České Vysoké Učení Technické v Praze
Fakulta Elektrotechnická

BAKALÁŘSKÁ PRÁCE

Praha, 2007

Štěpán Čejka

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA MĚŘENÍ

Dálkový odečet elektroměru prostřednictvím komunikačního kanálu standardu Meter Bus

BAKALÁŘSKÁ PRÁCE

Praha, 2007

vypracoval: Štěpán Čejka
vedoucí práce: Ing. Jiří Novák, Ph.D.

Prohlášení

Prohlašuji, že jsem zadanou bakalářskou práci zpracoval sám s přispěním vedoucího práce a konzultanta a používal jsem pouze literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé bakalářskou práce nebo její části se souhlasem katedry.

V Praze dne _____ _____

podpis

České vysoké učení technické v Praze
Katedra **měření K13138**

Fakulta elektrotechnická
Školní rok 2005/2006

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student **Štěpán Čejka**

Obor **Kybernetika a měření**

Název tématu: Dálkový odečet elektroměru prostřednictvím komunikačního kanálu standardu Meter Bus

Zásady pro vypracování:

Navrhněte a realizujte modul dálkového odečtu elektroměru s rozhraním M-bus. Zvolte vhodný mikroprocesor vybavený dvěma rozhraními UART a dalšími periferiemi, nezbytnými pro realizaci modulu – preferujte jednočipové řešení. Vzhledem k možnostem napájení ze sběrnice zohledněte příkon modulu. Programové vybavení pište modulárně v jazyce C tak, aby zejména implementace knihovny pro M-bus komunikaci byla opakovaně využitelná.

Anotace v českém jazyce:

Tato práce řeší využití standardu M-bus pro napájení a komunikaci s modulem elektroměru, který zajišťuje vyčtení údajů o spotřebě. Modul je postaven s mikroprocesorem TI MSP430F1610 a komunikačním převodníkem TSS721A. Jako programovací jazyk je použit jazyk C. Výsledkem je vytvoření knihovny zajišťující linkovou vrstvu standardu M-BUS.

Annotation in English:

This work discusses using M-bus standard for supply and communication with electrometer module which assures reading of data about consumption. The module is built with microprocessor TI MSP430F1610 and communication converter TSS721A. Programming language C is used. The result is creature of library serving as link layer for M-BUS standard.

Obsah

1 Úvod

Vzhledem ke snaze vytvářet samoobslužné systémy a zařízení je vyčítání elektroměrů jeden z oborů, kam ještě moc rozvoj technologie nepostoupil. Proto se objevují snahy, jak efektivně, spolehlivě a levně řešit tento problém. Nově vyráběné elektroměry jsou již vybaveny mikroprocesory a obsahují jednoduché komunikační rozhraní pro získání údajů o spotřebě, které omezuje chybu lidského faktoru při přepisu spotřeby, ale neřeší problém v tom smyslu, že do dané lokality musí fyzicky přijít pracovník, otevřít elektroměrovou skříň a fyzicky se dostat k elektroměru. Aby tato starost odpadla, je vhodné vytvořit modul pro elektroměr, který by na žádost řídící stanice „přečetl" údaje o spotřebě a odeslal je řídící stanici pro následné zpracování. Pro tento úkol lze zvolit z velkého množství standardních komunikačních rozhraních a uspořádání systému. S ohledem na předpokládané požadavky kdy vyčítání elektroměrů se provádí jednou ročně, objemu přenášených dat, požadavkům na spolehlivost a předpokládaným vzdálenostem je výběr mnohem více upřesněn, přesto je zde stále mnoho možností realizace. Pro tuto bakalářskou práci je v zadání specifikována sběrnice M-bus, která splňuje prakticky veškeré výše uvedené výchozí předpoklady. Zásadním důvodem pro implementaci sběrnice M-bus je možnost napájení modulu vyčtení elektroměru z komunikační sběrnice a tato práce by měla ověřit praktickou využitelnost této vlastnosti ve spojení s elektroměrem ZE310.

Sběrnice M-bus byla vyvinuta jako nový evropský standard pro vzdálené čtení měřičů tepla a je využitelná pro většinu typů měřičů spotřeby. Aktuální dokumentace je ve verzi M-bus 4.8 z roku 1997 a pravděpodobně se jedná o finální verzi komunikačního standardu. Hlavní výhody spočívají v možnostech vyčíst data elektronicky, nutnost přidání pouze jednoho páru kabelu, individuální adresaci každé jednotky, galvanické oddělení rozhraní, možnost síťového rozšíření, nízké náklady a přijatelnou přenosovou rychlost. Přenosová rychlost je standardem definována jako rychlost mezi 300-9600 Bd. Dle výrobce elektroměru je maximální množství dat při vyčtení elektroměru ZE 310 do 2 kB

dat. Při maximálním možném množství připojených jednotek v jednom segmentu (250 účastnických jednotek slave), využití dlouhých rámců (až 252 datových bajtů) a maximální přenosové rychlosti 9600 Bd dojdeme k závěru, že jedna jednotka může být při cyklickém vyčítání dotazována jednou za 61 sekund s žádostí o data a s kompletním datovým přenosem vyčtených dat.

Výpočet opakování vyčtení:

1xkrátký rámec se žádostí o data = 6 bajtů

9xdlouhý rámec s daty, nevyužit kompletně ≈ 2330 bajtů

maximum jednotek na segmentu = 250

maximální rychlost = 9600 Bd

minimální rychlost = 300 Bd

$$\frac{(2300+6)*250}{9600} = 60{,}833 \text{ sekundy}$$

$$\frac{(2300+6)*250}{300} = 1946{,}66 \text{ sekundy} \approx 32 \text{ minut } 27 \text{ sekund}$$

Z tohoto pohledu pro nás není přenosová rychlost omezujícím činitelem a proto může být použita i řádově nižší přenosová rychlost. Jako omezující činitel je pro nás vlastní elektroměr, který se vyrábí a má již předem definovaný komunikační protokol a požadavky pro vyčtení. Tato skutečnost se projevuje jak v hardwarovém, tak i softwarovém návrhu.

Celá stať je rozdělena na dva logické celky. První část se věnuje hardwarovému návrhu desky pro elektroměr s odkazy na typ elektroměru, význam použitých součástek vzhledem ke kompatibilitě k elektroměru a sběrnicí M-bus. Druhá část se bude zabývat vlastním programovým řešením úlohy s ohledem na rozdělení do logických celků vyčtení elektroměru a komunikace po sběrnici.

