Fakulta elektrotechnická
Katedra aplikované elektroniky a telekomunikací

Diplomová práce

Inteligentní řídící modul s dotykovým displejem

Autor práce: Bc. Jan Roth
Vedoucí práce: Ing. Radek Šalom

Plzeň 2013

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan ROTH**
Osobní číslo:	**E11N0086P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Dopravní elektroinženýrství a autoelektronika**
Název tématu:	**Inteligentní řídicí modul s dotykovým displejem**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Seznamte se s principy vzdáleného ovládání zařízení pomocí protokolu TCP/IP a rozhraní Ethernet.
2. Vyberte vhodné technické prostředky pro realizaci modulu.
3. Realizujte praktické řešení modulu za použití vhodného mikroprocesoru a vybraného dotykového displeje. Implementujte TCP/IP stack a s pomocí dotykového displeje vytvořte jednoduché a funkční grafické rozhraní vhodné pro ovládání vzdálených zařízení skrze Ethernet. Vytvořte vzorovou aplikaci, která bude snadno modifikovatelná a rozšířitelná pro další užití.
4. Zhodnoťte realizované řešení, celou práci podrobně popište a důkladně zdokumentujte vytvořené grafické rozhraní.

Rozsah grafických prací:	**podle doporučení vedoucího**
Rozsah pracovní zprávy:	**30 - 40 stran**
Forma zpracování diplomové práce:	**tištěná/elektronická**

Seznam odborné literatury:

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí diplomové práce:	**Ing. Radek Šalom** Katedra aplikované elektroniky a telekomunikací
Konzultant diplomové práce:	**Ing. Radek Šalom** Katedra aplikované elektroniky a telekomunikací
Datum zadání diplomové práce:	**15. října 2012**
Termín odevzdání diplomové práce:	**9. května 2013**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 15. října 2012

Abstrakt

Práce se zabývá návrhem řídícího modulu s dotykovým displejem, schopným komunikovat po Ethernet rozhraní. První kapitola se věnuje popisu použitých periferií a jejich vzájemné spolupráci s mikrokontrolérem Stellaris®LM3S9B92. Poté je vysvětlen princip činnosti grafické knihovny a vytvořen seznam API funkcí pro modifikaci grafického rozhraní. Následující část práce popisuje implementaci protokolového zásobníku uIP TCP/IP stack a odesílání UDP datagramů dle odezvy na stisknutí dotykového displeje. V poslední části práce je uveden souborový systém FatFs určený pro embedded systémy. Vzorová aplikace a výsledky práce jsou diskutovány v jejím závěru.

Klíčová slova

kalibrace dotykového displeje, uIP stack, FatFs modul

Abstract

Roth, Jan. *Smart control module with touchscreen* [*Inteligentní řídící modul s dotykovým displejem*]. Pilsen, 2013. Master thesis (in Czech). University of West Bohemia. Faculty of Electrical Engineering. Department of Applied Electronics and Telecommunications. Supervisor: Radek Šalom

---

The master thesis proposes a control module with touch screen capable of communication over the Ethernet. The first chapter is devoted to a description of the peripherals and their mutual cooperation with the Stellaris®LM3S9B92 microcontroler. The next part explaines the principles of the graphics library and lists API functions to modify the graphic user interface. The following part describes the implementation of the protocol stack uIP TCP/IP stack and the transmittion of UDP datagrams to respond to the touch screen pressing. The last part of the thesis inrtoduces the FatFs file system designed for embbeded area. The sample application and the results of the work are discussed in the conclusion.

Keywords

calibration of touch screen, uIP stack, FatFs module

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem svou závěrečnou práci vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 270 trestního zákona č. 40/2009 Sb.

Také prohlašuji, že veškerý software, použitý při řešení této diplomové práce, je legální.

V Plzni dne 7. května 2013

Bc. Jan Roth

.....

Podpis

Poděkování

Touto cestou bych rád poděkoval vedoucímu diplomové práce Ing. Radku Šalomovi za odborné vedení a užitečné rady při psaní této práce. Rád bych také poděkoval své rodině a mým blízkým za dlouhodobou podporu a výdrž po celou dobu studia, bez kterých bych do této fáze nemohl dospět.

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů a zkratek

AD	Analog to Digital converter. Analogově digitální převodník.
EEPROM	Electrically Erasable Programmable Read Only Memory. Typ permanentní paměti, kterou lze mazat elektrickým proudem.
FIFO	First in, First out. Paměť typu fronta, první dovnitř, první ven.
GDDRAM	Graphic Data Dynamic Random Access Memory. Vnitřní grafická DRAM paměť řadiče SSD1289.
JTAG	Join Test Action Group. Standard pro testování integrovaných obvodů.
LCD	Liquid Crystal Display. Technologie využívající tekutých krystalů pro zobrazení obrazu.
LDO	Low DropOut voltage regulator. Stabilizátor s malým úbytkem napětí.
LSB	Low signifacant bit. Bit s nejnižší vahou.
MAC	Media Access Controller. Logická vrstva Ethernet řadiče.
MDI	Media Dependent Interface. Obvykle připojovací konektor komunikačního rozhraní Ethernet.
MSB	Most signifacant bit. Bit s nejvyšší vahou.
PHY	Network Physical. Fyzická vrstva Ethernet.
PWM	Pulse Width Modulation. Pulsně šířková modulace.
SPI	Serial Peripheral Interface. Sériové rozhraní pro komunikaci s periferiemi.
SWD	Serial Wire Debug. Standard pro testování integrovaných obvodů.
TFT	Thin Film Transistor. Technologie tenkovrstvých tranzistorů.
VCP	Virtual COM Port. Virtuální COM port.
API	Application Programming Interface. Zde jsou jako API označeny funkce pro modifikaci grafického rozhraní.
RTC	Real Time Clock. Hodiny reálného času.

1

Úvod

Cílem diplomové práce je vytvořit řídící modul, který bude schopen ovládat vzdálená zařízení. Rozhraní mezi uživatelem a programem je dotykový displej, pro který byla naprogramována grafická knihovna. Důraz byl kladen na možnost jednoduchého rozšíření menu o další objekty. Dále je popsána kalibrace displeje, převod souřadnic stisknutého displeje, chyby kterými je tento převod zatížen a jejich potlačení.
První část práce se zabývá návrhem řídícího modulu. Tato kapitola je rozdělena do úrovní podle periferií. Zde je popsán princip činnosti periferie, způsob komunikace, případně je uvedeno schéma zapojení. Za účelem programování a ladění programu byl vytvořen programátor, který je popsán taktéž v této kapitole.
Následující část práce se věnuje popisu firmware. Jednotlivé podkapitoly se týkají konkrétních knihoven, které obsahují popis funkcí a vývojové diagramy. Každá knihovna je složena ze souboru .c a hlavičkového souboru .h, ve kterém jsou uvedeny prototypy funkcí, makra a definované datové typy. Tím je docíleno rychlejší orientace pro programátora. Dále je zde popsána implementace protokolového zásobníku TCP/IP μIP stack a jeho návaznost na aplikační vrstvu grafické knihovny. Závěr této kapitoly je vyhrazen souborovému systému FAT pro embedded zařízení. V poslední části je uvedena ukázková aplikace a způsob modifikování grafického rozhraní.

Řídící modul najde uplatnění především v průmyslu nebo v tzn. inteligentních domácnostech. Díky dotykovému displeji je možné sledovat okamžitý průběh veličin jako například teplotu, tlak, spotřebu elektrické energie atd. Poté lze pomocí ovládacích prvků na displeji řídit vzdálené akční členy, nebo s nimi pouze komunikovat. Věřím, že trend dotykového ovládání, hlavně v oblasti domácností, se brzy stane běžnou součástí života.

2

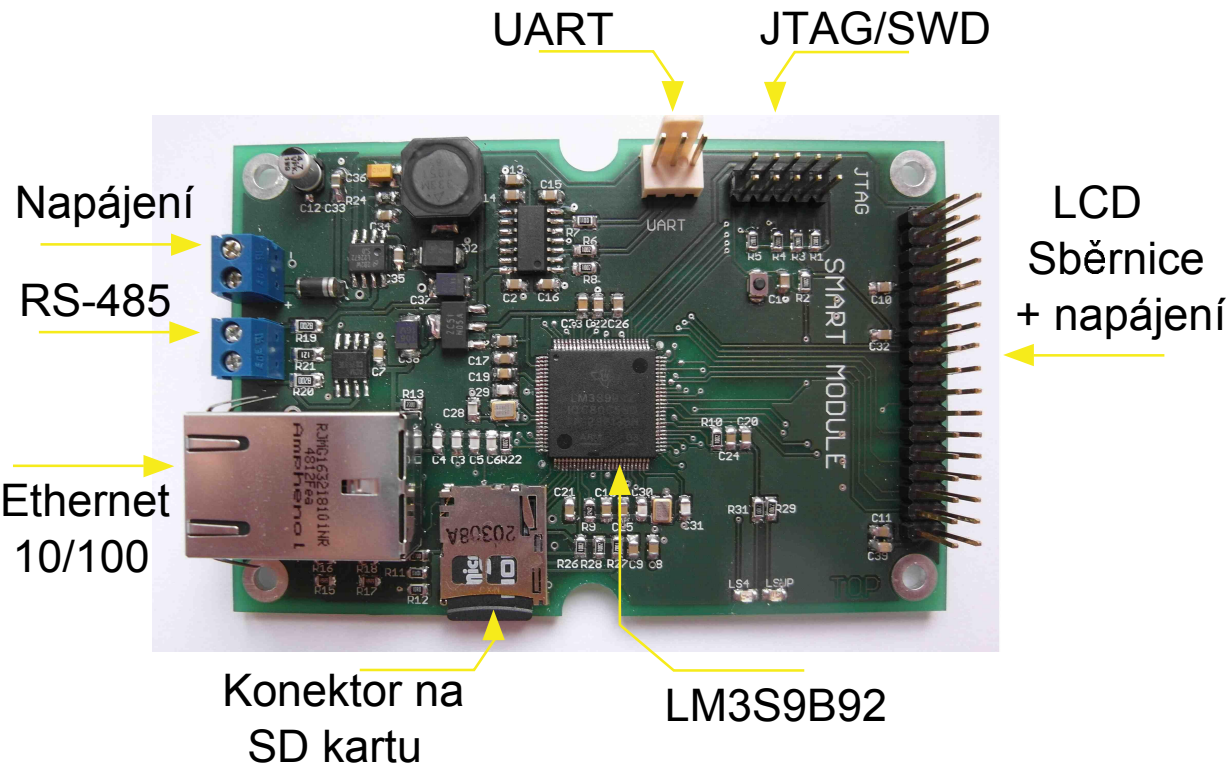
Hardware

Hlavním úkolem této kapitoly je představení dotykového modulu a jeho periferií. Pro účely programování a ladění programu byl dále vytvořen programátor, kterému je věnována poslední část této kapitoly.

2.1 Řídící modul

Řídící modul je vybaven *QVGA* dotykovým displejem, kterým lze pomocí komunikačního rozhraní *Ethernet 802.3* nebo *RS485* ovládat vzdálená zařízení. Pro grafická data, která svou velikostí přesahují interní paměť mikrokontroléru, bylo použito externího úložiště v podobě SD karty. Řídící modul je dále vybaven DC-DC měničem, který modulu umožňuje pracovat se vstupním napětím v rozsahu 12 V-24 V a tedy ho využít v různých aplikacích. Obsluha periferií je vykonávána mikrokontrolérem Stellaris *LM3S9B92*, který se řadí do rodiny ARM Cortex-M3. Ty jsou díky nízké spotřebě energie a dostatečnému výkonu používány v oblastech tenkých klientů, inteligentních domácích spotřebičů, nebo přístupových zařízeních. *LM3S9B92* byl vybrán dle konkrétních požadavků na počet a typ periferií. Kompletní informace k mikroprocesoru je možné nalézt v datových listech[1].

Konstrukce řídícího modulu byla rozdělena na dvě patra. Horní pohledová část je tvořena $3,2$" displejem, který je spojen s mikrokontrolérem pomocí dvouřadové lišty. Ta slouží jako komunikační sběrnice a zároveň napájení displeje. Spodní patro je osazeno Ethernetovým rozhraním, SD kartou, budičem sběrnice RS-485 a napájením. Touto konstrukcí řídícího modulu bylo dosaženo rozměrů 94 mm x 62 mm x 23 mm. Na obrázku 2.1 je zobrazen navržený modul. Příloha D obsahuje fotografickou dokumentaci.



Obr. 2.1: Zobrazení řídícího modulu

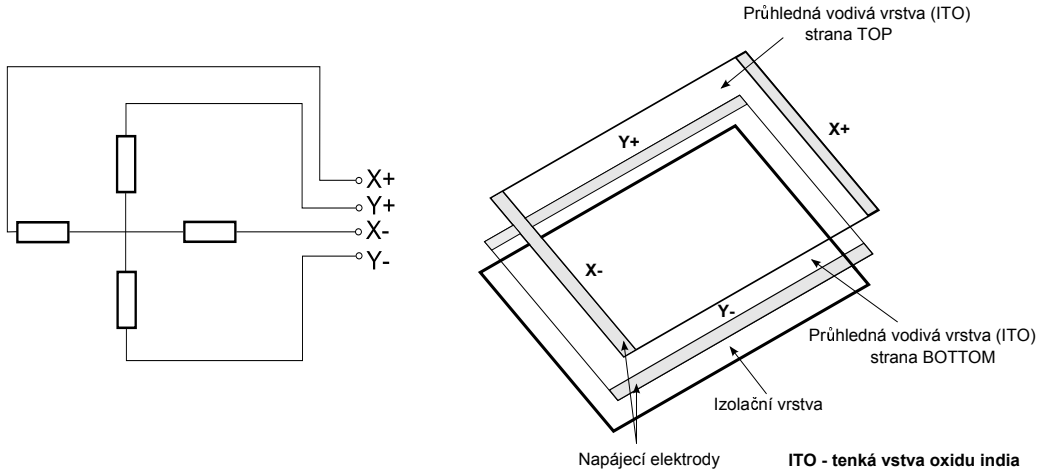
Následující kapitoly se věnují popisu periferií a komunikaci s mikrokontrolérem. Vyšší důraz bude kladen na vysvětlení činnosti dotykového displeje a potlačení chyb, které na něm vznikají.

2.1.1 Dotykový displej

Dotykový displej je klíčovou částí diplomové práce, proto je důležité pochopit princip jeho činnosti. Nejprve bude následovat teoretická část a poté bude uveden konkrétní dotykový displej a použitý řadič.

Princip činnosti dotykového displeje

Dotykový displej si lze představit jako čtyř-vodičovou odporovou síť, viz. obrázek 2.2*a)*. Po stisknutí displeje je na jeden pár vodičů přivedeno referenční napětí (např. $Y+/Y-$) a druhý pár slouží jako měřící sonda. Tím je získána první složka souřadnic. Poté je opačným způsobem změřena i druhá složka souřadnic. Díky vysoké vstupní impedanci převodníku není měření zatíženo chybou úbytku napětí na měřící sondě.
Schopnost přesně detekovat polohu prstu na displeji za různých pracovních podmínek se odvíjí od nastavení AD převodníku a potlačení přechodných jevů. Následující řádky budou blíže popisovat převodník $ADS7843$, který je přímo navržen pro aplikace s dotykovými displeji a také použit v diplomové práci.



(a) Čtyř-vodičová odporová síť (b) Struktura dotykového displeje

Obr. 2.2: Struktura a náhradní schéma dotykového displeje

AD převodník je schopen pracovat v režimech *single-ended mode* a *differential-mode*. V prvním případě, je-li detekováno stisknuti displeje, vyšle mikrokontrolér řídící byte, čímž začíná převod napětí. Na začátku vzorkovací periody je přes interní spínací tranzistory AD převodníku přivedeno napájecí napětí na jednotlivé vrstvy dotykového panelu, 2.2*b)*.

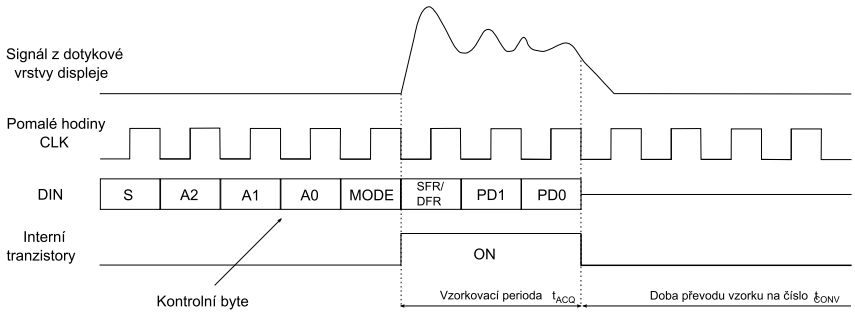
Toto napětí rychle roste a překmitává okolo stabilní úrovně. Doba, za kterou dojde k ustálení napětí je nazývána *settling time*. Tato doba a velikost rozkmitu napětí se výrazně podílejí na přesnosti převodu.

Po vzorkovací periodě jsou spínací tranzistory zavřeny a AD převodník zahájí převod vzorků na číslo. Podmínkou správného převodu je ustálení napětí během posledních tří bitů v řídícím bytu, viz.obrázek 2.3*a)*. Režim *differential-mode* se liší dobou spínání interních tranzistorů, které jsou sepnuty i během převodu napětí na číslo. Navíc napájecí napětí vrstev se stává referenčním napětím převodníku. Z toho plyne, že změní-li se napájecí napětí, např. v důsledku působení výkonových prvků na napájecí napětí, nebo změnou impedance displeje působením rozdílných teplot, budou tyto změny kompenzovány právě diferenčním měřením převodníku. V tomto pracovním režimu je také možné zkrátit dobu *settling time* tím, že interní tranzistory zůstanou otevřeny příchodu další řídící sekvence. Tím je získána delší doba pro ustálení vstupního analogového napětí.

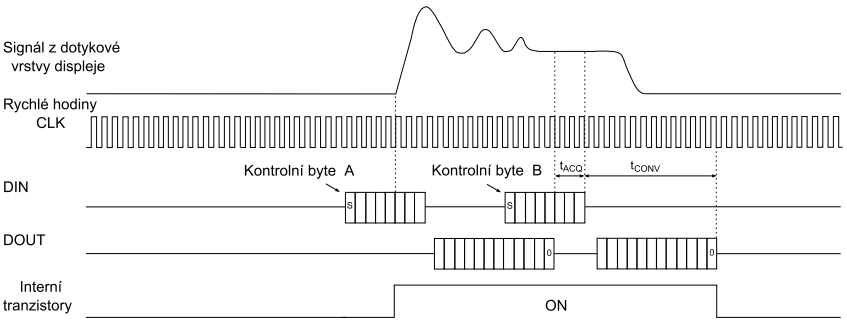
Potlačení přechodných jevů

Pro oba pracovní režimy převodníku platí, že hodnota vstupního analogového napětí musí být ustálená během posledních třech bitů řídícího bytu. První způsob potlačení nežádoucích kmitů je zobrazen na obrázeku 2.3*a)*. AD převodník je nastaven na *single-ended mode*. Po sepnutí spínacích tranzistorů nastává přechodný děj. Je-li použita nižší rychlost hodin CLK, narůstá tak doba pro odeznění přechodného jevu.

Druhá metoda využívá režimu *differential-mode* a mnohem vyšší rychlosti hodin CLK, obrázek 2.3*b)*. Spínací tranzistory jsou otevřeny řídícím bytem A. Opět dochází k nárůstu



(a) *Single-ended mode*, pomalé hodiny CLK



(b) *Differential mode*, rychlé hodiny CLK

Obr. 2.3: Způsoby potlačení přechodných dějů [2]

napětí a vzniku oscilací. Je provedena digitalizace napětí, avšak tentokrát nejsou spínací tranzistory vypnuty. Přichází druhý řídící byte B a je opět proveden převod napětí. Tímto způsobem je prodloužena doba pro odeznění přechodného děje a je docíleno převodu ustáleného napětí. Jak z popisu vyplývá, první paket převedených bitů se ignoruje [2].

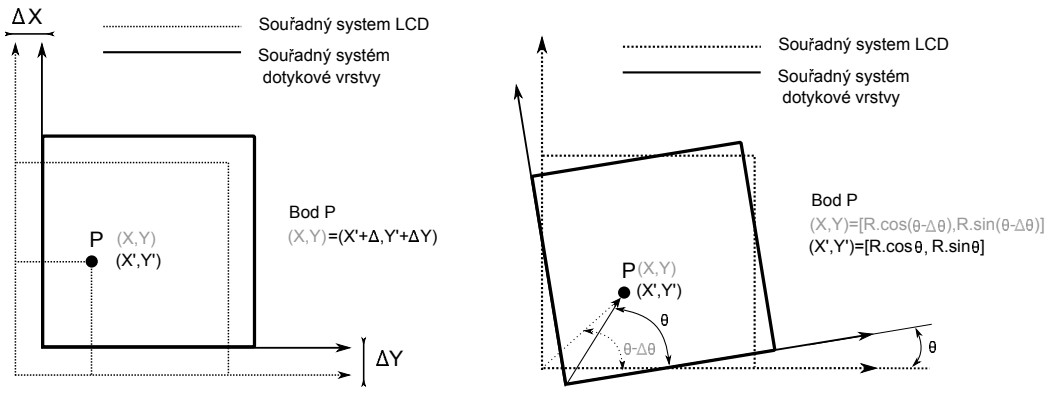
Filtraci signálu je dále dobré provádět také pomocí software. V kapitole 3.1 je popsána konkrétní metoda filtrace a zároveň průměrování signálu.

Kalibrace displeje

Kalibrace displeje by se měla provádět vždy, když souřadnice zobrazeného objektu na displeji nesouhlasí se souřadnicemi získanými z AD převodníku. Jejím účelem je kompenzovat zdroje chyb a dopočítat odchylku mezi zobrazenými a změřenými souřadnicemi tak, aby aplikace dokázala korektně odpovídat na stisknutí displeje.
K největším zdrojům chyb patří šum, chyba měřítka ($scalingfactor$) a mechanická nepřesnost. Šum je způsoben podsvícením displeje (jas je PWM signálem), elektrostatickým výbojem, nebo vibracemi povrchu displeje. Možnost určité korekce těchto chyb byla osvětlena v kapitole 2.1.1, případně lze čerpat z [3]. Tato kapitola se bude dále zabývat důsledky $scalingfactor$ a mechanické nepřesnosti.
Vlivem různého rozlišení displeje a AD převodníku je mikrokontrolér nucen upravit získané souřadnice určitým koeficientem tak, aby se co nejvíce blížily zobrazenému objektu.

Např. má-li displej rozlišení 320x240 bodů a AD převodník 12 bitů, je koeficient pro osu x roven $k_x = 240/4096 = 0,059$, pro osu y $k_y = 320/4096 = 0,078$. Naměřené hodnoty je poté nutné těmito koeficienty vynásobit a výsledek zaokrouhlit. Koeficienty k_x, k_y budou dále chápány jako lineární členy. Prakticky se však jedná o nelineární prvky v celé oblasti displeje, a proto je nutná kalibrace displeje.

