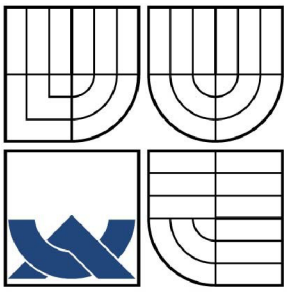
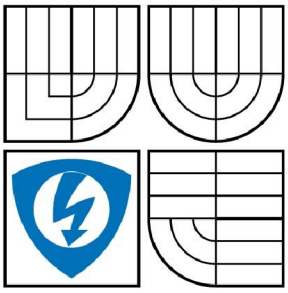


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

JEDNODUCHÝ PROGRAMÁTOR JEDNOČIPOVÝCH PROCESORŮ

SIMPLE ONE-CHIP MICROCONTROLLER PROGRAMMER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ VERNER
AUTHOR

VEDOUCÍ PRÁCE — ING. VÍT DANEČEK
SUPERVISOR

BRNO 2009

LIST SE ZADÁNÍM, KVŮLI SPRÁVNÉMU ČÍSLOVÁNÍ STRÁNEK

ABSTRAKT

Hlavním cílem bakalářské práce je seznámení s problematikou programování mikropočítačů z rodiny AVR firmy Atmel a vytvoření oživené konstrukce jednoduchého AVR programátoru, který je navíc rozšířen o testovací uživatelské prvky jako jsou LED, tlačítka, LCD displej, DA převodník.

První část práce se zabývá problematikou teorie programování pamětí mikropočítače a popisem programovacích algoritmů. Paměti mikropočítačů AVR lze programovat trojím způsobem. Nejrozšířenější způsob je metoda sériové programování přímo v systému, vžila se pro to zkratka ISP. Tento způsob programování poskytuje uživateli jednoduchou a rychlou obsluhu. Jiný způsob programování(paralelní nebo sériové) vyžaduje napětí 12 V, proto se mu říká programování vysokým napětím. Konstrukce programátoru je koncipována pro všechny výše zmiňované způsoby programování. Z předchozích znalostí o programování pamětí je napsán zdrojový kód programu, umožňující obsluhovat komunikaci příkazů z PC a následné provádění programovacích instrukcí v cílové součástce. V následující kapitole je popsáno vytvořené hardwarové části konstrukce zapojení, které zároveň poslouží k uživatelské obsluze k používání přípravku. Závěrečná část bakalářské práce pojednává o popisu konektorů potřebných pro programování metodou ISP i vysokým napětím. Je zde popsáno, který typy mikropočítačů AVR programátor podporuje a popis jak součástku do programátoru připojit a nastavit.

V přílohách bakalářské práce jsou výstupy potřebné pro zhotovení konstrukce a těmi jsou schematická dokumentace elektrického zapojení obvodu programátoru, deska plošného spoje, elektronická dokumentace v programu Eagle, rozpiska použitých součástek a zdrojový i binární obslužný program potřebný pro oživení řídícího mikropočítače programátoru.

KLÍČOVÁ SLOVA

programátor, AVR, mikropočítač, procesor, jednočip, ISP, programování, vývojový kit

ABSTRACT

The main purpose of my bachelor thesis is acquaint with issue programming of Atmel AVR microcontrollers and to create a simple chip-programmer with component units like LEDs, switches, LCD display, DA converter for testing and development.

The first part deal about questions of theory programming memory inside microcontrollers and descrption of programming algorithm. The memories is possible program in three way. The most used method is method call „In System Programming“ - ISP. This method provide easy and fast manupulation. Order method of programming memory require 12V supply. These method are parallel and serial high voltage programming. The concept of programmer is design to all method of programming memory. From previous information about programming memory was written source code of program to control programmer that receive command from computer and ensure performing of right algorithm in target microcontroller. In the next chapter is clarified design of hardware items and there is the simple user manual of items and installation of programmer. The final part of thesis explains how to use programmer's two-row-connectors and setting jumpers for programing in daily work.

In attachments are electrical scheme, list of devices, printed circuit board, files of circuit board and scheme in Eagle format, source code and binary program to control microcontroller of simple chip-programmer. These attachments are needed for make a chip-programmer.

KEYWORDS

programmer, AVR, microcontroller, procesor, single chip, ISP, programming, starter kit,

VERNER, L. *Jednoduchý programátor jednočipových procesorů: bakalářská práce.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 57 stran, 14 stran příloh. Vedoucí semestrální práce Ing. Vít Daneček.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Jednoduchý programátor jednočipových procesorů“ jsem vypracoval samostatně pod vedením vedoucího semestrálního projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Vítu Danečkovi za poskytnutí odborné a pedagogické pomoci během zpracování mé bakalářské práce.

V Brně dne ....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Zvládnutím technologie výroby integrovaných obvodů vysoké integrace umožnilo výrazně zmenšit cenu počítačů a hlavně rozměr natolik, že dnes se integruje „počítač“ do jediného čipu, kterým se říká mikropočítač.

Jednočipové mikropočítače nebo též z anglického jazyka mikrokontroléry(MCU, uP) jsou integrované obvody spojující mikroprocesor, paměť(malé množství RAM pro program a data) a obvody vstupu/výstupu v jediném čipu(v pouzdře jednoho integrovaného obvodu). Dále může mikropočítač obsahovat komunikační linky, A/D převodníky, čítače/časovače, řadiče přerušení, generátor hodinového signálu atd., takže může bez přídavných pomocných obvodů obsáhnout celou aplikaci. Mikropočítače se používají v automaticky řízených zařízeních jako jsou řídící jednotky v automobilovém průmyslu, kalkulačky, regulátory topení atd. Každé moderní měřící zařízení z oblasti měřící a regulační techniky si lze bez mikropočítačů jen z těží představit. Typické jednoduché periferní zařízení pro činnost mikropočítačů, které se připojují na obvody vstupu/výstupu jsou tlačítka, relé, LED diody, LCD displeje, serva, motorky(samozřejmě připojené přes výkonový spínací prvek jako tranzistor) a jiné.

V dnešní době je na trhu celá řada různých mikropočítačů. Cena těchto mikropočítačů je nízká a pohybuje se v řádu desítek až stovek korun. Mikropočítače vyrábí několik světových výrobců v širokém sortimentu typů a velikostí. Nejmenší typy jsou vyrobeny technologií SMD a mají jen 8 vývodů včetně napájení. Dražší typy mívají pouzdra s několika desítkami vývodů.

V roce 1997 představila firma ATMEL po zdokonalení původních mikroprocesoru z řady 8051 své nové 8b procesory s RISC architekturou s tzv. označením AVR. Zkratka AVR vznikla podle jmen autorů Alf Vegard RISC ale oficiálně znamená Advanced Virtual RISC. Jedná se o mikroprocesory s Harvardskou architekturou a redukovanou instrukční sadou(RISC). RISC architektura nabízí přednosti jako jednocyklové instrukce, vyšší taktovací frekvence spojená s vyšším výkonem stejně jako efektivní optimalizace překladu, což vyhovuje psaní kódu i ve vyšších programovacích jazycích, zejména jazyk C.

Cílem této bakalářské práce je nastínit problematikou programování 8b mikroprocesorů z rodiny mikrokontroléru typu AVR. Počáteční kapitola se bude zabývat teoretickým studiem a popisem způsobů programování vnitřních pamětí mikrokontroléru, kontroly a zamykání pamětí. Čipy lze programovat trojím způsobem. Klasickým paralelní programováním, čímž disponují 20pinové a větší mikročipy. Sériovým programováním vysokým napětím, což je jediná možnost jak u malých osmi vývodových AVR mikročipů se zotavit ze zamknutého stavu. Moderním sériovým způsobem přímo v zapojení, ve které se aplikace provozuje. Techniky je tato metoda označována zkratkou ISP – In system Programming. K naprogramování AVR mikroprocesoru je zapotřebí AVR programátor, který bude cílem této práce. V následující kapitole je z nastudovaných programovacích algoritmů sestaven, podrobně rozebrán a popsán zdrojový kód programu v jazyce symbolických instrukcí pro překladač assembler AVR, potřebný pro oživení a fungování konstrukce programátoru. V poslední části práce bude obsažena konstrukce programátoru AVR mikrokontrolérů(elektrické schéma zapojení, deska plošného spoje a rozpiska součástek) s velmi podrobným popisem technického zapojení všech uživatelských prvků. V konstrukci budou obsaženy uživatelské testovací prvky jako pole svítivých LED, pole tlačítek, pole přepínačů, DA převodník se sběrnicí I2C a dvouřádkový LCD displej. Vyrobený programátor bude kompatibilní s programem AVR-Studio(AVRProg), CodeVision, AVROSP, Avrdude. První dva programy jsou velmi oblíbené a komfortní vývojová prostředí s možností sdružování zdrojových kódů do projektu, simulátorem a překladačem. Poslední jmenovaný program Avrdude je používaný i v OS Linux.

2 Programování pamětí mikrokontrolérů AVR

Paměti v mikropočítačích jsou elektricky mazatelné a programovatelné. Většinou se jedná o paměti typu FLASH, která je typu EEPROM. Informace v těchto pamětech je uchovávána na principu přivedením elektrického náboje na polovodičový přechod hradel unipolárního tranzistoru, zvaných buňky. Díky tenké izolační vrstvičce oxidu křemičitého jsou přivedené elektrony v hradle „uvězněny“ a tím pádem je uložen 1 bit informace. Programování tedy spočívá v přivedení elektrického náboje do hradla, kde je „zapamatován“. Při adresaci naprogramované polovodičové buňky se tranzistor otevře a dává log. 0. Mazání paměti spočívá v tom, že se z hradla uložený náboj odvede pryč a tím vznikne log. 1.

2.1 Paměťové zámky

Všechny mikrokontroléry AVR poskytují dva zámkové bity pojmenované LB1 a LB2. Nenaprogramovaný bit zámku má hodnotu 1, naprogramovaný 0. Úroveň ochrany je rozdělena do tří režimů, kde režim jedna nepředstavuje žádnou ochranu a režim 3 představuje maximální ochranu. Význam je uveden v tabulce 2.1. Zámky mohou být vymazány pouze příkazem Chip Erase, který vymaže obsah paměti Flash. Tabulka je převzata z katalogového listu [12] v sekci Memory programming, která je stejná pro ostatní typy mikropočítačů AVR.

Tab. 2.1: Paměťové zámky

Paměťové zámky			**Typ ochrany**
Režim	**LB1**	**LB2**	
1	1	1	Nenaprogramováno. Není aktivována žádná ochrana
2	0	1	Zákaz programování FLASH a E2PROM, čtení povoleno
3	0	0	Čtení a programování(zápis) zakázáno

2.2 Propojky (Fuse Bits)

Programovací propojky, nebo-li také fuse bits, slouží k nastavení procesoru, jeho zdroje hodinového signálu, ochran proti přepisu a jiných vlastností. Špatným nastavením těchto propojek můžeme uvést procesor do již nepoužitelného stavu, z kterého se lze zotavit pouze přeprogramováním vysokým napětím.

Hodnota propojky může být buď naprogramovaná(logická hodnota bitu je v 0) nebo nenaprogramovaná(logická hodnota bitu je v 1. Hodnoty propojek jsou tedy reprezentovány jako binární nebo dekadické či hexadecimální číslo bajtu. Propojkových bajtů může být 1 až 3. V technické praxi se tyto bajty značí jako LOW,HIGH,EXTENDED. Množství propojek(bajtů) závisí na konkrétním typu mikrokontroléru. Např. ATmega8 má dva bajty: nižší(low), vyšší(high)) pro uložení všech 16 propojkových bitů. Nebo mikrokontrolér ATtiny11 má pouze 8 propojkových bitů. Podrobně rozebraný popis propojkových bitů a umístění v bajtu pro různé typy mikrokontrolérů je popsáno v aplikační poznámce [11] AVR061 na stranách 25 až 27.

Stručný výpis z katalogových listů [15] a popis některých propojek(fuse bits) u mikrokontrolérů AVR:

- SPIEN (SPI download ENabled) – Když je SPIEN naprogramován na 0, je umožněno sériové programování. Výchozí hodnota je 0, takže sériové programování je možné.
- RCEN(RC oscilator ENabled) – Volí osciátor. Je-li RCEN=0, používá se jako zdroj zabudovaný RC oscilátor. Je-li RCEN=1, používá se klasický vnější krystalový oscilátor.
- FSTR (Fast Start) - Když je FSTR naprogramován na 0, procesor rychleji startuje. Výchozí nenaprogramovaná hodnota je 1.

- BODLEVEL (Brown-up Detection level) – Jestliže je BOD povolen, BODLEVEL propojka změní napájecí napětí a startovací čas. Defaultní hodnota je 1.
- BODEN (Brown-out detection) – Obvod pro hlídání napájecího napětí. Pokud je propojka nastavena, při poklesu napájecího napětí pod určitou úroveň, tak se ihned aktivuje Brown-out RESET. Napájení musí být blokováno kondenzátorem o hodnotě 47 až 100 nF.
- CKSEL – Nastavuje dobu zotavení RESETu.
- RSTDISBL(Reset Disable) – Když je propojka naprogramována 0, není možný hardwarový reset. Výchozí hodnota je 1.

Některé propojky, například SPIEN, nelze programovat pomocí sériového programování(musí být použit paralelní programátor). Hodnota propojek se příkazem Smazání Čipu nezmění.

Na závěr poznamenejme, že mikrokontroléry nemají zabudovány všechny z výše uvedených propojek. Detailní přehled a přesný obsah propojek najdeme v katalogových listech k jednotlivým mikrokontrolérům [15]. Pro několik vybraných mikrokontrolérů je z jejich katalogových listů sestavena tabulka 2.2 s přehledem propojek.

Tab. 2.2: Propojky mikrokontroléru (+ obsahuje, - nemá)

Typ mikrokontroléru	SPIEN	FSTRT	RCEN	BODLEVEL	BODEN	CKSEL2..0	RSTDISBL
AT90S1200	+	-	+	-	-	-	-
AT90S2313	+	+	-	-	-	-	-
AT90S2343	+	-	+	-	-	-	-
AT90S4433	+	-	-	+	+	+	-
AT90S8515	+	+	-	-	-	-	-
AT90S8535	+	+	-	-	-	-	-
ATtiny22	+	-	+	-	-	-	-
ATtiny11	+	+	-	-	-	+	+
ATtiny12	+	-	-	+	+	CKSEL3..0	+
ATtiny15	+	-	-	+	+	CKSEL1..0	+

2.3 Signatura

Všechny AVR mikrokontroléry obsahují 3 bajty signatury, identifikující výrobce a typ mikrokontroléru. Tento kód může být čten sériovým i paralelním programováním. Tyto tři bajty jsou uloženy na speciálním adresním prostoru(mimo paměť Flash a EEPROM).

První bajt má vždy hodnotu $1E (indikuje výrobce ATMEL). Druhý bajt indikuje velikost FLASH($90 – 1KB, $91 – 2KB, $92 – 4KB, $93 – 8KB). Poslední bajt indikuje konkrétní typ (několik mikrokontrolérů může mít stejnou velikost FLASH).

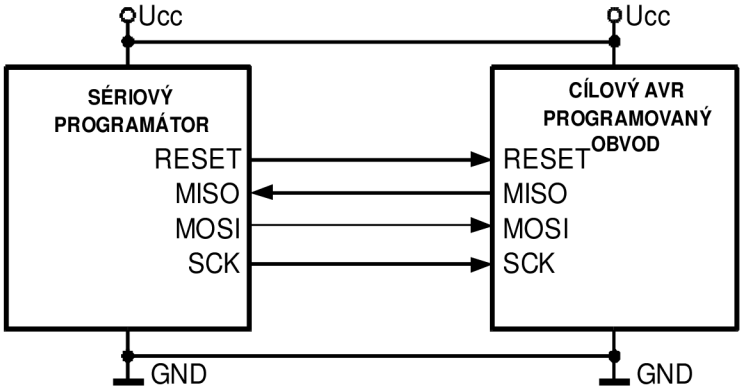
Signaturu nelze číst v režimu uzamknutí 3 (viz kapitola 2.1).

2.4 Sériové programování - ISP

Sériové programování umožňuje programovat mikrokontrolér přímo v systému. Odpadá složité vkládání/vyjímání(odletování) čipu z/do programátoru respektive do aplikační desky. To ušetří mnoho peněz a času během vývoje. Obvykle není u ISP potřeba vyššího napětí jak 5 V jako u paralelního programování. Tímto způsobem lze také programovat některé propojky, kromě propojky SPIEN, která povoluje sériové programování. Továrně je sériové programování povoleno.

Používané rozhraní pro sériové programování je SPI.

SPI (*Serial Peripheral Interface*) je sériové periferní rozhraní. Používá se také pro komunikaci mezi řídícími mikroprocesory a ostatními integrovanými obvody. Komunikace je realizována pomocí společné sběrnice. Adresace se provádí pomocí zvláštních vodičů, které při logické nule aktivují příjem a vysílání zvoleného zařízení.



Obr. 2.1: Šesti vodičové zapojení mezi programátorem a cílovým mikrokontrolérem

2.4.1 Ideové zapojení

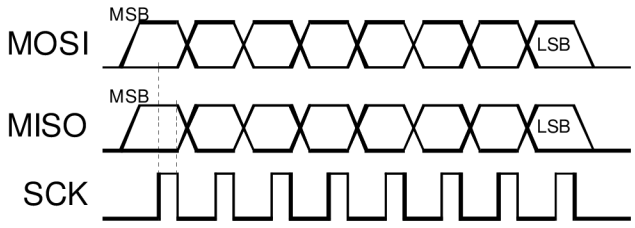
Sériové rozhraní obsahuje SCK(hodiny), MOSI(výstup) a MOSI(vstup). Viz obr. 2.1. Během programování AVR, sériový programátor vždycky pracuje jako řídící a cílový systém je podřízený. Při každém hodinovém pulsu na vodiči SCK se přenese jeden bit z řídícího programátoru do podřízeného cílového systému po vodiči MOSI. Současně se přenese jeden bit z cílového čipu do programátoru po vodiči MISO. Pro zajištění správné komunikace sériového programování je nezbytné propojit GND(země) programátoru a programovaného čipu. Napájecí napětí programovaného čipu musí být v rozsahu 2,6 až 6 V.