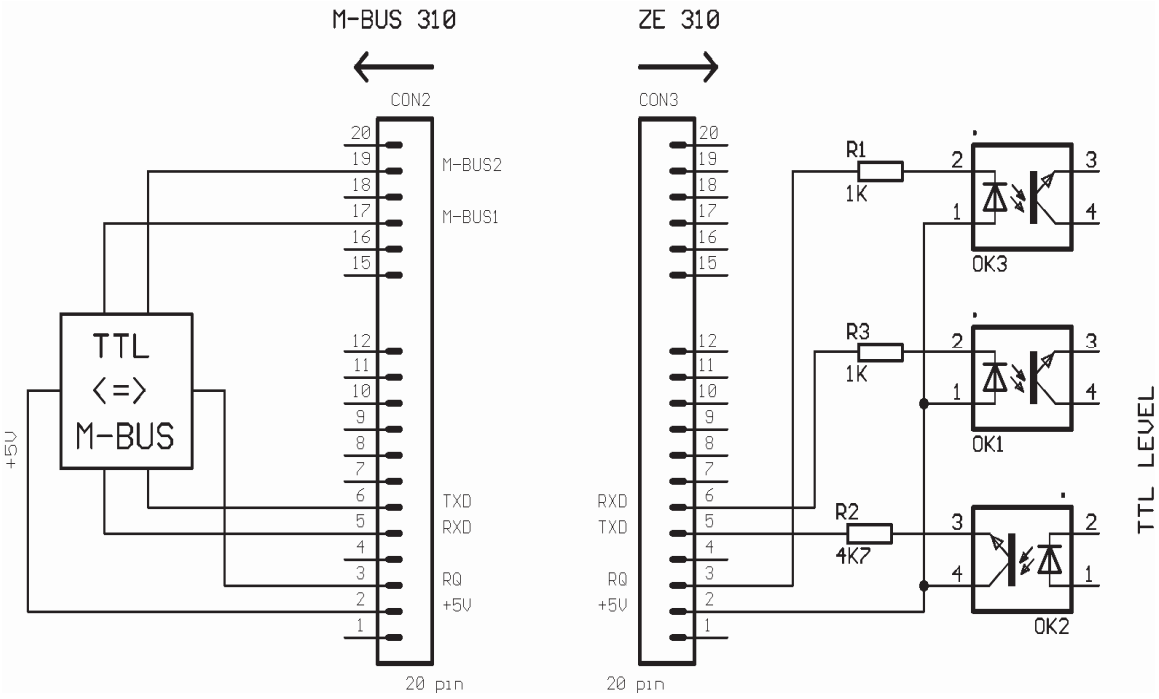
2 Hardwarový návrh

2.1 Návrh použitých prvků

Hardwarový návrh vycházel z datového listu elektroměru, poskytnutých rozměrů rozšiřujícího portu a požadavku vytvoření desky plošných spojů v návrhovém prostředí EAGLE. Vzhledem k omezení sběrnice M-bus pro napájení jednotlivých slave-jednotek na 600 μA a vzhledem k zadání bakalářské práce, která požaduje mikroprocesor vybavený dvěma porty UART jsem zvolil jako základní řídící jednotku procesor firmy Texas Instruments řady MSP430 v konkrétní konfiguraci MSP430F1610. Hlavní požadavky na tento procesor byla nízká spotřeba, která je dle katalogového listu při napětí 3 volty nominálně 500 μA. Další splněnou podmínkou jsou dva porty UART, kdy jeden bude využíván pro komunikaci se sběrnicí M-bus a druhý je použit na vyčítání dat z elektroměru. Dále je procesor osazen Flash pamětí pro snadné programování a pamětí 5 kB RAM pro ukládání výsledků vyčítání elektroměru a komunikace po sběrnici M-bus. Jako programovací rozhraní je použit standard JTAG a proto je na desce i unifikovaný konektor 2x7 pinů. Jako nezbytný mezičlánek je použit čip Texas Instruments TSS721A, který zajišťuje komunikaci mezi procesorem a sběrnicí a obstarává napájení mikroprocesoru. Tento čip byl vyvinut ve spolupráci Univerzity Paderborn a společností Texas Instruments Deutschland GmbH a je označován jako transceiver (transmitter a receiver). Použití tohoto čipu vede ke snížení počtu použitých součástek u slave jednotky, zjednodušuje se návrh této jednotky a dle vývojářů rozhraní M-bus i ke snížení ceny slave jednotky).

Jako hlavní problém se projevilo omezení sběrnice M-bus ohledně napájení jednotek, kdy v klidové úrovní se jedná konstantní odběr 1,5 mA. Výše jsem již uvedl podmínku pro napájení procesoru maximem 0,6 mA. Zdálo by se, že pro vlastní běh jednotky je tedy podmínka napájení splněna. Bohužel z pochopitelných důvodů je komunikační rozhraní s elektroměrem galvanicky oddělené a to dramaticky zvyšuje nároky na spotřebu. Tyto nároky na spotřebu se budu snažit pokrýt využitím kondenzátoru C_{STC} ze zapojení obvodu TSS721A.

Využití tohoto kondenzátoru je v souladu s katalogovým listem součástky TSS721A a dokumentace standardu M-bus.



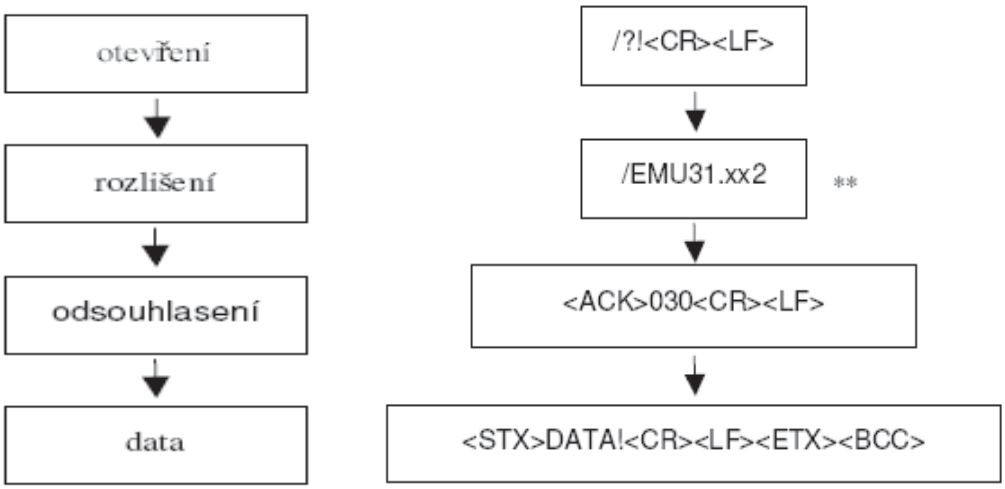
Obr. 01 – Vnitřní schéma galvanického oddělení elektroměru

2.2 Návrh kondenzátoru C_{STC}

Za předpokladu 5-ti voltového napájení a dodržení podmínek pro vyčtení elektroměru (alespoň 1,6 vteřiny před zahájením komunikace musí být signál RQ aktivní, aktivní musí být dále po celou dobu komunikace), informací výrobce elektroměru o maximální velikosti vyčtených dat 2 kB, (v praxi ověřená velikost okolo 1100 B) a z hodnoty odporů můžeme snadno odhadnout velikost náboje potřebného pro vyčtení elektroměru.

Komunikace je poloduplexní postup je takový, že pro otevření je použita rychlost 1200 Bd, následuje odpověď elektroměru s informací o maximální podporované rychlosti. Konkrétní maximální hodnota pro mě poskytnutý elektroměr je 4800 Bd. Na potvrzení příjmu identifikace elektroměru a nastavení vyšší komunikační rychlosti vysílá mikroprocesor potvrzení vyšší komunikační rychlosti (<ACK>040) a ukončovací sekvenci <CR><LF>.

Následuje zaslání dat z elektroměru nastavenou rychlostí. V našem případě bude elektroměr vysílat rychlostí 4800 Bd.



Obr. 02 – příklad vyčtení elektroměru

Pro jednoduchost výpočtu budeme uvažovat uběhlou dobu komunikace od vyčítajícího zařízení směrem k elektroměru při rychlosti 1200 za 0,0916 sekundy. Rozlišení je provedeno rychlostí 1200 Bd a objem přenášených dat pro modelovou řadu ZE310, potažmo i všech elektroměrů ZPA, je 20 bytů. Potřebná doba přenosu dat činí 0,166 sekundy. Přenos dat z elektroměru probíhá již na rychlosti 4800 Bd a bude mít předpokládanou dobu přenosu 4,166 sekundy.

Výpočet potřebné energie pro vyčtení dat z elektroměru:

U_N= 5V

Výstup RQ:

R_{RQ}= 1 kΩ

U_{LED}= 1,5 V (odhadnutá hodnota napětí v propustném směru)

$$I = \frac{U_N - U_{LED}}{R_{RQ}} = \frac{5-1,5}{1000} = 3,5mA$$

Výstup TX:

R_{TX}= 1 kΩ

U_{LED}= 1,5 V (odhadnutá hodnota napětí v propustném směru)

$$I = \frac{U_N - U_{LED}}{R_{TX}} = \frac{5 - 1{,}5}{1000} = 3{,}5\ mA$$

Vstup RX:

R_{RX}= 4,7 kΩ

U_{LED}= 0,5 V (odhadnutá hodnota napětí v propustném směru)

$$I = \frac{U_N - U_{LED}}{R_{RX}} = \frac{5 - 0{,}5}{4700} = 0{,}96\ mA$$

Všechny výše uvedené proudy byly před fyzickou realizací prakticky ověřeny.

Z výpočtů velikosti jednotlivých proudů zřejmé, že maximální možný odběr 0,6 mA nemůže být dodržen. Proto je nutné využití kondenzátoru C_{STC}. Z výše uvedených výpočtů a předpokladů bude vycházet návrh velikosti tohoto kondenzátoru.

