České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Josef Nouzák**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Výpočetní technika

Název tématu: **Konfigurovatelný řídicí modulární systém**

Pokyny pro vypracování:

Navrhněte řídicí systém s modulární architekturou. Systém bude obsahovat řídicí desku s procesorem ATmega2560 a jen nezbytné periferie jako je USB rozhraní, 2xRS232, ps/2 klávesnice, SPI, JTAG, RTC, sériová EEPROM, SD karta, výstup na display a 2x rozšiřující konektor. Dále pak navrhněte modul připojitelný k řídicí desce. Modul bude obsahovat: 2x vstup pro 0-24V ss a 2x výstupní relé dovolující spínat 220V. Výsledná deska by měla zabírat minimální plochu. Navržené řešení zrealizujte. Pro řídicí procesor napište knihovnu základních funkcí ovládajících jednotlivé periferie. Funkci knihovny otestujte s pomocí jednoduché demonstrační aplikace.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Pavel Kubalík, Ph.D.

Platnost zadání: do konce letního semestru 2009/2010

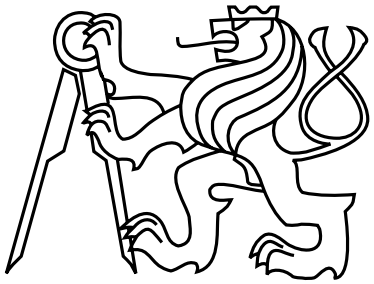
doc. Ing. Miroslav Šnorek, CSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 7. 1. 2009

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Diplomová práce

Konfigurovatelný řídicí modulární systém

Bc. Josef Nouzák

Vedoucí práce: Ing. Pavel Kubalík Ph.D.

Studijní program: Elektrotechnika a informatika, strukturovaný, Navazující magisterský

Obor: Výpočetní technika

5. ledna 2010

Poděkování

Rád bych zde poděkoval vedoucímu práce Ing. Pavlu Kubalíkovi PhD. za jeho ochotu okamžitě se mnou řešit problémy, které nastaly během tvorby této práce, jeho bleskovou odezvu při komunikaci a celkovou vstřícnost.

Zvláštní poděkování patří mým rodičům za jejich podporu po celou dobu mého studia.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

Bečvárech dne 5. ledna 2010 .....

Abstract

The thesis deals design and realization of modular control system. System can work everywhere, where we need collect data or do some regulation actions. Variability of the system is possible by dividing system to the modules. Modules enable configure system acording concrete requierements. During the work has been developed processors board, modules of otput and input and ethernet module. As part of work has been assebled module of LCD and USB-UART converter.

Abstrakt

Práce se zabývá návrhem a realizací modulárního řídícího systému. Systém může posloužit všude tam, kde je potřeba sbírat data, případně provádět regulační zásahy. Variabilita systému je zajištěna rozdělením do modulů, které lze připojovat dle konkrétních požadavků.

Během práce byla vytvořena hlavní procesorová deska a moduly vstupů, výstupů a ethernetu. V rámci práce byl osazen modul displeje a vytvořen převodník USB-UART.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Úkolem práce je navrhnout a postavit konfigurovatelný řídící modulární systém založený na procesoru ATmega2560. Tento systém je možné použít v řadě aplikací např. jako časomíru, vestavěný řídící systém (měření, řízení a regulace) s možností konfigurace dle požadavků, komunikační systém, případně jako pomůcku pro podporu výuky studentů. Hardware systému se bude skládat z různých modulů, které je možné spojovat dle konkrétních požadavků. Těmito moduly mohou být vstupy, výstupy, LCD displej, modul GPS, případně Ethernet atp.. Základním modulem je procesorová deska osazená procesorem ATmega2560, která obsahuje tyto periferie:

- USB,
- SD card slot,
- 2x RS232,
- PS2,
- Real Time Clock,
- Serial EEPROM,
- 2x Rozšiřující konektor.

Připojení k počítači bude realizováno pomocí USB. Programování desky by mělo probíhat přes SPI, JTAG nebo USB. K desce budou následně připojovány periferní desky, které rozšiřují vlastnosti procesorové desky. V této práci budou popsány moduly vstupu a výstupu, ethernet a napájecí redukce. Spolu s těmito deskami vznikl ještě převodník USB na UART s napětími 3.3V a 5V.

Pro otestování funkcí desky byl napsán firmware, který umožňuje otestovat jednotlivé periferie. Spolu s tímto firmware vznikla knihovna funkcí, s jejíž pomocí lze snadno ovládat jednotlivé periferie desky.

Kapitola 2

Popis problému, specifikace cíle

Hlavním úkolem je vytvořit jednoduchý levný systém, který najde uplatnění v jednodušších aplikacích. Systém by měl být snadno rozšiřitelný a upravitelný.

- Navrhnout a vyrobit procesorovou desku.
- Navrhnout a vyrobit moduly vstupů a výstupů.
- Navrhnout modul ethernetu.
- Naprogramovat demoaplikaci.
- Napsat firmware LCD displeje.
- Vyrobit USB <-> UART převodník.

Převodník USB<->UART je podpůrný vývojový prostředek, který by měl umožnit odladění firmwaru LCD displeje, může však být použit i v jiných aplikacích.

Kapitola 3

Analýza a návrh řešení

S různými modulárními systémy se můžeme setkat hlavně v automatizační technice, kde mají na starost řízení průmyslových linek, inteligetních domů atd. Systémy zpracovávají informace od čidel, jako jsou teplota, tlak, vlhkost, napětí, osvětlení atd. Na základě těchto měření mohou provádět regulační zásahy podle programu. Variabilita je zajištěna tím, že systém je rozdělen na desku s procesorem a další moduly vstupů a výstupů dat, případně modulů pro přenos informací. Odtud plyne název modulární. Modulární systémy vyvíjejí různé firmy např.: Siemens, Allen Bradley, Teco. Systémy těchto firem využívají většinou 32b procesory s taktovací frekvencí okolo 200MHz, jen v některých nejmenších verzích bývají osazeny 8b CPU. Cena uvedených systémů je ovšem značná, zákazník platí za odolnost a spolehlivost. Automaty těchto firem se často programují specializovanými SW dodávanými výrobcem automatu po zaplacení licence.

Obrázek 3.1: Automaty firem TECOMAT nahoře, SIEMENS vlevo a Allen Bradley vpravo

Mnou navrhovaná deska nemá za cíl vyrovnat se těmto systémům, ale jejím cílem je poskytnout levnější alternativu pro menší aplikace. Vyvíjená deska bude osazena procesorem

od firmy ATmel [5] z rodiny ATmega konkrétně ATmega2560. Jedná se o RISC procesor pracující na frekvenci až 16MHz, při této frekvenci dosahuje výkonu maximálně 16MIPS. Procesor má harwardskou architekturu a obsahuje 256Kbyte Flash Program memory, 8KByte SRAM a 4kB EEPROM. Programování bude prováděno přes SPI nebo JTAG volně dostupnými prostředky (C, Assembler). Deska najde uplatnění všude tam, kde je kladen důraz na cenu. Může posloužit jako vývojová platforma pro vývoj vstupně výstupních modulů, případně může být použita pro konkrétní aplikace např. časomíru, meteorologickou stanici atp.

3.1 Nabídka trhu

Nejpodobnějšími produkty k zamýšlené desce jsou výrobky firem DIGILENT [2] a JED Microprocessors Pty Ltd [3]. Dalším výrobcem, který vyrábí podobnou desku ovšem osazenou pouze procesorem ATmeaga128 je firma PK Design [20]. Výrobek byl do srovnání zařazen, protože je vyráběn v České republice.

Výrobek firmy DIGILENT obsahuje pouze mikroprocesor a RAM paměť o velikosti 128kB. Deska disponuje konektory, na které jsou vyvedeny jednotlivé brány procesoru. Procesor je možné programovat přes ISP rozhraní nebo přes JTAG.

Firma JED Microprocesor nabízí desku JED AVR256, která je osazená procesorem ATmega2560. Deska obsahuje požadované periferie USB, SDslot, EEPROM, RTC a pro použití rozhraní RS232 je pořeba dokoupit redukci. Navíc je zde akcelerometr a výstupy vyvedené na svorkovnice jsou opatřeny výkonovými tranzistory. Nevýhodou této desky jsou její větší rozměry. Největší nevýhodou je dostupnost a cena, firma sídlí až v Austrálii.

Třetí deskou, která byla zvolena pro srovnání, přestože nemá požadovaný typ mikroprocesoru, je deska MB-ATmega128 v4.0 firmy PK Design. Deska je osazená procesorem ATmega128, což je patrné již z názvu. Na desce se nachází kromě procesoru jeden RS232 port, RTC, USB a chybí na ní SDslot a EEPROM. Navíc je deska osazena 128kB paměti SRAM.

Ceny jednotlivých desek spolu s parametry jsou uvedeny v tabulce 3.1.

parametr	**CEREBOT-PLUS**	**JED AVR256**	**MB-ATmega128**
USB	-	x	x
SDslot	-	x	-
RS232	-	modul	x
EEPROM	-	x	-
RTC	-	x	-
CENA	60USD	100USD	1600Kč

Tabulka 3.1: Desky jednotlivých výrobců

K cenám v tabulce je nutné připočítat cenu dopravy, která je v případě přepravy z Austrálie téměř polovinou ceny výrobku.

3.2 Rozdělení do modulů

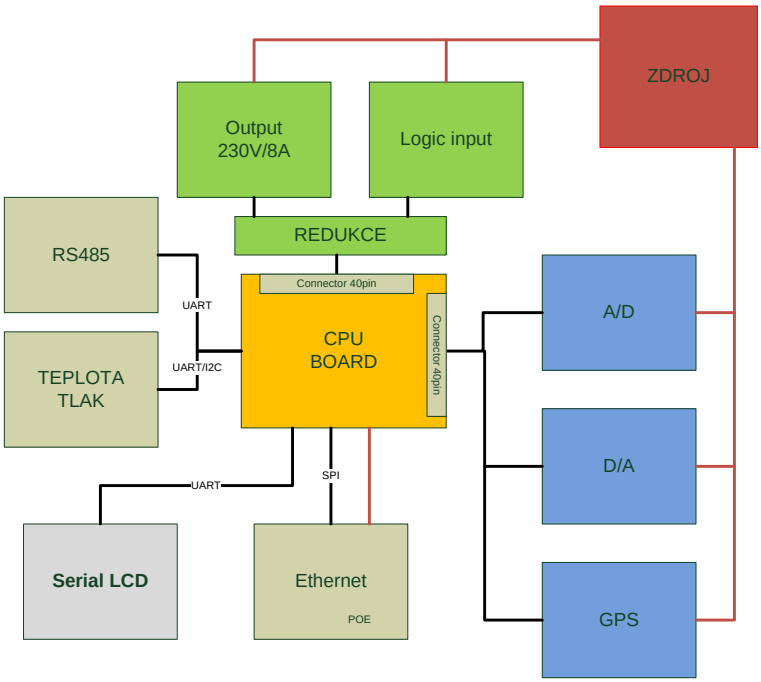
Navrhovaná deska by měla být co nejmenší, proto na ní není dostatek místa pro umístění některých vstupů a výstupů. Na procesorové desce budou vyvedeny pouze komunikační porty USB, RS232 a PS2. Pro rozšíření bude deska obsahovat dva 40pinové konektory, do kterých by měly být vyvedeny brány procesoru, seriové linky a komunikační sběrnice I^2C. Předpokládané rozložení systému je na obrázku 3.2.

Moduly digitálních vstupů a výkonových výstupů nebudou napájeny z desky, ale z přídavného zdroje (na obrázku červeně). Díky tomu bude možné zdroj na desce zmenšit a výkon přídavného zdroje, jenž by měl napájet různé moduly, se nechá upravit pro konkrétní aplikaci. Tento zdroj může zároveň posloužit jako redukce a rozdělit 40pinový konektor na menší konektory, které poslouží k připojení periferií.

Každý modul vstupů/výstupů (na obrázku zeleně) bude obsahovat dva vstupy/výstupy. Výstupy musí být schopné pracovat s napětím až 230V, proto budou rozměrnější, aby byly dodrženy izolační vzdálenosti na DPS. Vstupy musí být opatřeny galvanickým oddělením, aby při špatné manipulaci nedošlo k poškození procesorové desky.

K desce se dále nechá připojit modul Ethernetu. Ten by měl umožnit napájet desku prostřednictvím pasivního POE. Modul bude mít výstup napětí i pro zdroj, případně jiné zařízení tak, aby se nechal použít i samostatně u jiných zařízení.

Deska předpokládá, že periferie typu GPS, AD převodník budou opět obsahovat procesor, s kterým se bude komunikovat buď po seriové lince nebo paralelně.



Obrázek 3.2: Předpokládané rozdělení do modulů

3.3 Návrh desky

Procesorová deska (CPU BOARD) je hlavní součástí modulárního systému. Na desce jsou vyvedeny komunikační a rozšiřující porty. Blokové schéma zamýšlené desky je na obrázku 3.3.

Rozšiřujícími porty jsou konektory slot1 a slot2. Konektory mají za úkol umožnit rozšíření vlastností desky. Do slotů jsou vyvedeny IO porty, tři UARTY a sběrnice I^2C. Zapojení signálů ve slotu je zvolené tak, aby bylo možné pro propojení desky s periferiemi použít 80 žilový kabel ULTRA ATA. Signály sériových linek a sběrnice I^2C budou v obou slotech na stejných místech. To umožní připojovat periferie pracující s těmito signály do libovolného slotu.

Na desce se nachází PS2 konektor pro připojení klávesnice, dvojitý převodník UART <-> RS232 a převodník FTDI, který převádí rozhraní USB na sériovou linku se všemi signály pro řízení přenosu. Signály RX a TX jsou připojeny na UART0 procesoru. Tím je zajištěna možnost komunikovat s deskou přes USB rozhraní. Signály pro řízení přenosu z převodníku FTDI je možné pomocí propojek připojit k ISP procesoru. To by mělo umožnit naprogramovat procesor přes USB.

Na rozhraní SPI je dále připojena SD karta a konektor pro Ethernet modul, případně jiné zařízení komunikující přes SPI. K procesoru bude připojeno RTC a EEPROM přes I^2C sběrnici.

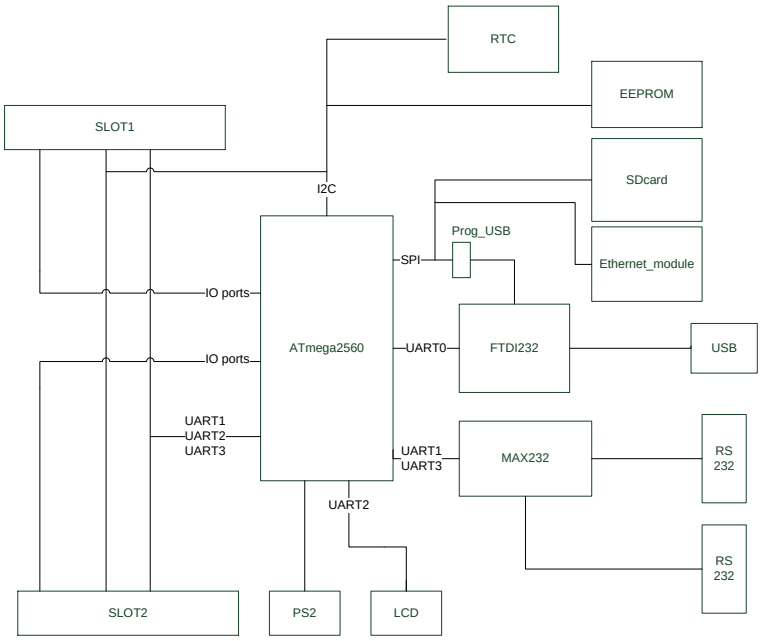
Jelikož se na desce nacházejí obvody pracující s napětím 5V i 3,3V, bude muset mít deska dva zdroje. Zdroj s napětím 5V bude napájet CPU a obvody, které s tímto napětím pracující. Napětí bude dále sníženo druhým stabilizátorem na 3,3V, kterým se bude napájet převodník úrovní a SD karta. Dva zdroje jsou nutné, protože procesor není schopen při napětí 3.3V pracovat na frekvenci vyšší než 10MHz.

3.4 Výběr obvodů

Základ desky tvoří procesor ATmega2560, který byl vybrán již na začátku práce. Procesor firmy Atmel byl vybrán pro dostupnost programátorů a vývojových prostředků.
Nejjednodušší programátor lze sestavit na sériový port za použití tří diod a odporů. Výběr obvodů pro realizaci rozhraní byl proveden podle jejich parametrů, ceny a dostupnosti.

RTC

Obvod realného času by měl plnit tyto požadavky: provoz na baterii, interní krystal, alarm a kalendář. Obvod by měl komunikovat po sběrnici I^2C. Z výběru byly hned na začátku odstraněny obvody komunikující po SPI. Rozhraní SPI by mělo být použito pro komunikaci s SD kartou a případně pro ethernet. Z výběru vypadly všechny obvody DS13xx firmy MAXIM tyto, obvody potřebují externí krystal. Po zvážení ceny, parametrů a také velikosti dostupného pouzdra zbývajících obvodů byl vybrán obvod firmy EPSON RTC-8564 [11], s jehož dostupností by neměl být problém. Tento obvod má všechny uvedené parametry a je schopen pracovat s napětím 1,8V - 5V.



Obrázek 3.3: Blokové schéma desky

UART RS232

Pro realizaci tohoto převodníku je nejjednodušší použít obvody firmy MAXIM, případně jeho klon od jiného výrobce. V úvahu přicházel MAX232 a MAX233 [15]. Rozdíl mezi verzí MAX232 a MAX233 je, že verze MAX233 nepotřebuje ke své činnosti žádné externí součástky (kondenzátory). Po prověření dostupnosti a ceny těchto převodníků jsem se rozhodl použít MAX232, který je běžně k dostání na rozdíl od druhého výše jmenovaného a jeho cena je mnohem nižší než u MAX233.

EEPROM

EEPROM musí komunikovat přes I^2C sběrnici a její velikost může být libovolná. Jako nejlepší řešení se jeví použít paměť z řady 24Cxxx, kde xxx uvádí kapacitu v bitech. Tyto paměti vyrábí téměř každá firma, jmenujme například Atmel, STmicroelectronics, Microchip. Všechny paměti komunikují po I^2C rychlostí 400kHz, jen paměti firmy Microchip mají v datasheetu uvedeno, že ve verzi 24Fxx dokáží komunikovat rychlostí až 1MHz. Verze 24F však není běžně k dostání.

SD Karta

SDkarta pracuje s napětím 3,3V a bude připojena na SPI rozhraní procesoru. Procesor pracuje s napětím 5V, proto je zde nutné realizovat zapojení, které upraví logické úrovně pro komunikaci. Po porovnání různých zapojení jsem se rozhodl použít integrovaný obvod 74LVX125 [12]. V pouzdře integrovaného obvodu jsou čtyři třístavové budiče. Jeden integrovaný obvod proto postačí pro převod signálů SPI rozhraní MISO, MOSI, SCK a SS.

Obvod se používá pro převod 5V na 3V. Při napájecím napětí 3,3V snáší na vstupu až 7V. Mezní frekvence obvodu je téměř 50MHz, a proto z hlediska rychlosti je obvod více než dostačující pro SPI. S kartou se bude komunikovat maximálně na frekvenci Fcpu/2.

3.5 Napájení desky

Napájení desky bude rozděleno mezi dva zdroje, 5V a 3.3V. Jako první byl proveden odhad spotřeby zařízení pro každý zdroj zvlášť tabulky 3.2 a 3.3

Součástka	**Proud**
ATmega2560	100mA
MAX232	10mA
RTC	1mA
EEPROM	5mA
LED	5mA
CELKEM	121mA

Tabulka 3.2: Spotřeba součástek 5V

Součástka	**Proud**
SDkarta	70mA
74LVX125	10mA
CELKEM	80mA

Tabulka 3.3: Spotřeba součástek 3.3V

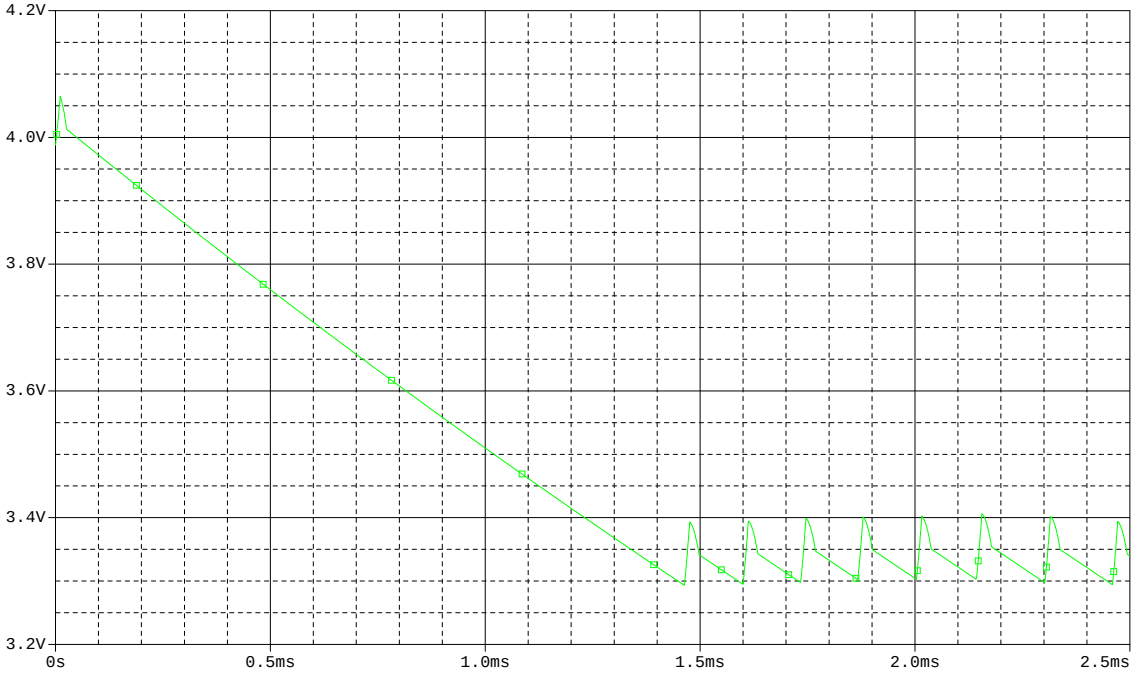
Proud uvedený u RTC je maximální. V případě, že se s RTC nekomunikuje, klesá jeho odběr na stovky nA. Stejný jev je u EEPROM v případě, že se s touto pamětí nekomunikuje klesne i spotřeba.