Mechanická nepřesnost mezi dotykovou vrstvou a displejem má charakter posuvu v horizontální i vertikální rovině a zároveň rotace o určitý úhel. Na obrázku 2.4 jsou zobrazeny oba případy. Z těchto obr. plyne, že pro posuv platí $X = X' + \Delta X$ a $Y = Y' + \Delta Y$, kde (X, Y) jsou souřadnice zobrazeného bodu, (X', Y') jsou souřadnice naměřené převodníkem, $(\Delta X, \Delta Y)$ je posuv v obou rovinách. Obdobně lze z Pythagorovy rovnice vyjádřit souřadnice pro rotaci $X = R{\cdot}\,cos(\Theta - \Delta\Theta)$, $Y = R{\cdot}\,cos(\Theta - \Delta\Theta)$, kde R je vzdálenost počátku souřadného systému, Θ úhel mezi R a osou x, $\Delta\Theta$ rotace mezi systémy AD převodníku a displejem.



(a) Mechanická chyba posuvu (b) Mechanická chyba rotace

Obr. 2.4: Mechanická nepřesnost mezi dotykovou vrstvou a displejem [4]

Spojením těchto rovnic lze vyjádřit souřadnice jako

$$\begin{aligned} X &= k_x{\cdot}\, R{\cdot} \cos(\Theta - \Delta\Theta) + \Delta X \\ &= k_x{\cdot}\, R{\cdot} \cos\Theta{\cdot} \cos(\Delta\Theta) + k_x{\cdot}\, R{\cdot} \sin\Theta{\cdot} \sin(\Delta\Theta) + \Delta X \\ &= k_x{\cdot}\, X'{\cdot} \cos(\Delta\Theta) + k_x{\cdot}\, Y'{\cdot} \sin(\Delta\Theta) + \Delta X \\ &= \alpha_x{\cdot}\, X' + \beta{\cdot}\, Y' + \Delta X\,, \end{aligned} \tag{2.1}$$

kde $X' = R{\cdot} \cos\Theta$, $Y' = R{\cdot} \sin\Theta$, $\alpha_x = k_x{\cdot} \cos(\Delta\Theta)$, $\beta_x = k_x{\cdot} \sin(\Delta\Theta)$, obdobně lze zapsat souřadnici Y.

$$\begin{aligned} Y &= k_y \cdot R \cdot \sin(\Theta - \Delta\Theta) + \Delta Y \\ &= k_y \cdot R \cdot \sin\Theta \cdot \cos(\Delta\Theta) - k_y \cdot R \cdot \cos\Theta \cdot \sin(\Delta\Theta) + \Delta Y \\ &= k_y \cdot Y' \cdot \cos(\Delta\Theta) - k_y \cdot X' \cdot \sin(\Delta\Theta) + \Delta Y \\ &= \alpha_y \cdot X' + \beta_y \cdot Y' + \Delta Y\,, \end{aligned} \tag{2.2}$$

kde $\alpha_y = -k_y \cdot \sin(\Delta\Theta)$, $\beta_y = k_y \cdot \cos(\Delta\Theta)$.

Z těchto rovnic je zřejmé, že pro výpočet proměnných α, β, $\Delta X/Y$ je zapotřebí tří nezávislých bodů. Termín nezávislý znamená, že body neleží na jedné přímce. Na obrázku 2.5 jsou nakresleny oba způsoby zobrazení bodů. Je nutné mít taktéž na paměti, aby body neležely blízko okraje displeje, kde se více projevuje nelineární snímaní hodnot a zároveň, aby body nebyly blízko sebe. Tím je dosaženo správné kalibrace.
Za předpokladu volby nezávislých bodů lze rovnice 2.1 a 2.2 přepsat do tvaru

$$\begin{aligned} X_1 &= \alpha_x \cdot X_1' \cdot \beta_x \cdot Y_1' + \Delta X \\ X_2 &= \alpha_x \cdot X_2' \cdot \beta_x \cdot Y_2' + \Delta X \\ X_3 &= \alpha_x \cdot X_3' \cdot \beta_x \cdot Y_3' + \Delta X \\ \\ Y_1 &= \alpha_y \cdot X_1' \cdot \beta_y \cdot Y_1' + \Delta Y \\ Y_2 &= \alpha_y \cdot X_2' \cdot \beta_y \cdot Y_2' + \Delta Y \\ Y_3 &= \alpha_y \cdot X_3' \cdot \beta_y \cdot Y_3' + \Delta Y. \end{aligned} \tag{2.3}$$

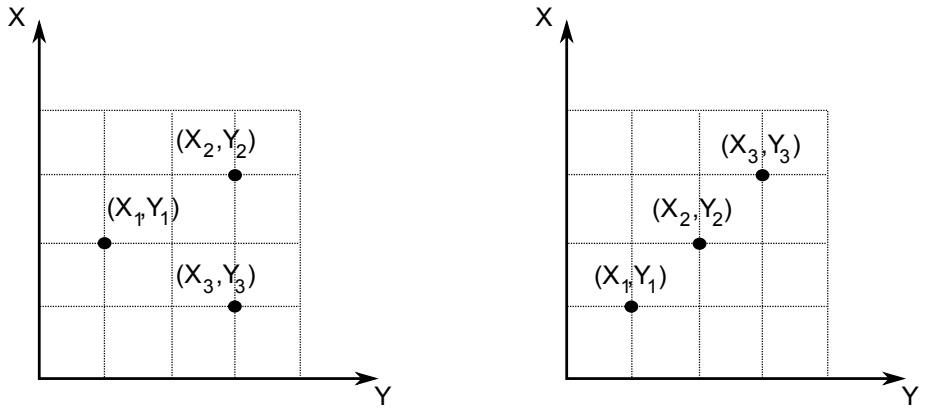
Soustava 2.3 je poté přepsána do maticové formy.

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = A \cdot \begin{pmatrix} \alpha_x \\ \beta_x \\ \Delta X \end{pmatrix}, \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = A \cdot \begin{pmatrix} \alpha_y \\ \beta_y \\ \Delta Y \end{pmatrix}, \tag{2.4}$$

kde A je

$$A = \begin{pmatrix} X_1' & Y_1' & 1 \\ X_2' & Y_2' & 1 \\ X_3' & Y_3' & 1 \end{pmatrix}. \tag{2.5}$$

Vynásobením matice A inverzní maticí A^{-1} zleva jsou vyjádřeny žádané koeficienty. Výpočet inverzní matice je proveden pomocí determinantu matice A a algebraických doplňků A^*.
Implementace algoritmu je popsána v kapitole 3.1. Další informace je možné nalézt v [4], [5].



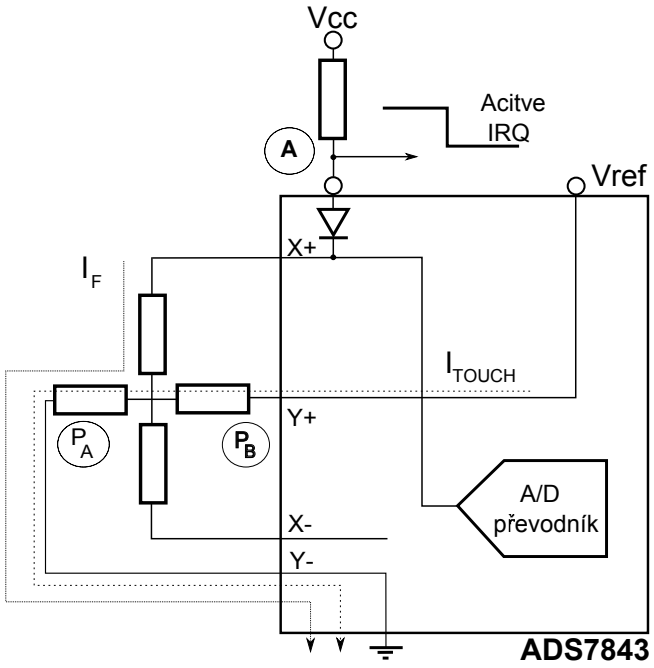
(a) Rozložení tří nezávislých bodů

(b) Rozložení tří závislých bodů

Obr. 2.5: Zobrazení závislých a nezávislých bodů na displeji [4]

Signalizace stisknutí displeje

ADS7843 je navržen tak, aby byl schopen jednoduše signalizovat stisknutí displeje. Na obrázku 2.6 je možné vidět vnitřní schéma *ADS7843*, které je tvořeno AD převodníkem, odporovou sítí a diodou. Ve stavu, kdy není displej stisknut, je dioda zavřená, neteče skrze ní elektrický proud a v bodě A je úroveň napětí logická 1. V případě, je-li displej stisknut, je dioda polarizována v propustném směru a proud I_F teče rezistory R_{X+}, R_{Y-}. Úroveň napětí v bodě A se změní na logická 0, což zhruba odpovídá prahovému napětí diody.
Tento typ signalizace ovšem není zcela optimální. Jak vyplývá z vnitřního schématu, během digitalizace signálu se sčítá proud diodou I_F a proud z dotykového displeje I_{TOUCH}. Vzniká tzv. offset, jehož hodnota není konstantní. Čím více se blíží místo stisknutí k bodu P_A, tím větší teče proud I_F. Důsledkem je nárůst potenciálu napětí na vstupu AD převodníku. V opačném případě (stisk blíže bodu P_B) offset klesá vlivem nárůstu kladného napětí na katodě diody. Touto chybou je zatíženo pouze měření *Y-ové* složky, neboť pro *X-ovou* složku souřadnic je dioda závěrně polarizována.



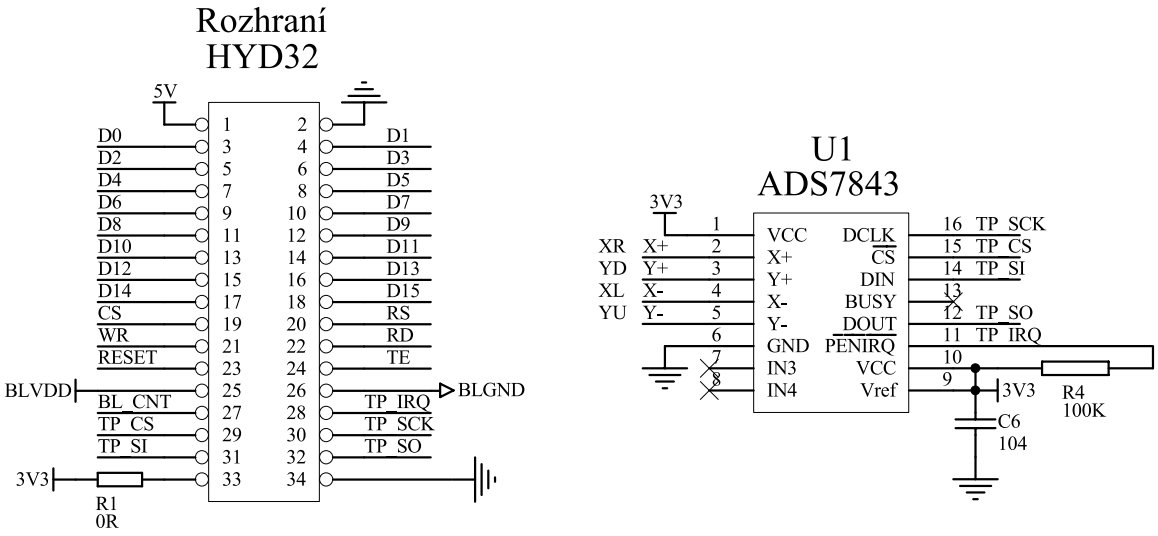
Obr. 2.6: Vnitřní schéma převodníku *ADS7843* [6]

HY32D

Jedná se o zařízení, které v sobě integruje $3,2''$ dotykový displej, řadič *SSD1289*, AD převodník *ADS7843*, dva *LDO* stabilizátory *CAT6219*. Schéma zapojení *HY32D* bylo rozděleno na dvě části. Na obrázku 2.7 je zobrazena 34 pinová patice, kterou lze rozdělit na napájení(VCC), 16 bitovou komunikační sběrnici řadiče ($D0 - D15$), řídící signály řadiče ($CS, RS, WR, RD, RESET$), SPI komunikaci s AD převodníkem a podsvícení displeje BL_CNT.
$HY32D$ je možné napájet napětím 5 V, které je stabilizováno na $3,3$ V a rozvedeno k dalším obvodům. Zároveň lze toto napětí naměřit mezi piny $33, 34$. Druhou možností je použít napájení $3V3$, čímž odpadá funkce stabilizátoru $U3$.

AD převodník ADS7348 slouží k převodu analogového napětí z dotykového displeje na číselné vyjádření souřadnic. Piny $X+, X-, Y+, Y-$ napájí dotykovou vrstvu displeje a zároveň snímají napětí úměrné poloze prstu na displeji. Piny $TP\_SCK, \overline{TP\_CS}, TP\_SI, TP\_SO$ jsou standardní komunikační signály sběrnice SPI. Aktivní úroveň $\overline{TP\_IRQ}$ signalizuje stisknutí displeje. Jak je možné vidět ve schématu, v klidovém stavu je na tento pin přivedeno napájecí napětí $3V3$ přes odpor $R4$.
Pin $BUSY$ zabraňuje chybnému čtení výstupních dat. Po vzorkovací periodě je s první hranou hodin TP_CLK signál $BUSY$ aktivní a první datový bit není platný, viz [6]. Jelikož je tato situace velmi dobře předvídatelná, lze tento problém řešit programově. Tento pin není zapojen stejně jako kanály $IN3$ a $IN4$.
Kmitočet hodin TP_SCK udává rychlost a přesnost převodu. V datových listech jsou stanoveny mezní hodnoty TP_SCK na $f_{min} = 10$ kHz a $f_{max} = 125$ kHz. V kapitole 2.1.1

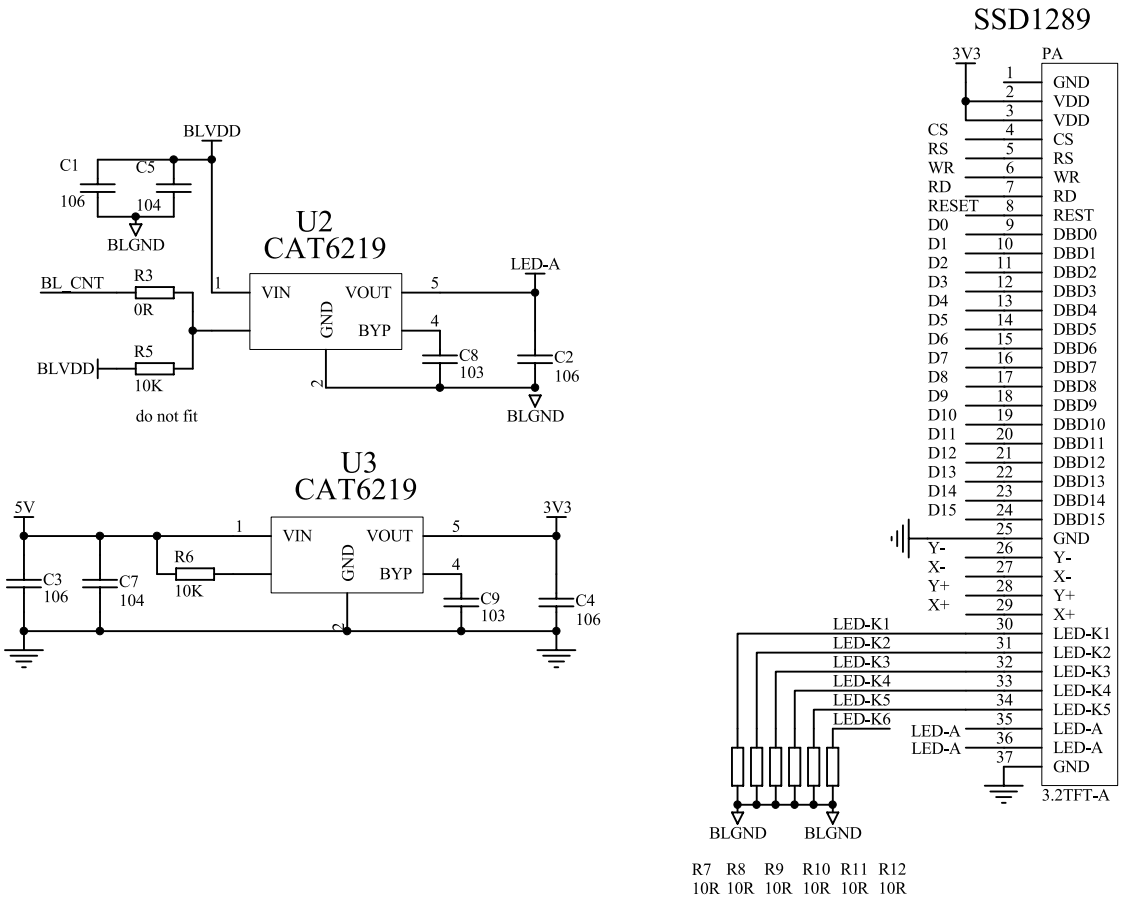
byl popsán vliv TP_SCK na vznik přechodných jevů během digitalizace signálu. Rychlost převodu dále koliduje s nastavením rozlišení AD převodníku (8/12 bitů) a režimu převodu (24/16 hodinových impulsů na převod).
Činnost stabilizátoru $U2$ je zřejmá ze schématu na obrázku 2.8. Je zde využit pin EN, který řídí úroveň výstupního napětí dle PWM signálu. Je-li na tomto pinu logická 0, je i výstupní napětí na nule. V opačném případě je na výstupu napětí $3V3$. *CAT6219M* má interní pull-down rezistor R_{EN}, který definuje stav na výstupu i při neošetřeném pinu EN.
Poslední komponentou je TFT LCD řadič $SSD1289$. Tento obvod je schopný pracovat s



Obr. 2.7: Schéma komunikačního rozhraní *HY32D* a AD převodníku *ADS7843*

displeji o rozlišení 320 řádek x 240 sloupců a barevné hloubce 16 nebo 18 bitů. Rozložení barev je v prvním případě $RGB565$, kde číslo vyjadřuje počet bitů na barvu. Druhá varianta je symetrické rozložení $RGB666$. Řadič obsahuje vnitřní $GDDRAM$ paměť pro ukládání grafických dat. Rozsah $GDDRAM$ paměti odpovídá velikosti jednoho snímku, tedy (240x320x18)/8 bytů. $GDDRAM$ Paměť komunikuje s mikrokontrolérem skrze rozhraní RGB, které umí využívat buď sériový protokol SPI nebo paralelní protokoly 6800/8080 a to s různou šířkou datové sběrnice $8-/9-/16-/18$ bitů.
Řadič je navržen tak, aby byl schopen fungovat s minimem externích součástek. Proto je na čipu umístěn $DC-DC$ měnič, který vytváří potřebné úrovně napájecího napětí. Dále je zde programově řiditelná jednotka pro korekci jasu, budiče obrazového řádku (*gate driver*) a datové budiče obrazového sloupce (*source driver*). Časování budičů, stejně tak jako čítače adresy AC, je řízeno z RC oscilátoru.
Komunikace mezi mikrokontrolérem a řadičem probíhá na bázi zápisu/čtení *Index & Status registr(IR)*. Index lze chápat jako offset od 0, kterým lze adresovat všech 50 řídících registrů, tak i $GDDRAM$. Je-li zvolen zápis do $GDDRAM$, jsou grafická data v závislosti na použité barvené hloubce převedena na 18 bitovou šířku. Tzn. že pro 16 bitovou barevnou hloubku jsou MSB bity modré a červené barvy zkopírovány na pozici LSB. Zelená barva je zachována.

Poté jsou data uložena do dočasného registru $GRAM$ a následně do interní paměti. Čtení z $GDDRAM$ je obdobou zápisu, kde pro získání dat je nejprve využit dočasný registr $GRAM$. Tento registr není po každém četní resetován na nulovou hodnotu. Proto jsou platná data získána až při druhém cyklu čtení.
Následující kapitola bude věnována bližšímu popisu rozhraní 8080/16bit, jež je továrním nastavením modulu $HY32D$.



Obr. 2.8: Schéma napájení *HY32D* a řadiče *SSD1289*

Komunikační rozhraní 8080

Jedná se o jednoduché paralelní rozhraní složené ze 16 bitové datové sběrnice $D[0-D15]$ a řídících signálů $\overline{RD}$, $\overline{WR}$, $D/\overline{C}$, $\overline{CS}$. Signál $\overline{RD}$ slouží jako hodinový signál klopného obvodu řízeného hladinou (*latch*). Je-li zároveň $\overline{CS}$ v aktivní úrovni, jsou data čtena z $GDDRAM$ nebo ze *status* registru dle $D/\overline{C}$ (Data/Command). Jak bylo popsáno výše, v prvním cyklu čtení $GDDRAM$ jsou vystaveny neplatná data.
Zápis dat do *status* registru nebo do $GDDRAM$ probíhá obdobně. Nejprve je zvolen cíl zápisu $D/\overline{C}$, poté jsou nastaveny aktivní úrovně $\overline{CS}$, $\overline{WR}$ a nakonec vystaveny data. Bližší informace o časování signálů lze nalézt v datových listech řadiče [7].

Spojení HY32D s řídícím modulem

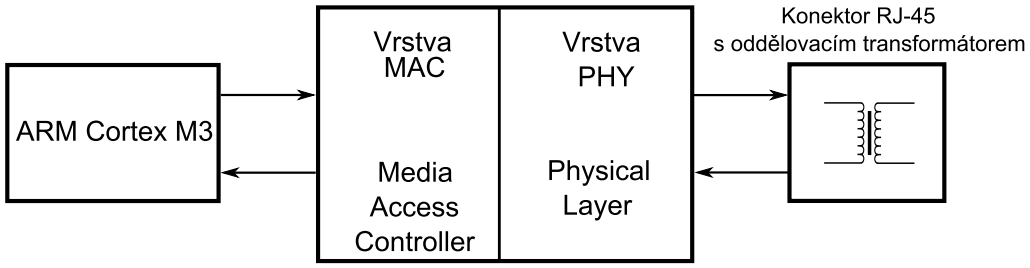
V příloze A se nachází schéma zapojení $HY32D$ s řídicím modulem. Datová sběrnice byla rozdělena na dvě brány mikrokontroléru P_J, P_H a to tak, aby mohla být data okamžitě přiřazena pinům bez dalšího zpracování, např. maskování či posunů. Bit s nejnižší vahou LSB připadá pinu 0, MSB pinu 7. Tento způsob vede na obtížnější návrh plošného spoje, avšak na efektivnější komunikaci s displejem.
Během návrhu řídícího modulu nebyl kladen požadavek na čtení dat z $GDDRAM$ ani případnou kompatibilitu s jinými grafickými moduly. Proto byl pin $\overline{RD}$ připojen trvale na napětí $3V3$.

2.1.2 Ethernet Controller

Tato kapitola se bude zabývat hardwarovou konfigurací Ethernet na desce řídícího modulu. Nejprve bude vysvětlen základní princip činnosti Ethernet Controller, který tvoří rozhraní mezi firmware a fyzickým přístupem na sběrnici. Dále pak bude uveden seznam signálů a jejich doporučené nastavení dle datových listů. Poslední část bude věnována konkrétnímu zapojení Ethernet.
Stellaris Ethernet Controller integruje vrstvy Media Access Controller (MAC) a Network Physical (PHY). Ethernet Controller splňuje specifikaci IEEE 802.3 a podporuje standardy 10BaseT, 100BaseT. Pro přístup na sběrnici je vyžadován pouze izolační transformátor s převodem 1:1. Následuje stručný výčet vlastností Ethernet Controller.