Výpočet velikosti náboje potřebného pro vyčtění elektroměru

$t_{RQ} \cong 6{,}5\ s \quad \Rightarrow Q_{RQ} = 22{,}75\ mC$

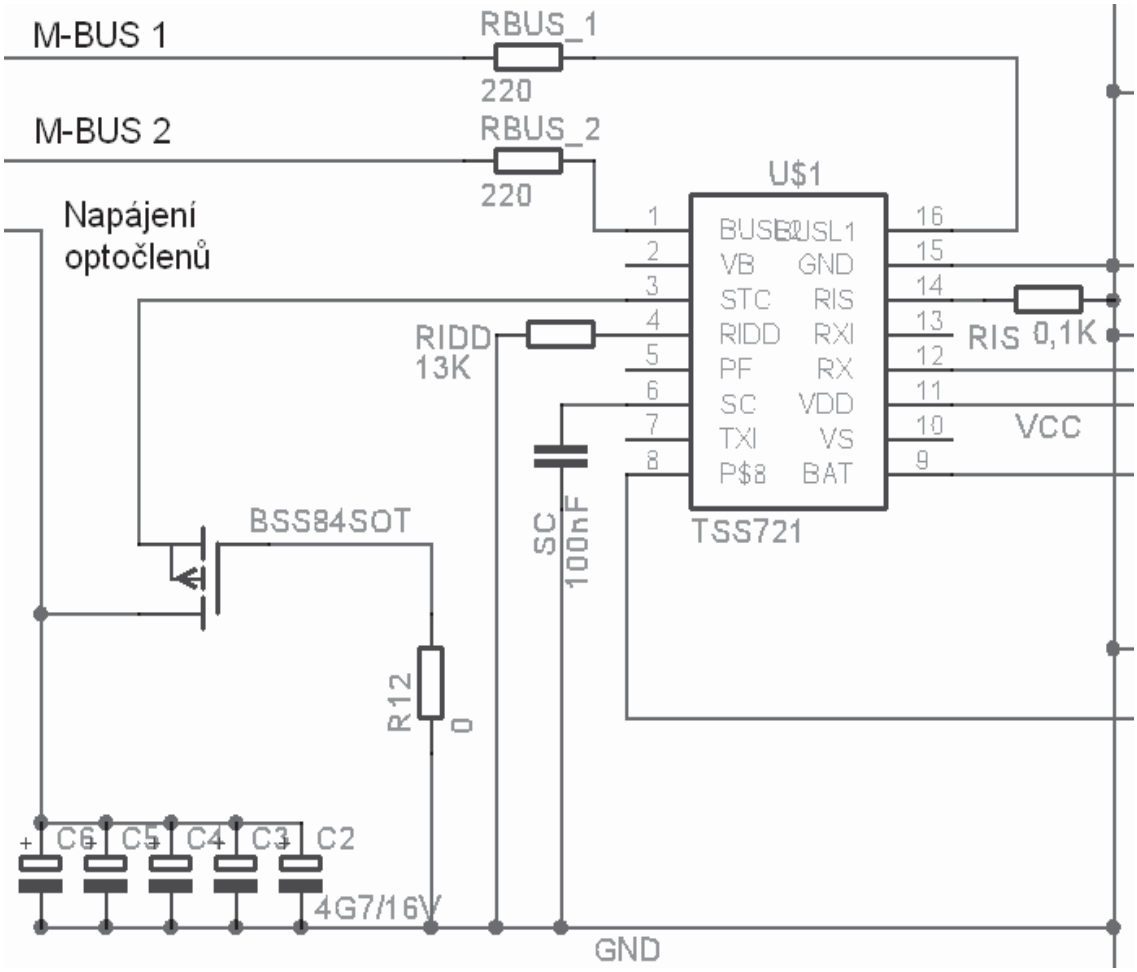
$t_{TX} \cong 0{,}1\ s \quad \Rightarrow Q_{TX} = 0{,}35\ mC$ Q_C= 27,276 mC

$t_{RX} \cong 4{,}35\ s \quad \Rightarrow Q_{RX} = 4{,}17\ mC$

Velikost vypočteného náboje je potřebná pouze pro vlastní vyčtení z elektroměru, další podmínkou je, že napětí na tomto kondenzátoru nesmí poklesnout pod 4,5 V, jinak dojde k odpojení napájecího napětí procesoru. Maximální napětí na kondenzátoru nepřekročí 7V. Z těchto podmínek jsem kondenzátor dimenzoval na velikost C_{STC}= 23,5 mF.

2.3 Zapojení obvodu TSS721A

Při vlastním návrhu schématu zapojení jsem vycházel z doporučeného katalogového zapojení pro obvod TSS721A.



Obr. 03 – Zapojení IO TSS721A

Kondenzátory C2-C6 reprezentují kondenzátor C_{STC} a i ve fyzické realizaci se jedná o 5 samostatných kondenzátorů, každý o kapacitě 4,7 mF. Tato hodnota překračuje zhruba 50x maximální doporučenou hodnotu. Odpory R BUS 1 a R BUS 2 jsou ochranné odpory. Odpor R_{RIDD} slouží k regulaci odebíraného proudu a je jej potřeba nastavit dle skutečné velikosti C_{STC}. Kondenzátor C_{SC} slouží k definování napěťové reference pro vnitřní komparátor zajišťující příjem dat po M-busu. Hodnota tohoto kondenzátoru je dle katalogového listu součástky TSS721A. Jako poslední je použit odpor R_{RIS},

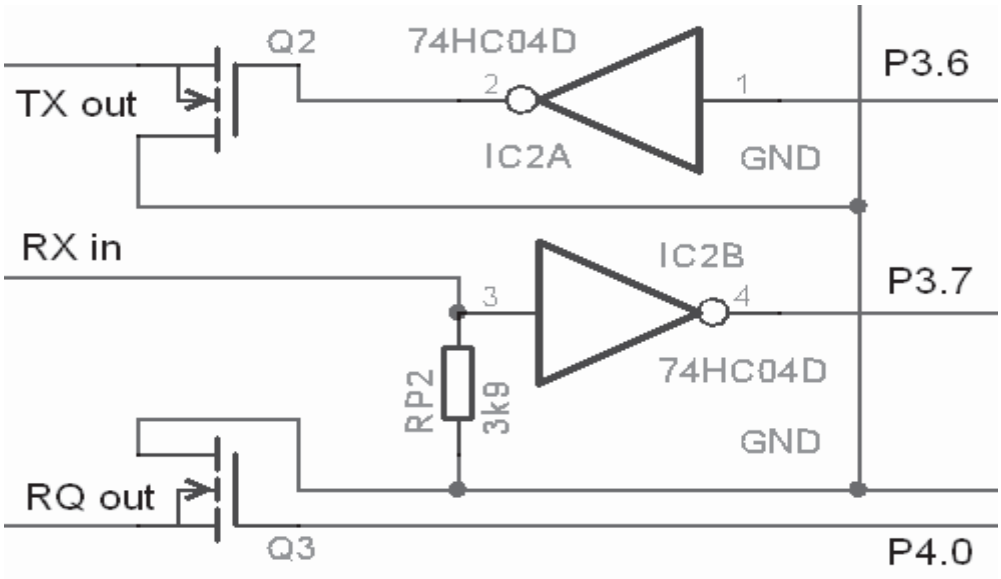
kterým se nastavuje velikost odebíraného proudu při vysílání logické 0. Odpor této velikosti způsobuje vyslání rozdílového proudu o velikosti cca 19 mA.

2.4 Napojení komunikačních rozhraní

Jako další část schématu je blok procesoru s připojením na komunikační rozhraní elektroměru a na komunikační rozhraní od sběrnice M-BUS. U připojení ke komunikační lince od sběrnice M-BUS se jedná o pouhé propojení. Pro vývojové účely jsou pouze vloženy nulové odpory, které mají zajišťovat přerušení komunikační linky a možnost napojení sériové linky od jiného komunikačního rozhraní. Já jsem tohoto využíval pro účely testování procesoru a jeho komunikačních schopností.

Připojení ke komunikačnímu rozhraní elektroměru byl komplikovanější problém, vzhledem k jinému napájecímu napětí. Ve větvi zajišťující vysílání je proto použit spínací tranzistor typu N-MOS BS108. Vzhledem k inverzní logice vysílání kdy logická jednička na gate tranzistoru znamená otevření tranzistoru a tím pádem i vyslání logické nuly, je nutno do vysílací cesty ještě vložit invertor. Já jsem použil standardní integrovaný obvod se šesticí invertorů použitelný při třívoltovém napájecím napětí 74HC04. Z praktické realizace vyplynula nutnost ošetření nezapojených vstupů, neboť jinak se jednotlivá hradla rozkmitají a způsobí tím prudký růst proudové spotřeby, která v konečném důsledku může vést k vypnutí napájecího napětí logických obvodů. U přijímací linky jsem musel řešit podobný problém. V klidovém stavu, který odpovídá logické jedničce na lince, je tranzistor v optronu OK2 uvnitř elektroměru zavřený a proto na výstupu je napěťová úroveň 0V. Při přímém zapojení do procesoru by tudíž klidový stav na lince odpovídal logické nule procesoru. Z tohoto důvodu je patrná nutnost použít opět invertor. Dalším nutným prvkem je přizpůsobovací odpor, opět vzhledem k jinému napájecímu napětí optronů. Velikost tohoto odporu byla navrhnuta pro správnou velikost vstupního napětí, tzn. že úroveň vstupního napětí invertoru je pak V_{IHmax}=3,17 V a V_{IHmin}=1,81 V. Díky tomu můžeme mluvit o napěťovém přizpůsobení vstupního napětí pro obvod CMOS

s napájecím napětím V_{DD}= 3,3 V. Výstup tohoto invertoru je již přímo zapojen na pin P3.7, který má funkci RX pinu UART1.