Pro napájení byl uvažován spínaný regulátor firmy ONsemiconductor mc34063 [18], který dokáže pracovat s proudem až 1,5A. Tímto spínaným regulátorem by šla realizovat větev 3,3V i 5V. Pro jednotlivé větve se jen přepočítají odpory v děliči, jenž tvoří zpětnou vazbu. Při simulaci 3,3V větve ovšem zdroj vykazoval nepříjemnou špičku, která vznikala po zapnutí a měla amplitudu cca 4V. Simulovaný průběh je na obrázku 3.4. Simulace obvodu zdroje byla provedena v prostředí Pspice, jako model regulátoru byl použit model ze stránek výrobce - firmy ONsemiconductor. Špička, kterou je vidět na obrázku 3.4, by se pravděpodobně nechala odfiltrovat pomocí transilu, který by ji svedl a tím by snížil nebezpečí poškození součástek. Transil by se musel připojit na výstup regulátoru.

Po zvážení všech pro a proti byl spínaný zdroj opuštěn a pro napájení desky byly vybrány zpětnovazební stabilizátory LE33CD [25] pro 3V3 větev a 7805 [19] pro 5V. Stabilizátory by měly pro napájení desky postačovat. Jejich výhodou je velikost, nejsou zdroji rušení a navíc dokážou potlačit rušení o nižších frekvencích. Největší nevýhodou je nízká účinnost. Oba stabilizátory jsou vybaveny nadproudovou i tepelnou ochranou.

Maximální vstupní napětí by nemělo přesáhnout 24V. Při tomto napětí a odběru 200mA by měla být ztráta na stabilizátoru cca 4,8W. Osobně ovšem odhaduji spotřebu 100mA a v tomto případě se bude ztrátový výkon pohybovat někde okolo 2,4W. Obě tyto hodnoty jsou maximální a vzhledem k povaze obvodu (zatížení CMOS součástkami) lze předpokládat, že reálný ztrátový výkon bude menší. Stabilizátor by měl v nejhorším případě, tedy bez chladiče a při teplotě okolí 30°C, snést výkonovou ztrátu 2W. Stabilizátor 7805 bude pro zlepšení odvodu tepla připájen naležato k desce. Tím by mělo být chlazení dostatečné a případně by se na něj mohl přilepit malý chladič.

Pro optimální napájení desky by mělo být použito napětí 7,5 - 18V a proud max 0,5A.



Obrázek 3.4: Průběh výstupního napětí v simulaci

Zdroj desky není určen pro napájení periferních modulů. Pro napájení modulů se použije samostatný zdroj, který bude dimenzován na potřebný výkon. Vytvářet na procesorové desce velký spínaný zdroj, který by dodával proud pro moduly, je neefektivní. Zdroj by ve většině aplikací pracoval s malým proudem, byl by zbytečně veliký a jeho napětí by muselo být maximálně 5V. V případě externího zdroje lze vytvářet libovolné napětí a tím pak třeba i ovládat akční členy.

3.6 Analýza nákladů

Cena navrhované desky je stanovena odhadem podle odhadnuté plochy DPS a ceny součástek.

Jako předloha pro odhad sloužil prototyp Bc. Libora Hrbka [1]. Maximální velikost desky byla stanovena na 1dm^2 s tím, že výsledná deska by měla být co nejmenší. Měla by jít realizovat na oboustranném plošném spoji. Vícevrstvým plošným spojem by se sice docílilo ještě dalšího zmenšení, ale cena by vzrostla. Součástky by měly být umístěny, pokud možno na jedné straně, aby se ušetřilo za výrobu masky pro potisk druhé strany. Předběžný odhad je vidět v tabulce 3.4.

V ceně DPS jsou započítány náklady na zhotovení podkladů pro výrobu, které jsou pouze jednorázové. V případě sériové výroby by se tato položka rozpočítala na každou vyrobenou desku. Výsledná cena desky by tedy byla nižší než tento odhad. Stejně tak by klesla cena součástek v případě nákupu většího množství.

Materiál	**Cena**
DPS 1dm	1000Kč
ATmega2560	250Kč
RTC	100Kč
EEPROM	30Kč
FTDI232	250Kč
Stabilizatory	70Kč
MAX232	60Kč
Konektory	100Kč
Pasivní součástky	180Kč
CELKEM	2040Kč

Tabulka 3.4: Rozpočet materiálu

Kapitola 4

Realizace

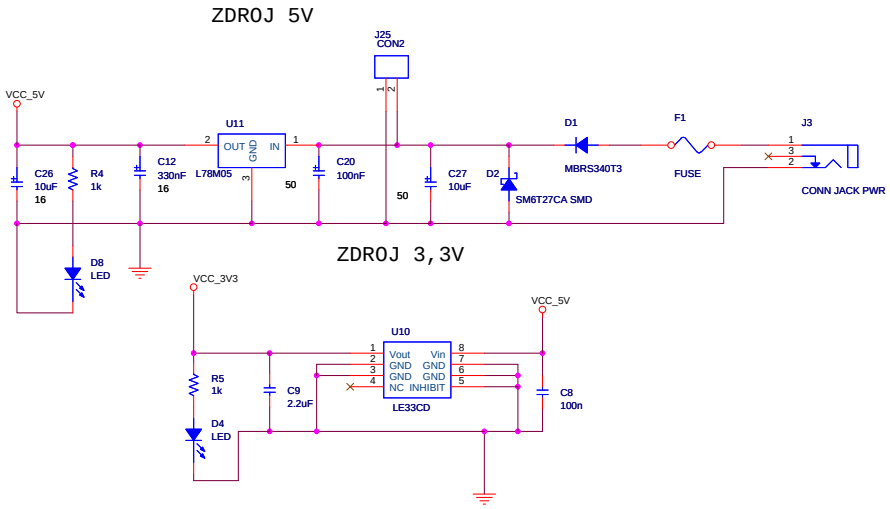
V této části práce je popsáno zapojení jednotlivých modulů systému, které je důležité pro programování, a dále jsou zde popisy portů, které slouží pro připojení periferií.

4.1 Procesorová deska

Hlavní modul celého modulárního systému je v dokumentaci označován jako CPU_BOARD. Schéma zapojení je v příloze pod písmenem A, vrstva TOP a BOTTOM spolu s rozmístěním součástek je v příloze B.3 a kótované údaje jsou uváděny v palcích (in). Foto osazené desky je v příloze pod písmenem C.

4.1.1 Napájecí zdroj

Deska obsahuje dva zdroje, zapojení je na obrázku 4.1.



Obrázek 4.1: Zapojení stabilizátorů

Použité stabilizátory jsou LE33CD a L7805. Oba stabilizátory jsou v zapojení doporučeném výrobcem. Stabilizátor L7805 lze nahradit stabilizátorem L78M05, který má stejné napětí 5V, ale výstupní proud je 0,5A. Jako ochrana před přepětím na vstupu stabilizátoru je použit transil D2. Diody D8 a D4 mají pouze informativní charakter - indikují přítomnost výstupního napětí stabilizátorů a nemusí být ve zdrojích osazeny. Ve zdroji bylo počítáno s pojistkou F1, ale tato pojistka není na deskách osazena, neboť ve verzi SMD se nepodařilo sehnat zpožděnou. Uvažovaná rychlá pojistka nesnese nabíjení kapacit při zapnutí a v případě osazení by měla být použita pojistka o velikosti T200mA.

Použité zpětnovazební stabilizátory jsou vybaveny nadproudovými pojistkami, proto v případě zkratu na výstupu dodávají do zátěže proud, který je roven jejich jmenovitému proudu. Větší pojistky by proto nemusely na zkrat vůbec zareagovat.

V případě potřeby provozu celé desky na 3,3V by bylo možné vyměnit stabilizátor L7805 za LF33CDT a stabilizátor LE33CD nahradit drátovou propojkou (piny 1 a 8). Dále se neosadí kondenzátory C8, C9 a dioda D8.

4.1.2 EEPROM

Zapojení EEPROM 24LC256 společně s RTC je na následujícím obrázku 4.2. Komunikace se provádí po sběrnici I^2C signály SDA a SDC. Signál WP_EPR slouží k potlačení zápisu do EEPROM, pokud je na logické 1, zápis je potlačen a v případě logické 0, je zápis do EEPROM povolen. Zapojení signálů na jednotlivé piny procesoru je v tabulce 4.1.

SIGNAL	PIN CPU
SCL	PD0
SDA	PD1
WP_EPR	PD5

Tabulka 4.1: Připojení signálů EEPROM na CPU

Adresa EEPROM je nastavena pevně. V případě připojení jiných zařízení na I^2C je potřeba zvolit adresu připojovaného zařízení tak, aby nekolidovala s adresou EEPROM nebo RTC. Adresa pro zápis do EEPROM je 0xA0, pro čtení je 0xA1.

4.1.3 RTC

Jako zdroj reálného času je použit obvod firmy EPSON, RTC-8564. Tento obvod má vestavěný krystal, výstup frekvence 1Hz, 32Hz, 1024Hz a 32,768kHz. Má také vestavěný kalendář a alarm, který v případě, že nastane, generuje přerušení pro procesor.

Obvod nemá vstup pro připojení baterie, proto musí být připojení baterie realizováno externě. V tomto případě přes dvě diody D7 a D10, které při výpadku hlavního napájení umožní napájení obvodu z baterie. Obvod je schopen pracovat s min. napětím 1,8V. Pro zálohování napájení byla vybrána baterie CR2032 s napětím 3V. Baterie byla vybrána pro napájení, protože deska může být bez napájení i delší dobu a v tomto případě by superkapacitor nevydržel zálohovat RTC. Použitá baterie CR2032 má kapacitu 230mAh. Při

spotřebě RTC přibližně 275nA a při samovybíjení 1%/rok by měla baterie vydržet přibližně 9let.

K tomuto číslu můžeme dojít jednoduchým výpočtem. Jako první si přepočítáme samovybíjení na hodinu, abychom sjednotili jednotky. Pak můžeme předpokládat, že samovybíjení je přibližně konstantní a že se baterie vybíjí stále o 0,1% původní kapacity. Za těchto předpokladů činí samovybíjení 2,6μA/h. Nyní již můžeme dosadit do rovnice:

$$t(2,6e^{-6} + 270e^{-9}) = 230e^{-3}$$

Ze vztahu vypočítáme čas, který je 80139 hodin, po přepočtení na roky dostaneme přibližně výše uvedených 9let.

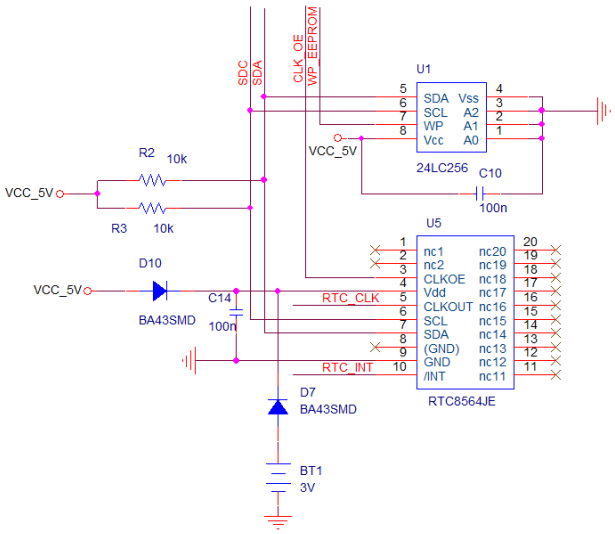
Komunikace je prováděna po I^{2}C signály SDA a SDC. Z obvodu jsou dále na procesor přivedeny RTC_INT a RTC_CLK. RTC_CLK se ovládá signálem CLK_OE. Pokud je signál CLK_OE na logické 1, je výstup RTC_CLK povolen, v opačném případě je výstup zakázán.

Zapojení RTC spolu s EEPROM obrázek 4.2, připojení signálů na piny procesoru je v tabulce 4.2

SIGNAL	**PIN CPU**
SCL	PD0
SDA	PD1
CLK_EN	PD4
RTC_CLK	PB7
RTC_INT	PB6

Tabulka 4.2: Připojení signálů RTC na CPU

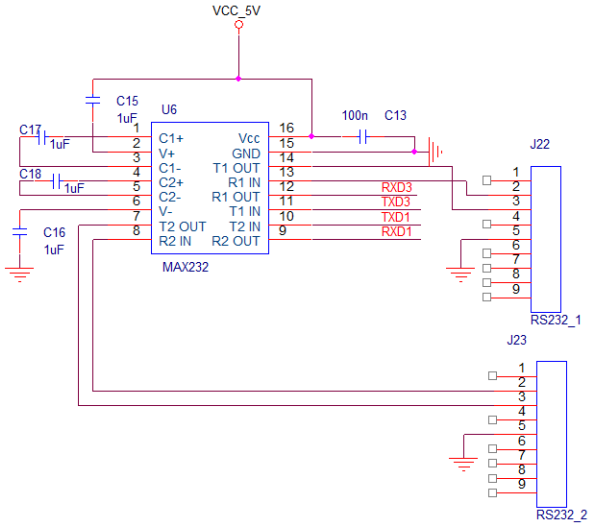
Pro komunikaci s RTC se používají adresy 0xa2 pro zápis a 0xa3 pro čtení.



Obrázek 4.2: Zapojení RTC a EEPROM

4.1.4 Převodník UART-RS232

Jako převodník seriové linky je použit obvod MAX232. Zapojení tohoto obvodu je podle doporučení výrobce. Pro svoji činnost potřebuje 4 kondenzátory o kapacitě $1\mu F$. Kondenzátory jsou nutné, neboť v pouzdře je nábojová pumpa, která z 5V vytvoří $\pm$12V. Obvod je připojen na UART1 a UART3, které převádí na dvojici RS232 portů. Maximální rychlost komunikace přes obvod je 120kbps pro každý port. Konektory RS232_1 a RS232_2 jsou standardní 9pinové cannon male. Zapojení signálů RX a TX v konektorech je schodné s PC, proto pro spojení s počítačem je nutné použít křížený kabel.

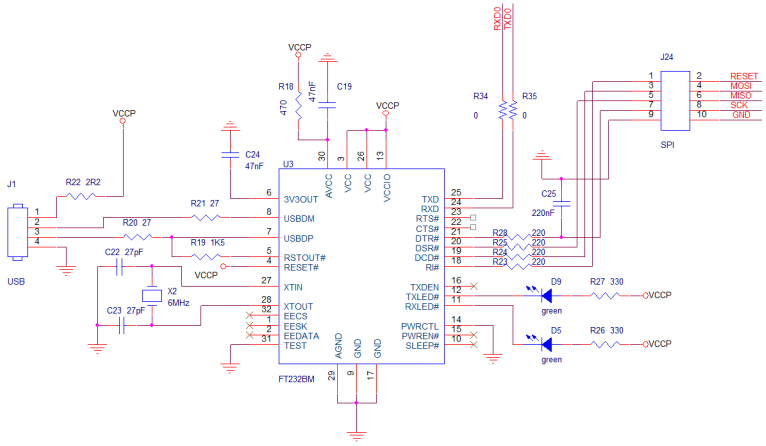


Obrázek 4.3: Zapojení převodníku UART-RS232

4.1.5 Převodník USB-UART

Pro realizaci USB spojení byl zvolen obvod FTDI232BL. Obvod převádí USB na UART se všemi signály pro řízení přenosu (vstupy DCD,DSR,CTS,RI a výstupy DTR, RTS). Zapojení obvodu je na obrázku: 4.4. Toto zapojení je doporučeno výrobcem.

Obvod převodníku není napájen z procesorové desky, ale napájení je přivedeno z USB portu po vodiči VCCP. Tato varianta byla zvolena z důvodu snížení zátěže zdroje procesorové desky. Obvod je využíván pouze při připojeném USB, a proto je napájen z USB. Po odpojení USB je vypnut. Konektor SPI slouží pro programování procesoru přes SPI rozhraní. Programování se nechá provádět pomocí externího programátoru, který se na tento konektor připojí (piny 2,4,6,8 a 10) nebo pomocí FTDI přes USB. V tomto případě by se měl konektor propojit pomocí propojek a programování by mělo proběhnout v BitBang módu FTDI. O programování procesoru prostřednictvím BitBang módu se lze dozvědět zde: [26]

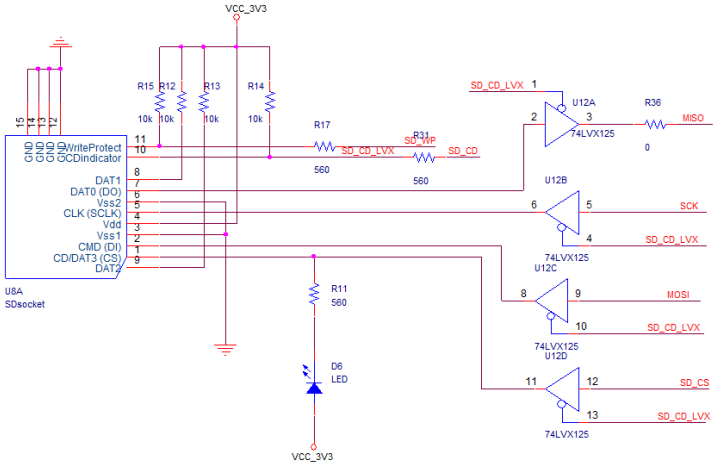


Obrázek 4.4: Zapojení převodníku USB-UART

4.1.6 SD karta

Jelikož SD karta pracuje s napětím 3,3V a procesor desky s napětím 5V bylo potřeba vyřešit, jak upravit úrovně tak, aby vyšší napětí nezničilo SD kartu. Jako rozumné řešení se jevilo použití integrovaného obvodu 74LVX125, jenž je pro tyto aplikace určen. Jeho vstupy jsou schopné odolat až 7V při napájení pouze 3,3V. Další výhodou je přítomnost povolovacího vstupu, který v případě vyjmutí karty odpojí signály jdoucí k SD kartě. PullUp rezistory R12 a R13 mají ošetřit nevyužité vstupy karty.

Zapojení signálů na piny procesoru je v následující tabulce: 4.3.



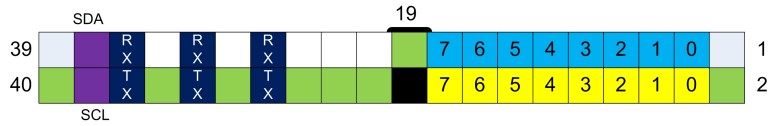
Obrázek 4.5: Zapojení SD karty

SIGNAL	PIN CPU
MISO	PB3
MOSI	PB2
SCK	PB1
SD_CS	PE7
SD_CD	PE6
SD_WP	PE5

Tabulka 4.3: Připojení signálů na CPU

4.1.7 Sloty

Deska disponuje dvěma sloty pro periferie. Rozmístění signálů v těchto konektorech je vidět na obrázku 4.6. V konektoru se nepřenáší napájení, neboť deska počítá s použitím externího zdroje pro napájení periferií. Spínaný zdroj popsaný v kapitole redukce umožňuje zajistit dostatečné napájení pro základní moduly a v případě potřeby je možné navrhnout libovolně silný zdroj pro konkrétní aplikaci.



Obrázek 4.6: Rozložení signálů v konektoru. (pohled shora)

Světle modrou a žlutou barvou jsou naznačeny IO porty procesoru. Zelenou barvu mají piny, jenž jsou připojeny na GND, fialovou pak sběrnice I^2C a tmavě modrou mají UARTY. Černou barvou je naznačen zámek konektoru. Detailní popis rozvedení pinů procesoru na konektory je v následujících tabulkách.

nc	SDA	RX1	nc	RX3	nc	RX2	nc	nc	nc	gnd	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	nc
gnd	SCL	TX1	gnd	TX3	gnd	TX2	gnd	gnd	gnd	nc	PK7	PK6	PK5	PK4	PL3	PL2	PL1	PL0	gnd

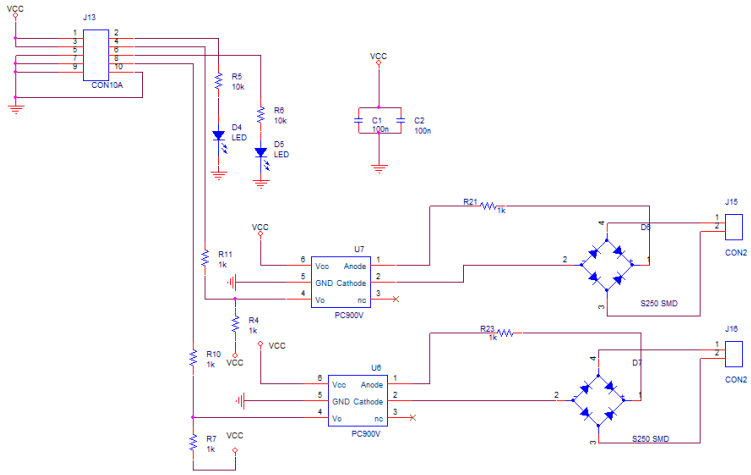
Tabulka 4.4: Piny vyvedené do slotu 1

nc	SDA	RX1	nc	RX3	nc	RX2	nc	nc	nc	gnd	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	nc
gnd	SCL	TX1	gnd	TX3	gnd	TX2	gnd	gnd	gnd	nc	PK7	PK6	PK5	PK4	PL3	PL2	PL1	PL0	gnd

Tabulka 4.5: Piny vyvedené do slotu 2

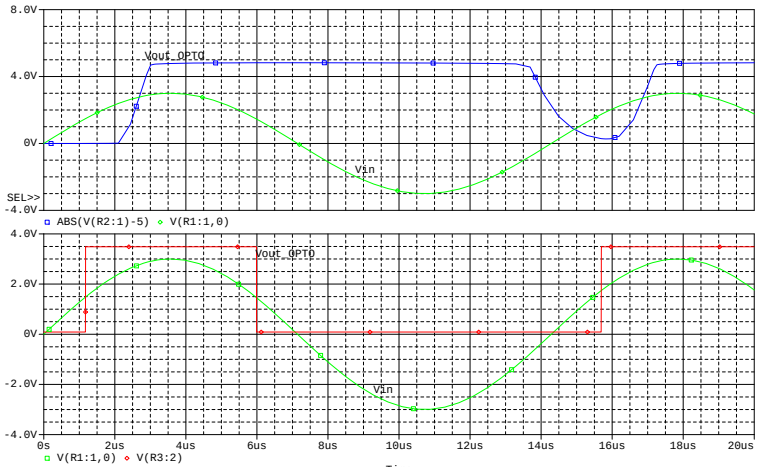
4.2 Input modul

Modul vstupů obsahuje dva nezávislé vstupy, které jsou schopné pracovat se střídavým i stejnosměrným napětím. Vstupní napětí je přivedeno přes usměrňovací můstek a sériový odpor R21 na vstup optočlenu PC900 [24] firmy SHARP, který zajišťuje galvanické oddělení vstupu od elektroniky desky. V pouzdře je integrován zesilovač a komparátor s hysterezí. Tím je zajištěno striktní překlápění výstupu z LOG 1 do LOG 0. Obvod by měl být schopen pracovat s frekvencí až 130kHz. Volbou odporu R21 je možné upravit rozsah spínání. Při velikosti odporu 1000Ω překlápí obvod do LOG 0 při napětí 4V a je schopen na vstupu snést dlouhodobě až 30V. Zapojení modulu vstupů je na následujícím obrázku 4.7. Pohledy na plošný spoj s rozmístěním součástek a fotky osazených modulů jsou umístěny v příloze pod písmenem E.



