České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Diplomová práce

Univerzální periferní deska s AVR32

Bc. Jan Šesták

Vedoucí práce: Ing. Pavel Kubalík Ph.D.

Studijní program: Elektrotechnika a informatika, strukturovaný, Navazující magisterský

Obor: Výpočetní technika, Projektování číslicových systémů

Květen 2013

Poděkování

Rád bych poděkoval panu Ing. Pavlu Kubalíkovi Ph.D. za vedení diplomové práce a jeho rady v průběhu návrhu a realizace projektu.
Dále bych rád poděkoval škole za finanční podporu projektu.
Poslední poděkování patří mé rodině za jejich podporu po celou dobu 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).

V Praze dne 9. 5. 2013 .....

Abstract

The diploma thesis is dealing with the design and realization of a universal peripheral board with 32-bit CPU AVR32 UC3A0. The board supports peripherals USB 2.0 OTG (switching between host and device), 2x RS232, RS485, SPI, I2C, SD Card, RTC, keyboard via PS / 2, connection LCD displays via UART, etc.

The thesis also includes the implementation of libraries for control blocks within the microcontroller and control of modules and chips mounted on the PCB.

Abstrakt

Diplomová práce se zabývá návrhem a realizací univerzální periferní desky s 32bitovým procesorem AVR32 UC3A0. Deska podporuje periferie USB OTG 2.0 (přepínání mezi režimy host a device), 2x RS232, RS485, SPI, I^2C, SD kartu, RTC obvod, klávesnici přes PS/2, připojení LCD displeje přes UART, atd.

Součástí práce je také návrh knihoven pro ovládání bloků uvnitř mikropočítače i ovládání modulů osazených na plošném spoji.

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Cílem této diplomové práce je navrhnout a realizovat univerzální periferní desku, která bude sloužit jako stavební prvek složitějších komunikačních nebo řídicích systémů či jako vývojový prostředek pro další studenty. Základními prvky by měly být 32bitový mikropočítač od společnosti Atmel a obvod pro měření reálného času, který bude schopný běžet na záložní napájení. Dalším úkolem práce je návrh základních knihoven a demo aplikací, jimiž bude možné otestovat základní chování navrženého systému.
Jelikož se jedná o univerzální periferní desku, jež by měla podporovat co největší množství periferií, které mohou mít větší paměťovou náročnost, bude v této práci prozkoumána možnost rozšířit vnitřní paměti mikropočítače přidáním externích modulů. Z periferií budou podporovány např.:

- RS232 – sériové propojení s PC či jiným modulem
- RS485 – sériová sběrnice vhodná pro komunikaci s různými moduly
- I^2C, SPI – sériové sběrnice pro komunikaci s rozšiřujícími moduly, externími pamětmi (EEPROM, FLASH, …), A/D či D/A převodníky, obvody pro měření reálného času, displeje, …
- Ethernet
- USB device: připojení zařízení k PC či jinému zařízení
- USB host: připojení různých USB zařízení např. Flash disk, klávesnice, …
- PS/2 – připojení externí klávesnice či myši
- Rozšiřující konektor – umožňuje připojení dalších modulů, které využívají jinou komunikační sběrnici, než je uvedeno výše.

Diplomová práce je rozčleněna do jednotlivých kapitol. První kapitola představuje úvod do celé problematiky. Druhá kapitola se věnuje specifikaci cílů diplomové práce. Následující kapitola blíže informuje o deskách dostupných na trhu. Čtvrtá kapitola se věnuje návrhu řešení. Pátá a šestá kapitola seznamuje s realizací hardwarové, resp. softwarové části. Sedmá kapitola se věnuje testování, objeveným problémů a možným vylepšením do budoucna. Osmá kapitola shrnuje výsledky celé diplomové práce.

2. Popis problému, specifikace cíle

Základním úkolem práce je navrhnout a realizovat univerzální periferní desku, která bude sloužit jako stavební prvek složitějších systémů či vývojový prostředek pro studijní účely. Systémová deska by měla zahrnovat:

- 32bitový mikropočítač od výrobce Atmel pracující na napětí 3,3V
- USB host / device
- Ethernet
- SD karta
- Sériová linka RS232
- Sériová sběrnice RS485
- Obvod pro měření reálného času (RTC) se záložním napájením
- Výstup na LCD displej
- PS/2 pro připojení klávesnice
- Rozšiřující konektor
- Externí paměť RAM

Dalším cílem práce je naprogramovat základní knihovnu funkcí pro obsluhu periferií a demo aplikaci pro předvedení funkčnosti celého zařízení.

3. Analýza dostupných řešení

Při průzkumu trhu lze nalézt velké množství vývojových desek s 32bitovými mikropočítači, které se liší výrobcem (např. Atmel, ST Microelectronics, Microchip), vnitřní architekturou (32-bit AVR UC3, ARM, 32-bit PIC) či různou podporou periferií integrovanou v hardwaru. Zbylé periferie je možné přidat pomocí externích modulů, jež jsou připojeny přes rozšiřující konektory nebo standardizované průmyslové rozraní (např. I^2C, SPI, UART, …). U externích modulů již bohužel nejde dosáhnout maximální přenosové rychlosti kvůli potřebě přesouvat data mezi jednotlivými čipy, případně kvůli přenosové kapacitě sdílené sběrnice.

Můj průzkum je omezen pouze na výrobce Atmel, který byl požadován v zadání diplomové práce, vzhledem ke zkušenostem s produkty a vývojovým prostředím této společnosti. Dále jsem vybíral pouze vývojové sady, jež mají hardwarovou podporu pro Ethernet a USB (režim host i device), aby bylo možné využít maximální možné rychlosti těchto periferií. Posledním kritériem byla integrovaná podpora pro externí paměti (SRAM a SDRAM)

Vzhledem k velkému množství produktů na trhu jsem se orientoval jen na desky, které mají integrovanou hardwarovou podporu pro většinu periferií, včetně podpory pro rychlé externí paměti (SRAM či SDRAM).

Na trhu lze najít vývojové kity přímo od výrobce mikropočítačů nebo jejich klony.

3.1. EVK1100

Obrázek 3.1: Základní deska EV1100

Vývojová sada přímo od výrobce Atmel, která je založena na mikropočítači AT32UC3A0512. Podpora pro rozhraní Ethernet, USB 2.0 mini AB, 2x RS232, I^2C, SPI. Dále lze nalézt modrý LCD displej (znakový 4x20, intenzita osvětlení ovladatelná přes PWM), 3 tlačítka, joystick, 6 LED diod a slot pro SD/MMC karty. Pro testování A/D převodníku je k dispozici fotorezistor, termistor či potenciometr. Další výhodou jsou rozšiřující paměti 8MB Flash či 32MB SDRAM. Napájení desky je možné přes USB či externí konektor. Pro programování a testování poslouží JTAG.

Cena: 5000 Kč
Odkaz: http://www.atmel.com/tools/EVK1100.aspx

Výhody:

- Integrována většina periferií
- Rozšiřující paměti
- Pole plošek s vývody čipu vhodné pro měření či připájení periferií
- Přítomnost LCD displeje

Nevýhody:

- Nepřítomnost sériové sběrnice RS485 a konektoru PS/2
- Nutnost externího modulu pro RTC
- USB host potřeba redukce na konektor miniA
- Vyšší pořizovací cena

3.2. EVK1105

Obrázek 3.2: Základní deska EVK1105

Další vývojová sada přímo od výrobce Atmel, která je založena na stejném mikropočítači, tedy AT32UC3A0512. Nechybí podpora pro USB 2.0 mini AB, Ethernet, I^2C, SPI, na desce lze nalézt slot pro SD/MMC karty, 4 LED diody, 2 tlačítka. Na rozdíl od předchozí verze jsou zde převodník USB na UART, plně barevný LCD displej QVGA (320 x 240), dotyková tlačítka, audio stereo výstup a vstup pro mikrofon. Interní paměť je opět rozšířena o 64MB Flash a 32MB SDRAM. Napájení je možné přes externí konektor či USB. Testování a programování opět možné přes JTAG. Deska se vzhledem ke svým vlastnostem hodí spíše pro oblast multimedií.

Cena: 5350 Kč
Odkaz: http://www.atmel.com/tools/EVK1105.aspx

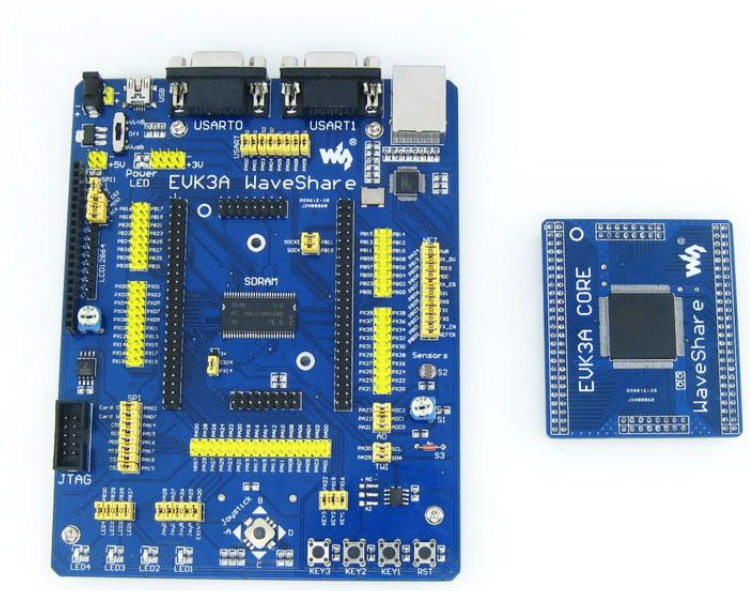
Výhody:

- Integrovány základní periferie
- Rozšiřující paměti
- Velký grafický LCD displej
- Audio vstup a výstup

Nevýhody:

- Nepřítomnost sériové linky RS232, sériové sběrnice RS485 a konektoru PS/2
- Nutnost externího modulu pro RTC
- USB host potřeba redukce na konektor miniA
- Vyšší pořizovací cena

3.3. EVK3A0512 (EVK3A0256)



Obrázek 3.3: Základní deska EVK3A0512

I tyto desky jsou založeny na stejném mikropočítači, liší se pouze velikostí interní flash paměti (tedy 512kB nebo 256kB). Deska se velmi shoduje s EVK1100, ze které nejspíše vychází. Hlavní rozdíl lze nalézt v rozdělení na dva samostatné plošné spoje spojené přes dvouřadé konektory. Jeden obsahuje mikropočítač a potřebné součástky k jeho běhu (oscilátory, kondenzátory, …). Druhý obsahuje zbytek systému – paměti Flash, 32MB SDRAM, 3 tlačítka, 4 LED diody, joystick, slot pro SD kartu, 2x RS232. Nechybí ani podpora Ethernetu, USB, I^2C, SPI, atd. Pro programování a testování opět nechybí JTAG. Lze zakoupit i s externím LCD displejem (128 x 64).

Cena s LCD: 161$ (159$)
Cena bez LCD: 117$ (114$)
Odkaz: http://www.wvshare.com/column/AVR32_DevelopmentBoard.htm

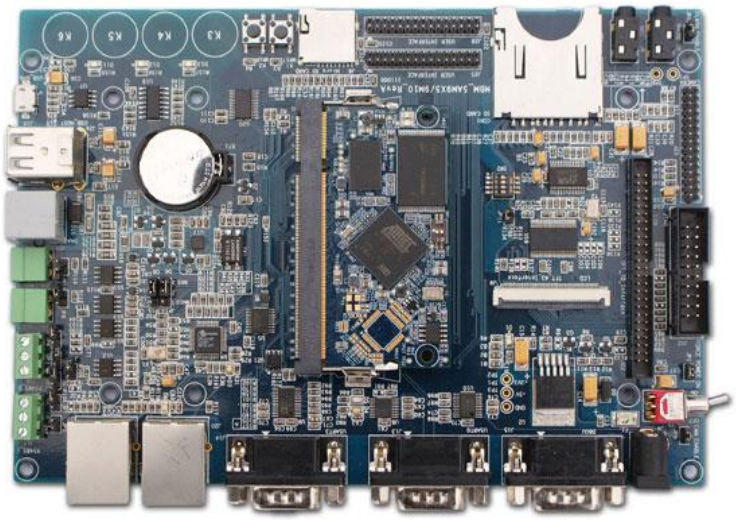
Výhody:

- Integrována většina periferií
- Rozšiřující paměti
- Rozšiřující konektory
- Možnost připojení LCD displeje

Nevýhody:

- Nepřítomnost sériové sběrnice RS485 a konektoru PS/2
- Nutnost externího modulu pro RTC
- USB host potřeba redukce na konektor miniA

3.4. MBS-SAM9G15



Obrázek 3.4: Základní deska MBS-SAM9G15

Tato deska je založena na procesoru AT91SAM9G15 (architektura ARM), který umí pracovat na frekvenci až 400 MHz, napájen je pouze 1V, ale vstupy jsou přizpůsobeny napětí 3,3V. Na desce lze nalézt paměti: 128MB DDR2, 256MB Flash, 4MB Flash, 64kB EEPROM, 256B EEPROM. Nechybí podpora Ethernetu (2x), RS232, RS485, sběrnice CAN, USB 2.0 Host (2x) a USB 2.0 OTG, 4 dotyková tlačítka, LED diody, slot pro microSD a SD karty, RTC obvod, audio výstup a konektor pro dotykový LCD displej.

Cena: 8200 Kč
Odkaz: http://www.embedinfo.com/en/list.asp?id=109

Výhody:

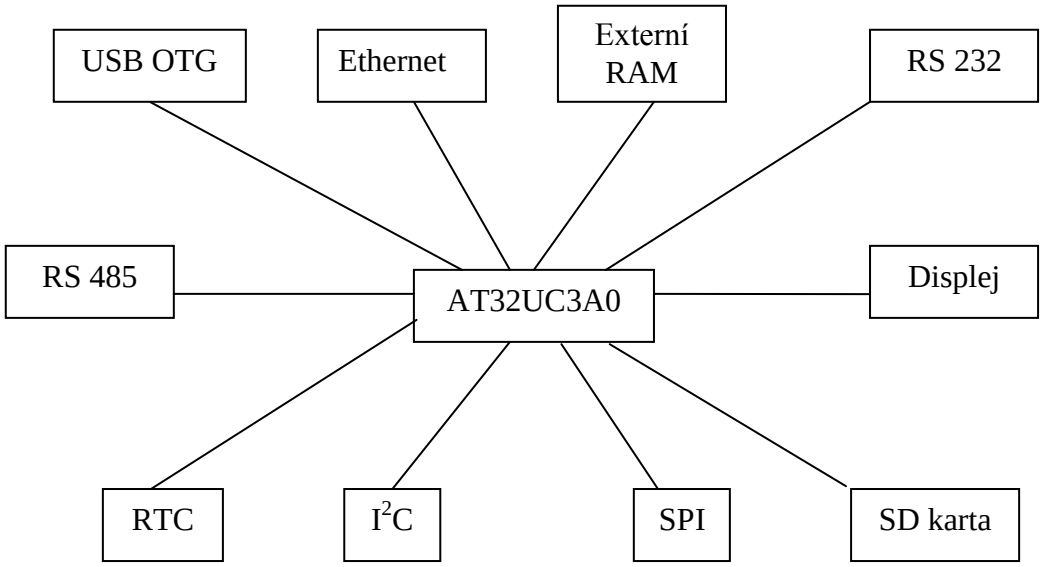
- Integrováno velké množství periferií (hodně jich je zastoupeno vícekrát)
- Rozšiřující paměti
- Rozšiřující konektory
- Možnost připojení LCD displeje
- Podpora linuxu

Nevýhody:

- Nepřítomnost konektoru PS/2
- Vysoká cena

4. Návrh řešení

Jelikož výsledkem této práce je univerzální periferní deska, která může být použita v různých aplikacích, jež se mohou lišit svými požadavky, rozhodl jsem se implementovat pouze základní či požadované periferie. Zbylé vývody čipu jsou připojeny na rozšiřující konektory. Blokové schéma, které lze vidět na obrázku 4.1, zobrazuje základní požadavky periferií v zadání této práce.



Obrázek 4.1: Předpokládané blokové schéma navrhované desky

4.1. Výběr čipů

Při výběru čipů jsem se snažil vybírat co nejvíce dle parametrů výkon / cena, abych docílil největšího možného výkonu za přiměřenou cenu. Všechny ceny v této kapitole jsou včetně DPH, platné k 1. 2. 2012 u společnosti Farnell.

4.1.1. Mikropočítač

Při výběru mikropočítače jsem se zaměřil hlavně na čipy s fyzickou podporou USB host. Od výrobce Atmel jsou k dispozici architektury ARM nebo 32-bit AVR UC3, které mají implementováno alespoň rozhraní USB OTG, jenž umožňuje přepínání mezi režimy host a device. ARM poskytuje větší výkon a některé čipy mají dokonce implementovanou podporu obou režimů USB současně, bohužel jeho vyšší cena mne vedla k využití architektury 32-bit AVR UC3.

Tuto architekturu můžeme nalézt v různých řadách, které se zaměřují na různé aplikace – automobilový průmysl, systémy s nízkou spotřebou, systémy s vysokou datovou propustností, šifrování dat atd. Svůj výběr jsem nakonec omezil na čipy z řad A0, A1 či B0, které se zaměřují na vysokou datovou propustnost a výpočetní výkon.

U řady B0 chybí fyzická podpora pro Ethernet, jež by mohla být pro některé aplikace zajímavá hlavně pro svou rychlost. Té nelze při použití externího čipu reálně dosáhnout, kvůli nutnosti přesouvat data mezi jednotlivými čipy, a proto jsem tuto možnost zavrhl. Zbylé dvě řady mají podporu stejných periferií, jediným rozdílem je podpora externích sběrnice pro paměti SRAM a SDRAM, připojených na vnitřní sběrnici u řady A0. Protože zadání vyžaduje přidání externí paměti, rozhodl jsem se využít této hardwarové podpory.

Posledním důležitým faktorem při výběru mikropočítače jsou požadavky na velikost interních pamětí, cena a dostupnost. Základní shrnutí rozdílu mezi těmito parametry lze spatřit v tabulce 4.1.

Typ	SRAM	Flash	Cena / 1 ks	Cena / 100 ks
AT32UC3A0128	32 kB	128 kB	290 Kč	210 Kč
AT32UC3A0256	64 kB	256 kB	340 Kč	260 Kč
AT32UC3A0512	64 kB	512 kB	490 Kč	290 Kč

Tabulka 4.1: Přehled čipů řady AT32UC3A0

Nakonec jsem se rozhodl osadit čip AT32UC3A0256, který je při nákupu jednoho kusu výrazně cenově dostupnější než čip s 512 kB Flash paměti, který se jeví výhodnější při nákupu více kusů, jak lze vidět z tabulky 4.1, kde již není rozdíl mezi cenami čipů tak velký a dvojnásobná velikost paměti programu může být v některých aplikací rozhodující.

4.1.2. Externí paměti

Zadání požaduje rozšířit interní RAM paměť mikropočítače o externí modul. K mikropočítači lze připojit přes vnější paměťovou sběrnici až čtyři moduly SRAM a jeden modul SDRAM, který může obsahovat čtyři interní banky. Oba typy pamětí lze připojit přes 8 nebo 16bitovou datovou sběrnici, kdy širší sběrnice umožní dosáhnout většího výkonu a je tedy výhodnější.

Typ	Název	Kapacita	Cena / 1 ks	Cena / 100 ks
SRAM	GS71116AGP-12	128 kB	95 Kč	70 Kč
	CY7C1019DV33-10VXI	128 kB	90 Kč	70 Kč
	AS7C34098A-12JIN	512 kB	140 Kč	100 Kč
SDRAM	A43L0616BV-7F	2 MB	45 Kč	35 Kč
	A43L2616BV-7F	8 MB	75 Kč	55 Kč
	A43L3616AV-7F	16 MB	100 Kč	70 Kč
	A43L4616AV-7F	32 MB	120 Kč	85 Kč

Tabulka 4.2: Přehled modulů SRAM a SDRAM

Při porovnání těchto modulů nesmíme zapomínat na vnitřní stavbu a požadavky na časování. SRAM, tedy statická paměť, je složena z bistabilního klopného obvodu a její přístupová doba je velmi nízká. SDRAM, neboli synchronní dynamická paměť, je založena na principu uchovávání náboje na kondenzátoru, který se samovolně vybíjí, proto tento druh pamětí vyžaduje automatickou obnovu dat. Přístupová doba dynamické paměti je nižší než u statické paměti, naopak výroba je jednodušší a umožňuje dosáhnout mnohem vyšší kapacity.

Svůj výběr jsem tedy orientoval směrem k dynamickým pamětem, které za stejnou cenu nabízejí mnohonásobně vyšší kapacitu. Vybral jsem model s 32MB pamětí, která je pro adresní rozsah mikropočítače maximální a dle mého názoru plně dostačující.

Další paměťové moduly (SRAM, Flash či EEPROM) na desce nebudou osazovány. Pro uchovávání většího množství dat bude lepší využít paměťové karty či USB flash disky, které mají velkou kapacitu a umožňují rychleji přenášet data.

4.1.3. Ethernet

Mikropočítač má v sobě implementovanou hardwarovou podporu MAC vrstvy Ethernetu, díky tomu můžeme komunikovat se síťovými zařízení s využitím vyšších rychlostí (až 100Mb/s), než v případě softwarového řešení, kdy je třeba přenášet všechna data mezi interní pamětí mikropočítače a externím čipem. Integrovaná MAC vrstva umožňuje použít pro přenos dat DMA přenosy společně s generováním přerušení a usnadňuje tak rychlost zpracování dat.

Desku je třeba doplnit ještě o fyzickou vrstvu, která převádí logické úrovně na napěťové a naopak, galvanické oddělení a konektor RJ45. Galvanické oddělení lze nalézt jako samostatnou součástku nebo integrované přímo v konektoru. Konektor s integrovaným galvanickým oddělením je sice dražší, ale velmi usnadňuje návrh plošného spoje, jeho ceny se pohybuje kolem 150 Kč.

Komunikace s fyzickou vrstvou může probíhat přes rozhraní MII (Media Independent Interface) nebo RMII (Reduced Media Independent Interface). Rozdíl, jak již název napovídá, je v počtu využitých vodičů. Dalším rozdílem je frekvence referenčního hodinového signálu. Jelikož vývody pro MII rozhraní se kryjí s paměťovou sběrnicí, je nutné využít fyzickou vrstvu s rozhraním RMII, např. KSZ8031RNLI, kterou lze sehnat za 40 Kč.

4.1.4. USB

Dalším požadavkem je realizace USB rozhraní. Mikropočítač zahrnuje podporu USB OTG, které odpovídá specifikaci 2.0, ale pouze s využitím maximální přenosové rychlosti 12 Mb/s. Označení OTG (On-The-Go) umožňuje mikropočítači pracovat v režimu host nebo device, případně mezi nimi přepínat.

Pro detekci režimu je možné využít vstup USB_ID, který je přiveden z konektoru USB miniA. Vstup je pak detekován podle typu kabelu zastrčeného do konektoru, pro režim host log 0 a pro režim device log 1. Velkou nevýhodou tohoto řešení je redukce, kterou je třeba použít v režimu USB host. Další variantou je osazení obou konektorů (A i B-mini) a detekci realizovat pomocí propojky či přepínače.

Pro napájení USB zařízení při využití desky v hostitelském režimu je podle specifikace USB 2.0 nutno poskytnou při napětí 5V maximální odběr proudu 500mA. Některé zařízení ale neodpovídají specifikaci a počítají s napájením v PC přímo ze zdroje, kde není odběr proudu omezen. Pro režim device je vhodné umožnit napájet desku z hostitelského počítače, kdy by měl být odběr 500mA plně dostačující. Pro ochranu napájení desky a počítače je vhodné použít ochranné pojistky. Na trhu se nacházejí trvalé a vratné pojistky. Hlavní nevýhodou trvalých pojistek je jejich trvalé znehodnocení, vratné pojistky jsou sice dražší, ale jejich schopnost opětovné funkce odstraňuje nutnost vyměnit proraženou trvalou pojistku.

4.1.5. RTC – hodiny reálného času

Pro měření reálného času v aplikaci lze použít obvod integrovaný v mikropočítači. Nevýhoda ovšem vzniká při výpadku napájení, kdy by muselo dojít k přepnutí zdroje na záložní baterii. Dále by bylo potřeba převést mikropočítač do režimu spánku a ukončit komunikaci s periferiemi, aby se minimalizovala spotřeba a nedošlo k rychlému vybití záložního zdroje.