- Pracovní režimy
  - Plný a poloviční duplex 100 Mbit/s
  - Plný a poloviční duplex 10 Mbit/s
- Uživatelsky programovatelné
  - MAC adresa
  - aktivita LED
  - Promiskuitní režim
  - Zapnutí/vypnutí generování CRC
- Podpora fyzické vrstvy
  - MDI/MDI-X podpora pro křížené/přímé kabely
  - Automatická korekce polarity signálu

Na obrázku 2.9 je zobrazeno blokové schéma Ethernet Controller. Účelem vrstvy MAC je zpracování příchozích a odchozích rámců. Zároveň tvoří skrze vnitřní vrstvu Media Independent Interface (MII) rozhraní pro PHY.



Obr. 2.9: Blokové schéma *Ethernet Controller* mikrokontroléru *LM3S9B92* [1]

Vrstva MAC

Na vrstvě MAC jsou implementovány dva paměťové bloky TX FIFO a RX FIFO. Oba o velikosti 2 KB. Pro odchozí komunikaci je možné do TX FIFO uložit vždy pouze jeden rámec. Dle specifikace IEEE 802.3 je maximální velikost datového pole *DF* ethernetového rámce 1500 Byte. Ethernet Controller tento limit neuvažuje a lze využít plného rozsahu TX FIFO, tedy 2032 Byte. Prvních 16 Byte v paměti je vyhrazeno pro adresu cíle, adresu zdroje a informačního pole *length/type*.
Příchozí komunikace je obsloužena pomocí RX FIFO. V tomto paměťovém bloku může být uloženo najednou až 31 rámců. Je-li přijat rámec, pro který není ve FIFO dostatečné místo, je generován příznak přetečení (*overflow error*) v příslušném stavovém registru.
Detailní pohled na organizaci obou paměťových bloků lze nalézt va datových listech *LM3S9B92* [1]. Jediný mezi RX FIFO a TX FIFO je v poli *Data lenght*. Pro TX FIFO zahrnuje toto pole pouze délku užitečných dat, tedy sekvence 5 až n. V bloku RX FIFO je započítána celková velikost rámce.

Na vysílací straně vrstva MAC obstarává generování *FCS* kontrolního součtu. Pro účely testování neplatných rámců může být tato funkce neaktivní. V zájmu rychlejší orientace je uvedena struktura ethernetového rámce na obrázku 2.10. Z důvodu detekce kolize je ve specifikaci IEEE 802.3 stanovena minimální délka datového pole *DF* ethernetového rámce na 46 Bytů. Je-li velikost dat menších než 46 Bytů, je vrstvou MAC přidána výplň tzv. PAD pro dosažení minimální délky rámce.

Preamble	SFD	Destination Adress (DA)	Source Address (SA)	Length/ Type	Data (DF)	FCS

Obr. 2.10: Formát rámce *Ethernet IEEE 802.3*

Na straně příjmu jsou v normálním pracovním režimu akceptovány pouze rámce, u kterých se shoduje adresa cíle *DA* s uživatelsky nastavenou MAC adresou v registrech *MACIA0* a *MACIA1*. Ethernet Controller je možné také nastavit do promiskuitního režimu nebo do režimu Multicast. Vrstva MAC dále kontroluje pole *FCS*, případně zahazuje rámce se špatným kontrolním součtem.

Určitou vizuální kontrolu stavu linky a komunikace nabízí signalizační LED diody, obvykle umístěné na konektoru RJ-45. Jejich aktivitu lze programově nastavit do různých režimů. Podporovány jsou:

- Stav linky
- Aktivita RX, TX
- Standard 100BaseT
- Standard 10BaseT
- Plný duplex
- Stav linky a aktivita RX, nebo TX

Programová obsluha FIFO pamětí

Programový přístup do FIFO pamětí je vykonán pomocí datových registrů *TX DATA, RX DATA*. Je-li čteno z *RX DATA*, jsou získána data na pozici ukazatele. Ten je poté automaticky inkrementován. Čtení z registru v době, kdy není přijat nový rámec, nebo není doposud zpracován, vede čtení nespecifikovaných dat.
Symetrickou situací je zápis do registru TX FIFO. Zapíše-li se větší objem dat než je indikován v poli *Data lenght*, dojde ke ztrátě informací. Zápis menších dat způsobí připojení náhodné informace za konec rámce. Oba registry nepovolují náhodný přístup do paměti a pracují pouze se sekvenčním přístupem dat [1].

Vrstva PHY

Fyzická vrstva zabezpečuje přístup na sběrnici prostřednictvím vrstvy MAC. Jsou zde zahrnuty filtrační, tvarovací obvody a adaptivní ekvalizéry pro regeneraci přijatého signálu, případně úpravu vysílaného signálu. Taktéž je zde omezeno vyzařované spektrum. V této části budou shrnuty protokoly $Auto-Negotation$ a MDI/MDI_x, které se podílí na automatické inicializaci fyzické vstvy PHY.

Protokol $Auto-Negotation$ umožňuje jednotlivým zařízením, připojeným ke komunikační lince vzájemnou výměnu informací o možnostech komunikace a výběru optimálního módu, který jsou schopny obě strany provozovat. Stav linky a komunikační rychlost je uložena v tzv. *Ethernet PHY Management* registrech. Stejně tak je zde uložen příznak dokončení $Auto-Negotation$. Tento protokol je záležitostí výhradně fyzické vrstvy a nevyžaduje kontrolu vyšších vrstev.

Podpora *MDI/MDIX* definovaná v IEEE 802.3-2002 eliminuje nutnost používání křížených kabelů mezi dvěma koncovými zařízeními. Po nastavení a povolení Ethernet Controller je kontrolován stav linky příznakem *LINK* v *Ethernet PHY Management* registru *MR1* přibližně každou 1 s. Není-li bit aktivní, je nastaven příznak *EN* v registru *MDIX* a zároveň prohozena funkce vysílací a přijímací linky fyzické vrstvy. Opět se kontroluje stav příznaku *LINK*. Je-li stále spojení neaktivní je příznak *EN* nulován. Takto je monitorován stav linky, dokud není vytvořeno aktivní spojení.

Popis signálů

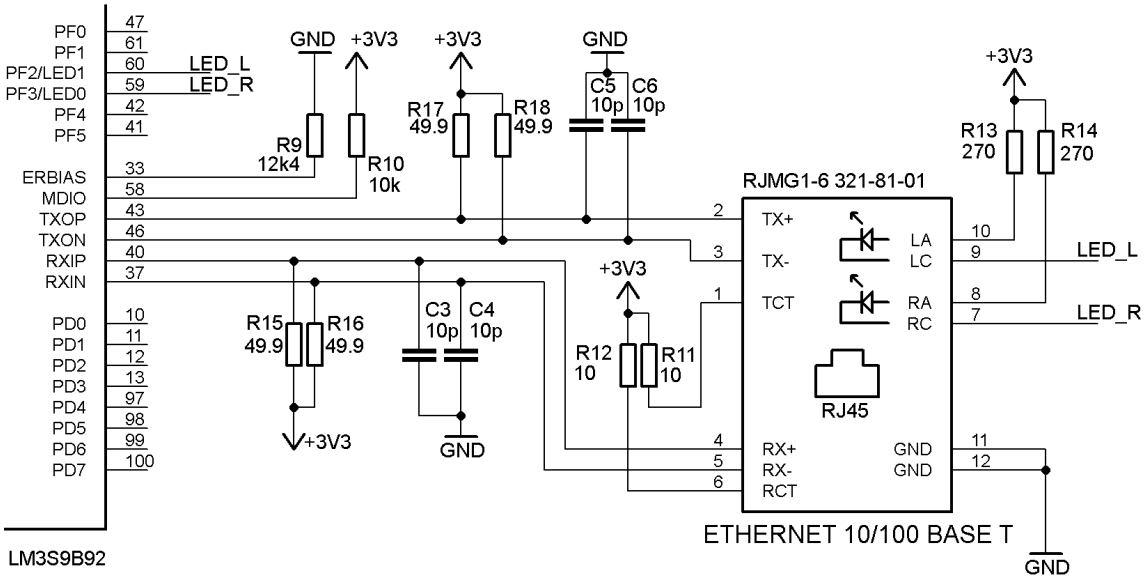
Tato kapitola se bude zabývat vstupními/výstupními signály Ethernet rozhraní $LM3S9B92$, které jsou seřazeny v tabulce 2.1. Sloupec *Alternativní funkce pinu* deklaruje nastavení bitů v registru *Alternative function registr (AFSEL)*. Položka fixní znamená, že funkce daného pinu mikrokontroléru nelze měnit. Sloupec *Popis* obsahuje stručný popis signálů.

Název pinu	**Alt. funkce pinu**	**Popis**
ERBIAS	fixní	Připojen na GND přes R 12,5 kΩ/1%, Nastavení předpětí pro *PHY*
LED0	$PF3$	Indikační LED0
LED1	$PF2$	Indikační LED1
MDIO	fixní	Připojen na 3V3 přes R 10 k, zabezpečení korektní funkce MII Management
RXIN	fixní	Diferenční pár RX
RXIP	fixní	Diferenční pár RX
TXON	fixní	Diferenční pár TX
TXOP	fixní	Diferenční pár TX
XTALNPHY	fixní	Připojení krystalu 25 MHz, je-li použit externí zdroj hodinového signálu, je tento pin nezapojen
XTALPPHY	fixní	Připojení krystalu 25 MHz, nebo externího hodinového signálu

Tab. 2.1: Tabulka vstupních/výstupních signálů Ethernet rozhraní $LM3S9B92$

Schéma zapojení Ethernet na desce řídícího modulu

V datových listech mikrokontroléru *LM3S9B92* [1] je uvedeno doporučené schéma zapojení Ethernet Controller. Zároveň jsou zde vypsány konkretní typy testovaných izolačních transformátorů a konektorů *RJ-45*.
Na obrázku 2.11 je zobrazeno schéma zapojení. Vysílaná data jsou generována *PHY* na vodičích TXOP, TXON. Tento signál budí oddělovací transformátor T1 s převodem 1:1,



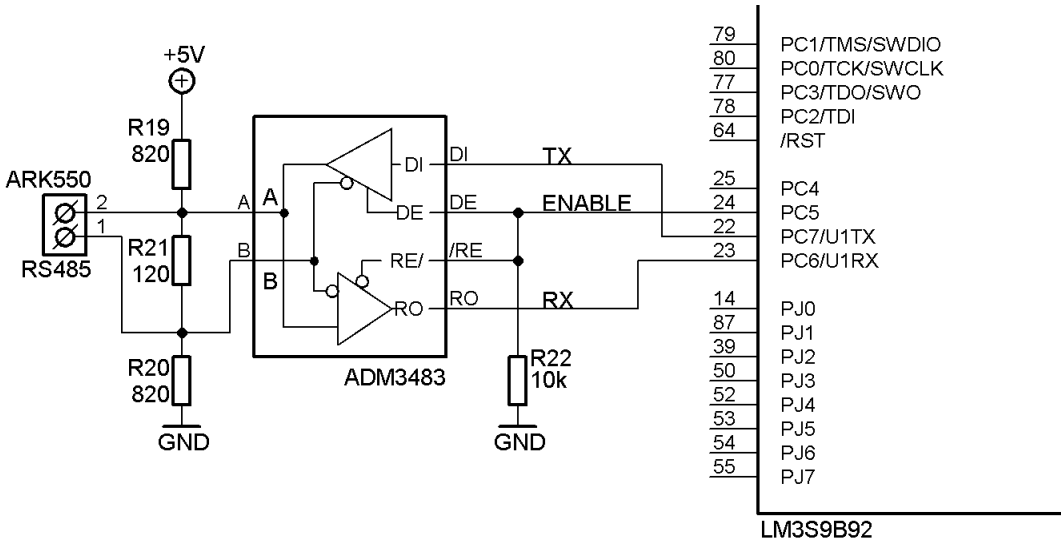
Obr. 2.11: Schéma zapojení Ethernet

který je integrovaný v konektoru RJMG1. Střed primárního vinutí T1 je připojen přes rezistory $R11$, $R12$ na napětí 3V3. Podle napěťové úrovně výstupů TXOP, TXON protéká budící proud příslušnou polovinou vinutí transformátoru T1. Na sekundárním vinutí je umístěna tlumivka s proudovou kompenzací. Jejím účelem je potlačení souhlasného napětí. Tato tlumivka je umístěna v šasi konektoru. Rezistory $R17$, $R18$ tvoří impedanční přizpůsobení vysílacího vedení. Přijímací strana funguje naprosto stejným způsobem.
LED diody *LED0, LED1* signalizují stav linky dle nastavení MAC. Anody diod jsou připojeny na napětí 3V3. Spínány jsou nízkou úrovní pinů *PF2, PF3*. Význam a připojení pinů ERBIAS, MDIO byl osvětlen v tabulce 2.1.

2.1.3 RS-485

Druhým komunikačním prostředkem řídícího modulu je sběrnice *RS-485*. Z důvodu absence budiče napěťových hladin *RS-485* na čipu mikrokontroléru, byl použit externí integrovaný obvod *ADM3483*.
Jedná se o zařízení pracující v režimu poloviního duplexu s omezenou přenosovou rychlostí $v_p = 250$ kbit/s. Tím jsou sníženy nároky na kvalitu impedančního přizpůsobení kabelů a také na hladinu elektromagnetického vyzařování *EMI*. Tento obvod v sobě zahrnuje tepelnou ochranu, která po dosažení určité teploty (např. při zkratu) přepne výstupy A, B do stavy vysoké impedance. V případě nezapojení vstupních pinů, obsahuje přijímač tvz. *fail-safe* obvod, který na výstupu nastaví úroveň logické 1. Na obrázku 2.12 je schéma zapojení *ADM3483*. Příjem/vysílaní dat je řízeno napěťovou úrovní signálu *Enable*. V logické 1 je aktivován budič DE a na výstupu je nastaven stav dle pinu DI. V opačném případě $Enable = 0$ je aktivní budič $\overline{RE}$ a dle rozdílu potenciálů je na vodičích A, B vygenerován výstup RO, vysílací výstupy jsou uvedeny do vysoké impedance.

Logické stavy *ADM3483* jsou uvedeny v tabulkách 2.2, 2.3. Rezistory $R19$, $R20$, $R21$ nastavují definovanou úroveň napětí na sběrnici v případě, že žádné koncové zařízení nevysílá. Napěťový rozdíl $|U_A\text{-}U_B|$ musí být vždy větší než $0,2$.
Na straně mikrokontroléru je pinům $PC6$, $PC7$ přiřazena periferie UART1. Struktura datového paketu je definována *Start* bitem, počtem *Data* bitů, povolením *Parity*, počtem *Stop* bitů.



Obr. 2.12: Schéma zapojení obvodu *ADM3483*

Vysílací vstup		Vysílací výstup	
Enable	**DI**	**B**	**A**
1	1	0	1
1	0	1	0
0	X	High - Z	High - Z

Tab. 2.2: Pravdivostní tabulka *ADM3483*, vysílání signálu

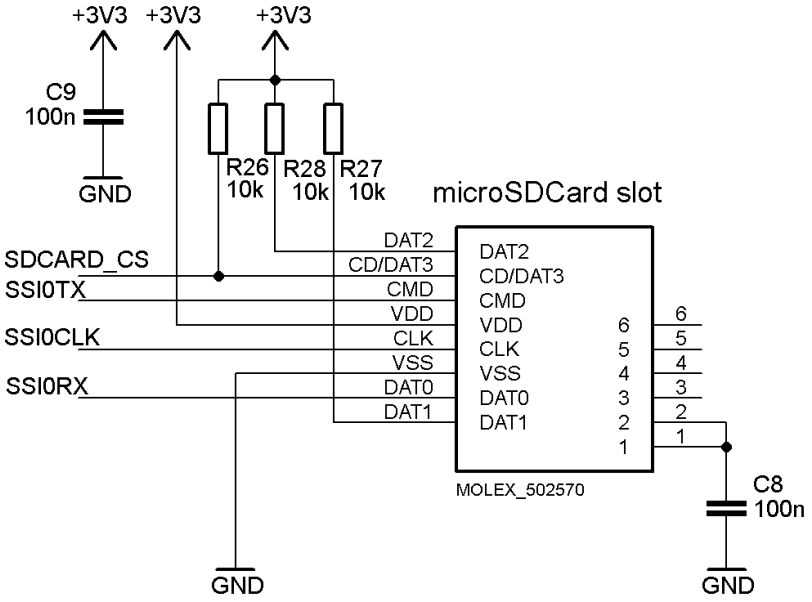
Přijímací vstup		Přijímací výstup
Enable	**A-B**	**RO**
0	$\geq +0,2V$	1
0	$\leq -0,2V$	0
0	plovoucí pin	1

Tab. 2.3: Pravdivostní tabulka *ADM3483*, příjem signálu

2.1.4 SD karta

Práce s grafickým displejem klade velké nároky na paměť zařízení. Na počátku návrhu řídícího modulu byl kladen požadavek na využití externí paměti pro ukládání fontů, případně bitmapových obrázků. Z tohoto důvodu byla vybrána SD karta. V kapitole 3.3 je probrána implementace souborového systému FAT, která usnadňuje výměnu dat mezi mikrokontrolérem a PC.

Na obrázku 2.13 je zobrazeno schéma zapojení SD konektoru. Signály *SDCARD_CS, SSIOTX, SSIOCLK, SSIORX* tvoří sériové rozhraní *SPI* mikrokontroléru, které je umístěno na bráně *PE[0-3]*. Nepoužité piny *DAT1, DAT2* jsou připojeny na napájecí napětí 3, 3 V, tak aby nedocházelo k rušení vnitřního řadiče karty. Jelikož je konektor vystaven při manipulaci elektrostatickému výboji, je zde umístěn kondenzátor $C8$, který pohlcuje nebezpečný náboj. Pracovní proud SD karty může dosáhnout až 100 mA po dobu jedné sekundy. Proto by měl být blokující kondenzátor $C9$ umístěn co nejblíže napájení.



Obr. 2.13: Schéma zapojení konektoru SD karty

2.1.5 Napájení řídícího modulu

Návrh řídícího modulu byl koncipován tak, aby byl schopen pracovat v rozsahu vstupního napětí U_{IN} $12 - 24$ V. Proudový odběr byl určen z předpokládané spotřeby jednotlivých periferií na hodnotu 1 A. S ohledem na omezené rozměry plošného spoje a s požadavkem na nízkou výkonovou spotřebu byla vybrána kombinace DC-DC měniče *LM22672* a *LDO* stabilizátoru *LM1117*. Na konci této kapitoly je uvedena tabulka 2.4, která zobrazuje spotřebu modulu v závisloti na jeho konfiguraci.

LM22672

Na obrázku 2.14 je schéma zapojení měniče *LM22672*. Je-li na kladnou svorku přiložen záporný pól U_{IN}, je dioda $D1$ polarizována v závěrném směru a obvod je chráněn proti přepólování. V opačném případě vzniká na diodě úbytek napětí a diodou teče napájecí proud. Z tohoto důvodu je navržena na proud 1 A.
LM22672 je zapojen dle datových listů [8] s fixním výstupním napětím 5 V. Kondenzátory $C12$ a $C33$ tvoří vstupní filtr měniče. Při sepnutí vnitřního *MOSFET* tranzistoru dochází na jeho vstupu ke vzniku přechodného jevu. Jeho potlačení je nejlépe dosaženo paralelní kombinací elektrolytického kondenzátoru a kondenzátoru s nízkým ekvivalentním odporem *ESR*.

Cívka $L1$ tvoří akumulační prvek měniče. Její parametry ovlivňují kvalitu výstupního napětí. Vstupními požadavky L1 jsou:

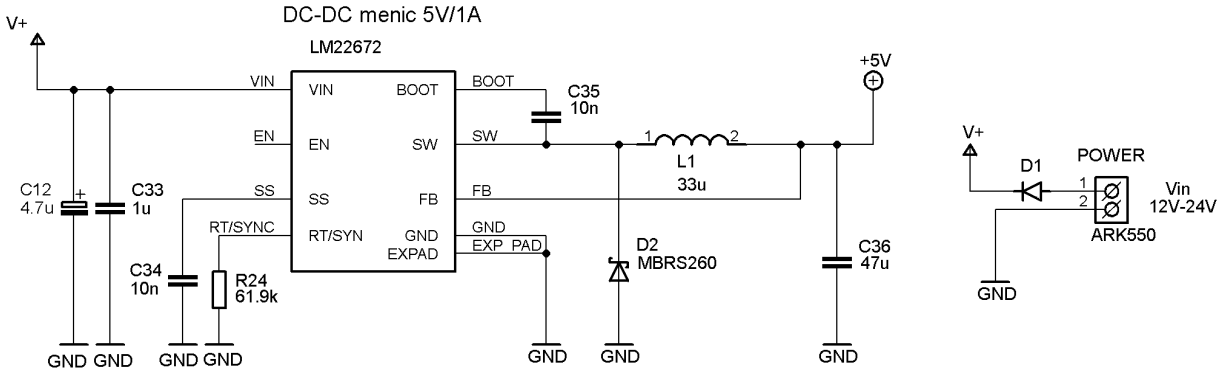
- Indukčnost cívky *L1*, určena z rovnice 2.6
- Celkový zatěžovací proud I_{0MAX}. Pro měnič *LM22672* pracující ve spojitém režimu je $I_{0MAX} = 1,8$ A
- Kmitočet oscilátoru f_{OSC}. Ten je nastaven odporem $R24$ na 500 kHz.

Velikost indukčnosti *L1* ovlivňuje zvlnění výstupního napětí a dobu odezvy cívky na kmitočet spínání. Dále by měl být vhodně zvolen saturační proud cívky, tak aby nedocházelo vlivem saturace k poklesu indukčnosti. Dle výše uvedených požadavků byla vybrána cívka *B82464G4*.

$$L = \frac{V_{OUT} \cdot (V_{IN(MAX)} - V_{OUT})}{I_{RIPPLE} \cdot f_{OSC} \cdot V_{IN(MAX)}}, \tag{2.6}$$

kde je V_{OUT} výstupní napětí měniče, $V_{IN(MAX)}$ maximální vstupní napětí měniče, I_{RIPPLE} proudové zvlnění, f_{OSC} kmitočet oscilátoru.

Dioda $D2$ odděluje vstupní napětí U_{IN} od výstupu v době rozepnutého spínacího *MOSFET* tranzistoru. S ohledem na vysoký spínací kmitočet oscilátoru a prahovém napětí diody, by měla být vždy použita Schotkyho dioda. Důležité parametry pro výběr diody jsou doba obnovy izolačních schopností *trr* (reverse recovery time), prahové napětí diody, které má vliv na efektivnost měniče a pracovní závěrné napětí, které by mělo být zvoleno min. 1, 3x vyšší než vstupní napětí U_{IN}
Kondenzátor *C36* tvoří výstupní filtr měniče a snižuje zvlnění výstupního napětí. Datové listy doporučují elektrolytický kondenzátor s velmi nízkým *ESR* nebo paralelní kombinaci elektrolytického kondenzátoru jako pomocného zdroje napětí a keramického kondenzátoru s nízkým *ESR*.