Obrázek 4.7: Zapojení vstupní desky

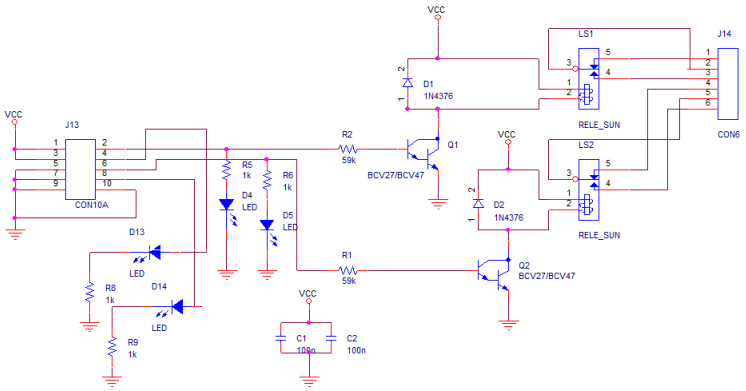
Na obrázku 4.8 je porovnání chování optočlenu s komparátorem a bez komparátoru, na vstupní frekvenci 70kHz. V horní části obrázku (modrý průběh) je vidět odezva optočlenu bez integrovaného zesilovače a komparátoru. Z průběhu je patrné, že tranzistor v optočlenu nestíhá zavírat a optočlen je stále otevřen. Ve spodní části je odezva na stejný signál u optočlenu s integrovaným zesilovačem a komparátoru. Oba tyto průběhy jsou výsledkem simulace, která je uložena v adresáři SIMULACE na CD.



Obrázek 4.8: Porovnání výstupní odezvy optočlenů

4.3 Output modul

Modul výstupů obsahuje dva nezávislé výstupy. Výstup je opatřen relé s přepínacím kontaktem, jenž má spínací schopnost 250V/10A. Spoje u kontaktů relé jsou realizovány na spodní straně plošného spoje, tím je zajištěna snadná opravitelnost v případě vypálení následkem přetížení. Izolační vzdálenost je mezi všemi výkonovými spoji minimálně 3mm. Tato hodnota splňuje zesílenou izolační pevnost minimálně 500V a funkční pro 1000V dle ČSN EN 60950. Šířka spoje je zvolena 60mil což je 1,524mm, tato hodnota by měla při tloušťce mědi 35μm a oteplení 50° snést min. 8A viz[29]. Hodnoty oteplení jsou stanoveny pro materiál plošného spoje FR4. Zapojení je na obrázku 4.9. Pohledy na plošný spoj a fotky osazených modulů lze nalézt v příloze pod písmenem E.

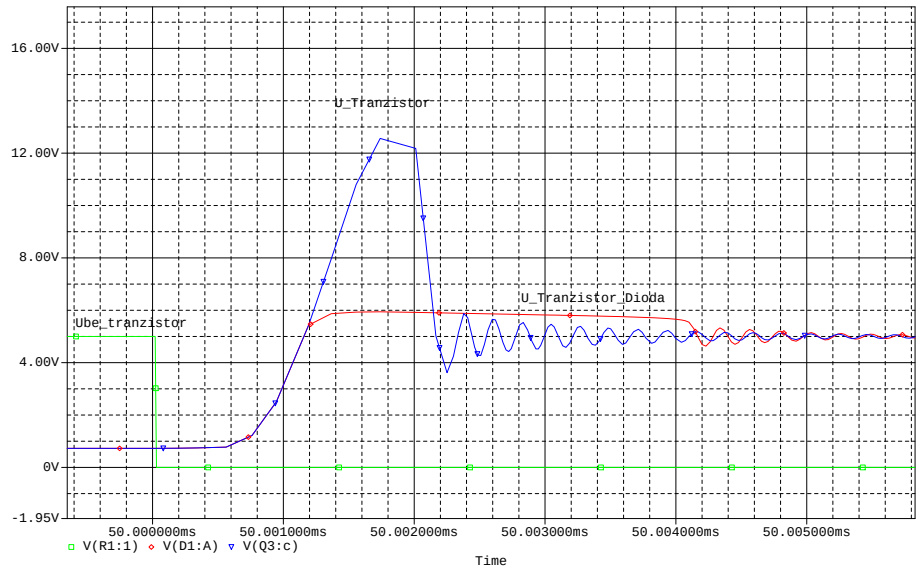


Obrázek 4.9: Zapojení výstupní desky

Pro spínání cívky relé jsou použity darlingtonovy tranzistory BCV27 nebo BCV47. Tranzistory by měly zajistit rychlé sepnutí a tím minimalizovat poškození kontaktů. Jako ochrana před přepětím, které vzniká na cívce při rozpínání, je použita dioda 1N4376, přes kterou se toto napětí uzavře a nehrozí tak poškození spínacích tranzistorů. Velikost vzniklého

napětí je úměrné energii naakumulované v cívce a ta odpovídá vztahu $\frac{1}{2}L * i^2$.

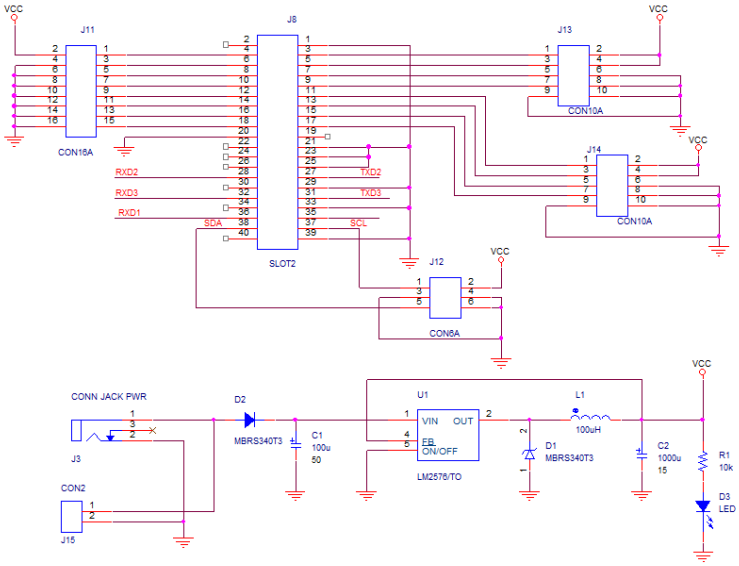
Přibližný průběh napětí na tranzistoru při zavírání je na obrázku 4.10. Průběh D_Tranzistor znázorňuje napětí Uce na tranzistoru bez použití ochranné diody, průběh D_Tranzistor_Dioda je napětí Uce na tranzistoru s ochrannou diodou.



Obrázek 4.10: Průběhy napětí na tranzistoru při zavírání

4.4 Redukce

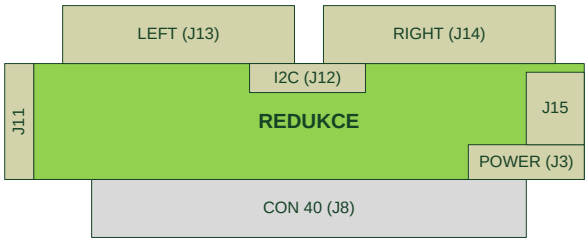
Redukce obsahuje pouze spínaný zdroj a je koncipována jako doplněk desky, který v případě potřeby může sloužit jako napájecí zdroj pro energeticky náročné periferie. Hlavním úkolem redukce je rozdělit velký 40 pinový konektor na menší konektory, které jsou vhodnější pro připojení jednodušších periferií. Zapojení redukce je na následujícím obrázku 4.11.



Obrázek 4.11: Zapojení reduce se zdrojem

Navržená redukce obsahuje spínaný zdroj s výstupním proudem 3A. Zdroj je realizován obvodem LM2576 [9], jedná se o spínaný step down regulátor (BUCK) pracující na frekvenci 52kHz. Pro svoji činnost vyžaduje pouze 4 externí součástky. LM2576 má pevné výstupní napětí 5V. Regulátor je zapojen dle doporučení výrobce a je umístěn nastojato na DPS v pouzdře TO220. K pouzdru je možné přišroubovat chladič pro lepší odvod tepla z pouzdra do okolí.

Rozmístění signálů na konektorech redukce při připojení do slotu 1 a slotu 2 je v následujících tabulkách. Na obrázku 4.12 je zobrazeno rozložení konektorů v redukci.



Obrázek 4.12: Rozmístění konektorů na redukci

Tabulka vždy uvádí názvy pinů vyvedených do konektorů při vložení do příslušného slotu. Pohled na konektor je shora a číslování je po sloupcích: tedy první sloupec 1, 2 druhý sloupec 3 a 4 atd.

SLOT1	PL0	PL1	PL2	PL3	GND
	GND	GND	GND	Vcc	Vcc
SLOT2	PK0	PK1	PK2	PK3	GND
	GND	GND	GND	Vcc	Vcc

Tabulka 4.6: Signály v konektoru J13 LEFT

SLOT1	PK4	PK5	PK6	PK7	GND
	GND	GND	GND	Vcc	Vcc
SLOT2	PL4	PL5	PL6	PL7	GND
	GND	GND	GND	Vcc	Vcc

Tabulka 4.7: Signály v konektoru J14 RIGHT

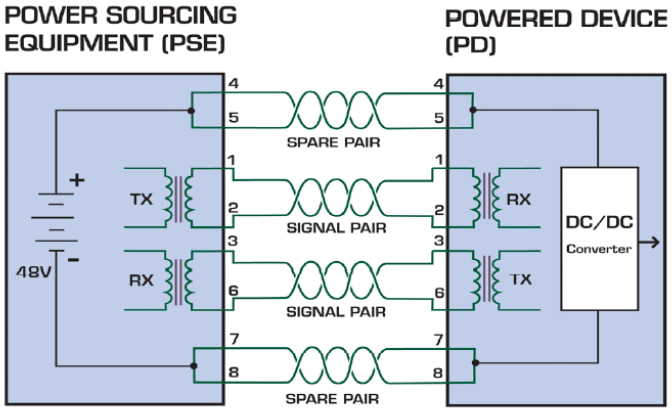
SLOT1	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7
	GND	GND	GND	GND	GND	GND	GND	Vcc
SLOT2	PA0	PA1	PA2	PA3	PA4	PA5	PA6	PA7
	GND	GND	GND	GND	GND	GND	GND	Vcc

Tabulka 4.8: Signály v konektoru J11

4.5 Ethernet

Modul Ethernetu komunikuje s hlavní deskou po sběrnici SPI a je realizován pomocí řadiče ENC424J600 [10] od firmy Microchip. ENC424J600 je Ethernet Controller s SPI a paralelním rozhraním. Řadič dokáže pracovat rychlostí 10/100 Mb/s. Obvod pracuje s napájecím napětím 3,3V, proto je zde na rozhraní SPI použit integrovaný obvod 74LVX125, který stejně jako u SD karty upravuje úrovně rozhraní.

Modul ethernetu je navržen tak, aby procesorová deska šla napájet pomocí pasivního POE. Tato vlastnost může být využita v případě umístění desky na nepřístupné místo bez zdroje elektrické energie. Napájení je do desky přivedeno po nevyužitých párech v ethernetu. Standart tohoto pasivního POE je popsán v IEEE 802.11af a schematicky rozkreslen v obrázku 4.13.



Obrázek 4.13: Pasive Power Over Ethernet (převzato z [13])

Schéma zapojení celého modulu ethernetu je v příloze pod písmenem F.

Rozpis signálů jdoucích do modulu ethernet je v následující tabulce 4.9. Tyto signály jsou vyvedeny v konektoru J5 na procesorové desce a konektor je textově označen jako ethernet. Signál RESET_NET je přiveden na enable vstup stabilizátoru a umožňuje tak vypnutí řadiče ethernetu. Řadič ethernetu obsahuje několik typů resetů a všechny jsou ovládané příkazy vysílanými po SPI rozhraní. Vypnutím a zapnutím napájení řadiče se nechá vyvolat powerON reset, který by měl sloužit v případě největší nouze.

4.5.1 Napájení

Jako zdroj napájení pro řadič ENC424J600 je použit integrovaný stabilizátor LT1616 [14] firmy Linear technology. Stabilizátor pracuje s proudem až 600mA. Spínací frekvence regulátoru je 1,5MHz, proto pro svoji činnost potřebuje cívku o indukčnosti pouze 10μH. Stabilizátor je zapojen dle doporučení výrobce a měl by v použité konfiguraci dávat výstupní proud 300mA. Spotřeba uvedená výrobcem pro ENC424J600 je 300mA.

SIGNAL	PIN CPU
MISO	PB3
MOSI	PB2
SCK	PB1
SS	PE7
NET_INT	PE6
RESET_NET	PE5

Tabulka 4.9: Připojení signálů na CPU

4.6 Výroba DPS

Desky plošných spojů byly navrženy v prostředí ORCAD 16.2. Projekty z Orcadu jsou uloženy v adresáři PODKLADY na CD. Pro návrh byly vytvořeny padstacky a footprinty jednotlivých součástek a tyto jsou uloženy v adresáři ORCAD. Výrobu DPS provedla firma PragoBoard [21]. Vygenerované soubory pro výrobu jednotlivých desek jsou na CD v adresáři výroba. V něm se nacházejí soubory pro souřadnicovou vrtačku a soubory jednotlivých vrstev pro fotoploter.

Desky mají nejmenší vzdálenost mezi spoji 0,21mm a šířku vodiče také 0,21mm, což by odpovídalo třídě přesnosti 5. Je potřeba však podotknout, že v současné době neexistuje v České republice předpis, který by třídy přesnosti definoval. Z tohoto důvodu jsou vzdálenosti mezi spoji a šířky spojů omezeny pouze možnostmi výrobce DPS a cenou.

Kapitola 5

Software

5.1 Prostředí

Pro psaní knihoven bylo použito vývojové prostředí AVRstudio4 [4], které je po registraci ke stažení zdarma na stránkách Atmelu. Základní AVRstudio, které je dostupné u Atmelu, nemá překladač jazyka C a po nainstalování je možné programovat pouze v assebleru. AVRstudio počítá s doinstalováním GNU GCC překladače s názvem WINAVR. WINAVR lze stáhnout zdarma ze stránek sourceforge.net [28]. Po instalaci se překladač integruje do prostředí AVRstudia a je možné pohodlně programovat v C.

Druhou možností pro programování mikroprocesorů firmy ATMEL je použít nějaký placený nástroj, jako je mikroC PRO pro AVR 2009 [16]. Tento nástroj disponuje celou řadou knihoven a simulátorem. Jeho cena je však 199$. Na stránkách mikroElektronika je ke stažení DEMO, které je omezeno velikostí programu na 2kB. Při psaní v tomto prostředí je důležité mít na paměti, že mikroC disponuje makry jako je stebit atp. Tyto makra však pro WINAWR nejsou definována, proto program napsaný v prostředí mikroC nemusí jít přeložit pomocí WINAVR.

K programování byl použit programátor s rozhraním STK500 [26] ve své první verzi, který využívá rozhraní ISP. Programátor neumožňuje On-Chip debug, komunikuje s procesorem po SPI pomocí signálů MISO, MOSI, SCK a RESET.

5.2 Knihovny funkcí

Pro použití desky byly vytvořeny knihovny funkcí, které ovládají jednotlivé periferie na desce. Pro tvorbu těchto knihoven byly použity zdroje z AVRfeaks konkrétně [23] a manuály výrobců pamětí a RTC.

5.2.1 UART0

Knihovna pro komunikaci prostřednictvím seriové linky UART0. K sériové lince je na desce připojen převodník FT232B, který převádí UART na USB. Knihovna je ve dvou verzích UART0_routines a UART0_routines_i.

- UART0_routines - nepoužívá pro příjem dat přerušení procesoru.
- UART0_routines_i - používá pro příjem dat přerušení procesoru.

5.2.1.1 UART0_routines

Konstanty pro nastavení UARTU jsou spočítány pro rychlost komunikace 230400b/s, 8 datových bitů, žádná parita a 1 stopbit při použití krystalu 14,7456MHz. Konstanty jsou nastavovány ve funkci UART0_init. V případě potřeby změny rychlosti komunikace se musí přepočítat konstanty a přepsat v inicializaci.

void UART0_init(void) Inicializuje UART0 nastaví přenosovou rychlost, počet datových bitů a počet stopbitů.

unsigned char receiveByte(void) Funkce přijímá byte z UART0. Na byte se aktivně čeká. Čekání je provedeno testováním bitu RXC0 v registru UCSR0A. Bit RXC0 říká, že v registru UDR0 je přijatý byte. Návratovou hodnotou funkce je přijatý byte.

void transmitByte(unsigned char data) Funkce odešle byte, který je jejím parametrem. Ve funkci se aktivně čeká dokud není nastaven příznak UDRE0 v registru UCSR0A. Příznak říká, že je vysílací buffer prázdný. V případě, že je vysílací buffer prázdný, jsou data zapsána do UDR0. Funkce nevrací žádnou hodnotu.

void transmitString_F(char* string) Funkce přenáší řetězec, který je uložen v paměti programu. Adresa řetězce je předána v parametru funkce. Ve funkci se provádí čtení z programové paměti pgm_read_byte() a přenos transmitByte(). Návratová hodnota funkce je prázdná.

void transmitString(unsigned char* string, unsigned char length) Funkce přenáší řetězec o délce length, který je uložen v paměti RAM. Adresa řetězce je předána v parametru funkce. Ve funkci se volá transmitByte(). Návratová hodnota funkce je prázdná.

void transmitHex(unsigned char dataType, unsigned long data) Funkce převede data na hexadecimální tvar a přenese v podobě ASCII. Parametr dataType specifikuje délku dat ve znacích, která se vytvoří z druhého parametru data. Možné délky jsou 2,4,8. Převod se provádí dělením mod16 a pak přenosem. Před přenosem je ke každému číslu 0-9 přičteno 0x30, případně 0x41 pro písmeno a-f. Počet opakování dělení a přenosu je podle délky dat 2,4,8. Funkce nevrací žádnou hodnotu.

5.2.1.2 UART0_routines_I

Knihovna, která používá pro příjem dat přerušení. Přijatá data jsou ukládána do kruhového bufferu typu FIFO. Délka tohoto bufferu je nastavena na 64byte parametrem BUFF_SIZE. Rychlost komunikace je stejná jako u knihovny bez přerušení.

void UART0_init(void) Inicializuje UART0 nastaví přenosovou rychlost, počet datových bitů a počet stopbitů. Funkcí je dále nastaveno přerušení na přijímaná data.

ISR(USART0_RX_vect) Obsluha přerušení, která se volá v případě, že přijímač sériové linky přijal data. Funkce přečte data z registru UDR0 a uloží je do kruhového bufferu. Pokud je kruhový buffer plný, jsou data ztracena.

void transmitByte(unsigned char data) Funkce odešle byte, který je jejím parametrem. Ve funkci se aktivně čeká, dokud není nastaven příznak UDRE0 v registru UCSR0A. Příznak říká, že je vysílací buffer prázdný. V případě, že je vysílací buffer prázdný, jsou data zapsána do UDR0. Funkce nevrací žádnou hodnotu.

void transmitHex(unsigned char dataType, unsigned long data) Funkce převede data na hexadecimální tvar a přenese v podobě ASCII. Parametr dataType specifikuje délku dat ve znacích, která se vytvoří z druhého parametru data. Možné délky jsou 2,4,8. Převod se provádí dělením mod16 a pak přenosem. Před přenosem je ke každému číslu 0-9 přičteno 0x30, případně 0x41 pro písmeno a-f. Počet opakování dělení a přenosu je podle délky dat 2,4,8. Funkce nevrací žádnou hodnotu.

void transmitString_F(char* string) Funkce přenáší řetězec, který je uložen v paměti programu. Adresa řetězce je předána v parametru funkce. Ve funkci se provádí čtení z programové paměti pgm_read_byte() a přenos transmitByte(). Návratová hodnota funkce je prázdná.

void transmitString(unsigned char* string, unsigned char length) Funkce přenáší řetězec, který je uložen o délce length v paměti RAM. Adresa řetězce je předána v parametru funkce. Ve funkci se volá transmitByte(). Návratová hodnota funkce je prázdná.

unsigned char getChar_uart0buff(void) Funkce provádí výběr dat z bufferu. Není-li buffer prázdný, přečte funkce první zapsaný znak a posune ukazatel na následující znak. Pokud je buffer prázdný vrátí funkce 0.

unsigned char getSize_uart0Buff(void) Vrací aktuální počet znaků v bufferu. Maximální počet je BUFF_SIZE.

5.2.2 serial LCD

Jedná se knihovnu, která zajišťuje komunikaci s modulem LCD. Funkce obstarávají přenos příkazů a dat do řadiče LCD. Pro komunikaci se používá UART2. Rychlost komunikace je nastavena na 19200b/s, žádná parita a jeden stopbit.

5.2.2.1 UART2_LCD

void UART2_LCD_init(void) Inicializuje UART pro komunikaci s LCD, nastaví rychlost komunikace, počet datových bitů a stopbitů.

unsigned char receiveByteLCD(void) Přijme byte od LCD. Příjem je stejný jako u receiveByte() v knihovně UART0_routines.

void transmitByteLCD(unsigned char data) Přenese byte, který je parametrem do LCD. Pozor samostatné použití funkce transmitByteLCD() nezajistí zobrazení znaku na LCD, funkce slouží pouze pro přenos dat a příkazů do řadiče displeje.

void LCD_clear(void) Provede smazání celého LCD. Před provedením další operace s displejem je nutné alespoň 5mS počkat, jinak dojde ke ztrátě dat, která mají být po smazání zobrazena. Kurzor se po smazání displeje nastaví na levý horní roh.

void LCD_rowClear(unsigned char row) Smaže zadaný řádek na LCD a nastaví kurzor na první pozici mazaného řádku. Parametr row může být v rozsahu 0-7.

void LCD_displayByte(unsigned char data) Funkce zobrazí data zadaná parametrem na LCD. Funkce je vhodná pro kreslení obrázků na LCD.

void LCD_setPossX(unsigned char poss) Nastaví kurzor na zadanou horizontální pozici. Parametr poss je v rozsahu 0 - 127.

void LCD_setRow(unsigned char row) Přesune kurzor na zadaný řádek. Kurzor je po provedení příkazu na začátku řádku.

void LCD_displayChar(unsigned char data) Zobrazí na displeji znak předaný jako parametr a posune kurzor na další pozici pro zápis dalšího znaku.

void LCD_displayCharR(char data) Zobrazí ASCII znak na LCD a přesune kurzor zpět na jeho pozici.

void LCD_displayCharRR(char data) Zobrazí znak ASCII a přesune kurzor o znak před zapsaný. Funkce je vhodná pro psaní zprava doleva.

void LCD_backspace(void) Smaže naposledy zapsaný znak (funkce backspace).

void LCD_displayCharE(unsigned char data) Zapíše na displej znak, který je uložen v EEPROM LCD řadiče. Kurzor je na konci znaku.

void LCD_displayCharER(unsigned char data) Zapíše na displej znak, který je uložen v EEPROM LCD řadiče a přesune kurzor zpět na jeho pozici.

void LCD_displayCharERR(unsigned char data) Zapíše na displej znak, který je uložen v EEPROM LCD řadiče a přesune kurzor o pozici před zapsaný znak, psaní zprava doleva.

void LCD_displayString_F(char* string) Zobrazí na displeji řetězec uložený v paměti flash. Pracuje stejně jako transmitStringF u UARTU.

void LCD_displayString(unsigned char* string, unsigned char length) Zobrazí na displeji řetězec o délce length uložený v paměti RAM.