Tab. 2.3: Zapojení vývodů potřebné pro sériové programování (převzato z [10] str. 3)

Vývod	Popis
SCK	Hodinový signál, generované sériovým programátorem(řídící)
MOSI	Komunikace z programátoru(řídící) do programovaného mikrokontroléru(podřízený)
MISO	Komunikace z AVR do sériového programátoru
GND	Systémy musí být galvanicky propojeny zemí
RESET	K povolení sériového programování, RESET AVR musí být po celou dobu aktivní(log0). Toto řídí programátor.
Ucc	Cílový AVR musí být během programování napájen. Napájení může být z externího zdroje nebo odebírána z napájení programátoru.

2.4.2 Algoritmus sériového downloadu

Zapisovaná data jsou vzorkována náběžnou hranou SCK. Čtená data se objevují souběžně se sestupnou hranou SCK (obr. 2.2).



Obr. 2.2: Průběhy signálů při sériovém downloadu

Vysvětlivka k obrázku: MSB – Most Significant Bit: bit s nejvyšší hodnotou v binárním vyjádření
Vysvětlivka k obrázku: LSB – Least Significant Bit: bit s nejnižší hodnotou v binárním vyjádření

- Připojíme napájecí napětí mezi Ucc a GND, SCK = 0. Jakmile je RESET na cílovým čipu aktivní, tak sériové programování je připraveno. Vnitřní sériová sběrnice je aktivována a čeká na příkazy od programátoru. Je velmi důležité, aby vývod SCK byl při náběhu napájení stabilní, protože jediná hrana může být příčinou ztráty synchronizace s programátorem. Pokud programátor nedokáže zaručit SCK = 0 při náběhu napájení, je třeba na vývod RESET přivést kladný impuls(log. 0, log. 1, log. 0) od délce trvání alespoň dvojnásobku periody hodin XTAL1 po náběhu SCK = 0.
- Po aktivování resetu do log. 0 se vyčká alespoň 20ms a poté se vyšle první příkaz.

Tab. 2.4: Příklad instrukce pro povolení programování

Instrukce	MOSI, posílání do AVR	MISO, vráceno z cílového AVR
Povolení programování	$9C 53 xx yy	$zz 9C 53 xx

- Pokud dojde k synchronizaci, bude druhý bajt instrukce Povolení programování vrácen zpět v průběhu zápisu třetího bajtu. Pokud tomu tak není, lze přidat jeden impuls SCK a vyslat instrukci Povolení programování znovu. Pokud se nepodaří získat odpověď ani po 32 pokusech, je programovaný čip vadný. Tabulka 2.4 zobrazuje příklad vyslání instrukce do cílového čipu.
- Pokud je provedena instrukce Smazání čipu (vykoná se pro smazání obsahu FLASH), je třeba vyčkat dobu t_{WD_ERASE}, než je možno pokračovat dále. Viz Tab 2.6.
- FLASH a E2PROM paměti se programují po jednom bajtu instrukcí Zápis(Write), která nese jak data, tak i jejich adresu. E2PROM paměťové místo je před zápisem nových dat nejdříve automaticky vymazáno. Pro detekci úspěšného dokončení zápisu do FLASH nebo E2PROM tzv. Datový pooling (viz katalogový list [15]). Jinou možností je vyčkat dobu t_{WD_PROG}, než vyšleme další instrukci.
- Libovolné místo paměti může být verifikováno(zpětně čteno) použitím instrukce Čtení (Read), která vrátí obsah vybrané adresy.
- Na konci programování lze uvést RESET do log. 1.
- Po programování lze vypnout napájecí napětí (taková operace však není příliš častá). Současně je třeba zajistit: XTAL = 0 (krystal lze ponechat připojený), RESET = 1.

2.4.3 Formát příkazů

Všechny příkazy mají běžný formát sestavený ze čtyř bajtů První bajt obsahuje kód příkazu, druh operace a cílovou paměť. Druhý a třetí bajt obsahuje adresu paměti. Čtvrtý bajt obsahuje data v každém směru. Tabulka a algoritmy jsou převzaty z dokumentace katalogových listů [15] firmy Atmel, které jsou pro většinu součástek stejné.

Tab. 2.5: Instrukční soubor sériového downloadu

Instrukce	Formát				Operace
	Bajt 1	Bajt 2	Bajt 3	Bajt 4	
Povolení programování	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Aktivuje programovací rozhraní
Smazání čipu	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Smaže Flash, E^2PROM a zámky
Čtení kódu	0010 H000	aaaa aaaa	bbbb bbbb	oooo oooo	Čtení instrukce programu z FLASH
Zápis kódu	0110 H000	aaaa aaaa	bbbb bbbb	iiii iiii	Zápis instrukce programu do FLASH
Čtení dat	1010 0000	aaaa aaaa	bbbb bbbb	oooo oooo	Čtení dat z E^2PROM
Zápis dat	1110 0000	aaaa aaaa	bbbb bbbb	iiii iiii	Zápis dat do E^2PROM
Zápis zámku	1010 1100	1111 1AB11	xxxx xxxx	xxxx xxxx	Zápis zamykacích bitů
Čtení signatury	0011 0000	xxxx xxxx	xxxx xxbb	oooo oooo	Čtení signatury obvodu

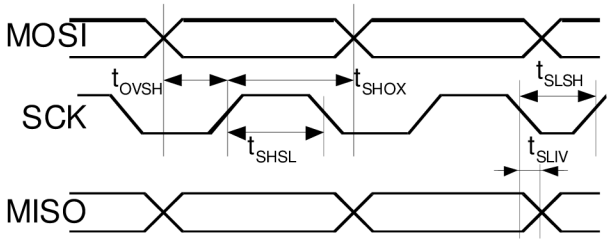
Vysvětlivka k tabulce: a = horní bity adresy, b = dolní bity adresy, H = 0 – Dolní bajt, 1 Horní bajt, o = data vysílána na vývodu MISO, i = data přijímána na vývodu MOSI, x = cokoliv,
A = zámek1, B=zámek2(bit zámku je nastaven, pokud má hodnotu 0)

2.4.4 Charakteristiky sériového downloadu

Pro úspěšné provedení sériového programování je nezbytné dodržet minimální časové intervaly řídících signálů. Důležitý je zejména kmitočet signálu SCK, předstih a přesah dat na lince MOSI. Dále jsou určeny doby mazání programu.

Po aplikaci RESET=0, musí být jako první vyslána instrukce Povolení programování (Programming enable Instruction). Instrukce Smazání čipu(Chip Erase) uvede obsah všech paměťových buněk programové i datové paměti do stavu $FF.

Algoritmus sériového programování, obrázky a tabulky byly převzaty z [4] , [10] a [15].



Obr. 2.3: Časování sériového programování

Tab. 2.6: Charakteristiky sériového programování

Symbol	Parametr	Min	Typ	Max	Jednotka
$1/t_{CLCL}$	Frekvence oscilátoru (Ucc = 2,7 - 4V)	0		4	Mhz
t_{CLCL}	perioda oscilátoru (Ucc = 2,7 - 4V)	255			ns
$1/t_{CLCL}$	Frekvence oscilátoru (Ucc = 4 - 6V)	0		12	Mhz
t_{CLCL}	Perioda oscilátoru (Ucc = 4 - 6V)	83,3			ns
t_{SHSL}	Délka kladného impulsu SCK	$4.0t_{CLCL}$			ns
t_{SLSH}	Délka záporného impulsu SCK	t_{CLCL}			ns
t_{OVSH}	Předstih dat na MOSI před SCK	$1.25t_{CLCL}$			ns
t_{SHOX}	Přesah dat na MOSI přes SCK	$2t_{CLCL}$			ns
t_{SLIV}	Vystavení dat na MISO po sestupné hraně SCK	10	16	32	ns

Tab. 2.7: t_{CLCL} je perioda kmitočtu oscilátoru, je určena mutací mikropočítače

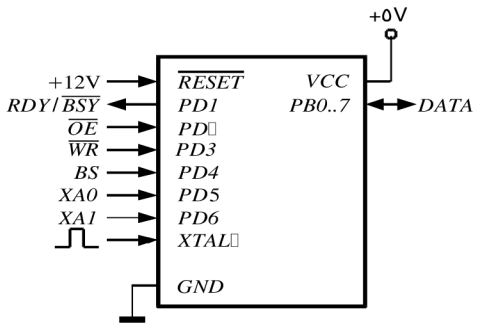
f_{MAX} [Mhz]	$t_{CLCL\ MIN}$ [ns]
4	250
8	125
10	100
12	83,3

Tab. 2.8: Minimální doby mazání a programování

Symbol	Parametr	Napájecí napětí			
		3,2 V	3,6 V	4,0 V	5,0 V
t_{WD_ERASE}	Minimální prodleva po instrukci smazání čipu	18 ms	14 ms	12 ms	8 ms
t_{WD_PROG}	Minimální prodleva po zápisu jednoho místa do Flash nebo E^2PROM	9 ms	7 ms	6 ms	4 ms

2.5 Paralelní programování

Paralelní programování je starší metoda(podobná klasickému programování mikrokontroléru z rodiny x51) k programování FLASH paměti programu, datové EEPROM paměti, zámků a propojek. Lze programovat všechny propojky. Oproti sériovému programování je složitější. Programovaný procesor je nutno vyjmout z aplikační desky a vložit do programátoru, což zpomaluje vývoj i následnou výrobu. Paralelním programováním disponují 40, 28 a 20ti vývodové mikrokontroléry AVR.



Obr. 2.4: Paralelní programování – ideové zapojení

Vysvětlivka k obrázku 2.4.: Ne všechny mikrokontroléry mají stejné rozmístění vývodů pro signály. Např. ATmega8 a další nemají kompletní bránu PB pro bajt DATA. Bližší informace viz katalogové listy [15] k jednotlivým typům AVR.

2.5.1 *Názvy signálů*

Názvy a jejich popis funkce jsou stručně znázorněny v tabulce 2.9.

Vývody XA1/XA0 určují akci, která se vykoná když na vývodu XTAL1 se objeví kladný impulz. Kódování je v tabulce 2.10.

Řídící bajt(příkaz) se posílá jako sekvence různých bitů, které jsou objasněny v tabulce 2.11, na obousměrnou datovou sběrnici a potvrdí se impulsem na XTAL1. Tímto se mikropočítač přichystá na určitou akci a očekává požadovaný bajt(adresa, data) na obousměrné datové sběrnici. Data, která takto pomocí impulsů na XTAL paralelně posíláme do mikropočítače, se zapisují do bufferu. Impulsem na vývodu WR data zapíšeme .

Tab. 2.9: Přiřazení názvů k vývodům

Názvy signálů v paralelním programování	**Název vývodu**	**I/O**	**Funkce**
RDY/BSY	PD1	O	0:zařízení je zaneprázdněno, 1:zařízení je připraveno přijmout příkaz
OE	PD2	I	Povolení programování(aktivní na nulu)
WR	PD3	I	Impuls zápisu(aktivní na nulu)
BS	PD4	I	Výběr bajtu(„0“ vybere dolní bajt, „1“ vybere horní bajt)
XA0	PD5	I	Akce XTAL bit 0
XA1	PD6	I	Akce XTAL bit 1
DATA	PB0-7	I/O	Obousměrná datová sběrnice (Výstup když OE = 0)

Tab. 2.10: Kódování XA1 a XA0

XA1	**XA0**	**Akce během kladné pulzu na XTAL1**
0	0	Načte adresu z Flash nebo EEPROM(BS určuje zda vyšší/nižší bajt)
0	1	Načte příkaz(zda dolní nebo horní bajt, určuje BS)
1	0	Načte příkaz
1	1	Žádná akce

2.5.2 *Algoritmus paralelního programování*

Tab. 2.11: Instrukce pro paralelní programování

Bajt	Instrukce
1000 0000	Smazání čipu
0100 0000	Zápis propojek
0010 0000	Zápis zámkových bitů
0001 0000	Zápis Flash
0001 0001	Zápis EEPROM
0000 1000	Čtení signatury
0000 0100	Čtení propojek a zámkových bitů
0000 0010	Čtení Flash
0000 0011	Čtení EEPROM

Tabulky 2.9 až 2.11 jsou převzaty z [13] ze sekce Parallel Programming.

Vstup do programovacího režimu

Tento algoritmus uvede zařízení do paralelního programovacího módu

1. Mezi Vcc a GND připojíme napájecí napětí +5 V a počkáme 20 až 60 us.
2. RESET = 0, BS = 0 a počkáme alespoň 100 ns.
3. Na RESET přivedeme napětí 11,5-12,5 V

Smazání čipu

Příkaz smazání čipu vymaže paměti Flash, EEPROM a zámkové bity. Propojky se nezmění. Smazaní čipu se musí provést před každým přeprogramováním.

1. Nastav XA1, XA0 na „10“. Umožní načtení instrukce.
2. BS = 0
3. DATA = “10000000“. Toto je instrukce pro smazání čipu.
4. Kladný impulz na XTAL1 načte instrukci.
5. Záporný impulz na WR o délce t_{WLWH_CE}, který vykoná smazání čipu. Časové informace jsou v tabulce 2.12.

Zápis paměti(programu) Flash

Načte příkaz „Zápis Flash“

1. Nastav XA1, XA0 na „10“. Umožní načtení instrukce.
2. BS = 0
3. DATA = 0001 0000 Toto je příkaz pro zápis flash
4. Kladný impulz na XTAL1 načte instrukci

Načtení adresu horního bajtu

5. Nastav XA1, XA0 na „00“. Umožní načtení adresy.
6. BS = 1. To vybere horní bajt.
7. DATA = adresa horního bajtu
8. Kladný impulz na XTAL1 načte adresu horního bajtu

Načtení adresu dolního bajtu

9. Nastav XA1, XA0 na „00“. Umožní načtení adresy.
10. BS = 0 To vybere dolní bajt.
11. DATA = adresa dolního bajtu
12. Kladný impulz na XTAL1 načte adresu dolního bajtu

Načtení dolního bajtu dat

13. Nastav XA1, XA0 na „01“. Umožní načtení dat
14. DATA = dolní bajt dat
15. Kladný impulz na XTAL1 načte data dolního bajtu

Zápis dolního bajtu dat

16. BS = 0. Vybere dolní bajt dat
17. Vytvoř záporný impulz na WR. Tímto se začne zapisovat bajt dat. RDY/BSY přejde do nuly
18. Čekej dokud RDY/BSY nepřejde do jedničky. Potom je možno zapsat další bajt.

Načtení horního bajtu dat

19. Nastav XA1, XA0 na „01“. Umožní načtení dat
20. DATA = horní bajt dat
21. Kladný impulz na XTAL1 načte data horního bajtu

Zápis horního bajtu dat

22. BS = 1. Vybere horní bajt
23. Vytvoř záporný impulz na WR. Tímto se začne zapisovat bajt dat. RDY/BSY přejde do nuly
24. Čekej dokud RDY/BSY nepřejde do jedničky. Potom je možno zapsat další bajt (obr. 2.5)

Pro zvýšení efektivity mohou být instrukce pro načítání čtení nebo zápisu během programování volány pouze jednou.

Čtení paměti(programu) Flash

1. Nastav XA1, XA0 na „10“. Umožní načtení instrukce.
2. BS = 0
3. DATA = “0000 0010“. Toto je instrukce pro čtení Flash.
4. Kladný impulz na XTAL1 načte instrukci.
5. Nastav XA1, XA0 na „00“. Umožní čtení programu.
6. BS = 1 (výběr horního bajtu)
7. DATA = adresa horního bajtu programu
8. Kladný impulz na XTAL1, přečte adresu vyššího bajtu.
9. BS = 0 (výběr dolního bajtu)
10. DATA = adresa dolního bajtu programu
11. Kladný impulz na XTAL1, přečte adresu dolního bajtu.
12. OE = 0, BS = 0. Dolní bajt je nyní na bráně DATA
13. BS = 1. Horní bajt je nyní na bráně DATA
14. OE = 1

Zápis paměti dat EEPROM

1. Nastav XA1, XA0 na „10“. Umožní načtení instrukce.
2. BS = 0
3. DATA = “0001 0001“. Toto je instrukce pro zápis EEPROM.
4. Kladný impulz na XTAL1 načte instrukci.
5. Nastav XA1, XA0 na „00“. Umožní čtení programu.
6. BS = 0 (výběr dolního bajtu)
15. DATA = adresa dolního bajtu paměti dat
16. Kladný impulz na XTAL1, přečte adresu dolního bajtu.
17. Nastav XA1, XA0 na „01“. Umožní načtení dat.
18. DATA = data dolního bajtu
19. Kladný impulz na XTAL1, načte data dolního bajtu.
20. BS = 0
21. Vytvoř záporný impulz na WR. Tímto se začne zapisovat bajt dat. RDY/BSY přejde do nuly
22. Čekej dokud RDY/BSY nepřejde do jedničky. Potom je možno zapsat další bajt.

Čtení paměti dat EEPROM

1. Nastav XA1, XA0 na „10“. Umožní načtení instrukce.
2. BS = 0
3. DATA = “0000 0011“. Toto je instrukce pro čtení EEPROM.
4. Kladný impulz na XTAL1 načte instrukci.
5. Nastav XA1, XA0 na „00“. Umožní čtení programu.
6. BS = 0 (výběr dolního bajtu)
7. DATA = adresa dolního bajtu paměti dat
8. Kladný impulz na XTAL1, přečte adresu dolního bajtu.
9. OE = 0. BS = O. EEPROM data jsou nyní na bráně DATA.