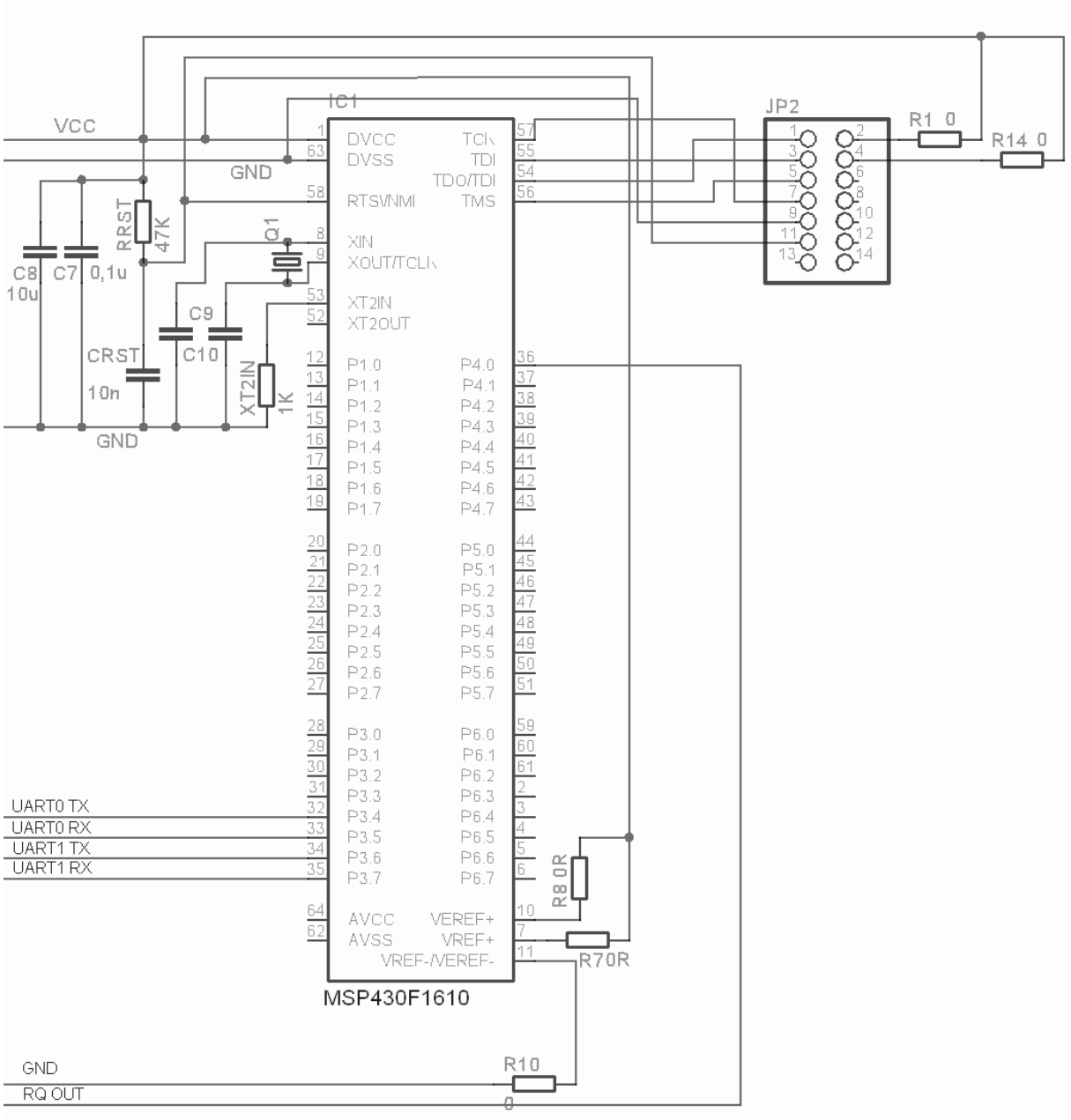


Obr. 04 – zapojení rozhraní elektroměru

2.5 Zapojení procesoru

Vlastní zapojení procesoru je realizováno podle doporučeného zapojení pro procesor řady MSP430 a 4-drátové JTAG rozhraní. V tomto zapojení je procesor již plně připraven k programování a jakékoliv další činnosti. Kapacity C8 a C7 jsou pro filtrování napájecího napětí. Zapojení R_{RST} a C_{RST} je resetovací obvod pro start procesoru. Krystal Q1 a kapacity C9 a C10 tvoří generátor hodinového signálu. Krystal Q1 je standardní hodinový krystal o frekvenci f_{clk}=32768 Hz. Kapacity C9=C10=12pF. XT2IN je standardní odpor pro ukončení nepoužívaného hodinového vstupu. Konektor JP2 reprezentuje připojovací rozhranní JTAG, se čtveřicí signálů, napájecím napětím, zemním vodičem a resetovacím výstupem. Odpory R1 a R14 slouží k selekci napájecího napětí. Při připojení odporu R1 je celý systém napájen z programovací jednotky JTAG. Připojení odporu R14 je programovaná jednotka napájena z vlastního zdroje napětí. Připojení obou odporů najednou není možné. Dále při používání napájení z programátoru může docházet ke kolizím napěťových úrovní.

V případě mého projektu se konkrétně jednalo o zvýšený odběr proudu, který vedl k odpojení napájecího napětí ze sběrnice M-BUS.



Obr. 05 – Schéma zapojení mikroprocesoru

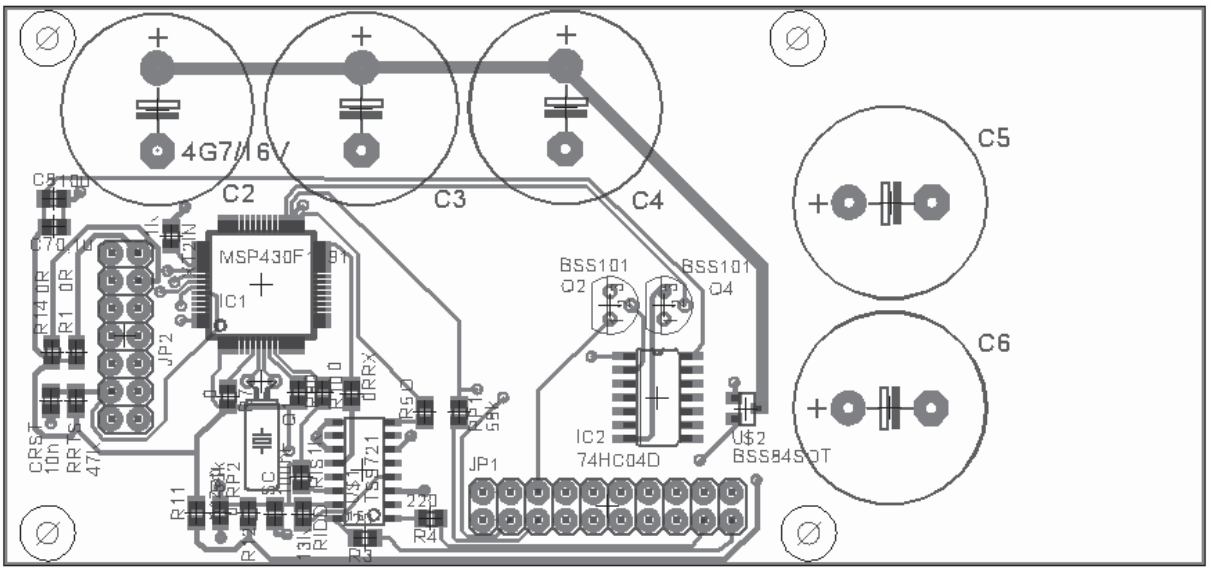
2.6 Praktická realizace

Při vlastní prvotní realizaci desky plošných spojů došlo k několika chybám, které je potřeba pro průmyslovou realizaci opravit. Jedna z chyb bylo otočení pohledu ze strany spojů a ze strany součástek. Kvůli tomu došlo

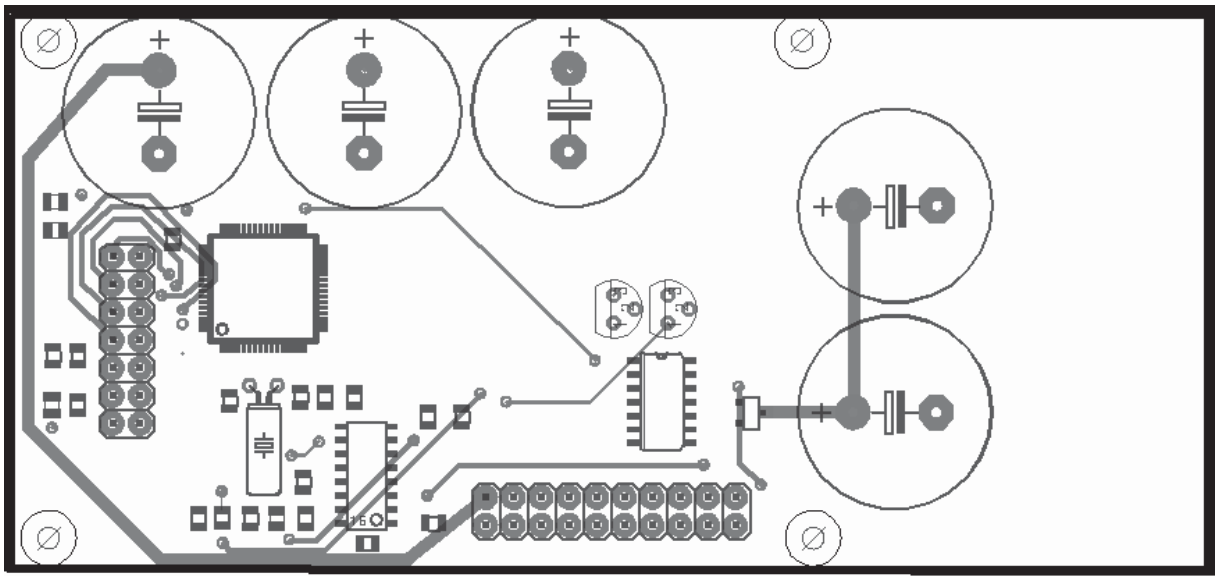
k rozdílu v umístění děr pro montážní sloupky. Druhým problémem který jsem v prvotním návrhu opomněl bylo již zmiňované neošetření nazapojených vstupů. Na prototypu desky jsem tedy tuto chybu musel opravit použitím samopájitelného drátu a propojením vstupních pinů se zemním vodičem. Druhou chybou kterou jsem musel opravit bylo připojení vstupu UART1 RX. V původním návrhu jsem měl pouze odporový dělič, který přizpůsoboval vstupní napěťové úrovně. Výše jsem již uvedl správné a opravené schéma vstupní části. Další věcí která souvisí s návrhem HW je okamžik po startu procesoru, kdy ještě nedojde k inicializaci jednotlivých portů. V tento okamžik je jako standardní logická úroveň před softwarovou inicializací logická 0. To neodpovídá běžnému klidovému stavu na vysílací části sériové linky, která je definována do logické jedničky. To způsobí otevření tranzistoru Q2 a tím pádem i nežádoucí odběr proudu z kondenzátoru C_{STC}. V konečném důsledku by tento stav mohl vést ke špatné inicializaci jednotky, která by se tím dostala do trvale nefunkčního stavu. V případě laboratorní práce s jednotkou odečtu elektroměru se jednalo skutečně o stav těsně po spuštění jednotky. Pokles napětí během inicializace jsem změřil na 1V. Tato hodnota je 40% možného poklesu napětí. Pro průmyslovou realizaci by bylo vhodné doplnit schéma napájení optronů vložením spínacího tranzistoru, např. typu N-MOS BS108, mezi kondenzátor C_{STC} a výstupní napájecí pin. Tento tranzistor by měl zajistit okamžik po připojení napájecího napětí V_{CC}= 3,3V odpojení napájecího pinu optronů do okamžiku inicializace sériové linky. Tento tranzistor nebyl prozatím realizován a je to otázka definitivní průmyslové realizace.