LCD_cursorBegin Přesune kurzor zpět na začátek aktuálního řádku.

LCD_cursorHome Přesune kurzor do pravého horního rohu.

void LCD_showLOGO1(void) Zobrazí na dipleji LOGO1, které je uložené v řadiči displeje.

void LCD_showLOGO1(void) Zobrazí na dipleji LOGO2, které je uložené v řadiči displeje.

5.2.3 Výstupy

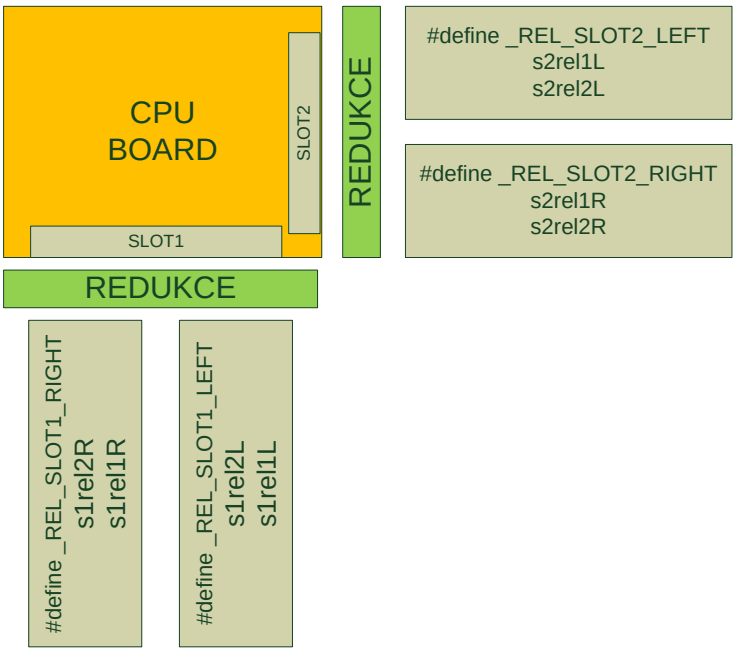
Pro ovládání relé se nechá použít knihovna otput_routines.h. Relé jsou rozdělena do sekcí podle toho, které místo v redukci a slotu je obsazeno. Na obrázku 5.1 jsou vidět jména sekcí, které je potřeba definovat pro práci s výstupy. Výstupní moduly by se měly připojovat do pozic SLOT1_LEFT a SLOT2_RIGHT, kde je vyvedena brána PL, která nemá přerušení.

5.2.3.1 output_routines

Obsahuje funkce SET pro sepnutí relé a funkce RESET pro rozepnutí relé. Funkce se volá ve tvaru set_s1rele1L, kde s1 nebo s2 znamená slot, rel1 nebo rel2 relé na výstupním modulu a poslední písmeno označuje pozici modulu v redukci - levou nebo pravou. Sepnutí relé provede funkce Set a rozepnutí funkce Res.

5.2.4 Vstupy

Knihovna obsluhující vstupní modul nastaví přerušení na bráně PK pro opticky oddělené vstupy. Brána PK je vyvedena pouze do slot1_right a slot2_left viz 5.1, proto při použití přerušení musí být modul vstupů zapojen v jednom z těchto dvou slotů. Pro použití je potřeba dopsat funkci ISR(), která se stará o obsluhu přerušení. Ukázka funkce ISR je ve zdrojovém souboru input_routines.



Obrázek 5.1: Názvy výstupních rutin pro relé

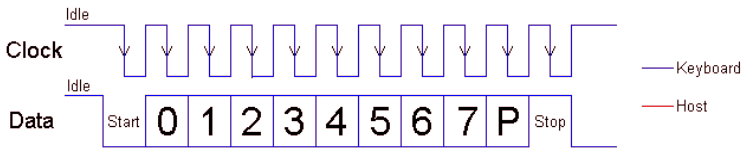
5.2.4.1 input_routines

void input_init(unsigned char vstupy) Inicializuje bránu PK jako vstup a zapne na ní přerušení. Maska přerušení je nastavena dle parametru vstupy.

ISR(PCINT2_vect) Ukázka funkce ISR pro obsluhu přerušení, která se volá při každé změně na vstupu. Obsluha přerušení se nyní provede změnou stavu led diod LED1 a LED2 na vstupním modulu.

5.2.5 Klávesnice

Komunikace s klávesnicí je obousměrná, pro přenos dat se používají dva vodiče KB_dat a KB_clk. V knihovně klávesnice je však implementován pouze příjem scankódů z klávesnice a následné dekódování. Komunikace směrem ke klávesnici slouží pro rozsvěcení diod Num-Lock, CapsLOck, ScrollLock a nastavení parametrů, jako je použití bufferu, selftest atd. Pro jednoduché použití klávesnice není potřeba komunikovat směrem ke klávesnici.



Obrázek 5.2: Přenos dat z klávesnice převzato z [8]

Data z klávesnice jsou posílána v 11bitech. První bit je start bit, pak následuje 8datových bitů a 1paritní bit, poslední je stop bit. Přenos začíná se sestupnou hranou signálu KB_clk. Přenášený byte je zabezpečen lichou paritou.

Pro uživatelskou obsluhu klávesnice stačí dvě rutiny. První se klávesnice inicializuje a druhou se vybírají data z bufferu. Čtení bytů přijatých z klávesnice provádí přerušení, které se volá při změně signálu KB_clk.

5.2.5.1 scancodes

Obsahuje pole s přijatými kódy od klávesnice a jejich ASCII reprezentací. Pole se scankódy musí být rozděleno na shifted a unshifted, protože velké a malé písmeno se pozná jen podle toho, že je před velkým písmenem vysílán scancode klávesy shift.

5.2.5.2 kb_routines

init_kb(void) Inicializuje signály pro klávesnici a nastaví přerušení PCINT0.

unsigned char get_kbbuff(void) Funkce provádí čtení aktuálního bytu z bufferu klávesnice. Pokud je buffer prázdný, vrací 0x00, jinak vrací obsah aktuální pozice v bufferu. Po přečtení je posunut ukazatel na následující znak.

void put_kbbuff(unsigned char data) Vloží byte data do volné pozice na konci bufferu.

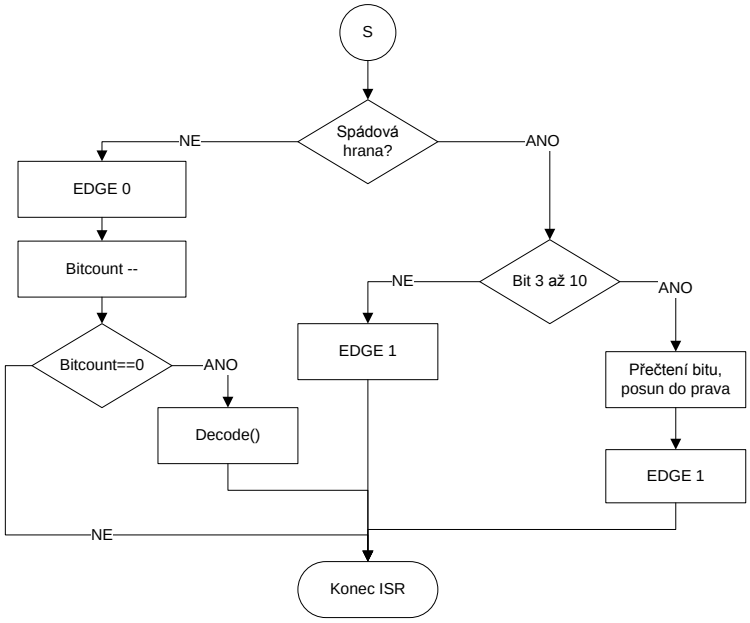
ISR(PCINT0) Funkce se volá při příchodu přerušení od klávesnice. Přerušení je generováno na náběžnou i spádovou hranu signálu KB_clk. Pro rozpoznání hrany slouží proměnná edge. Je-li edge=0 čeká se na spádovou hranu, je-li edge=1, je očekáván příchod náběžné hrany. Vývojový diagram obsluhy přerušení je na obrázku 5.3. Po inicializaci je nastaven edge na 0 a bitcount na 11. Při příchodu přerušení od klávesnice se provede kontrola hrany, je-li spádová a počet bitů je v intervalu <3,10>, provede se posun a čtení bitu z KB_dat, pak se nastaví edge na 1. Po příchodu náběžné se edge nastaví na 0 a dekrementuje se bitcount. Na náběžnou hranu se provádí kontrola počtu přijatých bitů, je-li bitcount==0, je zavolána funkce decode, která provede dekódování znaku na ASCII. Výsledný ASCII znak je uložen do bufferu. Pokud je buffer plný je byte zahozen.

void decode(unsigned char sc) Provádí dekódování scancodu na znak ASCII. Funkce nevrací žádnou hodnotu, dekódovaný znak je uložen do bufferu.

Pro dekódování je nutné vědět, jak jsou data z klávesnice přenášena.

Pokud je na klávesnici stisknuto tlačítko, vysílá klávesnice jeho scancode, po uvolnění klávesy vysílá klávesnice 0xF0 a opět scancode uvolněné klávesy. V případě psaní velkého písmena je nejdříve stisknuta klávesa shift a následně písmeno, uvolňování probíhá v opačném pořadí tedy uvolněno písmeno a pak shift. Ukázky odeslaných kódů v případě stisknutí a uvolnění tlačítka jsou v následující tabulce.

V případě, že je tlačítko stisknuto a není uvolněno do doby nastavené v řadiči klávesnice, začne klávesnice opakovat scankod tlačítka, dokud nedojde k jeho uvolnění. Rychlost



Obrázek 5.3: Diagram obsluhy přerušení od klávesnice

Akce	**vysílané scancody**
stisknutí a uvolnění 'a'	0x1C 0xF0 0x1C
napsání 'A'	0x12 0x1C 0xF0 0x1C 0xF0 0x12

Tabulka 5.1: Příklady vysílaných scancodu

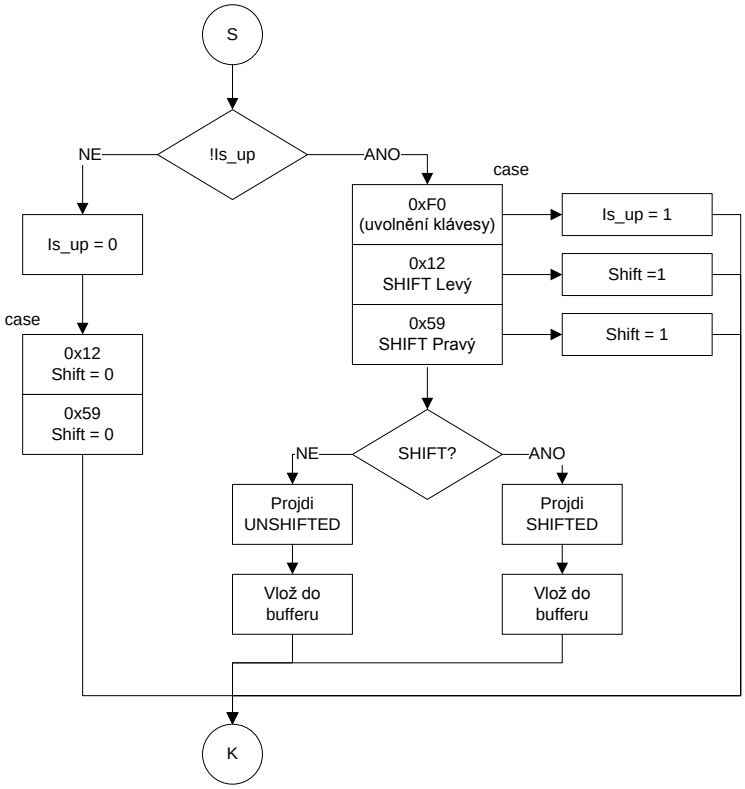
opakování je nastavena parametrem v řadiči klávesnice. Přijímaná sekvence pak může vypadat takto: 0x1C delay 0x1C 0x1C 0x1C 0x1C 0xF0 0x1C.

Dekódování se provádí podle následujícího schématu 5.4.

Vývojový diagram dekódování scankódu na ASCII znak je na obrázku 5.4. Po inicializaci je příznak is_up nastaven na 0. Příznak říká, že nebyla přijata sekvence 0xF0 (uvolnění klávesy). Při stisku tlačítka se bude procházet pravá půlka diagramu. Dojde k nastavení příznaku shift nebo se provede dekódování znaku. Pokud je přijat scancode 0xF0 (uvolnění klávesy) je nastaven příznak is_up na jedna. Při příštím přijetí bytu se prochází levá půlka diagramu. V levé půlce dojde k porovnání scankódu se scan kódy shift kláves, pokud je nalezena shoda, je nastaven příznak shift na 0 (uvolněna klávesa shift). Není-li nalezena shoda, byla uvolněna jakákoliv jiná klávesa a je pouze nastaven příznak is_up na 0. Příznak shift nese informaci o tom, jestli se budou psát velká nebo malá písmena.

5.2.6 I^2C

Jedná se o dvouvodičovou sběrnici, jenž pracuje s rychlostí max. 400kHz vyjímečně 1MHz, z tohoto ohledu lze I^2C zařadit mezi pomalé sběrnice. Na této desce je nastaven režim, kdy procesor je master a generuje signál SCL, kterým se řídí přenos.



Obrázek 5.4: Dekódování scancodů přijímaných z klávesnice

5.2.6.1 i2c_routines

Knihovna obsahuje funkce pro master komunikaci se zařízeními na sběrnici.

void i2c_init(void) Inicializuje I^{2}C sběrnici na procesoru a nastaví rychlost komunikace. Procesor je nastaven tak, že je masterem na sběrnici. Procesor vždy zahajuje komunikaci se slave zařízením, a proto je zbytečné používat na příjem dat přerušení.

Nastavení I^{2}C je v registru TWCR, který je v tomto případě nastaven na 0x44. Tedy zapnuté I^{2}C a posílání ACK po přijmu dat. Dále je nastavena rychlost komunikace pomocí dvou registrů TWBR a TWSR. Registr TWSR obsahuje nastavení předděličky. V tomto případě je předdělička nastavena na 0, registr kromě nastavení obsahuje stavové bity I^{2}C. TWBR celý obsahuje dělící konstantu, která je nastavena na 0x0A. Tyto konstanty lze spočítat dosazením do vztahu:

$$SCLfreq = \frac{Fcpu}{16 + 2 * 16 * TWBR * 4^{TWPS}}$$

Rychlost komunikace je tedy nastavena na 409600Hz, pokusně bylo zjištěno, že se s RTC nechá komunikovat i na frekvenci 819200Hz. Ta více než dvakrát převyšuje údaj udaný výrobcem RTC. S EEPROM je situace horší při zvýšení frekvence komunikace přestane jako první fungovat zápis do paměti a je tedy potřeba dodržet doporučení výrobce a komunikovat s pamětí maximálně rychlostí 400kHz.

unsigned char i2c_start(void) Odešle startovací byte na I^2C. Vrací 0, pokud se odeslání povedlo, jinak vrací 1.

unsigned char i2c_repeatStart(void) Odešle opakovaný start na I^2C. Vrací 0, pokud se odeslání povedlo, jinak vrací 1.

unsigned char i2c_sendAdress(unsigned char address) Odešle adresu zařízení, se kterým se bude komunikovat. Adresa zařízení je předána jako parametr. Funkce čeká dokud nepřijde od přijímajícího zařízení odpověď. Vrací 0, pokud se přenos povedl, jinak vrací 1.

unsigned char i2c_sendData(unsigned char data) Funkce odesílá data, která jsou předána jako parametr, a čeká dokud nepřijde potvrzení od přijímacího zařízení. Vrací 0, pokud se přenos povedl, jinak vrací 1.

unsigned char i2c_receiveData_ACK(void) Funkce přijímá data od slave zařízení, po přijetí pošle zpět pozitivní potvrzení. Návratovou hodnotou funkce jsou přijatá data, po přijetí dat je dobré zavolat funkci i2c_statusACK(), která ověří jestli se příjem povedl.

unsigned char i2c_receiveData_NACK(void) Funkce přijímá data od slave zařízení, po přijetí posílá zpět negativní potvrzení. Návratovou hodnotou funkce jsou přijatá data, po přijetí dat je dobré zavolat funkci i2c_statusNACK(), která ověří jestli se příjem povedl.

void i2c_stop(void) Odešle příkaz STOP po I^2C a zastaví tak komunikaci se slave zařízením na sběrnici.

unsigned char i2c_statusNACK(void) Zkotroluje registr TWSR a pokud je nastaven NACK, vrací funkce 0, jinak vrací 1. Používá se po funkci i2c_receiveData_NACK() pro ověření správnosti příjmu dat.

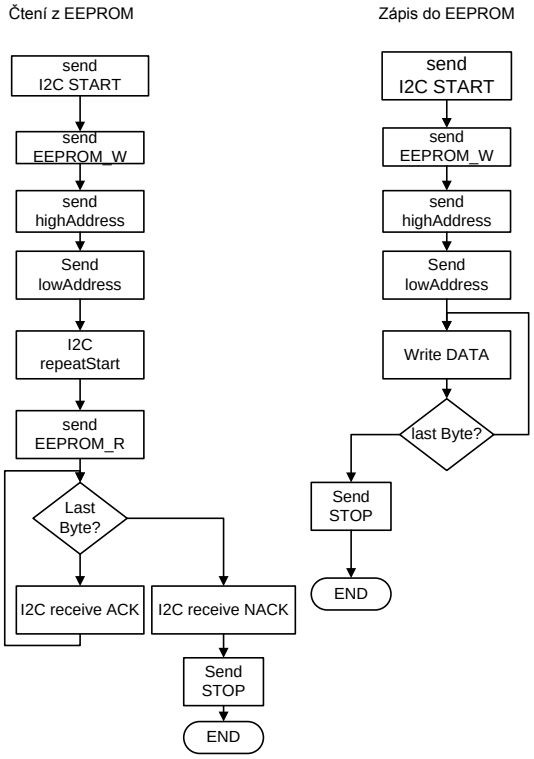
unsigned char i2c_statusACK(void) Zkotroluje registr TWSR a pokud je nastaven ACK, vrací funkce 0, jinak vrací 1. Používá se po funkci i2c_receiveData_ACK() pro ověření správnosti příjmu dat.

5.2.7 EEPROM

Komunikace s EEPROM se provádí po I^2C sběrnici. Na této desce je adresa EEPROM paměti 0xa0 pro zápis a 0xa1 pro čtení z paměti. Tyto konstanty pro komunikaci lze upravit v souboru EEPROM_routines.h

5.2.7.1 EEPROM_routines

Knihovna se používá pro komunikaci s EEPROM 24LC256 a pro svoji činnost potřebuje I2C_routines.h. Knihovna může být použita i pro jiný typ paměti používající stejné schéma komunikace po I^2C. V hlavičkovém souboru jsou definovány adresy paměti. Diagram komunikace pro čtení a zápis do EEPROM je na obrázku 5.5.



Obrázek 5.5: Diagram komunikace EEPROM

unsigned char* EEPROM_read(unsigned char highAddress, unsigned char lowAddress, unsigned char totalChar) Přečte zadaný počet bytů z EEPROM a vrátí ukazatel na pole s přečtenými daty. Maximum přečtených bytů je nastaveno na 64. Pokud se čtení nepovede vrací fukce 0.

unsigned char EEPROM_write(unsigned char highAddress, unsigned char lowAddress, unsigned char* string, unsigned char length) Zapíše pole zadané parametrem string do EEPROM paměti na adresu předanou ve dvou bytech highAdres a lowAdress. Délka pole pro přenos je zadána parametrem length. Je-li zápis úspěšný vrací funkce 0.

unsigned char* EEPROM_readPage(unsigned int pageNumber) Přečte stránku paměti (64byte) z EEPROM a vrátí ukazatel na pole, kde jsou data uložena. Pokud se čtení nepovede, vrací funkce 0.

unsigned char EEPROM_writePage(unsigned int pageNumber, unsigned char* page) Zapíše stránku do paměti EEPROM, parametrem je číslo stránky a ukazatel na pole o velikosti 64byte, které má být do paměti zapsáno. Funkce vrací 0, pokud se zápis povede.

unsigned char EEPROM_erase(void) Provede smazání [1] celé EEPROM, funkce vrací 0 pokud se smazání podaří v opačném případě vrací 1. Průběh komunikace je stejný jako v případě zápisu do paměti.

void EEPROM_lock(void) Nastaví signál WP_EPR na LOG 1 a potlačí tak zápisy do paměti. Zápis do paměti EEPROM projde, i když je WP_EPR na LOG 1, ale EEPROM neprovede změny v poli s daty.

void EEPROM_unlock(void) Zruší nastavení WP_EPR na LOG 0 a povolí tak zápis do paměti.

5.2.8 RTC

Komunikace s obvodem reálného času probíhá po sběrnici I^2C. RTC má adresu 0xa2 pro zápis a 0xa3 pro čtení. Obvod reálného času obsahuje sadu registrů uchovávajících informace o čase, alarmu a nastavení obvodu. Registry jsou uvedeny v tabulce 5.6. Registry jsou při požadavku na čas všechny přečteny do procesoru a procesor vybere požadovanou informaci. Při zápisu do obvodu RTC je upravována pouze určitá sada registrů.

Při nastavování alarmu je potřeba dát pozor na nejvyšší bit. Bit povoluje generování alarmu pro danou funkci. Například je-li nastaven bit AE (Alarm Enable) u hodin, generuje se alarm pokaždé ve stejnou hodinu. Je-li nastaven AE u hodin a minut, generuje se alarm ve stanovenou hodinu a minutu. Je zakázáno používat dohromady bity AE u dnů a dnů v týdnu.