Programování propojek

1. Nastav XA1, XA0 na „10“. Umožní načtení instrukce.
2. BS = 0
3. DATA = “0100 0000“. Toto je instrukce pro zápis propojek.
4. Kladný impulz na XTAL1 načte instrukci.
5. Nastav XA1, XA0 na „01“. Umožní načtení dat.
6. DATA = data Bit n=“0“ naprogramuje, n=“1“ smaže propojku
   Bit 5 = SPIEN
   Bit 0 = RCEN
   Bity 7..6, 4..1 =“1“. Tyto bity jsou rezervovány a mohou být nenaprogramovány („1“).
7. DATA = data Bit n = 0 naprogramuje, n=1 smaže propojku
8. Záporný impulz na WR o délce t_{WLWH_PFB}, který vykoná naprogramování propojky. Časové informace jsou v tabulce 2.12. Programování propojky nevytvoří žádnou aktivitu na vývodu RDY/BSY.

Programování zámkových bitů

1. Nastav XA1, XA0 na „10“. Umožní načtení instrukce.
2. BS = 0
3. DATA = “0010 0000“. Toto je instrukce pro zápis zámkových bitů.
4. Kladný impulz na XTAL1 načte instrukci.

5. OE = 0, BS = 1
6. Nastav XA1, XA0 na „01“. Umožní načtení dat.
7. DATA = data Bit n = 0 naprogramuje zámkový bit
   Bit 2 = Zámkový Bit 2
   Bit 1 = Zámkový Bit 1
   Bit 7..3, 0 = 1 Tyto bity jsou rezervovány a mohou být nenaprogramovány (log. 1).
8. DATA = data Bit n = 0 naprogramuje zámkový bit
9. BS = 0
10. Vytvoř záporný impulz na WR. Tímto se začne zapisovat bajt dat. RDY/BSY přejde do nuly
11. Čekej dokud RDY/BSY nepřejde do jedničky.

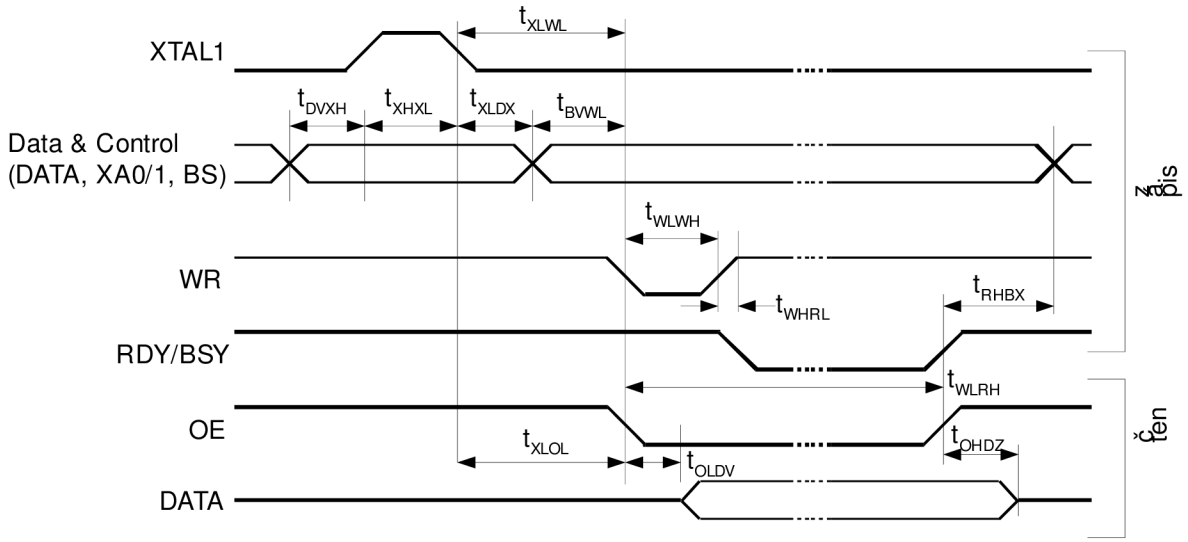
Zámkové bity mohou být vymazány pouze příkazem Smazání čipu

Čtení propojek a zámkových bitů

1. Nastav XA1, XA0 na „10“. Umožní načtení instrukce.
2. BS = 0
3. DATA = “0000 0100“. Toto je instrukce pro čtení zámkových bitů a propojek.
4. Kladný impulz na XTAL1 načte instrukci.
5. OE = 0. BS = 1. Stav propojek a zámkových bitů je nyní dostupný na bráně DATA.
   Bit 7 = Zámkový Bit 1
   Bit 6 = Zámkový Bit 2
   Bit 5 = SPIEN Propojka
   Bit 0 = RCEN Propojka
6. OE = 1

Čtení signatury

1. Nastav XA1, XA0 na „10“. Umožní načtení instrukce.
2. BS = 0
3. DATA = “0000 0100“. Toto je instrukce pro čtení zámkových bitů a propojek.
4. Kladný impulz na XTAL1 načte instrukci.
5. Nastav XA1, XA0 na „00“. Umožní čtení programu.
6. BS = 0 (výběr dolního bajtu)
7. DATA = adresa dolního bajtu paměti dat
8. Kladný impulz na XTAL1, přečte adresu dolního bajtu.
9. OE = 0, BS = 0. Signatura můžeme nyní přečíst z brány DATA



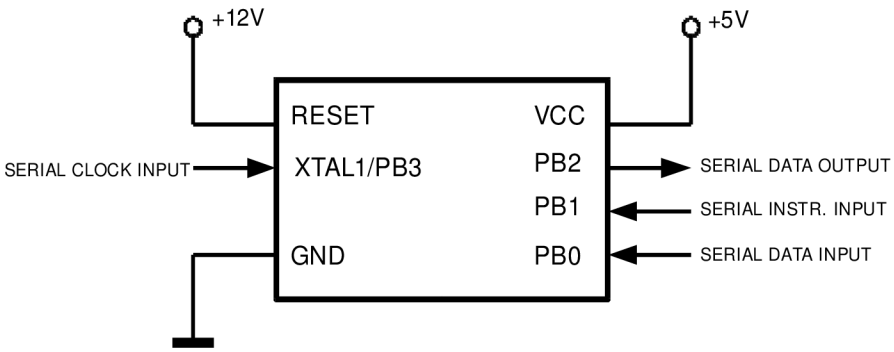
Obr. 2.5: Charakteristiky paralelního programování (převzato z [13] str. 271)

Tab. 2.12: Časové údaje paralelního programování (převzato z [13] str. 272)

Symbol	Min	Typ	Max	Jednotka
t_{DVXH}	67			ns
t_{XHXL}	67			ns
t_{XLDX}	67			ns
t_{XLWL}	67			ns
t_{BVWL}	67			ns
t_{RHBX}	67			ns
t_{WLWH}	67			ns
t_{WHRL}		20		ns
t_{WLRH}	0,5	0,7	0,9	ms
t_{XLOL}	67			ns
t_{OLDV}		20		ns
t_{OHDZ}			20	ns
t_{WLWH_CE}	5	10	15	ms
t_{WLWH_PFB}	1	1,5	1,8	ms

2.6 Sériové programování vysokým napětím

Některé menší typy AVR(např. AT90S2323, ...) kvůli nedostatku pinů nemohou disponovat paralelním programováním. Tyto malé mikroprocesory kromě klasického ISP podporují ještě režim programování napětím 12 V, které je nutno připojit na vývod RESET. Fyzické zapojení a význam jednotlivých pinů je vidět na obrázku. Oproti ISP má tato metoda několik výhod: Je rychlejší a bez tohoto tzv. High-Voltage(12V) Serial Programming bychom nebyli schopni naprogramovat propojkové bity. Též tato metoda nepotřebuje externí krystal pro oscilátor. Nevýhodou je nemožnost programovat mikročip přímo v obvodu.



Obr. 2.6: Sériové programování vysokým napětím – ideové zapojení

2.6.1 Algoritmus sériového programování vysokým napětím

1. Mezi Vcc a GND připojíme +5 V. RESET = 0 V, PB0 = 0 a počkáme alespoň 100 ns. Jestliže propojka RCEN není naprogramována, tak přivedeme na pin XTAL1/PB3 nejméně čtyřikrát za sebou impuls o délce alespoň 100 ns. PB3 = 0 a počkáme alespoň 100 ns. Nebo pokud RCEN je naprogramována, tak PB3 = 0 a počkáme 4 us. V obou případech je nutné mít RESET na +12 V a nejdříve po uplynutí 100 ns můžeme měnit stav PB0. Před vysláním instrukce počkáme 8 us.

2. Flash paměť je programována bajt po bajtu. Nejprve je dodán bajt adresy, poté bajt dolní a horní poloviny dat. Zápis instrukce je samospouštěcí; čeká dokud se na PB2(RDY/BSY) neobjeví vysoká logická úroveň.

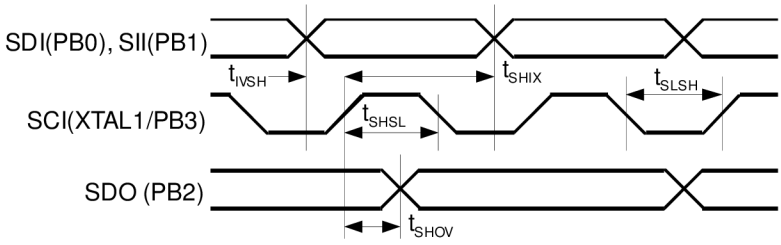
3. EEPROM paměť je programována stejně jako Flash. S tím rozdílem, že je vyslán jen jeden bajt dat.

4. Jakákoliv adresová oblast může být zkontrolována instrukcí čtení, která vrátí obsah dané adresy po sériovém výstupu PB2.

5. PB3 = 0, RESET = 0 a vypneme napájení.

Když zapisujeme nebo čteme sériová data, tak data jsou citlivá na náběžnou hranu hodinového vstupu.

Časové Charakteristiky



Obr. 2.7: Časování sériového programování vysokým napětím (převzato z [14] str. 42)

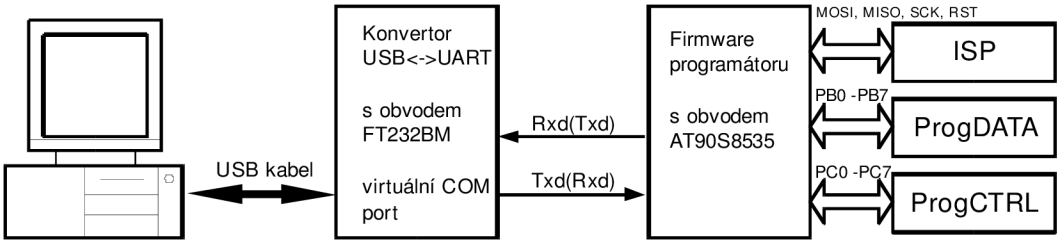
Tab. 2.13: Časové údaje pro sériové programování vysokým napětím

Symbol	Parametr	Min	Typ	Max	Jednotka
t_{SHSL}	SCI (XTAL1/PB3) délka impulsu v log. 1	100			ns
t_{SLSH}	SCI (XTAL1/PB3) délka impulsu v log. 0	100			ns
t_{IVSH}	Délka výdrže SDI, SII před náběhem hrany SCI	50,0			ns
t_{SHIX}	Délka výdrže SDI(PB0), SII(PB1) po náběhu hrany SCI	50,0			ns
t_{SHOV}	Délka výdrže impulsu SCI(XTAL) do změny SDO(PB2)	10,0	16,0	32,0	ns
t_{WLWH_CE}	Doba po vykonání instrukce Smazání čipu	5,0	10,0	15,0	ms
t_{WLWH_PFB}	Doba po vykonání instrukce Zápis propojky	1,0	1,5	1,8	ms

Programování sériově vysokým napětím probíhá posíláním bajtů po čtyřech vodičích(PB0, PB1, PB2, PB3). PB0(Serial Data Input) slouží pro posílání dat do mikropočítače. PB1(Serial Instr. Input) slouží pro vstup řídících instrukcí. PB2(Serial Data Output) slouží jako výstup dat z mikropočítače. A poslední vývod PB3(XTAL1) se používá jako zdroj taktovacích impulsů. Data jsou citlivá na náběžnou hranu hodinového signálu. Instrukce jsou o délce 2 až 4 bajty. Význam jednotlivých bitů příkazů najdeme v katalogovém listu součástky podporující sériové programování vysokým napětím. Typickým příkladem může být Attiny11, která pouze podporuje tento režim programování.

3 Popis softwarové části firmwaru

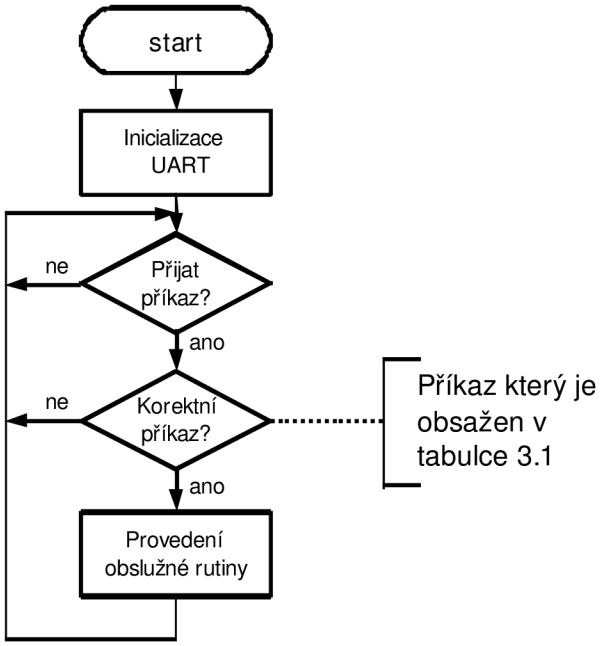
Program uložený v AT90S8535 na svých UART pinech PD0(RxD) a PD1(TxD) neustále naslouchá a očekává příkazy přes převodník(FT232BM) ze sériového portu počítače, na kterém běží obslužná aplikace jako je AVRProg(AVRStudio), CodeVision, AVROSP, Avrdude. Nastavení a hodnota rychlosti přenosu dat sériového portu emulovaná integrovaným obvodem FT232BM je popsána v kapitole 6.2. Pro komunikaci a programování s cílovým mikropočítačem, který chceme programovat, firmware(AT90S8535) využívá obousměrné vstupně/výstupní porty PB a PC. Pro ISP programování jsou využity 4 ISP vývody brány B. Porty PB a PC slouží pro programování vysokým napětím. Popis zapojení portů na konektory ISP, ProgData, ProgCtrl najdeme v kapitole 5.9 a 5.10.



Obr. 3.1: Mikropočítač s firmwarem v blokovém komunikačním schématu

3.1 Komunikační protokol

Komunikační protokol vychází z definice pro AVRProg a z aplikační poznámky Atmelu [9], [10]. Seznam podporovaných příkazů je v tabulce 3.1. Tabulka je převzata z aplikační poznámky [9] ze str. 8. Každý příkaz jdoucí z PC(z AVRProgu) začíná jedním ASCII znakem. Programátor vrací 13d(carriage return) nebo požadovaná data po dokončení příkazu. Na neznámý příkaz programátor odpoví znakem „?“.



Obr. 3.2: Vývojový diagram

Tab. 3.1: Příkazy AVRProgu

Příkaz	Data z PC		Data do PC	
	ID	*data*	*data*	
Vstup do programovacího módu	„P“			13d
Automatická inkrementace adresy	„a“		dd	
Nastavení adresy	„A“	ah al		13d
Zapis paměti programu, nižší bajt	„c“	dd		13d
Zápis paměti programu, vyšší bajt	„C“	dd		13d
Záležitost zápisu stránky	„m“			13d
Čtení paměti programu	„R“		2*dd	
Zápis paměti dat	„D“	dd		13d
Čtení paměti dat	„d“		dd	
Vymazání čipu	„e“			13d
Zápis zámkových bitů	„l“	dd		13d
Čtení propojkových a zámkových bitů	„F“		dd	
Výběr typu AVR součástky	„T“	dd		13d
Opuštění programovací módu	„L“			13d
Čtení bajtů signatury	„s“		3*dd	
Vrátí podporované typy mikroprocesorů	„t“		n*dd	
Vrátí označení programátoru	„S“		s[7]	
Vrátí verzi software	„V“		dd dd	
Vrátí verzi hardware	„v“		dd dd	
Vrátí typ programátoru	„p“		dd	

- Příkaz „P“ musí být vyslán bezprostředně před jakýmkoliv jiným příkazem kromě příkazů „t“, „S“. „V“, „v“, „T“.
- Ah a Al značí vyšší a nižší bajty adresy.
- Dd značí jeden bajt(8bitů).

3.2 Podrobný popis funkce programu

Program neustále běží v tzv. hlavní smyčce, kde čeká na příkaz z PC. Přijme-li korektní příkaz, provede příslušnou obslužnou rutinu a čeká na další příkaz. Zjednodušený vývojový diagram je na obrázku 3.2.