Další variantou je využívat interní obvod mikropočítače v rámci aplikace a externí čip pro měření času i v případech výpadku hlavního napájení. Tyto čipy jsou stavěny s ohledem na spotřebu a lze je tedy dlouhodobě napájet z baterií. Pro svou aplikaci jsem vybral čip MCP79410, se kterým lze komunikovat přes I^2C sběrnici. Tento čip se v případě výpadku hlavního napájení automaticky přepíná na záložní zdroj. Cena je 35 Kč.

4.1.6. RS232 a RS485

Obě tato rozhraní jsou na straně mikropočítače podporována jednotkou UART (někdy též USART – synchronní varianta). Komunikace začíná vysláním start bitu (log 0), následují data (délka může být proměnná, ale nejčastěji se jedná o 8 bitů) a končí stop bitem (log 1).

Sériová linka RS232 již dnes není tolik rozšířena, ale ve světě vestavných systémů patří díky snadné implementaci stále mezi nejpoužívanější komunikační prostředky. V asynchronní verzi navíc stačí pouze tři vodiče, dva pro přenos dat a jeden vodič pro uzemnění. Na plošný spoj stačí přidat obvod pro převod mezi

logickými a napěťovými úrovněmi, např. obvod MAX3232 pracující na napětí 3,3V s cenou 35 Kč, a konektor CAN9.

Sériová sběrnice RS485 umožňuje propojení více jednotek a využívá pouze dvouvodičové vedení. Tento standard slouží hlavně pro získávání dat od měřících modulů v průmyslovém prostředí. Vysílání a příjem dat probíhá po jednom vedení, proto je třeba ovládat směr komunikace, tuto vlastnost přímo podporuje mikropočítač, a tak není ani tato komunikace příliš složitá. Plošný spoj musíme tedy doplnit pouze o obvod pro převod úrovní, např. SN65HVD10DG, pracující na napětí 3,3V s cenou kolem 60 Kč, konektor a ovladatelný zakončovací rezistor sběrnice.

4.1.7. **Displej**

Existuje mnoho druhů displejů, některé jsou ovládány pomocí paralelního rozhraní, kdy je komunikace specifická pro jednotlivé typy řadičů displeje, nebo pomocí standardních rozhraní – I^2C, SPI, UART.

Deska bude osazena konektorem pro připojení displeje přes UART s firmwarem popsaným v diplomové práci „Konfigurovatelný řídící modulární systém“ [4].

4.1.8. **SD karta**

SD karty lze dělit podle velikosti na SD, mini SD a micro SD, z nichž jsou nejvíce používané formáty SD a micro SD. Klasický formát se používá hlavně v digitálních fotoaparátech a přenosných počítačích, druhý naopak využívá svých menších rozměrů v zařízeních jako mobilní telefon či MP3 přehrávač. Micro SD karty lze do čteček klasického formátu připojit pomocí redukce.

Dále lze karty dělit podle technologie (SD, SDHC a SDXC). Technologie se liší maximální možnou kapacitou a rychlostí. Maximální kapacita karet SD je 4 GB, SDHC 32 GB a pro technologii SDXC zatím 256 GB, přičemž by měla umožnit velikost karet až 2 TB.

Komunikace probíhá přes paralelní rozhraní (1 nebo 4 datové vodiče) nebo přes SPI, jelikož SPI šetří vývody a navíc je podporováno mikropočítačem, stačí doplnit plošný spoj pouze o konektor.

4.1.9. **Klávesnice**

Standardní klávesnice umožňuje přidat velký počet vstupů s využitím minimálních nákladů. Je možné ji připojit pomocí PS/2 rozhranní nebo pomocí USB. Vzhledem k jednoduchosti protokolu PS/2 oproti složitému USB, bude na desce osazen konektor PS/2. Toto rozhranní vyžaduje přidat pouze dva pull-up rezistory a náklady na něj jsou tedy minimální. Ani 5V napájení klávesnice není nevýhodou díky podpoře 5V vstupů na některých vývodech mikropočítače.

4.1.10. Spotřeba a napájení

Napájení mikropočítače vyžaduje 1,8V (pro PLL a jádro) a 3,3V pro I/O. Pro generování napětí 1,8V lze využít externí nebo interní regulátor. Jelikož je toto napětí nutné pouze pro mikropočítač, tak na pokrytí spotřeby plně dostačuje vnitřní regulátor.

Zdroj pro USB v režimu host by dle specifikace USB 2.0 [22] měl poskytnout napětí 5V a proud 500mA. Některé zařízení nedodržují toto doporučení a předpokládají, že je port USB napájen přímo ze zdroje počítače a odebírají i 1A, proto je lepší vybavit desku zdrojem s možností odběru většího proudu a zároveň zabránit odběru nadměrného proudu – např. pomocí vratné pojistky.

V režimu USB device by měla deska umožnit napájení z hostitelského systému nebo pomocí externího zdroje, jehož vstup by měl být chráněn proti opačné polaritě vstupního napájení. Plošný spoj je tedy vhodné doplnit o spínaný zdroj pro napětí 5V, který umožňuje větší rozsah vstupního napětí a nemá velké tepelné ztráty. Pro napětí 3,3V již stačí stabilizátor z 5V. Vzhledem k maximálním odběrům proudů osazených součástek, které jsou shrnuty v tabulkách 4.3 a 4.4, jsem se rozhodl osadit spínaný zdroj pro napětí 5V s maximálním odběrem 3A a lineární stabilizátor pro napětí 3,3V s odběrem až 1A.

Název	Popis	Spotřeba
AT32UC3A0256	CPU	550 mA
A43L4616AV-7F	SDRAM	150 mA
KSZ8031RNLI	Ethernet	50 mA
MAX3232	RS232	10 mA
SN65HVD10DG	RS485	20 mA
MCP79410	RTC	1 mA
Celkem		**781 mA**

Tabulka 4.3 Spotřeba proudu pro napětí 3,3V

Název	Spotřeba
Zdroj 3,3V	1000 mA
USB	1000 mA
Klávesnice PS/2	100 mA
Celkem	**2100 mA**

Tabulka 4.4: Spotřeba proudu pro napětí 5V

4.1.11. Náklady na výrobu

V této podkapitole jsem se pokusil odhadnout náklady na výrobu periferní desky prototypovou i sériovou výrobu. Velikost byla vzhledem k počtu periferií a konektorů stanovena na 1,5dm^2. Náklady na osazení plošného spoje počítají se 150 součástkami,

kdy z vrchní strany bude 30 klasických a 80 SMD a ze spodní strany 40 SMD součástek. Výsledná cena je přepočtena v obou případech na jeden kus.

Název	Cena / 1ks	Cena / 100 ks
IO obvody	740 Kč	530 Kč
Konektory	300 Kč	250 Kč
Pasivní Součástky	70 Kč	50 Kč
Plošný spoj	750 Kč	90 Kč
Osazení	630 Kč	100 Kč
Celkem:	**2490 Kč**	**1020 Kč**

Tabulka 4.5: Náklady na výrobu plošného spoje

4.2. Použití programovacího jazyka

Mikropočítač lze programovat použitím asembleru nebo pomocí programovacích jazyků C/C++. Jazyk C++ přináší v programování mnoho výhod např. objektově orientované programování. Nicméně tyto výhody se ve světě vestavných systémů příliš neprojeví, a proto postačí klasická verze jazyka C. Obrovskou výhodou těchto jazyků je možnost přenositelnosti kódu mezi různými platformami a hlavně větší přehlednost kódu. Při návrhu knihoven je vhodné využívat typy z knihovny „stdint.h“ (např. int32_t či uint32_t), které mají na všech systémech stejnou velikost.

4.3. Programování mikropočítače

Programování mikropočítače je možné přímo z vývojového prostředí s využitím JTAG programátorů. Hlavní nevýhodou této metody je pořizovací cena originálních nástrojů, která je uvedena v tabulce 4.5.

Název	Cena
AVR Dragon	1 300 Kč
JTAGice mkII	11 200 Kč
JTAGICE3	3 000 Kč

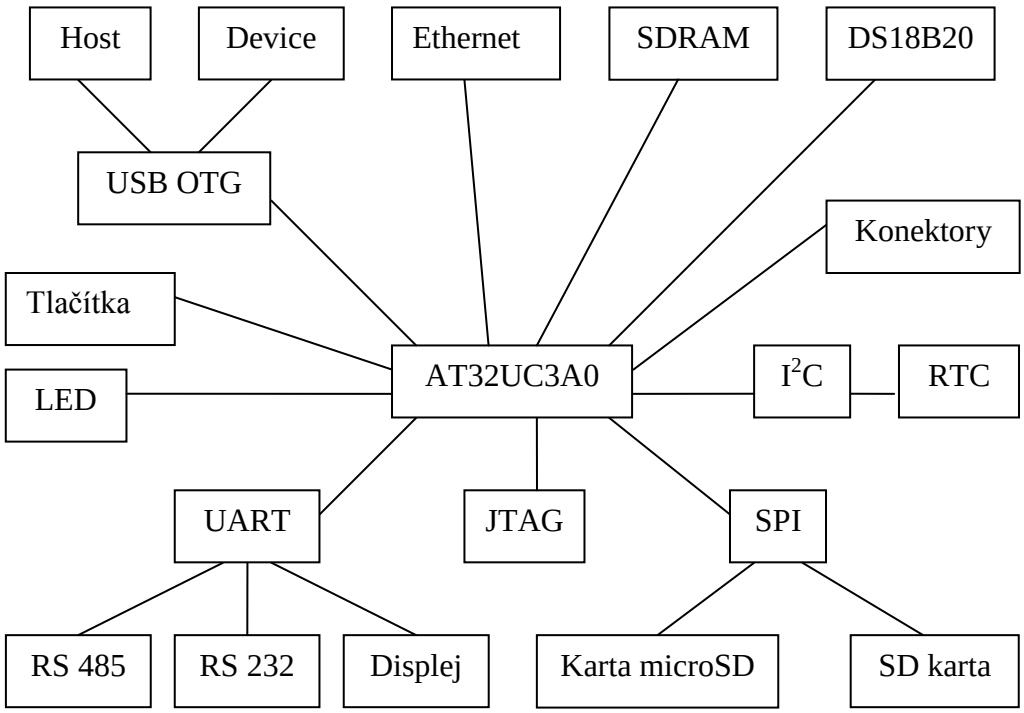
Tabulka 4.6: Podporované JTAG programátory

Další možností je využít USB bootloader, který musí být součástí paměti mikropočítače. Pro první variantu stačí připojit desku k počítači s programy Flip a BatchISP – verze USB DFU Bootloader [20]. Druhá varianta umožňuje programovat paměť mikropočítače pomocí USB Flash disku – verze UC3 USB Mass Storage Bootloader [21].

5. HW Realizace

V této kapitole se detailně věnuji zapojení jednotlivých částí systému. Vzhledem k velikosti schématu jsou v této kapitole rozebrány a zobrazeny pouze nejdůležitější části systému, celé schéma je součástí příloh této práce.

5.1. Zapojení periferní desky

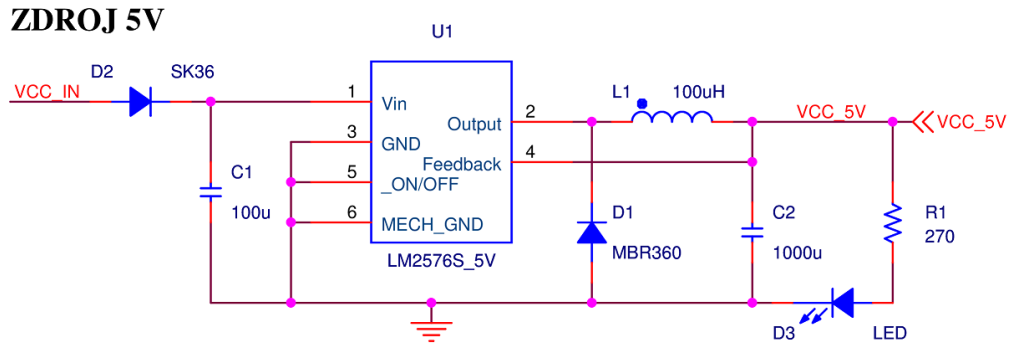


Obrázek 5.1: Blokové schéma navrhovaného systému

Blokové schéma navrhovaného systému (obr. 5.1) zobrazuje propojení jednotlivých součástí desky. Jádrem celého systému je mikroprocesor AT32UC3A0 firmy Atmel, který ovládá veškerou činnost periferní desky. Dále jsou k němu připojeny čtyři tlačítka, umožňující základní ovládání, a čtyři LED diody. Zapojení systému je detailněji zpracováno v této kapitole.

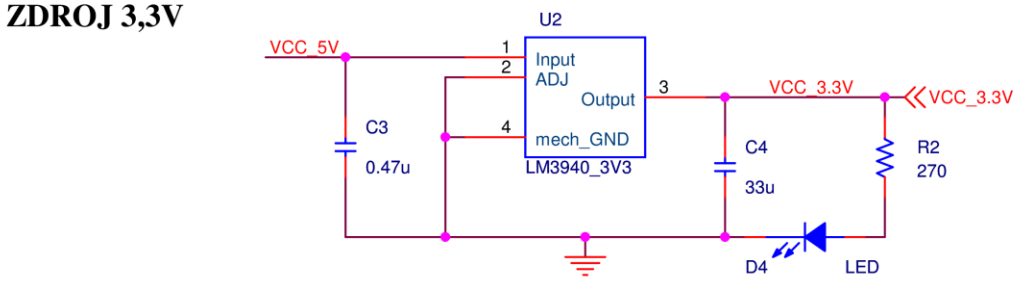
5.1.1. Napájecí zdroj

Napájení desky se skládá ze dvou částí. První reguluje napětí ze vstupního konektoru na 5V, které je použito pro USB či klávesnici přes PS/2. Ve druhé části je napětí 5V převáděno na 3,3V, které využívá hlavní část desky – mikropočítač, paměť SDRAM, RTC, převodníky RS232 a RS485, SD karta atd. Obě větve jsou vybaveny signalizačními LED diodami (D3, D4) jejichž proud je omezen odpory R1 a R2. Vstupní napájení je možné přivést pomocí konektoru (J2) nebo pomocí svorkovnice (J1). Další dvě svorkovnice J3 a J4 slouží k vyvedení 5V a 3,3V napájení, které lze použít pro další moduly.



Obrázek 5.2: Napájení obvodu – zdroj 5V

První část se skládá ze spínaného zdroje LM2576S, dvou diod (D1, D2), cívky L1 a dvou kondenzátorů (C1, C2). Výhoda spínaného zdroje, jehož výstupní proud při napětí 5V je až 3A, je malý ztrátový výkon a velký rozsah vstupního napětí, které dosahuje u této verze až 40V. Vstupní napětí je chráněno diodou D2 proti otočení polarity. Zapojení odpovídá dokumentaci čipu LM2576S.



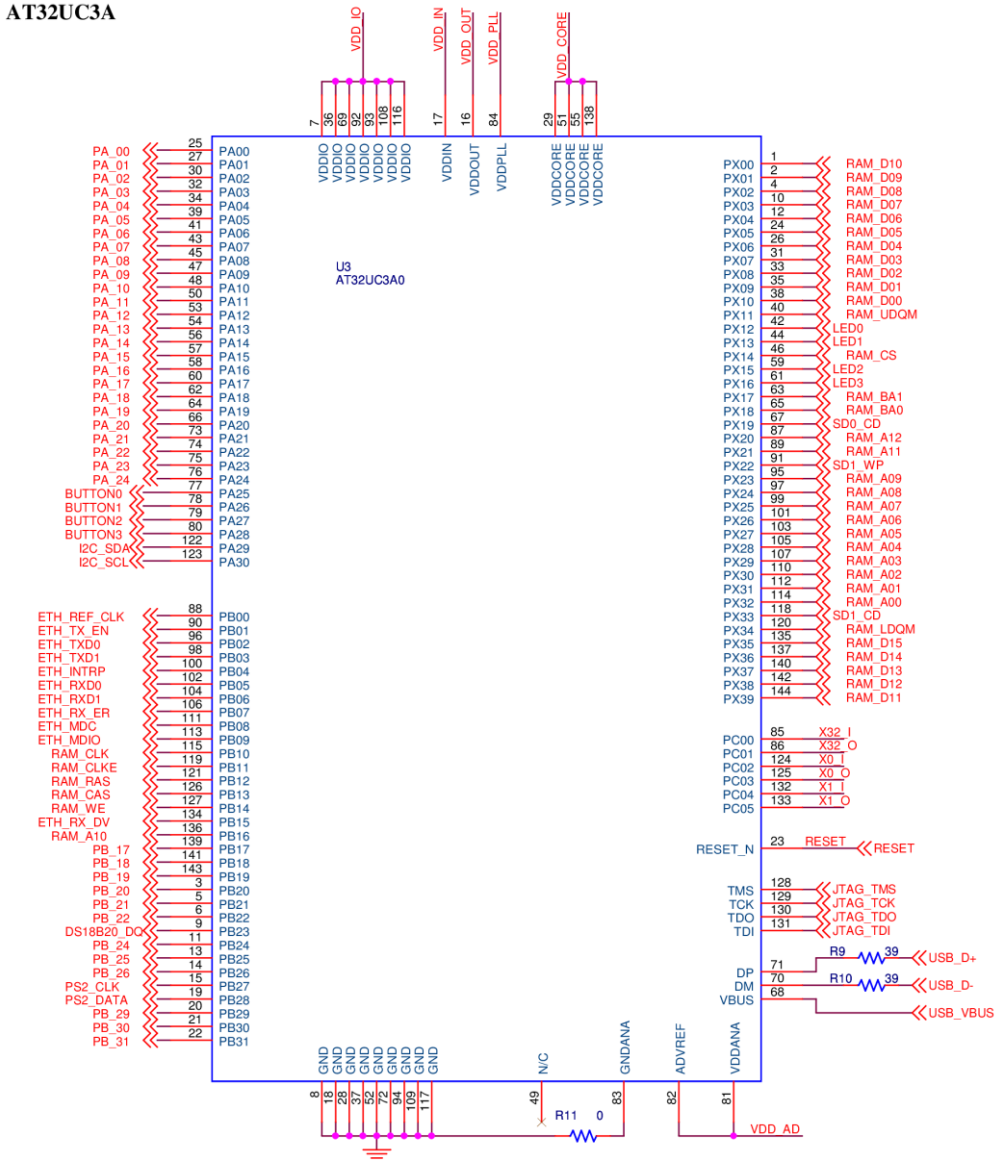
Obrázek 5.3: Napájení obvodu – zdroj 3,3V

Druhá část je tvořena lineárním stabilizátorem LM3940 a dvěma kondenzátory (C3, C4). Nevýhodou lineárního stabilizátoru je ztrátový výkon, odpovídající součinu proudu protékajícího stabilizátorem a úbytku napětí. Výhodou je naopak menší počet součástek v obvodu. Tato varianta se hodí pro malé rozdíly vstupního a výstupního vstupního a výstupního vstupního a výstupního napětí při odběru menších proudů.

5.1.2. Mikropočítač AT32UC3A0

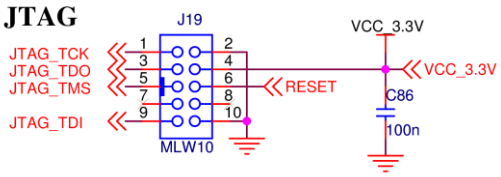
Napájení mikropočítače je zajištěno napětím 3,3V přivedeným na vstupy VDDIO, VDDANA a na vstup vnitřního napěťového regulátoru VDDIN. Výstup regulátoru VDDOUT je připojen na vstupy VDDPLL a VDDCORE. Jednotlivé skupiny vstupů jsou odděleny nulovými odpory (R3 – R7) a dle dokumentace doplněny blokovacími kondenzátory (C5 – C39).

Pro generování hodin je deska osazena krystalem Y1 s frekvencí 12 MHz, jenž slouží ke generování hlavních hodin, případně lze osadit krystal Y2, který může být použit ke generování hodin periferií či jádra, ale pouze s využitím vnitřní PLL. Poslední krystal Y3 slouží pro měření reálného času a jeho frekvence je 32,768 kHz.



Obrázek 5.4: Zapojení AT32UC3A0

Programování a debugování čipu je umožněno připojením JTAG programátoru, přes 10pinový konektor, jehož zapojení odpovídá zařízením od společnosti Atmel.

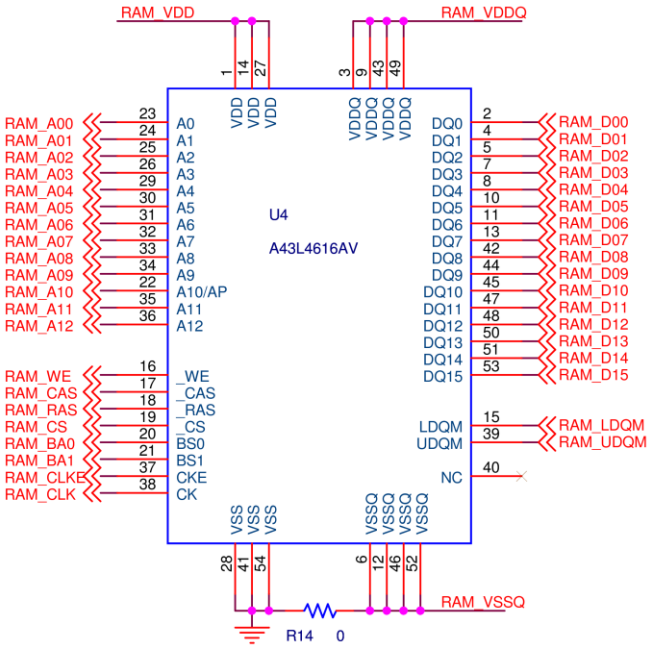


Obrázek 5.5: Zapojení JTAG konektoru

5.1.3. Paměť SDRAM

Napájení paměti A43L4616AV je rozděleno na napájení čipu a datové části. Obě skupiny napájecích vstupů jsou připojeny ke 3,3V větvi přes nulové odpory (R12, R13) a jsou vybaveny 10nF a 100nF blokovacími kondenzátory pro každý vstup. Napájení čipu využívá kondenzátory C47 – C50, datová část pak C53 – C60. Zem datové části je též oddělena nulovým odporem R14.

Adresní, kontrolní a datové vývody paměti jsou připojeny přímo na vývody EBI sběrnice mikropočítače, která umožňuje využít interní SDRAM kontrolér.



Obrázek 5.6: Zapojení paměti SDRAM

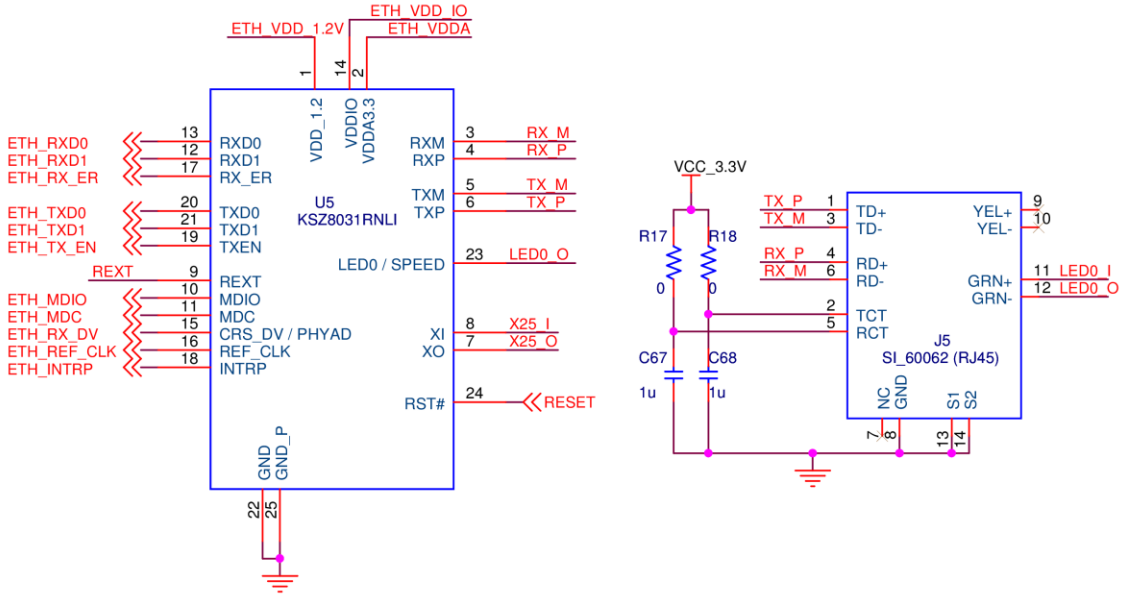
5.1.4. Ethernet

Zapojení je opět usnadněno pomocí vyvedeného rozhraní pro fyzickou vrstvu, kterou je možné připojit přes MII (Media Independent Interface) nebo přes její redukovanou podobu RMII. Kvůli kolizi vývodů pro MII a signálů EBI sběrnice, přes kterou je připojena paměť SDRAM, je nutné využít verzi RMII.

