //[RCAP2H, RCAP2L]=SET_SERIAL_SPEED;
  RCAP2H=(word)SET_SERIAL_SPEED>>8;
  RCAP2L=(byte)SET_SERIAL_SPEED;
  //T2CON - TF2=0, EXF2=0, RCLK=1, TCLK=1, EXEN2=0, TR2=1, C/T2'=0,
CP/RL2'=0
  T2CON=0x34;
  //SCON - SM0=0, SM1=1, SM2=0, REN=1, TB8=0, RB8=0, TI=0, RI=0
  SCON=0x50;
  RS485_SMER=RS485_PRIJEM;
  cnt_ticks=0;
  LcdInit();
  stav=stKlid;
  irx=0;
  itx=0;
  adr_slave=1;

}
void SendBuf(byte *bf,byte len){
 ES=0;
 RS485_SMER=RS485_VYSILAT;
 while(len)
 {
       SBUF=*bf | 0x80;
       while(!TI);
```

```
        TI=0;
        bf++;
        len--;
 }
 RS485_SMER=RS485_PRIJEM;
 ES=1;
}
void serial(void) interrupt 4
{
 RI=0;
 if(RS485_SMER==RS485_VYSILAT)return;
 byteIn=SBUF&0x7F;
 if(stav==stCekani && byteIn==':')
 {
        irx=0;
        stav=stPrijem;
        bfin[0]=byteIn;
 }
 else if(stav==stPrijem)
 {
        if(byteIn==':')irx=0;
        else irx++;
        bfin[irx]=byteIn;
        if(byteIn=='\n')
        {
        P4_2=1;
        P4_4=0;
        cnt_ticks=0;
        if(MbLrc(bfin+1,irx-4)==(MbRdByte(bfin+irx-3)))
        {
        kod_r=MbRdByte(bfin+3);
        switch(kod_r){
              case FCE_RREG:
              pocet=MbRdByte(bfin+5);
              if(pocet==1)
              {
              val=MbRdByte(bfin+7);
              napis_wcislo(val);
              }else napis("Spatny pocet", 12);
              break;
```

```
        case (0x80+FCE_RBIT):
        case (0x80+FCE_WBIT):
        case (0x80+FCE_RREG):
        case (0x80+FCE_WREG):
        case (0x80+FCE_WBITS):
        case (0x80+FCE_WREGS):
              val=MbRdByte(bfin+5);
              if(val==1)napis("Neznamy kod fce", 15);
              else if(val==2)napis("Neznamy objekt", 14);
              else if(val==3)napis("Chyba dat obj.", 14);
              else napis("Neznama chyba", 13);
              break;
              default:
              napis("Neznama odpoved", 15);
              break;
        }
        }else napis("spatne LRC\n", 11);
        stav=stKlid;      //klid
        }
 }
}
void timer0(void) interrupt 1
{
 cnt_ticks++;
 if(cnt_ticks==INTERVAL&&stav==stKlid)
 {
        napis("\n\n", 2);
        napis("slave", 5);
        napis_bcislo(adr_slave);
        napis(" fce", 4);
        itx=MbRd(adr_slave,FCE_RREG,REG_RD,1,bfout);
        napis("RREG", 4);
        napis("\n", 1);
        itx+=MbWrByte(MbLrc(bfout+1,itx-1),bfout+itx);
        itx+=MbWrEoT(bfout+itx);
        SendBuf(bfout,itx);
        stav=stCekani;
 }
 if(cnt_ticks==TIMEOUT&&stav==stCekani)
 {
```

```
        cnt_ticks=0;
        P4_2=!P4_2; // signalizace TimeOutu, P4_2 - LED_RED
        P4_4=1;
        stav=stKlid;
        napis("\ntimeout", 8);
  }
}
```

Příloha C Zdrojový kód programu pro výukový přípravek, režim RTU

```
#include "modbus.h"
#include "AT89C51CC03.h"
#include "typy.h"
#include "LCD.h"
#include "klavesnice.h"
#define FCE_RBIT 0x01
#define FCE_WBIT 0x05
#define FCE_RREG 0x03
#define FCE_WREG 0x06
#define FCE_WBITS 0x0F
#define FCE_WREGS 0x10
#define BIT_RD 0
#define BIT_WR 0
#define BITS_WR 0
#define REG_RD 0
#define REG_WR 0
#define REGS_WR 0
#define INTERVAL 20
#define TIMEOUT 30
#define MAX_SLAVE 4
#define MAX_MENU 5
#define RS485_SMER P3_7                    //1 - vysilani, 0 - prijem
#define RS485_PRIJEM 0
#define RS485_VYSILAT 1
#define SET_SERIAL_SPEED 65503         //19200Bd
//Fbit=18939 ->t3,5=2032
#define T1_3_5 6215                          //19200Bd
byte xdata bfin[512];
byte xdata bfout[512];
word adr_slave;
byte hodnoty[1];
byte cnt_ticks;
byte itx, irx;
word pocet;
word val;
byte byteIn;
byte kod_r;
enum {stKlid,stCekani,stPrijem} stav;
```