Address	Function	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
00	Control 1	TEST	0	STOP	0	TEST	0	0	0
01	Control 2	0	×	0	TI / TP	AF	TF	AIE	TIE
02	Seconds	VL	40	20	10	8	4	2	1
03	Minutes	×	40	20	10	8	4	2	1
04	Hours	×	×	20	10	8	4	2	1
05	Days	×	×	20	10	8	4	2	1
06	Weekdays	×	×	×	×	×	4	2	1
07	Months / Century	C	×	×	10	8	4	2	1
08	Years	80	40	20	10	8	4	2	1
09	Minute Alarm	AE	40	20	10	8	4	2	1
0A	Hour Alarm	AE	×	20	10	8	4	2	1
0B	Day Alarm	AE	×	20	10	8	4	2	1
0C	Weekday Alarm	AE	×	×	×	×	4	2	1
0D	CLKOUT frequency	FE	×	×	×	×	×	FD1	FD0
0E	Timer control	TE	×	×	×	×	×	TD1	TD0
0F	Timer	128	64	32	16	8	4	2	1

Obrázek 5.6: Registry RTC zdroj [11]

[1] Smazání - naplnění paměti 0xff

5.2.8.1 RTC_routines

Knihovna zabezpečuje základní komunikaci s RTC a pro svoji činnost potřebuje knihovnu I2C_routines.h. V knihovně jsou k dispozici následující funkce.

void RTC_init(void) Inicializuje komunikaci s EEPROM, provede nastavení I^2C.

void RTC_init_I(void) Inicializuje komunikaci s EEPROM, nastaví I^2C a nastaví přerušení od RTC na pin procesoru. Obsluha přerušení je ve funkci ISR(PCINT0_vect), která se volá při přerušení od RTC.

unsigned char* RTC_getTime(void) Funkce přečte registry RTC obvodu. Z registrů 02 až 04 přepočte čas do ASCII a vrátí ukazatel na pole, v kterém je čas uložen. Pole má tvar hh:mm:ss a je dlouhé 8bytů, čas je tedy uložen včetně znaků ':'.

unsigned char* RTC_getDate(void) Funkce přečte registry RTC. Z registrů 05 až 08 přepočte datum do ascii ASCII a vrátí ukazatel na pole, ve kterém je uloženo datum. Datum je ve tvaru dd/mm/yyyy. Pole je dlouhé 10bytů a datum je tedy uloženo včetně znaků '/'.

unsigned char RTC_getWeekDay(void) Funkce přečte den v týdnu (registr 06) a vrátí číslo 0 - 6, 0x00 je sobota a 0x07 je neděle.

unsigned char RTC_getControl1(void) Funkce přečte první konfigurační byte. V tomto bytu se nachází bit5, který zastavuje funkce obvodu.

unsigned char RTC_getControl2(void) Funkce přečte druhý konfigurační byte. Tento byte je důležitý pro povolení a nastavení přerušení od alarmu nebo časovače. Byte také informuje o tom, čím bylo přerušení vyvoláno (časovač, nebo alarm).

void RTC_clkoeEnable(void) Funkce povolí výstup frekvence z RTC. Nastaví na bráně PD4 logickou 1, a tím povolí generování signálu RTC_CLK. Frekvence signálu je nastavena funkcí RTC_setClkOut() .

void RTC_clkoeDisable(void) Funkce zakáže výstup frekvence z RTC. Nastaví na bráně PD4 logickou 0 .

unsigned char RTC_getAllarm_Minute(void) Přečte registr s minutovým alarmem. Nejvyšší bit informuje o povolení alarmu.

unsigned char RTC_getAllarm_Hour(void) Přečte registr s hodinovým alarmem. Nejvyšší bit informuje o povolení alarmu.

unsigned char RTC_getAllarm_Day(void) Přečte registr s denním alarmem. Nejvyšší bit informuje o povolení alarmu.

unsigned char RTC_getAllarm_Weekday(void) Přečte registr s alarmem na den v týdnu. Nejvyšší bit informuje o povolení alarmu.

unsigned char RTC_getClkOut(void) Přečte registr s nastavením výstupní frekvence. Nejvyšší bit v registru informuje o povolení alarmu.

unsigned char* RTC_getTimer(void) Přečte konfiguraci čítače a vrátí ukazatel na pole, ve kterém jsou tyto hodnoty uloženy. Délka pole jsou dva byte. První byte udává nastavení a povolení časovače, druhý byte nastavuje dělící konstantu v rozsahu 0 - 255.

unsigned char RTC_updateTime(unsigned char *new_time) Funkce změní čas v registrech obvodu RTC. Čas je funkci předán ukazatelem na pole new_time. Pole by mělo mít délku 8byte a data v něm by měla být uložena v ascii ve formátu hh:mm:ss.

(unsigned char RTC_updateDate(unsigned char *new_date)) Funkce upraví datum v RTC. Datum je předáno ukazatelem na pole new_date. Formát data pro uložení je DD/MM/YYYY, délka pole je 10byte.

unsigned char RTC_updateWeekDay(unsigned char day) Nastaví den v týdnu 0 (sobota) až 7 (neděle).

unsigned char RTC_setControl1(unsigned char control1) Nastavuje první řídící registr viz. 5.6.

unsigned char RTC_setControl2(unsigned char control2) Nastavuje druhý řídící registr viz. 5.6.

unsigned char RTC_setAllarm_Minute(unsigned char minute) Nastavuje minutový alarm. Nejvyšším bitem se alarm povoluje.

unsigned char RTC_setAllarm_Hour(unsigned char hour) Nastavuje hodinový alarm. Nejvyšší bit povoluje alarm.

unsigned char RTC_setAllarm_Day(unsigned char day) Nastavuje denní alarm. Nejvyšší bit povoluje alarm.

unsigned char RTC_setAllarm_Weekday(unsigned char weekday) Nastavuje týdenní alarm, nejvyšším bitem se provede povolení.

unsigned char RTC_setCLKout(unsigned char clkout) Nastavuje výstup frekvence 5.6.

unsigned char RTC_setTimer(unsigned char* timer) Funkce provede nastavení registrů časovače.

5.2.9 SPI

Serial Pheripheral Interface je sériové rozhraní se signály MISO,MOSI,SCK a SS, které má jednoho mastera. Master řídí komunikaci na sběrnici - generuje signál SCLK a pomocí signálů SS nebo CS vybírá, se kterým zařízením na sběrnici bude komunikovat.

5.2.9.1 SPI_routines.h

Soubor obsahuje definice SS signálů pro SD kartu a Ethernet a makra, která umožňují změnit rychlost komunikace.

void SPI_init(void) Provádí inicializaci SPI rozhraní, nastavuje porty se signály SS jako výstup.

unsigned char SPI_transmit(unsigned char data) Odesílá data zadaná jako parametr po SPI, funkce čeká dokud není odeslání dokončeno. Dokončení je signalizováno bitem SPIF v registru SPSR.

unsigned char SPI_receive(void) Funkce přijímá data z SPI rozhraní, aktivně čeká dokud není nastaven bit SPIF v registru SPSR.

5.2.10 SD_card

Knihovna s funkcemi pro SD kartu a FAT32 byla převzata z projektu [23]. V souboru SD_card.h byla upravena definice portů pro SD kartu. Komunikace s SD kartou funguje pomocí této knihovny bez problémů. Pro komunikaci s SD kartou by se měly používat výhradně FAT32 funkce. Pokud se nevhodně použije přístup k sektorům SD karty, hrozí poškození filesystemu a SD kartu je potřeba znovu naformátovat na FAT32.

5.2.10.1 SD_routines

Posílá příkazy SD kartě a tvoří rozhraní pro FAT32. Obsahuje buffer o velikosti 512byte tedy jednoho sektoru. Pro komunikaci s SD kartou se používají funkce:

- unsigned char SD_init(void),
- unsigned char SD_sendCommand(unsigned char cmd, unsigned long arg),
- unsigned char SD_readSingleBlock(unsigned long startBlock),

- unsigned char SD_writeSingleBlock(unsigned long startBlock),
- unsigned char SD_readMultipleBlock (unsigned long startBlock, unsigned long totalBlocks),
- unsigned char SD_writeMultipleBlock(unsigned long startBlock, unsigned long totalBlocks),
- unsigned char SD_erase (unsigned long startBlock, unsigned long totalBlocks).

5.2.10.2 FAT32

Ve FAT32 jsou definovány struktury pro reprezentaci file systému, do kterých jsou přečteny informace v době inicializace. Pro operace s file systémem jsou definovány v knihovně funkce, které přečtou a upraví cluster. Tyto funkce při nevhodném použití mohou poškodit file systém. Z hlediska použití knihovny jsou nejzajímavější funkce:

- void deleteFile (unsigned char *fileName),
- void memoryStatistics (void),
- void displayMemory (unsigned long memory),
- void writeFile (unsigned char *fileName),
- unsigned char readFile (unsigned char flag, unsigned char *fileName).

Tyto funkce stačí pro přístup a vytváření souborů na SD kartě.

5.3 Komunikace s deskou

Komunikace s deskou probíhá po USB respektive po sériové lince. Pro komunikaci se dají použít dva přístupy, buď je možné použít virtuální COM port VCP nebo knihovnu FT2xx.dll. Výhodou použití VCP je nezávislost na použitém ovladači. V případě použití FT2xx je nutné mít správnou verzi ovladače a použité knihovny, jinak se program vůbec nepodaří spustit. V examples na CD jsou umístěny zdrojové kódy pro jednotlivé jazyky.

5.3.1 JAVA

V JAVE nejsou definovány funkce pro komunikaci prostřednictvím COM portu pod OS Windows. Jedinou možností jak otevřít komunikaci přes VCP ve Windows, je použít open-source knihonu RXTX. Pro použití RXTX knihovny je potřeba si stáhnout její distribuci ze stránek projektu [22]. Výhodou této knihovny je, že není závislá na verzi použitých ovladačů. Pokud máme staženou knihovnu můžeme do projektu přidat rxtxSerial.dll a RXTXcomm.jar, po přidání se v kódu provede import potřebných knihoven viz. příklad:

```
import gnu.io.CommPort;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;

public class BasicComm{

void test_RXTX()throws NoSuchPortException, PortInUseException,
UnsupportedCommOperationException, IOException{

//Vytvori port idettifikator
CommPortIdentifier portIdentifier = CommPortIdentifier.getPortIdentifier("COM4");

//vytvori commport
CommPort commPort = portIdentifier.open(this.getClass().getName(),2000);

//pretypovani na seriovy port
SerialPort serialPort = (SerialPort) commPort;

//nastvaeni parametru komunikace
serialPort.setSerialPortParams(230400,SerialPort.DATABITS_8,
SerialPort.STOPBITS_1,SerialPort.PARITY_NONE);

//vytvoreni vstpniho a vystupniho proudu
InputStream in = serialPort.getInputStream();
OutputStream out = serialPort.getOutputStream();

//zapis a precteni znaku z portu
out.write('b');
int znak = in.read();

System.out.println((char) znak);
}
}
```

Spuštění se provede tak, že ve funkci main() vytvoříme instanci BasicComm bcomm = new BasicComm(); a pak se zavolá bcomm.test_RXTX().

Druhou možností by bylo použít FT2xx.dll.

5.3.2 C++

V prostředí C++ se přístup k sériové lince provádí pomocí virtuálního souboru, který se otevře pro čtení a zápis. K souboru jsou možné dva přístupy: buď překrývaný nebo nepřekrývaný. Typ přístupu se specifikuje při volání funkce CreateFile(). Nepřekrývaný způsob přístupu má nevýhodu u vícevláknových aplikací, protože k portu může přistupovat

pouze jeden proces, ale je jednodušší na použití. Zatímco u překrývaného je možné, aby více vláken provádělo I/O operace. Na příkladu je ukázáno otevření komunikace nepřekrývaně.

```
HANDLE   hCom; //handle souboru pro com port
CString  port;
DCB   comPort;  //definice struktury pro port
BOOL    comReady;
BOOL status;
DWORD   Readed;
DWORD   Writen;
port = "com4";

//Vytvoření souboru parametry port, typ operací,typ sdílení,
zabezpečení,typ vytvoření,atributy souboru, template souboru)
hCom = CreateFile(port,GENERIC_READ | GENERIC_WRITE,0,0,OPEN_EXISTING,0,0);

//Nastaveni parametru komunikace
SetupComm(hCom, 256, 256);
comReady = GetCommState(hCom, &comPort);

comPort.BaudRate = 230400;
comPort.ByteSize = 8;
comPort.Parity = NOPARITY;
comPort.StopBits = ONESTOPBIT;
comPort.fAbortOnError = TRUE;

comReady = SetCommState(hCom, &comPort);

/Zápis nebo čtení portu se provádí příkazy

//parametry pro cteni / zapis
//prvni je soubor kam se pise, pak data char[], delka dat,
 počet prectenych/zapsanych bytu, overlaped
status = WriteFile(hCom, "AHOJ",4,&Written,0);

status = ReadFile(hCom, &sBuffer, 4, &Readed,0);
```

Více informací o jednotlivých funkcích a překrývaném režimu lze získat na stránkách: msdn.microsoft.com [17].

5.3.3 C#

C# je na komunikaci po sériovém portu nejjednodušší, pokud se spokojíme s využitím pouze VCP. V tomto případě do projektu přidá knihovna System.IO.Ports. Knihovna ob-

sahuje funkce pro otevření komunikace po sériovém portu. Příklad otevření portu a zápisu dat do něj, případně čtení je ukázáno na příkladu, který je přiložen v adresáři example.

```
         (*UART COMM *)
using System.IO.Ports;
namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
//vytvoreni struktury SerialPort s nastavenim parametru komunikace
SerialPort port = new SerialPort("COM8",230400,Prity.None,8,StopBits.One);

//otevreni portu
port.Open();

//nyni zapis do portu data, offset, delka
port.Write("TRANSMIT DATA",0,13);

//definice bufferu
char [] rx_data = new char[64];

//cteni z portu buffer, offset, delka
port.Read(rx_data,0,13)

}
}
}
```

5.4 Testování a oživení

Testování funkčnosti jednotlivých částí se provádělo po krocích. Jako první krok byly osazeny zdroje procesorové desky a zkontrolováno jejich výstupní napětí. Ve druhém kroku byl osazen procesor, který byl otestován komunikací s programátorem. Po osazení krystalu byly nastaveny Fusebity, které určují zdroj hodinového signálu dělička atd. Ve třetím kroku byla osazena část s FTDI, která zajišťuje komunikaci procesoru po USB. Po odzkoušení této části byly osazeny EEPROM, RTC a převodník MAX232. Osazení převodníku pro SD kartu a slotu bylo provedeno jako poslední. V poslední fázi osazování celé procesorové desky byly osazeny moduly redukce, vstupů a výstupů.

5.4.1 Programování

Pro programování je nejjednodušší a nejlevnější použít sériový programátor ponyprog, který se nechá postavit s použitím tří diod. Návod na jeho stavbu je možné nalézt na

`http://www.lancos.com/prog.html`. Programátor ponyprog používá pro programování sériovou linku. Druhou a pohodlnější možností je použít programátor s STK500 rozhraním, jehož schéma lze nalézt třeba na stránce [26]. Tento programátor spolupracuje s AVR studiem. Na CD jsou v adresáři PROGRAMATOR podklady pro výrobu druhé verze tohoto programátoru [27]. Programátor má výstup frekvence 1MHz pro oživení zablokovaného procesoru. Po dobu programování je procesor držen programátorem ve stavu reset a pro naprogramování je nutné, aby procesoru běžel zdroj hodinového signálu. Pokud oscilátor nepoběží, nejde procesor naprogramovat. Třetí možností je použít JTAG ICE mkII [6], který dovoluje v procesoru provádět On-Chip debuging[2]. Bohužel tento programátor je nutné koupit, jeho cena se u nás pohybuje okolo 10000,-Kč.

5.4.2 Fusebity

Pomocí fusebitů se nechá nastavit chování procesoru na výpadek proudu, povolení onchip debug, nastavení oscilátoru atd. Přesný popis jednotlivých fuse bitů je v dokumentaci procesoru. Pro správnou funkci demo programů musí být deska osazena krystalem o frekvenci 14.7456MHz a fusebity jsou nastaveny následovně EXTENDED = 0xFF, HIGH=0x99, LOW=0xFF. Ve fusebitech je nastaven externí oscilátor a vypnuta dělička frekvence osmi. Problém může nastat při nevhodném nastavení fusebitů, procesor se může zastavit a nelze s ním komunikovat. V tomto případě pravděpodobně neběží oscilátor. Oživení zamčeného procesoru je možné pomocí externího zdroje hodinového signálu o frekvenci 1MHz. Zdroj se připojí na pin XTAL1 procesoru, tím by se měl procesor rozběhnout a fuse bity by měly jít pomocí programátoru přeprogramovat.

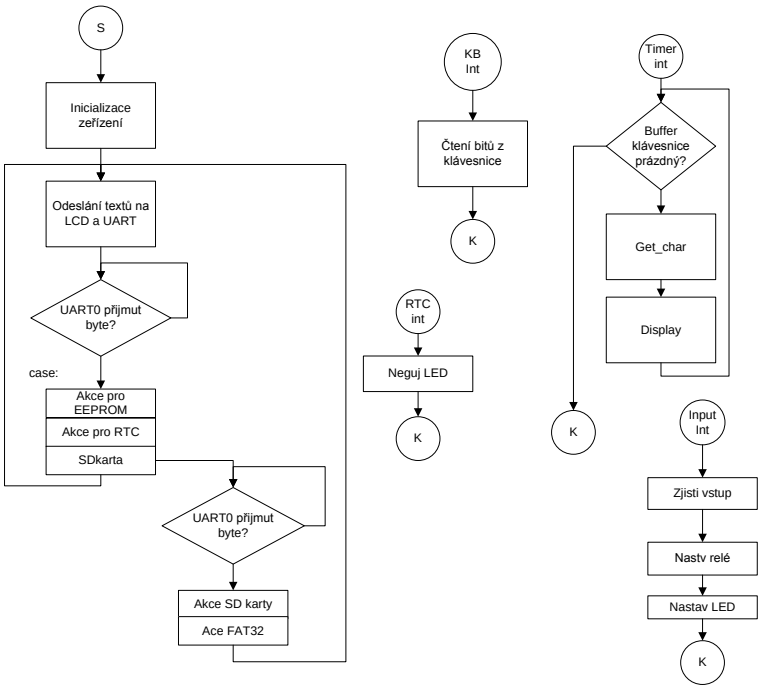
5.4.3 Testovací aplikace

V adresáři FIRMWARE na CD lze nalézt miniaplikace, které mohou pomoct při osazování desky. Je zde aplikace LED_blik, která bliká s led diodou v intervalech daných rychlostí hodinového oscilátoru, při 14,7456Mhz je to cca 0.5s. Dále je zde program ECHO, který posílá po všech sériových linkách text "AHOJ".

5.4.4 DEMO

Přeložené soubory DEMO aplikace jsou na CD v adresáři FIRMWARE. Po naprogramování flash paměti procesoru deska komunikuje na USB portu přes terminál. Rychlost komunikace je 230400b/s 8datových bitů žádná parita. Stisknuté klávesy na klávesnici se zobrazují na připojeném LCD. LCD by mělo být připojeno na UART2 procesorové desky. Přes sériovou linku je možné pracovat s SD kartou, číst a zapisovat do EEPROM, číst a nastavovat čas v RTC. Vstupy jsou naprogramovány tak, aby při změně vstupu negovaly stav led diod, které jsou u nich umístěny. Relé reagují na změnu vstupů tak, že se zapíná jedno nebo druhé. Diagram DEMO programu a podprogramů je na následujícím obrázku 5.7.

[2] ladění programu v procesoru



Obrázek 5.7: Diagram Demo programu

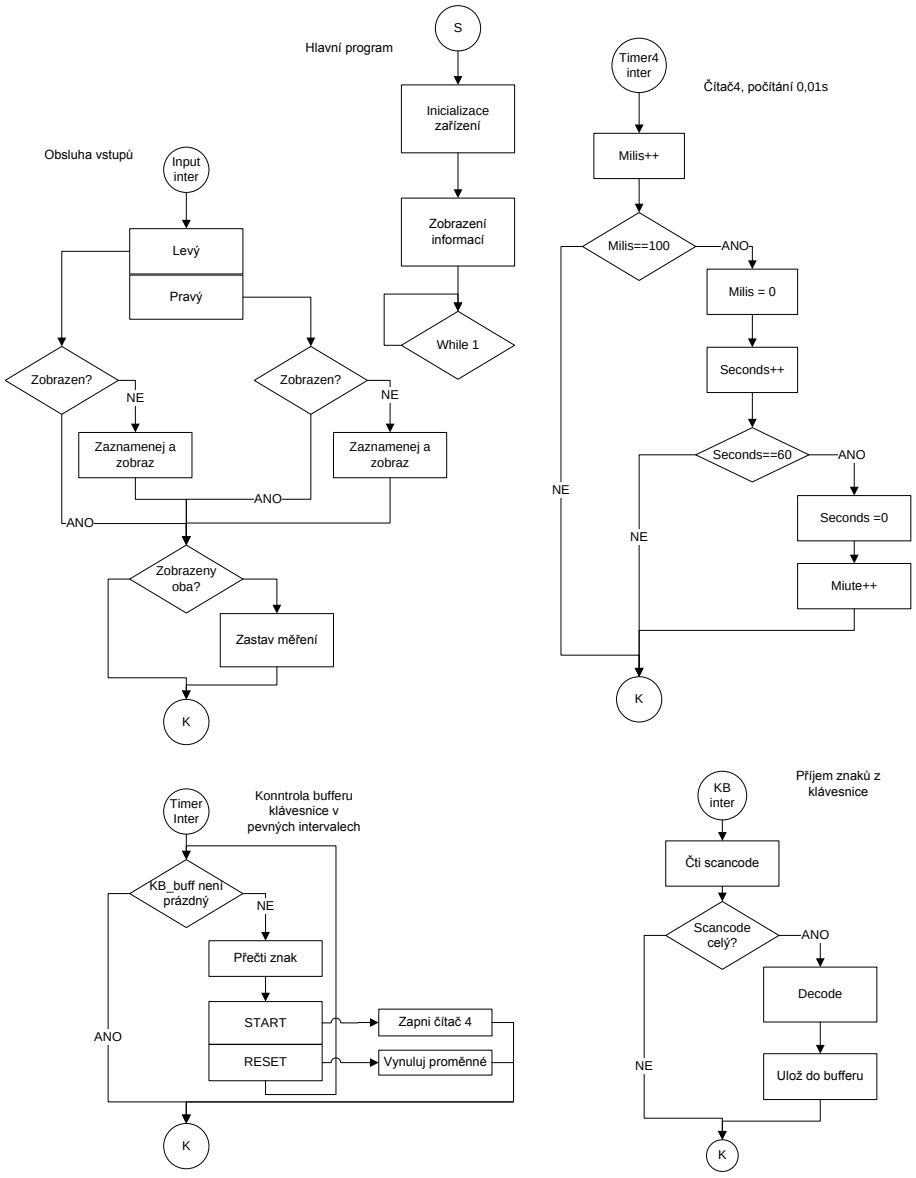
Základ programu tvoří hlavní smyčka, kde se přijímají data z USB (sériové linky) a provádí skoky do jednotlivých podprogramů, podle toho jestli se bude pracovat s SD kartou, RTC nebo EEPROM. Přerušení KB_int čte bity z klávesnice a po přijmutí scankódu jej dekóduje na znak, který uloží do bufferu. Tento buffer je pravidelně čten přerušením Timer_int, pokud jsou v bufferu přijaté nějaké znaky, jsou přečteny a zobrazeny na LCD. Přerušení Input_int se stará o obsluhu vstupů, jako signalizace stavu jsou přepínány LED diody na modulu vstupů a relé na výstupním modulu. Poslední přerušení rozsvěcí a zhasíná led na procesorové desce.