Fyzická vrstva KSZ8031RNLI od společnosti Micrel vyžaduje navrhovanou desku doplnit o blokovací kondenzátory, pull-up a pull-down rezistory, oscilátor s frekvencí 50 MHz nebo krystal s frekvencí 25 MHz a konektor RJ45 s magnetickou ochranou proti přepětí, která je integrována uvnitř konektoru. Magnetická ochrana vyžaduje pouze přidat filtrační kondenzátory a oddělující odpory (C67, C68, R17 a R18).

Blokovací kondenzátory je nutné doplnit pro jednotlivé napájecí vývody. Čip obsahuje integrovaný napěťový regulátor na 1,2V pro napájení jádra, který je nutno doplnit o kondenzátory 2,2µF (C65) a 100nF (C66). Další kondenzátory 22µF a 100nF jsou potřeba pro napájecí vstupy digitální a analogové části (C61 a C62, resp. C63 a C64). Tyto napájecí vstupy jsou opět kvůli odstranění rušení připojeny přes nulové odpory (R15 a R16) k napájecí větvi 3,3V.

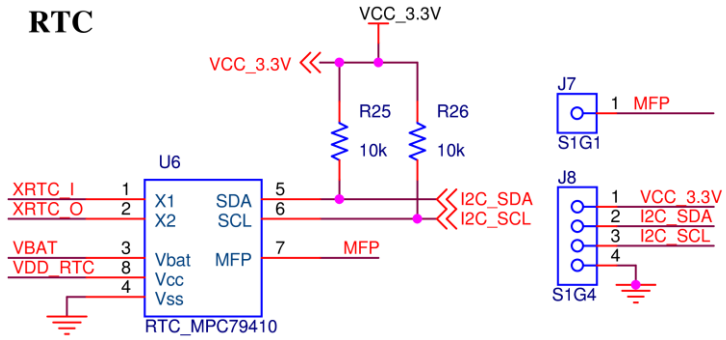
Posledními součástkami, jimiž je nutno doplnit navrhovaný plošný spoj, jsou rezistory. Ve variantě pull-up jsou vyžadovány pro signalizační diodu konektoru (R19), signál MDIO (R20), který je využíván k nastavení fyzické vrstvy, a pro vývod přerušení (R21). Varianta pull-down je vyžadovaná pro vývody REXT (R22) a RX_DV (R23). První slouží k nastavení výstupního proudu a jeho hodnota musí být 6,49kΩ. Druhý slouží k nastavení spodních bitů adresy fyzické vrstvy na nulu.



Obrázek 5.7: Zapojení fyzické vrstvy Ethernetu a konektoru RJ45

5.1.5. I²C a RTC

Sběrnice I²C je velmi jednoduchá na implementaci, jelikož se skládá pouze ze dvou vodičů – datový SDA a hodinový SCL. Dále vyžaduje jen dva 10kΩ pull-up rezistory připojené k napájecí větvi 3,3V, které zajišťují klidový stav sběrnice. Ovládání je opět usnadněno pomocí integrovaného bloku TWI.

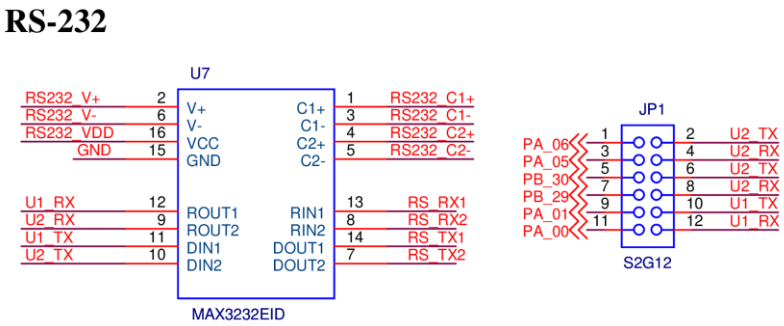


Obrázek 5.8: Zapojení sběrnice I^{2}C a obvodu MPC79410

Obvod MPC79410 slouží k měření reálného času. Jeho výhodou je možnost připojení baterie k oddělenému vstupu, v případě výpadku hlavního napájení pak dojde k přepnutí na záložní zdroj a k zakázání komunikace po sběrnici. Nevýhodou je požadavek na krystal s frekvencí 32,768 kHz. Vývod MFP umožňuje generovat hodinový signál s různou frekvencí, nebo slouží jako výstupní signál nastavených alarmů.

5.1.6. RS232

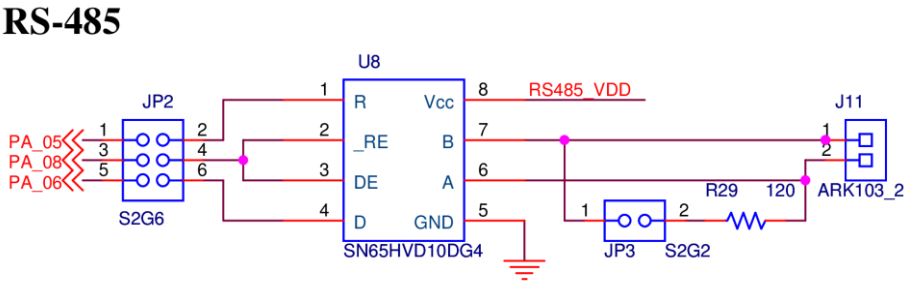
Mikropočítač obsahuje čtyři integrované moduly UART, z nichž jeden nelze využít kvůli kolizi vývodů se signály pamětí SDRAM a Ethernetu. K převodu signálů mezi logickými hodnotami a napěťovými úrovněmi RS232 je nutné přidat obvod MAX3232. Tento obvod je nutné doplnit o pět 100nF kondenzátorů (C74 – C78), které jsou vyžadované zdroji integrovanými v převodníku. Tyto zdroje pracují na principu nábojové pumpy a generují z 3,3V napětí ± 7V. Logické signály jsou k mikropočítači připojeny přes propojky (JP1), které umožňují volitelné zapojení případně vytvoření zpětných vazeb. Ke konektoru J9 lze připojit UART1 či UART2, ke konektoru J10 lze připojit UART0. Oba konektory jsou typ samec, tedy stejné jako na počítači či redukci z USB, a proto k propojení vyžadují křížený kabel.



Obrázek 5.9: Zapojení RS232

5.1.7. **RS485**

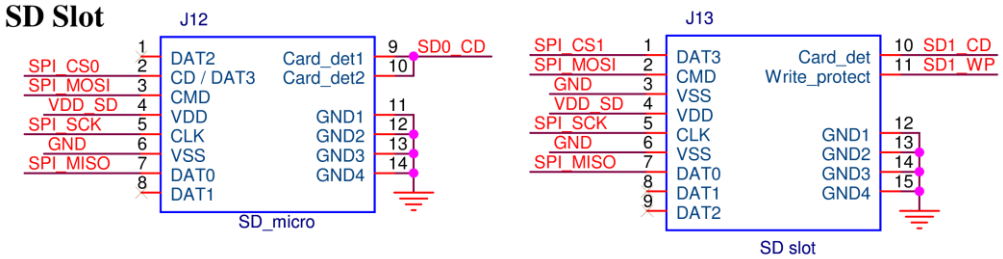
Průmyslová sběrnice RS485 využívá polo-duplexní dvouvodičové diferenciální vedení, kvůli této vlastnosti je nutné ovládat směr vysílání. Dále je třeba přidat obvod pro převod na napěťové úrovně, volitelně zapojitelný zakončovací odpor (R29) připojený přes propojku JP3 a konektor. Použitý obvod SN65HVD10DG4 pracuje na napětí 3,3V a umožňuje samostatné ovládání vysílače, aktivní v log 1, a přijímače, aktivní v log 0. Jelikož je možné využít jenom jeden směr a pro ovládání směru jsou využity opačné úrovně, lze tyto vývody spojit v jeden ovládací signál. Modul UART1 mikropočítače zahrnuje podporu pro automatické ovládání směru komunikace na vývodu RTS. Jelikož lze tento modul použít i pro komunikaci po RS232, jsou signály opět připojeny přes propojky JP2.



Obrázek 5.10: Zapojení RS485

5.1.8. **SD karta**

Jelikož jsou SD karty často používané v klasickém formátu, ale i ve verzi microSD, jsou na desce osazeny konektory pro oba typy. Tuto volbu usnadnila i komunikace po sériové sběrnici SPI, kdy je třeba využít navíc pouze jeden ovládací signál. Oba konektory umožňují detekci vložení karty a formát SD navíc zakázat zápis pomocí přepínače na straně karty. Tyto signály, stejně jako signály sběrnice SPI, vyžadují připojení 100kΩ pull-up rezistorů (R31 – R38).

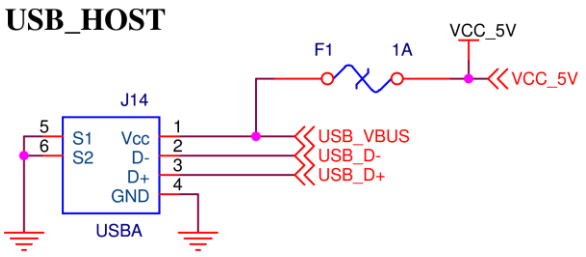


Obrázek 5.11: Zapojení konektorů pro karty SD a microSD

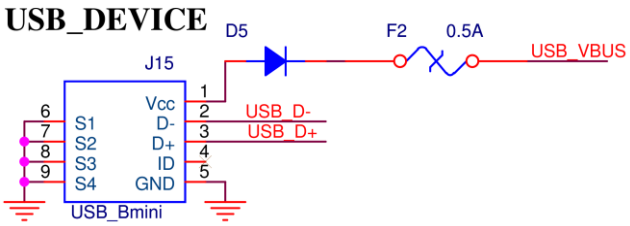
5.1.9. USB

USB kontrolér je integrován uvnitř mikropočítače a vyžaduje pouze dva odpory s hodnotou 39Ω (R9, R10) k odstranění odrazů na vedení. Kvůli odstranění potřeby redukce jsou osazeny konektory typu A pro režim USB host a typ miniB pro režim USB device. Napájení zařízení připojených k desce pracující v režimu host, tedy konektor typu A, je chráněno pomocí pojistky F1 s maximálním proudem 1A. Vstupní napájení v režimu device je chráněno pojistkou F2 s maximálním proudem 500mA. Obě pojistky jsou vratné a umožňují tedy opětovnou funkci. Mikropočítač umožňuje detekci režimů na základě hodnoty signálu, tato funkce je umožněna pomocí přepínače SW2 připojeného přes propojku JP4.

Datové vodiče konektorů A a mini B jsou přímo propojeny, proto je možné využívat pouze jeden z režimů. Pokud je periferní deska připojena k počítači a současně je připojeno zařízení, může dojít k poničení navrhovaného systému, ale i připojených zařízení včetně počítače.



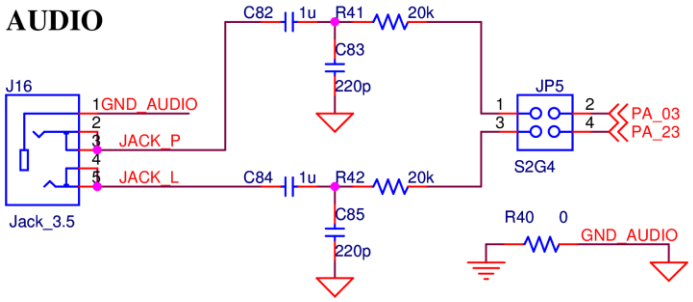
Obrázek 5.12: Zapojení režimu USB host



Obrázek 5.13: Zapojení režimu USB device

5.1.10. Audio výstup

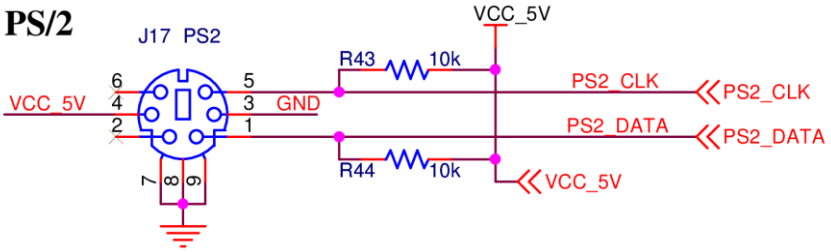
Mikropočítač umožňuje generování audio výstupu pomocí modulu ABDAC, jehož výstupem je stereo signál. Plošný spoj je tedy vybaven konektorem Jack 3,5mm, pasivním filtrem a kondenzátorem oddělujícím stejnosměrné napětí. Vývody jsou opět připojeny přes propojky JP5. Vzhledem k vlastnostem výstupního signálu je nutné připojit zesilovač signálu.



Obrázek 5.14: Zapojení audio výstupu

5.1.11. Klávesnice PS/2

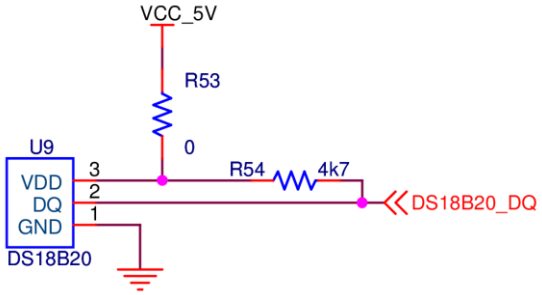
Rozhraní PS/2 využívá 5V napájení a dva kontrolní vodiče, jeden pro hodinový signál a druhý pro data, tyto vodiče vyžadují 10kΩ pull-up rezistory. Ochranu vývodů mikropočítače, jenž pracuje na 3,3V, před průrazem napětím 5V lze vynechat vzhledem k podpoře vstupního napětí až 5V na vývodech čipu.



Obrázek 5.15: Zapojení klávesnice PS/2

5.1.12. Teploměr DS18B20

Čip DS18B20 slouží k měření teploty v rozsahu -55 až 125 °C. Komunikace je založena na principu sběrnice a využívá pouze jeden datový vodič s 4,7kΩ pull-up rezistorem. Vzhledem k možnosti připojit více měřících zařízení, je na desce osazen pouze konektor.

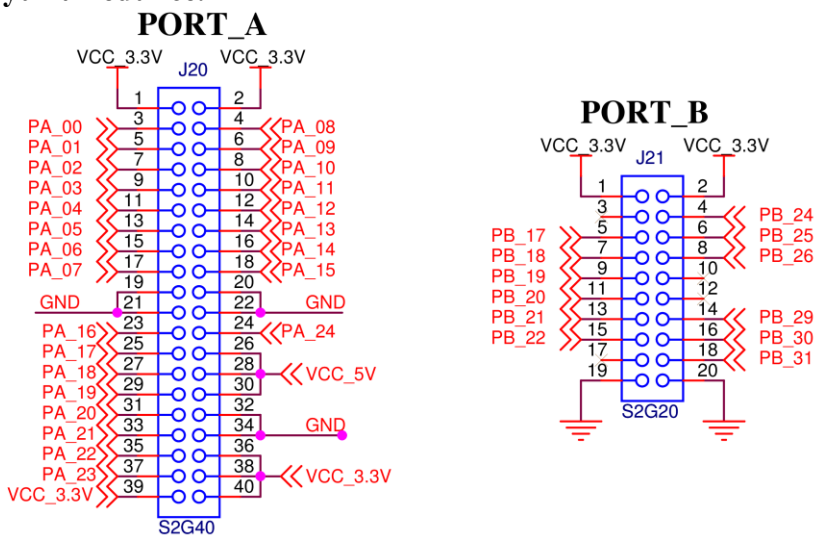


Obrázek 5.16: Zapojení teploměru DS18B20

5.1.13. Rozšiřující konektory

Všechny nezapojené vývody či volitelné vývody mikropočítače jsou přivedeny na rozšiřující konektory. Uvnitř čipu jsou tyto vývody rozděleny na tři porty – port A, port B a port X. Po propojení všech částí systému zbylo 25 vývodů pro port A, 12 vývodů pro port B a port X byl využit celý.
Rozšiřující konektory by měly umožnit připojení dalších modulů a měly by kromě datových signálů obsahovat také přívody napájení a zemnění. Vzhledem k univerzálnosti systému jsou zvoleny konektory pro ploché kabely s rozměry 2 x 20 pro port A, resp. 2 x 10 pro port B. Konektory jsou vzhledem k snadnějšímu přístupu k vývodům tvořeny lámací lištou.

40pinový konektor nepodporuje 80žilové Ultra ATA kabely, které sloužily k připojení pevných disků uvnitř počítače, kvůli vnitřnímu propojení zemnících vodičů. Tato varianta by velmi komplikovala vyvedení 8 sousedních datových bitů portu bez využití redukce.



Obrázek 5.17: Zapojení rozšiřujících konektorů

5.2. Výroba plošného spoje

Deska plošného spoje byla navržena v prostředí OrCAD 16.3. Šířka vodiče a minimální izolační vzdálenosti mezi spoji je 8 milů (~0,2 mm) a odpovídá třídě přesnosti 5.

Projekt a knihovny použité pro návrh plošného spoje jsou uloženy na přiloženém CD. Součástí knihoven jsou schematické značky, pady a pouzdra součástek. Součástí projektu je schéma, layout plošného spoje a vygenerované soubory pro výrobu, které zahrnují měděné plochy (bot.gbr a top.gbr), nepájivé masky (smb.gbr a smt.gbr), potisk (plb.gbr a plt.gbr) a souřadnicovou vrtačku (pth.exc). Výrobu desky provedla firma PragoBoard[17].

6. SW realizace

V této kapitole jsou popsány jednotlivé části firmwaru pro vývojovou desku. Jsou rozděleny na tři základní části. V první části jsou knihovny pro mikroprocesor AT32UC3A0, které ovládají jednotlivé integrované bloky uvnitř čipu. Druhá část se věnuje knihovnám ostatních modulů systému použitých na vývojové desce (např. komunikaci a ovládání externích bloků – SDRAM paměť, obvod RTC atd.). Poslední část nakonec spojuje obě předešlé knihovny do malých ukázkových demo aplikací.

6.1. Vývojové prostředí a programovací jazyk

Pro implementaci a ladění firmwaru se nejvíce hodí program Atmel Studio 6, který je vydáván společností Atmel a je po registraci volně dostupný. Tento program zahrnuje podporu pro 8 i 32bitové AVR a procesory ARM.

Firmware lze psát v asembleru nebo v jazyku C či C++. Asembler se hodí spíše pro malé systémy či části systému s požadavkem na velkou optimalizaci. Pro větší systémy je vhodné využít vyšší programovací jazyky, které nevyžadují znalost architektury mikroprocesoru, ale jejich překlad nemusí být vždy optimální.

Program v sobě dále zahrnuje knihovnu ASF neboli Atmel Software Framework. Tato knihovna zahrnuje ukázky softwaru pro jednotlivé procesory a umožňuje usnadnit a urychlit návrh systému.

Největší výhodu však program poskytuje při debugování kódu, kdy umožňuje zobrazit interní registry procesoru a měnit jejich hodnoty. Tuto vlastnost lze využít, jak při použití interního simulátoru, tak v případě JTAG debugeru, který navíc umožní využít všechny periferie v reálném čase.

6.2. Architektura mikroprocesoru

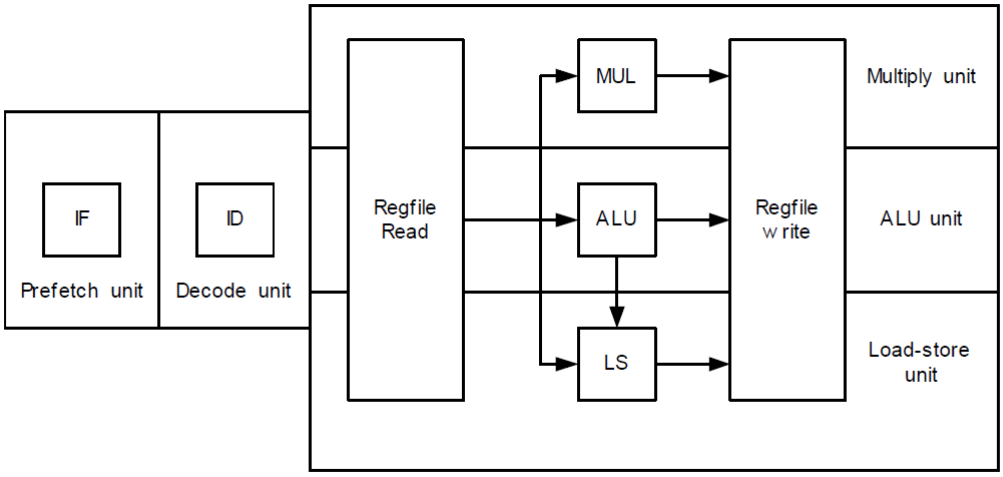
Pro psaní firmwaru je vhodné znát alespoň trochu architekturu procesoru, podrobnosti lze nalézt v dokumentech AVR32 Architecture Manual [23] a AVR32UC Technical Reference Manual [24].

AVR32 je 32bitový RISC mikroprocesor, navržený s důrazem na nízkou spotřebu, cenovou dostupnost, vysokou hustotu kódu a vysokou datovou propustnost.

Instrukční sada obsahuje více formátů instrukcí. Lze je dělit na krátké (16bitové), jež využívají pouze dva operandy či menší konstanty, a na dlouhé (32bitové), které umožňují využití tří operandů a zadání větších konstant. Další různé formáty jsou použity např. pro instrukce načítání a ukládání dat do paměti, které umožňují minimalizovat velikost kódu a zkrátit dobu běhu programu.

Registrové pole se skládá z šestnácti 32bitových registrů včetně Program Counteru, Link registru a Stack Pointeru. Registr R12 je navržen hlavně pro uchování návratových hodnot funkcí.

Vlastní jádro procesoru, které lze vidět na obrázku 6.1, se skládá ze třístupňové pipeline – načtení instrukce (IF), dekódování instrukce (ID) a vykonání instrukce (EX). Poslední část je rozdělena na tři paralelní sekce – aritmetické a logické funkce (ALU), násobení (MUL) a přístup k paměti (LS). Dělení není hardwarově podporováno a jeho softwarové řešení vyžaduje několik taktů.



Obrázek 6.1: Jádro procesoru

6.3. Knihovny AT32UC3A0

Tato část se věnuje nastavení jednotlivých bloků uvnitř mikroprocesoru, které jsou nutné pro použití s vývojovou deskou navrženou během této diplomové práce. Při psaní bylo čerpáno hlavně z manuálu k mikroprocesoru [7], kde jsou zmíněny všechny možnosti nastavení.

Vzhledem k velkému množství funkcí sloužících k nastavení a čtení registrů jednotlivých bloků jsou zmíněny pouze ty nejdůležitější. K ovládání lze využít taktéž soubory, které jsou součástí knihovny ASF.