```
void SendBuf(byte *bf,byte len);
void init();
void cekej3_5();
void vypisbuffer(byte *bf, byte len);
void main (void) {
 init();
 ES=0;
 do{
       RI=0;
       cekej3_5();
 }while(RI);
 ES=1;
 TR0=1;
 napis("\n", 1);
 while (1);
 return;
}
void init(){
 TMOD=0;
 //IE - EA=1, X, X, ES=1, ET1=1, EX1=0, ET0=1, EX0=0
 IE=0x9A;
 //SMOD v PCON =1
 PCON=0x80|(PCON&0x7F);
 //TMOD - T1: GATE=0, C/T' = 0, M1=0, M0=1 |||| T0: GATE=0, C/T' = 0,
M1=0, M0=1
 TMOD = 0x11;
 //TCON - TF1=0, TR1=0, TF0=0, TR0=0, IE1=0, IT1=0, IE0=0, IT0=0
 TCON=0x00;
 //[RCAP2H, RCAP2L]=SET_SERIAL_SPEED;
 RCAP2H=(word)SET_SERIAL_SPEED>>8;
 RCAP2L=(byte)SET_SERIAL_SPEED;
 //T2CON - TF2=0, EXF2=0, RCLK=1, TCLK=1, EXEN2=0, TR2=1, C/T2'=0,
CP/RL2'
 T2CON=0x34;
 //SCON - SM0=0, SM1=1, SM2=0, REN=1, TB8=0, RB8=0, TI=0, RI=0
 SCON=0x50;
 RS485_SMER=RS485_PRIJEM;
 cnt_ticks=0;
 LcdInit();
 stav=stKlid;
```

```
 irx=0;
 itx=0;
 return;
}
void cekej3_5(){
 ET1=0;
 TH1=(word)T1_3_5>>8;
 TL1=(byte)T1_3_5;
 TF1=0;
 TR1=1;
 while(!TF1);
 TF1=0;
 TR1=0;
 ET1=1;
 return;
}
void SendBuf(byte *bf,byte len){
 ES=0;
 RS485_SMER=RS485_VYSILAT;
 while(len--)
 {
       SBUF=*bf++;
       while(!TI);
       TI=0;

 }
 RS485_SMER=RS485_PRIJEM;
 cekej3_5();
 ES=1;
 return;
}
void timer0(void) interrupt 1
{
 TF0=0;
 cnt_ticks++;
 if(cnt_ticks==INTERVAL && stav==stKlid)
 {
       napis("\n\n", 2);
       napis("slave", 5);
       napis_bcislo(adr_slave);
```

```
        napis(" fce", 4);
        itx=MrtuRd(adr_slave,FCE_RREG,REG_RD,1,bfout);
        napis("RREG", 4);
        napis("\n", 1);
        itx+=MrtuWrCrc(MrtuCrc(bfout,itx),bfout+itx);
        SendBuf(bfout,itx);
        stav=stCekani;
 }else if(cnt_ticks==TIMEOUT && stav==stCekani){
        P4_4=1;     //zelena
        P4_2=0;     //cervena
        napis("timeout", 7);
        cnt_ticks=0;
        stav=stKlid;
 }
}
void timer1(void) interrupt 3
{
 TR1=0;
 TF1=0;
 P4_2=1;
 if(stav==stPrijem)
 {
        if(MrtuCrc(bfin,irx-2)==MrtuRdCrc(bfin+irx-2))
        {
              kod_r=bfin[1];

              switch(kod_r){
                    case FCE_RREG:
                    val=bfin[3];
                    napis_wcislo(val);
                    break;
                    case (0x80+FCE_RBIT):
                    case (0x80+FCE_WBIT):
                    case (0x80+FCE_RREG):
                    case (0x80+FCE_WREG):
                    case (0x80+FCE_WBITS):
                    case (0x80+FCE_WREGS):
                    if(pocet==1)napis("Neznamy kod fce", 15);
                    else if(pocet==2)napis("Neznamy objekt", 14);
                    else if(pocet==3)napis("Chyba dat obj.", 14);
                    else napis("Neznama chyba", 13);
```

```
                    break;
                    default:
                    napis("Neznama odpoved", 16);
                    break;

             }
       }else napis("chyba crc", 10);
       stav=stKlid;
 }
 irx=0;
}
void serial(void) interrupt 4{
 byteIn=SBUF;
 RI=0;
 P4_4=0;
 switch(stav)
 {
       case stCekani:
             cnt_ticks=0;
             irx=0;
             bfin[irx]=byteIn;
             irx++;
             stav=stPrijem;
             TH1=(word)T1_3_5>>8;
             TL1=(byte)T1_3_5;
             TF1=0;
             TR1=1;
             break;
       case stPrijem:
             bfin[irx]=byteIn;
             irx++;
             TH1=(word)T1_3_5>>8;
             TL1=(byte)T1_3_5;
             TF1=0;
             break;
       default:
             break;
 }
}
```