5.4.5 Časomíra

V této sekci je popsán jednoduchý firmware, který také může posloužit pro odzkoušení modulárního systému. Časomíra využívá dva optické vstupy a dva výstupy. Po stisku start (ENTER) začne měřit čas. Při sepnutí jednoho ze dvou vstupů je zaznamenán čas, jeho sepnutí a měření pokračuje dál, dokud nedojde k sepnutí druhého vstupu. Zaznamenání času sepnutí se provede pouze poprvé. Další spínání již nemá na zaznamenaný čas vliv. Diagram programu je na následujícím obrázku 5.8.

V hlavním programu se provádí pouze nastavení čítačů a přerušení, následně se zobrazí informace o stavu časomíry na LCD. Časomíra čeká na stisk klávesy ENTER, kterým se spustí čítání. Spouštění a zastavování provádí přerušení od timeru0, které se volá v pravidelných intervalech. Při přerušení od timeru0 je zkontrolován buffer klávesnice, pokud je zde klávesa ENTER provede se zapnutí. Klávesa TAB provádí RESET časomíry. Přerušení KB_inter se stará o příjem a ukládání kláves do bufferu klávesnice. Posledním přerušením je přerušení

od vstupů. Toto přerušení provádí zaznamenání času sepnutí a v případě, že je zaznamenán čas levého i pravého vstupu, je časomíra zastavena.



Obrázek 5.8: Diagram programu časomíry

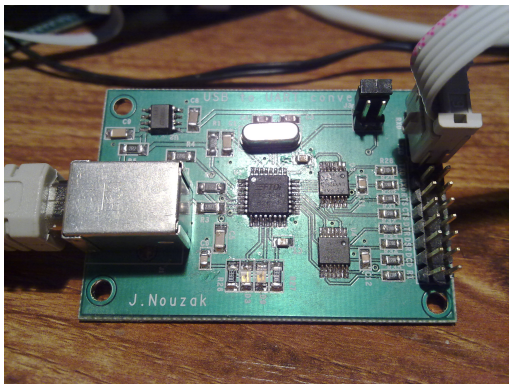
Kapitola 6

Přídavné moduly

V této části jsou popsány moduly, které přímo nesouvisí s modulárním systémem, ovšem vnikly nebo byly použity při realizaci systému.

6.1 Převodník USB<->SERIAL

Jedná se opět o převodník firmy FTDI, který je zapojen stejně jako na procesorové desce. Ovšem v tomto případě je realizován na samostatné desce a je k němu připojen zdroj s napětím 3,3V LE33CD. Zdroj vytváří napětí pro budiče 74LVXC125, které tvoří výstupní rozhraní převodníku. Na desce převodníku je umístěna propojka, která dovoluje změnit napájecí napětí těchto budičů buď na 3.3V nebo na 5V. Převodník pak může komunikovat jak s 5V logikou tak s 3.3V logikou. Celý převodník je napájen z USB. Na převodníku jsou vyvedeny všechny signály sériového rozhraní RX, TX, RTS, CTS, DTR, DSR, DCD, RI.



Obrázek 6.1: Převodník USB <-> UART

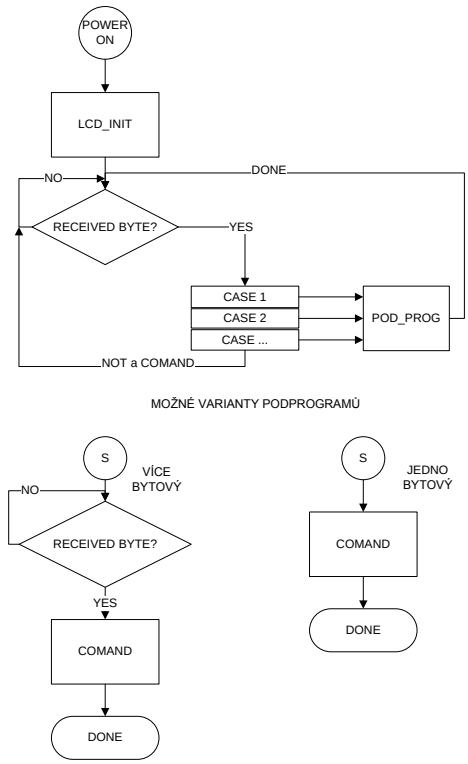
Převodník byl použit pro ladění Firmware v modulu LCD, který se připojuje k procesorové desce. Úspěšně byl odzkoušen při komunikaci s HDD firmy Seagate, kde s jeho pomocí lze odstranit chybu způsobující nefunkčnost disku. Převodník je používán při sériové komunikaci s obvody FPGA, které používají 3.3V interface.

6.2 LCD displej

Jako modul k desce je možné připojit sériový LCD display, který byl vytvořen Ladislavem Boreckým. Modul displeje obsahuje display ATM12864D [7] s řadičem KS0108, tento display je připojen k desce s procesorem ATmega8. Deska s procesorem tvoří rozhraní pro sériovou komunikaci a vytváří znakovou sadu displeje. Modul řadiče je postaven na DPS, kterou navrhl Ladislav Borecký. Modul byl v rámci této práce osazen a oživen. Do modulu byl napsán firmware, jenž zajišťuje zobrazení ASCII znaků a obrázků. Modul využívá pro komunikaci sériovou linku. Při programování byly odhaleny malé nedostatky, jejich řešení je popsáno v sekci Zlepšení.

6.2.1 Firmware

Firmware LCD displeje je v adresáři LCD_Firmware na CD, v adresáři se nachází celý projekt AVRstudia. Komunikace s modulem displeje využívá sériové linky, po které se přenáší příkazy. Displej nevyužívá přerušení, protože vše co přijme okamžitě zobrazuje. Rychlost komunikace je nastavena tak, aby displej stihl přijatý znak zobrazit. Vývojový diagram firmwaru 6.2.



Obrázek 6.2: Vývojový diagram firmwaru LCD

Řadič displeje čeká na přijetí bytu, po přijetí se provede skok do podprogramu podle toho, jaký byte byl přijat a jaký podprogram se má vykonat. V případě jednobytového příkazu se okamžitě provede požadovaná operace s displejem. V případě dvou bytové operace se čeká

na přijmutí druhého bytu. Druhý byte jsou většinou data, která jsou zapisována na LCD, po jejich zapsání se čeká opět na příjem příkazu v hlavní smyčce programu.

6.2.2 Seriová Komunikace

Pro komunikaci s displejem se používá sériová linka s TTL úrovněmi. Komunikuje se rychlostí 19200b/s bez parity a jedním stopbitem. Většina instrukcí je dvoubajtových. První byte je instrukce a druhý byte jsou data. Přehled funkcí je v následující tabulce 6.1.

Kód	**Délka**	**Funkce**
0xFF	1	Provede smazání celého LCD a přesune kurzor do levého horního rohu
0xFE	2	Provede smazání zadaného řádku a přesune kurzor na začátek řádku
0xFD	2	Zapíše přijatý byte přímo na obrazovku (kreslení obrázků)
0xFC	2	Posune horizontálně kurzor na zadanou pozici 0 - 127
0xFB	2	Posune kurzor na zadaný řádek
0xFA	2	Zapíše ASCII znak z paměti programu na LCD, kurzor na konci znaku
0xF9	2	Zapíše ASCII znak z paměti programu na LCD, kurzor na začátku znaku
0xF8	2	Zapíše ASCII znak z paměti programu na LCD, kurzor na místo o znak před zapsaným znakem
0xF7	1	Smaže znak za kurzorem (Backspace)
0xF6	2	Zapíše ASCII znak z EEPROM, kurzor na konci znaku
0xF5	2	Zapíše ASCII znak z EEPROM, kurzor na začátku znaku
0xF4	2	Zapíše ASCII znak z EEPROM, kurzor na místo o znak před zapsaným znakem
0xF3	2	Zapíše negovaný znak z paměti programu na LCD kurzor na místo o znak před zapsaným znakem
0xF2	1	Přesune kurzor na začátek aktuálního řádku
0xF1	1	Přesune kurzor do levého horního rohu na pozici 0,0
0xF0	4 + data	Zapíše přijatá data do EEPROM
0xEf	1	Zobrazí LOGO ATMEL na celý display, LOGO je uloženo v paměti programu
0xEE	1	Zobrazí LOGO JN na celý display, LOGO je uloženo v paměti programu

Tabulka 6.1: Příkazy LCD řadiče pozn.

Poznámka: pokud délka instrukce není uvedena přesně, může se její délka měnit v závislosti na datech.

Struktura instrukce pro zápis do EEPROM je následující. Jako první se posílá příkaz 0xF0, pak následuje délka dat 1Byte a na závěr dva bajty adresy High a Low. Po této sekvenci následují data se zadanou délkou.

6.2.3 Zlepšení

Na displeji není možno kreslit obrazce. Pro kreslení čar, případně obdélníků koleček atp. by bylo zapotřebí číst obsah LCD, aby bylo možné měnit jednotlivé pixely. Modul, který byl použit pro LCD, nemá zapojen pin Read/Write, kterým se přepíná mez čtením a zapisováním do/z LCD.

Pro uložení obsahu LCD by se nechala použít paměť procesoru ATmega8, který je na modulu osazen. V tomto případě by se do paměti muselo uložit pole o velikosti 1kB (64*128bitů). Paměť RAM procesoru je velká přesně 1kB a není možné ji celou použít pro uložení obsahu LCD.

Druhou možností by bylo použít EEPROM na desce jako paměť, ve které se udržuje obraz LCD. Paměti EEPROM je na desce k dispozici dostatek. Tato možnost by ovšem značně zpomalila komunikaci s displejem, neboť I^2C sběrnice běží maximálně na 400kHz a pro jednu operaci je potřeba provést minimálně 6příkazů po 8bitech, což dělá na I^2C zpoždění 0.24mS, rychlost komunikace se pak pohybuje někde okolo 4kHz. Rychlost komuniace by bylo možné zvýšit i změnou typu EEPROM. Firma Microchip dodává stejný typ pamětí, které místo označení 24LCxx nesou označení 24FCxx. Tyto paměti jsou dle výrobce schopné komunikovat po I^2C rychlostí až 1MHz.

Největší nevýhodou, která úplně vylučuje možnost použití EEPROM, je jejich životnost. Výrobce uvádí, že paměť snese min 1000000 cyklů. V případě, že se použije jako paměť s obsahem LCD by se musel při každém zápisu na LCD upravit obsah paměti. To znamená provést zápisový cyklus do paměti, těchto cyklů s ohledem na použití displeje mohou být řádově stovky milionů. Nápor zápisových cyklů nemůže paměť EEPROM vydržet a proto jsou EEPROM paměti nevhodné.

Jako nejrozumnější řešení bych viděl upravit zapojení desky řadiče a použít procesor ATmega16, který je v pouzdře se 44piny. Pouzdro má dostatečný počet vývodů na to, aby se zapojily i piny potřebné pro čtení z řadiče displeje. Procesor sám obsahuje 16kByte Flash program memory, 512Bytes EEPROM a 1kByte SRAM. Tento procesor by umožnil pomocí PWM řídit podsvětlení, případně kontrast displeje. Navržené schéma je součástí přílohy G.

Kapitola 7

Závěr

Během návrhu modulárního systému se vyskytly běžné problémy s dostupností některých součástek. Obvod 74LVX125 tvořící rozhraní pro SD kartu nebyl k dostání, ač prodejci deklarovali, že je skladem. Nakonec se ho podařilo sehnat ještě v menším pouzdře, než bylo původně zamýšleno.

Při vývoji procesorové desky byla provedena simulace spínaného zdroje, který měl napájet procesorovou desku. Tento zdroj byl hlavně kvůli své velikosti a špičce, kterou vytvářel při výstupním napětí 3,3V, opuštěn. Zdroj byl nahrazen dvojicí zpětnovazebních stabilizátorů.

Jako další byly simulovány obvody vstupního modulu. U vstupů byla zvažována možnost použít optočlen bez integrovaného zesilovače a komparátoru. Po provedené simulaci bylo toto řešení opuštěno a byl použit dražší optočlen s integrovaným zesilovačem a komparátorem.

U výstupů se simulovala velikost špičky vznikající při rozpínání obvodu s relé. Všechny simulace byly provedeny v prostředí Oracad simulátorem PSpice. Jako modely součástek byly použity modely, které měli výrobci na svých stránkách, případně podobné součástky jiných výrobců. Odpory, kondenzátory, cívky byly vždy ideální.

Desky plošných spojů, které vznikly během této práce, byly navrženy v prostředí ORCAD PCBeditor. Pro všechny součástky byly definovány padstacky a footprinty. Desky plošných spojů mají nejmenší rastr 0,2mm, jen u procesorové desky druhé verze je rastr menší, protože pouzdro QFN se do původního rastru nevejde.

Po sestavení a oživení mile překvapila spotřeba, která se u procesorové desky pohybuje kolem 80mA, což je méně než polovina předpokládané spotřeby.

Problém nastal s programováním přes USB. Programování přes FTDI v BitBang módu je velice pomalé, řádově desítky minut. Problém s pomalým programováním vznikl nešťastným výběrem převodníku FT232B. Ten byl vybrán, protože podle datasheetu uměl BitBang mód a byl dostupný v pouzdře TQFP32, které je jednodušší na ruční připájení. Zlepšení přinese nahrazení FT232B obvodem FT232R, který nepoužívá krystal 6MHz, ale má vestavěný oscilátor 40MHz, a proto je při použití BitBang módu mnohem rychlejší. Upravené schéma a DPS pro obvod FT232R lze nalézt v adresáři CPU_Board_V2, v adresáři se nachází také podklady pro výrobu DPS. Nevýhodou obvodu je, že je k dispozici pouze v bezvývodovém provedení QFN, které je náročnější na připájení.

Jako první byl napsán firmware pro LCD modul, kde byl navržen sériový protokol pro ovládání displeje a zobrazení znaků. Při psaní a ladění sériového protokolu byl použit modul

převodníku USB na UART, který vznikl v průběhu této práce.

Během programování funkce, která by na LCD kreslila libovolnou čáru případně obrazec, byl odhalen nedostatek desky modulu displeje. Nepřipojený pin R/W u LCD způsobuje, že nelze zjistit, co na displeji svítí. Jedinou nejschůdnější možností je přepracovat modul a použít jiný typ procesoru, k němuž byl proveden návrh zapojení.

Při vývoji demonstrační aplikace bylo postupováno od sériové linky, která posloužila jako ladící rozhraní. Programátor komunikující po SPI neumožňuje krokovat kód v procesoru. Po zprovoznění sériové komunikace byl oživován modul LCD, na kterém bylo možné po napsání knihovny zobrazovat další informace. Komunikace s EEPROM a RTC po I^2C byla zprovozněna s pomocí dokumentace procesoru, kde jsou definovány příznaky posílané po sběrnici.

Pro příjem dat z klávesnice bylo napsáno přerušení, které sleduje náběžné a spádové hrany, aby bylo schopné zaznamenat datové bity scankódu. Použití přerušení jen na spádovou hranu by bylo nejjednodušší, ale klávesnice je připojena na pin, který generuje přerušení typu pin change, a není zde možnost volby hrany. Pro dekódování scankódu byla vytvořena funkce, která rozlišuje sekvence vysílané klávesnicí.

Ke komunikaci s SD kartou byla použita knihovna, ve které byly upraveny definice signálů karty. Tato knihovna spolu s FAT32 po otestování pracovala výborně.

Navržený modul ethernetu nebyl v závěru práce k dispozici v HW podobě, proto nebyla ověřena možnost komunikace s ním. Rozhraní SPI a funkce by však měly být schopné odesílat příkazy i pro Ethernet.

Modulární systém a všechny jeho periferie fungují nyní bezproblémově. Největší nevýhodou, na kterou jsem narazil, je pevné napětí redukce. Mnohem lepší by bylo použít spínaný stabilizátor, který umožňuje nastavit napětí pomocí dvou odporů. Další zlepšení by bylo možné udělat na procesorové desce a připojit klávesnici na pin s externím přerušením. Za úvahu by možná stálo přidat napájení do rozšiřujících konektorů a do konektorů sériových linek. To by umožnilo připojovat malé rozšiřující modulky.

Podařilo se vytvořit funkční prototypy modulárního systému. Výsledná cena systému je po rozpočítání nákladů na jeden modulární systém, tedy procesorovou desku s redukcí, jedním vstupním a jedním výstupním modulem 1600,-Kč. K systému je v současné době vyvíjen převodník na RS485 a moduly pro měření teploty, vlkosti a tlaku.

Literatura

[1] L. Hrbek. Univerzální periferní deska. https://dip.felk.cvut.cz/browse/pdfcache/hrbekl2_2008dipl.pdf, stav z 1.11.2009.

[2] D. Inc. Digilent cerebot-plus. http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,396,611&Prod=CEREBOT-PLUS, stav z 1.11.2010.

[3] J. M. P. Ltd. Jed avr256 single board computer using the atmega2560 cpu. http://www.jedmicro.com.au/avr256.htm, stav z 26.12.2008.

[4] Atmel avrstudio 4. http://www.atmel.com/dyn/Products/tools_card.asp?tool_id=2725, stav z 1.11.2009.

[5] Atmel products - product card. http://www.atmel.com/dyn/products/product_card.asp?part_id=3632, stav z 1.11.2010.

[6] Atmel avr jtagice mkii. http://www.atmel.com/dyn/Products/tools_card.asp?tool_id=3353, stav z 1.11.2009.

[7] Lcd s radicem atm12864d. http://www.gme.cz/_dokumentace/dokumenty/513/513-118/dsh.513-118.1.pdf, stav z 1.11.2009.

[8] Datasheet lm2576. http://cache.national.com/ds/LM/LM2576.pdf, stav z 1.11.2009.

[9] Datasheet lm2576. http://cache.national.com/ds/LM/LM2576.pdf, stav z 1.11.2009.

[10] Datasheet enc424j600 microchip. http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en541877, stav z 1.11.2009.

[11] Epson toyocom corporation | products | rtc-8564je/nb. http://www.epsontoyocom.co.jp/english/product/RTC/set03/rtc8564je_nb/index.html, stav z 1.11.2009.

[12] Datasheet 74lvx125 fairchild. http://www.fairchildsemi.com/ds/74%2F74LVX125.pdf, stav z 1.11.2009.

[13] Ieee802.3af power over ethernet. http://www.zcomax.cz/FileDown.aspx?File=211, stav z 1.11.2009.

[14] Lt1616 linear technology. http://www.linear.com/pc/productDetail.jsp?navId=H0,C1,C1003,C1042,C1032,C1064,P1791, stav z 1.11.2009.

[15] Quickview - max220, max222, max223, max225, max230, max231, max232, max232a, max233, max233a, max234, max235, max236 multichannel rs-232 drivers/receivers. http://www.maxim-ic.com/quick_view2.cfm?qv_pk=1798, stav z 1.11.2009.

[16] Vývojové prostředí mikroc. http://www.mikroe.com/en/compilers/mikroc/avr/, stav z 1.11.2009.

[17] Msdn library microsoft. http://msdn.microsoft.com/en-us/library/default.aspx, stav z 1.11.2009.

[18] On semiconductor - mc34063a. http://www.onsemi.com/pub_link/Collateral/MC34063A-D.PDF, stav z 1.11.2009.

[19] Datasheet 7805 onsemiconductor. http://www.onsemi.com/pub_link/Collateral/MC7800-D.PDF, stav z 1.11.2009.

[20] Pk deign - atmel avr, xilinx cpld, fpga development (evm) boards. http://pk-design.net/HtmlCz/Mainboards.html#mcu_atmel, stav z 1.11.2009.

[21] Pragoboard s.r.o výroba dps. http://www.pragoboard.cz/, stav z 1.11.2009.

[22] Rxtx java. http://rxtx.qbang.org/wiki/index.php/Download, stav z 1.11.2009.

[23] Sdcard interface. http://www.dharmanitech.com/2009/01/sd-card-interfacing-with-atmega8-fat32.html, stav z 1.11.2009.

[24] Datasheet sharp pc900. http://document.sharpsma.com/files/pc900v0nszx_ej.pdf, stav z 1.11.2009.

[25] Datasheet le33c. http://www.st.com/stonline/products/literature/ds/2573/le33c.pdf, stav z 1.11.2009.

[26] Avrusb500 – an open source atmel avr programmer. http://www.tuxgraphics.org/electronics/200510/article05101.shtml#05101lfindex8, stav z 1.11.2009.

[27] Avrusb500 – an open source atmel avr programmer. http://tuxgraphic.org/electronics/200705/article07052.shtml, stav z 1.11.2009.

[28] Winavr. http://winavr.sourceforge.net/, stav z 1.11.2009.

[29] V. Záhlava. *Návrh a konstrukce desek plošných spojů*, volume 1. České vysoké učení technické v Praze, nakladatelství ČVUT, Thákurova 1, 16041 Praha 6, 1th edition, 2005.