Obr. 2.14: Schéma napájení řídícího modulu

LM1117

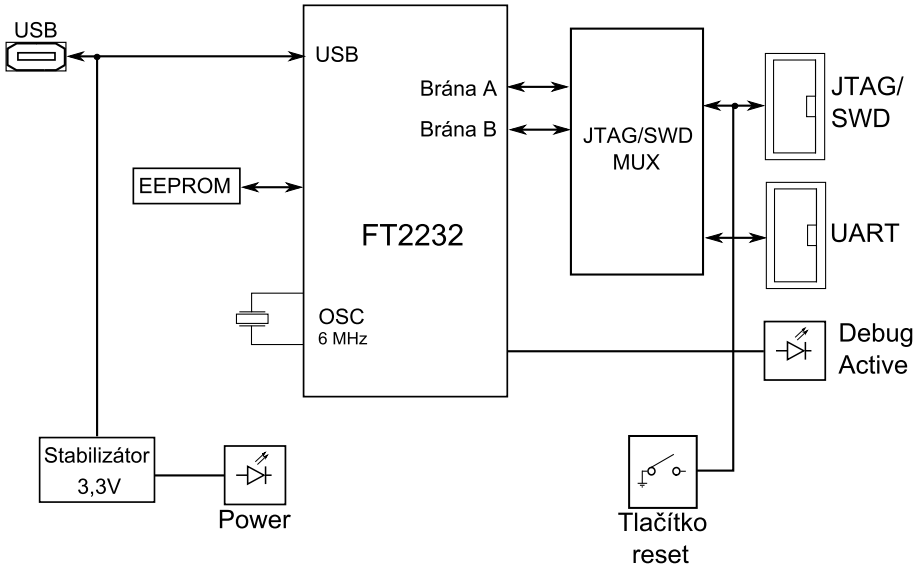
LM1117 je klasický *LDO* stabilizátor s výstupním napětím 3, 3 V a proudem 1 A, který pro svou funkci vyžaduje vstupní a výstupní filtrační kondenzátory. Hodnoty kondenzátorů byly použity dle datových listů. Ze zdroje $3V3$ jsou napájeny veškeré integrované obvody na desce řídícího modulu, kromě displeje a řadiče displeje. Z toho důvodu byla zvolena proudová rezerva stabilizátoru na 1 A.

Povolené periferie	**Odběrový proud [mA]**
Displej, Ethernet, SD karta	170
Ethernet, SD karta	104
Displej, SD karta	130
Periferie zablokovány	68

Tab. 2.4: Tabulka naměřených odběrových proudů

2.2 ICDI programátor

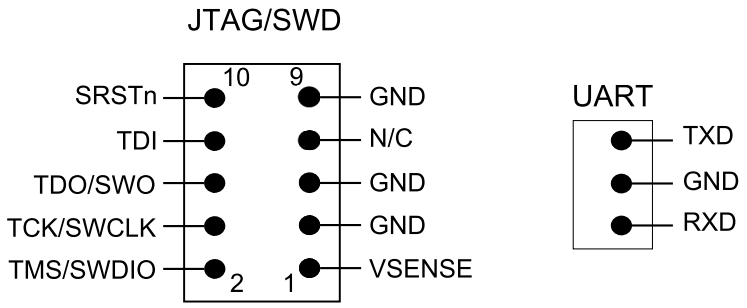
Pro účely programování a debuggovaní byl vytvořen *ICDI* (In-Circuit Debug Interface) programátor, který podporuje standardy *JTAG/SWD*. Tyto standardy nemají definovaný pinout konektoru. Proto, aby byla zaručena kompatibilita s i dalšími Stellaris mikrokontroléry, bylo využito rozložení konektoru dle [9].
Na obrázku 2.15 je blokové schéma *ICDI*. Programátor je připojen k PC pomocí *USB* miniB. To slouží jako napájení EEPROM paměti a stabilizátoru *MCP1824T*. Dále je zde použit obvod FT2232 konfigurovaný jako převodník *USB* na *JTAG/SWD* na bráně *A* a jako Virtual COM Port (*VCP*) na bráně *B*. Vnitřním signálem *SWO_EN* v multiplexeru *JTAG/SWD MUX* je přepínáno mezi periferií UART a serial wire output(SWO), pokud je daným mikrokontrolérem podporován. Dále je možné využít tlačítka reset pro uvedení zařízení do počátečního stavu.



Obr. 2.15: Blokové schéma programátoru *ICDI*

Indikace stavu *ICDI* je provedena pomocí dvou LED diod. První signalizuje připojení k napájení (USB), druhá svítí pokud je aktivní některý z bufferů uvnitř *JTAG/SWD MUX*. Schéma zapojení je uvedeno v příloze B.2.

EEPROM paměť slouží jako konfigurační banka. Na stránkách [10] poskytuje *Texas Instruments* návod na naprogramování paměti. Pro diplomovou práci byl využit firmware z komerčního výrobku, který byl pomocí programu FT Prog přenesen do vyrobeného programátoru. Další informace o *ICDI* lze nalézt v [11], [12]. Pro úplnost je na obrázku 2.16 uveden pinout použitých konektorů.



Obr. 2.16: Popis konektorů *JTAG/SWD*, *UART* programátoru *ICDI*

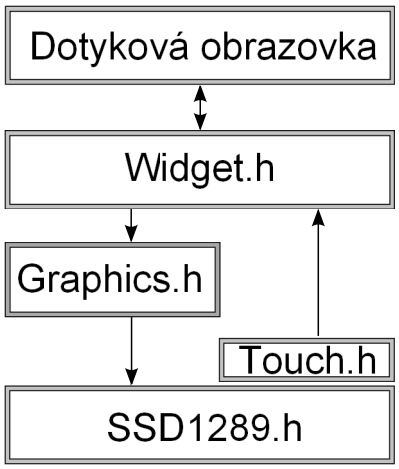
3

Firmware

Diplomová práce je zaměřena na ovládání vzdálených zařízení pomocí dotykového displeje a komunikačního rozhraní. Pro tyto periferie byly vytvořeny knihovny *Grlib*, *SDlib* a *Ethlib*. Tato kapitola bude rozdělena na části dle jednotlivých knihoven a budou zde vysvětleny obslužné funkce (*API*) pro práci s periferiemi.

3.1 Knihovna Grlib

Grlib je grafická knihovna, která v sobě obsahuje *API* pro kalibraci displeje, tvorbu základních ovládacích prvků (*widget*) a v neposlední řadě zpracovaní zpětné vazby mezi uživatelem a displejem. Při návrhu *Grlib* byl důraz kladen na to, aby knihovna byla snadno rozšiřitelná o další widgety, a aby byl programátor schopen jednoduše modifikovat grafický vzhled displeje.
Knihovna byla rozdělena do čtyř vrstev, jak je vidět na obrázku 3.1. Rozhraní mezi uživatelem a programem tvoří aplikační vrstva reprezentující dotykový displej. Na něm jsou zobrazeny ovládací prvky (z vrstvy Widget.h), skrze něž jsou ovládána vzdálená zařízení. Každý ovládací prvek je složen z elementárních bloků (čtverec, kruh, čára, text, atd. . .), které jsou zahrnuty ve vrstvě Graphics.h. Pod touto vrstvou se nachází driver SSD1289.h, komunikující přímo s displejem.
V případě stisknutí ovládacího prvku je na podvrstvě driveru Touch.h aktivován AD převodník *ADS7843* a je zahájen proces získání souřadnic. Data z vrstvy Touch.h jsou zpracována ve Widget.h a na základě vyhodnocení podmínek je buď ovládací prvek překreslen nebo ne. Zároveň zde mohou být odeslána řídící data komunikačním rozhraním *Ethernt/RS485*.



Obr. 3.1: Blokové schéma funkce grafické knihovny Grlib

SSD1289.h

SSD1289.h tvoří fyzickou vrstvu grafické knihovny. Na počátku programu je zde provedena inicializace komunikačního rozhraní 8080. Řadič displeje je resetován a signály $\overline{WR}$, $\overline{CS}$ jsou nastaveny do neaktivní úrovně. Následuje sekvenční zápis hodnot do *Index & Status registr*. Inicializace je zakončena resetováním celé interní paměti řadiče *GDDRAM* a konfigurací PWM časovače, který je využit pro řízení podsvícení displeje.

void WriteCommand (uint16_t command)

Tato funkce vykonává zápis instrukce *command* do *Index & Status registr*. Jako první je na datovou *D[0-15]* sběrnici zapsán *command*, poté je $D\overline{C}$ nastaven na logickou nulu a následně je řídícími signály $\overline{WR}$, $\overline{CS}$ zahájena komunikace a zároveň proveden zápis do registru *Index & Status registr*. Funkce nemá návratovou proměnnou.

void WriteData (uint16_t data)

Jedná se o duální funkci k *WriteCommand()*. Rozdíl je v napěťové úrovni signálu $D\overline{C}$, která je pro zápis dat aktivní v logické 1.

void SSD1289_init (void)

Inicializace řadiče SSD1289. Tato funkce sekvenčně nastaví obsah *Index & Status* registrů a resetuje interní $GDDRAM$ paměť. Zároveň je konfigurován PWM časovač pro řízení podsvětlení.

Touch.h

Vrstva *Touch.h* se stará o zpracování zpětné vazby mezi aplikační vrstvou a uživatelem. Je-li stisknut objekt na displeji, tato vrstva si vyžádá souřadnice o poloze stisku od AD převodníku. Vlivem chyb, které byly popsány v kapitole 2.1.1, nelze souřadnice okamžitě předat vyšším vrstvám. Nejprve je proveden tzv. *de-bouncing*, tedy odstranění překmitů, poté jsou souřadnice průměrovány a nakonec přepočítány pomocí kalibračních koeficientů.

Stisknutí displeje je signalizováno aktivní úrovní napětí na pinu $\overline{PENIRQ}$. Tato událost může být hlášena mikrokontroléru jako přerušení (*interrupt*). Bylo však zjištěno, že získané souřadnice mají rozptyl až ±20 pixelů od zobrazeného bodu. Druhou alternativou je konstantní vzorkování logické úrovně pinu $\overline{PENIRQ}$. V tomto případě je rozptyl v rozmezí ±3 pixely. Perioda vzorkování T_{VZ} byla stanovena na 50 ms. To umožní objektům plynule měnit svůj stav. Jinými slovy, pokud je zobrazen objekt, který má posuvník a T_{VZ} bude vyšší než 50 ms, bude posuvník "poskakovat" a pohyb po displeji nebude plynulý.

void XYcoordinate(void)

Účelem této funkce je získání souřadnic polohy stisknutí displeje. Na obrázku 3.2 je zobrazen vývojový diagram *XYcoordinate()*. Nejprve je proveden *de-bouncing* souřadnic. Zde jsou využity proměnné *temp1*, *temp2*, jejichž prostřednictvím jsou srovnávány hodnoty současného a předchozího převodu. Souřadnice se berou za platné, pokud hodnoty těchto proměnných jsou shodné.
Tento cyklus naplní pole *Xaxis*[ ], *Yaxis*[ ] třemi vzorky, ze kterých jsou následně vypočítány průměry *x,y*. Posledním krokem *XYcoordinate()* je přepočet *x,y* dle kalibračních koeficientů.

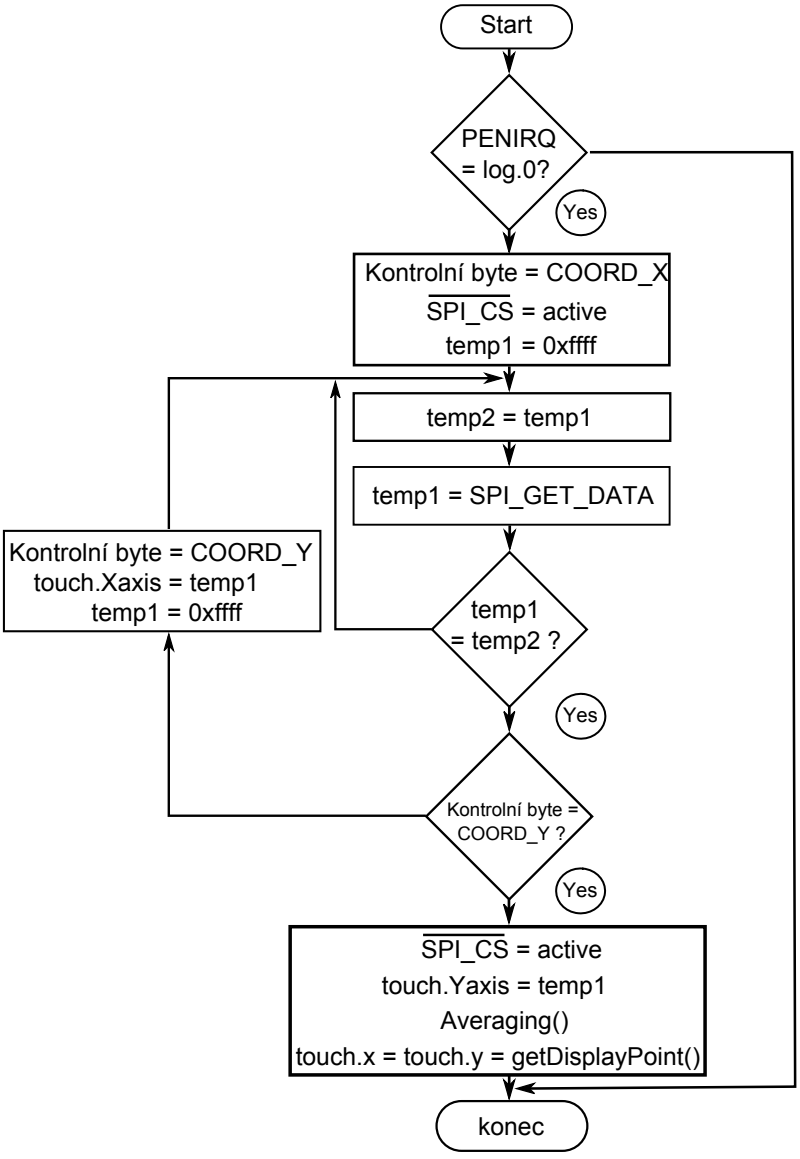
Pole vzorků *Xaxis*[ ], *Yaxis*[ ] a filtrované souřadnice jsou sloučeny do struktury *touch*, která je definována v souboru main.c jako globální proměnná.

void Averaging(void)

Tato funkce provádí průměrování nejbližších dat. Jsou vybrány všechny vzorky pole *Xaxis*[ ], vypočítány diference mezi nimi a určeny dva nejbližší prvky, ze kterých je vypočítán aritmetický průměr. To samé platí pro pole *Yaxis*[ ]. Výsledek je uložen do prvků struktury *touch.x* a *touch.y*.

boolen calibration(void)

Kalibrace se provádí postupným zobrazením třech nezávislých bodů, jejichž souřadnice jsou definovány ve struktuře *calibpoint*. Stisknutím kalibračních bodů je naplněno pole *screensample*[ ], které reprezentuje souřadnice bodů z dotykové vrstvy zatížené chybou posuvu, rotace a *scaling factor*.
Matematickou stránku výpočtu kalibračních koeficientů (popsáno v kapitole 2.1.1) zajišťuje pomocná funkce setCalibMatrix(). Výsledkem je struktura *Matrix*, kde jsou uloženy kalibrační koeficienty. Funkce *getDisplayPoint()* provádí korekci bodů *touch.x*, *touch.y*, viz obrázek 3.2.
calibration() automaticky ukládá strukturu *Matrix* do souboru Calib.bin na SD kartu. Po resetu mikrokontroléru je tento soubor načten a je okamžitě dostupný pro funkci *getDisplayPoint()*.



Obr. 3.2: Vývojový diagram XYcoordinate

Graphics.h

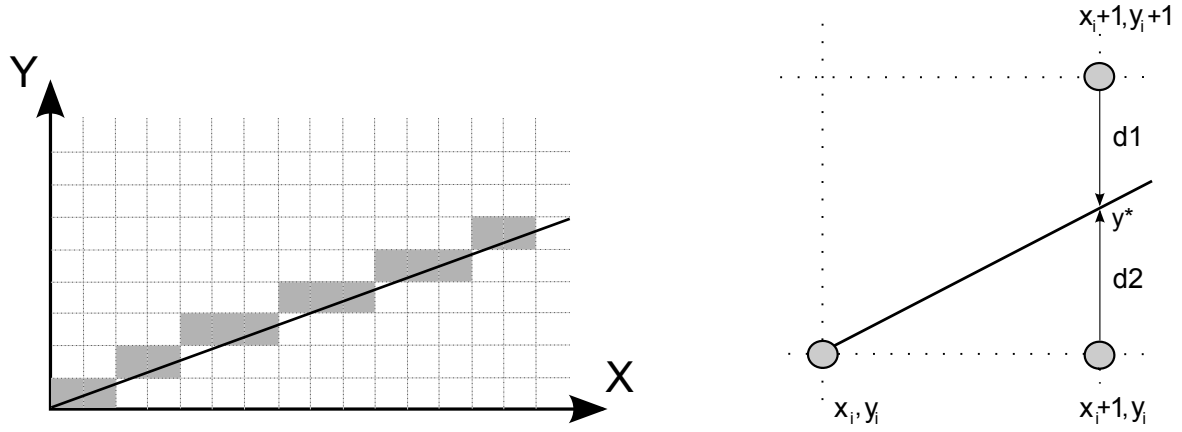
Vrstva *Graphics.h* se stará o tvorbu jednoduchých geometrických tvarů jako jsou kružnice, obdélníky, křivky atd. Jsou zde k dispozice rovněž funkce pro zobrazení samostatných znaků nebo celého textu. V této kapitole nejsou uvedeny funkce s přívlastkem *Full*, které kreslí geometrické tvary s vyplněnou vnitřní plochou, jelikož pracují na stejném principu a mají identické argumenty např. *DrawCircle()* a *DrawFullCircle()*.

void DrawPixel(uint16_t x, uint16_t y, uint16_t color)

DrawPixel() je elementární funkce pro kreslení libovolných objektů. Pixel je zobrazen na adrese dané argumenty *x,y*. Je třeba si uvědomit, že není zcela ideální vykreslovat rozsáhlé barevné objekty pixel po pixelu, ale volit raději některý z algoritmů vytvořený přímo pro konkrétní tvar.

void DrawUniLine(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)

Cílem této funkce je vykreslení úsečky, která je definována počátečním a koncovým bodem. Problém při zobrazování "nerovných" geometrických tvarů nastává vlivem nenulové velikosti pixelů. Jak je vidět na obrázku 3.3 *a)*, skutečná úsečka je aproximována přibližným průběhem. Druhým úskalím je rychlost vykreslování, která je závislá na použité aritmetice.



(a) Aproximace zobrazené úsečky

(b) Predikce následujícího pixelu dle vzdáleností rastru

Obr. 3.3: Bresenham's line algoritmus

Jednou z metod vykreslování úseček je *Bresenham's line* algoritmus. Ten funguje na principu predikce následujícího bodu ze vzdáleností skutečné přímky od dvou nejbližších bodů rastru, obrázek 3.3 *b)*.
Je-li brán v úvahu počáteční bod $[x_i, y_i]$ a směrnice k nabývající hodnot v rozsahu $< 0, 1 >$, pak může následovat pouze bod se souřadnicemi $[x_i+1, y_i]$, nebo $[x_i+1, y_i+1]$. Na základě rozdílu vzdáleností $(d_1 - d_2)$ a rovnice přímky $y = kx + c$ je odvozen parametr p_i. Jeho znaménko určuje *y-ovou* souřadnici následujícího bodu. Výhodou tohoto algoritmu je využití pouze celočíselné aritmetiky.

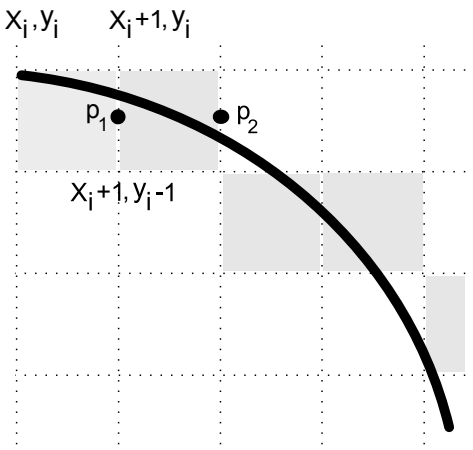
- pokud $p_i \leqq 0 \ldots y_{i+1} = y_i$,
- pokud $p_i > 0 \ldots y_{i+1} = y_{i+1}$.

boolen DrawRectangle(uint16_t Xpos, uint16_t Ypos, uint16_t width, uint16_t height, uint16_t color);

DrawRectangle je funkce pro kreslení obvodů obdélníků. Zadáním stejné velikosti argumentů *width* a *height* jsou zobrazeny čtverce.

void DrawCircle(uint16_t Xc, uint16_t Yc, uint16_t Rad, uint16_t color)

K základním geometrickým tvarům, ze kterých jsou následně tvořeny ovládací prvky, patří kružnice. Pro výpočet souřadnic bodů ležících na obvodu kružnice byl vybrán *Midpoint circle* algoritmus. Ten využívá pomocného bodu p_i pro určení pixelu, který bude následně vykreslen, obrázek 3.4.
Vychází se z obecné rovnice kružnice $g(x, y) = x^2 + y^2 + r^2$, ze které plyne, že pro body p_i uvnitř kružnice bude $g(x, y) < 0$, body vně kružnice $g(x, y) > 0$. Pokud je $g(x, y) = 0$ pixel není vykreslen. Výhodou této metody je výpočet bodů pouze pro jeden oktant, zbylé body jsou díky symetrii automaticky vykresleny pouhým přehozením znamének u složek souřadnic *x,y*.



Obr. 3.4: Midpoint circle algorithm

void DrawChar(uint16_t x, uint16_t y, unsigned char znak, unsigned char SizeOfChar, uint16_t color)

Zobrazení znaku z ASCII tabulky je možné provést pomocí funkce *DrawChar*. Znak je v paměti uložen jako bitová mapa, která je čtena řádek po řádku. Počáteční souřadnice vykreslování *x,y* je v levém horním rohu. Argument *znak* udává offset v připraveném poli font, na kterém leží daný znak. Pořadí znaků v poli je identické s pořadím v ASCII tabulce. *SizeOfChar* definuje velikost znaku.
Byly připraveny dva fonty s velikostí znaků *ASCII8x8* a *ASCII16x24*. Rozlišovat mezi nimi lze pomocí maker *Size_8x8, Size_16x24*.

boolen DrawString(uint16_t x, uint16_t y, unsigned char *text, unsigned char size, uint16_t color)

Řetězec znaků je vykreslen funkcí *DrawString*. Počáteční souřadnice řetězce *x,y* a počet znaků udává celkovou velikost zobrazeného textu. Pokud již některé znaky přesáhnou rozlišení displeje, jsou ignorovány. Pointer **text* funguje jako ukazatel na první znak textu.

Widget.h

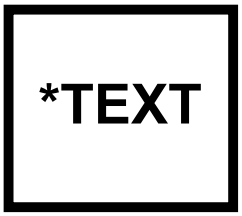
Jak bylo uvedeno na obrázku 3.1, *Widget.h* pracuje ve dvou úrovních. Směrem od aplikační vrstvy vytváří menu a ovládací prvky, které jsou součástí grafického vzhledu displeje. Směrem od fyzické vrstvy zpracovává informace z AD převodníku do formy, která je předána zpět ovládacím prvkům k vyhodnocení. Cílem této kapitoly je vysvětlit konstrukci grafických prvků, tvorbu vazeb mezi grafickými objekty a nakonec způsob zpracování souřadnic z AD převodníku.