6.3.1. **Power Manager (PM)**

Blok slouží k ovládání hodinových signálů uvnitř mikroprocesoru. Hodinové signály jsou rozděleny na obecné (GCLK0 – GCLK3, USBB a ABDAC) a synchronní (CPU, HSB, PBA a PBB). PM dále umožňuje maskování jednotlivých výstupů a změnu jejich hodinové frekvence. Inicializace tohoto bloku by měla být provedena na začátku každého programu.

void pm_main_clock_source(uint8_t source)
Vybírá vstup pro hlavní hodiny mikroprocesoru, které lze vybrat z vnitřního RC oscilátoru (115kHz), externího krystalu OSC0 (až 16 MHz) nebo výstupu bloku PLL0 (až 240 MHz).

void pm_cpu_clock_select(uint8_t divider)
void pm_pba_clock_select(uint8_t divider)
void pm_pbb_clock_select(uint8_t divider)
Nastavení hodinových vstupů pro procesor a interní periferní sběrnice. Parametr divider určuje dělící poměr, výsledné hodiny odpovídají $CLK_{IN} / 2^{DIVIDER}$.

void pm_oscillator0(uint8_t enable, uint32_t mode_value)
void pm_oscillator32(uint8_t enable, uint32_t mode_value)
Nastavuje vlastnosti oscilátoru, parametr mode_value obsahuje informaci o počtu taktů potřebným k ustálení hodinového signálu společně s jeho frekvencí.

void pm_pll0(uint8_t enable, uint8_t oscilator, uint8_t option, uint8_t multiply, uint8_t divider, uint8_t startup)
void pm_pll1(uint8_t enable, uint8_t oscilator, uint8_t option, uint8_t multiply, uint8_t divider, uint8_t startup)
Funkce nastavuje vnitřní PLL dle parametr. Vstupem může být OSC0 nebo OSC1. Parametr option určuje, zda je vnitřní frekvence PLL nižší než 160 MHz (bit 0) a zda má být výstupní frekvence poloviční než vnitřní (bit 1). Bit 2 zakazuje režim s větší šířkou pásma, který zkracuje dobu potřebnou k dosažení požadované frekvence. Parametr startup umožňuje zadat časový interval, během kterého není zaručena stabilita hodinového signálu. Výstup obvodu je během této doby odpojen a nelze jej používat jako vstupní hodinový signál pro bloky mikropočítače. Parametry multiply a divider určují vnitřní frekvenci PLL, pokud je druhý parametr nulový je vypočtena dle prvního vzorce (a) jinak dle druhého (b):

a) $f_{VCO} = 2 * (multiply + 1) * f_{OSC}$
b) $f_{VCO} = (multiply + 1) / divider * f_{OSC}$

void pm_generic_clock(uint8_t clk_offset, uint8_t enable, uint8_t source, uint8_t divider)
Slouží k nastavení obecných hodin (GCLK0 – GCLK3, USBB a ABDAC) dle parametrů. Parametr clk_offset vybírá jednotlivé bloky obecných hodin, source určuje jejich vstup (osc0, osc1, pll0 či pll1), poslední parametr určuje dělící poměr, kdy výsledná frekvence je $CLK_{IN} / 2^{DIVIDER}$.

6.3.2. Real Time Counter (RTC)

void rtc_control_set(uint8_t enable, uint8_t prescaler, uint32_t mode)
Provádí konfiguraci bloku. Parametr prescaler umožňuje nastavit požadovaný dělící poměr. Parametr mode vybírá vstupní signál a dále určuje, zda má obvod probouzet procesor z režimu spánku.

void rtc_value_set(uint32_t value)
uint32_t rtc_value_get()
Umožňují nastavit a zjistit aktuální hodnotu čítače RTC.

void rtc_top_set(uint32_t value)
uint32_t rtc_top_get()
Nastavení a čtení hodnoty top pro čítač RTC, při této hodnotě dojde k vyvolání přerušení a čítač se resetuje na hodnotu 0.

6.3.3. Interrupt Controller (INTC)

Blok seskupuje signály žádostí o přerušení od jednotlivých periferií. Signály jsou spojeny do skupin, podpora až 64 bloků, v jedné skupině může být až 32 signálů. Žádosti o přerušení společně s vektorem obsluhy jsou následně dle priority (čtyři úrovně) doručeny CPU. V tabulce 6.1 je seznam nejdůležitější skupin a signálů použitých v této diplomové práci.

Skupina	Signál	Popis
0	0	AVR32 UC CPU – SYSBLOCK COMPARE
1	8	RTC
1	9	Power Manager
2	0 – 13	GPIO
3	0 – 14	Peripheral DMA
5 - 7	0	USART 0 – USART 2
9 - 10	0	SPI 0 – SPI 1
11	0	TWI
12	0	PWM
14	0 - 2	Timer Counter
15	0	Analog to Digital Converter
16	0	Ethernet MAC
17	0	USB 2.0 OTG
18	0	SDRAM Controller
19	0	Audio Bitstream DAC

Tabulka 6.1: Seznam skupin a signálů bloku INTC

void intc_interrupt_set (uint8_t group, uint8_t priority, uint16_t auto_vector)
Funkce slouží k nastavení priority a vektoru obsluhy přerušení pro určitou skupinu.

uint32_t intc_request_get(uint8_t group)
Vrací jednotlivé žádosti o přerušení pro požadovanou skupinu.

uint8_t intc_cause_get(uint8_t priority)
Umožňuje zjistit, která skupina vyvolala přerušení o zadané prioritě.

6.3.4. Peripheral DMA Controller (PDCA)

Blok se skládá z 15 kanálů, z nichž každý realizuje přesuny dat mezi pamětí (uvnitř nebo mimo čip) a periferiemi (např. UART, SPI, TWI) bez využití procesoru a tím zvyšuje výkon aplikace. Podporovány jsou 8, 16 i 32 bitová slova.

Modul	ID – příjem dat	ID – vysílání dat
ADC	0	–
SSC	1	9
USART 0	2	10
USART 1	3	11
USART 2	4	12
TWI	6	14
SPI 0	7	15
SPI 1	8	16
ABDAC	–	17

Tabulka 6.2: Seznam hlavních periferií a jejich ID

void pdca_transfer_set(uint8_t channel, uint32_t address, uint16_t count)
void pdca_reload_transfer_set(…)
Nastavuje adresu a velikost bufferů pro jednotlivé kanály. Druhá funkce nastavuje záložní buffer, který je použit po vyprázdnění, resp. naplnění prvního.

void pdca_channel_set(uint8_t channel, uint8_t peripheral_id, uint8_t size)
Základní nastavení kanálu – šířka datového slova a ID dle periferie a směru.

void pdca_transfer_enable(uint8_t channel)
void pdca_transfer_disable(uint8_t channel)
Funkce pro povolení a zakázání přenosu pro určitý kanál.

6.3.5. General-Purpose Input/Output Controller (GPIO)

Blok ovládající jednotlivé vývody mikropočítače. Vývody lze nastavit jako vstup či výstup nebo přiřadit jedné ze tří funkcí periferií. Zapojení jednotlivých periferií je součástí přílohy této práce. Blok dále umožňuje vyvolat přerušení při každé změně, případně pouze na vzestupnou či sestupnou hranu signálu. Vstupní signál může využít filtr pro odstranění zákmitů kratších než jeden hodinový takt. Samozřejmostí jsou konfigurovatelné interní pull-up rezistory.

Funkce pro ovládání tohoto bloku jsou ve třech variantách lišících se přístupem a hlavním vstupním parametrem:

- gpio funkce – pouze číslo gpio vývodu (0 – 109)
- pin funkce – číslo portu a určitého pinu
- port funkce – umožňují přístup k registru celého portu

void gpio_function_set(uint8_t gpio, uint8_t function)
Přiřadí vývodu periferní funkci dle parametru (A – 0, B – 1, C – 2).

void gpio_control_set(uint8_t gpio)
Slouží k vypnutí periferní funkce a zapnutí manuálního ovládání.

void gpio_input_set(uint8_t gpio)
void gpio_output_set(uint8_t gpio)
Přepínání mezi vstupem a výstupem.

void gpio_interrupt_set(uint8_t gpio, uint8_t mode)
Funkce ovládá přerušení při změně signálu (není povoleno – 0, vzestupná hrana – 1, sestupná hrana – 2, každá změna – 3).

void gpio_glitch_filter_set(uint8_t gpio, uint8_t enable)
Umožňuje odstranit zákmity kratší než jeden hodinový takt.

void gpio_pullup_set(uint8_t gpio, uint8_t enable)
Ovládání interního pull-up rezistoru.

void gpio_value_set(uint8_t gpio, uint8_t value)
Nastaví požadovanou hodnotu na výstup.

uint8_t gpio_value_set(uint8_t gpio)
Funkce přečte a vrátí hodnotu na vstupu.

void gpio_interrupt_flag_clear(uint8_t gpio)
Smaže příznak přerušení pro daný vývod.

6.3.6. Serial Peripheral Interface (SPI)

Blok může pracovat v režimech master i slave. V režimu master umožňuje komunikovat se čtyřmi zařízeními, ale počet lze zvýšit s využitím externího dekodéru až na patnáct. Velikost dat může být 8 až 16 bitů. Dále podporuje čtyři režimy pro nastavení polarity a fáze hodinového signálu. Nastavení je možné provést pro jednotlivé čipy resp. skupiny čipů v případě externího dekodéru.

Mikropočítač obsahuje dva stejné bloky, které jsou na sobě nezávislé. Kvůli tomu funkce obsahují parametr „uint16_t spi“, kterým se vybírají jednotlivé bloky. Parametr nabývá hodnot SPI_0 nebo SPI_1.

Pro zvýšení výkonu lze využít DMA přenos.

void spi_enable(uint16_t spi)
void spi_disable(uint16_t spi)
Funkce povoluje resp. zakazuje blok SPI.

void spi_master_mode(uint16_t spi, uint8_t mode, uint8_t delay_cs)
Nastavuje režim SPI na master. Parametr mode umožňuje nastavit režim adresace periferie (CS), vstupní hodinový signál, detekci chyb či zpětnou lokální smyčku pro testování. Adresace může být pevná (nastavení probíhá pomocí mode registru – MR) nebo proměnná (hodnota v registru s odesílanými daty – TDR).

void spi_slave_mode(uint16_t spi)
Funkce nastaví režim SPI na slave.

uint32_t spi_data_read(uint16_t spi)
Funkce čeká, dokud není dokončen přenos (bit RDRF ve Status registru). Následně vrací přijatá data [15:0] a adresu periferie [19:16].

void spi_data_write(uint16_t spi, uint16_t data, uint8_t cs, uint8_t last)
Funkce slouží k odeslání dat pro režim proměnné adresace. Pokud probíhá předchozí přenos, čeká nejdříve na jeho dokončení (bit TXEMPTY ve Status registru). Parametr CS určuje periferii a parametr last určuje, zda se jedná o poslední zapisované slovo a zda může být po ukončení zápisu uvolněn signál vybírající čip.

Funkce pro režim s pevnou adresací periferie:
void spi_master_set_fix_cs(uint16_t spi, uint8_t chip_select)
Nastavuje adresu periferie, probíhá přes Mode registr – MR.

void spi_last_transfer_fix_cs(uint16_t spi)
Určuje, zda je přenášené slovo poslední a může být uvolněn signál CS.

void spi_data_write_fix_cs(uint16_t spi, uint16_t data)
Funkce slouží k odeslání dat. Před jejich zápisem čeká, dokud není registr prázdný (bit TXEMPTY ve Status registru).

uint32_t spi_status_get(uint16_t spi)
Vrací hodnotu status registru obsahující informace o přenosu – např. dokončení, výskyt chyb, ukončení DMA přenosu.

void spi_cs_delay_set(uint16_t spi, uint8_t cs, uint8_t dlybct, uint_8t dlybs)
Nastavuje hodnoty zpoždění pro jednotlivé periferie. Parametr dlybct určuje mezeru mezi dvěma přenosy pro stejnou periferii. Parametr dlybs určuje mezeru mezi výběrem periferie (nastavení CS signálu) a první změnou hodinového signálu.

void spi_cs_set(uint16_t spi, uint8_t cs, uint8_t mode, uint8_t size, uint8_t baud)
Nastavení komunikace s periferií. Parametr mode určuje chování hodinového signálu – klidovou úroveň a fázi (hranu pro změnu a záznam dat), a chování signálu CS po dokončení přenosu dat. Parametr size určuje velikost přenášených dat (8 – 16 bitů). Baud určuje frekvenci hodinového signálu v závislosti na vstupních hodinách.

6.3.7. Two Wire Interface (TWI)

Blok umožňuje komunikaci se zařízeními po sběrnici I^2C s rychlostí až 400kHz. Podporuje režimy master, multi–master a slave (včetně interní adresace 1 – 3 bajty). V režimu master umožňuje využít DMA pro přenos dat.

void twi_master_mode_enable()
void twi_master_mode_disable()
void twi_slave_mode_enable()
void twi_slave_mode_disable()
Funkce povolují (lze použít i k přepínání) a zakazují režim bloku TWI.

void twi_sw_reset()
Provede reset bloku.

void twi_send_start()
Vyšle na sběrnici „start bit“ – sestupná hrana dat při klidové úrovni hodin.

void twi_send_stop()
Vyšle na sběrnici „stop bit“ – vzestupná hrana dat při klidové úrovni hodin.

void twi_slave_addr_set(uint8_t twi_addr)
Funkce slouží k nastavení adresy zařízení v režimu slave. Hodnota musí být nastavena před povolením režimu, v ostatních případech bude ignorována.

void twi_master_mode_set(uint8_t dev_addr, uint8_t iaddr_size, uint8_t direct)
Funkce slouží k nastavení cílové adresy, velikosti interní adresy a směru komunikace (čtení či zápis) v režimu master.

void twi_internal_addr_set(uint32_t twi_addr_i)
Funkce nastaví vnitřní adresu cílového zařízení, její velikost je nastavena funkcí twi_master_mode_set(…).

void twi_clock_generator_set(clk_low_div, clk_high_div, clk_div)
Funkce slouží k nastavení minimální a maximální periody hodinového signálu. Výpočet probíhá dle vzorce:

- $T_{low} = ((clk\_low\_div * 2^{clk\_div}) + 4) * T_{CLK}$.

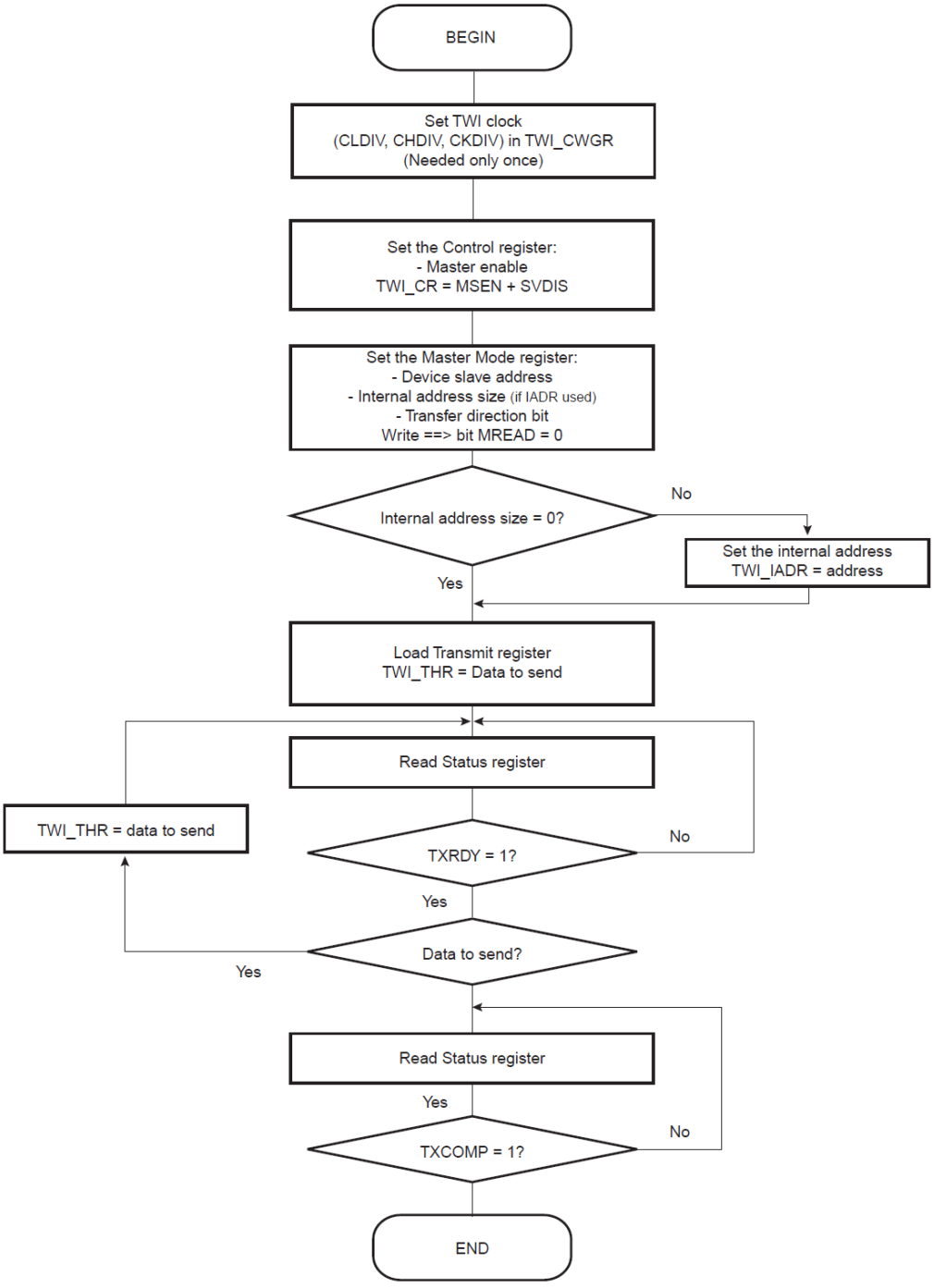
uint32_t twi_status_get()
Vrací hodnotu stavového registru, který obsahuje informace o dokončení přenosu, chybách či ztrátě arbitrace v režimu multi–master.

uint8_t twi_receive_data()
Funkce vrací přijatá data. Pokud je registr prázdný, čeká se na dokončení přenosu (bit RXRDY ve Status registru).

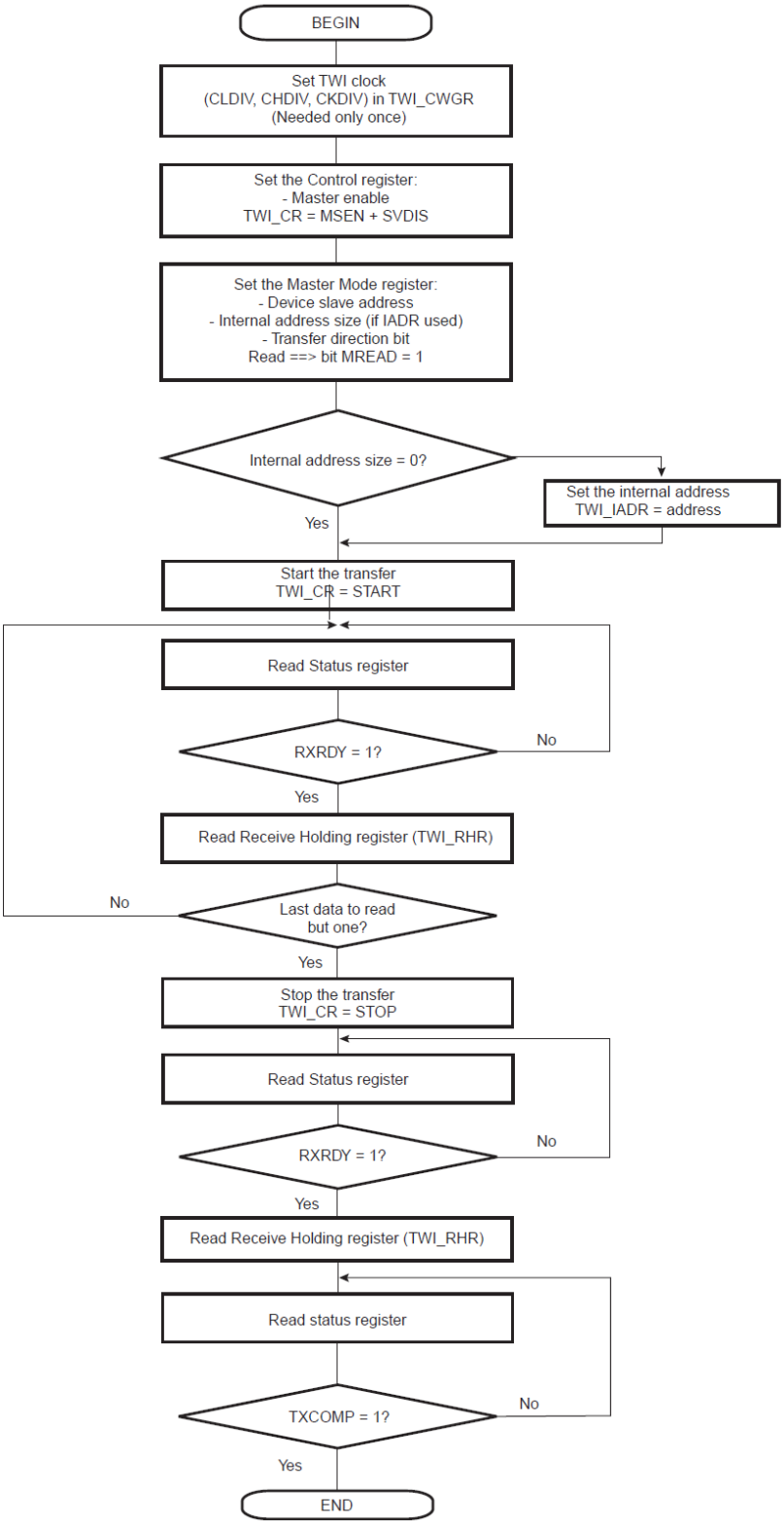
void twi_send_data(uint8_t data)
Funkce odesílá data předaná parametrem. V případě, že probíhá předchozí přenos, čeká na jeho dokončení (bit TXRDY ve Status registru).

void twi_write_data(uint8_t twi_addr, uint32_t iaddr, uint8_t iaddr_size, const uint8_t* data, uint8_t size)
Funkce slouží k provedení zápisu bloku dat do zařízení, jehož adresa je určena parametrem twi_addr. Parametry iaddr a iaddr_size určují adresu uvnitř zařízení a její velikost. Data k přenosu jsou určena ukazatelem a velikostí. Přenos probíhá dle obr. 6.2.

void twi_read_data(uint8_t twi_addr, uint32_t iaddr, uint8_t iaddr_size, const uint8_t* data, uint8_t size)
Funkce slouží ke čtení dat ze zařízení s adresou stejnou jako parametr twi_addr. Vnitřní adresa o velikosti iaddr_size je určena parametrem iaddr. Data o velikosti size jsou zapisovaná do paměti od adresy určené parametrem data. Přenos probíhá dle obr. 6.3.



Obrázek 6.2: Zápis dat přes sběrnici TWI (I^2C), dle dokumentace [7]



Obrázek 6.3: Čtení dat přes sběrnici TWI (I^2C) , dle dokumentace [7]

6.3.8. Universal Synchronous / Asynchronous Receiver / Transmitter (USART)

Tento blok se v mikropočítači vyskytuje ve čtyřech provedeních. Blok USART 1 podporuje režimy SPI, RS232, RS485 s ovládáním směru, IrDA modulaci a demodulaci, ISO7816 (komunikace s čipovými kartami) či Manchester kódování. Bloky USART 0, USART 2 a USART 3 podporují pouze režimy RS232 a SPI.