Kapitola 8

Seznam použitých zkratek

CPU Central Procesor Unit

LCD Liquid Crystal Display

GPS General Position System

USB Universal Serial Bus

I^{2}C Abstract Boolean Networks

EEPROM Electric Erasable Programable Read Only Memory

RTC Real Time Clock

UART Universal Asynchronous Receiver Transmiter

ISP In System Programing

SPI Serial Pheripherial Interface

JTAG Joint Tes Action Group

POE Power Over Ethernet

IEEE Internation Electrical and Electronic Engineers

AD Analog Digital

DPS Deska Plošného Spoje

ULTRA ATA Druhá generace Advanced Technology Attachment

MISO Master Input Slave Output

MOSI Master Output Slave Input

SCK Serial Clock

SS Slave Select

SMT Surface Mount Technology

SDA Synchronous Data

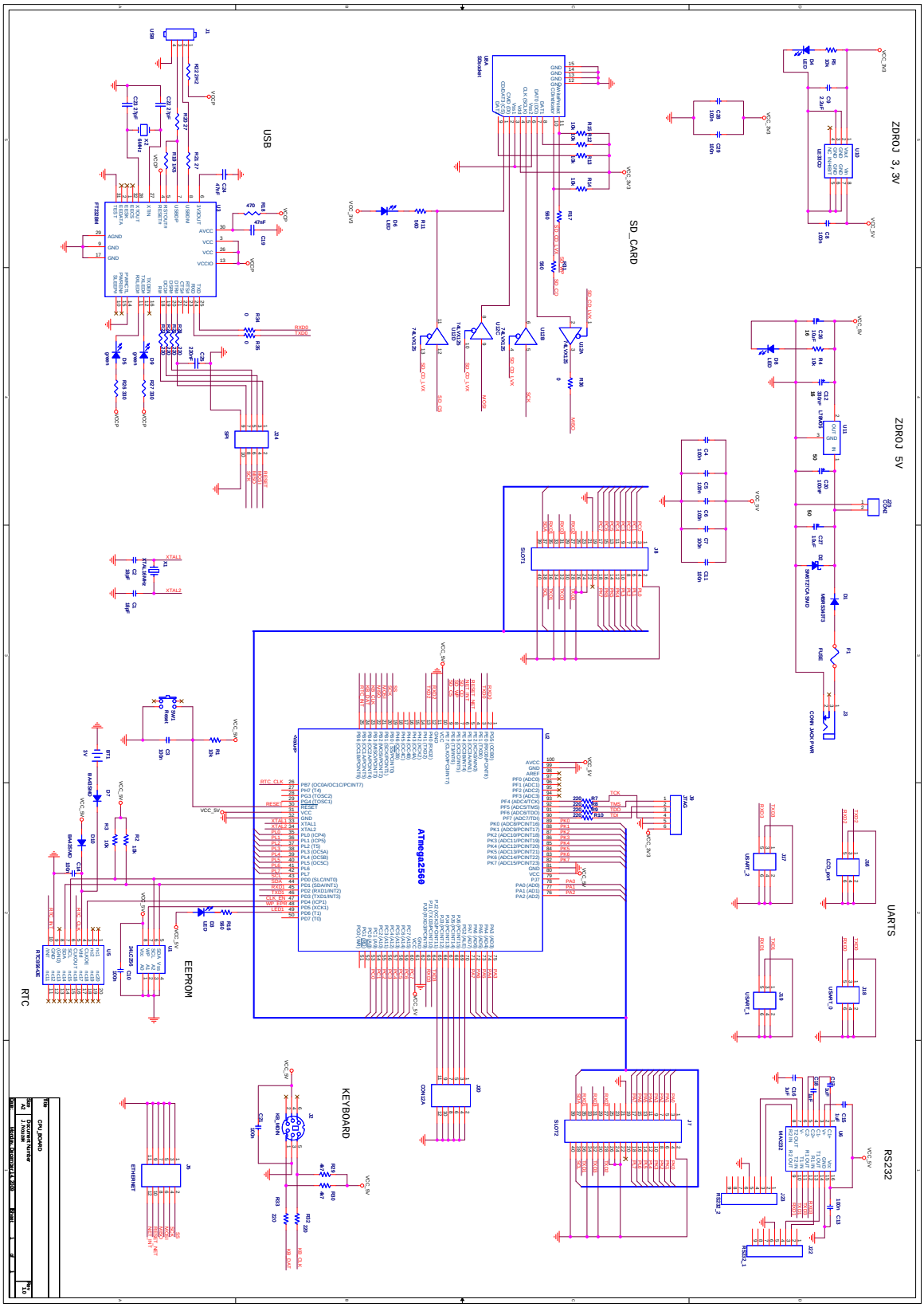
SDC Synchronous Clock

VCP Virtual Comm Port

⋮

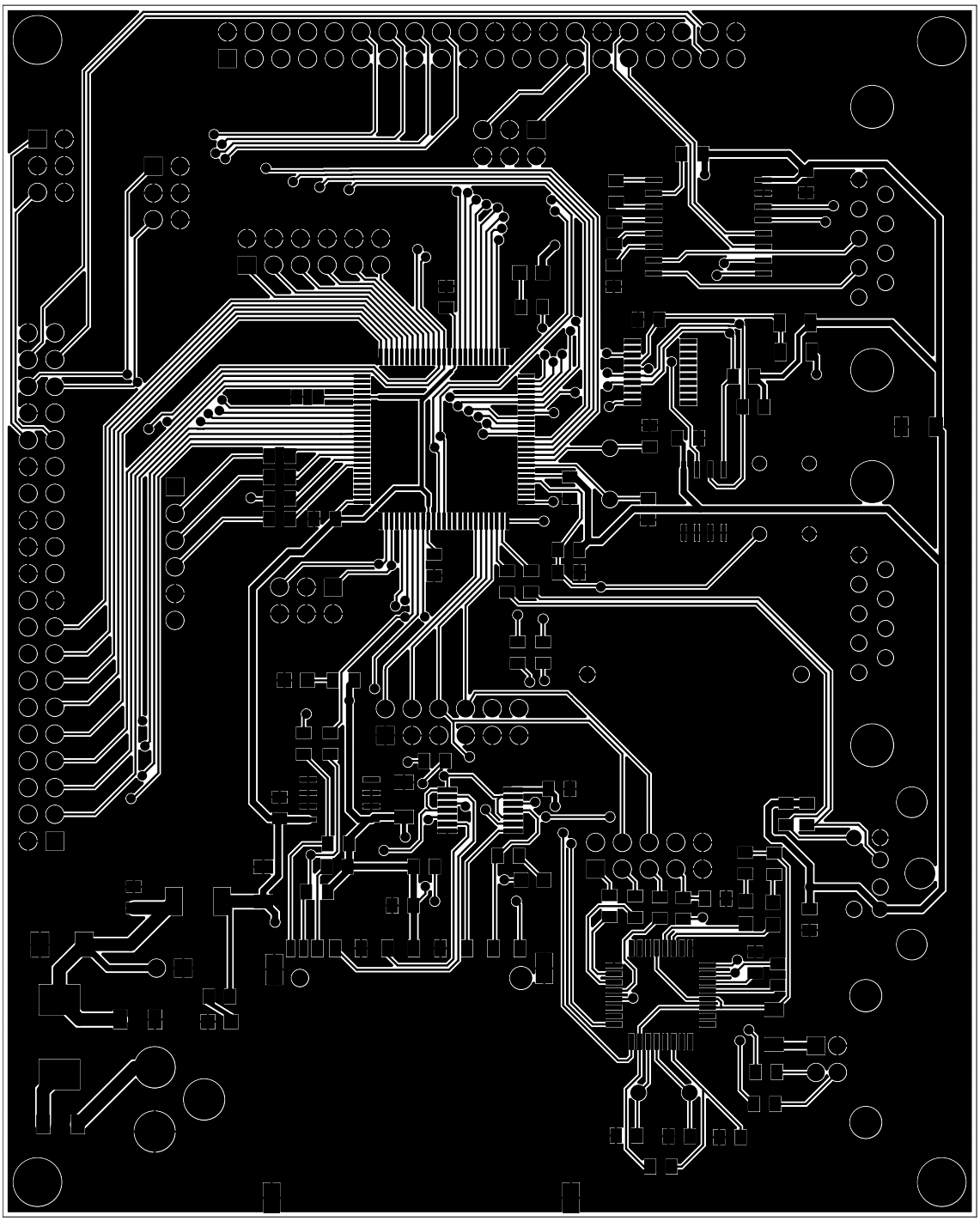
Příloha A

Schéma procesorové desky

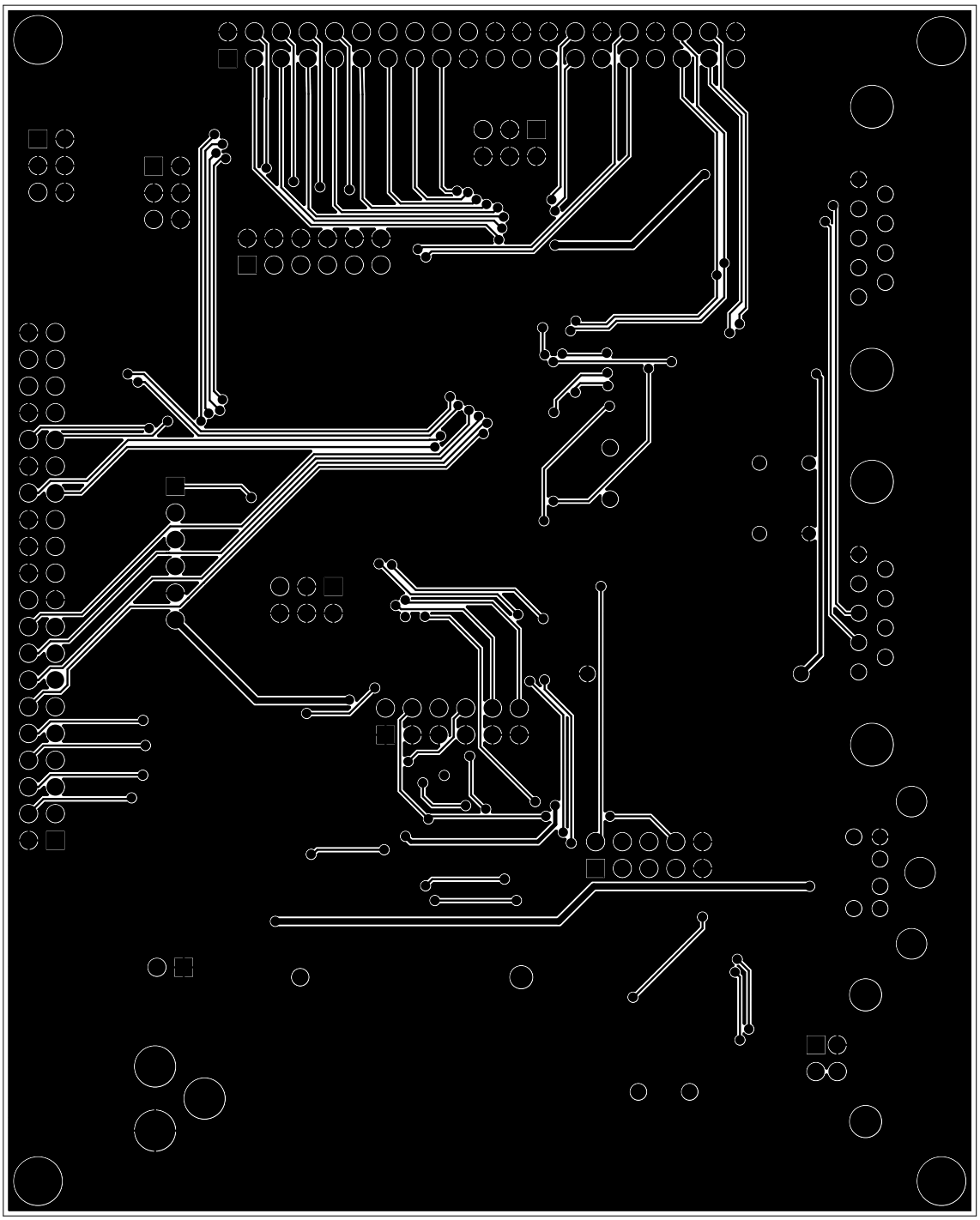

USB
SD_CARD
ZDROJ 3,3V
ZDROJ 5V
UARTS
RS232
RTC
EEPROM
KEYBOARD
ATmega2560
CPU_BOARD

Příloha B

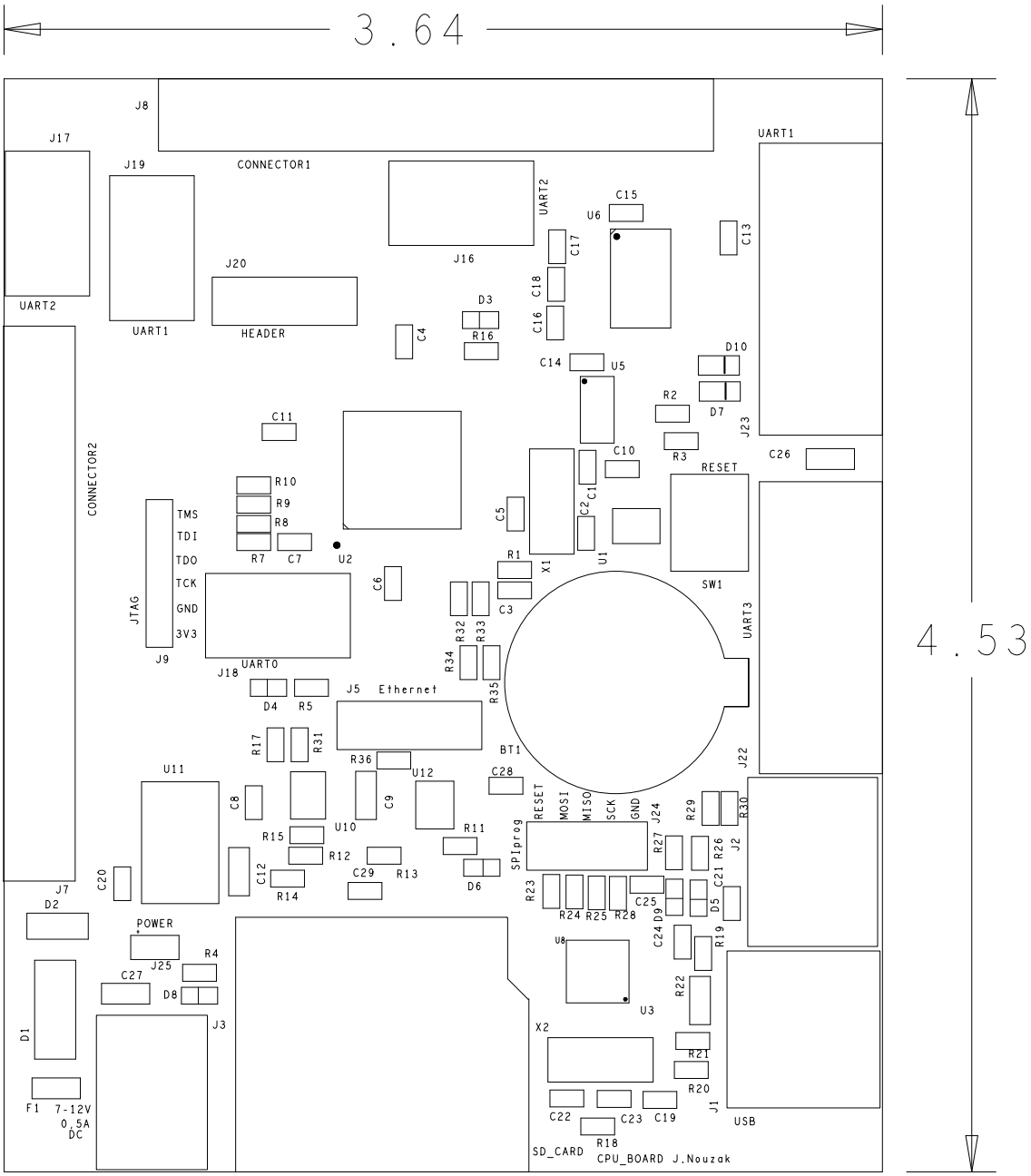
DPS a rozmístění součástek



Obrázek B.1: Strana top procesorové desky



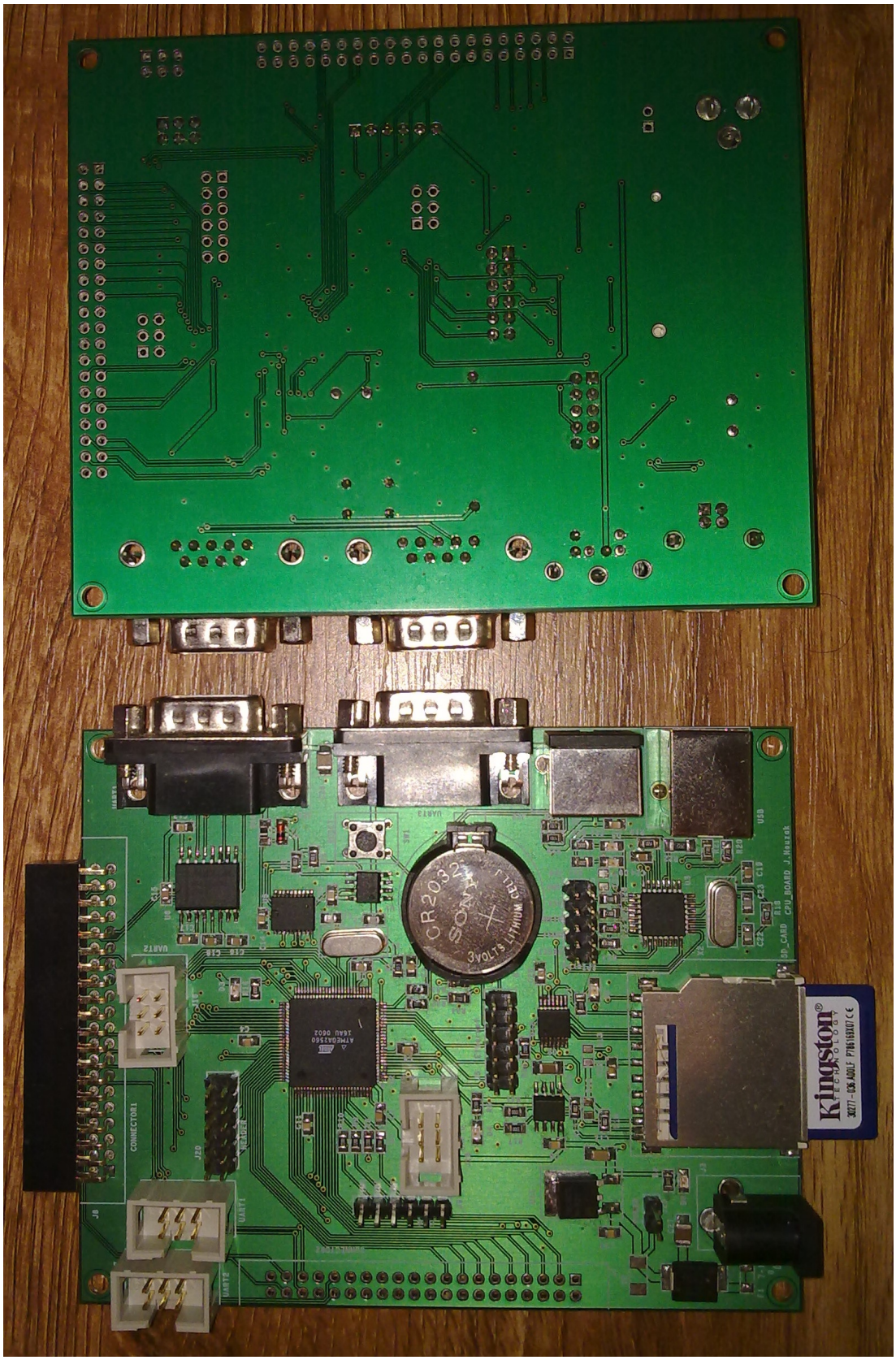
Obrázek B.2: Strana bottom procesorové desky