Grafické objekty

Grafické objekty jsou definované datové typy, které v sobě zahrnují povinnou hlavičku, počáteční souřadnice, rozměry a barvu. Jako příklad je na obrázku 3.5 uveden *MenuButton* s odpovídající strukturou v tabulce 3.1.
První čtyři položky tvoří povinnou část, která je identická pro každou strukturu. Pointery *pMenuNextObject* a *pContrNextObject* fungují jako vazba s okolními objekty. Jejich obsahem jsou adresy na následující *MenuButton* a ovládací prvky. Tímto tématem se zabývá kapitola 3.1.
Identifikátor *ID* je složen z dvojice bytů. Dolních 8 bitů je vyhrazen maskám jednotlivých prvků a horním je přidělené číslo, které z každého *ID* dělá unikátní kód.
Proměnná *State* definuje aktivní nebo pasivní stav zobrazených prvků. Aby bylo možné rozlišit, které objekty jsou v dané chvíli vykresleny na displeji a přidělit jim zprávu od AD převodníku, je testován stav této proměnné.

Datový typ	Jméno	Popis
void *	pMenuNextObject	Pointer na následující MenuButton
void *	pCtrNextObject	Pointer na následující ovládací prvek
uint16_t	ID	Identifikátor objektu
uint16_t	State	Stav vykreslení prvku (ACT\PASS)
uint16_t	Xpos	x-ová počáteční souřadnice prvku
uint16_t	Ypos	y-ová počáteční souřadnice prvku
uint16_t	widht	Šírka prvku
uint16_t	height	Výška prvku
uint16_t	Xtext	x-ová počáteční souřadnice textu
uint16_t	Ytext	y-ová počáteční souřadnice textu
ColorSet_t	ColorSet	Paleta barev
uint16_t	SizeText	Velikost fontu
char *	Text	Pointer na text

Tab. 3.1: Datový typ MenuButton



Obr. 3.5: Grafický objekt MenuButton

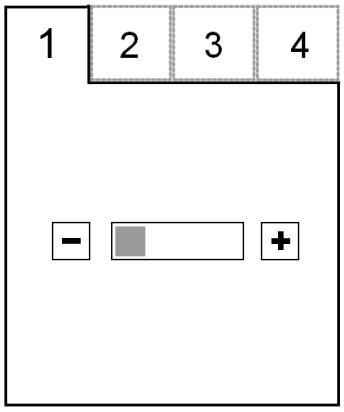
Následující položky jsou závislé na typu objektu. V tomto případě jsou to rozměry, sada barev, velikost textu a pointer na text, který je v menu prvku zobrazen.

Příklad *ID*:

- objekt 1 $ID...MENU\_MASK$ | 0x0100
- objekt 2 $ID...MENU\_MASK$ | 0x0200
  atd...

Grafické rozhraní displeje

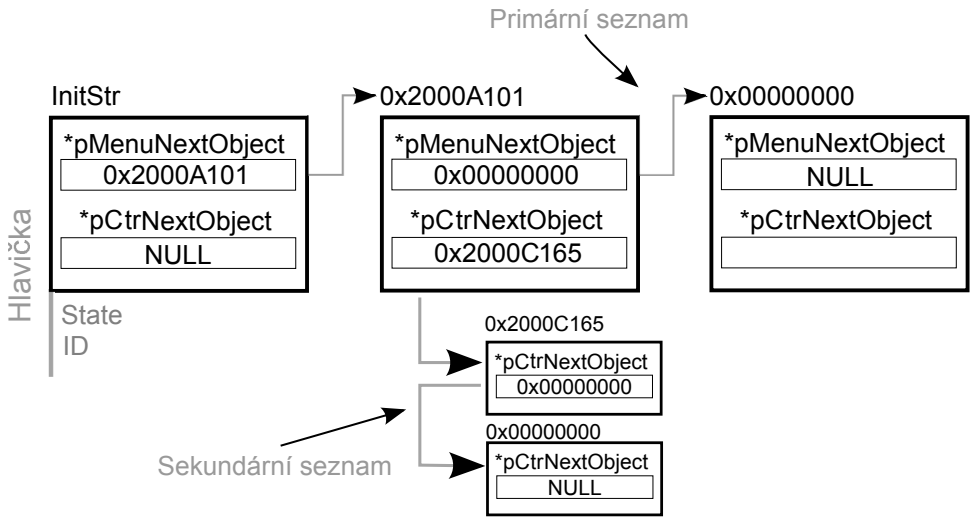
Koncepce grafického rozhraní displeje je volena tak, že horní lišta je vyplněna menu prvky a zbylá oblast přísluší ovládacím prvkům. Procházení menu poté funguje na principu otáčení listů v knize. Pod každým tlačítkem menu (*MenuButton*) jsou schovány objekty, které se po jeho stisknutí stávají aktivní a viditelné. Volbou jiného *MenuButton* se stav ovládacích prvků uloží, proměnná *State* je nastavena na *PASSIVE* a dochází k překreslení celé pracovní oblasti, obrázek 3.6.



Obr. 3.6: Grafické rozhraní displeje

Aby bylo možné spolu svázat různé objekty bez ohledu na typ a jejich funkci, byla definována hlavička, ve které jsou obsaženy pointery na typ void obsahující adresy na následující prvky. Princip funkce je založen na vázaných seznamech. Jak je vidět na obrázku 3.7, je zde jeden primární seznam, který spojuje menu prvky a množina sekundárních seznamů, jež připojuje soubor ovládacích prvků ke konkrétnímu menu prvku.
Pokud je do jakéhokoliv seznamu přidán nový objekt OBJ_{NEW}, je nejprve jednoduchým cyklem while nalezen poslední prvek OBJ_{LAST} a na pozici **pNextObject* uložena adresa OBJ_{NEW}. Posledním krokem je zapsání *NULL* do ukazatele $OBJ_{NEW} \rightarrow$**pNextObject*. Tím, že grafické rozhraní vzniká při kompilaci, není důvod řešit způsob ošetření odebrání vnitřních prvků seznamu [13].

Struktura InitStr slouží jako startovní bod menu seznamu a je do ní uložena adresa prvního prvku. Zároveň obsahuje paletu barev, kterou je možné využít při inicializaci grafických prvků. Na rozdíl od ostatních objektů, položka *State* je využita pro příznak *BLANK / NOT_BLANK*, viz níže.



Obr. 3.7: Spojení grafických prvků

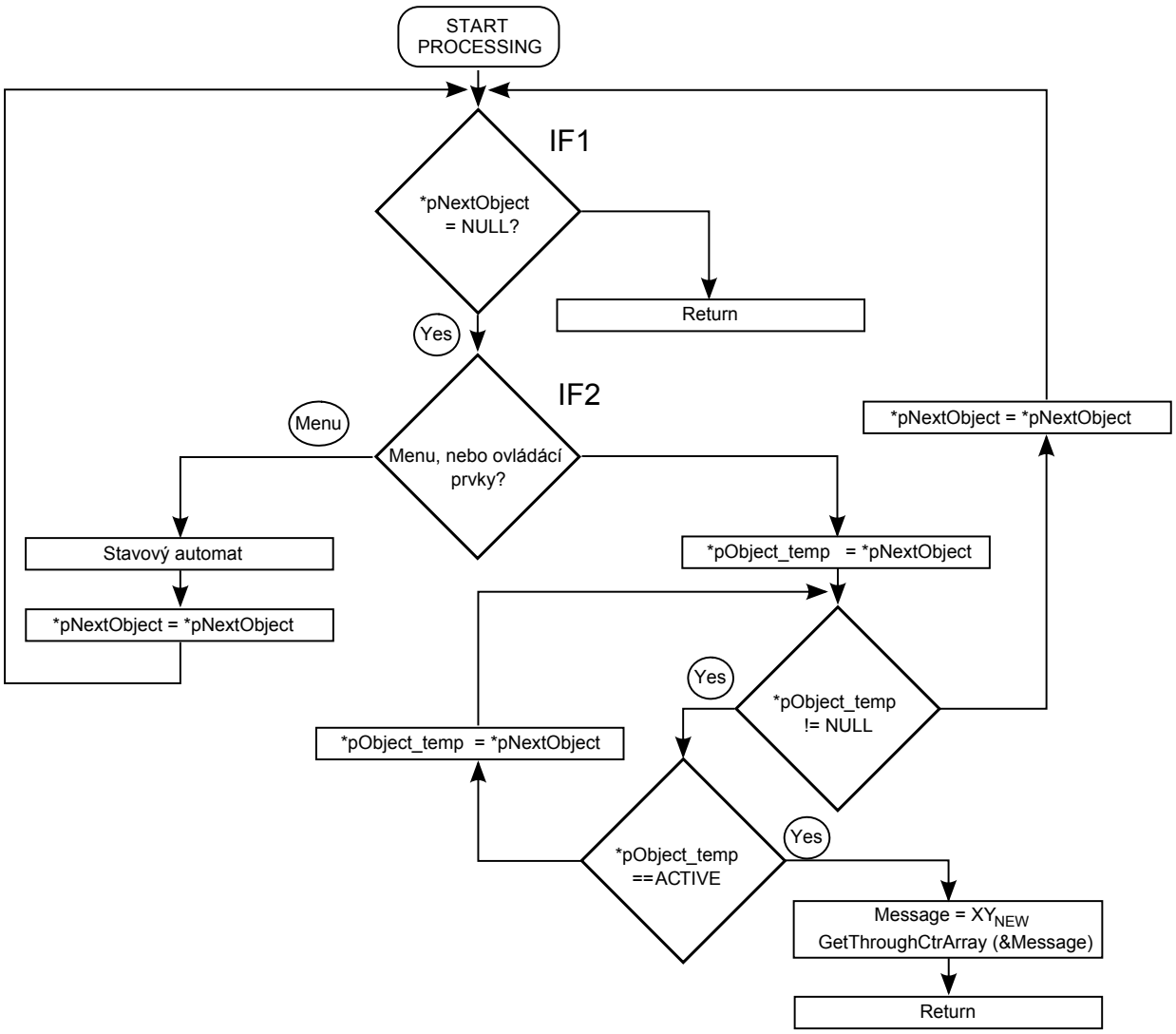
Zpracování souřadnic

Při práci s dotykovým displejem je nutné rozeznat, jaký prvek byl stisknut a co se má vykonat. Jinými slovy jde o filtraci dat a jejich předání odpovídající funkci. Na obrázku 3.8 *a)*, je uveden vývojový diagram popisující zpracování souřadnic.
Prvním krokem zpracování je kontrola stavu primárního seznamu (podm.IF_1). Není-li v něm obsažen žádný prvek, cyklus končí. V opačném případě se do dočasné proměnné uloží první prvek seznamu a porovná se jeho pozice se souřadnicemi z převodníku XY_{NEW} (podm. IF_2). Je-li podmínka splněna, nacházíme se v menu oblasti. Činnost, která bude následně vykonána, odpovídá stavu stavového automatu. Ten funguje na základě dvou příznaků. *ACTIVE/PASSIVE* definuje, jaký list je právě aktivní a vykreslen. Druhý *BLANK / NOT_BLANK* signalizuje, zda je pracovní oblast displeje prázdná.

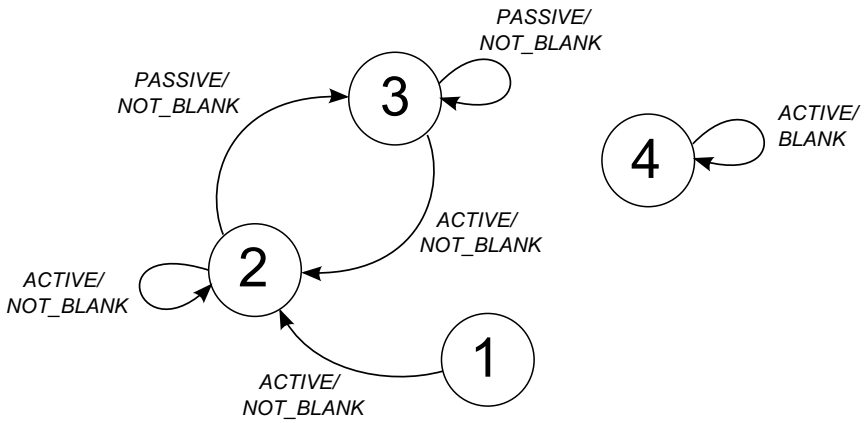
Po náběhu napájení nebo resetování řídícího modulu jsou všechny listy pasivní a není zobrazen žádný ovládací prvek. Tím je reprezentován počáteční stav automatu ***1***. Po prvním stisknutí menu jsou vykresleny všechny ovládací prvky, které k němu přísluší a zároveň nastaveny příznaky *ACTIVE*, *NOT_BLANK*. Automat přechází do stavu ***2***. Je-li opakovaně stisknut stejný menu prvek, tak automat setrvává ve stavu ***2***. Tím je zamezeno překreslování stejného listu.
Pokud je zmáčknut jiný menu prvek, dojde nejprve k vymazání pracovní oblasti displej. Poté je v pomocné proměnné *TempState* nastaven příznak minulého prvku na *PASSIVE* a současného na *ACTIVE*. Následuje vykreslení nových ovládacích prvků. Stavový automat přechází do stavu ***3***. Na obrázku 3.8 *b)* je zakreslen i nedosažitelný stav ***4***, který odpovídá prázdné obrazovce a aktivním prvkům. Tímto jsou zpracovány souřadnice XY_{NEW} v případě, že byl stisknut menu prvek.

Druhá větev podmínky IF_2 pokrývá zpracování XY_{NEW} pro ovládací prvky. Opět se jedná o smyčku, ve které se prochází primární seznam a hledá se aktivní list. Je-li nalezen, vytváří se zpráva (datový typ nazvaný *Message*), která v sobě nese souřadnice XY_{NEW}, *ID* ovládacího prvku a jeho adresu v položce *headerType *Object*.
Jelikož nelze predikovat místo stisknutí displeje a přímo tak spojit XY_{NEW} s objektem, byl zvolen přístup rozeslání *Message* všem ovládacím prvků.

V následující části této kapitoly bude uveden seznam základních funkcí pro práci s objekty na vrstvě *Widget.h*.



(a) Vývojový diagram zpracování souřadnic



(b) Stavový automat řídící vykreslování displeje

Obr. 3.8: Zpracování souřadnic XY_{NEW} na vrstvě Widget.h

void ISinit()

ISinit() provádí inicializace struktury *InitStr*. Je zde naplněn pointer na první prvek primárního seznamu *pMenuFirstObject* hodnotou NULL. Do pole *State* je uložen příznak BLANK, který definuje počáteční stav stavového automatu. Rovněž je zde definována paleta barev, která je využita při návrhu menu prvků. Volání funkce *ISinit()* musí být provedeno na začátku programu.

boolen CreateMenuObject(MenuObject *MenuButton)

Menu je tvořeno řadou tlačítek, které ve svém středu obsahují popis funkce. Založení menu prvku (*MenuButton*) je provedeno vyplněním ID, určením velikosti fontu a zadáním textu, který je automaticky zarovnán na střed. Souřadnice objektů jsou vypočítány podle celkového počtu menu prvků. Na displej je možné umístit maximálně osm prvků a to z důvodů rozlišení displeje a schopnosti stisknout prvek prstem. Jako příklad je uvedena část zdrojového kódu.

```
MenuObject1.ID = MENU_MASK | 0x10;
MenuObject1.SizeText = Size_16x24;
MenuObject1.Text = "Teplota";

CreateMenuObject(&MenuObject1);
```

boolen CreateSlider(Slider *SliderObject)

Ovládací prvek *Slider* (posuvník) patří k základním grafickým objektům. Jeho založení je provedeno stejným způsobem jako *MenuButton*. Nejprve je nutné zadat *ID* slideru, poté *ID MenuButton*, čímž je vytvořen sekundární seznam. Posledním parametrem jsou souřadnice, na kterých bude objekt umístěn. Barevný vzor slideru odpovídá nastaveným hodnotám v *InitStr*.

boolen DrawMenuObjects()

Tato funkce je volána po inicializaci všech grafických objektů. Nejprve je vypočítán celkový počet *MenuButton* v primárním seznamu. Poté je vybrán vhodný dělitel, kterým je nastaven poměr stran *MenuButton*. Posledním krokem této funkce je vykreslení celého menu.

boolen DrawAllCtrObjects(headerType* Object)

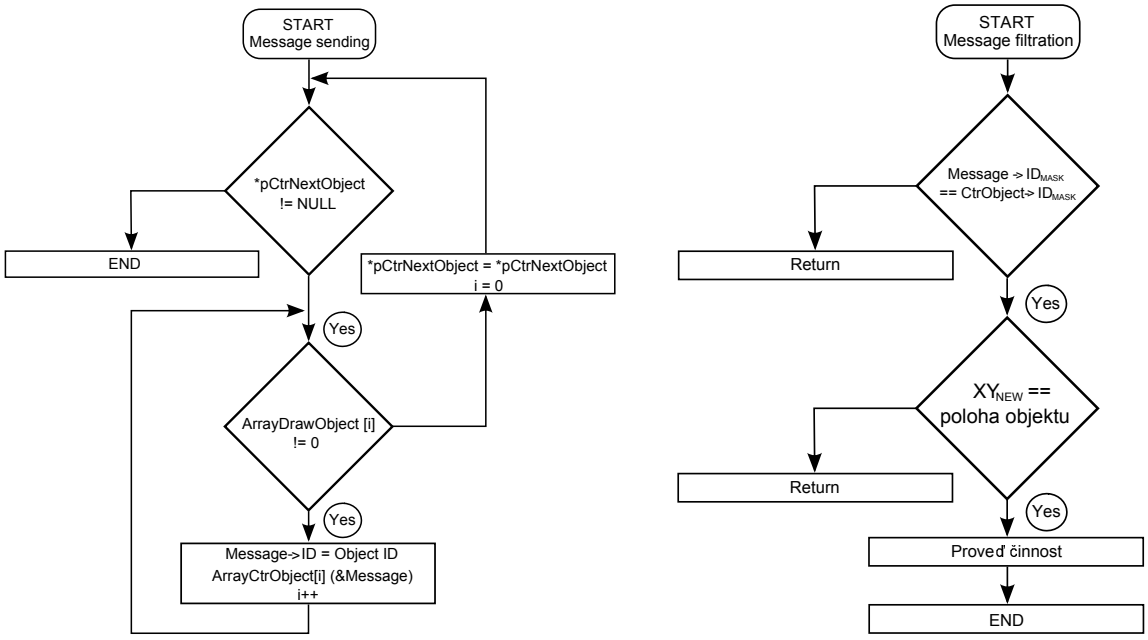
Tato funkce vykresluje sekundární seznam, na který ukazuje aktivní menu prvek. Princip je založen na smyčce while, prohledávající pole *ArrayDrawObject*[ ], ve kterém jsou uloženy hlavičky funkcí ovládacích prvků. Cyklus končí po nalezení NULL.

boolen ProcessCoordinate(uint16_t x, uint16_t y)

ProcessCoordinate() zpracovává souřadnice XY_{NEW}. Tato funkce je volána s periodou T_{VZ} a kontroluje, zda jsou dostupné nová data pomocí pinu $\overline{PENIRQ}$. Rozhoduje zda XY_{NEW} patří do oblasti menu nebo ovládacích prvků a podle toho buď mění stav stavového automatu, nebo vytváří *Message* a oslovuje ovládací prvky. Tento proces je podrobně popsán v kapitole 3.1.

boolen GetThroughCtrArray(Message* Message)

Pomocí *GetThroughCtrArray* jsou rozesílány *Message* všem ovládacím prvkům. Na obrázku 3.9 *a)* je zobrazen vývojový diagram funkce. Cyklus začíná kontrolou stavu sekundárního seznamu. Má-li podmínka kladný výsledek, přiřadí se *Object.ID* do *Message.ID* a odešle všem ovládacím prvkům v poli *ArrayCtrObject*[ ].
Na obrázku 3.9 *b)* je vývojový diagram, který popisuje filtraci *Message* na úrovni každého ovládacího prvku. Nejprve je kontrolována shoda identifikátorů ID a poté souřadnice stisknutí displeje XY_{NEW} s polohou ovládacího prvku na displeji. Jsou-li obě tyto podmínky splněny, vykoná se činnost ovládacího prvku.



(a) Odesílání Message všem kontrolním objektům

(b) Zpracování Message na úrovni každého objektu

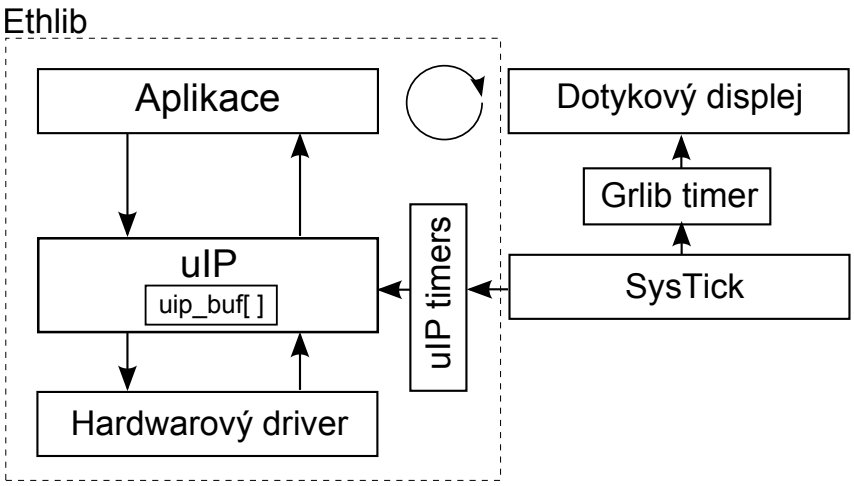
Obr. 3.9: Vývojový diagram odesílání a filtrace Message

3.2 Knihovna Ethlib

Tato kapitola se bude zabývat komunikací po Ethernetovém rozhraní, skrze níž jsou ovládána vzdálená zařízení. Knihovna *Ethlib* implementuje protokolový zásobník *uIP stack* ve verzi 1.0, který je volně dostupný na stránkách [14] a je rovněž distribuován v rámci programového balíku *StellarisWare* firmy Texas Instruments.
uIP stack je navržen s dostatečnou podporou protokolů, běžně používaných v datových sítích, tak aby mohl být použit v embedded zařízeních. Jeho součástí jsou protokoly *TCP, UDP, IP, ICMP, ARP*. Jelikož *uIP* není plnohodnotný TCP/IP stack, jsou jeho vlastnosti limitovány. Avšak pro rámec diplomové práce je dostačující.

uIP rozhraní

Na obrázku 3.10 je blokové schéma začlenění *uIP stacku* do programu. Jeho činnost je založena na obsluze událostí, které vznikají při přetečení čítače *uip_periodic*, příjmu nových dat atd. Na základě toho jsou volány odpovídající funkce. Rozhraním mezi aplikací a hardwarovým driverem je obousměrný buffer pojmenovaný *uip_buf*[ ]. V něm jsou obsaženy příchozí rámce, které jsou zpracovány na úrovni samotného stacku, jedná-li se například o ARP protokol, nebo jsou předány aplikaci. Druhou variantou jsou odchozí data z aplikace, jež odpovídají stisknutým ovládacím prvkům z dotykového displeje.
V následující části této kapitoly budou uvedeny základní vlastnosti *uIP stacku*, které je vhodné zmínit před popisem jeho implementace.



Obr. 3.10: Začlenění uIP stack do programu

Aplikační rozhraní

V rámci snížení nároků na programovou a datovou paměť mikrokontroléru, není *uIP stack* navržen v souladu s referenčním modelem ISO/OSI. Fyzickou a linkovou vrstvu obstarává hardwarový driver umístěný na chipu mikrokontroléru. Vyšším vrstvám pak musí pomoci samotná aplikace. Platí to například pro opakované odeslaní dat, kde na rozdíl od jiných TCP/IP stacků, *uIP* žádá aplikaci a opětovné vygenerování vysílaných dat.