2.7 Deska plošných spojů

Hardwarový návrh ukončím uvedením obrazu desky plošných spojů. Tato deska je prototyp pro vyzkoušení možné realizace a implementace rozhraní M-BUS. Jedná se o dvouvrstvou desku, s osazením převážně součástkami typu SMD velikosti 0805. Uvedená deska odpovídá prototypu před zjištěnými změnami. Pro přehlednost je na spodní straně desky plošných spojů nezobrazena zemnící plocha, která vyplňuje veškerý zbylý prostor spodní strany.



Obr. 06 – Horní strana desky plošných spojů



Obr. 07 – Spodní strana desky plošných spojů.

3 Softwarový návrh

3.1 Princip návrhu

Návrh softwaru se rozdělil do 2 samostatných částí. První realizovanou částí byla funkce pro vyčtení elektroměru. Druhou část softwaru tvoří samostatná knihovna pro práci se sběrnicí M-BUS. Tato knihovna by měla být nezávislá na použitém rozhraní a procesoru a opětovně využitelná v jiných programech. Podobným způsobem by šla realizovat i funkce pro vyčtení elektroměru, ale vzhledem k požadavku prostého vyčtení jsem vytvořil pouze tuto jednoduchou funkci.

3.2 Funkce vyčtení elektroměru

Funkce vyčtení elektroměru je vytvořena jako bezparametrová funkce, která naplní 2 globální proměnné programu. Tato funkce z větší části pracuje pod přerušením a celková doba vyčtení elektroměru je okolo 4 vteřin. Po vyčtení je potřeba čekat alespoň 30 vteřin před dalším spuštěním této funkce, neboť jinak by mohlo dojít k odpojení napájecího třívoltového napětí. Jako myšlenka pro další vývoj je ve finální verzi návrhu desky plošných spojů použit vstup, umožňující realizaci A/D převodníku. K funkci vyčtení elektroměru je přidáno několik dalších inicializačních funkcí. Funkce vyčtení elektroměru dále nedělá prvotní inicializaci procesoru. Prvotní inicializaci zajišťuje hlavní program. Funkce vyčtení elektroměru zajistí signál na vstupu RQ a dvě vteřiny vyčká. Teoretická minimální doba na vstupu RQ je 1,65 vteřiny, ale pro jistotu je zde inicializován dvouvteřinový interval. Po této době dojde k přepnutí komunikačního rozhraní z vnějšího vstupu na vstup vnitřní, který používá mnou navržený hardware. Funkce pracuje na principu stavového automatu a prochází jednotlivými stavy v návaznosti na odpovědi elektroměru.

Vyčtení je provedeno jako formátové čtení dat podle IEC 1015, mód C. Po přepnutí komunikačního rozhraní následuje povolení přerušení a odvysílání otevírací sekvence. Tato komunikace probíhá na základní komunikační rychlosti, která je pro elektroměry ZPA této řady 1200 Bd. Při programové realizaci

v tomto místě docházelo k chybě, kterou jsem musel analyzovat pomocí osciloskopu s využitím Single módu, normálního spouštění časové základny a spuštění od sestupné hrany. Z této analýzy jsem bohužel nezjistil žádnou chybu, pouze jsem potvrdil správně vyslanou inicializační sekvenci. Chyba byla ve skutečnosti v technické dokumentaci, kterou jsem dostal od výrobce, kde se jako základní komunikační rychlost uváděla hodnota 300 Bd. Po dotazu na výrobce mi zaslali aktualizovaný datový list, kde jako základní komunikační rychlost je již využívána rychlost 1200 Bd. Po změně v nastavení sériového portu již skutečně došlo k rozlišení elektroměru.

Elektroměr se ohlašuje svým přiděleným kódem a lze z tohoto kódu vyčíst výrobce, maximální použitelnou přenosovou rychlost, použitý typ elektroměru v konkrétní verzi. Pro naše potřeby je důležitý hlavně údaj o maximální použitelné rychlosti. Funkce vyčtení elektroměru ani nekontroluje tento údaj, protože jsem vycházel z informací výrobce, že všechny elektroměry by měli komunikovat rychlostí 4800 Bd.

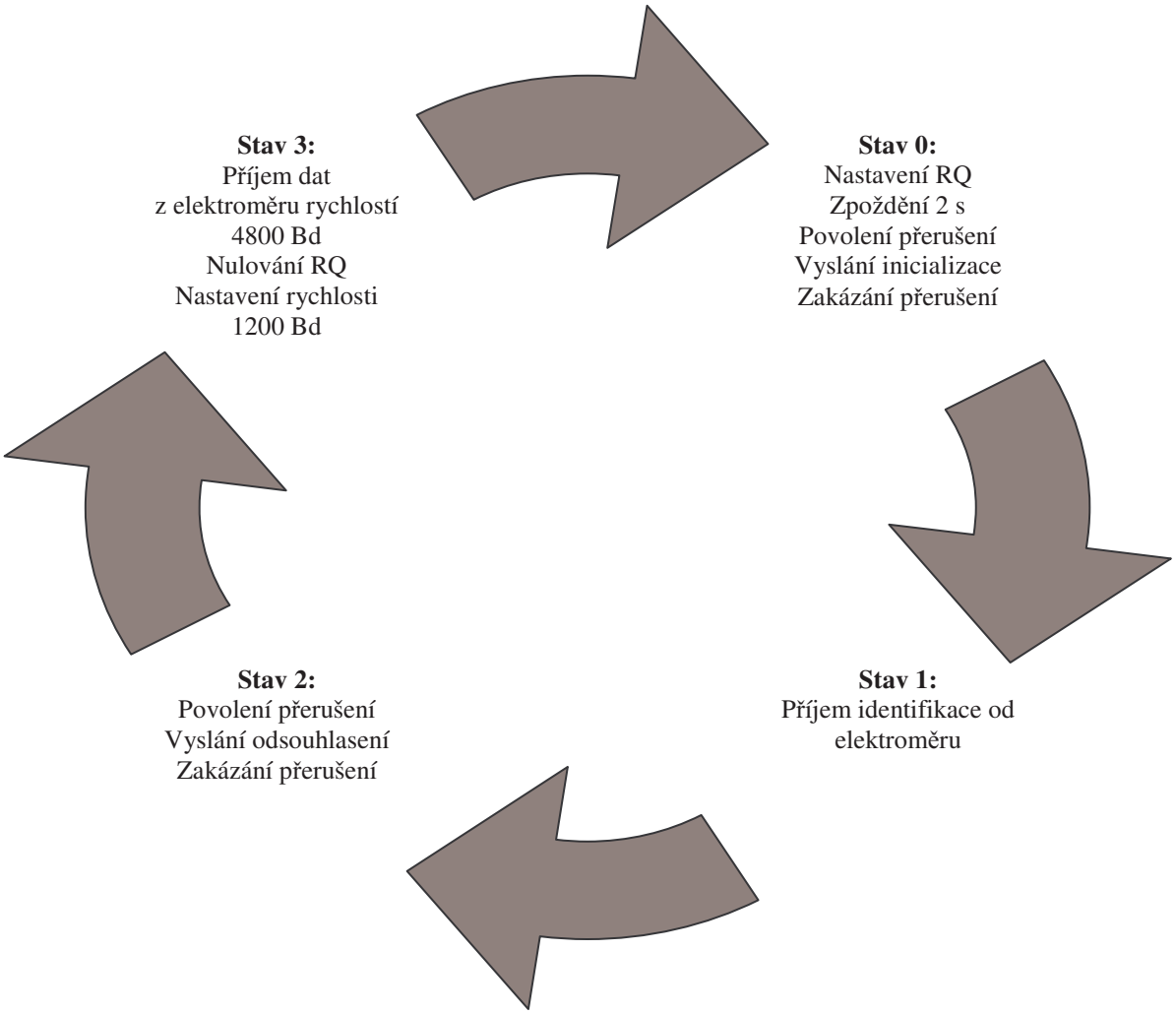
Po přijetí rozlišení proto postoupí stavový automat do dalšího stavu a vyšle se pomocí přerušení odsouhlasující sekvence, ve které je příkaz na vyslání dat rychlostí 4800 Bd. Po ukončení vyslání dat dojde k přenastavení sériového portu na rychlost 4800 Bd pomocí funkce ***inicializace4800()***. Následuje čekání na příjem dat. To probíhá opět v přerušení procesoru a v jiném stavu stavového automatu. Detekuje se znak <ETX>, který dává informaci o ukončení zaslání bloku dat. Po tomto znaku ještě následuje kontrolní součet.