Funkce pro ovládání bloku jsou vzhledem k potřebám této práce navrženy pouze pro režimy RS232 a RS485. Výpočetní výkon aplikace lze zvýšit využitím DMA kanálů. Společný parametr „uint16_t uart“ všech funkcí specifikuje blok, se kterým probíhá komunikace.

void uart_transmitter_disable(uint16_t uart)
void uart_transmitter_enable(uint16_t uart)
void uart_receiver_disable(uint16_t uart);
void uart_receiver_enable(uint16_t uart);
Funkce slouží k ovládání vysílače a přijímače pro specifický blok.

void uart_transmitter_reset(uint16_t uart)
void uart_receiver_reset(uint16_t uart)
Pomocí těchto funkcí lze vyvolat reset přijímače či vysílače určeného bloku.

void uart_control_set(uint16_t uart, uint32_t mode)
Funkce umožňuje ovládání bloku. Povolené hodnoty parametru mode lze najít v dokumentaci [7] či souboru uart.h v sekci Constants / Control Registr.

void uart_mode_set(uint16_t uart, uint32_t mode)
Slouží k nastavení režimu a chování bloku odpovídajícímu parametru mode. Umožňuje zapínat testovací režimy, nastavit počet stop bitů, synchronní či asynchronní režim, druh parity, šířku datového slova, hodinový vstup či vybrat jeden z režimů (např. RS232, RS485, IrDA, SPI).

uint32_t uart_status_get(uint16_t uart)
Funkce vrací hodnotu Status registru specifikovaného bloku.

uint8_t uart_read_word(uint16_t UART)
Funkce vrací přijatá data. Pokud registr neobsahuje platná data, čeká se ve funkci, dokud nebudou platná (bit RXRDY ve Status registru). V případě, že by byl povolen režim „Multidrop“ či 9bitová komunikace, je třeba rozšířit velikost návratové hodnoty na uint16_t. V režimu „Multidrop“ pak bit 15 (RXSYNH) rozlišuje, zda se jedná o data (0) či příkaz (1), tento bit je přenášen místo parity.

void uart_send_word(uint16_t uart, uint8_t data)
Slouží k odeslání dat předaných parametrem. V případě, že registr obsahuje ještě neodeslaná data, čeká se na jejich odeslání (bit TXRDY ve Status registru). Pokud bude povolena 9bitová komunikace či režim „Multidrop“ je nutné zvětšit velikost parametru

data na uint16_t. V režimu „Multidrop“ pak bit 15 (TXSYNH) rozlišuje, zda se jedná o data (0) či příkaz (1), tento bit je přenášen místo parity.

void uart_baudrate_set(uint16_t uart, uint16_t clock_div)
Slouží k nastavení frekvence generátoru hodin přijímače i vysílače. Vstupní hodinový signál závisí na hodnotě v Mode registru. Výsledná hodnota je dle vzorce:

- Baudrate = (CLK / (8 * (2 – Over) * CD).

void uart_send_hex4(uint16_t uart, uint8_t value)
void uart_send_hex8(uint16_t uart, uint8_t value)
void uart_send_hex16(uint16_t uart, uint16_t value)
void uart_send_hex32(uint16_t uart, uint32_t value)
Funkce slouží k odeslání hexadecimální hodnoty předané parametrem value. Počet bitů vstupu je určen v názvu funkce.

void uart_send_unumber8(uint16_t uart, uint8_t value)
void uart_send_unumber16(uint16_t uart, uint16_t value)
void uart_send_unumber32(uint16_t uart, uint32_t value)
Funkce slouží k odeslání čísla bez znaménka v dekadické podobě. Hodnota v názvu funkce určuje počet bitů a maximální rozsah čísla (0 – 255, 0 – 65535, …).

void uart_send_string(uint16_t uart, char *text)
Funkce odešle textový řetězec, který musí být zakončen znakem \0. Počáteční adresa řetězce je předaná parametrem text.

6.3.9. SDRAM Controller (SDRAMC)

Řadič umožňuje rozšířit kapacitu vnitřní paměti mikropočítače o jeden externí čip SDRAM s 16bitovou datovou sběrnicí a se dvěma nebo čtyřmi interními bankami. Před použitím řadiče je nutné provést inicializaci:

a) Nastavit konfigurační registr (časování a parametry paměti)
b) Pro mobilní SDRAM nastavit registr Low Power
c) Nastavit typ paměti v registru Memory Device
d) Musí být proveden příkaz NOP (1) a zápis na určitou adresu
e) Vyčkat 200 μs
f) Provést příkaz All Banks Precharge (2) a provést zápis do paměti
g) 8x provést příkaz Auto–refresh (CBR, 4) a zápis do paměti
h) Mode Registr Set (MRS, 3) a provést zápis na adresu, aby BA[1:0] bylo 0
i) Pro inicializaci mobilních SDRAM, provést Extended Mode Registr set (EMRS, 5) a vykonat zápis na adresu, aby BA[1:0] bylo 0b01 či 0b10
j) Aplikace musí zapnout normální režim (0) a provést zápis do paměti
k) Zapsat hodnotu do registru Refresh Timer

void sdramc_command_set(uint8_t command)
Funkce nastaví příkaz pro paměť, který je vykonán při přístupu k paměti. Příkazy jsou nutné pro inicializaci a jejich hodnoty jsou NOP (1), All Banks Precharge (2), Load

Mode Register (3), Auto Refresh (4), Extended Load Mode Register (5), Deep Power-down Mode (6).

void sdramc_refresh_timer_set(uint16_t timer)
Slouží k nastavení 12bitové hodnoty čítače, který v zadaném intervalu vydává příkazy pro obnovu dat. Hodnota závisí na frekvenci hodinového signálu, obnovovací frekvenci a době potřebné k obnovení jednoho řádku paměti.

void sdramc_config_set(uint32_t config)
Funkce slouží k nastavení konfigurace paměti – počet sloupců, řádků a bank paměti, CAS latency a jednotlivá zpoždění.

6.3.10. Ethernet MAC (MACB)

Blok implementuje 10/100 Ethernet MAC, který odpovídá standardu IEEE 802.3 a podporuje rozhraní MII či RMII pro připojení fyzické vrstvy včetně rozhraní MDIO pro její konfiguraci. Registry lze dělit na kontrolní, umožňující nastavení bloku, a statistické, které udržují čítače událostí. Statistické registry umožňují zjistit počet přijatých a odeslaných paketů či např. počet chyb při přenosu, vzhledem k jejich počtu nejsou podporovány funkcemi, jejich seznam lze nalézt v dokumentaci v sekci MACB či v souboru „macb.h“.

Komunikace s pamětí probíhá pomocí DMA. Data v paměti jsou ukládány do bufferů, jejichž adresy a kontrolní informace jsou uloženy v deskriptorech. Buffer pro přijímání dat je veliký 128 bajtů a pro odesílání může být 0 – 2047 bajtů.

void macb_control_set(uint32_t value)
Slouží k základnímu ovládání bloku, resp. ovládání vysílače, přijímače či rozhraní MDIO, nulování statistických registrů atd.

void macb_transmission_start()
Funkce zahájí přenos ethernetových rámců.

void macb_transmission_halt()
Funkce slouží k pozastavení vysílání ethernetových rámců, které provede ihned po dokončení probíhajícího přenosu.

void macb_configuration_set(uint32_t value)
Funkce umožňuje základní nastavení bloku – rychlost 10/100 Mbit/s, povolení broadcastů, velkých a Jumbo rámců, hodiny pro MDIO atd.

void macb_receive_buffer_pointer_set(uint32_t *pointer)
void macb_transmit_buffer_pointer_set(uint32_t *pointer)
Funkce nastavuje adresu seznamu deskriptorů bufferů pro přijímání, resp. odesílání. Tyto funkce je nutné volat před povolením přijímače resp. vysílače.

uint32_t *macb_receive_buffer_pointer_get()
uint32_t *macb_transmit_buffer_pointer_get()
Funkce vrací adresu aktuálně používaných bufferů.

void macb_phy_write(uint8_t phy_addr, uint8_t reg_addr, uint16_t data)
Provádí zápis dat do registru fyzické vrstvy. Parametry phy_addr a reg_addr určují adresu fyzické vrstvy resp. registru. Parametr data určuje 16bitové slovo, které má být na danou adresu zapsáno. Funkce nečeká na dokončení zápisu.

uint16_t macb_phy_read(uint8_t phy_addr, uint8_t reg_addr)
Funkce slouží ke čtení dat z registru fyzické vrstvy. Parametry phy_addr a reg_addr určují adresu fyzické vrstvy a registru. Přečtená data jsou předána přes návratovou hodnotu.

6.3.11. USB OTG (USBB)

Blok slouží k realizaci komunikace přes sběrnici USB v režimu host či device. Odpovídá standardu 2.0 s podporou rychlostí 1,5 a 12 Mb/s. Obsahuje interní pull-up rezistory (pro určení rychlosti v režimu USB device) a pull-down rezistory pro režim USB host.

Vzhledem k velkému počtu registrů a složitosti nastavení byl tento blok testován pouze pomocí ASF Frameworku, který je součástí vývojového prostředí AtmelStudio.

6.3.12. Timer / Counter (TC)

Blok obsahuje tři 16bitové čítače, které mohou sloužit např. k měření frekvence či intervalu, počítání a generování pulzů a PWM. Čítače jsou nezávislé a mohou využít tři externí nebo pět interních hodinových signálů. Režimy čítače a jejich nastavení jsou v dokumentaci v sekci Timer / Counter (TC) [7].

Parametr funkcí uint8_t channel slouží k výběru čítače a jeho hodnota musí odpovídat definovaným hodnotám v souboru „tc.h“ (např. TC_CHANNEL_0).

void tc_block_sync()
Funkce nastavuje SYNC signál, který generuje softwarový trigger pro všechny kanály současně.

void tc_channel_enable(uint8_t channel)
void tc_channel_disable(uint8_t channel)
Slouží k povolení či zakázání čítače, který je vybrán parametrem.

void tc_channel_reset(uint8_t channel)
Funkce resetuje hodnotu čítače a vyvolá jeho spuštění.

void tc_channel_mode_set(uint8_t channel, uint32_t mode)
Funkce slouží k nastavení režimu čítače – vybírá hodinový vstup, hranu, se kterou dojde ke změně hodnoty, možnosti resetování hodnoty a zastavení čítače, generování výstupu atd. Jednotlivé režimy a nastavení jsou popsány v dokumentaci [7] viz. TC Channel Mode Register.

uint16_t tc_value_get(uint8_t channel)
Funkce vrací aktuální hodnotu čítače.

void tc_regA_set(uint8_t channel, uint16_t value)
uint16_t tc_regA_get(uint8_t channel)
Funkce slouží k nastavení, resp. čtení hodnoty registru RA (stejné funkce jsou i pro registry RB a RC). Význam registru závisí na režimu čítače, pokud není generován výstup (Mode registr – parametr WAVE = 0) jsou registry RA a RB pouze pro čtení.

uint32_t tc_status_get(uint8_t channel)
Funkce slouží ke čtení status registru čítače, umožňuje kontrolovat, zda došlo k přetečení čítače či nastavení externího triggeru nebo zda byla hodnota čítače stejná jako v registru RA, RB či RC. Při Status registru (SR) dojde k jeho nulování.

void tc_interrupt_enable(uint8_t channel, uint8_t mask)
void tc_interrupt_disable(uint8_t channel, uint8_t mask)
Funkce slouží k povolení či zakázání přerušení pro jednotlivé čítače dle zvolené masky. Proč došlo k přerušení lze zjistit přečtením Status registru.

6.3.13. Pulse Width Modulation Controller (PWM)

Blok umožňuje generování pulzně šířkové modulace. Obsahuje sedm nezávislých kanálů a generátor hodinových signálů s třinácti výstupy. Čítač „modulo n“ generuje jedenáct hodinových signálů, které slouží i jako vstup dvou lineárních děliček. Každý kanál obsahuje 20bitový čítač, registry pro periodu a střídu, výstupní komparátor a multiplexor hodinových signálů.

Funkce obsahují parametr uint8_t channel, který slouží k výběru kanálu. Registry pro střídu, periodu a hodnotu čítače obsahují pouze 20 platných bitů.

void pwm_clock_set(uint32_t value)
Funkce slouží k nastavení obou lineárních děliček hodinového vstupu.

void pwm_clka_set(uint8_t source, uint8_t div)
void pwm_clkb_set(uint8_t source, uint8_t div)
Funkce umožňují nastavení lineárních děliček hodinového signálu. Parametr source vybírá vstupní signál, parametr div určuje dělící poměr.

void pwm_channel_enable(uint8_t channel)
void pwm_channel_disable(uint8_t channel)
Funkce slouží k povolení či zakázání jednotlivých kanálů.

void pwm_channel_mode(uint8_t channel, uint16_t mode)
Nastavuje režim PWM kanálu – vstupní hodinový signál (výběr ze 13 možností), zarovnání a polaritu výstupního signálu.

void pwm_channel_duty_cycle_set(uint8_t channel, uint32_t value)
Funkce nastaví hodnotu střídy do registru CDTY.

void pwm_channel_period_set(uint8_t channel, uint32_t value)
Funkce nastaví hodnotu periody do registru CPRD. Výsledná perioda závisí na vstupním hodinovém signálu a zarovnání výstupního signálu. Pokud je zarovnaný na střed, tak odpovídá (CPRD * 2) / CLK, pokud je zarovnaný vlevo, pak odpovídá CPRD / CLK.

uint32_t pwm_channel_counter_get(uint8_t channel)
Funkce vrací aktuální hodnotu čítače.

6.3.14. Analog to Digital Converter (ADC)

Blok slouží k převodu analogového signálu na digitální. Podporuje až osm nezávislých analogových vstupů. Převod může být vyvolán softwarově nebo pomocí externího vstupu. Rozlišení převodu může být 8 či 10bitové. Výsledek lze vyčíst jednotlivě pro každý kanál (vstup). Pro ukládání výsledků do paměti lze použít DMA přenosy.

Funkce obsahují parametr uint8_t channel, který vybírá jednotlivé kanály.

void adc_conversion_start()
Funkce provede zápis do kontrolního registru, po kterém u povolených kanálů začne převod analogových hodnot na digitální.

void adc_reset()
Vyvolá reset celého bloku.

void adc_mode_time_set(uint8_t mode, uint8_t prescal, uint8_t startup, uint8_t shtim)
Funkce slouží k nastavení Mode registru. Parametr mode určuje režim bloku – rozlišení, HW či SW trigger, atd. Prescal nastavuje děličku hodinového signálu (a), startup slouží k nastavení času potřebného blokem ADC (b) a parametr shtim určuje minimální vzorkovací čas – Sample & Hold Time (c). Výsledné hodnoty jsou dle vzorců:

a) $CLK_{ADC} = CLK_{IN} / ((prescal + 1) * 2)$
b) $Startup\ Time = (startup + 1) * 8 / CLK_{ADC}$
c) $Sample\ \&\ Hold\ Time = (shtim + 1) / CLK_{ADC}$

void adc_channel_enable(uint8_t channel)
void adc_channel_disable(uint8_t channel)
Funkce slouží k povolení či zakázaní určitého kanálu převodníku.

uint32_t adc_state()
Funkce vrací hodnotu status registru, který obsahuje informace o dokončení převodu, stavu DMA přenosu, povolených kanálech atd.

uint16_t adc_read_data(uint8_t channel)
Funkce vrací poslední naměřená data pro daný kanál, pokud nejsou data platná, čeká se na dokončení převodu.

6.3.15. Audio Bitstream DAC (ABDAC)

Blok obsahuje dva kanály. Z nichž každý převádí 16bitovou digitální hodnotu na audio výstup pomocí sigma–delta DA převodníku. Výstup obou kanálů je nutné doplnit o filtr dolní propusti.

void abdac_data_channel_set(int16_t channel0, int16_t channel1);
Funkce slouží k zápisu dat pro oba kanály současně, data jsou předána pomocí parametrů funkce.

void abdac_enable()
void abdac_disable()
Funkce umožňují povolení resp. zakázání bloku.

6.4. Knihovny pro vývojovou desku AT32BOARD

V této podkapitole jsou rozepsány knihovny pro univerzální periferní desku AT32BOARD, která byla navržena během této diplomové práce. Funkce se věnují nastavení a komunikaci s periferiemi a čipy osazenými na plošném spoji.

6.4.1. Board

Knihovna slouží k nastavení jednotlivých periferií osazených na desce. Provede inicializaci paměti SDRAM a nastavení fyzické vrstvy Ethernetu včetně přiřazení periferních funkcí jednotlivým vývodům mikropočítače.

void board_init()
Provede základní konfiguraci desky

6.4.2. Common

Knihovna obsahuje funkce, které lze použít v různých aplikacích. Slouží např. k převodu mezi desítkovou a binární hodnotou či k převodu hodnoty na ASCII symbol v hexadecimální podobě (0 – 9, A – F).

hex4(uint8_t value) hex4_inline(value)
Funkce slouží k převodu 4bitové hodnoty na ASCII znak 0 – 9, A – F.

uint8_t convert_bin_to_bcd(uint8_t value)
Funkce převede číslo z binární hodnoty (předané parametrem) na číslo v desítkové soustavě (BCD). Číslo v desítkové soustavě je ve formátu: bity 0 – 3 zastupují jednotky a bity 4 – 7 zastupují desítky.

uint8_t convert_bcd_to_bin(uint8_t value)
Funkce převede číslo v desítkové soustavě (předané parametrem) na číslo v binární podobě. Formát čísla v desítkové soustavě (BCD) bity 4 – 7 pro desítky a bity 0 – 3 pro jednotky.

6.4.3. Delay (zpožďovací smyčky)

Tato knihovna se věnuje funkcím, které umožňují vkládat mezi části kódu zpoždění, jež je nutné k ovládání některých periferií. Vzhledem k architektuře procesoru je vhodné využít blok Timer / Counter (TC) uvnitř mikropočítače, který umožňuje minimalizovat zpoždění během přerušení a potřeby přepisování smyček při změně frekvence procesoru. Varianta s využitím bloku navíc umožňuje přepínání během vykonávání programu. Vzhledem ke zpoždění při volání funkcí přistupuje tato knihovna přímo k registrům bloku TC. **Kanál (TC_CHANNEL_2) nesmí být využit žádnou jinou aplikací, která by mohla změnit jeho hodnotu a narušit délku jednotlivých smyček.** Blok umožňuje použít statické a dynamické přepínání frekvence, které je definované parametrem DYNAMIC_DELAY.

void delay_init()
Funkce slouží k inicializaci čítače bloku TC – nastavení režimu, výběr hodinového signálu a povolení kanálu.

void delay_update(uint32_t freq_hz)
Funkce nastavuje proměnnou cycle_1us, která uchovává hodnotu čítače pro minimální zpoždění 1µs. Hodnota se využívá pouze při režimu dynamického přepínání frekvence.

delay_1us()
Funkce umožňuje vložení zpoždění 1µs. Jelikož procesor při frekvenci 12 MHz může vykonat maximálně devět instrukcí a volání funkce zabere několik taktů, využívá se metody „inline funkcí“, které jsou vloženy přímo do kódu.

delay_us(value)
Funkce slouží ke vložení zpoždění v řádech mikrosekund. Stejně jako předchozí funkce využívá metodu vkládání funkcí.

void delay_ms(uint32_t value)
void delay_s(uint32_t value)
Funkce vkládají zpoždění v milisekundách, resp. sekundách. Doba volání a návratu funkce je vzhledem k době trvání zpoždění zanedbatelná, proto není nutné využívat metodu vkládání funkcí.

6.4.4. LED diody

Knihovna slouží k ovládání čtyř uživatelský LED diod. Parametr funkcí musí odpovídat hodnotám definovaným v souboru led.h, tedy LED_1 – LED_4.

void led_init()
Funkce provede inicializaci knihovny, kdy dojde k nastavení vývodů mikropočítače na výstupy, ovládané pomocí bloku GPIO.

void led_on(uint8_t led)
void led_off(uint8_t led)
Umožňuje rozsvítit, resp. zhasnout LED diodu, která je vybrána parametrem.

void_led_all_on()
void_ledd_all_off()
Umožnuje vypnout resp. zapnout všechny LED diody najednou

6.4.5. Tlačítka (buttons)

Knihovna slouží ke čtení uživatelských tlačítek. Parametr pro výběr tlačítka musí odpovídat definovaným hodnotám BUTTON_1 – BUTTON_4.

void button_init()
Funkce provede inicializaci knihovny, kdy dojde k nastavení směru vývodů na vstup a ovládání pomocí bloku GPIO.

uint8_t button_read(uint8_t button)
Funkce přečte hodnotu na vstupu mikropočítače a vrátí hodnotu 1, pokud bylo stisknuto tlačítko. Součástí funkce je odstranění zákmitů na tlačítku. Pozor na způsob zapojení tlačítek, kdy je při stisknutí tlačítka na vstupu log 0.

6.4.6. RTC MPC79410

Obvod MPC79410, který slouží k měření reálného času se záložním napájením, komunikuje s mikropočítačem po sběrnici I^2C. Hodnoty a nastavení obvodu jsou uloženy v SRAM, pro přístup k této paměti musí být použita I^2C adresa 0x6F. Obvod umožňuje nastavení dvou alarmů, jejichž výstup může sloužit k vyvolání přerušení na multifunkčním vývodu (MFP). Na tomto vývodu může být generován hodinový

signál s požadovanou frekvencí. Obvod umožňuje pomocí kalibrace doladit frekvenci hodinového signálu, přidáním či odebráním x * 2 taktů za minutu.

Adresa	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	ST	Sekundy – 10			Sekundy – 1			
0x01	–	Minuty – 10			Minuty – 1			
0x02	–	12/24	H20	H10	Hodiny – 1			
0x03	–	–	OSCON	V_{BAT}	V_{BATEN}	Den		
0x04	–	–	Datum – 10		Datum – 1			
0x05	–	–	LP	M10	Měsíc – 1			
0x06	Rok - 10				Rok – 1			
0x07	OUT	SQWE	ALM1	ALM0	EXT_{OSC}	RS2	RS1	RS0
0x08	Kalibrace							
0x0A 0x11	–	Sekundy – 10			Sekundy – 1			
0x0B 0x12	–	Minuty – 10			Minuty – 1			
0x0C 0x13	–	12/24	H20	H10	Hodiny – 1			
0x0D 0x14	POL	OC2	OC1	OC0	IF	Den		
0x0E 0x15	–	–	Datum – 10		Datum – 1			
0x0F 0x16	–	–	–	M10	Měsíc – 1			

Tabulka 6.3: Obsah SRAM obvodu MPC79410

Hodnoty na adrese 0x00 – 0x08 slouží k uložení hodnot, stavu a ovládání obvodu RTC. Adresy 0x0A – 0x10 resp. 0x11 – 0x16 obsahují nastavení alarmů ALM0 resp. ALM1. Význam jednotlivých stavových a kontrolních bitů:

- ST – slouží k ovládání (povolení) krystalového oscilátoru
- 12/24 – určuje formát hodin (0 pro 24h; 1 – 12h a H20 určuje AM/PM)
- OSCON – obsahuje stav oscilátoru (pouze pro čtení)
- V_{BAT} – informuje, zda je k napájení použita záložní baterie (pouze pro čtení)
- V_{BATEN} – slouží k povolení záložního zdroje
- LP – obsahuje informaci, zda je aktuální rok přestupný (pouze pro čtení)
- OUT – určuje hodnotu na vývodu MFP, pokud negeneruje výstupní signál
- SQWE – povoluje generování hodinového signálu
- ALM1, ALM0 – povoluje jednotlivé alarmy
- EXT_{OSC} – povoluje externí oscilátor namísto krystalu
- RS[2:0] – slouží k nastavení frekvence výstupního hodinového signálu
- POL – definuje hodnotu signálu na MFP, pokud je splněna podmínka alarmu
- C[2:0] – určuje hodnoty, které se musí rovnat pro vyvolání alarmu
- IF – příznak přerušení (pouze pro čtení)

void rtc_mpc_set(uint8_t *bcd_value)
Funkce slouží k nastavení hodnot pro RTC obvod, hodnoty jsou doplněny o bity nutné ke správnému běhu obvodu. Parametrem funkce je adresa na data v desítkovém formátu.

void rtc_mpc_get(uint8_t *bcd_value)
Funkce umožňuje načíst hodnoty z RTC obvodu, z těchto hodnot jsou odstraněny kontrolní a stavové bity. Parametrem funkce je adresa pro uložení dat v desítkovém formátu.

void rtc_mpc_get(uint8_t *bin_value)
void rtc_mpc_set(uint8_t *bin_value)
Funkce mají stejný význam jako předchozí, akorát data jsou předávány v binární podobě.

6.4.7. SD a micro SD karta

Tato knihovna vychází z projektu „MMC/SD/SDHC card library“ [16], jejímž autorem je Roland Riegel. Knihovna je upravena pro použití na vývojové desce AT32BOARD a obsahuje podporu pro SD a SDHC karty včetně oddílů, systému souborů FAT16, resp. FAT32, čtení i zápis souboru. Kvůli způsobu implementace knihovny FAT je rozdělena na dvě části, z nichž jedna přistupuje ke klasické SD kartě a druhá k microSD kartě. Pokud by bylo potřeba podporovat oba formáty najednou, je nutné provést zásah do jednotlivých částí této knihovny.

Vzhledem k velikosti této knihovny a četné dokumentaci funkcí v jednotlivých souborech či na původní stránce projektu jsem se rozhodl tuto knihovnu v textu diplomové práce přeskočit.