Program začíná inicializací ukazatele zásobníku na konec paměti Flash a nastavením vývodu reset jako výstup do logické jedničky. Dále pak povolením a nastavením sériového kanálu UART na požadovanou přenosovou rychlost 19200 Bd při krystalu oscilátoru 7,3728 Mhz. Část těchto zdrojových kódů je na obrázku 3.3 a 3.4.

```
RESET:
      ldi   temp1,low(RAMEND)
      out   SPL,temp1         ; nastaveni zasobniku
      ldi   temp1,high(RAMEND)
      out   SPH,temp1
      clr   temp1
      out   GIMSK,temp1       ; zakaze externi preruseni
      ser   temp1
      out   PORTD,temp1
      set_reset               ; set RESET=1
      out   PORTB,temp1
      sbi   ddrd,resetpin     ;(RESET) je vystup
      rcall uart              ; inicializace UART
```

Obr. 3.3: Zdrojový kód, který proběhne po resetu

```
;Inicialize UART.
uart:
      ldi   temp1,N                 ; nastavi baudovou rychlost
      out   UBRRL,temp1
      ldi   temp1,1<<TXEN|1<<RXEN ; inicializace UART pro TX a RX
      out   UCSRB,temp1
      ldi   temp1,(1<<URSEL)|(3<<UCSZ0)     ;8data, 1 stop bit
      out   UCSRC,temp1
      ret
```

Obr. 3.4: Inicializace kanálu UART

Po tomto počátečním nastavení začíná hlavní program, který přijme příkaz z PC. Přijmutí příkazu není nic jiného než přečtení prostého znaku ze sériového portu. Obslužná rutina pro příjem a vyslání znaku čeká v nekonečné smyčce tak dlouho dokud není celý uart registr plný nebo vyslaný. Ukázky zdrojových kódů pro vysílání/přijímaní znaků z/do PC jsou na obrázku 3.5.

```
posli_znak:
      sbis  UCSRA,UDRE ; skakej dokud nebude registr TX prazdny?
      rjmp  posli_znak
      out   UDR,data_uart    ; posli bajt na uart
      ret
prijmi_znak:
      sbis  UCSRA,RXC        ; cekej dokud neni prijat znak
      rjmp  prijmi_znak
      in    data_uart,UDR    ; precti z uartu bajt
      ret
```

Obr. 3.5: Rutina vysílající obsah registru uart a rutina provedená při ukončení příjmu

Dále program přijatý příkaz rozpoznává, tak jak je znázorněno v podrobném vývojovém diagramu v příloze F (obr 10.9). Levá část diagramu má spíše informativní charakter pro PC, zjišťuje zde věci jako typ programátoru, podporovaná zařízení programátoru, verze programátoru, název programátoru. Programátor na tyto příkazy z tabulky 3.1 odpoví pomocí obslužných rutin sériové linky UART z obrázku 3.5.

Má-li počítač ověřeny všechny počáteční informace od programátoru je připraven vyslat příkaz 'P', což je „Vstup do programovacího módu“. Obslužný kód příkazu 'P' najdeme ve zdrojovém kódu na adrese programu s návěštím *vstup_do_isp*. Přesný algoritmus pro vstup do programovacího módu byl popsán v kapitole 2.4.2.

Podprogram používaný pro čtení a zápis bitů na sběrnici SPI, chceme-li vývody MISO, MOSI a SCK je na obrázku 3.6. Bajt, který chceme zapisovat na SPI je uložen v registru *spi_data*, který osmkrát rotujeme přes příznakový bit C a zároveň na základě hodnoty bitu C měníme napěťovou úroveň na vývodu MOSI a potvrzujeme impulsem na vývodu SCK, čímž posíláme data registru *spi_data* po sériové lince ISP. Algoritmus je popsán kapitole 2.4.4 a hodinový signál SCK znázorněn na obrázku 2.3. Čtení z SPI se provádí v cyklu posuvem vynulovaného registru *precti_spi_data* o jeden bit vlevo a testováním vývodu MISO na logickou jedničku. Je-li vývod MISO v logické jedničce, nejméně významný bit v registru *precti_spi_data* se nastaví, jinak zůstane nulový. Po osmi průchodech cyklem máme v registru přečtený celý bajt.

```
precti:
      clr   spi_data
zapis:
      ldi   temp1,8
      ldi   precti_spi_data,0
crs0:
      rol   spi_data
      brcc  crs1
      sbi   portb,MOSI
      rjmp  crs2
crs1:
      cbi   portb,MOSI
crs2:
      lsl   precti_spi_data
      sbic  pinb,MISO
      ori   precti_spi_data,1
      impulsSCK
      dec   temp1
      brne  crs0
      mov   spi_data,precti_spi_data
      ret
```

Obr. 3.6: Podprogram pro zápis a čtení z ISP rozhraní programovaného mikropočítače

Tento podprogram spolu s podprogramy pro UART komunikaci jsou stěžejní a často používané při vykonávání programovacích algoritmů jako jsou:

- čtení a zápis paměti programu *(zdrojový kód je na adrese s návěštími c8, c9, c12)*
- čtení a zápis paměti dat *(zdrojový kód je na adrese s návěštími c11, c13)*
- načítání adresy *(zdrojový kód je na adrese s návěštím c10)*
- zápis zámkových bitů *(zdrojový kód je na adrese s návěštím c16)*
- čtení signatury *(zdrojový kód je na adrese s návěštím c17)*
- mazání čipu. *(zdrojový kód je na adrese s návěštím c15)*

Popis algoritmů pro tyto ISP rutiny byl popsán v kapitole 2.4.3.

Kompletní zdrojový kód firmwaru najdeme v příloze H: CDROM.

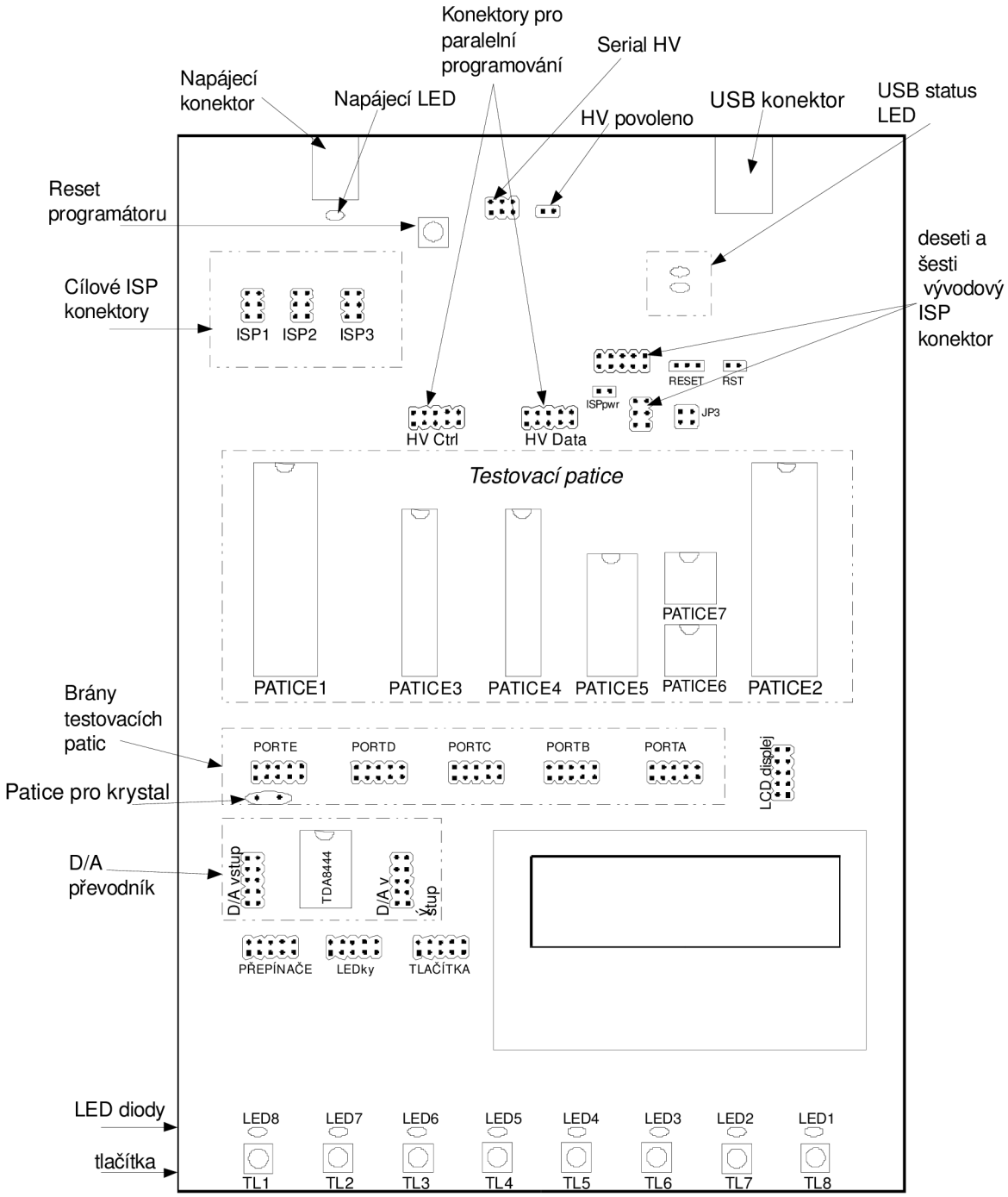
4 Konstrukce univerzálního AVR programátoru

Programátor se připojuje do USB portu počítače. Jako konvertor USB<->TTL je použit integrovaný obvod FT232BM [16] od firmy FTDI Chip. Programátor používá vnější napájecí zdroj +5 a +12 V(není napájený ze sběrnice USB), řešený pomocí stabilizátorů 78xx. Tímto se snižuje riziko poškození USB portu počítače nevhodnou manipulací. Vlastní programování cílového mikrokontroléru, uloženého v příslušné patici, je zajištěno firmwarem uloženým v ATmega8535(IO1). Firmware používá pro komunikace s USB konvertorem asynchronní sériový kanál. Pro získání programovacího napětí pro RESET, při paralelním programování je použit externí zdroj 12 V, řešený stabilizátorem 7812. Přepínání programového napětí(0 V nebo 5 V nebo 12 V) na RESETu cílového mikrokontroléru je řešeno elektronicky vývody PD2, PD3 IO1 a spínacími tranzistory T1 až T3. Dioda D1 zabraňuje galvanickému spojení mezi zdroji různých potenciálů. JMP1 slouží pro výběr režimu RESETu: Pro běžný provoz jsou zkratovány piny 2 a 3. Pro aktualizaci firmwaru ATmega8535 externím ISP programátorem musí být jumper na propojkách 1 a 2. K resetování mikropočítače IO1 je k dispozici tlačítko TL1.

Pinové konektory SV1 až SV5 se používají k propojení mikrokontroléru firmwaru s programovaným cílovým mikrokontrolérem umístěným v příslušné patici – viz druhá část schématu na obrázku 10.2. V třetí části schématu (obr. 10.3) jsou konektory jednotlivých bran programovaného mikrokontroléru umožňující testovaní a vývoj připojením cílového mikrokontroléru k poli LED diod, k poli spínačů, k poli tlačítek, k DA převodníku, LCD displeji. Každá tato testovací periferie se může připojit k libovolné bráně k mikrokontroléru, umístěném v příslušné patici, pomocí externího kabelu. Pole svítivých diod je, je od brány odděleno budičem typu 74HCT245. V daném zapojení přecházejí vstupy A0 až A7(po proudovém zesílení) na výstupy B0 až B7, zde jsou zapojeny LED. LED svítí při logické nule. Pokud místo obvodu 74HCT245 použijeme vývojově shodný obvod 74HCT640(obsahuje invertory), bude LED svítit při log. 1. Čtení informací z vnějších zařízení je zajištěno polem spínačů DIP a osmi mikrotlačítky. LCD displej je typu MC1602E-SYL jehož cena je v GM asi 200Kč. Displej je řízen řadičem od firmy Hitachi HD4478. Poslední modul obsahuje 8násobný 6bitový D/A převodník řešený obvodem TDA8444, který disponuje sběrnicí I2C.

5 Popis hardware

Programátor je řešen na jedné dvoustranné desce s prokovenými otvory. Jeho součástí je USB konektor typu A k připojení do PC a tak je nutné programátor spojit s počítačem propojovacím kabelem. Dále pak konektor pro napájení k připojení usměrněného adaptéru 15 V. Řídící část, testovací patice i uživatelské prvky jsou na jedné desce plošného spoje. Většina součástek je řešena technologií SMD.



Obr. 5.1: Komponenty kompletní konstrukce

5.1 Napájecí zdroj

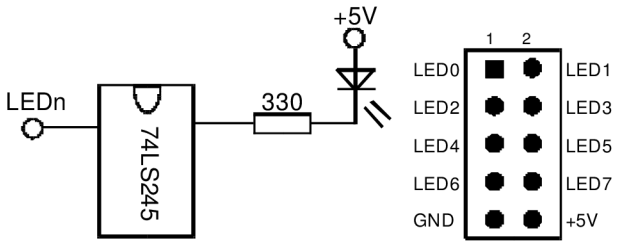
Programátor vyžaduje externí usměrněný síťový napájecí adaptér o napětí 15 V. Nebudeme-li používat paralelní metodu programování a DA převodník, postačí napájecí adaptér o hodnotě 7 V až 15 V(vyhoví i 9V baterie). O přítomnosti připojení napájecího napětí informuje zelená svítivá dioda LED5.

5.2 USB konektor

Programátor se připojuje s osobním počítačem přes port USB. Použit je kabel s koncovkou typu A-B USB2.0(1.5/12/480MBps). Podporován je i starší standart USB1.1.

5.3 Pole svítivých diod

Programátor obsahuje 8 červených svítivých diod LED, svítící při nízké logické úrovni. Pole LED jsou připojeny na LED konektor odděleně od zbytku desky. Můžeme je spojit deseti žilovým propojovacím kabelem s konektorem portu u testovacích patic.



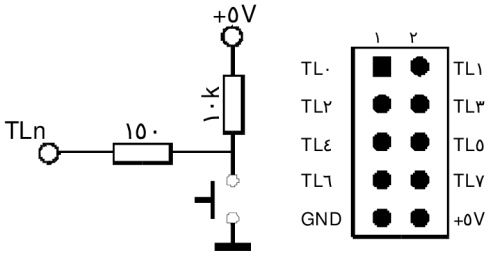
Obr. 5.2: Zapojení pole LED

Přípustná vstupní hodnota pro log. 0: 0 až 0,8V.

Přípustná vstupní hodnota pro log. 1: 2 až 5V.

5.4 Pole tlačítek

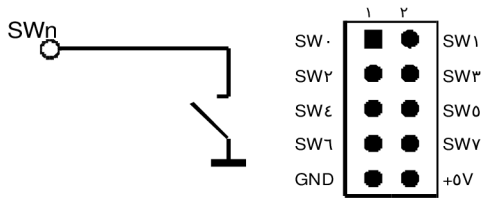
Způsob připojení osmi tlačítek na TL konektor je znázorněn na obrázku 5.3.



Obr. 5.3: Zapojení konektoru tlačítek

5.5 Pole přepínačů

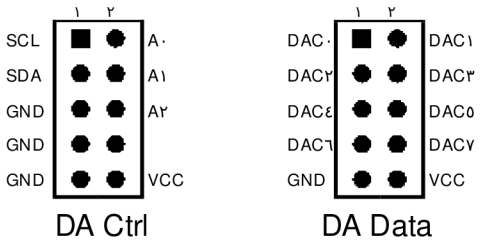
Použit je miniaturní přepínač DIP s osmi přepínacími kontakty (obr. 5.4).



Obr. 5.4: Zapojení konektoru přepínačů

5.6 DA převodník

Integrovaný obvod TDA8444 je 8kanálový 6bitový DA převodník přijímající číslicová data sériově přes sběrnici I2C. Převodník je 6bitový, pracuje tedy s čísly 0 až 63. Vstupy A_3, A_2, A_1 slouží pro adresování obvodu. Bližší informace viz katalogový list převodníku. Analogové výstupy DA převodníku jsou k dispozici na vývodech D_7 až D_0. Zapojení vývodů z převodníku a popis konektorů je na obrázku 5.5.



Obr. 5.5: Zapojení konektorů DA převodníku

Výstupní napětí každého DA převodníku je určeno vztahem:

$$U_{výst}(N) = \frac{N}{64} \cdot \frac{U_{výst}(٦٣) - U_{výst}(0)}{Ucc - 2} U_{MAX} + U_{výst}(٠) \qquad (5.1)$$

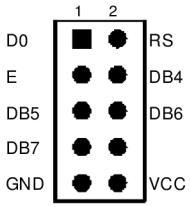
Pokud $U_{max} = U_{cc}$, $U_{výst}(63)$ je maximální výstupní napětí a $U_{výst}(0)$ je minimální výstupní napětí. Jelikož je převodník 6bitový, pracuje s čísly N = 0 až 63.

- SDA datový vstup/výstup sběrnice I2C
- SCL hodinový signál I2C

5.7 LCD displej

V dané konstrukci programátoru se používá levný dvouřádkový displej se 16 znaky typu MC1602E-SYL. Displej je řízen běžným řadičem HD44780 od firmy Hitachi.

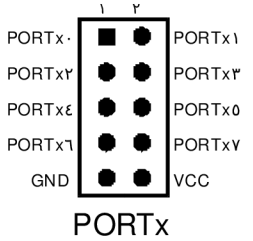
Vývod D_0 ovládá podsvícení displeje. Podsvícení zapnuto pro D_0=0 a odpojeno pro D_0=1.



Obr. 5.6: Zapojení konektoru LCD displej

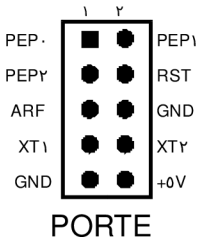
5.8 Vstupně/výstupní porty testovacích patic

Vstupně/výstupní brány testovacích patic jsou vyvedeny na konektory PORTA, PORTB, PORTC, PORTD, PORTE. Tyto porty jsou určeny pro propojování deseti žilovým kabelem s některým uživatelským prvkem jako je pole LED, LCD displej a jiné, viz výše.



Obr. 5.7: Zapojení portu PORTA až PORTD

PORTE má některé speciální přídavné funkce používané při programování vysokým napětím. Zapojení portu je na obrázku 5.8.

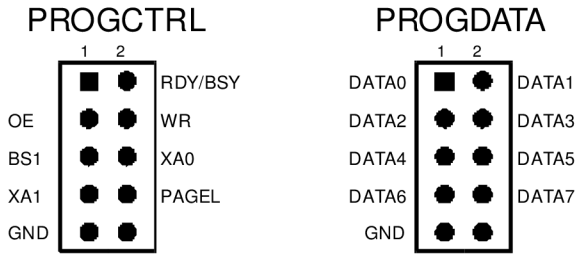


Obr. 5.8: Zapojení portu PORTE

- **PEP0 až PEP2** – vstupně výstupní brány nebo u některých mikrokontrolérů(ATmega161, AT90S4414, AT90S8515) alternativní funkce ICP/INT2, ALE, OC1B.
- **XT1** – Vnitřní hodinový signál pro všechny testovací patice.
- **XT2** – S pinem XT1 může sloužit pro připojení krystalu jako externí hodinový zdroj.
- **RST** – Vývod resetu všech testovacích patic

5.9 Konektory ProgCtrl a ProgData

Konektory ProgCtrl a ProgData slouží pro paralelní programování vysokým napětím. Význam jednotlivých pinů je na obrázku. Během paralelního programování se spojuje plochým deseti žilovým kabelem konektor ProgCtrl s PortemD a ProgData s PortemB cílové patice.