Úspěšné přijetí je ukončeno návratem do výchozího stavu stavového automatu, změnou komunikační rychlosti sériového portu na 1200 Bd pomocí volání funkce ***inicializace1200()*** a je vynulován požadavek na připojení vnitřní seriové linky. Následuje opuštění funkce ***elektromer()*** do hlavního programu. Toto je zakončení principiální funkce zajišťující vyčtení elektroměru. Pokračovat budu ukázkou ze zdrojového kódu funkce, kde lze nalézt všechny výše uvedené činnosti.

3.2.1 Inicializace používaných proměnných, vytvoření zpráv odesílaných elektroměru

```
static int zprava,id_prijem,id_ident,id_odsouhlaseni,state,timerAready;
static char inicializace[5]={0x2F,0x3F,0x21,0x0D,0x0A};
char identifikace[21];
static char odsouhlaseni[6]={0x06,0x30,0x34,0x30,0x0D,0x0A};
char prijem[2400];
```

3.2.2 Funkce elektroměr a jednotlivé obsluhy přerušení, stavový model




```
void elektromer(void)
{ state = 0,                          //inicializuji stav na 0
  P4OUT = 0x01;                       //nastaveni výstupu RQ do aktivního stavu
  delay(0xA8C3);                      //zpoždění 2 sekundy
  TXBUF1=inicializace[zprava];        //vyslání prvního znaku inicializace
  zprava++;                           //posunuti ukazatele
  IE2 |= UTXIE1;                      //povolení přerušení pro vyslání dat
  while (state!=2){ }                 //čekání na přijetí rozlišení elektroměru
  TXBUF1=odsouhlaseni[id_odsouhlaseni];//vyslání prvního znaku odsouhlasení
```

```
  id_odsouhlaseni++;                  //posunutí ukazatele
  IE2 |= UTXIE1;                      // povolení přerušení pro vyslání dat
  while (state!=3){}                  // čekám dokud nevyšlu všechna data
  while (!(U1TCTL&TXEPT)){}           // proběhla celá akce vyslání odsouhlasení
  inicializace4800();                 // volání funkce na změnu rychlosti
  while (state!=0){}                  // čekání na dokončení vyčtení dat
}

// UART1 TX ISR                       //obslužná rutina přerušení pro vyslání dat
#pragma vector=USART1TX_VECTOR __interrupt void usart1_tx (void)
{
  switch (state){                     //rozpoznání stavu stavového automatu
  case 0:                             //rozpoznání stavu 0 stavového automatu
    {TXBUF1=inicializace[zprava];     //vysílám znak
      zprava++;                       //posouvám ukazovátko
      if (zprava == 5)                //kontrola vyslání celé zprávy
        {state = 1;                   //posunu se do dalšího stavu automatu
         zprava = 0;                  //ukazovátko nastavím na začátek zprávy
         IE2 &= ~UTXIE1;              //zakázání přerušení
        }
    }
      break;                          //opuštění rutiny
  case 2:                             // rozpoznání stavu 2 stavového automatu
      TXBUF1=odsouhlaseni[id_odsouhlaseni]; //vysílám znak
      id_odsouhlaseni++;              //posouvám ukazovátko
      if (id_odsouhlaseni == 6)       //kontrola vyslání celé zprávy
        {state = 3;                   //posunu se do dalšího stavu automatu
         id_odsouhlaseni = 0;         //ukazovátko nastavím na začátek zprávy
         IE2 &= ~UTXIE1;              //zakázání přerušení
        }
      break;                          //opuštění rutiny
  }
}

// UART1 RX ISR                       // obslužná rutina přerušení pro příjem dat
#pragma vector=USART1RX_VECTOR __interrupt void usart1_rx (void)
{ switch (state){                     //rozpoznání stavu automatu
    case 1:                           //pokud je stav 1, pokračuje se zde
      identifikace[id_ident] = RXBUF1;        // co mi prijde po seriovce, ulozim
      id_ident++;                     // posunu si ukazatel na prázdné místo
      if (identifikace[id_ident-1]==0x0A)     // hledám znak <LF> - konec zprávy
        {state = 2;                           // měním stav stavového automatu
         id_ident= 0;                         // ukazovátko nastavuji na nulu
        }
      break;                                  // opuštění rutiny
    case 3:                                   // pokud je stav 3, pokračuje se zde
       prijem[id_prijem]=RXBUF1;              // uložení příchozích dat od
elektroměru
```

```
      id_prijem++;                          // posunutí ukazovátka
      if (prijem[(id_prijem-2)]==0x03){     // kontroluji zda nepřišel znak <ETX>
        id_prijem = 0;                      // pokud přišel, nuluji ukazovatko
        P4OUT = 0x00;                       //nastaveni RQ do neaktivního stavu
        inicializace1200();                 //inicializace pro rychlost 1200 Bd
        state = 0;                          //změna stavu stavového automatu
        }
      break;                                //opuštění rutiny
    }
}

void inicializace4800(void)
{
 UBR01 = 0x06;                  // nastavení děličky
 UBR11 = 0x00;                  // ... ~300Baud z 32.768 Hz...
 UMCTL1 = 0x77;                 // ... s modulací pro 300Baud
 UCTL1 &= ~SWRST;               // opuštění reset modu, inicializace UART
}
void inicializace1200(void)
{
 ME2 |= UTXE1 + URXE1;          // USART1 TXD/RXD aktivace
 UCTL1 |= PENA+PEV;             // 7 Bitová data + parita
 UTCTL1 |= SSEL0;               // UCLK = ACLK - hodinový krystal
 UBR01 = 0x1B;                  // nastavení děličky
 UBR11 = 0x00;                  // ... ~300Baud z 32.768 Hz...
 UMCTL1 = 0x94;                 // ... s modulací pro 300Baud
 UCTL1 &= ~SWRST;               // opuštění reset modu, inicializace UART
}
```

Tímto končí výpis funkcí souvisejících s vyčtením elektroměru. Samozřejmě by tato funkce šla napsat lépe, ale pro demonstrativní použití funkčnosti navrhnutého hardwaru je myslím dostatečná. Z výše uvedeného je patrná hardwarová závislost pro použití s procesorem MSP430F1610. Knihovna zajišťující komunikaci po M-BUSu je psaná pro opětovné využití a tudíž by měla splňovat plnou hardwarovou nezávislost.

3.3 Knihovna MBUS

Knihovna M-BUS slouží ke zpracování linkové vrstvy protokolu M-BUS. Pro hardwarovou nezávislost bylo nutné definovat 2 hlavičkové soubory, jeden pro komunikaci s aplikačním úrovní, druhou pro komunikaci s fyzickou úrovní. Fyzická úroveň je již hardwarově závislá a proto vznikl druhý hlavičkový soubor. Po připojení na sběrnici a tím pádem i získání dat z fyzické vrstvy automaticky knihovna vyhodnotí přijatá data. Protože komunikace je typu Master-Slave a jsou vyžadována potvrzení přijatých dat, knihovna automaticky zajišťuje potvrzování přijatých zpráv a předání došlých dat aplikační vrstvě. Pouze v případě přijetí žádosti o data knihovna pouze tuto žádost předá a očekává se, že aplikační vrstva na tuto žádost bude reagovat předáním dat k vyslání. Přijatá data jsou aplikační vrstvě předána pomocí globální proměnné *rxdata* a velikosti *rxlenght*. Pokud je *rxlenght* nenulové, indikuje to aplikační vrstvě platná příchozí data, která je potřeba zpracovat. Příjem dat se provádí voláním funkce ***mbus_receive()*** z fyzické vrstvy, která jako parametr předává příchozí data. V praktické aplikaci je zprostředkování dat jednoduché. V mém případě se jedná o prosté vložení funkce ***mbus_receive()*** do obslužné rutiny přerušení od seriového portu.