Aplikace je v *uIP stacku* implementována jako funkce volána v případě vzniku událostí (dále označována jako call-back funkce). Její jméno musí být uloženo v makru *UIP_UDP_APPCALL*. Informace o stavu aplikace a konfigurační data přípojení (IP adr. a čísla portů) jsou specifikovány ve struktuře *uip_udp_appstate_t*. Obě tyto položky jsou součástí hlavičkového souboru *uipopt.h* (viz kapitola 3.2).
V diplomové práci byl použit pouze protokol UDP, z toho plynou i názvy funkcí a maker. Pro TCP protokol je vypuštěn ve jménech přívlastek UDP.

Obsluha událostí

Vnitřní mechanismus *uIP stacku* nastavuje příznaky dle stavu, ve kterém se nachází. Těmito příznaky se rozlišuje o jaký typ události se jedná. Na úrovni aplikace lze tak testovat hodnotu těchto příznaků a rozhodnout, jaká činnost bude aplikací vykonána.
Nutno poznamenat, že *UDP* protokol není plně podporován (což bylo zjištěno až v průběhu testování řídícího modulu), a tak nebylo možné využít kontroly spojení se vzdáleným bodem. V tabulce 3.2 je výčet nejdůležitějších příznaků.

Vnitřní příznak uIP	**Popis**
uip_newdata()	testování nových dat v *uip_buf*[ ]
uip_connected()	testování připojení vzdáleného bodu k portu
uip_connected()	testování úspěšného vytvoření spojení
uip_poll()	testování, zda je aplikace poolována
uip_closed()	testování ukončení spojení
uip_aborted()	testování přerušení spojení

Tab. 3.2: Tabulka vnitřních příznaků *uIP stack*

Příjem dat

Přítomnost dat v *uip_buf*[ ] je testována příznakem uip_newdata(). Je-li rovna 1, vzdálený uzel poslal nová data. Pomocí pointeru *uip_appdata* jsou čtena užitečná data z rámce. Jejich velikost je určena funkcí *uip_datalen()*. Jelikož je použit pouze jeden *uip_buf*[ ], musí být data okamžitě zpracována, jinak dochází k jejich přepsání.

Odesílání dat

Aplikace odesílá data pomocí funkce *uip_udp_send()*. Jejím argumentem je pouze délka vysílaných dat, avšak *uIP stack* předpokládá zarovnání pointeru *uip_appdata* na první byte vysílaných dat. Jelikož aplikace může odesílat pouze část dat v daném čase, lze volat funkci *uip_udp_send()* pouze jednou pro danou událost.

Ukončení spojení

Aplikace řádně ukončuje spojení voláním funkce *uip_close()*. Nastane-li chyba, která z nějakého důvodu brání použití *uip_close()*, je spojení přerušeno funkcí *uip_abort()*. Je-li spojení ukončeno vzdáleným uzlem, je tento stav testován příznakem *uip_closed()*.

Polling

Je-li spojení v klidovém stavu, dochází k dotazování (polling) aplikace s každým přetečením čítače *lPeriodicTimer*. Dotazování má dva účely. Prvním je informování aplikace o nečinnosti spojení. Ta následně může uzavřít spojení, pokud je neaktivní příliš dlouho. Druhým účelem je poskytnutí prostoru pro aplikaci k odeslání dat.

Nastavení spojení

Nastavení nového UDP spojení se vzdáleným bodem je provedeno funkcí *uip_udp_new(ripaddr, rport)*. Ta přidělí danému spojení číslo portu a IP adresu dle uvedených argumentů. Je-li návratová hodnota *uip_udp_new()* rovna *NULL*, nepodařilo se provést alokaci paměti pro nové spojení. Jako příklad je uvedena část zdrojového kódu.

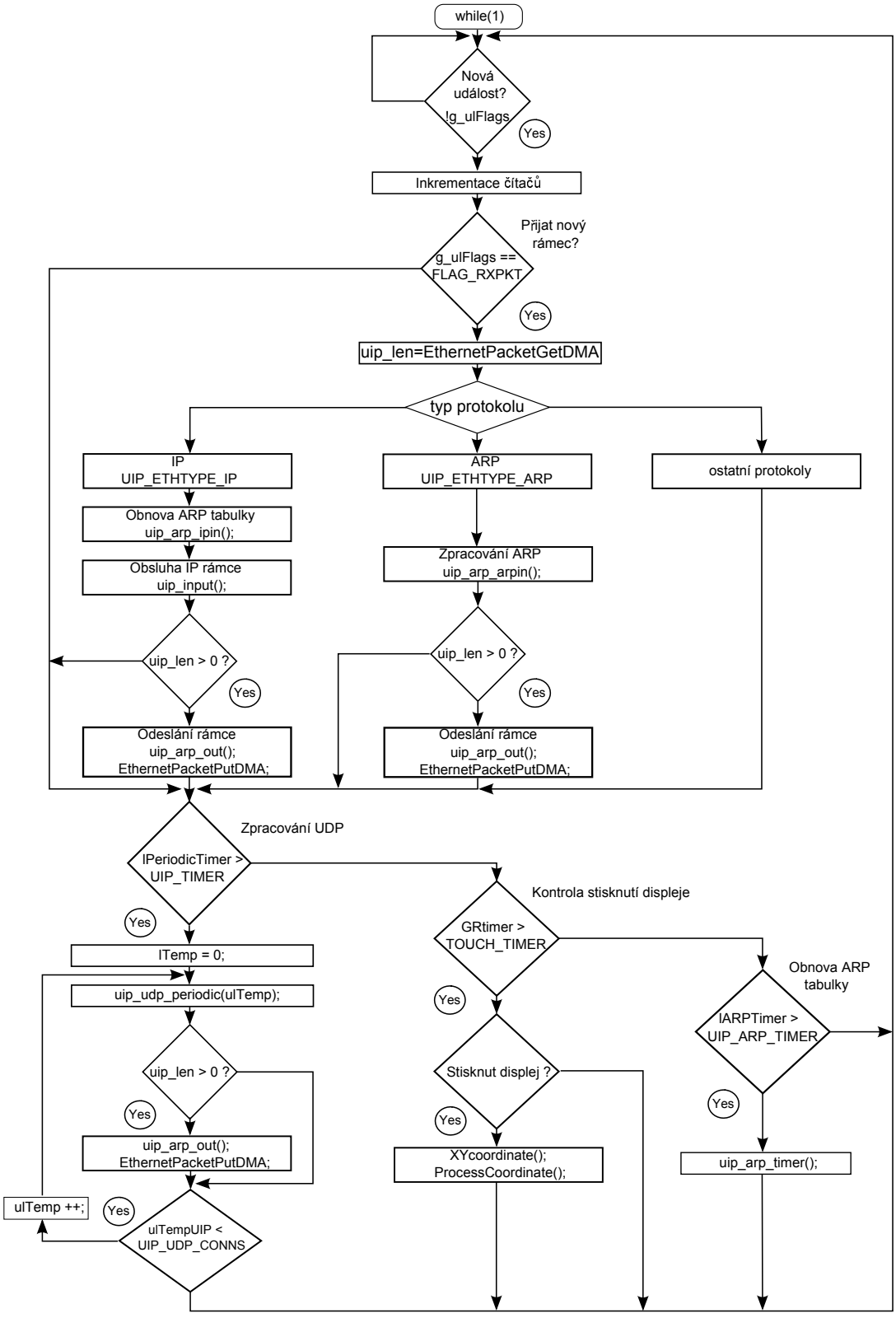
```
//vytvoreni IP adresy se ctyr cisel
uip_ipaddr(addr, 192,168,0,1);

//alokace IP adresy a portu vzdalenoho bodu
s.conn = uip_udp_new(&addr, HTONS(FREEPORT));

//testovani alokace pameti a prideleni lokalniho portu
if(s.conn != NULL)
{
        uip_udp_bind(s.conn, HTONS(FREEPORT));
}
```

Implementace uIP stacku

Na obrázku 3.11 je zobrazeno implementování *uIP stacku* do programu. V závislosti na stavu vnitřního časovače *SysTick* nebo nových datech v *uip_buf*[ ] jsou uloženy příznaky *FLAG_SYSTICK* a *FLAG_RXPKT* do proměnné *g_ulFlags*, jejímž testováním je spouštěna kontrolní smyčka. Po přetečení *SysTick* jsou inkrementovány čítače *lPeriodicTimer, GRtimer, lARPTimer*, které se starají o časování *uIP stacku*.
Dalším krokem je testování, zda byl přijat nový rámec. Pokud ano, je délka dat uložena do *uip_len* a je analyzován typ protokolu z pole length/type Ethernetového rámce. V případě IP paketu je volána funkce *uip_arp_ipin()*, která provede obnovu položky s přijatou IP adresou v ARP tabulce, nebo pokud ještě neexistuje, tak tuto novou adresovou položku vytvoří. Poté následuje funkce *uip_input()*. Jejím úkolem je obsluha IP, ICMP a UDP. Odezva ICMP je řešena interně, pro UDP je voláno příslušná makro *UIP_UDP_APPCALL*.
Jsou-li výsledkem této funkce data, která mají být odvysílána, je uložena jejich velikost v *uip_len* a volána *uip_arp_out*. Tato funkce vytváří v *uip_buf*[ ] hlavičku Ethernetového rámce a IP paketu. Je-li nalezena platná IP adresová položka v ARP tabulce, je daný rámec odvysílán s konkrétní MAC adresou. Pokud v tabulce položka neexistuje, je vyslán ARP request pro zjištění cílové MAC a odvysílání původního rámce je ponecháno na *uIP stacku*.

Byl-li analyzován typ protokolu příchozího Ethernetového rámce jako ARP, je volána funkce *uip_arp_arpin*. Ta v případě odpovědi na ARP request naplní ARP tabulku cílovou MAC adresou. Je-li žádáno o IP adresu řídícího modulu, vytvoří *uip_arp_arpin* ARP reply a uloží do *uip_buf*[ ].

Následující část *uIP stacku* kontroluje stav čítače *lPeriodicTimer*. Je-li dosaženo hodnoty *UIP_PERIODIC_TIMER_MS*, obslouží *uIP stack* veškerá aktivní UDP spojení (počet dán *UIP_UDP_CONNS*) pomocí funkce *uip_udp_periodic()* a případně odvysílá data z aplikace. Druhý čítač *GRtimer* obstarává zpracování souřadnic z dotykového displeje. Funkce *ProcessCoordinate()* byla popsaná v kapitole 3.1. Posledním čítačem *lARPTimer* je spouštěna funkce *uip_arp_timer()*, která odstraňuje staré položky z ARP tabulky.
V této kapitole nebyla zmíněna inicializace *uIP stakcu uip_init()*, která nuluje porty všech pasivních spojení a inicializace hardwarového driveru. Tento stav je ovšem předpokládán.



Obr. 3.11: Implementace *uIP stack* do programu

Vzorová aplikace

Kontrola činnosti *uIP stacku* byla provedena vytvořením call-back funkce nazvané *app_call*, která reaguje na stisknutí grafických prvků a odesílá data vzdálenému uzlu s pevně zvolenou IP adresou. Grafické prvky ukládají data do kruhového bufferu *RingBuff*, který slouží jako vyrovnávací paměť mezi snímáním dotykové vrstvy a dotazováním *app_call*. Na základě příznaku *uip_poll()* je přidělen funkci *app_call* prostor pro čtení stavu kruhového bufferu a jsou-li v něm obsažena dostupná data, odešle je pomocí *uip_udp_send()*.
Řídící modul je schopen reagovat na příchozí data testováním příznaku *uip_newdata()* a poté vykonat danou akci. Zde bylo využito spínání uživatelské diody na základě přijatých znaků + (zapnout), − (vypnout).

V následující části této kapitoly budou rozvedeny nejdůležitější hlavičkové a aplikační soubory použité v *uIP stack*.

uip_arp.c, uip_arp.h

Tyto soubory se zabývají zpracováním ARP protokolu. Je zde definována ARP tabulka a funkce, které se starají o obnovu adresových položek. Zároveň je zde tvořena Ethernetová hlavička odchozích IP paketů a kontrolován stav, zda je nejprve nutné odeslat ARP request.

uip.c, uip.h

Tyto soubory jsou jádrem *uIP stack*. Jsou zde definovány funkce pro inicializaci stacku, zpracování IP, ICMP ,UDP a TCP paketů, definiční makra pro změnu pořadí bytů ve slově a příznaky určující vnitřní stav *uIP stacku*.

uipopt.h

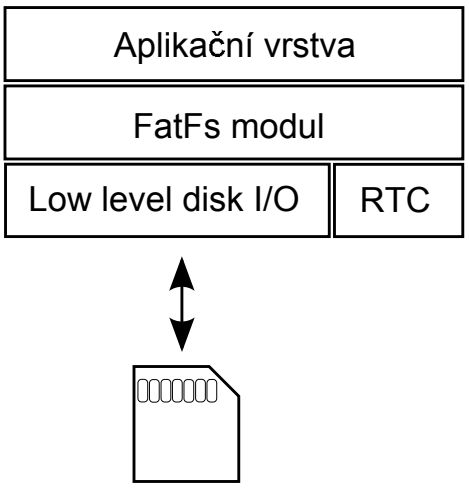
Jedná se o konfigurační soubor *uIP stack*. Pomocí maker je nastavena podpora TCP, UDP protokolů, velikost ARP tabulky, délka bufferu *uip_buf*[ ] atd. Rovněž je v tomto souboru uložena hlavička call-back funkce, její stavová struktura *app_state* a definováno makro *UIP_UDP_APPCALL*. V rámci diplomové práce byla změněna pouze podpora TCP protokolu, zbytek souboru je kompilován v defaultním stavu.

app.c, app.h

V těchto souborech je definována call-back funkce *app_call*, spolu s kruhovým bufferem. V hlavičkovém souboru lze nastavit délku *RingBuff*, která by měla odpovídat délce zpráv, které do ní ukládají grafické prvky.

3.3 Knihovna SDlib

Souborové systémy již dávno nejsou pouze doménou stolních počítačů, ale jejich působení se uplatňuje rovněž v embedded oblasti. Pro diplomovou práci byl vybrán projekt *FatFs*, který je šířen pod BSD licencí a je ho možné využít i pro komerční účely [15]. *FatFs* je napsán v souladu s ANSI C a je kompletně oddělen od fyzického rozhraní mikrokontroléru. Z tohoto důvodu je nezávislý na použité hardwarové platformě.
Na obrázku 3.12 je zobrazena struktura *FatFs* modulu. Pro oživení *FatFs* je nutné vytvořit funkce *low level disk I/O*, závislé na použitém mikrokontroléru, v tomto případě $LM3S9B92$. Knihovna *SDlib* zahrnuje všechny výše uvedené funkce, které budou vysvětleny v následující části této kapitoly.



Obr. 3.12: Struktura modulu *FatFs*

Low level disk I/O

Texas Instruments nabízí podporu ke svým Evaluation kitům s již vytvořeným rozhraním *low level disk I/O*, které bylo importováno do knihovny *SDlib* a pojmenováno *diskio.h*. V něm jsou obsaženy prototypy funkcí uvedené v tabulce 3.3. *FatFs* modul předpokládá velikost char/short/long 8/16/32 bitů a pro int 16 nebo 32 bitů. Z tohoto důvodu je součástí každého souboru knihovny hlavičkový soubor *integer.h*, ve kterém jsou definovány dané typy.
Fyzický přístup na SD kartu je proveden pomocí komunikačního rozhraní *SPI*. Báze, port a piny *SPI* periferie jsou používány jako makra, což vede na jednoduchou modifikovatelnost alternativních funkcí mikrokontroléru. Tato makra jsou uložena v souboru *ssi_hw.h*. V *mmc-dk-lm3s9b96.c* jsou obsaženy veškeré funkce pro správu paměti. Jsou zde *API* pro čtení/zápis dat, kontrola přítomnosti karty, řízení komunikační rychlosti, hodiny reálného času *RTC*(pokud mikrokontrolér nepodporuje *RTC*, musí být poskytny jiné platné hodiny), atd...

Název funkce	Popis funkce
disk_initialize	Inicializace disku
disk_status	Získání informace o stavu inicializace disku
disk_read	Čtení sektoru/ů
disk_write	Zápis sektoru/ů
disk_ioctl	Zápis/čtení řídících příkazů
disk_timerproc	Vnitřní časování FAT modulu

Tab. 3.3: Tabulka funkcí *low level disk I/O* rozhraní

FatFs modul

Tato vrstva, jak plyne z obrázku 3.12, odpovídá souborovému systému FatFs. V knihovně *SDlib* je uložena v souborech *ff.h* a *ff.c*. Pro účely diplomové práce byl využit pouze omezený počet funkcí rozvedený v následující části.

FRESULT f_mount (BYTE, FATFS*)

f_mount je použit pro alokaci pracovní oblasti. Pro práci s SD kartu musí být tato funkce volána jako první. Jejím účelem není fyzický přístup na medium, ale inicializace interní FAT tabulky. Její velikost je ovlivněna počtem připojitelných zařízení (ve výchozím nastavení max. 2).
Argumenty funkce jsou číslo svazku v rozsahu 0 až $\_VOLUMES - 1$ a pointer na pracovní oblast *FATFS**. Návratové hodnoty jsou FR_OK, nebo neplatné číslo svazku $FR\_INVALID\_DRIVE$.

FRESULT f_open (FIL*, const char*, BYTE)

f_open je funkce pro práci se soubory. Je obdobou klasického fopen z jazyka C. Prvním argumentem je pointer na strukturu *FIL*, ve kterém je mimo jiné uložen ukazatel na požadovaný soubor a I/O buffer. Druhým prvkem je jméno souboru a poslední částí hlavičky funkce je příznak, který definuje, zda-li bude ze souboru čteno nebo do něj zapisováno. Návratová hodnota *FRESULT* symbolizuje, s jakým úspěchem byla funkce provedena.

FRESULT f_close (FIL*)

f_close zavírá otevřené soubory. Nejprve je kontrolován stav I/O bufferu a nachází-li se v něm platná data, dojde k jejich odeslání na kartu. Pokud by soubor nebyl korektně uzavřen a došlo by např. k výpadku napájení nebo vyjmutí karty, hrozí ztráta dat, nebo kolaps celého FAT systému. V případě požadavku na kompletní odeslání dat a vyčistění bufferu před uzavřením souboru, je možné použít funkci *f_sync()*.

FRESULT f_write (FIL* fp, const void* buff, WORD btw, WORD* bw)

f_write zapíše data uložená v paměti *buff* do souboru jehož adresa je uložená ve struktuře *fp*. Počet bytů, které mají být zapsány, vyjadřuje proměnná *btw*. *bw* je návratová proměnná *f_write*, ve které je uložen celkový počet zapsaných bytů. Srovnáním *btw* a *bw* lze ošetřit stav plné paměti.

f_read (FIL* fp, void* buff, WORD btr, WORD* br)

f_read čte data ze souboru, jehož adresa je uložená ve struktuře *fp* a ukládá je do paměti *buff*. Argument *btr* vyjadřuje počet bytů ke čtení. *br* vrací skutečný počet přečtených bytů. Je-li *br* menší než *btw*, znamená to, že bylo dosaženo konce souboru za běhu funkce *f_read*.

Aplikační vrstva

V době psaní této kapitoly byla SD karta využita pouze pro ukládání a čtení kalibračních dat, jak bylo popsáno v kapitole 3.1. Jelikož je však přístup a obsluha externí paměti již v režii znalosti několika příkazů, je budoucí využití karty velmi jednoduché.

4

Vzorová aplikace

Účelem této kapitoly je jednoduše shrnout funkci řídícího modulu a vytvořit ukázkovou aplikaci, na které bude předveden způsob práce s firmwarem. Nejprve bude popsán obsah adresáře projektu spolu s použitými knihovnami. Poté bude následovat příklad vytvoření grafického menu a nakonec vysvětlena obsluha Ethernetového komunikačního rozhraní.

Adresář projektu

Kořenový adresář projektu je nazván *Stellaris_LCD* a jeho obsah je shrnut v tabulce 4.1. První tři podadresáře byly využity z programového balíku *StellarisWare*, jejichž obsahem jsou aplikační a hlavičkové soubory driverů, definiční makra registrů a periferií, makra pro přístup do paměti (přímý přístup, bit-banding) a definované datové typy.
Podadresáře *Grlib, SDlib, Ethlib* byly podrobně popsány v kapitole 3. Jejich účelem je implementace grafického rozhraní, embedded *FatFs* modulu a Ethernetového zásobníku *uIP* stack. V kořenovém adresáři se dále nachází soubor *main.c*, soubor pro nastavení příznaků a přerušení *startup_ccs.c* a konfigurační soubory *uip-conf.h, clock-arch.h uIP* stacku.

Adresáře	**Popis**
driverlib	Hardwarové drivery periferií
inc	Makra pro práci s periferiemi
utils	Knihovna obsahující funkce pro kruhový buffer a UART konzoli
Grlib	Grafická knihovna
SDlib	Souborový systém FatFs
Ethlib	Implementace uIP TCP/IP stack
main.c	Tělo programu
startup_ccs.c	Nastavení příznaků a přerušení
uip-conf.h, clock-arch.h	Konfigurační soubory uIP stacku

Tab. 4.1: Obsah adresáře projektu

Grafické rozhraní

Na obrázku 4.1 je zobrazen vývojový diagram programu řídícího modulu. Ten se skládá z volání inicializačních funkcí, které postupně nastaví kmitočet hodin na 80 MHz, poté provede pinout mikrokontroléru, kde jsou povoleny hodiny periferiím a nastaveny alternativní funkce bran. Dalším postupem je konfigurace hardwarového driveru Ethernetového rozhraní.
Následující kroky ovlivňují práci s displejem. Tyto funkce byly již vysvětleny v kapitole 3.1, ale v zájmu snazší orientace bude proveden jejich stručný popis. Funkce *ssd1289_init()* sekvenčně plní registry řadiče SSD1289 a poté vymaže grafickou paměť *GDDRAM*. Následuje nastavení podsvícení. Argumentem funkce *SSD1289BacklightOn()* je číslo od 0 (off) do 255 (on). Poté je kalibrován displej v závislosti na kalibračních datech z SD karty. V dalším kroku je volána funkce *ISinit()*, která naplní počáteční strukturu grafických prvků *InitStr* defaultními hodnotami.

Funkce *MenuObjects()* a *CtrObjects()* tvoří seznamy prvků, které budou zobrazeny na displeji. Pro tlačítko menu je definována proměnná *MenuButton* typu *MenuObject*. Poté jsou deklarovány povinné položky *MenuButton*, jak je vidět z výpisu zdrojového kódu.

```
//deklarace promenne menu tlacitka
MenuObject MenuButton;

void MenuObjects(void)
{
// Vypleni povinnych polozek MenuButton
MenuButton.ID = MENU_MASK | 0x10;
MenuButton.SizeText = Size_16x24;
MenuButton.Text = "1";
CreateMenuObject(&MenuButton);
}
```

Z důvodu různorodosti ovládacích prvků byla definována dvě pole pointerů. Prvním případem je pole *(*ArrayDrawObject[ ])(headerType* Object)*. To obsahuje funkce pro vykreslení ovládacích prvků. Je-li stisknuto tlačítko menu, jsou zobrazeny všechny ovládací prvky s ním spojené (viz kapitola 3.1).
Druhým polem je *(*ArrayCtrObject[ ])(Message* Message)*. Zde jsou umístěny funkce pro obsluhu ovládacích prvků. V případě slideru je v poli umístěna funkce pro pohyb posuvníku. Obě pole jsou uložena v souboru *widget.c*
Opět je uveden výpis kódu pro deklaraci ovládacího prvku typu slider. Jelikož není předem známá pozice, kde bude slider ležet, musí být zadány jeho souřadnice.

```
Slider SliderObject;

void CtrObjects(void)
{
SliderObject.ID = SLIDER_MASK | 0x10;
// ID MenuButton, pod kterym bude ovladaci prvek lezet
SliderObject.ID_MB =  MenuButton.ID;
SliderObject.Xpos = 10;
SliderObject.Ypos = 200;
CreateSlider(&SliderObject);
}
```

Posledním krokem vývojového diagramu je volání funkce *DrawMenuObjects()*, která dle počtu menu tlačítek vypočítá jejich souřadnice a poté je vykreslí.