Obr. 5.9: Popis pinů portů ProgCtrl a ProgData

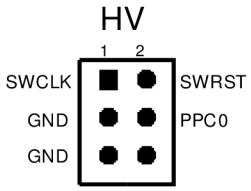
5.10 Konektory ISP6 a ISP10

Tyto konektory mohou sloužit pro ISP programování v externí aplikaci. ISP6 je dále je nutný při programování v testovací patici kvůli správnému propojení pinů(MOSI, MISO, SCK) šesti žilovým plochým propojovacím kabelem na konektory ISP1, ISP2, ISP3 u testovacích patic. Zapojení pinů ISP6 a ISP10 je podle standardu firmy Atmel.

Obr. 5.10: Zapojení vývodů konektorů ISP

5.11 Konektor HV

Konektor slouží pro sériové programování vysokým napětím. Způsob použití je popsán v kapitole 6.6.



Obr. 5.11: Popis vývodů konektoru HV

5.12 Popis testovacích patic

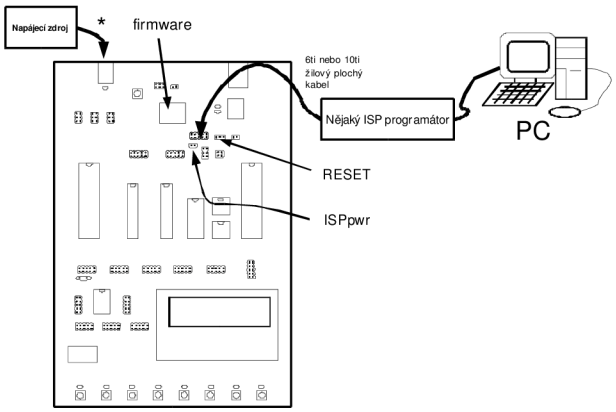
Konstrukce programátoru obsahuje sedm patic vprostřed desky s označením PATICE1 až PATICE7. Viz obrázek 5.1. Tyto patice slouží pro vložení mikroprocesoru k programování nebo pro testování aplikace. Který typ mikropočítače přijde do jaké patice je popsáno v následující kapitole.

6 Oživení a příručka programátora

V této kapitole je vysvětleno jak pomocí AVRStudia do programátoru nahrát obslužný program a jak propojit a nainstalovat programátor s počítačem. Dále je zde nezbytná příručka o propojení konektorů, umístění součástky do správné patice, nastavení jumperů při programování cílových mikropočítačů.

6.1 Počáteční instalace

K oživení programátoru je nutné nahrát program do řídící jednotky(IO3 - AT90S8535) programátoru, protože tento obvod je na počátku prázdný. Firmware je dostupný na CDROM v příloze. Je potřeba vlastnit nějaký jiný externí ISP programátor.



Obr. 6.1: Zapojení pro aktualizaci firmware

Pracovní postup

1. Propojíme šesti nebo deseti žilovým kabelem nějaký cizí ISP programátor s našim AVR programátorem na konektor ISP6 nebo ISP10.
2. Připojíme jumper RESET(JP1) s piny 1 a 2. Připojíme jumper na ISPpwr(JP5)
3. * Nemá-li cizí ISP programátor napájení pro cílovou desku, odpojíme ISPpwr(JP5) a připojíme síťový napájecí zdroj.
4. Na PC spustíme spusť příslušný program. Například AVRStudio a v nabídce zvolíme Tools/AVRProg.
5. Flashneme firmware. V nastavení propojek před programováním je nutné zvolit: Externí zdroj hodinového signálu.
6. Odpojíme napájení a poté propojovací kabel.
7. Vrátíme jumper RESET(JP2) do výchozí pozice na piny 2 a 3.

6.2 Instalace ovladačů programátoru v PC

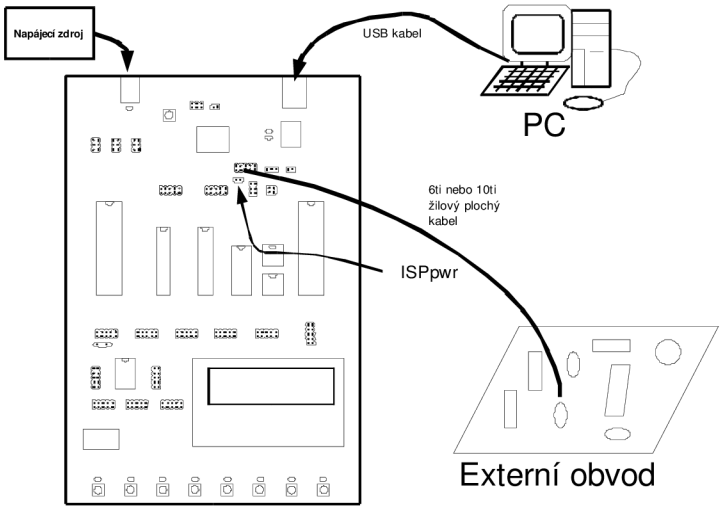
Daná konstrukce je zrealizovaná pro komunikaci s PC přes port USB. Jako převodník je zde použit integrovaný obvod FT232BM od firmy FTDI [16]. Využívá se ovladače VCP (virtuální sériový port), dostupný v [18]. Návod k instalaci ovladače je v [17]. Pokud máme v PC k dispozici Windows XP SP2 a novější, nepotřebujeme ovladače. Všechny nezbytné ovladače jsou součástí tohoto operačního systému a po připojení USB zařízení do PC se automaticky spustí standardní instalační průvodce nového hardware.

Po nainstalování se v OS objeví nový sériový port COM.

Programátor komunikuje s těmito parametry:

- **Přenosová rychlost**: 19200Bd
- **Počet datových bitů:** 8
- **Parita:** žádná
- **Počet stop bitů:** 1
- **Řízení toku:** žádné

6.3 ISP programování v externím obvodu



Obr. 6.2: Zapojení pro ISP programování v externím obvodu

Pracovní postup

1. Má-li cílová deska vlastní zapnuté napájení, odpojíme ISPpwr(JP5), jinak necháme připojené.
2. Připojíme PC, napájecí zdroj, z ISP6 nebo ISP10 konektoru propoj plochým kabelem cílový mikropočítač v externí desce.
3. Programujeme.
4. Odpojíme propojovací kabely a připojíme jumper ISPpwr(JP5) do výchozí podoby.

6.4 ISP programování v testovací patici

Protože v různých mikrokontrolérech je různé rozmístění pinů pro ISP programování(MISO, MOSI, SCK,RST), jsou zde tři konektory ISP1 až ISP3 sloužící ke správnému připojení programovacích vodičů k patici. Viz obrázek 6.3. K propojení se používá šesti žilový plochý kabel. Správné propojení ISP konektoru a umístění mikropočítače do patice je vysvětleno v tabulce 6.1.

Podporované typy:

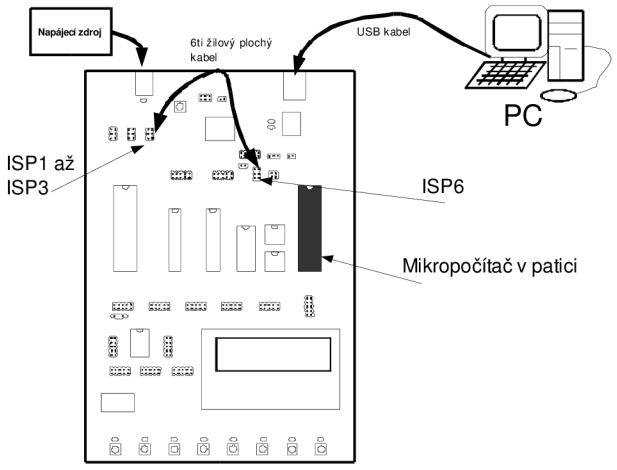
AT90S1200, AT90S2313, AT90S2323, AT90S2333, AT90S4414, AT90S4433, At90S4434, AT90S8515, AT90S8535.

ATmega8, ATmega16, ATmega161, ATmega163, ATmega323, ATmega8515, ATmega8535

Attiny12, Attiny15, Attiny22

Tab. 6.1: Umístění v patici pro ISP programování

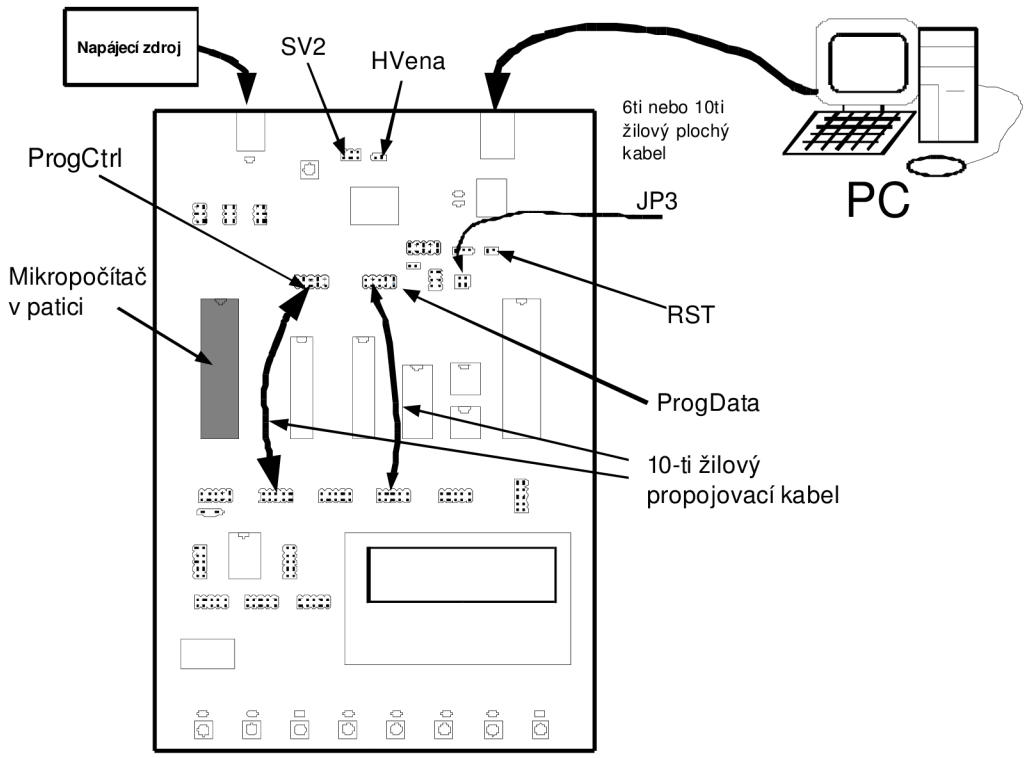
Typ mikropočítače	patice	Cílový ISP konektor
AT90S1200	PATICE5	ISP3
AT90S2313		
AT90S2323	PATICE6	ISP1 propoj drátem: RST(PORTE) – PB5P(PORTB) mikropočítač musí být taktován vnitřním RC oscilátorem
AT90S2343		
ATtiny12		
ATtiny22		
AT90S4434	PATICE2	ISP3
AT90S8535		
ATmega8515		
ATmega16		
ATmega163		
ATmega323		
AT90S4414	PATICE1	ISP3
AT90S8515		
ATmega161		
AT90S2333	PATICE4	ISP2
AT90S4433		
ATmega8		
ATtiny15	PATICE7	ISP1 propoj drátem: RST(PORTE) – PB5P(PORTB)



Obr. 6.3: Zapojení pro ISP programování v testovací patici

6.5 Paralelní programování vysokým napětím

Paralelní programování vyžaduje napájecí zdroj 15V, abychom dosáhli 12V pro RESET. Pro použití paralelního programování umístíme součástku do dané patice, propojíme příslušné piny deseti žilovými propojovacími kabely. Konektor „ProgData“ se propojí s „Portem B“ a konektor „Prog Ctrl“ se propojí s „Portem D“, tak jako ukazuje obrázek 6.4.



Obr. 6.4: Paralelní programování

Pracovní postup

1. Umístíme součástku do příslušné patice. Viz tabulka 6.2.
2. Deseti žilovým plochým propojovacím kabelem propojíme PROGDATA a PORTB
3. Deseti žilovým plochým propojovacím kabelem propojíme PROGCTRL a PORTD
4. Propojíme vodičem SWCLK(SV2) s XT1(PORTE)
5. Připojíme jumper RST(JP4) a HVena(JP2)
6. Programujeme-li AT90S2333, AT90S4433 nebo ATmega8 připoj oba jumpery na JP3.
7. Programujeme-li ATmega16, ATmega163, ATmega161, ATmega128 nebo ATmega323 propojíme vodičem PPC0(SV2) s PAP0(PORTA).
8. Odpojíme propojovací kabely, jumpery.

Tab. 6.2: Umístění v patici pro paralelní programování

Typ mikropočítače	**patice**
AT90S1200	PATICE5
AT90S2313	
AT90S4434	PATICE1
AT90S8515	
ATmega161	
AT90S4434	PATICE2
AT90S8535	
ATmega16	
ATmega163	
ATmega323	
AT90S2333	PATICE4
AT90S4433	
ATmega8	

6.6 Sériové programování vysokým napětím

Pracovní postup

1. Umístíme mikropočítač do příslušné patice dle tabulky.
2. Propojíme vodičem SWCLK(SV2) s XT1(PORTE).
3. Připojíme jumper RST.
4. Vodičem propojíme PB3(PORTB) s XT1(PORTE).
5. Vodičem propojíme PB5(PORTB) s RST(PORTE).
6. Vodiči propojíme piny PB0 a PB1 z konektoru ISP1 na DATA0 a DATA2 v konektoru PROGDATA.
7. Propojíme vodičem PB1(ISP1) s DATA1(PROGDATA).
8. Nyní jsme připraveni programovat.

Tab. 6.3: Umístění v patici pro sériové programování vysokým napětím

Typ mikropočítače	**Patice**
AT90S2323	PATICE6
AT90S2343	
ATtiny11	
ATtiny12	
ATtiny22	
ATtiny15	PATICE7

6.7 Řešení problémů a poruch během používání

- AVRStudio při pokusu spustit AvrProg skončí hláškou "**No supported board found! AVRProg version 1.4**", přitom ostatní programy AVROSP, CodeVision fungují.

Chyba může být způsobena, že programátor je připojen na port vyšší jak COM4. Obsluha programátoru založená na AVRProg má tu vlastnost, že detekuje zařízení pouze na portech COM1 až COM4. **Řešení pro Windows XP:** Změnit číslo sériového portu. Tento počítač->Vlastnosti-> Hardware-> Správce zařízení-> Vlastnosti Porty (COM a LPT)-> *vybereme naše připojené zařízení na nevhodném portu*-> Nastavení portu->Upřesnit->Číslo portu COM *a zde vybereme nějaké volné číslo portu COM v rozsahu 1 až 4*.

7 Závěr

Tento semestrální projekt se zabývá problematikou programování, kontrolou a zamykáním jednočipových mikrokontrolérů. Dále pak návrhem a realizací programátoru komunikujícího s programem AVR-Studio, CodeVision, AVROSP a Avrdude pomocí sběrnice USB.

Na počátku práce byly popsány metody programování pamětí a popis jejich algoritmů. Z úvodní studie vyplývá, že nejjednodušší a nejpoužívanější metodou je sériové programování přímo v systému, odborníky označovaná jako ISP. Mezi největší výhodu ISP, patří možnost programování cílového mikrokontroléru přímo v dané aplikaci. K propojení ISP programátoru a cílové aplikace stačí pouze šesti vodičové metalické spojení. Dané spojení obsahuje jak řídící signály tak napájecí, tudíž cílový mikrokontrolér může být napájen přímo programátorem. Méně často se pak používá zastaralejší(paralelní) metoda vysokým napětím, která je konstrukčně složitější, ale oproti metodě ISP umožňuje programovat propojky.

Programovací propojky, nebo-li také fuse bits, slouží k nastavení procesoru, jeho zdroje hodinového signálu, ochran proti přepisu a jiných vlastností. Špatným nastavením těchto propojek můžeme uvést procesor do již nepoužitelného stavu, z kterého se lze zotavit pouze přeprogramováním vysokým napětím. Zámkové bity slouží k nastavení přístupových práv do paměti. To je výhodné pro komerční aplikace, u kterých požadujeme ochranu proti neoprávněnému přístupu, tzn. proti krádeži softwaru.

Po nastudování výše zmíněné problematiky bylo v návrhovém prostředí Eagle vytvořeno elektrické schéma zapojení programátoru využívající oba dva způsoby programování, ISP i programování vysokým napětím. Konstrukce dále obsahuje desku plošného spoje a seznam součástek pro fyzickou realizaci univerzálního AVR programátoru, připojujícího se do PC přes USB rozhraní. Součástí programátoru jsou i doplňkové obvody(LCD displej, pole svítivých diod, pole spínačů, pole tlačítek, DA převodník) určené pro návrháře, umožňující vývoj a testování programu mikropočítače přímo v programátoru.

V souladu se zadáním je mnou navržená konstrukce programátoru koncipována pro programování metodou ISP i metodou vysokým napětím. Daný programátor podporuje téměř všechny typy a velikosti AVR mikropočítačů, což je zajištěno velkým množstvím patic na programátoru. Patice jsou pro čtyřiceti vývodové součástky řešeny paticemi s páčkou(nehrozí riziko ohnutí nožiček při vyjímání součástky z patice) a zbylé patice pro menší typy mikropočítačů jsou tzv. precizní(s pozlacenými kontakty).