Obrázek B.3: Rozmístění součástek procesorové desky

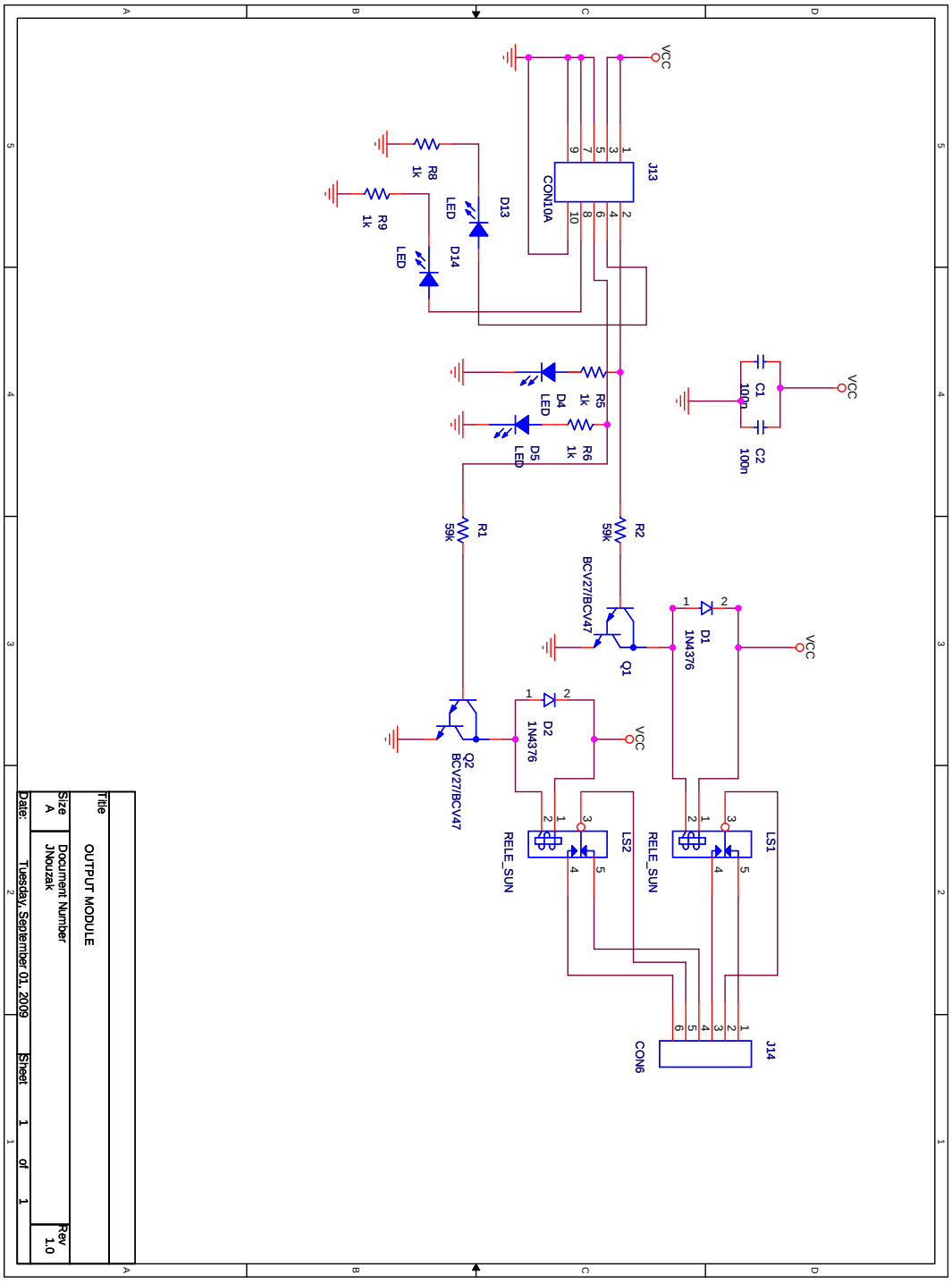
Příloha C

Osazená deska

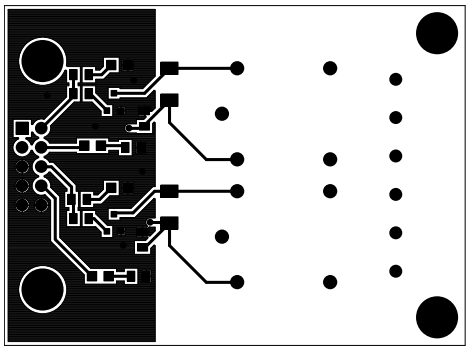

CR2032
3VOLTS LITHIUM
Kingston

Příloha D

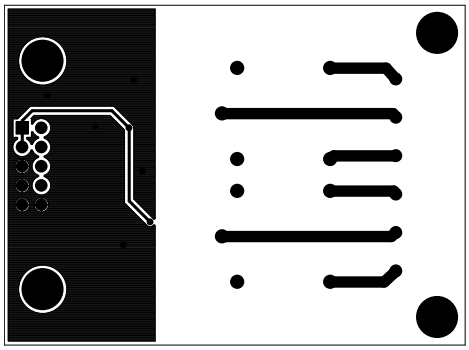
Modul vstupu a výstupu



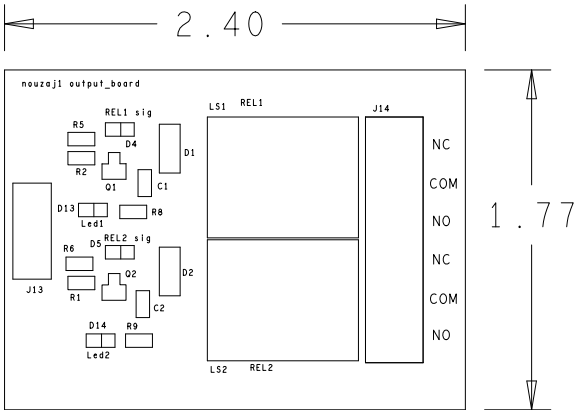
Obrázek D.1: Schéma výstupní desky



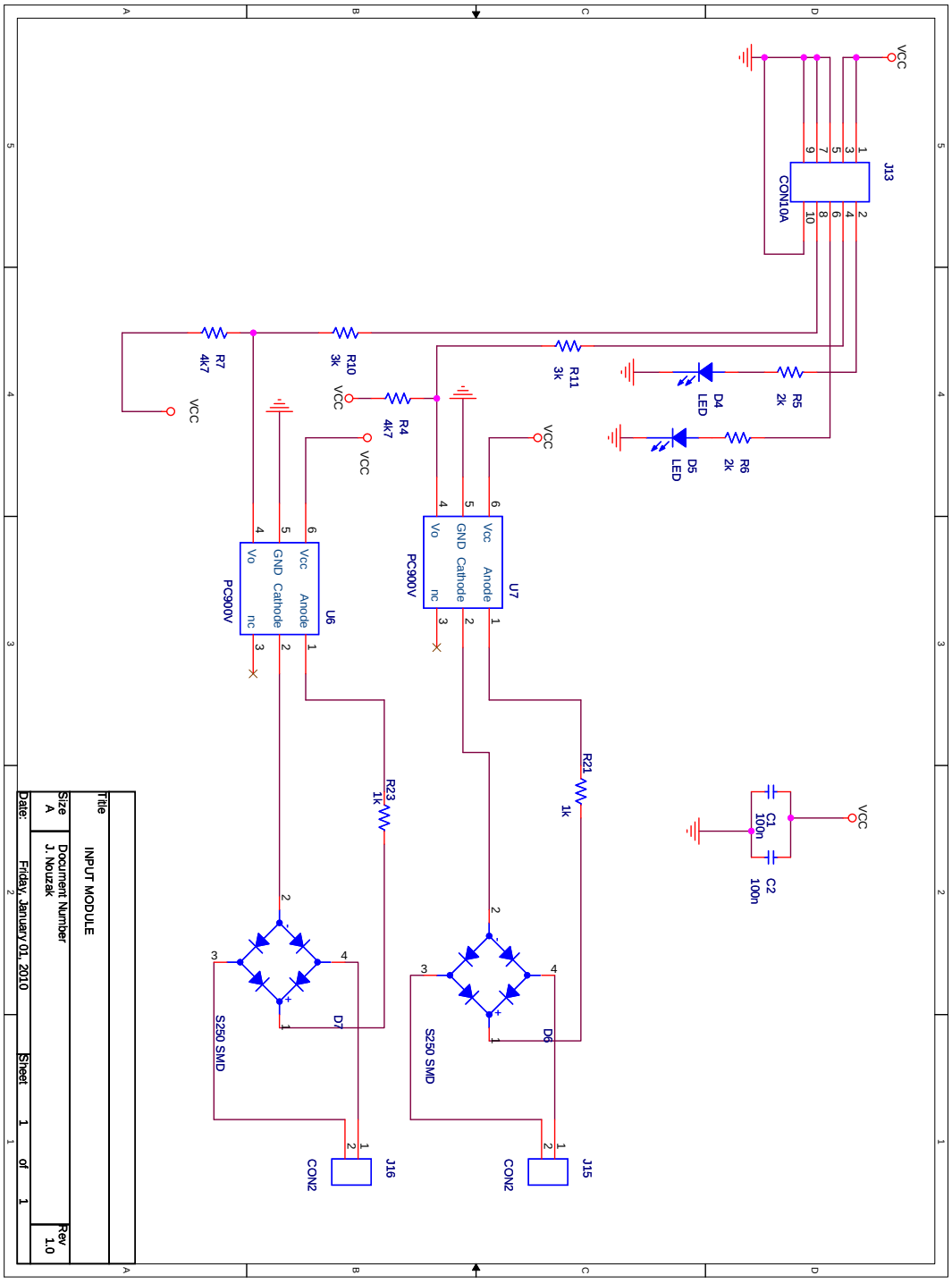
Obrázek D.2: Strana top výstupní desky



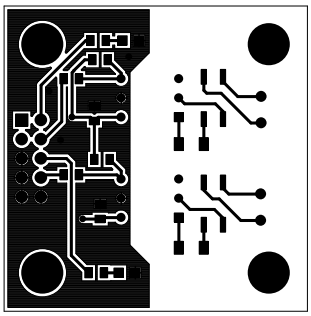
Obrázek D.3: Strana bottom výstupní desky



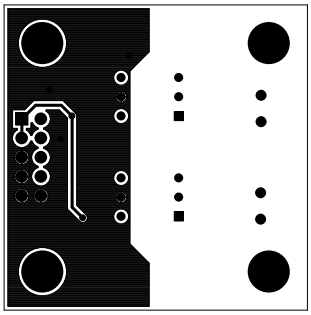
Obrázek D.4: Rozmístění součástek výstupní desky



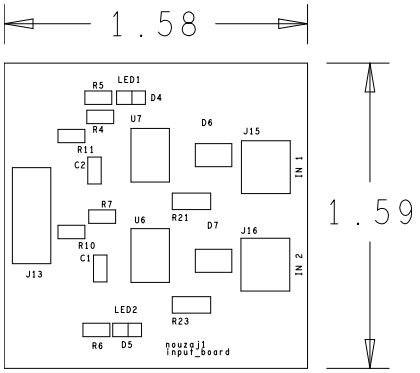
Obrázek D.5: Schéma vstupní desky



Obrázek D.6: Strana top vstupní desky



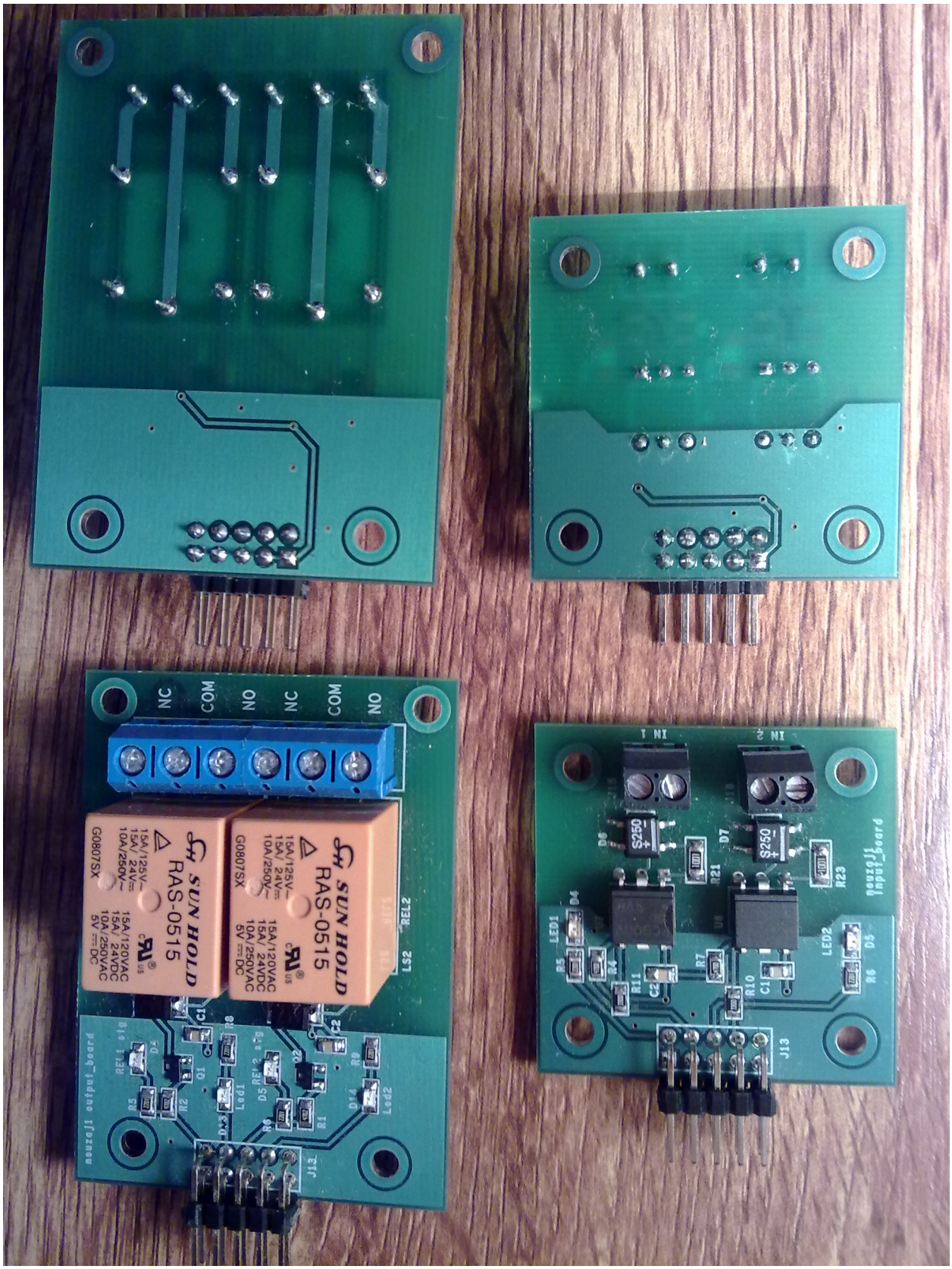
Obrázek D.7: Strana bottom vstupní desky



Obrázek D.8: Rozmístění součástek vstupní desky

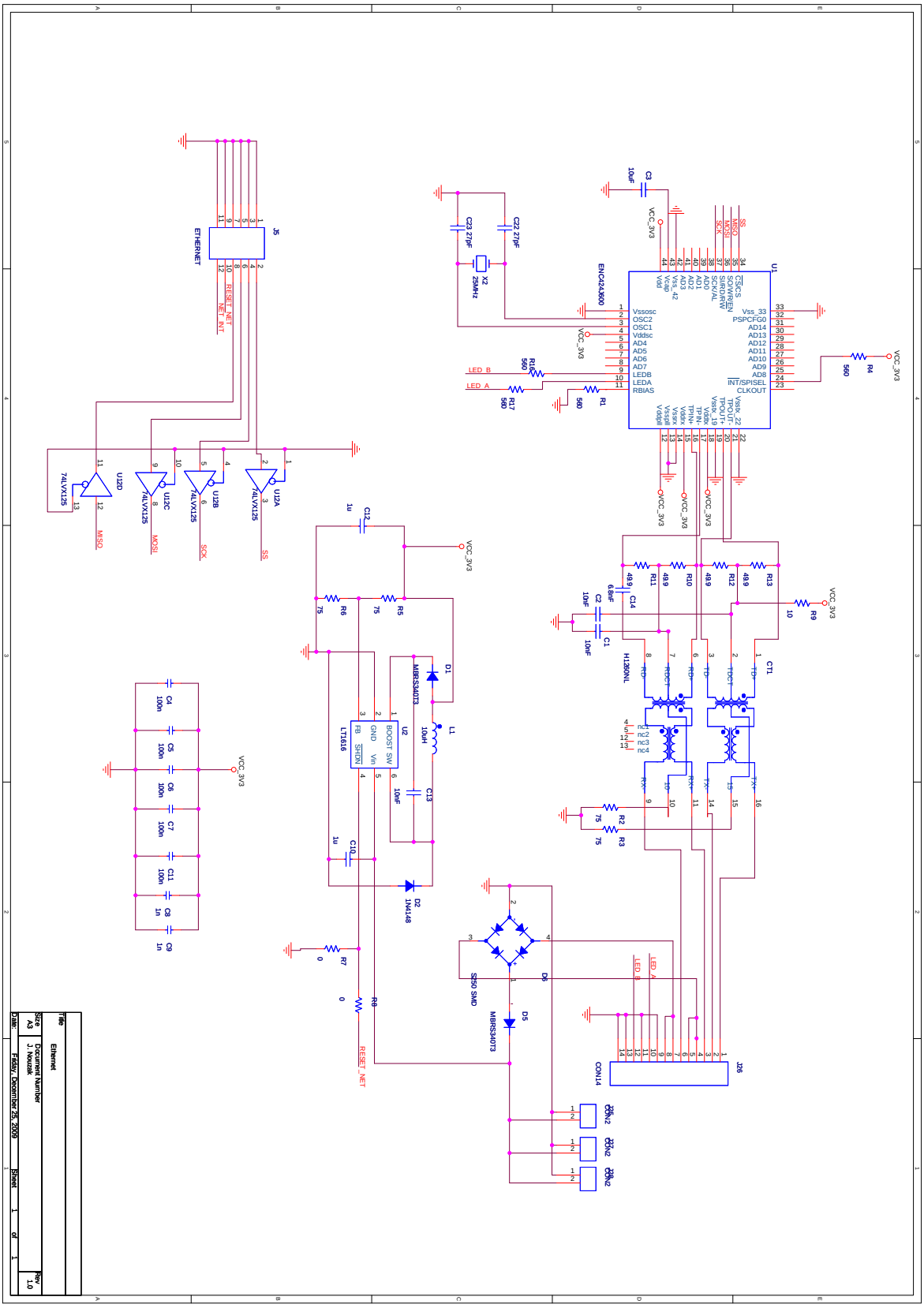
Příloha E

Osazené moduly vstupu a výstupu


NC
COM
NO
NC
COM
NO
SUN HOLD
RAS-0515
G0807SX
15A/125V~
15A/ 24V⎓
10A/250V~
15A/120VAC
15A/ 24VDC
10A/250VAC
5V ⎓DC
IN 1
IN 2
S250
S250

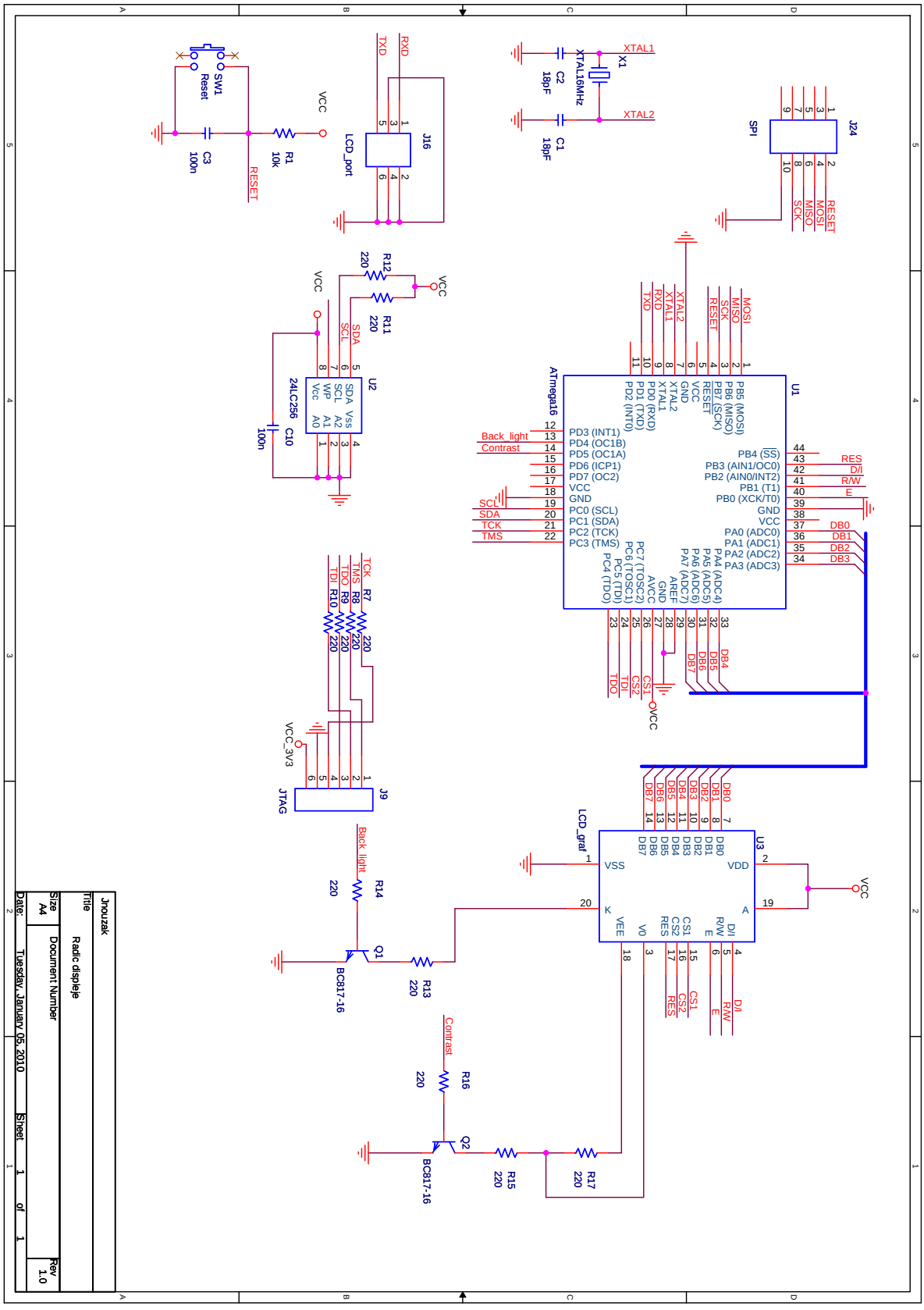
Příloha F

Modul Ethernetu


ETHERNET
J5
RESET_NET
INT_NET
74LVX125
U2A
U2B
U2C
U2D
MISO
MOSI
SCK
SS
VCC_3V3
C12
1u
R6
75
R5
75
D1
MBRS340T3
L1
U2
LT1616
GND
BOOST
SW
Vin
FB
SHDN
10uH
C13
10nF
C10
1u
D2
1N4148
R7
0
R8
0
C4
100n
C5
100n
C6
100n
C7
100n
C11
100n
C8
1n
C9
1n
C3
10uF
C22 27pF
C23 27pF
X2
25MHz
ENC424J600
Vssosc
OSC2
OSC1
Vddosc
AD4
AD5
AD6
AD7
LEDB
LEDA
RBIAS
Vss_33
PSPCFG0
AD14
AD13
AD12
AD11
AD10
AD9
AD8
INT/SPISEL
CLKOUT
LED_B
LED_A
R15
560
R17
560
R1
560
R4
560
R11
49.9
R10
49.9
R12
49.9
R13
49.9
C14
6.8nF
C2
10nF
C1
10nF
R9
10
H1260NL
CT1
R2
75
R3
75
SB350 SMD
D6
MBRS340T3
D5
CON14
J26

Příloha G

Schéma LCD modulu V2


Jnouzak
Radic displeje
Document Number
Rev 1.0
Size A4
Date: Tuesday, January 05, 2010
Sheet 1 of 1
ATmega16
U1
LCD_graf
U3
LCD_port
J16
SPI
J24
JTAG
J9
24LC256
U2
SW1
Reset
C3 100n
R1 10k
RESET
C2 18pF
C1 18pF
XTAL16MHz
X1
XTAL1
XTAL2
R12 220
R11 220
C10 100n
SDA
SCL
R7 220
R8 220
R9 220
R10 220
TCK
TMS
TDO
TDI
VCC_3V3
R14 220
R13 220
Q1
BC817-16
Back_light
R16 220
R15 220
R17 220
Q2
Contrast
VCC
RES
D/I
R/W
E
CS1
CS2
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7

Příloha H

Obsah přiloženého CD

```
CD:
├──BOARD_V2
│   └──deska_v2        Obsahuje Schema a PCD druhé verze desky
│
├──C_library           Obsahuje *.h a *.c soubory knihoven
│
├──Example             Obsahuje ukázky komunikace s deskou
│   ├──C#
│   ├──C++
│   └──JAVA
│
├──FIRMWARE
│   ├──BIN             Obsahuje *.hex soubory všech demo programmů
│   │
│   ├──DEMO            Obsahuje zdrojové soubory programu DEMO
│   │
│   ├──ECHO            Obsahuje zdrojové soubory programu ECHO
│   │
│   ├──LED_BLIK        Obsahuje zdrojové soubory programu LED_BLIK
│   │
│   └──STOPKY          Obsahuje zdrojové soubory programu STOPKY
│
├──LCD_firmware        Obsahuje Firmware pro modul LCD
│
├──ORCAD               Footprinty a padstacky použitých součástek
│   ├──footprint
│   └──padstack
│
├──PICS                Fotky modulárního systému
│
├──PODKLADY            Obsahuje projekty ORCAD se shématy a DPS
│   ├──CPU_BOARD
│   ├──Ethernet
│   ├──IN_module
│   ├──OUT_module
│   ├──programator_STK500V2
│   ├──redukce
│   └──USB_SERIAL
│
├──serial_LCD_v2       Obsahuje Schéma druhé verze LCD řadiče
│
├──simulace            Simulace částí obvodů procsorové desky
│   ├──DC-DC
│   ├──IN_opto
│   ├──IN_opto2
│   └──OUT_module
│
└──vyroba            Obsahuje soubory pro fotoploter a souřadnicovou vrtačku
    ├──CPU_BOARD
    ├──Ethernet
    ├──input_board
    ├──output_board
    ├──programator_STK500v2
    ├──redukce
    └──USB_SERIAL
```