6.4.8. PS/2 klávesnice

Knihovna slouží k ovládání klávesnice připojené přes rozhraní PS/2. Jednotlivé klávesy je možné rozdělit na základní a speciální (přidaný kód 0xE0). Propojení hodnot a kláves lze nalézt v tabulce Scan Codes verze 2 [25].

void keyboard_init()
Funkce slouží k inicializaci PS/2 rozhraní – ovládání vývodů pomocí GPIO a povoluje přerušení. Následně provede nastavení konfigurace klávesnice – četnost opětovného odeslání kódu klávesy při jejím dlouhodobém stisknutí atd.

void keyboard_led()
Funkce zapne led diody dle globálních proměnných, které jsou použity vzhledem k různým přístupům k těmto hodnotám.

uint8_t keyboard_code_get()
Funkce slouží k přečtení přijatého znaku. Dle jeho hodnoty je následně vykonána další operace, 0xE0 označuje speciální znaky a 0xF0 označuje uvolnění klávesy.

void keyboard_code_set(uint8_t code)
Funkce odesílá příkaz do klávesnice. Hodnoty příkazu musí odpovídat dokumentaci.

uint8_t keyboard_parity(uint8_t data)
Funkce vypočte hodnotu paritního bitu, který je přenášen po odeslání dat. Protokol PS/2 využívá lichou paritu.

6.4.9. Teplotní čidlo DS18B20

Knihovna pro komunikaci s teplotním čidlem firmy Dallas, která probíhá přes jednodrátovou sběrnici byla poupravena z původní verze, která vznikla během mé bakalářské práce „Programovatelný termostat k bojleru“ [5].

Komunikace začíná resetovacím pulsem, na ten zařízení odpovídá přítomnostním pulsem, po kterém následuje ROM příkaz, jenž umožňuje specifikovat adresu zařízení na sběrnici, a funkční příkaz.

Čidlo obsahuje 64bitový ROM kód (umožňuje adresaci určitého čidla) na sběrnici, a „Scratchpad“ paměť obsahující data a konfiguraci – naměřenou teplotu 2B, nastavení alarmu 2B, nastavení režimu konverze 1B, pouze interní použití 3B a CRC 1B.

void ds18b20_init()
Funkce slouží k inicializaci čidel na sběrnici – nastavení rozlišení převodu teploty a spodní, resp. hodní úrovně alarmů, které je zkopírováno do EEPROM paměti kvůli problémům s výpadkem napájení jednotlivých čidel.

void ds18b20_send_byte(uint8_t cmd)
Funkce slouží k odeslání bajtu po sběrnici.

uint8_t ds18b20_read_byte()
Funkce slouží k přijetí bajtu po sběrnici.

void ds18b20_convert ()
Funkce vyvolá měření teploty a převod její hodnoty do digitální hodnoty. Převod probíhá ve všech čipech připojených ke sběrnici.

uint8_t ds18b20_read_temp()
Funkce slouží k načtení a zpracování teploty od teplotního čidla. Funkce umožňuje čtení hodnoty pouze s jedním čipem připojeným ke sběrnici. Návratovou hodnotou je naměřená teplota, která je oříznuta o desetinnou část.

uint8_t ds18b20_read_temp_spec(uint64_t id)
Funkce opět slouží ke čtení teploty, avšak tentokrát čtení hodnoty probíhá z čipu, jehož 64bitová adresa odpovídá hodnotě předané parametrem id. Pokud čidlo není přítomno, bude načtena neplatná hodnota.

uint64_t ds18b20_read_id()
Funkce umožňuje čtení 64bitového id.

6.4.10. **LCD displej**

Knihovna slouží k ovládání LCD displeje, který je připojen k vývojové desce přes rozhraní UART. Firmware displeje, který mi byl zapůjčen vedoucím práce, byl navržen v diplomové práci „Konfigurovatelný řídící modulární systém“ [4].

void lcd_init()
Funkce slouží k inicializaci komunikace s LCD displejem – nastavení vývodů mikropočítače, nastavení režimu a generátoru bloku UART_2.

void lcd_clear()
Slouží ke smazání celého LCD displeje. Po odeslání příkazu čeká 5ms, aby nedošlo ke ztrátě dalších příkazů.

void lcd_backspace()
Funkce smaže znak zapsaný před kurzorem.

void lcd_position_X(uint8_t position)
void lcd_position_Y(uint8_t position)
Umožňují přesunout kurzor na pozici X, resp. Y zadanou parametrem.

void lcd_position_line_begin()
Slouží k přesunutí kurzoru na začátek aktuálního řádku.

void lcd_position_home()
Přesune kurzor do levého horního rohu displeje.

void lcd_display_char(uint8_t char)
Provede zápis znaku předaného parametrem a posune kurzor na následující pozici.

void lcd_display_string(uint8_t *text)
Provede zápis textového řetězce, jehož adresa je předaná parametrem. Textový řetězec musí být zakončen hodnotou 0.

void lcd_display_hex4(uint8_t value)
void lcd_display_hex8(uint8_t value)
void lcd_display_hex16(uint16_t value)
void lcd_display_hex32(uint32_t value)
Funkce slouží k zobrazení hexadecimální hodnoty předané parametrem value. Počet bitů vstupu je určen v názvu funkce.

void lcd_display_unumber8(uint8_t value)
void lcd_display_unumber16(uint16_t value)
void lcd_display_unumber32(uint32_t value)
Funkce slouží k zobrazení čísla bez znaménka v dekadické podobě. Hodnota v názvu funkce určuje počet bitů a maximální rozsah čísla (0 – 255, 0 – 65535, …).

void lcd_display_logo1()
void lcd_display_logo2()
Zobrazí LOGO1, resp. LOGO2, které je uložené v řadiči displeje.

6.5. Demo aplikace

Tato podkapitola se věnuje demo aplikacím, které slouží k demonstraci základních funkcí vývojové desky AT32BOARD. Zdrojové kódy jednotlivých aplikací jsou uložené na CD. Každý program začíná inicializací desky, požitých knihoven a bloků mikropočítače.

6.5.1. DEMO1

Aplikace slouží k demonstraci komunikace se sériovou linkou RS232, sběrnicí RS485, teplotním čidlem DS18B20, LCD displejem, SD kartou a obvodem pro měření reálného času připojeným přes sběrnici I^2C.

Po dokončení inicializace je načten čas a datum z obvodu RTC – MPC79410, jenž využívá záložní napájení, následně je pro měření času využit interní blok mikropočítače. Aplikace každou minutu provede 2x konverzi a čtení teploty z čidla DS18B20. Aktuální hodnota teploty je zobrazena na LCD displej. Pokud byla naměřena jiná teplota než v předchozím případě, dojde k odeslání hodnot (čas a teplota) přes sériovou linku a zápisu do souboru „log1.txt“ na SD kartu.

6.5.2. DEMO2

Aplikace slouží k demonstraci komunikace s klávesnicí připojenou přes rozhraní PS/2 či USB, displejem LCD, sériovou linkou RS232 a microSD kartou.

Program simuluje psací editor. Pomocí klávesnice je možné provádět zápis textu do bufferu, který umožňuje zapsat sto řádků s délkou až sto znaků na řádek. Průběh psaní je zobrazen na LCD displej. Při stisku klávesy F1 dojde k odeslání bufferu přes sériovou linku. Při stisku klávesy F2 dojde k uložení textu na konec souboru „log2.txt“.

K realizace USB je využita knihovna ASF.

6.5.3. DEMO3

Aplikace využívá rozhraní USB v režimu device a generátor audio výstupu blok ABDAC. Program vznikl na základě UC3 USB Audio Class [26] s využitím knihovny ASF.

7. Testování

7.1. Problémy prvního prototypu

Pří návrhu prvního prototypu plošného spoje došlo k několika zásadním chybám.

- Otvory pro konektory byly navrženy s nedostatečnou rezervou, a proto některé konektory vyžadovaly před osazením na plošný spoj úpravu.
- Chyby v pouzdrech součástek, které vyžadovaly např. osazení konektorů ze spodní strany plošného spoje.
- Při návrhu pouzdra a schématické značky lineárního stabilizátoru došlo k prohození vývodů GND a OUTPUT, které vedlo ke zkratu výstupního napětí. Stabilizátor sice nebyl poškozen, ale bylo nutné opravit a vyrobit nový plošný spoj.

7.2. Osazování a oživování druhého prototypu

První testování plošného spoje proběhlo po výrobě formou elektronického testu. Následovala vizuální kontrola a cvičné opasování konektorů a čipů, poté bylo provedeno osazování a oživování jednotlivých částí.

Nejprve byl osazen spínaný zdroj a součástky potřebné pro napájecí větev 5V. Testování zdroje probíhalo pomocí digitálního voltmetru připojeného k jeho výstupu a zvyšování vstupního napětí z hodnoty 0V až na 24V. Při tomto testu byla ověřena správná hodnota napětí, která odpovídá toleranci.

Další test byl proveden po osazení lineárního stabilizátoru a součástek pro napájecí větev 3,3V. Výstupní hodnota byla opět kontrolována pomocí voltmetru.

Po ověření napětí jednotlivých větví byly osazovány jednotlivé čipy včetně blokovacích kondenzátorů a potřebných součástek. Testování této fáze probíhalo hlavně kontrolou před zkraty, např. měřením napětí u blokovacích kondenzátorů, vizuální kontrolou vývodů čipů, kde není umožněno měření či jsou od sebe dostatečně vzdáleny, a měřením sousedních vývodů např. na konektorech.

7.3. Problémy druhého prototypu

7.3.1. Studené spoji

Problémy se studenými spoji vznikají chybou během pájení součástek na plošný spoj a občas může být velmi komplikované je najít. Mohou být způsobeny nedostatečnou teplotou při pájení, nedostatkem tavidla či nečistotou (např. oxidace a mastnota) na vývodech součástek či plošném spoji.

Největší problém nastal při pokusu o naprogramování mikropočítače pomocí JTAG programátoru. Při připojení programátoru JTAGice mkII k vývojové desce se rozsvítily diody signalizující správné propojení. Bylo možné naměřit správnou hodnotu referenčního napětí, ale každý pokus o programování čipu z vývojového prostředí (Atmel Studio 6 i AVR32 Studia) skončil chybovou hláškou. K odstranění problému nakonec vedlo až přeletování JTAG konektoru, který obsahoval studený spoj na datovém vodiči TDO. Po této opravě již bylo možné mikropočítač bez problému naprogramovat.

Další studené spoje byly objeveny při ovládání LED diod či tlačítek. I zde bylo opět potřeba přeletovat vývody součástek.

7.3.2. Komunikace s LCD displejem

Knihovna pro komunikaci s LCD displejem byla navržena dle dokumentace k firmwaru, který vznikl jako součást diplomové práce „Konfigurovatelný řídící modulární systém“ [4], a její testování proběhlo na zapůjčeném školním displeji.

Při testování byl zjištěn problém v komunikaci, kdy jednotlivé příkazy na displeji zobrazovaly pouze speciální znaky. Stav se nezměnil ani při testu různých příkazů s požadovanou rychlostí, ani při testu stejných příkazů s různými rychlostmi dle standardu. Tento problém byl nejspíše způsoben nahráním jiného firmwaru, než popisuje dokumentace.

Problém bylo možné vyřešit napsáním vlastní knihovny pro ovládání displeje nebo sehnáním původního firmwaru. Nakonec jsem se rozhodl pro druhou možnost, vypůjčil si požadovanou diplomovou práci [4] na katedře a použil původní firmware, který odpovídá dokumentaci a komunikuje s knihovnou.

7.3.3. Ethernet

Součástí projektu je realizace komunikace po Ethernetu, která se skládá z konfigurace mikropočítače (blok MACB), osazení a konfigurace fyzické vrstvy a napsáním knihovny pro komunikaci.

Tento cíl nakonec nebyl splněn, kvůli problémům s fyzickou vrstvou osazenou na desce.

- Všechny pokusy o čtení registrů fyzické vrstvy končí slovem 0xFFFF, který je dán klidovou úrovní datového vodiče MDIO.
- Při připojení kabelu nedojde k nastavení rychlosti a režimu spojení.

Problém může být způsoben např. špatným čipem KSZ8031RNLI či studeným spojem.

7.4. Testování komunikačních rozhraní

USB
Připojením periferní desky s počítačem, který načetl informace o mikropočítači (USB bootloader).

RS232
Propojení s počítačem s využitím převodníku USB <–> RS232.

RS485
Obvod pro sériovou sběrnici RS485 byl otestován pouze v režimu vysílání s kontrolou signálu na osciloskopu, jiné testování nebylo možné, protože jsem neměl k dispozici žádný modul podporující tuto komunikaci. Jelikož byl ale blok UART1 otestován pomocí RS232, lze předpokládat správné chování v obou režimech.

Klávesnice PS/2
Komunikace s klávesnicí byla provedena nejprve pomocí osciloskopu a následně pomocí výpisu jednotlivých znaků na LCD displeji.

7.5. Práce do budoucna

V této části bych se chtěl věnovat možným vylepšením periferní desky.

Popisky konektorů JPx, sloužící k volitelnému propojení periferií, a rozšiřujících konektorů by bylo vhodné umístit přímo na navrhovanou desku. Tato varianta by umožnila snadnější návrh softwarových knihoven a testování, stačil by pouze pohled na plošný spoj, ale komplikovala by návrh plošného spoje kvůli nedostatku místa pro jednotlivé popisky.

Umístit na desku plošného spoje testovací vývody, pro měření signálů mezi periferiemi (např. mezi CPU a SDRAM nebo mezi CPU a fyzickou vrstvu Ethernetu). Tyto vývody sice komplikují návrh plošného spoje, ale přináší výhody při testování, kdy je možné změřit hodnotu daného signálu např. pomocí osciloskopu.

8. Závěr

Zadáním diplomové práce bylo navrhnout a realizovat univerzální periferní desku, která dostala pracovní označení AT32_BOARD, osazenou 32bitovým mikroprocesorem, pracujícím s napětím 3,3V. Při její realizaci se vyskytla řada problémů např. nedostatky při návrhu prvního prototypu plošného spoje, které musely být odstraněny opravením a výrobou nové verze plošného spoje.

Prvním krokem hardwarové části práce byl výběr čipů a periferií vhodných pro vývojovou desku a návrh schématu zapojení. Návrh schématu komplikoval požadavek na využití, co největšího množství integrovaných periferních funkcí mikropočítače AT32UC3A0. Inspirací byly schémata pro vývojové desky od společnosti Atmel EVK1100 a EVK1105, které jsou osazeny čipem stejné řady.

Dalším úkolem byl návrh pouzder součástek, které jsem se rozhodl vytvořit dle jejich dokumentace, a návrh plošného spoje, jenž vyžadoval rozmístění součástek na plošném spoji a jejich následné propojení. Tento návrh by měl odpovídat pravidlům elektronického návrhu – odolnost vůči elektromagnetickému rušení a vyzařování.

Posledním krokem hardwarové části bylo ruční osazení a oživení vývojové desky, které vyžadovalo osazení asi 160 součástek. Největším problémem, který se v této části vyskytl, bylo hledání a následné odstranění problémů se studenými spoji, které částečně nebo zcela narušovaly správný chod jednotlivých bloků. Nakonec se povedlo rozchodit všechny bloky, kromě Ethernetu, u kterého mohlo dojít k znehodnocení či špatnému připájení čipu fyzické vrstvy.

Softwarová část spočívala v přípravě knihoven pro mikropočítač AT32UC3A0 a periferní desku, které byly následně otestovány a využity v ukázkových aplikacích. Během psaní a testování těchto knihoven byly ještě objeveny některé problémy s deskou plošného spoje (např. nepřipájená strana LED diody), které nebyly na první pohled patrné a byly odstraněny.

Pro realizaci knihoven pro bloky uvnitř mikropočítače jsem čerpal z dokumentace k mikropočítači, kde je detailně rozepsáno nastavení a příklady možné konfigurace. K ovládání těchto bloků lze využít také knihovnu ASF, kterou vyvinula firma Atmel.

Knihovny pro nastavení a komunikace modulů připojených k desce, které taktéž vznikly během softwarového návrhu této práce, umožňují konfiguraci desky – nastavení vstupů a výstupů, komunikaci s LCD displejem připojeného přes UART, vkládat zpožďovací smyčky, atd. Ve většině případů využívají knihovnu pro ovládání bloků mikropočítače.

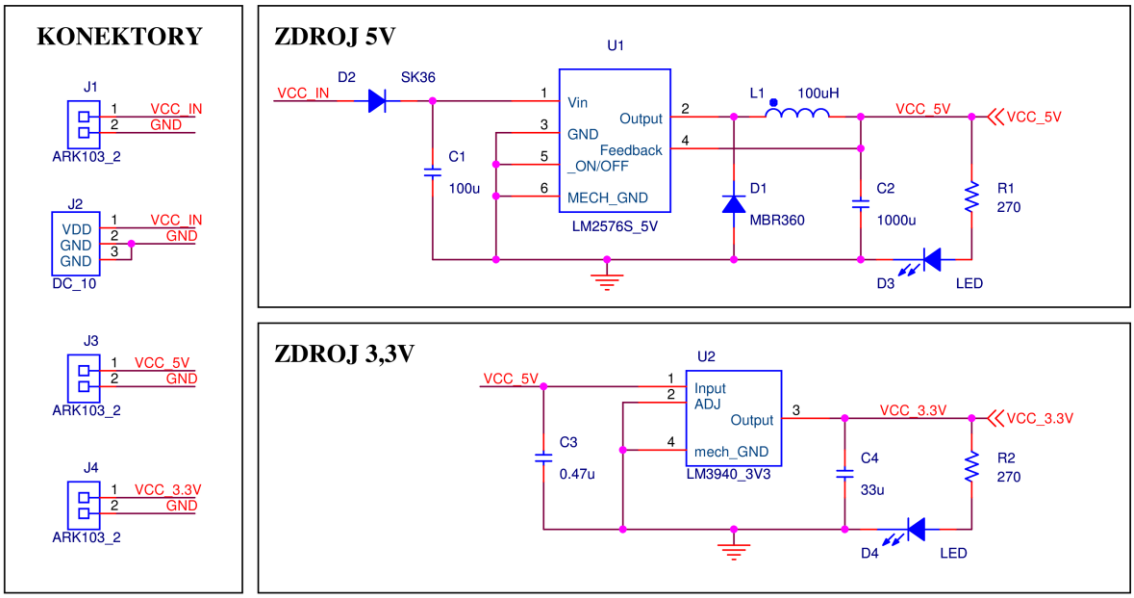
Zbytek softwarové části se věnuje Demo aplikacím, které umožňují demonstrovat správnou funkci knihoven pro AT32UC3A0 a AT32_BOARD. Tyto aplikace můžou sloužit jako inspirace pro další studenty pracující s touto vývojovou deskou.

9. Literatura

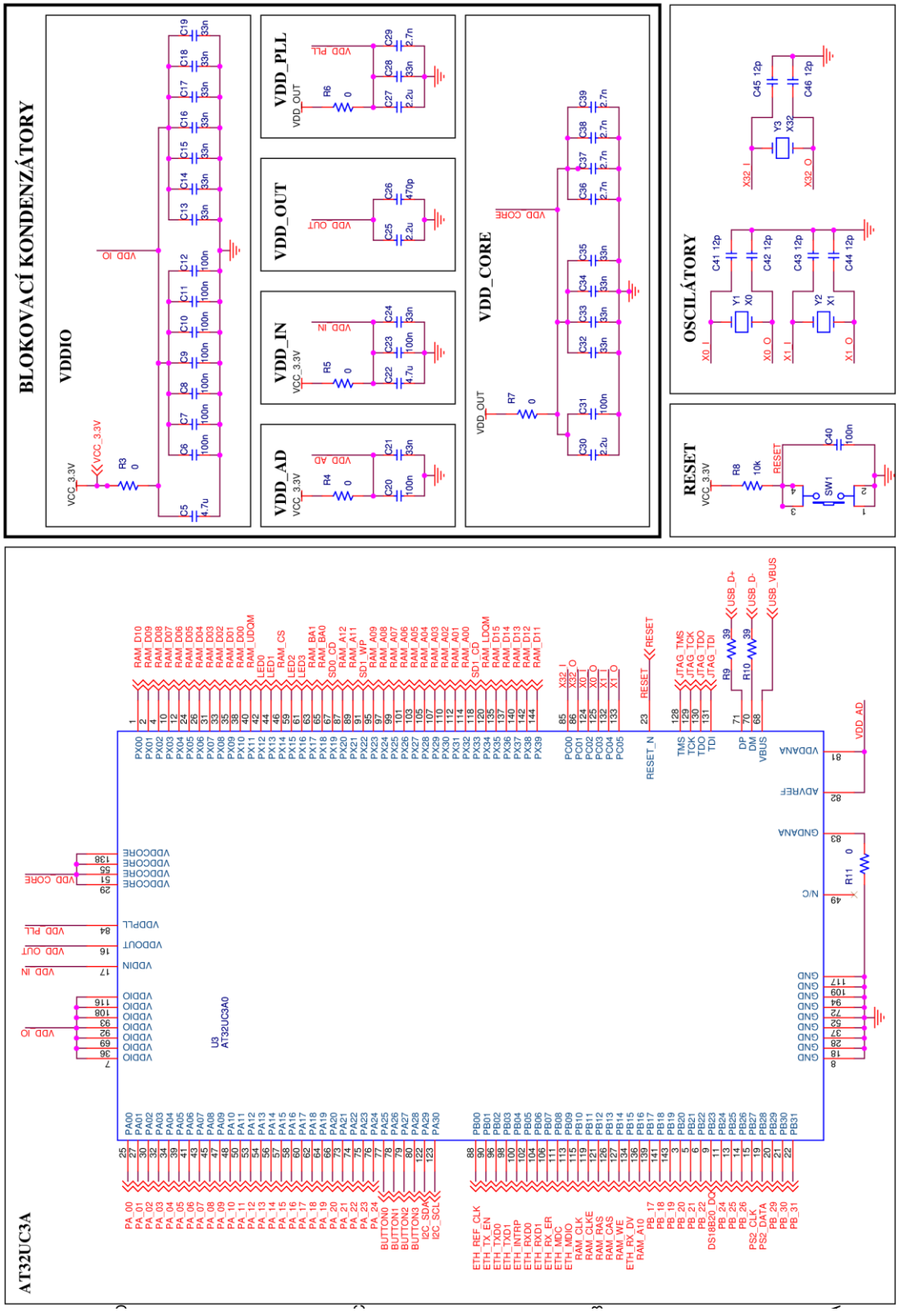
[1] Záhlava V., Návrh a konstrukce desek plošných spojů, Vydavatelství ČVUT, Praha 2005
[2] Vobecký, J. a Záhlava, V., Elektronika, Grada Publishing, Praha 2002
[3] Hrbek, L., Univerzální periferní deska, 2009
https://dip.felk.cvut.cz/browse/pdfcache/hrbekl2_2008dipl.pdf
[4] Nouzák, J., Konfigurovatelný řídící modulární systém, 2010
https://dip.felk.cvut.cz/browse/pdfcache/nouzajos_2010dipl.pdf
[5] Šesták, J., Programovatelný termostat k bojleru, 2010
[6] Atmel: UC3A Schematic Checklist
http://www.atmel.com/Images/doc32090.pdf
[7] Atmel: AT32UC3A0/A1 Series Complete
http://www.atmel.com/Images/doc32058.pdf
[8] Amic: A43L4616A
http://www.amictechnology.com/pdf/A43L4616A.pdf
[9] Micrel Semiconductor: KSZ8031RNLI
http://www.micrel.com/index.php/en/products/lan-solutions/phys/article/8-ksz8031rnl.html
http://www.farnell.com/datasheets/1447923.pdf
[10] Microchip: MPC79410
http://ww1.microchip.com/downloads/en/DeviceDoc/22266D.pdf
[11] Texas Instrument: MAX3232EID
http://www.ti.com/lit/ds/symlink/max3232e.pdf
[12] Texas Instrument: SN65HVD10DG4
http://www.ti.com/litv/pdf/slls505k
[13] Texas Instrument: LM2576S-5.0
http://www.ti.com/lit/gpn/lm2576
[14] Texas Instrument: LM3940IMP-3.3
http://www.ti.com/lit/gpn/lm3940
[15] Maxim: DS18B20 Programmable Resolution 1-wire Digital Thermomether
http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf
[16] Knihovny pro SD/SDHC (MMC/SD/SDHC Card Reader Library)
http://www.roland-riegel.de/sd-reader/index.html
[17] Pragoboard s.r.o., výrobce plošných spojů
http://www.pragoboard.cz