Postupně byly všechny součástky osazeny a připájeny, všechny bloky konstrukce úspěšně oživeny, dále pak byly napsány doplňkové programy pro ověření komunikace s osobním počítačem. Pro sledování provozu jsem používal nástroje na sledování komunikace sériového portu jako je například freewarový program ComSpy. Na základě poznatků z tohoto testování byl postupně sestavován obslužný program pro ATmega8535, umožňující programátor použít pro nejvíce rozšířenou metodou programování ISP. Programátor s tímto firmwarem byl úspěšně vyzkoušen na těchto programech: AVRStudio, CodeVision, AVROSP a Avrdude, který je dokonce k dispozici i pro OS Linux. Všechny tyto programy podporují ISP programování komunikačního protokolu AVR910, na kterém je můj programátor založen. Z důvodů velké časové náročnosti této práce, algoritmus programováním vysokým napětím není implementován do obslužného programu mikropočítače.

Navržená konstrukce tvoří dohromady výkonný, uživatelsky příjemný, jednoduchý a cenově i konstrukčně nenáročný nástroj pro programování mikrokontrolérů Atmel AVR vhodný jak pro techniky, kteří tvoří profesionálně složité komerční přístroje, tak i pro studenty, kteří se s problematikou AVR dosud nikdy nesetkali.

Literatura

[1] MATOUŠEK, David. *Práce s mikrokontroléry ATMEL AVR*. 2. vyd. Praha : BEN – technická literatura, 2006. 376 s., CD ROM. ISBN 80-7300-209-4.

[2] VÁŇA, Vladimír. *Mikrokontroléry AVR : popis procesoru a instrukční soubor*. 1. vyd. Praha : BEN - technická literatura, 2003. 336 s. ISBN 80-7300-083-0.

[3] MATOUŠEK, David. *USB prakticky s obvody FTDI - 1. díl*. Praha : BEN - technická literatura, 2008. 272 s., CD ROM. ISBN 80-7300-103-9.

[4] MATOUŠEK, David. *Práce s mikrokontroléry ATMEL AT89C2051*. 2. vyd. Praha : BEN - technická literatura, 2002. 264 s., CD ROM. ISBN 80-7300-094-6.

[5] PLÍVA, Zdeněk. *EAGLE prakticky - řešení problémů při běžné práci*. 1. vyd. Praha : BEN - technická literatura, 2007. 184 s. ISBN 978-80-7300-227-5.

[6] JURÁNEK, Antonín, HRABOVSKÝ, Miroslav. *EAGLE - uživatelská a referenční příručka : návrhový systém systém pro plošné spoje pro začátečníky*. 2. vyd. Praha : BEN - technická literatura, 2007. 192 s. ISBN 80-7300-213-2.

[7] GM Electronic. [online]. *Katalog elektronických součástek firmy GM.* [cit. 2008-12-15]. Dostupný z WWW: <http://www.gme.cz>.

[8] Atmel Corporation*: Industry Leader in the Design and Manufacture of Advanced Semiconductors* [online]. 1995 [cit. 2008-12-15]. Dostupný z WWW: <http://www.atmel.com>.

[9] AVR109 : Self-Programming. *Application note* [online]. 2004 [cit. 2009-03-12]. Dostupný z WWW: <http://www.atmel.com/dyn/resources/prod_documents/doc1644.pdf>.

[10] AVR910 : In system programming. *Application note* [online]. 2008 [cit. 2009-05-18]. Dostupný z WWW: <http://www.atmel.com/dyn/resources/prod_documents/DOC0943.PDF>.

[11] AVR061 : STK500 Communication Protocol. *Application note* [online]. 2003 [cit. 2009-05-18]. Dostupný z WWW: <http://www.atmel.com/dyn/resources/prod_documents/doc2525.pdf>.

[12] AT90S2313 : 8-bit AVR Microcontroller with 2K Bytes of In-System Programmable Flash. Datasheet [online]. 2002 [cit. 2009-05-18]. Dostupný z WWW: <http://www.atmel.com/dyn/resources/prod_documents/doc0839.pdf>

[13] ATmega16 : 8-bit AVR Microcontroller with 16K Bytes of In-System Programmable Flash. Datasheet [online]. 2002 [cit. 2009-05-18]. Dostupný z WWW: <www.atmel.com/dyn/resources/prod_documents/doc2466.pdf>

[14] AT90S2323 : 8-bit AVR Microcontroller with 2K Bytes Flash. Datasheet [online]. 2002 [cit. 2009-05-18]. Dostupný z WWW: <http://www.atmel.com/dyn/resources/prod_documents/doc1004.pdf>

[15] Katalogové listy k AVR mikropočítačům

[16] FT232BM : jednočipový převodník USB <-> UART. Datasheet [online]. 2005 [cit. 2009-05-21]. Dostupný z WWW: http://www.ftdichip.com/Documents/DataSheets/DS_FT232BM.pdf

[17] FTDI Installation Guides [EN] (instalační příručka ovladače FTDI Chip). Dostupný z WWW: <http://www.ftdichip.com/Documents/ProgramGuides/D2XXPG31.pdf>

[18] Virtual Com Port drivers [EN] (ovladače pro FTDIChip). Dostupný z WWW: <http://www.ftdichip.com/Drivers/CDM/CDM%202.04.16%20WHQL%20Certified.zip>

Seznam použitých zkratek

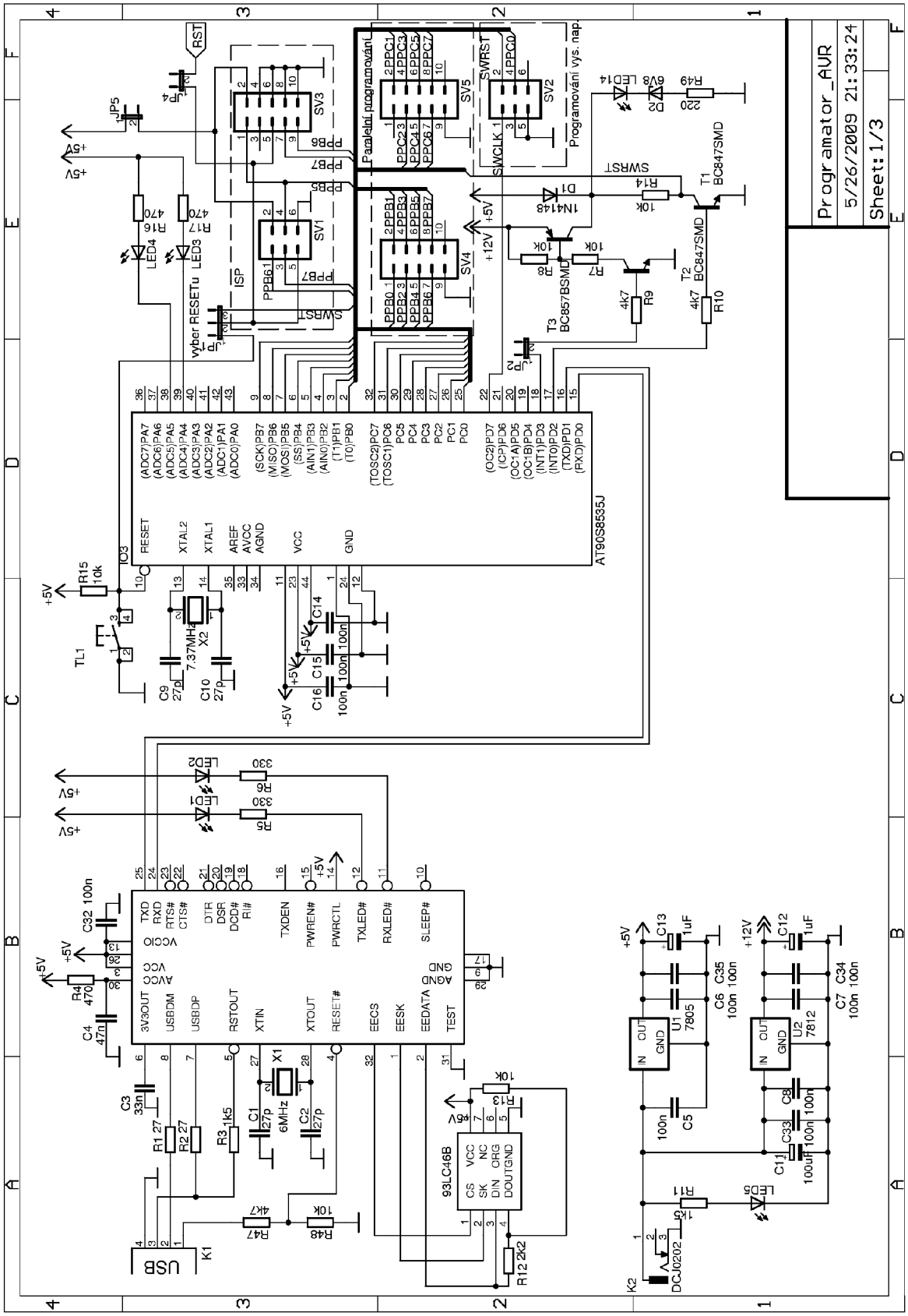
Zkratka	Anglický význam	Český význam
AVR	Alf Vegard RISC	Alf Vegard RISC
EEPROM	Electronically Erasable Programmable Read-Only Memory	elektricky mazatelná a programovatelná paměť typu ROM-RAM
ISP	In system Programming	Programování v systému
LCD	Liquid Crystal Display	Displej z tekutých krystalů
LED	Light Emitting Diode	Světlem zářící dioda
MCU	Micro Controller unit	mikrokontrolér
PC	Personal Computer	Osobní počítač
RAM	Random access memory	Paměť s libovolným přístupem
RISC	Reduced Instruction Set Computer	Počítač s redukovanou instrukční sadou
SPI	Serial peripheal interface	Sériové periferní rozhraní
USB	Universal seriál bus	Univerzální sériová sběrnice

Seznam příloh

V následujících přílohách jsou uvedena schéma zapojení, desky plošných spojů, blokové schéma zapojení a soupiska součástek. Dále zde je CD-ROM obsahující elektronický text bakalářské práce ve formátu pdf, elektronické soubory schématu a desky plošného spoje ve formátu Eagle, zdrojový i binární kód programu pro ATmega8535 a fotografie přístroje.

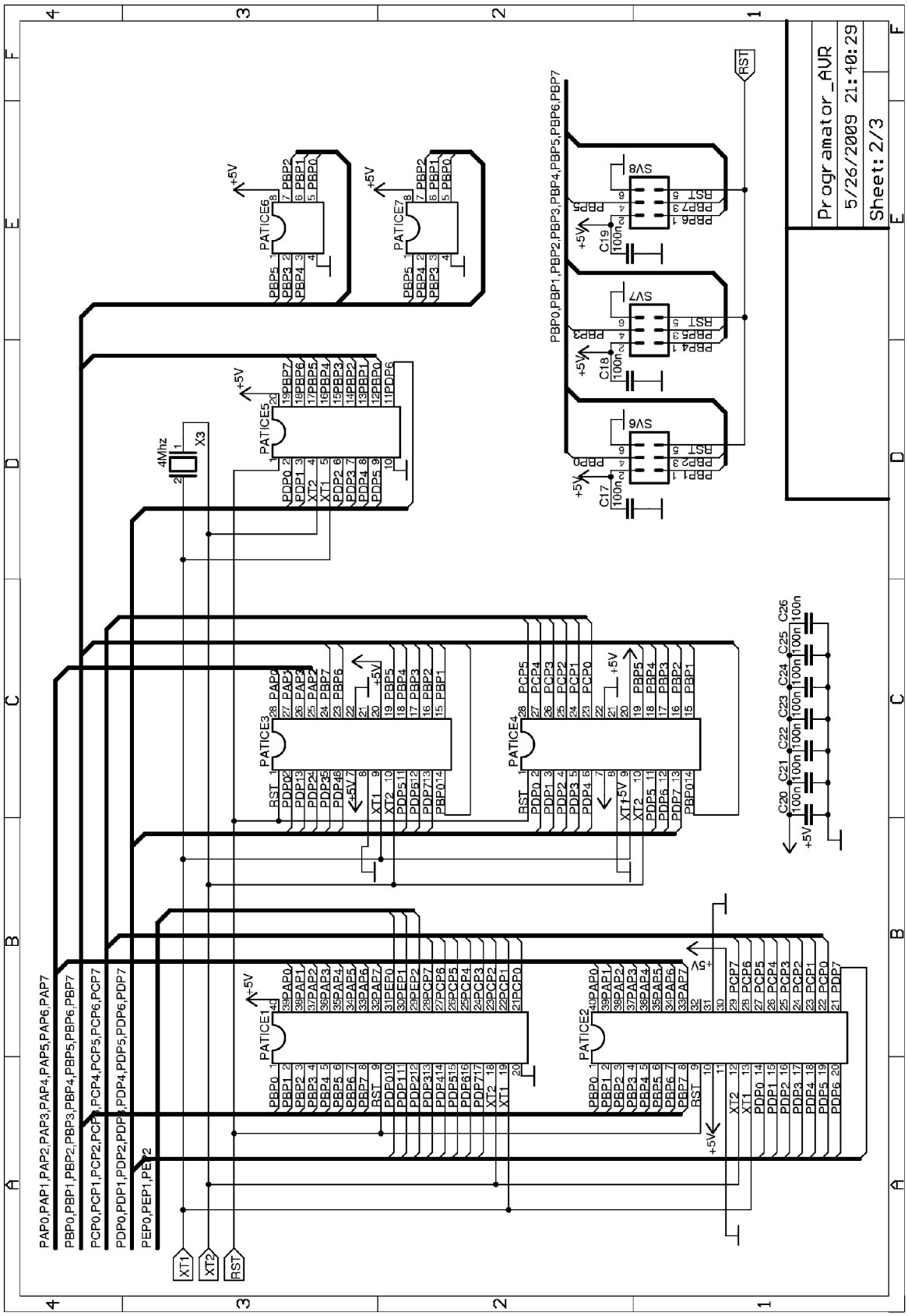
Příloha A: Elektrické schéma zapojení

A.1 První část ze tří



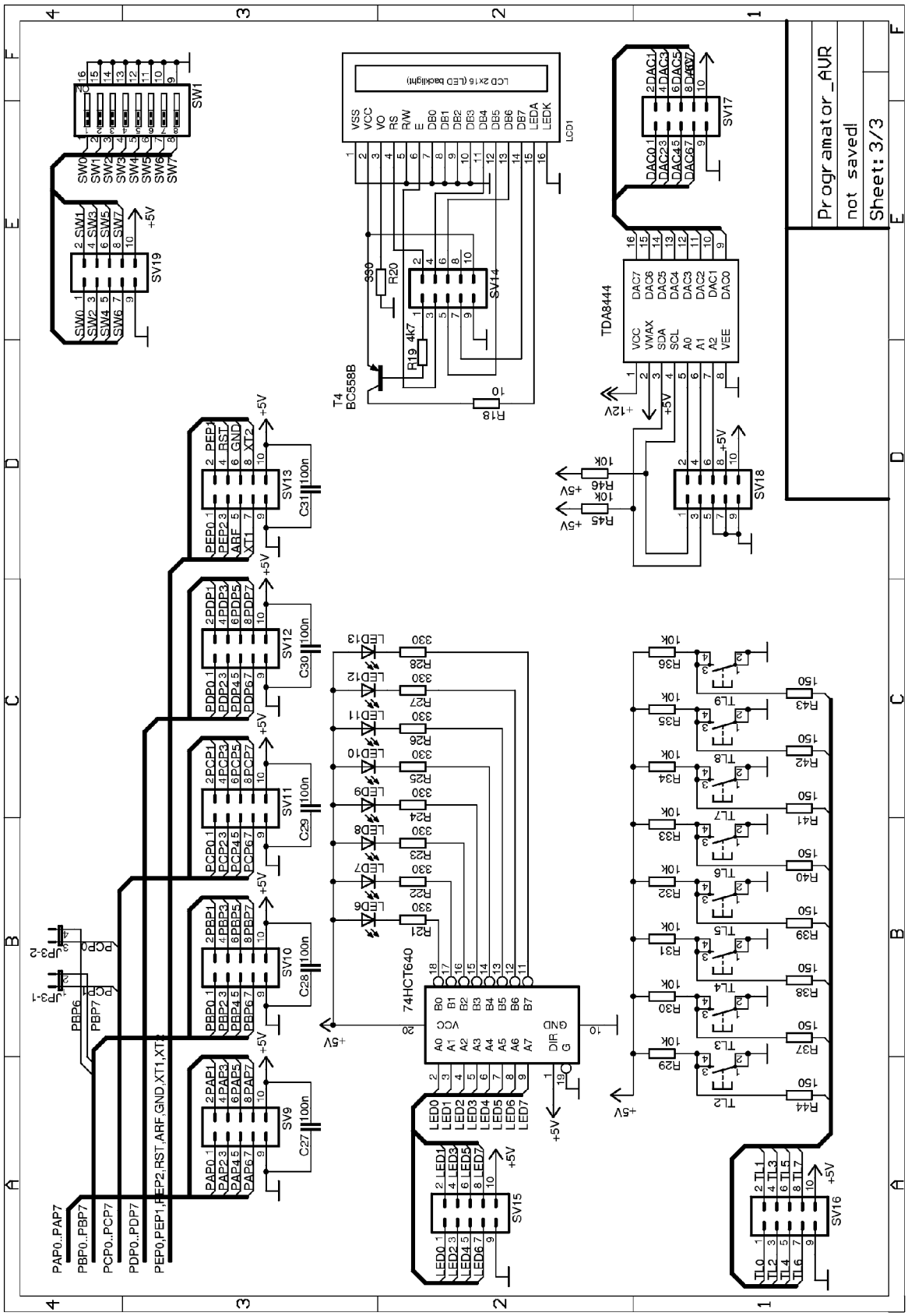
Obr. 10.1: Schéma zapojení 1. ze 3.

A.1 Druhá část ze tří



Obr. 10.2: Schéma zapojení 2. ze 3.