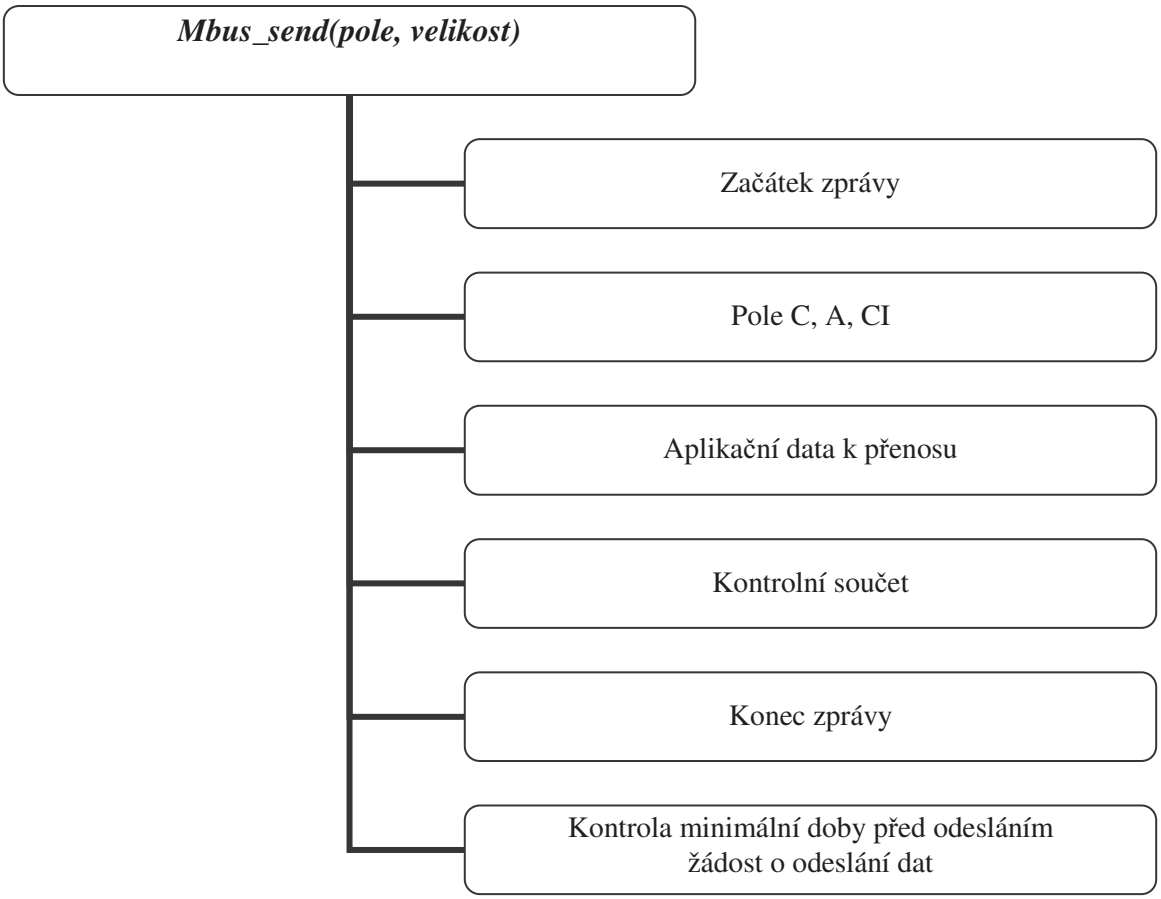
```
// UART0 RX ISR
#pragma vector=USART0RX_VECTOR__interrupt void usart0_rx (void){
  mbus_receive(RXBUF0);
}
```

Z toho příkladu je vidět, že na fyzickou vrstvu jsou nároky skutečně minimální a použití knihovny by mělo být velmi snadné, ba přímo i intuitivní. Pro správnost příjmu je nutné mít správně nastavenou adresu jednotky. Implicitní nastavení je adresa 100. Pro nastavení adresy je připravena funkce ***mbus_setaddress()***, která předává požadovanou adresu nastavovanou aplikační vrstvou. Pro kontrolu nastavení adresy je připravena funkce ***mbus_getaddress()***, která vrací aktuální adresu, na které přijímá daná jednotka. Další použitou funkcí je funkce ***mbus_send()***, použitelná pro vyslání aplikačních dat k master jednotce. Funkce je předáváno pole a jeho velikost. Funkce doplní zprávu do správného formátu doplněním startbytu a dalších

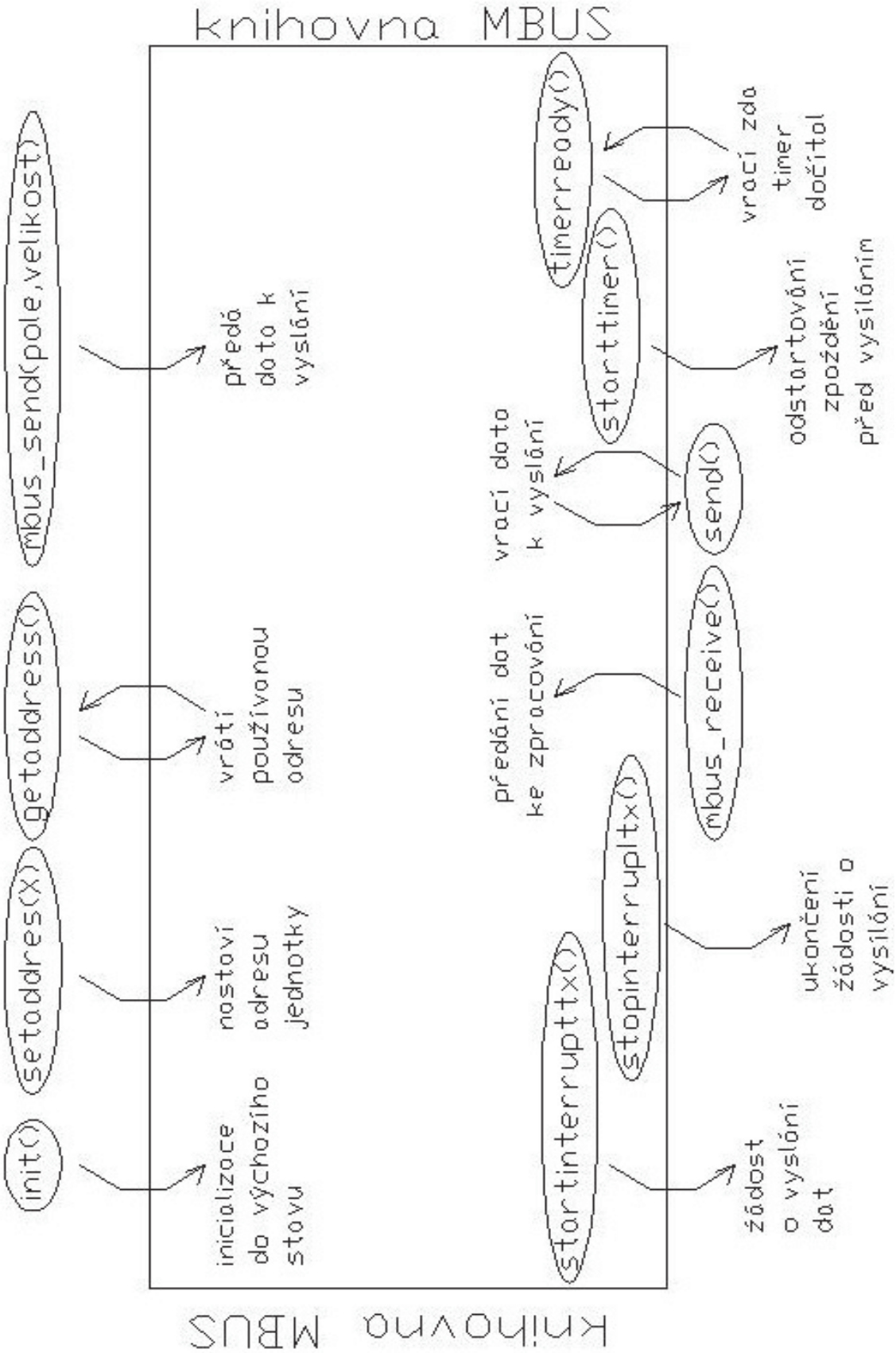
potřebných dat. Tato funkce potřebuje ke správné činnosti možnost označit nutnost vyslání dat. Toto je realizováno funkcí ***startinterrupttx()***. Knihovna M-BUS tuto funkci volá a funkce je definována na úrovni fyzické vrstvy. Další takto využívanou funkcí je ***stoptinterrupttx()***. Tato funkce je opět definována na fyzické vrstvě a ukončuje vysílání dat. Předání dat je realizováno voláním funkce ***send()*** z fyzické vrstvy. Funkce ***send()*** vrací jako návratovou hodnotu data k vyslání. Funkce je definována v knihovně M-BUS a při jejím volání se zkontroluje zda již nebyla odvysílána celá zpráva. Pokud je vysílán poslední znak, tak dojde k volání funkce ***stoptinterrupttx()*** a tím i k ukončení vysílání.

Vzhledem k složitosti a komplikovanosti funkce považuji za dostatečné uvedení principielních schémat knihovny a nejdůležitějších funkcí. Zdrojový kód knihovny i uvedených funkcí je přiložen na disku CD.