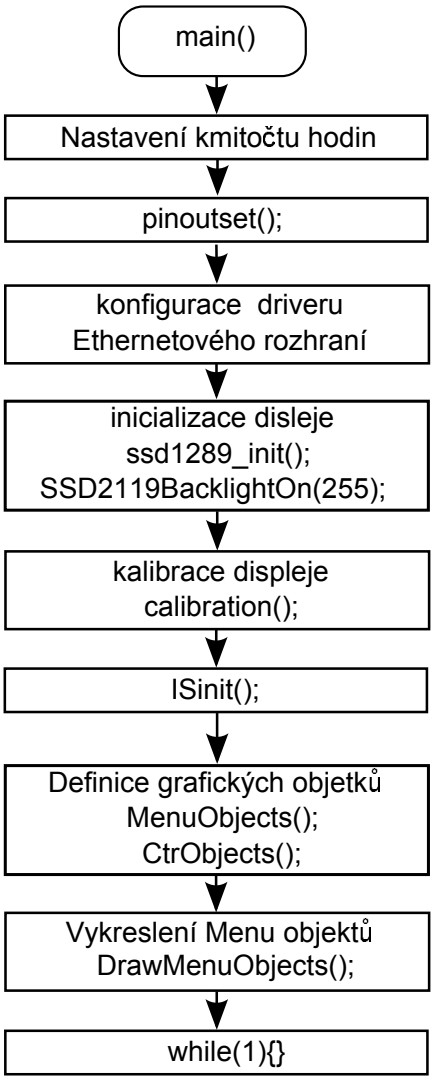
Ethernetové komunikační rozhraní

Pro komunikaci po Ethernetovém rozhraní byl implementován *uIP TCP/IP* stack. Jde o sadu protokolů, které provádějí operace nad daty. V této kapitole bude popsána pouze aplikační část komunikace.
První krokem je nastavení MAC adresy. Pro tento účel je možné použít program LM Flash Programmer od Texas Instruments, který přes rozhraní JTAG uloží MAC adresu do programové paměti. Pokud tak není provedeno, je firmwarem kontrolován stav registrů MAC0 a MAC1, které obsahují hodnotu MAC adresy. Je-li rovná *0xFFFF FFFF FFFF* je automaticky změněna. Viz tabulka 4.2 defaultních adres komunikačního rozhraní.
Dalším prvkem je deklarace síťové adresy IP a masky sítě dle uvedeného výpisu zdrojového kódu. Nejprve je volána funkce *uip_ipaddr()*, která ze čtyř čísel poskládá IP adresu a uloží do proměnné *ipaddr*. Poté je pomocí *uip_sethostaddr()* nastavena IP adresa řídícího modulu. Stejný princip platí pro masku sítě. Uvedená makra *DEFAULT_IPADDR0* atd. jsou součástí hlavičkového souboru *Ethlib.h* v podadresáři *Ethlib*.

```
//Nastaveni IP adresy
uip_ipaddr(ipaddr, DEFAULT_IPADDR0, DEFAULT_IPADDR1, \
                   DEFAULT_IPADDR2, DEFAULT_IPADDR3);
uip_sethostaddr(ipaddr);

//Nastaveni masky site
uip_ipaddr(ipaddr, DEFAULT_NETMASK0, DEFAULT_NETMASK1,\
                   DEFAULT_NETMASK2, DEFAULT_NETMASK3);
uip_setnetmask(ipaddr);
```

```
// inicializace uIP stack
uip_init();
```

Aplikační vrstvou *uIP stack* je call-back funkce, která v závislosti na testování vnitřních příznaků *uIP stack* vytváří UDP datagramy, a poté je odesílá.
Pro volání call-back funkce využívá *uIP stack* makro *UIP_UDP_APPCALL* uložené v souboru *uipopt.h* v podadresáři *Ethlib/uip/uip*. Zároveň je v tomto souboru také stavová struktura call-back funkce *uip_udp_appstate_t*.
V následujícím výpisu zdrojového kódu je zjednodušená call-back funkce *app_call* použitá v diplomové práci pro odesílaní dat získaných z dotykové vrstvy displeje.
Příznakem *uip_poll()* je zjištěno, zda je aplikace dotazována. Poté je provedena kontrola stavu kruhového bufferu a v případě, že obsahuje data, zkopíruje se obsah *RingBuff_Stack* o patřičné velikost *buffer_CtrObject* do dočasného pole *temp*. V dalším kroku je zarovnán pointer *uip_appdata* na data v poli *temp* a pomocí funkce *uip_udp_send* je vytvořeno záhlaví UDP protokolu a datagram je odeslán.



Obr. 4.1: Vzorová aplikace - grafické rozhraní

```
void app_call(void)
{
  unsigned char temp[10];
  if(uip_poll()) // je call-back funce dotazována?
   {
     if(RingBufEmpty(&RingBuff_Stack)) // je buffer prazdny?
     {return;}
     //cteni obsahu bufferu
     RingBufRead(&RingBuff_Stack, temp, strlen(buffer_CtrObject));
     memcpy(uip_appdata, temp, strlen(buffer_CtrObject));
     // vytovreni UDP datagramu
     uip_udp_send(strlen((char*)temp));
   }
}
```

MAC adresa	00-00-02-B6-1A-00
IP adresa	192.168.0.2
Maska podsítě	255.255.255.0

Tab. 4.2: Nastavení síťových adres řídícího modulu

5

Závěr

Cílem diplomové práce bylo navrhnout řídící modul, který bude schopen pomocí dotykového displeje ovládat vzdálená zařízení. Dle zadání byl realizován řídící modul s $3,2$" dotykovým displejem, který komunikuje skrze rozhraní Ethernet. K jeho vlastnostem patří ukládání a čtení dat z SD karty, která dovoluje rychlou modifikovatelnost zařízení, případně komunikace po sběrnici *RS-485*. V zájmu testování řídícího modulu ve fázi programování byl dále realizován JTAG programátor. Fyzický návrh obou zařízení je shrnut v kapitole *Hardware*.

Ve druhé části diplomové práce byla vytvořena grafická knihovna, která je rozdělena do tří základních vrstev. Jejich popis lze stručně shrnout v následujících bodech:

- Fyzická vrstva nastavuje pracovní registry řadiče displeje a zapisuje do jeho interní grafické paměti *GDDRAM*. Zároveň se zde nachází obsluha AD převodníku pro konverzi souřadnic polohy stisknutí displeje.
- Vrstva Graphics zobrazuje základní geometrické tvary a znaky, ze kterých jsou dále skládány složitější objekty.
- Vrstva Widget vykresluje ovládací prvky a zajišťuje zpracování získaných souřadnic z AD převodníku.

Grafické rozhraní je modifikováno voláním API funkcí, které vytvořený ovládací prvek umístí na displej.
Následující část diplomové práce implementuje protokolový zásobník *uIP TCP/IP stack*, zajišťující potřebnou podporu funkcí pro komunikaci přes Ethernet rozhraní. Nad tímto zásobníkem byla postavena uživatelská funkce, která na základě odezvy ovládacích prvků odesílá UDP datagramy vzdáleným zařízením.
V poslední části práce byla provedena implementace souborového systému *FatFs*, pomocí kterého je možné jednoduše editovat soubory v PC, a poté použít v řídícím modulu.

Literatura

[1] *Stellaris LM3S9B92 Microcontroller.* [Cit. 23.3.2013] Dostupné z http://www.ti.com/product/lm3s9b92.

[2] OSGOOD S., DOWNS R., *Touch screen controller tips.* Texas Instruments, 2000. [Cit. 23.3.2013], Dostupné z http://www.ti.com/lit/an/sbaa036/sbaa036.pdf.

[3] FANG W., *Reducing analog input noise in touch screen systems.* Texas Instruments, 2007. [Cit. 23.3.2013], Dostupné z http://www.ti.com/lit/an/sbaa155a/sbaa155a.pdf

[4] FANG W., CHANG T. *Calibration in touch-screen systems.* Texas Instruments, 2007.[Cit. 23.3.2013], Dostupné z http://www.ti.com/lit/an/slyt277/slyt277.pdf

[5] VIDALES C., *How to calibrate touch screens.* Embedded, 2002. [Cit.3.2013] Dostupné z http://www.embedded.com/design/configurable-systems/4023968/How-To-Calibrate-Touch-Screens

[6] *Touch screen controller.* [Cit. 23.3.2013] Dostupné z http://www.ti.com/lit/ds/sbas090b/sbas090b.pdf

[7] *TFT LCD Controller driver.* [Cit. 23.3.2013] Dostupné z http://www.solomon-systech.com/en/product/display-ic/smart-tft-lcd-driver-controller/ssd1289/

[8] *Simple switch, Step-down voltage regulator LM22672* [Cit. 23.3.2013] Dostupné z http://www.ti.com/lit/ds/symlink/lm22672.pdf

[9] ARM support. *ARM Information Center.* [Cit. 23.3.2013] Dostupné z http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0499d/BEHEIHCE.html

[10] TI E2E Community. *Stellaris-ICDI Programming.* [Cit. 23.3.2013] Dostupné z http://processors.wiki.ti.com/index.php/Stellaris-ICDI˙Programming

[11] *Stellaris LM3S9D92 Evaluation Kit.* [Cit. 23.3.2013] Dostupné z http://www.ti.com/lit/ug/spmu174/spmu174.pdf

[12] *Stellaris LM3S6965 Evaluation Board.* [Cit. 23.3.2013] Dostupné z http://www.ti.com/lit/ug/spmu029a/spmu029a.pdf

[13] BARTOVSKÝ, J., *Universal Graphic Library for STM32 Family*. Plzeň, 2009. Diplomová práce. Západočeská univerzita v Plzni. Fakulta elektrotechnická.

[14] Dunkels,A., *The uIP Embedded TCP/IP Stack*. [Cit. 23.3.2013] Dostupné z http://sourceforge.net/projects/uip-stack/

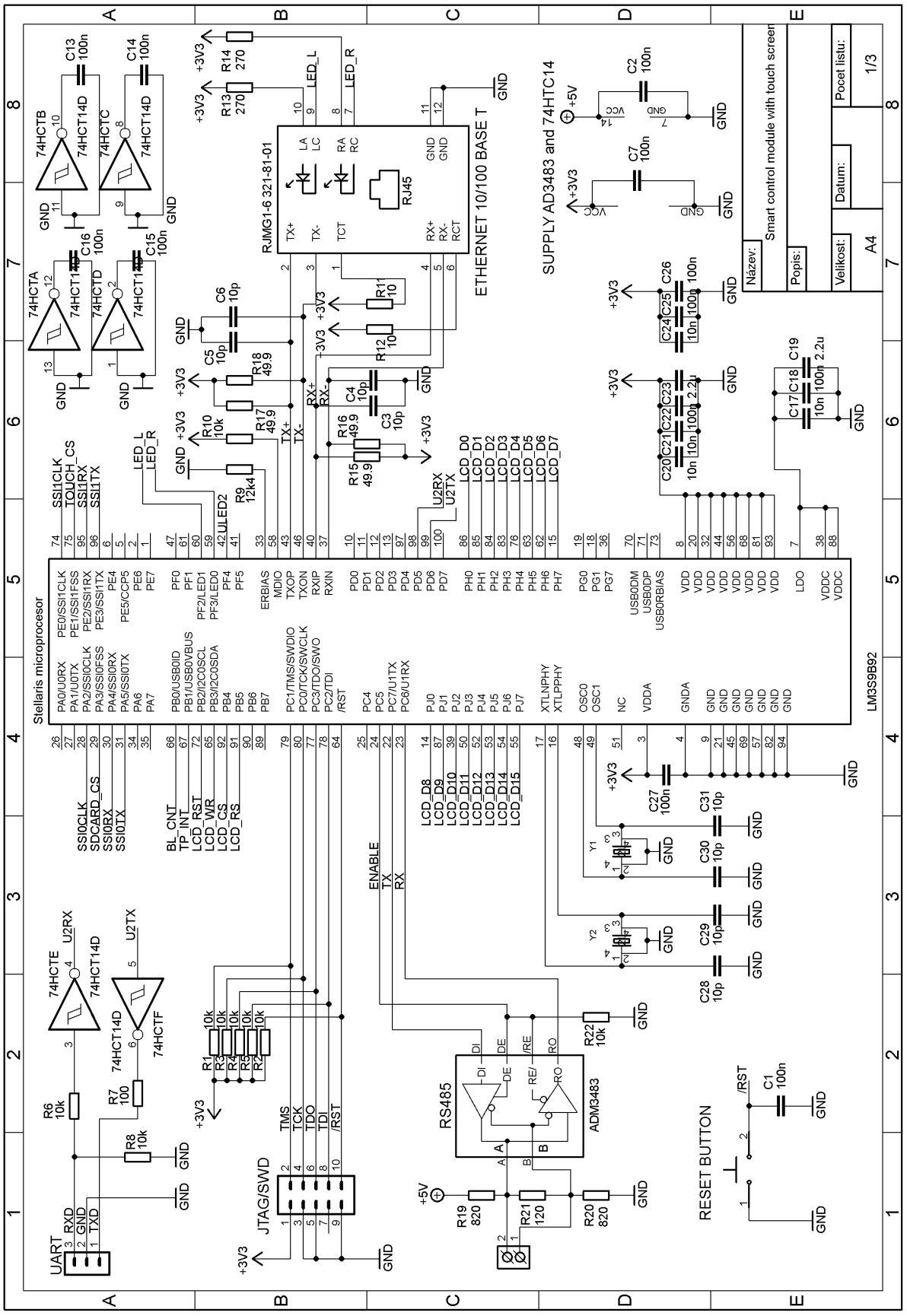
[15] Chan, *FatFs - Generic FAT File System Module*. [Cit. 23.3.2013] Dostupné z http://elm-chan.org/fsw/ff/00index˙e.html

Příloha A

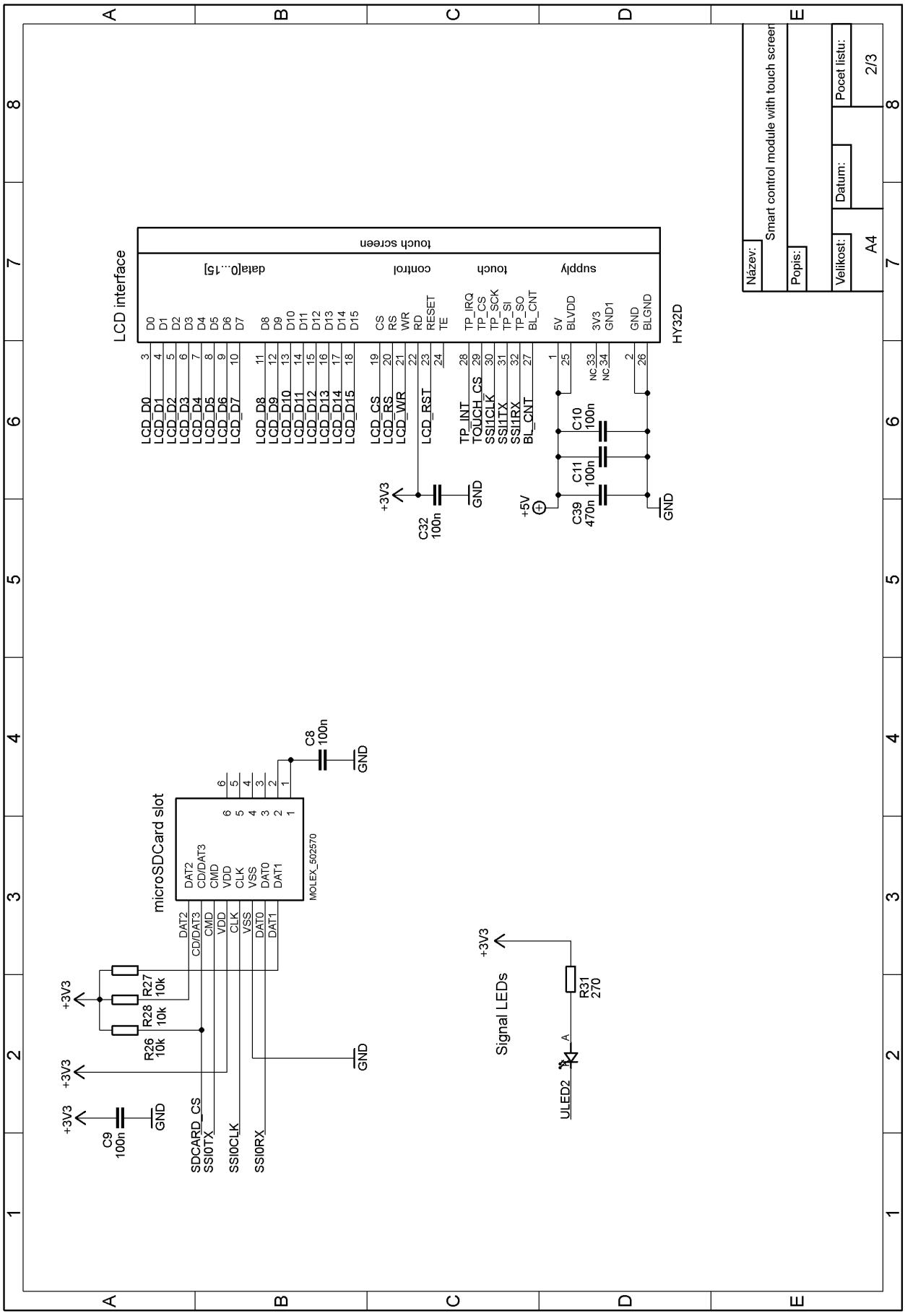
Schémata zapojení

A.1 Řídící modul

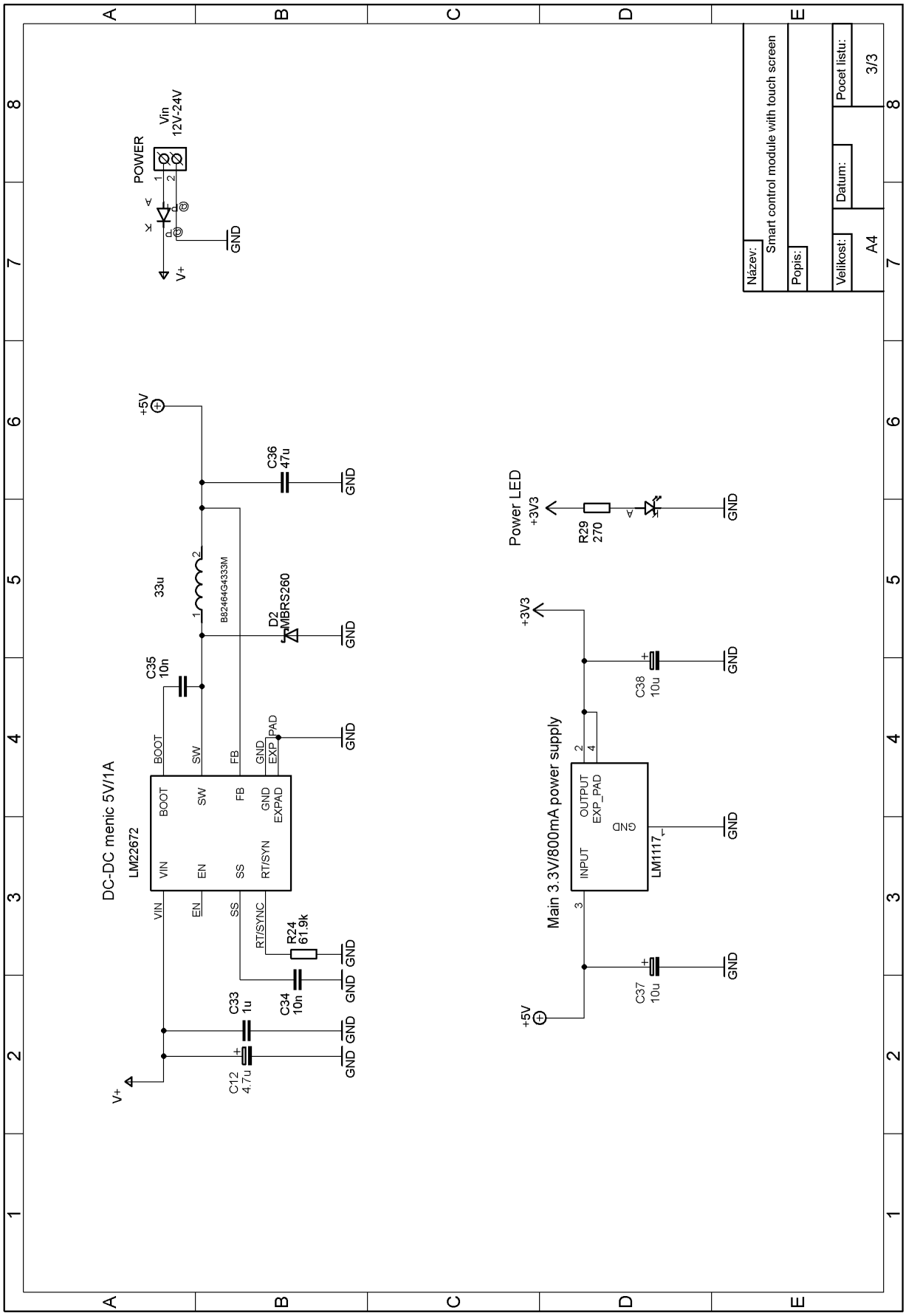
A.2 Programátor ICDI



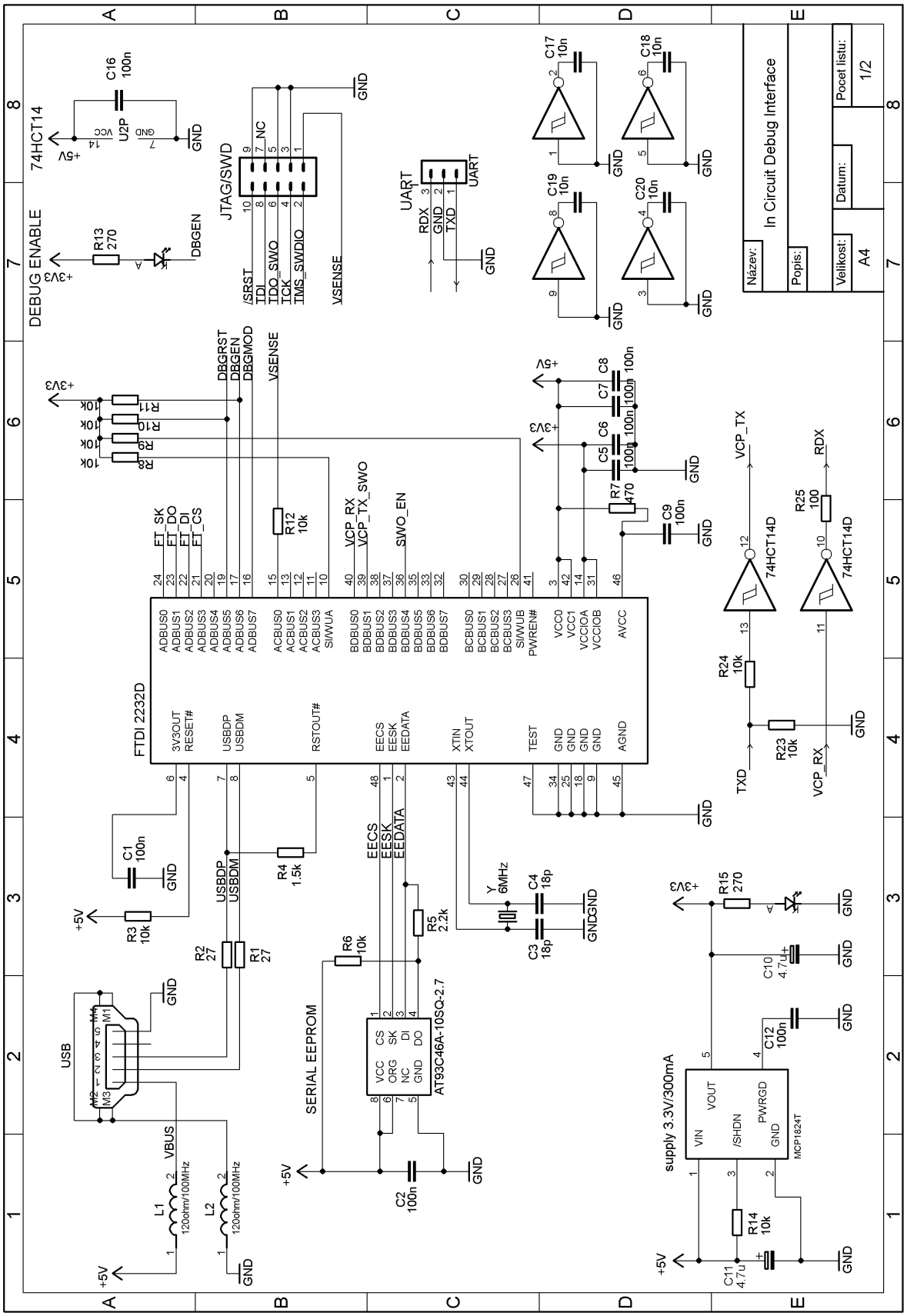
Obr. A.1: Schéma zapojení řídícího modulu, list 1/3