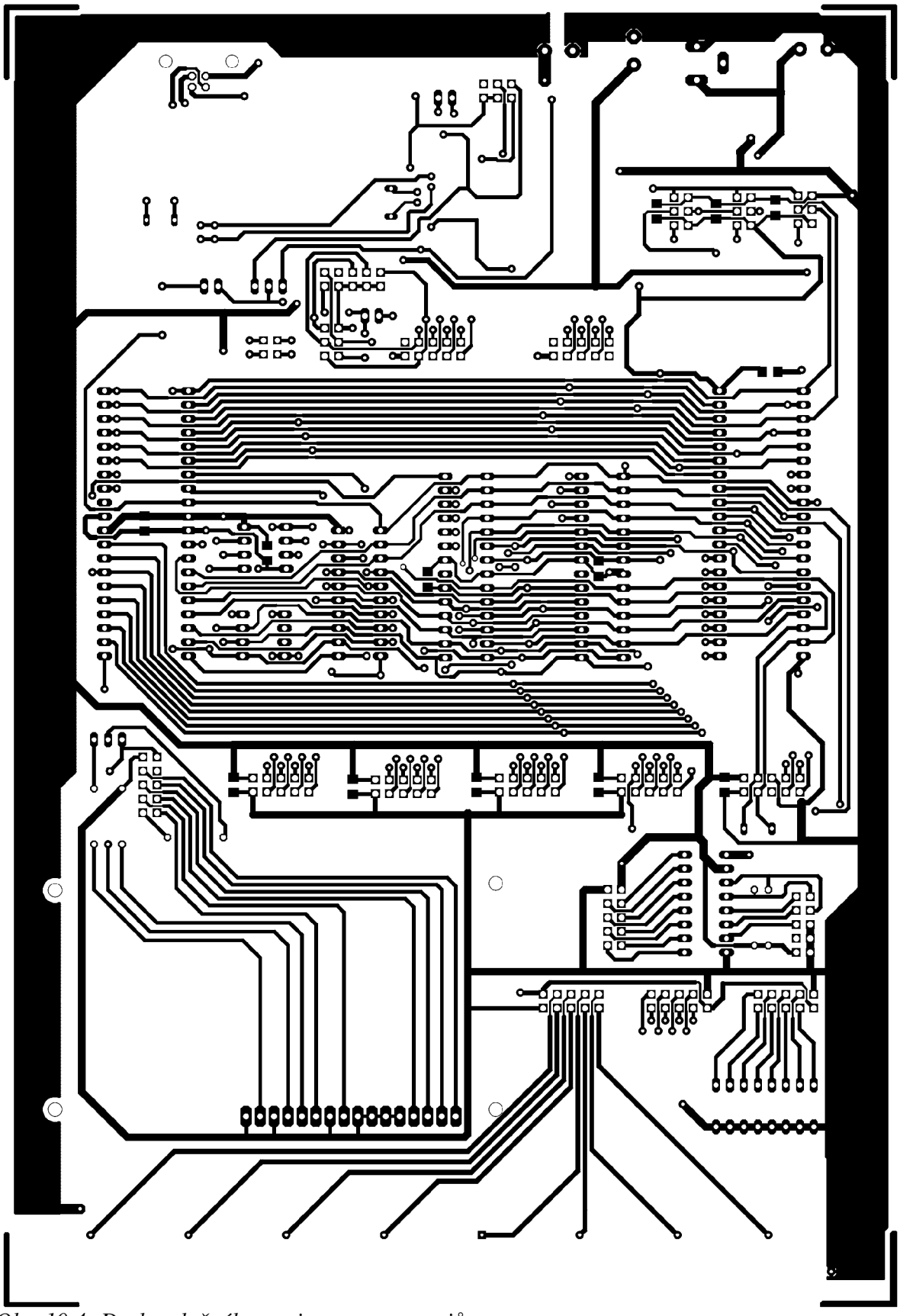
A.1 Třetí část ze tří



Obr. 10.3: Schéma zapojení 3. ze 3.