[18] Atmel: AVR JTAGice moji
http://www.atmel.com/tools/AVRJTAGICEMKII.aspx
[19] Atmel studio
http://www.atmel.com/tools/ATMELSTUDIO.aspx
[21] AVR32 UC3 USB DFU Bootloader Complete
http://www.atmel.com/Images/doc7745.pdf
[22] USB Host Mass Storage Bootloader on 32-bit AVR UC3
http://www.atmel.com/Images/doc7818.pdf
[23] USB Specification 2.0
http://www.usb.org/developers/docs/usb_20_040413.zip
[24] Atmel AVR 32-bit Architecture Manual Complete
http://www.atmel.com/Images/doc32000.pdf
[25] AVR32UC Technical Reference Manual
http://www.atmel.com/Images/doc32002.pdf
[26] PS/2 Keyboard Scan Codes - Set2
http://www.computer-engineering.org/ps2keyboard/scancodes2.html
[27] AVR UC3 USB Audio Class
http://www.atmel.com/Images/doc32139.pdf

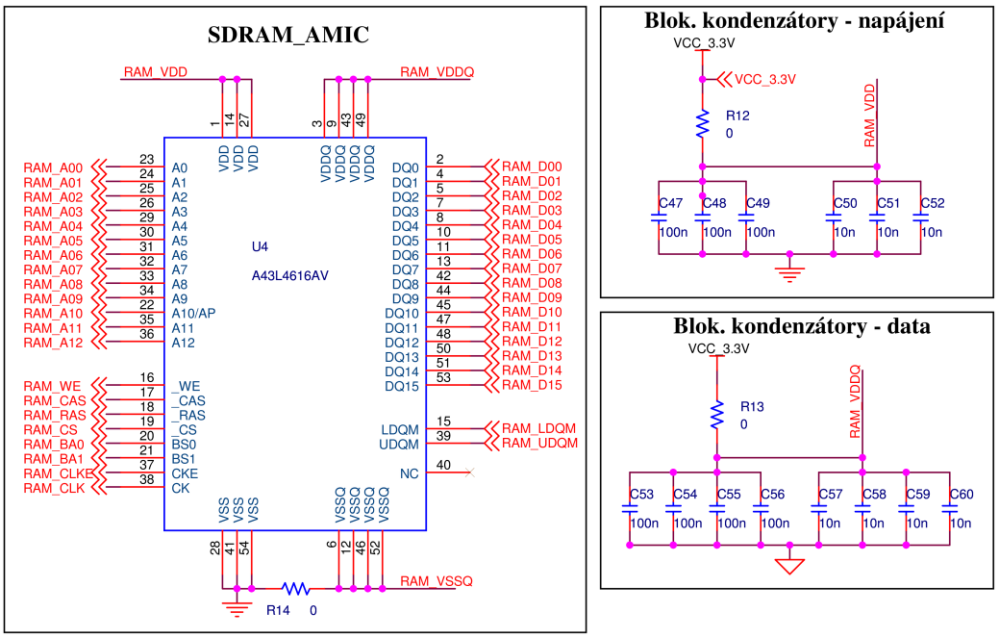
10. Schéma desky



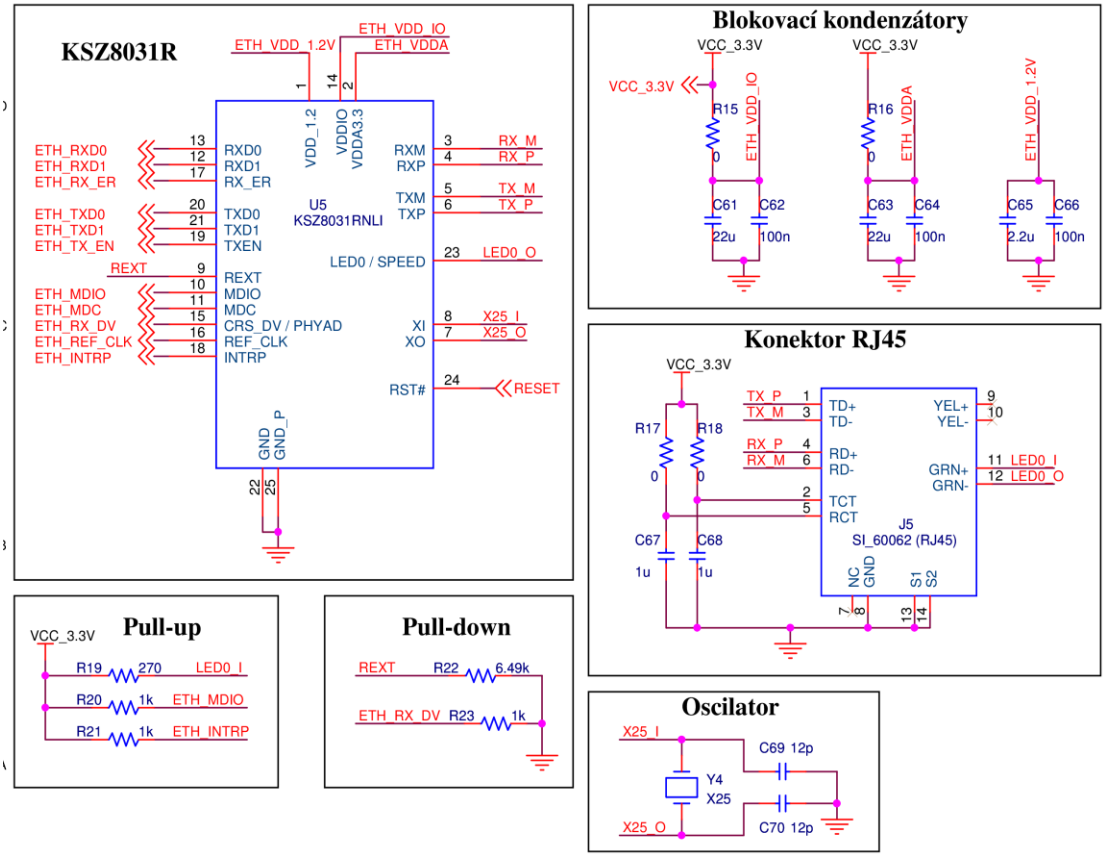
Obrázek 10.1: Napájení



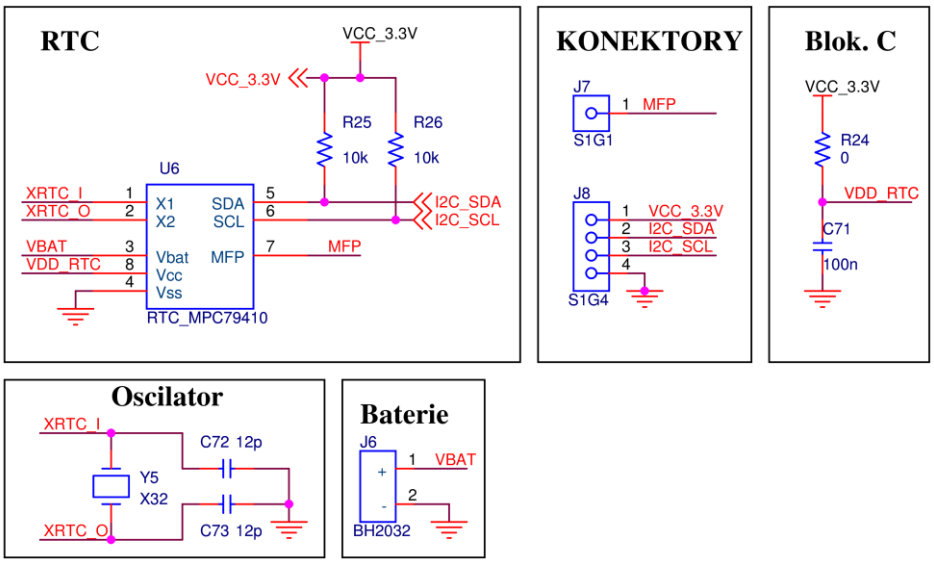
Obrázek 10.2: Mikropočítač AT32UC3A0



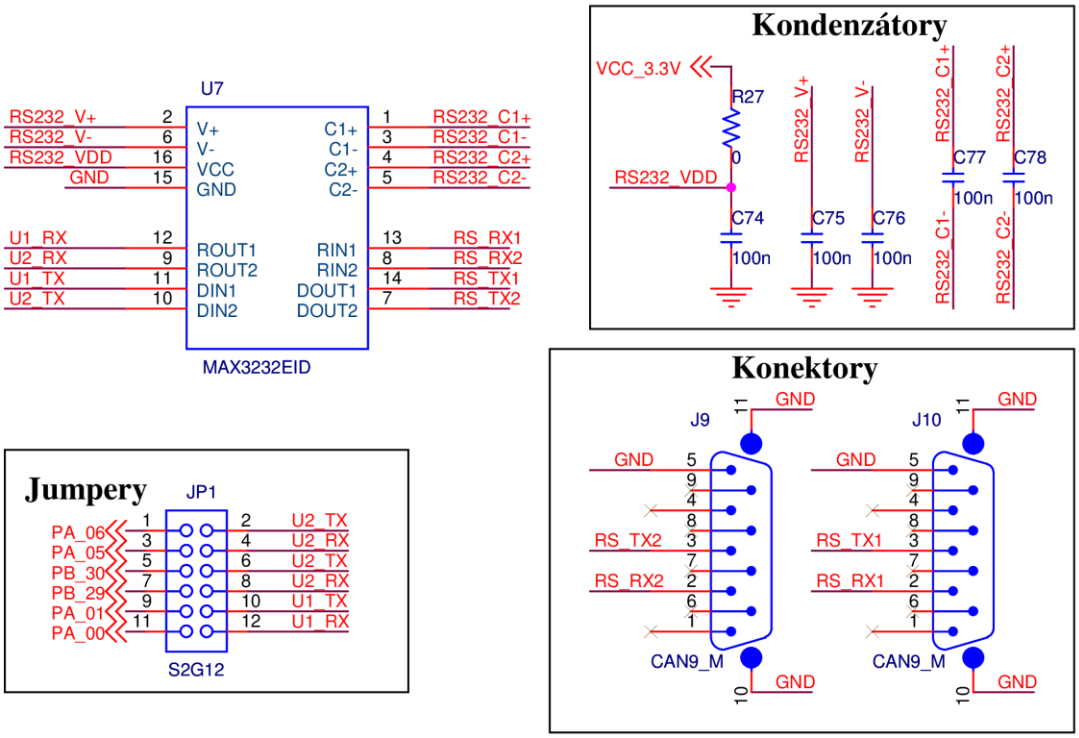
Obrázek 10.3: SDRAM paměť



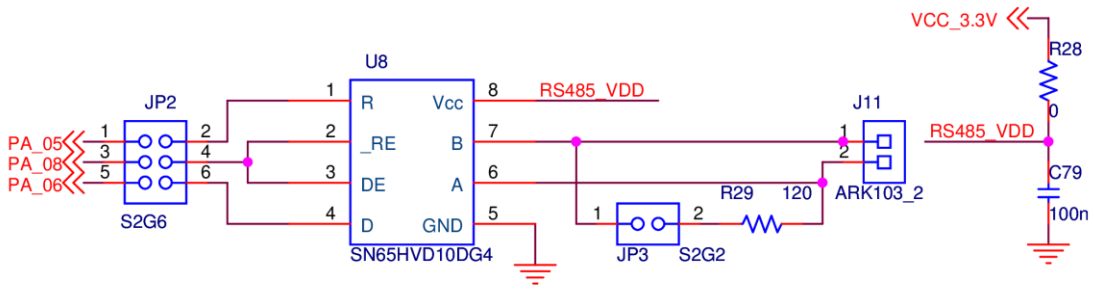
Obrázek 10.4: Ethernet



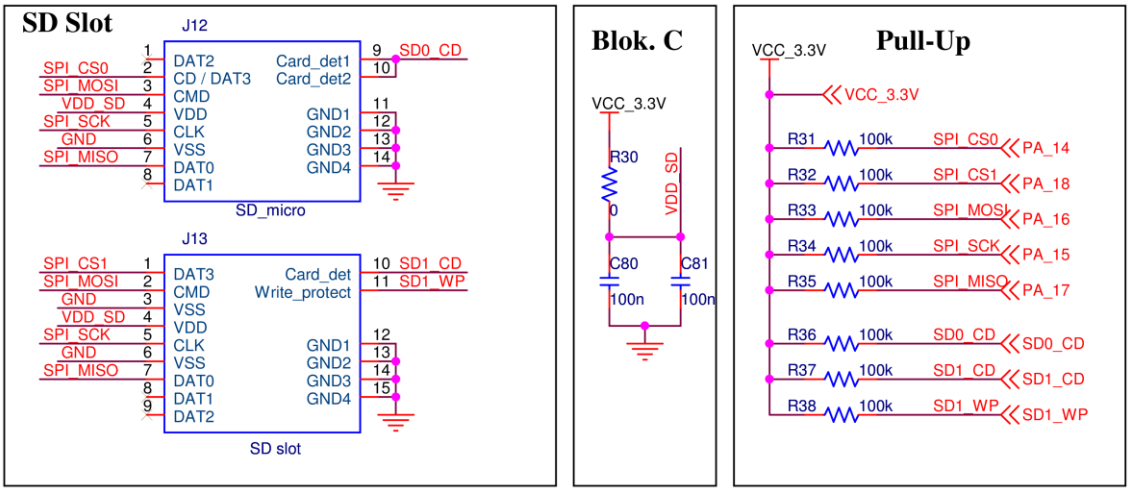
Obrázek 10.5: Sběrnice I^{2}C a obvod RTC



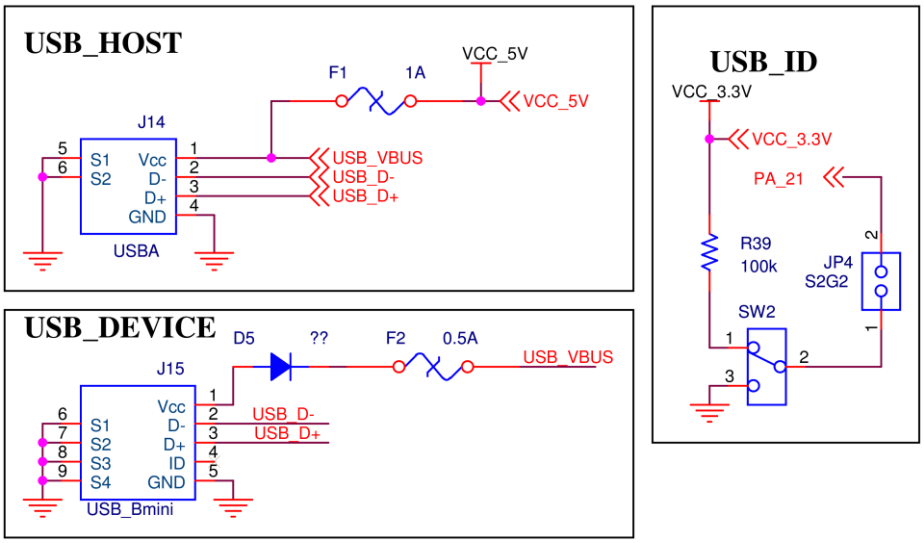
Obrázek 10.6: RS232



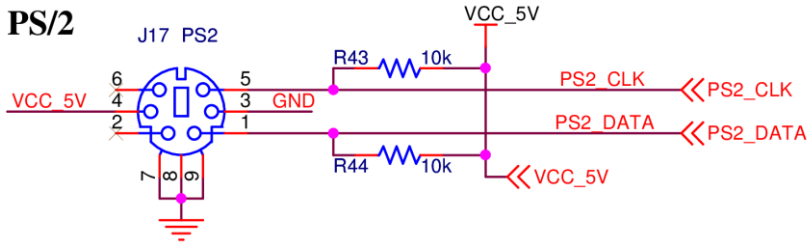
Obrázek 10.7: RS485



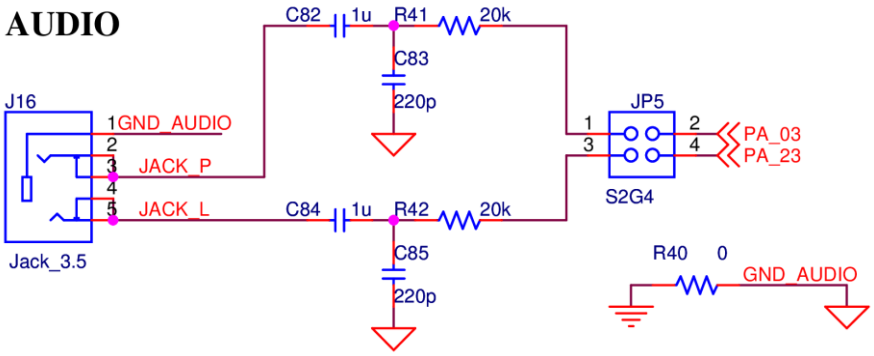
Obrázek 10.8: SPI a sloty pro SD a micro SD karty



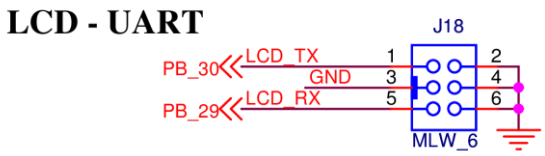
Obrázek 10.9: USB



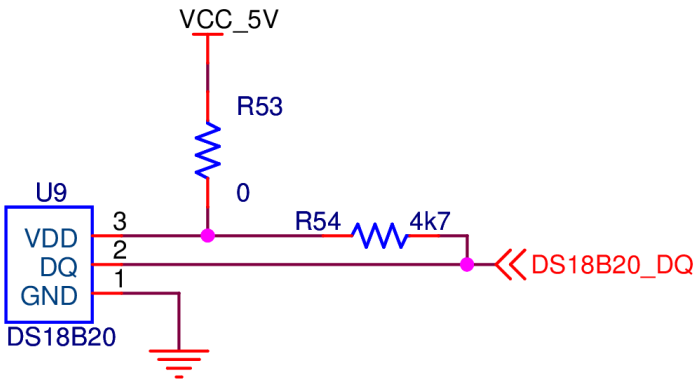
Obrázek 10.10: PS/2 klávesnice



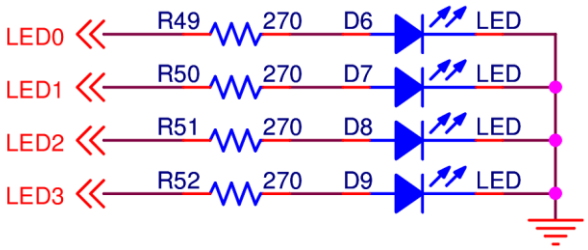
Obrázek 10.11: Audio výstup



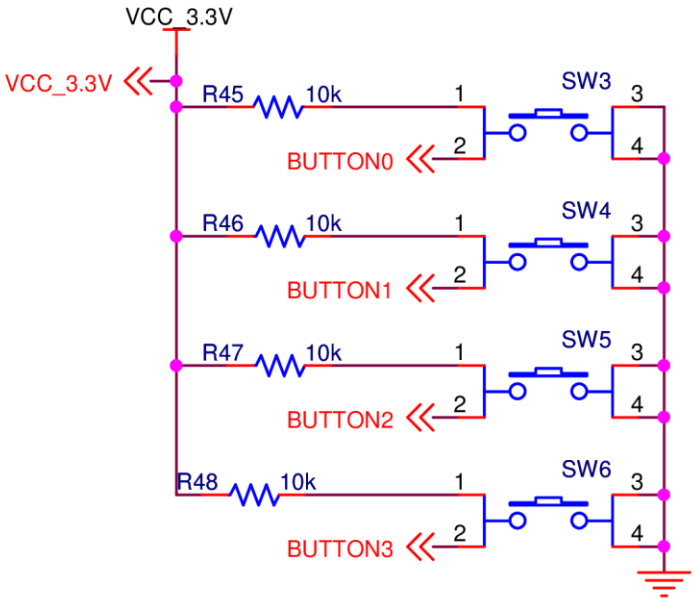
Obrázek 10.12: Konektor pro LCD přes UART