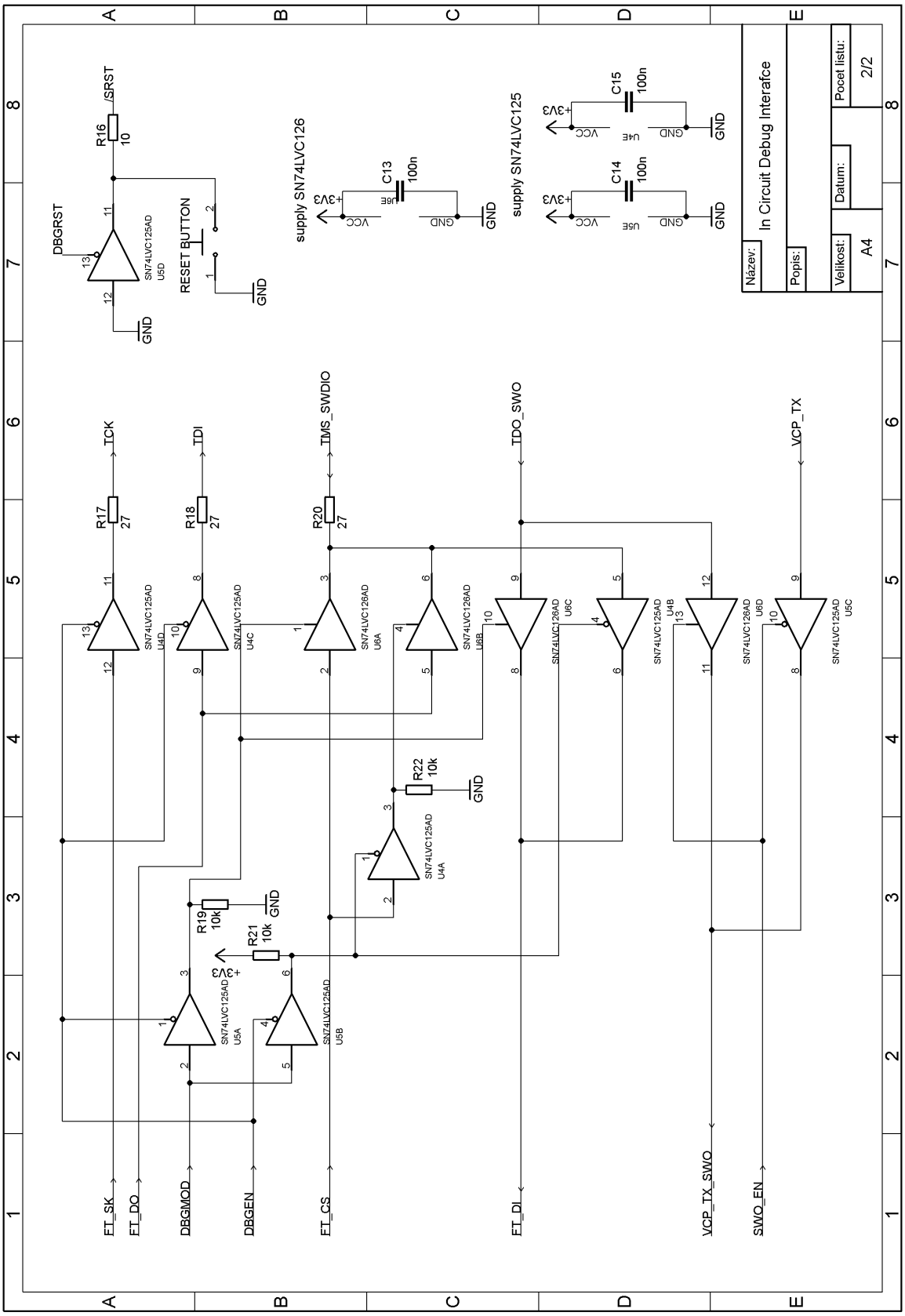
Obr. A.2: Schéma zapojení řídícího modulu, list 2/3



Obr. A.3: Schéma zapojení řídícího modulu, list 3/3



Obr. A.4: Schéma zapojení ICDI programátoru, list 1/2



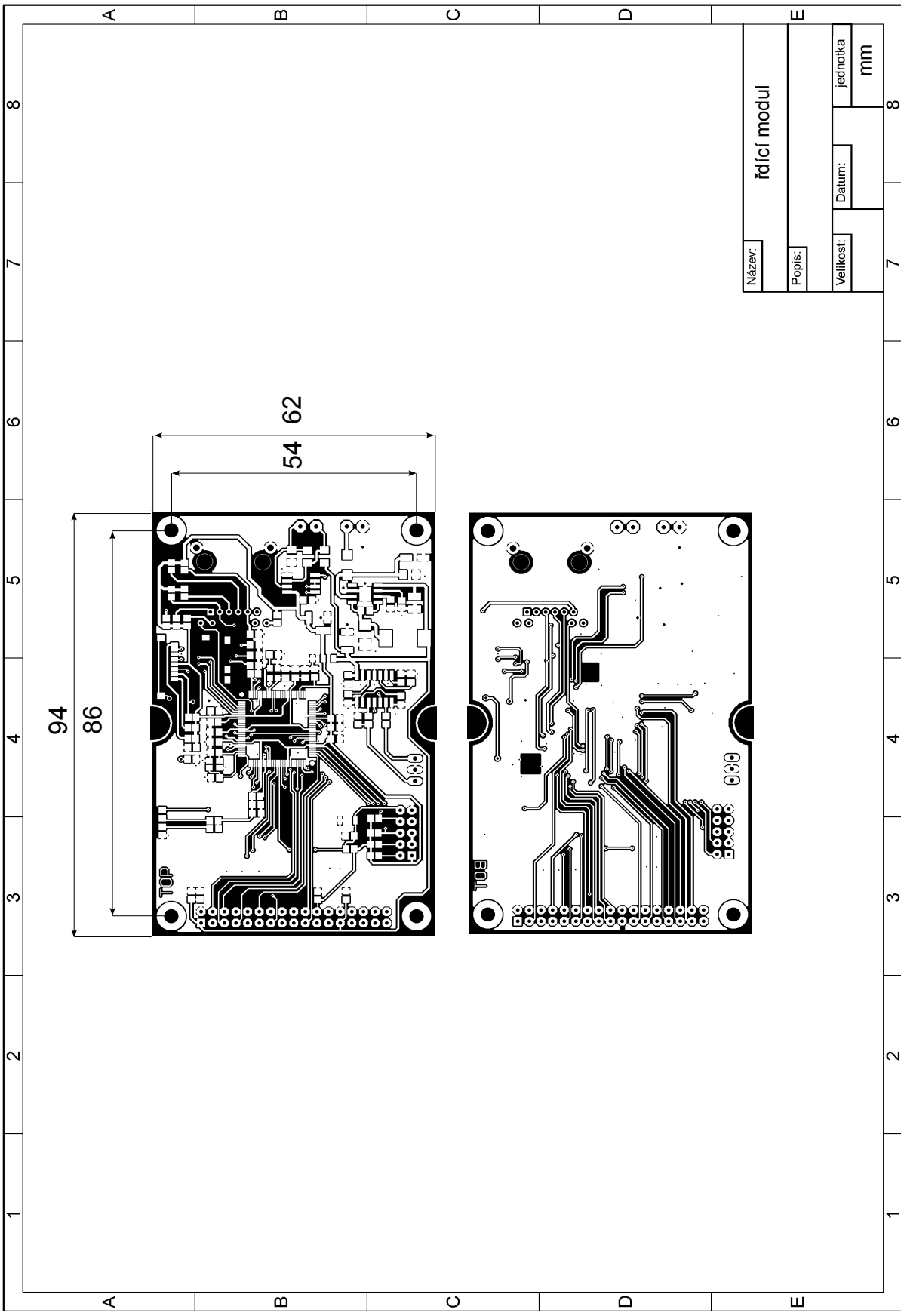
Obr. A.5: Schéma zapojení ICDI programátoru, list 2/2

Příloha B

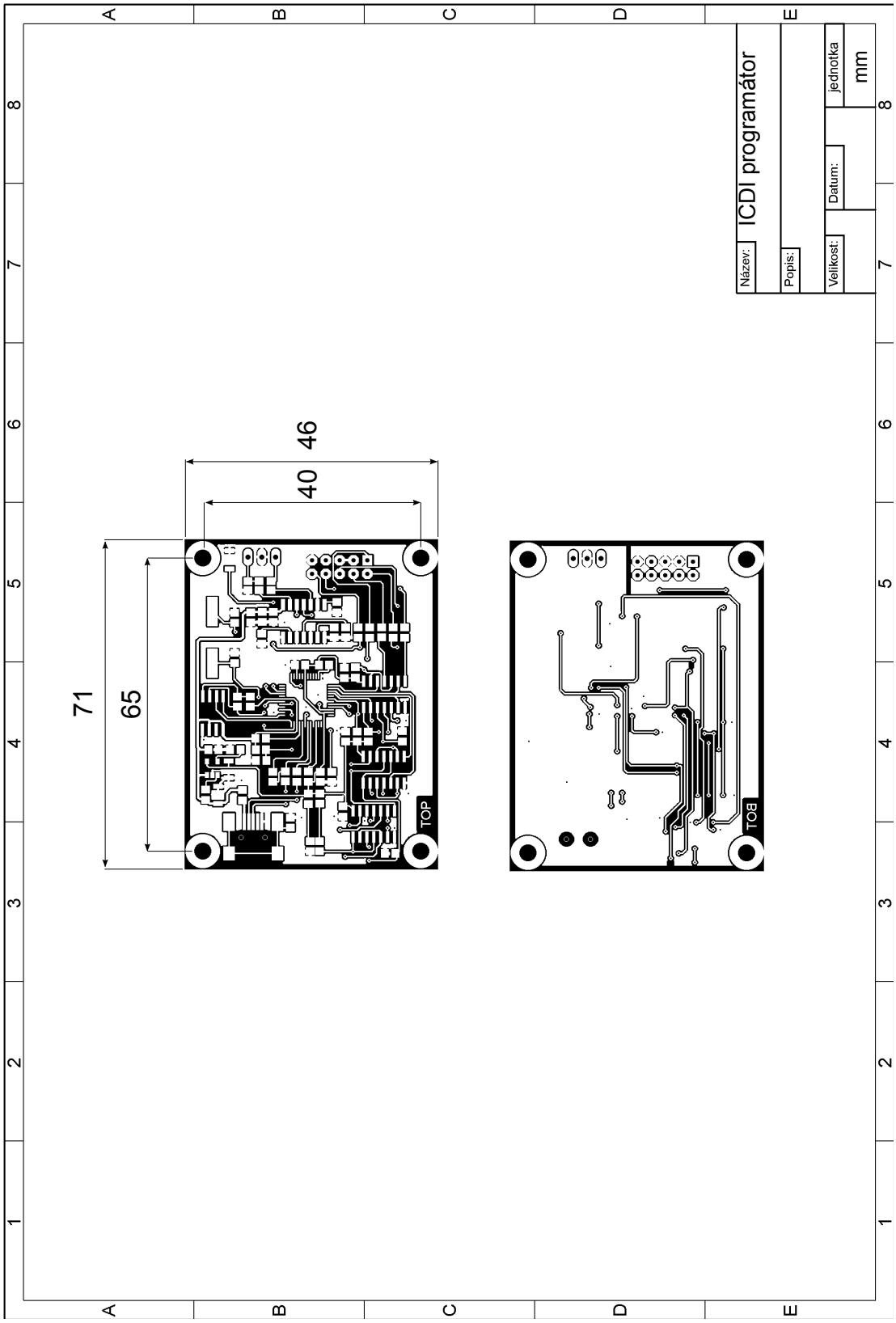
Desky plošných spojů

B.1 Řídící modul

B.2 Programátor ICDI



Obr. B.1: DPS řídící modul



Obr. B.2: DPS programátor ICDI

Příloha C

Seznam součátek

C.1 Řídící modul

C.2 ICDI programátor

Součástka	Popis	Počet
U1	mikrokontrolér, LM3S9B92	1
LCD	dotykový modul HYD32	1
74HCT	HEX SCHMIT, 74HCT14D	1
ADM3483	Budič RS485, ADM3483	1
L1	Inductor, B82464G4333M	1
BUT	tactile spínač, B3U-1000P	1
D1	Schott. dioda, MBRS260	1
U10	Spínaný reg. 5V/1A, LM22672	1
U11	LDO stabilizátor 3V3, LM1117	1
Y1	Oscilátor 16MHz, FA-238	1
Y2	Oscilátor 25MHz, FA-238	1
U6	Kon. SDkart, MOLEX - 502570-0893	1
U3	Ethernt. konekt., RJMG163218101NR	1
C36	TPSB476K010R0250, C 47uF	1
C12	EEEFK1V4R7R, C 4u7F	1
C1, C2, C7, C8, C9, C10, C11 C13, C14, C15, C16, C18 C22, C25, C26, C27, C32	C 100nF 0805	17
C3, C4, C5, C6, C28, C29, C30, C31	C 10pF 0805	8
C17, C20, C21, C24, C34, C35	C 10nF 0805	6
C19, C23	C 2u2F 0805	2
C33	C 1uF 0805	1
C37, C38	C 10uF 0805	2
C39	C 470nF 0805	1
R1, R2, R3, R4, R5, R6 R8, R10, R22, R26, R27, R28	R 10k 0805	13
R7	R 100 0805	1
R9	R 12k4 0805	1
R10, R11	R 10 0805	2
R13, R14, R29, R31	R 270 0805	4
R15, R16, R17, R18	49R9 0805	4
R19, R20	R 820 1206	2
R21	R 120 1206	1
R24	R 61k9 0805	1
POWER, RS485	Konektor ARK550	2
LS4	Červená LED, 0805	1
LSUP	Zelená LED, 0805	1
D1	Dioda, SS16	2

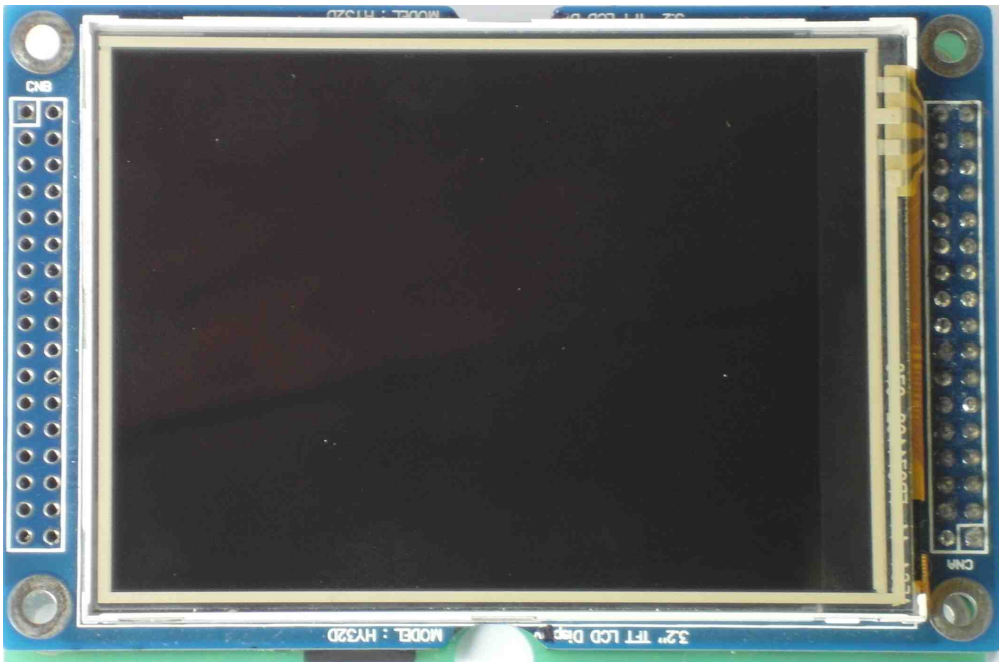
Tab. C.1: Seznam součástek pro desku řídícího modulu

Součástka	**Popis**	**Počet**
MCP1824T	Stabilizátor napětí 3V3/300mA	1
FT2232L	FT2232L	1
74HCT14D	HEX SCHMITT	1
SN74LVC125AD	Třistav. budiče	2
SN74LVC126AD	Třistav. budiče	1
USB	miniUSB	1
BUT	tactile spínač	1
EEPROM	EEPROM, AT93C46A-10SQ-2.7	1
LDEB	červená LED 0805	1
LSUP	zelená LED 0805	1
C1, C2, C5, C6, C7, C8, C9 C12, C13, C14, C15, C16	C 100nF 0805	12
C3, C4	C 18pF 0805	2
C10, C11	C 4u7F tantal SMDA	2
C17, C18, C19, C20	C 10n 0805	4
R1, R2, R17, R18, R20	R 27 0805	5
R3, R6, R8, R9, R10, R11, R12, R14 R19, R21, R22, R23, R24	R 10k 0805	13
R4	R 1k5 0805	1
R5	R 2k2 0805	1
R7	R 470 0805	1
R13, R15	R 270 0805	2
R16	R 10 0805	1
R100	R 100 0805	1

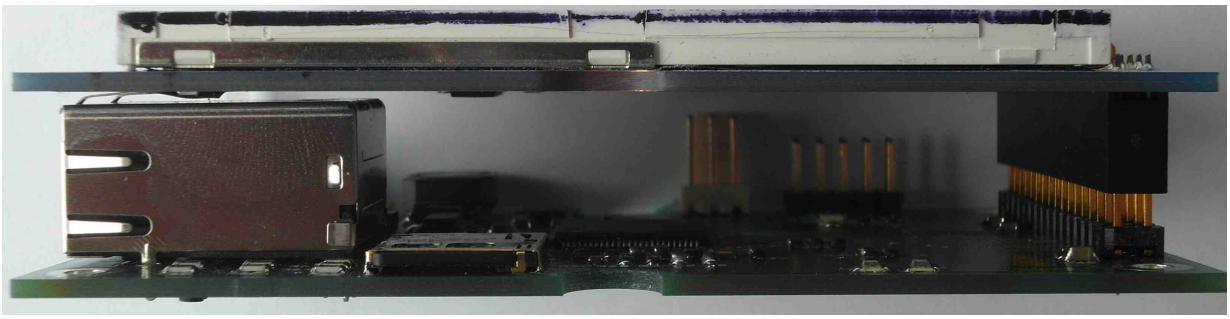
Tab. C.2: Seznam součástek pro desku ICDI programátoru

Příloha D

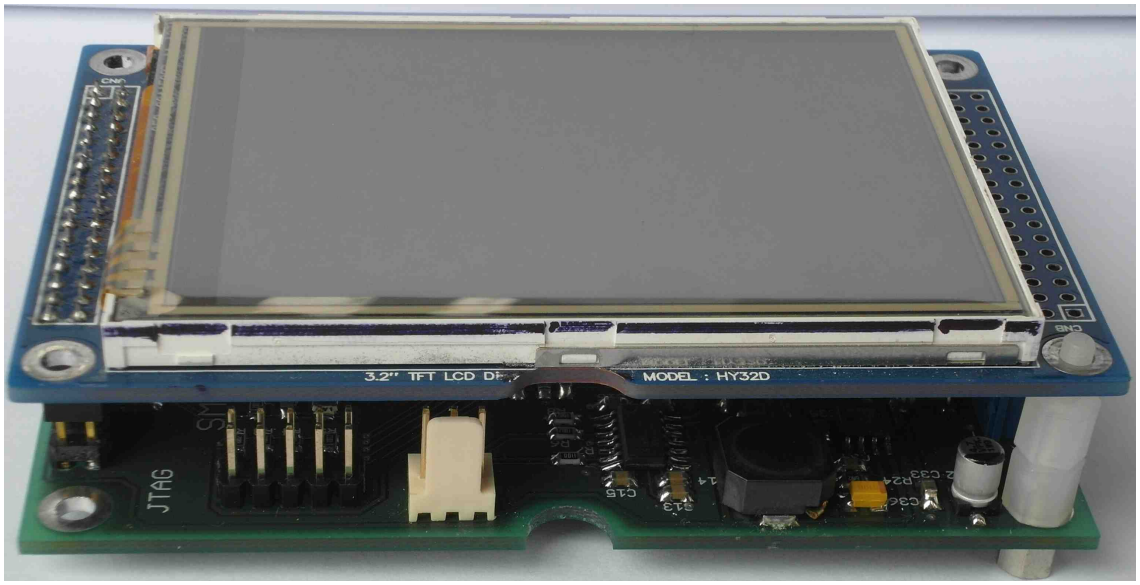
Fotografická dokumentace



Obr. D.1: zobrazení pohledové části



Obr. D.2: Spojení horního a dolního patra řídícího modulu



Obr. D.3: Boční pohled

Příloha E

CD-ROM

Součástí přílohy je CD, které obsahuje zdrojové kódy a text práce.