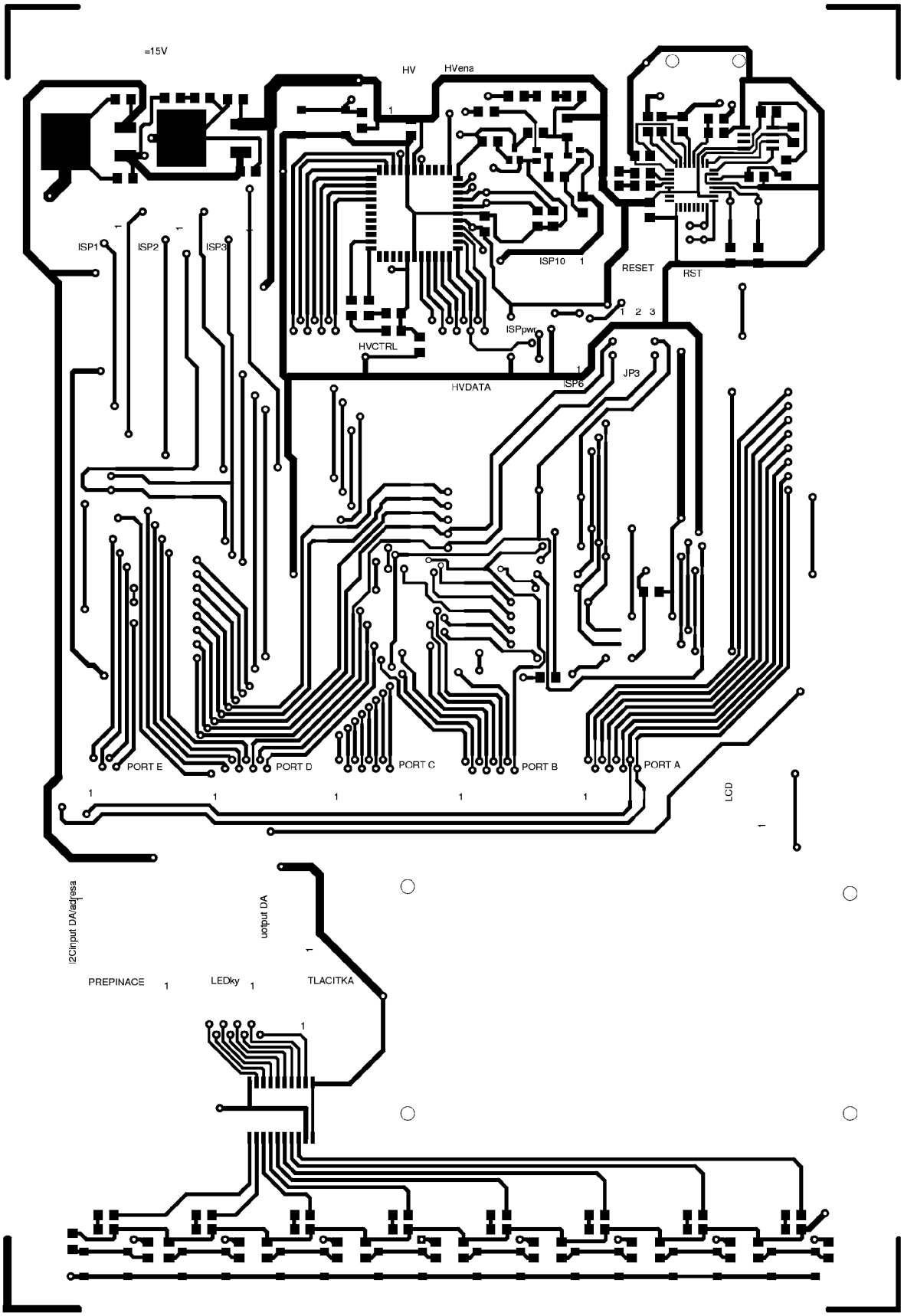
Příloha B: Desky plošných spojů

B.1 Spodní strana



Obr. 10.4: Deska plošného spoje ze strany spojů

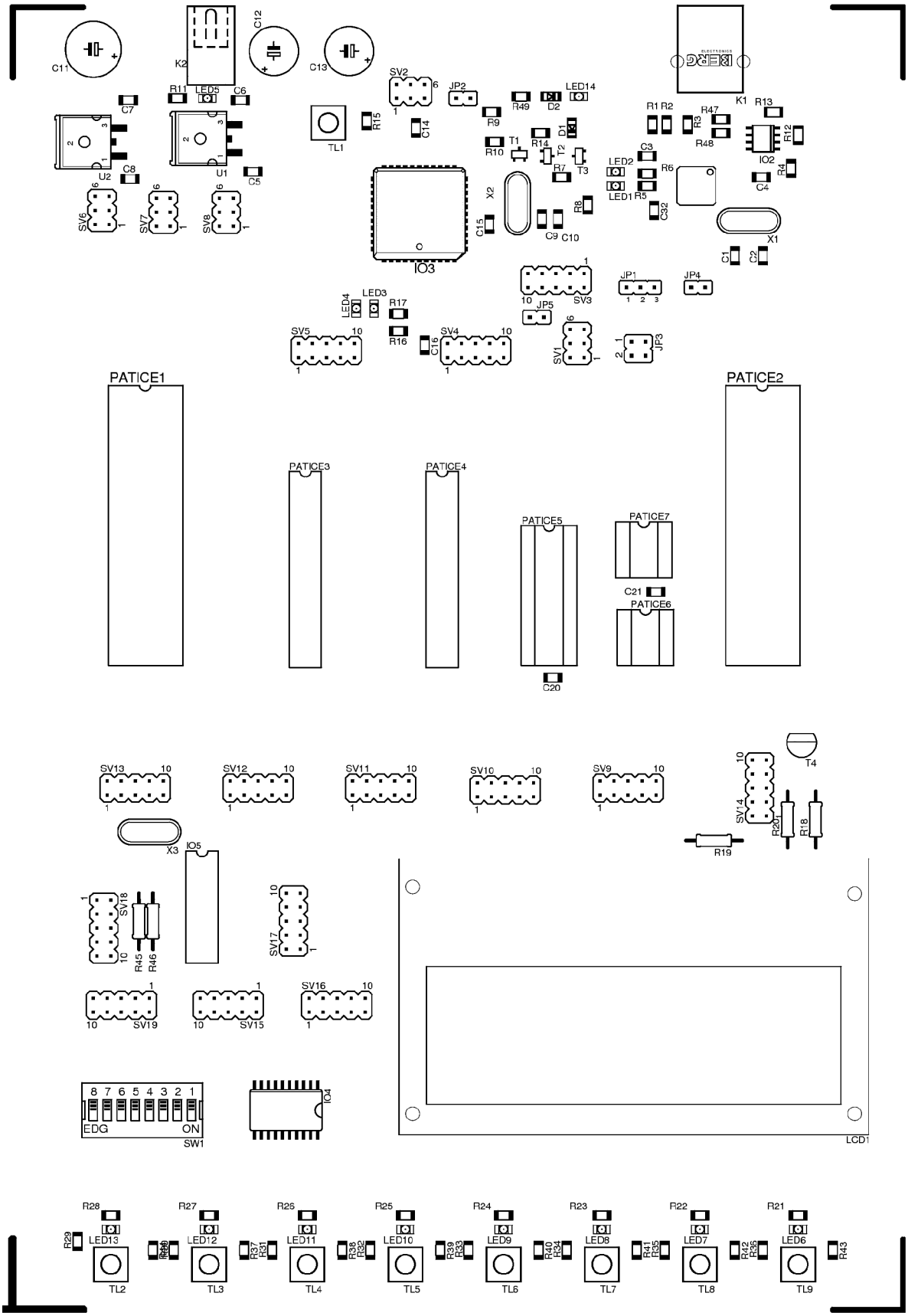
B.1 Horní strana



Obr. 10.5: Deska plošného spoje ze strany součástek

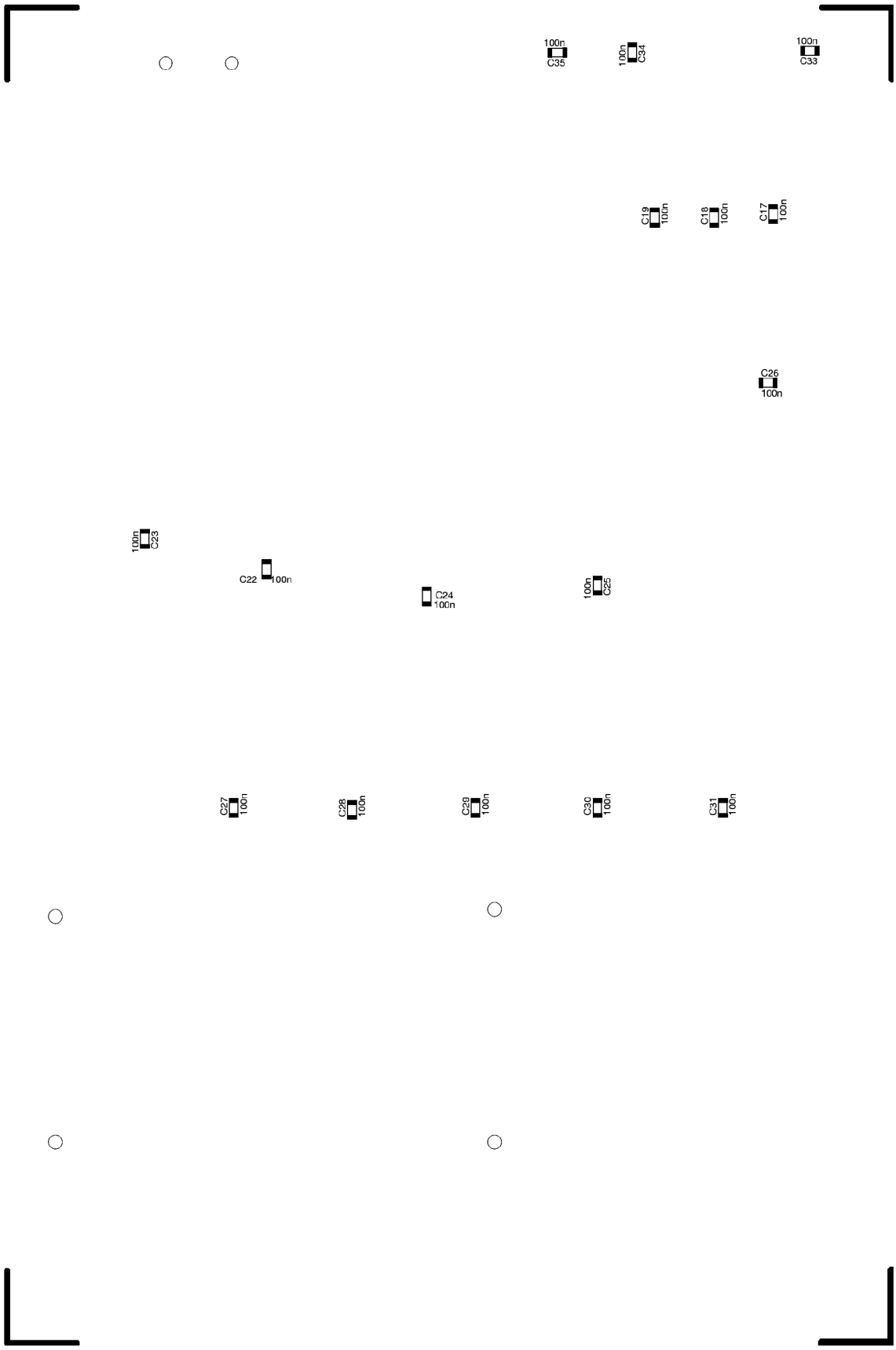
Příloha C: Osazovací plánek

C.1 Horní strana



Obr. 10.6: Osazovací plánek ze strany součástek

C.1 Horní strana



Obr. 10.7: Osazovací plánek ze strany spojů

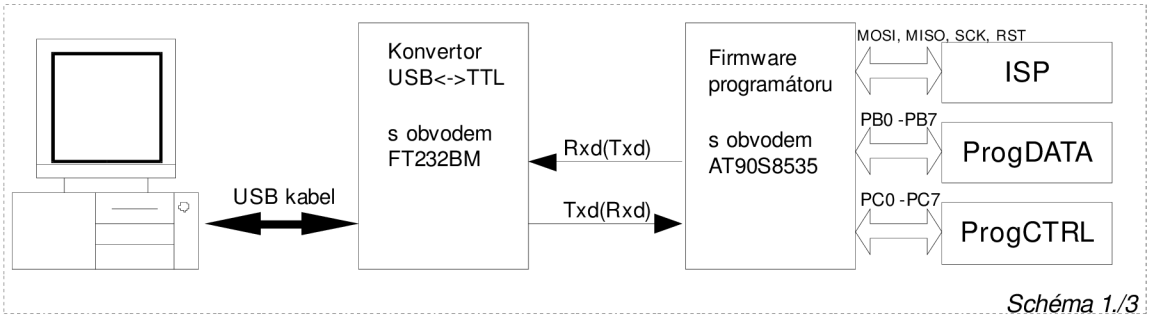
Příloha D: Rozpiska použitých součástek

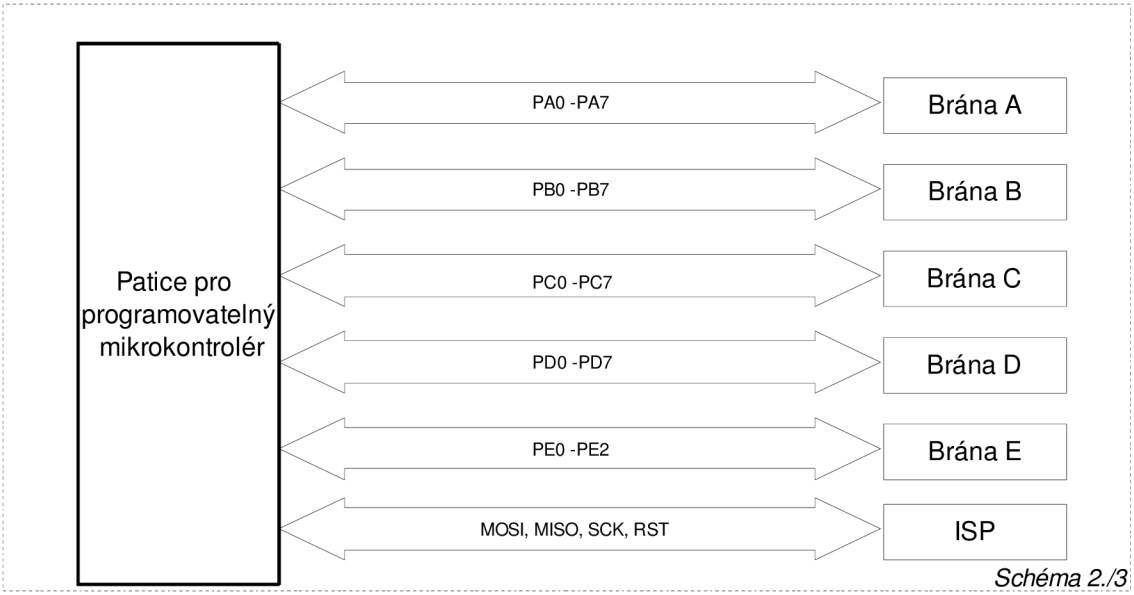
Tab. 10.1: Rozpiska součástek

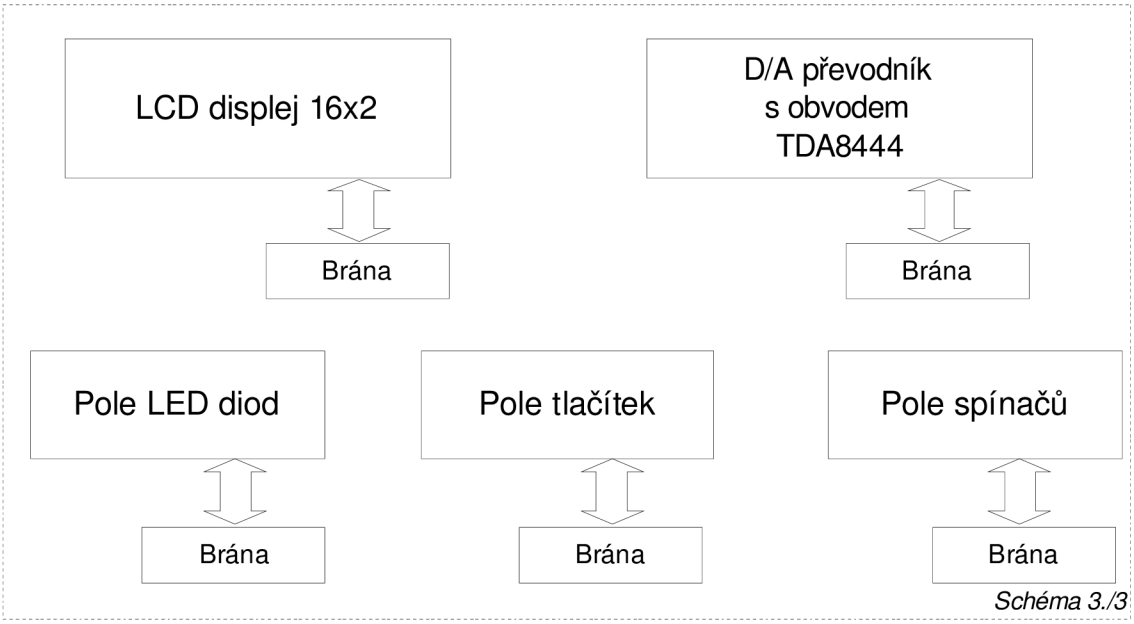
rezistory			
Označení ve schématu	*Hodnota [Ω]*	*pouzdro*	*Typ (z katalogu GM)*
R1, R2	27	SMD 1206	R1206, odpor SMD 0,25W 1% vel.1206
R3, R11	1k5	SMD 1206	R1206, odpor SMD 0,25W 1% vel.1206
R4, R16, R17	470	SMD 1206	R1206, odpor SMD 0,25W 1% vel.1206
R5, R6, R21 až R28	330	SMD 1206	R1206, odpor SMD 0,25W 1% vel.1206
R20	330	1207	RRU, uhlíkový rezistor do 0,5W
R7, R8, R13, R14, R15, R29 až R36, R45, R46, R48	10k	SMD 1206	R1206, odpor SMD 0,25W 1% vel.1206
R9, R10, R47	4k7	SMD 1206	R1206, odpor SMD 0,25W 1% vel.1206
R19	4k7	1207	RRU, uhlíkový rezistor do 0,5W
R12	2k2	SMD 1206	R1206, odpor SMD 0,25W 1% vel.1206
R18	10	1207	RRU, uhlíkový rezistor do 0,5W
R37 až R44	150	SMD 1206	R1206, odpor SMD 0,25W 1% vel.1206
kondenzátory			
Označení ve schématu	*Hodnota [F]*	*pouzdro*	*Typ (z katalogu GM)*
C1, C2, C9, C10	27p	SMD 1206	CK1206, keramický kondenz.SMD 1206 50V
C3	33n	SMD 1206	CK1206, keramický kondenz.SMD 1206 50V
C4	47n	SMD 1206	CK1206, keramický kondenz.SMD 1206 50V
C5 až C8, C14 až C32	100n	SMD 1206	CK1206, keramický kondenz.SMD 1206 50V
C11	100uF/25V	-	E100M/25V2, elyt radiální 5x11mm RM2
C12, C13	10uF/25V	-	E10M/25V2, elyt radiální 5x11mm RM2
diody			
Označení ve schématu	*typ*	*pouzdro*	*Typ (z katalogu GM)*
D1	1N4148	MICRO-MELF	Univerzální dioda 75V 150mA MCL4148
LED1 až LED13	zelená	SMD 1206	L-HSMG-C150
stabilizátory			
Označení ve schématu	*typ*	*pouzdro*	*Typ (z katalogu GM)*
U1	7805	DPAK SMD	7805 DPAK SMD, stabil.SMD+5V1A SMD DPAK
U2	7812	CD2T SMD	7812 CD2T SMD, stabil.SMD+12V1A D2PAK (SMD)
Integrované obvody			
Označení ve schématu	*typ*	*pouzdro*	*Typ (z katalogu GM)*
IO1	FT232BM SMD	LQFP-32	FT232BL, =FT232BM, jednočipový převodník USB <-> RS232, USB 1.1 i USB 2.0 kompatibilní, podpora Win98, Win98SE, Win 2000 / ME / XP, Mac OS-8, Mac OS-9, Linux 2.40 a vyšší, TQFP32(LQFP-32)
IO2	93LC46B	SOIC 8	93LC46B-I/SN, Sériová EEPROM s rozhraním I2C, pevná organizace 64x16 bit, Ucc=2,5V až 5,5V, industry temp.range, SOIC 8
IO3	AT90S8535	PLCC44	ATmega8535-16JI, ATMEL AVR, 8kB program FLASH, 512byte SRAM, 512byte EEPROM,2x8bit Timer, 1x16bit Timer, 4 PWM, USART,SPI, I2C, 8x10bit ADC, Analog comparator, internal RC oscilator, Brown-out detector, PLCC44, Nahrada za 90S8535
IO4	74HCT640	SO20	74HCT640 SMD, 8bit inv.výk.BUS trans.
IO5	TDA8444		TDA8444, Integrovaný obvod D/A převodník 8x6bit IIC-Bus

			DIP16
tranzistory			
Označení ve schématu	typ	pouzdro	Typ (z katalogu GM)
T1, T2	BC847B	SOT23	BC847BPN, N-50V 0.1A ß=200-450 SMD 300MH
T3	BC857B	SOT23	BC857B, P-50V 0.1A ß=220-475 SMD 150MH
T4	BC558B	TO-92	PNP 30V 0,1A ß= 180-460
rezonátory			
Označení ve schématu	Hodnota [MHz]	pouzdro	Typ (z katalogu GM)
X1	6	HC49US	QM 6.000MHZ, Krystal mini HC49US 50ppm -10/+60°C
X2	7,37	HC49US	QM 7.372MHZ, Krystal mini HC49US 50ppm -10/+60°C
X3	4	HC49US	QM 4.000MHZ, Krystal mini HC49US 50ppm -10/+60°C
Patice			
Označení ve schématu	*typ*	*pouzdro*	*Typ (z katalogu GM)*
Patice1, patice2	40DIP	DIL40	TEXTOOL40UNI, testovací patice 40 pin
patice3, patice4	28DIP	DIL28	DIL28PZ, precizní patice 28 pin
patice5	20DIP	DIL20	DIL20PZ, precizní patice 20 pin
patice6, patice7	8DIP	DIL8	DIL08PZ, precizní patice 8 pin
konektory			
Označení ve schématu	*typ*	*pouzdro*	*Typ (z katalogu GM)*
K1	USB konektor	-	USB1X90B PCB, USB zásuvka B do DPS, 90st.
K2	Napájecí konektor	-	K375A, nap.vidl.2.1mm do PLS 90°
SV3, SV4, SV5, SV9 až SV19	Konektorová lišta 2x5 do DPS	-	S2G80, pozlacená lámací lišta
SV1, SV2, SV6 až SV8	Konektorová lišta 2x3 do DPS	-	S2G80, zlacená lámací lišta
JP1	Lišta 3x1 do DPS	-	S1G20, zlacená lámací lišta
JP2	Lišta 2x1 do DPS	-	S1G20, zlacená lámací lišta
JP3	Konektorová lišta 2x2 do DPS	-	S2G80, pozlacená lámací lišta
displeje			
Označení ve schématu	*typ*	*pouzdro*	*Typ (z katalogu GM)*
LCD1	2x16 znaků	-	MC1602E-SYL, LCD, 16x2, STN, LED podsvětlení, žlutozelené pozadí
Přepínače, tlačítka			
Označení ve schématu	*typ*	*pouzdro*	*Typ (z katalogu GM)*
SW1	DIP8	DIL8	DIP 8X, spínač DIL 8x
TL1 až TL9	mikrospínač	SMD	P-B1720/SMD, tact switch 12V 0.05A, půdorys 6x6 mm, výška 4,4mm

Příloha E: Blokové schéma zapojení

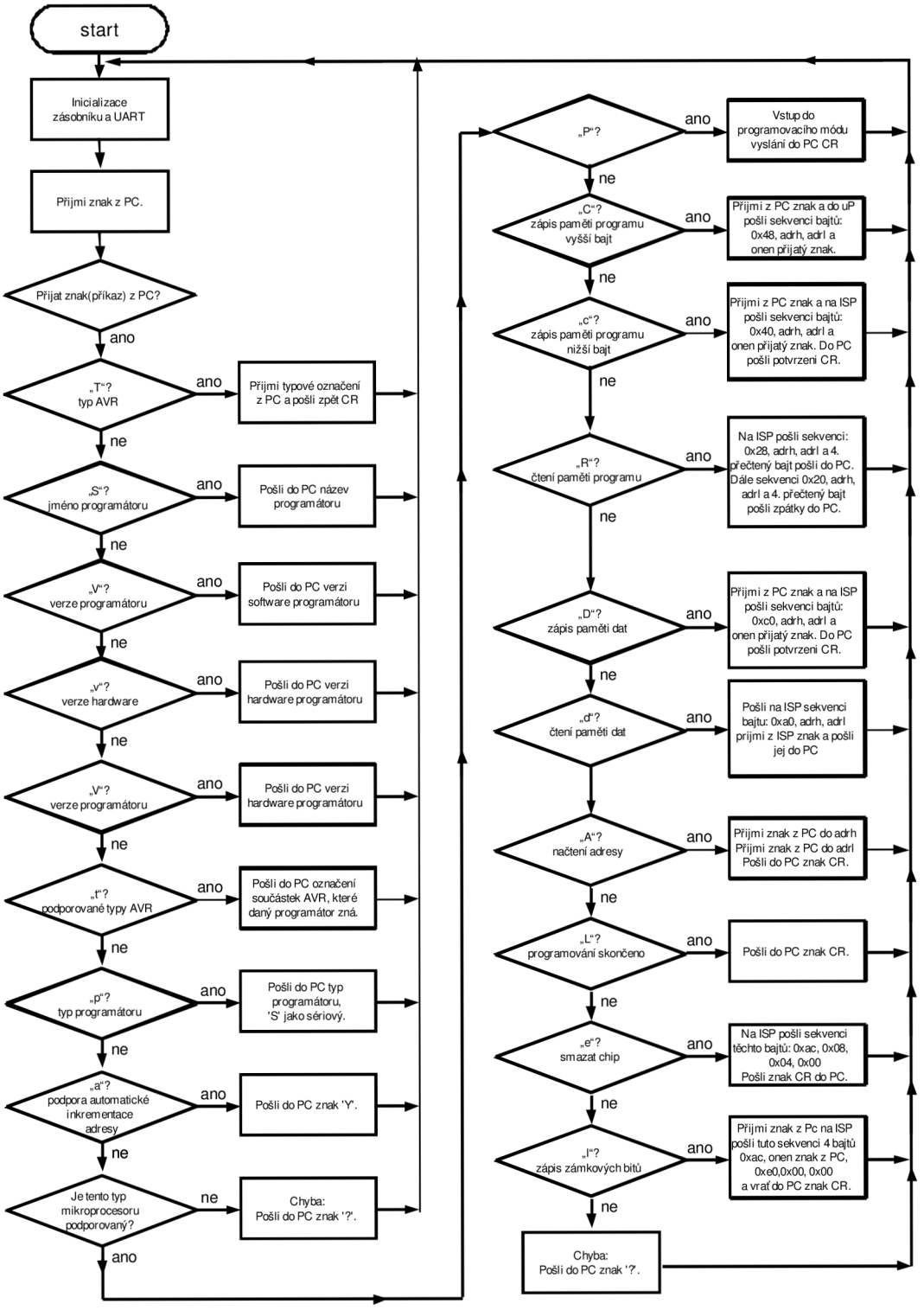






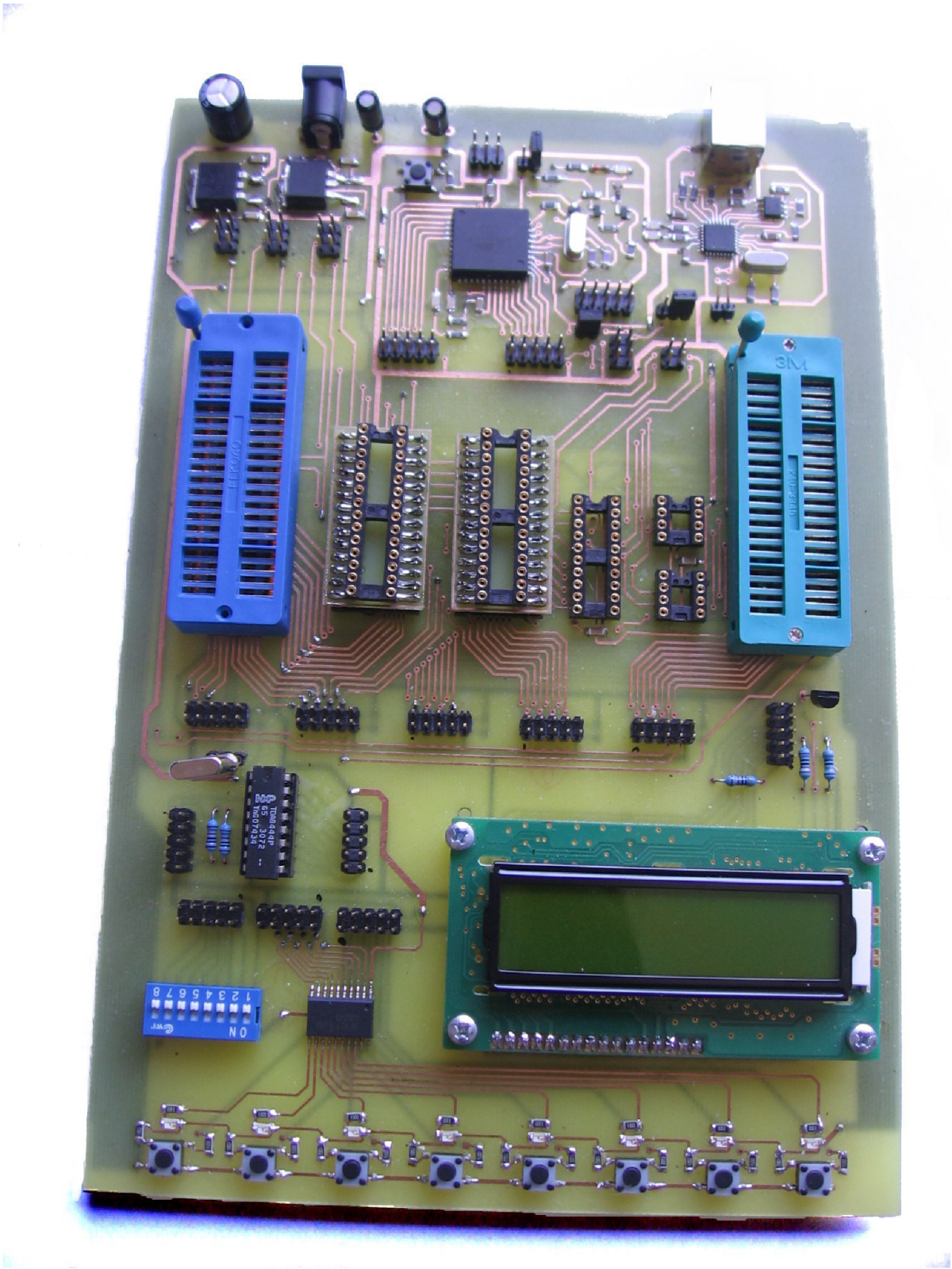
Obr. 10.8: Blokové schéma zapojení

Příloha F: Podrobný vývojový diagram obslužného softwaru



Obr. 10.9: Podrobný vývojový diagram

Příloha G: Fotografie přístroje



Obr. 10: Fotografie přístroje

Příloha H: CDROM

H.1 Elektronický text bakalářské práce ve formátu pdf

H.2 Návrh konstrukce v elektronickém formátu Eagle

H.3 Zdrojový a binární kód obslužného programu

H.4 Fotografie přístroje