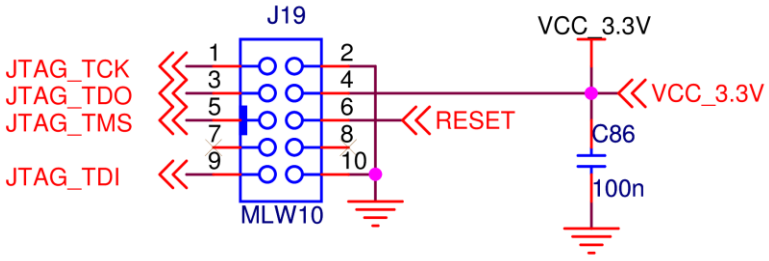
Obrázek 10.13: Teplotní čidlo DS18B20



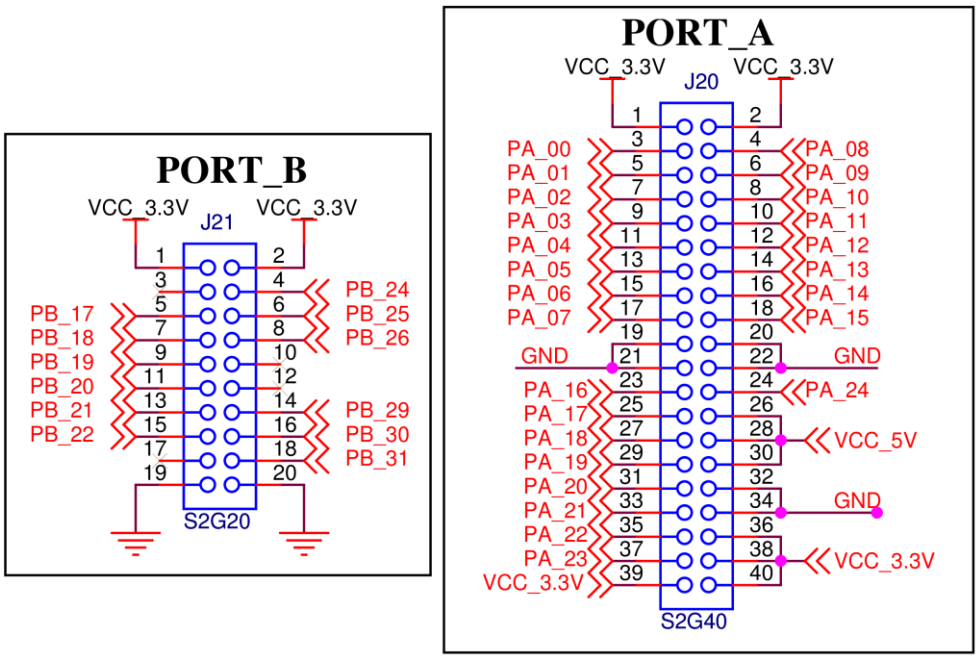
Obrázek 10.14: Led diody



Obrázek 10.15: Tlačítka

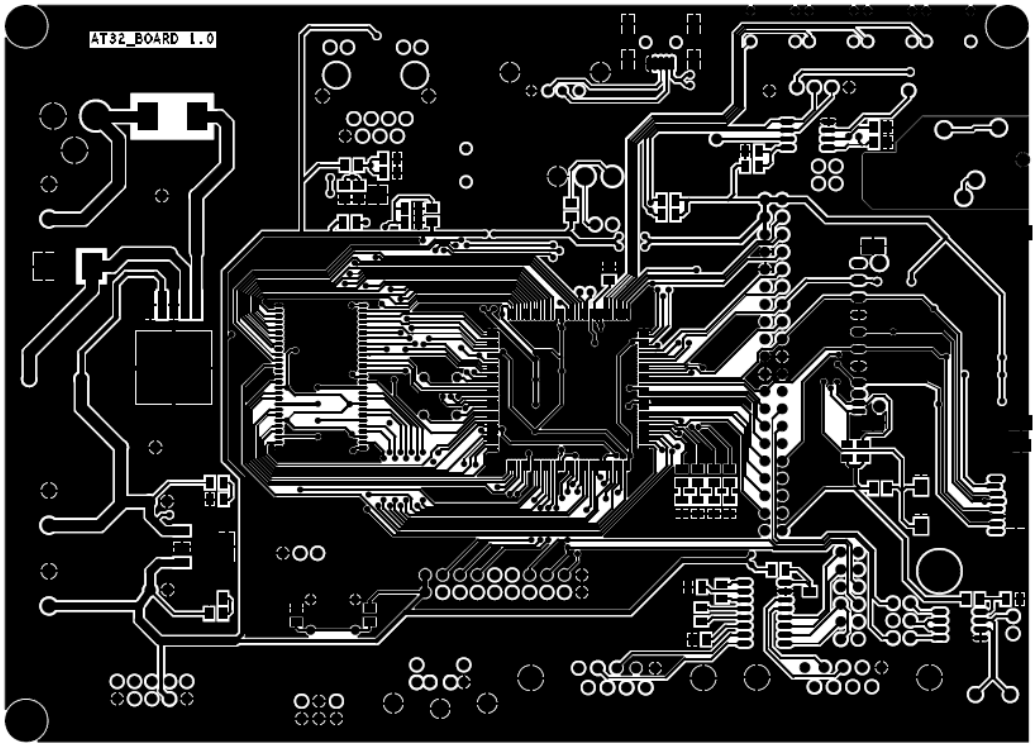


Obrázek 10.16: JTAG konektor

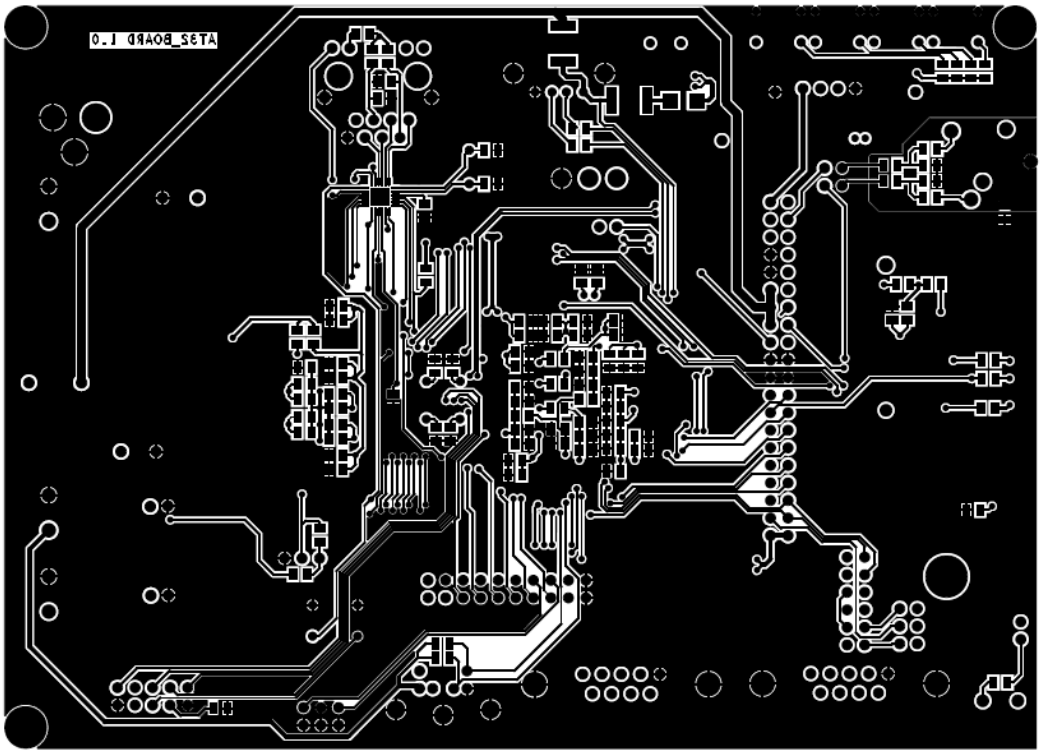


Obrázek 10.17: Rozšiřující konektory

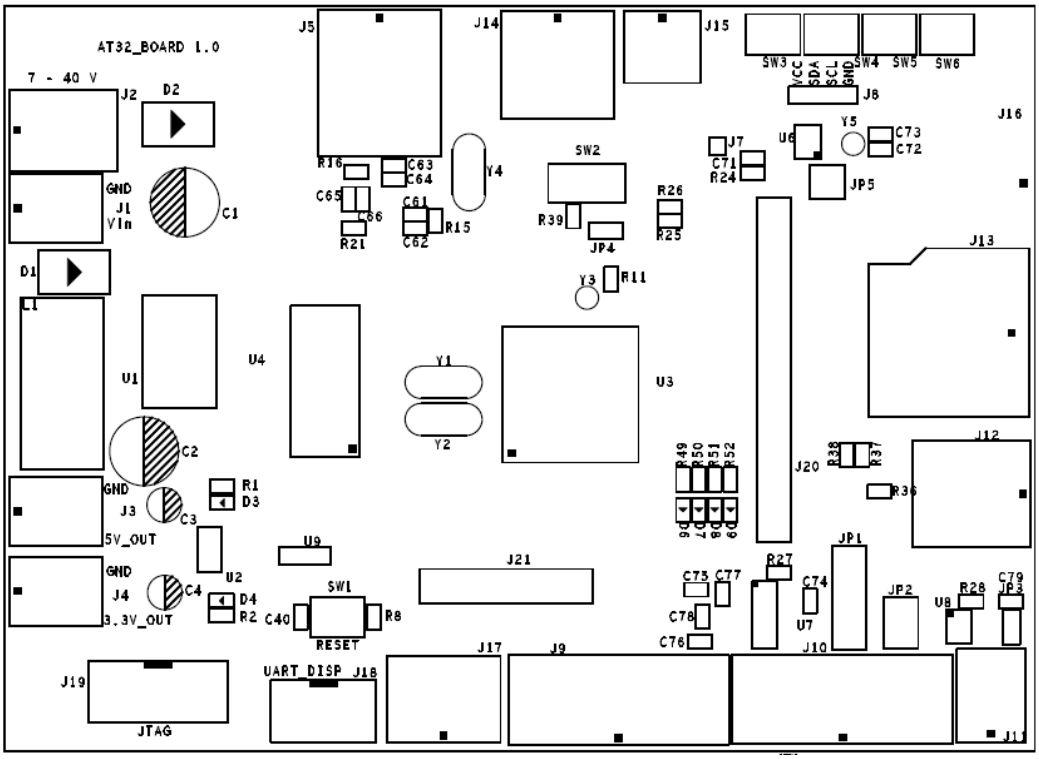
11. DPS a rozmístění součástek



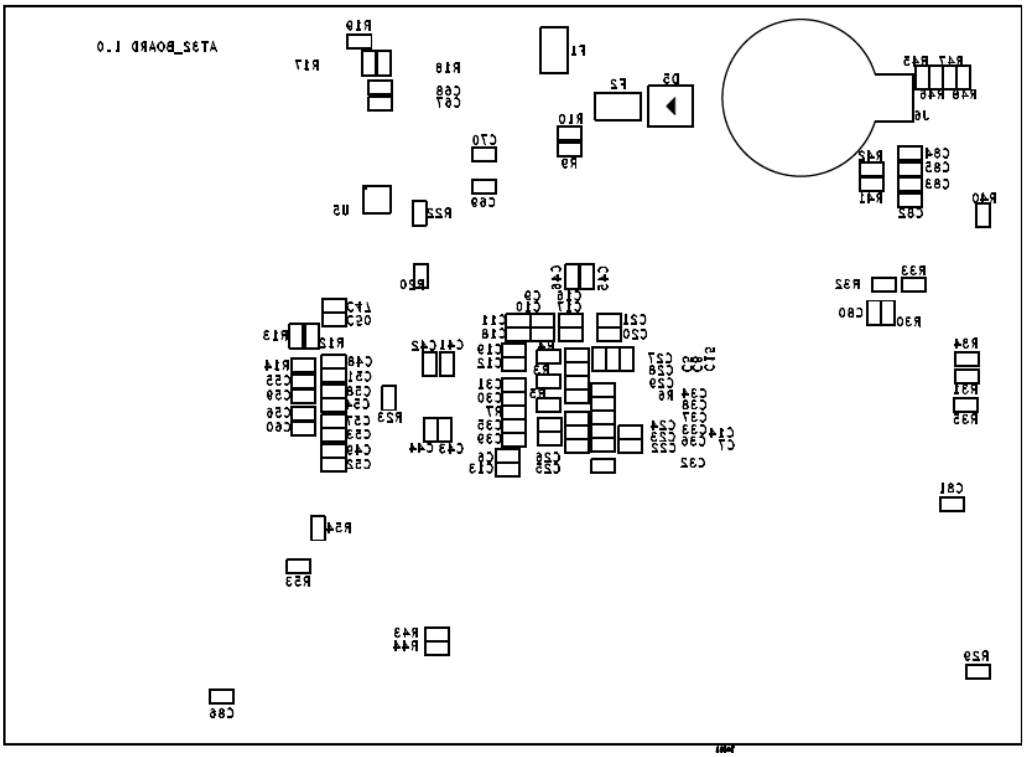
Obrázek 11.1: Měděná plocha – vrchní strana desky



Obrázek 11.2: Měděná plocha – spodní strana desky

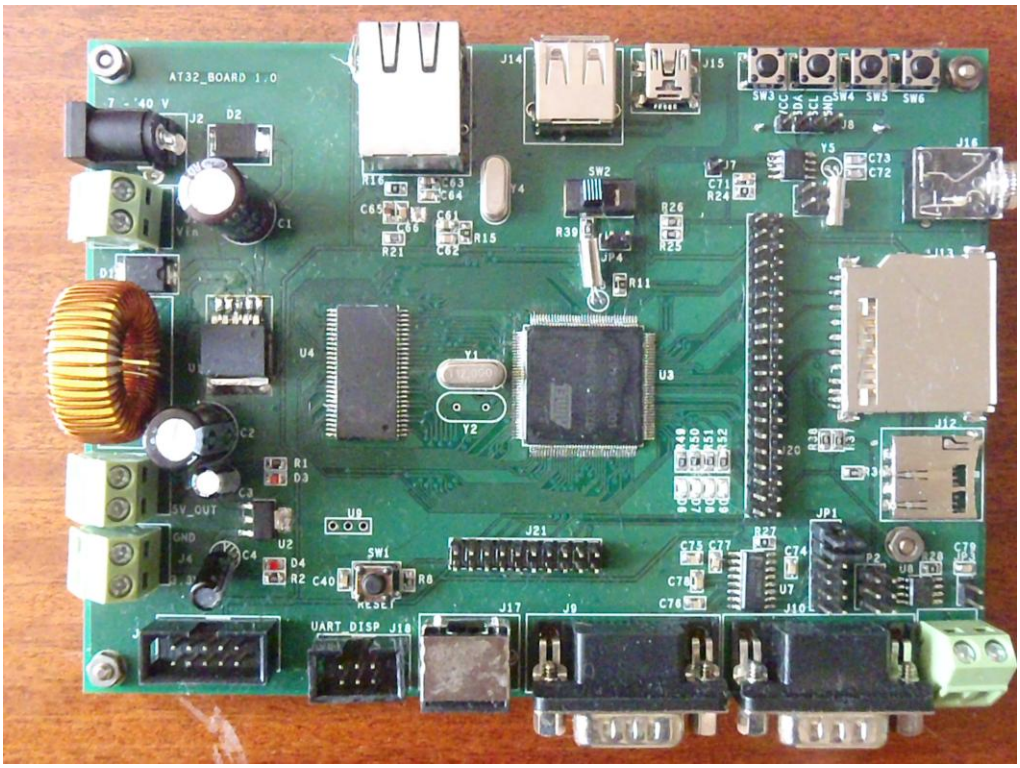


Obrázek 11.3: Rozmístění součástek – vrchní strana desky

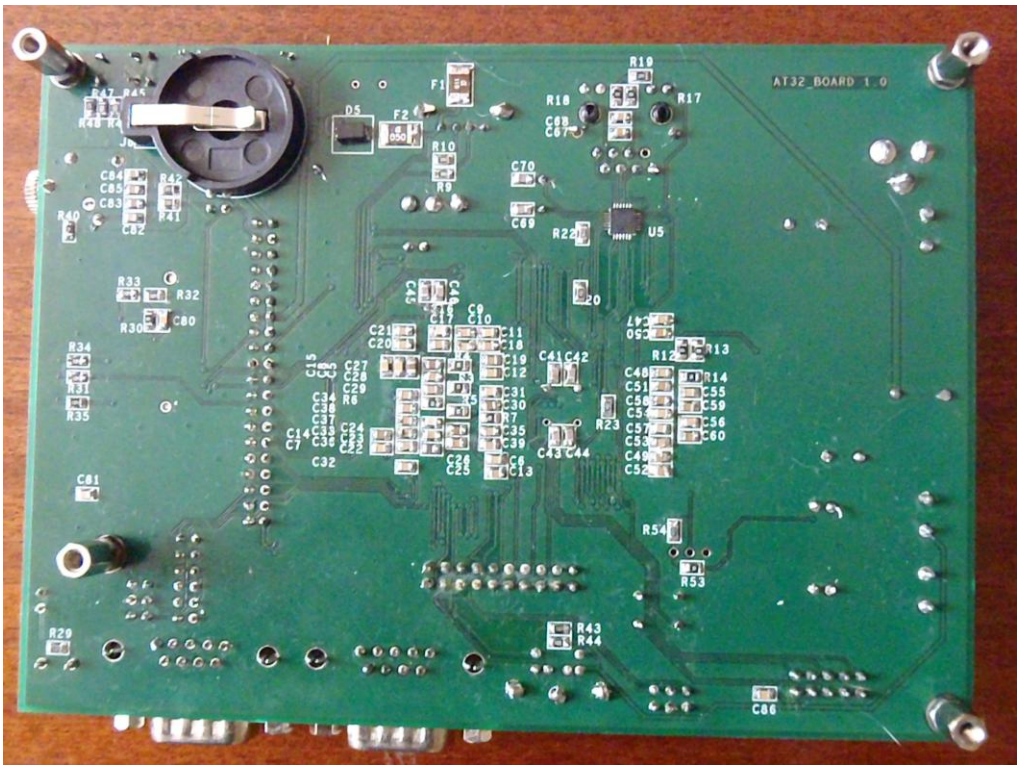


Obrázek 11.4: Rozmístění součástek – spodní strana desky

12. Osazená deska



Obrázek 12.1: Osazená deska – vrchní strana



Obrázek 12.2: Osazená deska – spodní strana

13. Zapojení vývodů mikropočítače

	PORT	Možnosti	
		GPIO	Funkce
70	DM(usb)		USB – DATA -
71	DP(usb)		USB - DATA +
49	n/C		
85	PC00	GPIO_064	Oscilátor – XIN32
86	PC01	GPIO_065	Oscilátor – XOUT32
124	PC02	GPIO_066	Oscilátor – XIN0
125	PC03	GPIO_067	Oscilátor – XOUT0
132	PC04	GPIO_068	Oscilátor – XIN1
133	PC05	GPIO_069	Oscilátor – XOUT1
23	RESET_N		RESET
129	TCK		JTAG_TCK
131	TDI		JTAG_TDI
130	TDO		JTAG_TDO
128	TMS		JTAG_TMS

Tabulka 13.1: Zapojení vývodů – USB, Port_C, JTAG

	PORT	GPIO	Připojeno	
77	PA25	GPIO_025	BUTTON 0	GPIO
78	PA26	GPIO_026	BUTTON 1	GPIO
79	PA27	GPIO_027	BUTTON 2	GPIO
80	PA28	GPIO_028	BUTTON 3	GPIO
122	PA29	GPIO_029	I2C (TWI) – SDA	Funkce A
123	PA30	GPIO_030	I2C (TWI) – SDL	Funkce A

Tabulka 13.2: Zapojení vývodů – Port_A pevné

PIN	PORT	Možnosti			
		GPIO	FA	FB	Fc
25	PA00	GPIO_000	USART0 – RXD	TC – CLK0	
27	PA01	GPIO_001	USART0 – TXD	TC – CLK1	
30	PA02	GPIO_002	USART0 – CLK	TC – CLK2	
32	PA03	GPIO_003	USART0 – RTS	EIM – EXTINT[4]	DAC – DATA[0]
34	PA04	GPIO_004	USART1 – CTS	EIM – EXTINT[5]	DAC – DATAN[0]
39	PA05	GPIO_005	USART1 – RXD	PWM – PWM[4]	
41	PA06	GPIO_006	USART1 – TXD	PWM – PWM[5]	
43	PA07	GPIO_007	USART1 – CLK	PM – GCLK[0]	SPI[0] – NPCS[3]
45	PA08	GPIO_008	USART1 – RTS	SPI[0] – NPCS[1]	EIM – EXTINT[7]
47	PA09	GPIO_009	USART1 – CTS	SPI[0] – NPCS[2]	MACB – WOL
48	PA10	GPIO_010	SPI0 – NPCS[0]	EIM – EXTINT[6]	
50	PA11	GPIO_011	SPI0 – MISO	USB – USB_ID	
53	PA12	GPIO_012	SPI0 – MOSI	USB – USB_VBOF	
54	PA13	GPIO_013	SPI0 – SCK		
56	PA14	GPIO_014	SSC – TX_FRAME_SYNC	SPI1 – NPSC[0]	EBI – NCS[0]
57	PA15	GPIO_015	SSC – TX_CLOCK	SPI1 – SCK	EBI – ADDR[20]
58	PA16	GPIO_016	SSC – TX_DATA	SPI1 – MOSI	EBI – ADDR[21]
60	PA17	GPIO_017	SSC – RX_DATA	SPI1 – MISO	EBI – ADDR[22]
62	PA18	GPIO_018	SSC – RX_CLOCK	SPI1 – NPSC[1]	MACB – WOL
64	PA19	GPIO_019	SSC – RX_FRAME_SYNC	SPI1 – NPSC[2]	
66	PA20	GPIO_020	EIM – EXTINT[8]	SPI1 – NPSC[3]	
73	PA21	GPIO_021	ADC – AD[0]	EIM – EXTINT[0]	USB – USB_ID
74	PA22	GPIO_022	ADC – AD[1]	EIM – EXTINT[1]	USB – USB_VBOF
75	PA23	GPIO_023	ADC – AD[2]	EIM – EXTINT[2]	DAC – DATA[1]
76	PA24	GPIO_024	ADC – AD[3]	EIM – EXTINT[3]	DAC – DATAN[1]

Tabulka 13.3: Zapojení vývodů – Port_A volitelné

PIN	PORT	GPIO	Připojeno	
88	PB00	GPIO_032	ETH – REF_CLK	Funkce A
90	PB01	GPIO_033	ETH – TX_EN	Funkce A
96	PB02	GPIO_034	ETH – TXD[0]	Funkce A
98	PB03	GPIO_035	ETH – TXD[1]	Funkce A
100	PB04	GPIO_036	ETH_INTRP	GPIO
102	PB05	GPIO_037	ETH – RXD[0]	Funkce A
104	PB06	GPIO_038	ETH – RXD[1]	Funkce A
106	PB07	GPIO_039	ETH – RX_ER	Funkce A
111	PB08	GPIO_040	ETH – MDC	Funkce A
113	PB09	GPIO_041	ETH – MDIO	Funkce A
115	PB10	GPIO_042	RAM – SDCK	Funkce C
119	PB11	GPIO_043	RAM – SDCKE	Funkce C
121	PB12	GPIO_044	RAM – RAS	Funkce C
126	PB13	GPIO_045	RAM – CAS	Funkce C
127	PB14	GPIO_046	RAM – SDWE	Funkce C
134	PB15	GPIO_047	ETH – RX_DV	Funkce A
136	PB16	GPIO_048	RAM – ADDR[10]	Funkce C
9	PB23	GPIO_055	DB18B20_DQ (Temp)	GPIO
15	PB27	GPIO_059	PS2_CLK	GPIO
19	PB28	GPIO_060	PS2_DATA	GPIO

Tabulka 13.4: Zapojení vývodů – Port_B pevné

PIN	PORT	Možnosti			
		GPIO	FA	FB	Fc
139	PB17	GPIO_049	MACB –RX_CLK	USB – USB_VBOF	EBI – ADDR[23]
141	PB18	GPIO_050	MACB – SPEED	ADC – TRIGGER	PWM – PWM[6]
143	PB19	GPIO_051	PWM – PWM[0]	PM – GCLK[0]	EIM – SCAN[4]
3	PB20	GPIO_052	PWM – PWM[1]	PM – GCLK[1]	EIM – SCAN[5]
5	PB21	GPIO_053	PWM – PWM[2]	PM – GCLK[2]	EIM – SCAN[6]
6	PB22	GPIO_054	PWM – PWM[3]	PM – GCLK[3]	EIM – SCAN[7]
11	PB24	GPIO_056	TC – B0	USART1 – DSR	
13	PB25	GPIO_057	TC – A1	USART1 – DTR	
14	PB26	GPIO_058	TC – B1	USART1 – RI	
20	PB29	GPIO_061	USART2 – RXD	PM – GCLK[1]	EBI – NCS[2]
21	PB30	GPIO_062	USART2 – TXD	PM – GCLK[2]	EBI – SDCS
22	PB31	GPIO_063	USART2 – CLK	PM – GCLK[3]	EBI – NWAIT

Tabulka 13.5: Zapojení vývodů – Port_B volitelné

PIN	PORT	GPIO	Připojeno	
1	PX00	GPIO_100	RAM – DATA[10]	Funkce A
2	PX01	GPIO_099	RAM – DATA[09]	Funkce A
4	PX02	GPIO_098	RAM – DATA[08]	Funkce A
10	PX03	GPIO_097	RAM – DATA[07]	Funkce A
12	PX04	GPIO_096	RAM – DATA[06]	Funkce A
24	PX05	GPIO_095	RAM – DATA[05]	Funkce A
26	PX06	GPIO_094	RAM – DATA[04]	Funkce A
31	PX07	GPIO_093	RAM – DATA[03]	Funkce A
33	PX08	GPIO_092	RAM – DATA[02]	Funkce A
35	PX09	GPIO_091	RAM – DATA[01]	Funkce A
38	PX10	GPIO_090	RAM – DATA[00]	Funkce A
40	PX11	GPIO_109	RAM - UDQM	Funkce A
42	PX12	GPIO_108	LED0	GPIO
44	PX13	GPIO_107	LED1	GPIO
46	PX14	GPIO_106	RAM - CS	Funkce A
59	PX15	GPIO_089	LED2	GPIO
61	PX16	GPIO_088	LED3	GPIO
63	PX17	GPIO_087	RAM – BA1	Funkce A
65	PX18	GPIO_086	RAM – BA0	Funkce A
67	PX19	GPIO_085	SD1_CD	GPIO
87	PX20	GPIO_084	RAM – ADDR[12]	Funkce A
89	PX21	GPIO_083	RAM – ADDR[11]	Funkce A
91	PX22	GPIO_082	SD1_WP	GPIO
95	PX23	GPIO_081	RAM – ADDR[09]	Funkce A
97	PX24	GPIO_080	RAM – ADDR[08]	Funkce A
99	PX25	GPIO_079	RAM – ADDR[07]	Funkce A
101	PX26	GPIO_078	RAM – ADDR[06]	Funkce A
103	PX27	GPIO_077	RAM – ADDR[05]	Funkce A
105	PX28	GPIO_076	RAM – ADDR[04]	Funkce A
107	PX29	GPIO_075	RAM – ADDR[03]	Funkce A
110	PX30	GPIO_074	RAM – ADDR[02]	Funkce A
112	PX31	GPIO_073	RAM – ADDR[01]	Funkce A
114	PX32	GPIO_072	RAM – ADDR[00]	Funkce A
118	PX33	GPIO_071	SD0_CD	GPIO
120	PX34	GPIO_070	RAM – LDQM	Funkce A
135	PX35	GPIO_105	RAM – DATA[15]	Funkce A
137	PX36	GPIO_104	RAM – DATA[14]	Funkce A
140	PX37	GPIO_103	RAM – DATA[13]	Funkce A
142	PX38	GPIO_102	RAM – DATA[12]	Funkce A
144	PX39	GPIO_101	RAM – DATA[11]	Funkce A

Tabulka 13.6: Zapojení vývodů – Port_X pevné

14. Obsah přiloženého CD

Demo Application
Zdrojové kódy demo aplikací

Library
Knihovny pro ovládání bloků mikropočítače a modulů na desce

PCB
Obsahuje data pro výrobu plošného spoje a projekt programu OrCAD – schéma, knihovny a plošný spoj

Text
Obsahuje text a obrázky diplomové práce