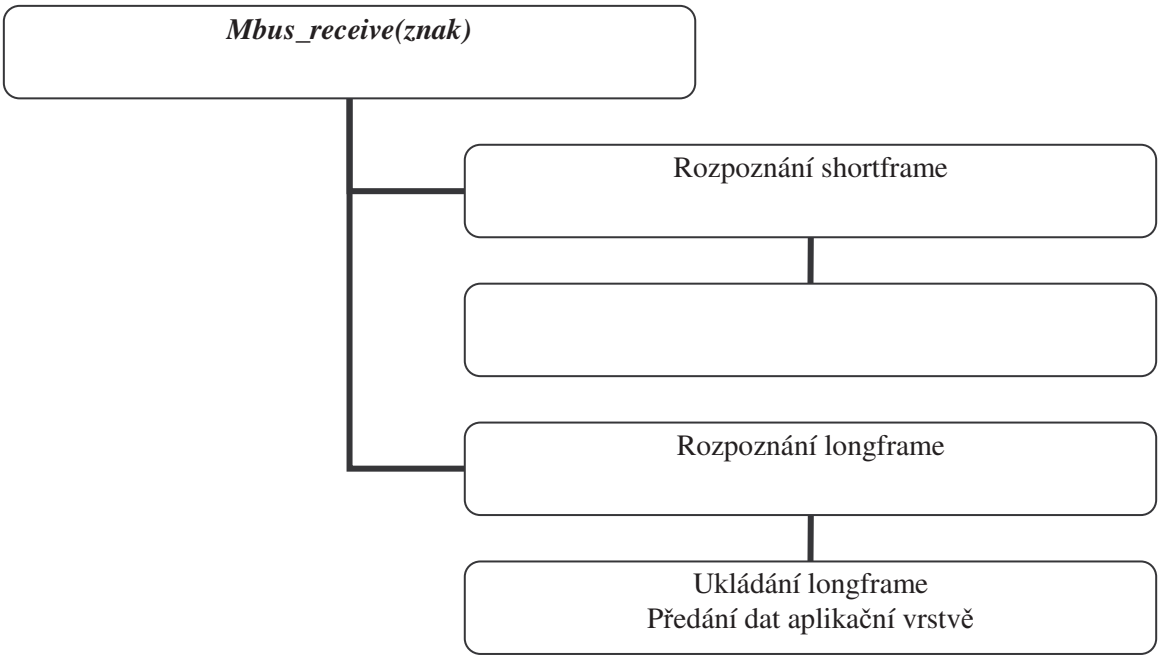
3.3.1 Diagramy vytvořených funkcí



Obr. 08 – diagram funkce ***mbus_send()***



Obr.09 – principielní schéma fungování knihovny ***M-BUS***



Obr. 10 – diagram funkce ***mbus_receive()***

Uvedené diagramy by měli osvětlit fungování knihovny ***mbus*** pro její následné použití. Na základě knihovny zabezpečující komunikaci na linkové vrstvě standardu je již možné psát aplikační protokol pro libovolnou aplikaci.

3.4 Aplikační vrstva programu

Aplikační vrstva není použita ze standardu M-BUS, protože již existuje aplikační vrstva využívaná firmou ZPA. Bohužel tento aplikační protokol neznám, vytvořil jsem pouze jednoduchý program, zajišťující vyčtení elektroměru prostřednictvím funkce elektroměr() a zajišťuje zpracování příchozích dat od knihovny M-BUS.

Vyčtení elektroměru se provádí jednou za 10 minut a tento čas je měřen prostřednictvím spuštění časovače a čekáním na jeho přetečení. V obsluze přerušení se inkrementuje čítač průchodů až do doby uplynutí požadovaných 10-ti minut. Až je proveden poslední průchod je zavolána funkce vyčtení elektroměru.

Komunikace prostřednictvím knihovny M-BUS probíhá tak, že Master jednotka vyšle požadavek na předání dat. Aplikace tento požadavek zpracuje a reaguje odesláním dat k Master jednotce. Vzhledem k velikosti dat Master své vysílání opakuje a aplikace pokračuje v odesílání dalšího bloku dat. Tento proces se opakuje až do úplného vyslání dat z elektroměru.

Druhou možností je že Master jednotka zašle požadavek na hodnotu konkrétního registru elektroměru. Aplikace na tuto událost reaguje nalezením příslušných dat v a jejich vysláním prostřednictvím knihovny M-BUS. Typicky by se mělo jednat o sběr aktuálního stavu odebrané energie v jednotlivých tarifech. Tím by se snížila doba komunikace a zrychlilo by se vyčtení jednotek, protože by nedocházelo k několikanásobnému vysílání žádostí o data a odpovědí. Údaj o spotřebované energii je naprosto dostatečným údajem pro vyúčtování energií a proto má tato funkce smysl.

Aplikační vrstva je zde vytvořená hlavně kvůli demonstraci skutečného předpokládaného použití a nepočítá se s tím, že by tato konkrétní aplikace byla použita jinak než pro laboratorní účely.

3.5 Použité vývojové prostředí

Používal jsem k vytváření, simulaci a testování volně šiřitelnou verzi vývojového prostředí IAR Embedded Workbench společnosti IAR Systems. Toto vývojové prostředí je ve své volně dostupné verzi ke stažení například z webových stránek společnosti Texas Instruments a obsahuje podporu procesoru MSP430 v různých verzích. Tento produkt je plně funkční až do velikosti 4 KB.

K navázání spojení mezi počítačem a modulem elektroměru jsem používal standardního připojení JTAG. Toto rozhraní je hardwarově podporováno procesorem MSP430, ale není standardní součástí počítačů. Vzhledem k této skutečnosti jsem si musel ještě sestavit programátor. Ten využívá paralelního portu a je schopen napájet přes JTAG i jednotku využívající třívoltové napájení. Tato skutečnost se hodí pro realizaci a ladění programu, protože již nepotřebuje další napájení. To usnadňuje práci v domácích podmínkách.

V GUI lze nastavit jako cíl ladění vestavěný simulátor, nebo hardwarový prostředek, v mém případě prostřednictvím paralelního portu. Samozřejmostí je kompilace s kontrolou formální správnosti napsaného kódu. Po slinkování dochází k nahrání programu do cílové jednotky. K dispozici je možnost krokování programu, vstupu do funkcí, běhu programu až ke kurzoru a spuštění do nepřetržitého běhu. Podle použitého procesoru lze nastavit různé množství breakpointů.

Další z vlastností je zobrazení registrů procesoru, automatického zobrazování měněných parametrů, zobrazení aktuálního obsahu proměnných.

Omezení na 4 kilobajty kódu se ukázalo pro vývoj takto jednoduché aplikace jako neomezující činitel. Myslím si, že toto vývojové prostředí splňuje všechny současné požadavky na funkce a přehlednost. Práce je velice intuitivní a je podporováno i více pracovních ploch, pro práci na více projektech současně.

4 Závěr

Tato práce ukazuje možnost realizace modulu pro vyčtení elektroměru, včetně návrhu hardwaru a uvedení možných chyb při vývoji. Uvedené schéma lze skutečně využít pro požadovanou funkci v průmyslové praxi. Použitím návrhového systému EAGLE je splněna funkce přenositelnosti a opětovné reprodukovatelnosti. Jako jeden z výstupů je i soubor ve formátu Gerber274x, který byl použit pro vytvoření desky plošných spojů. Práce na přepracovaném modulu pro testování firmou ZPA již začaly.

Použití softwarové vybavení týkající se knihovny M-BUS je podrobně popsáno v hlavní stati. Zdrojový kód se nachází na přiloženém disku CD a je komentován pro snadnější pochopení prováděných operací. Pro hlubší pochopení je ovšem potřeba seznámit se se standardem M-BUS a definicí jeho linkové vrstvy.

Uvedený zdrojový kód lze opětovně využívat v jiných aplikacích za předpokladu dodržení požadavků jednotlivých funkcí. Pravděpodobně lze kód optimalizovat pro dodržení maximální bezpečnosti přenosu dat a odpovědí.

5 Přílohy

5.1 Obsah disku CD

Rozměry desky do elektroměru

Schéma zapojení optočlenů

Princip komunikace mezi elektroměrem a vyčítajícím zařízením

Seznam prefixů

Start ZE310

Knihovna ***mbus.c*** s hlavičkovými soubory

Program s funkcí vyčti elektroměr

Katalogový list tranzistoru BSS 84

Deska plošných spojů v několika verzích

Vývojové prostředí IAR Embedded Workbench

Standard M-BUS v. 4.8

Katalogový list MSP430F1610

MSP430x1xx Family Users guide

MSP-FET430 Flash Emulation Tool

Arbeit mit dem MSP430-Mikrokontroller

Obrázky z osciloskopu – rozlišení elektroměru

Bakalářská práce ve formátu PDF

6 Seznam použité literatury a jiných zdrojů

[1] Kocourek, P., Novák, J.: Přenos informace. Skripta ČVUT, Praha 2003
[2] Haasz, V., Roztočil, J., Novák, J.: Číslicové měřicí systémy. Monografie ČVUT, Praha 2000
[3] Standard Meter Bus v. 4.8
[4] Produktový list součástky TSS721A – SLAS222, www.m-bus.com
[5] Produktový list součástky MSP430F1610 – MSP430F1610, www.ti.com
[6] MSP430x1xx Family Users guide – SLAU049f, www.ti.com