VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

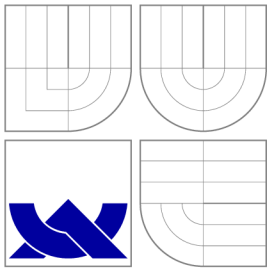
PROGRAMÁTOR PRO ICSP ROZHRANÍ MIKROKONTROLÉRŮ PIC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

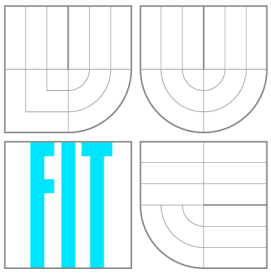
AUTOR PRÁCE
AUTHOR

PAVEL DOLEŽAL

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

PROGRAMÁTOR PRO ICSP ROZHRANÍ MIKROKONTROLÉRŮ PIC

IN-SYSTEM PROGRAMMER FOR ICSP INTERFACE

OF PIC MICROCONTROLLERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PAVEL DOLEŽAL
AUTHOR

VEDOUCÍ PRÁCE — Ing. VÁCLAV ŠIMEK
SUPERVISOR

BRNO 2010

Abstrakt

Tato práce se zabývá návrhem programátoru pro programování mikrokontrolérů Microchip PIC 16F a 18F. Její součástí je návrh zapojení, deska plošných spojů a obslužná aplikace. Navržený programátor je osazen mikrokontrolérem PIC18F67J50, grafickým displejem a paměťovou kartou. Konstrukce umožňuje snadné přizpůsobení většině cílových součástek. V práci jsou popsány principy programování pomocí protokolu ICSP firmy Microchip se zaměřením na zmíněné rodiny procesorů. Součástí práce je funkční programátor.

Abstract

This thesis is dealing with a design of programming device with the expected use for Microchip PIC 16F and 18F microcontrollers. Individual chapters are dedicated to aspects of circuitry, printed circuit board design and user software tools. The proposed device is based on PIC18F67J50 microcontroller, graphic LCD diplay, and memory card interface. Design can be easily adjusted in order to support wide range of devices. Microchip's In-Circuit Serial Programming protocol is described with focus on previously mentioned family of devices. Working prototype of the programmer is attached to this thesis as a demonstration of a practical outcome.

Klíčová slova

Programátor, Mikrokontrolér, Microchip, PIC, ICSP, Vestavěný systém, Linux, Intel HEX, CDC

Keywords

Programmer, Microcontroller, Microchip, PIC, ICSP, Embedded system, Linux, Intel HEX, CDC

Citace

Pavel Doležal: Programátor pro ICSP rozhraní
mikrokontrolérů Pic, bakalářská práce, Brno, FIT VUT v Brně, 2010

Programátor pro ICSP rozhraní mikrokontrolérů Pic

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Václava Šimka

.....
Pavel Doležal
19. května 2010

Poděkování

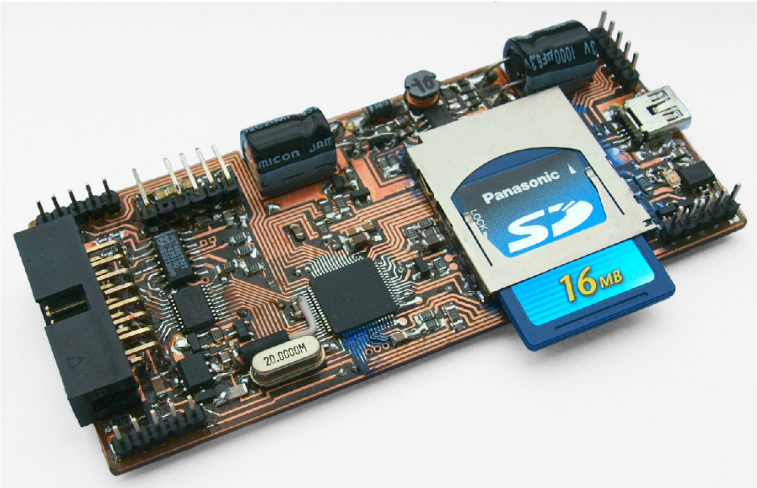
Rád bych poděkoval Ing. Václavu Šimkovi za vedení této práce.

Obsah

Kapitola 1

Úvod

Vynález tranzistoru roku 1947 odstartoval novou éru v dějinách elektroniky. Vlastnosti této nové součástky stály na začátku miniaturizace elektronických zařízení, která úspěšně pokračuje dodnes. Jedenáct let poté, v roce 1958 se podařilo do pouzdra společně s tranzistorem vložit ještě několik pasivních součástek, a tak vznikl první integrovaný obvod.



V 70. letech minulého století svět zažíval rozmach kapesních kalkulaček a firma Intel za tímto účelem vyvinula integrovaný obvod Intel 4004. Vědecká společnost však brzy zjistila, že lze na jeho základu sestavit jenoduchý počítač. Pozdější mikroprocesory jako MOS 6502, Intel 8080 nebo Zilog Z80 doplněné o paměť RAM a další periferie se staly základem domácích počítačů.

S pronikáním digitální techniky do většiny odvětví však vzrůstala i poptávka po jednoduchých a levných počítačích, které by ke své činnosti nevyžadovaly velké množství externích komponent. Jednou z prvních takových součástek byl například obvod Intel 8051. Díky konstrukci, která v sobě spojuje mikroprocesorové jádro, paměť programu a periferie se pro takové obvody vžilo označení „mikrokontrolér“.

Ruku v ruce s výrobou mikrokontrolérů ale vznikl problém. Jakým způsobem dostat do těchto univerzálních čipů chování, které od nich očekáváme – tzv. firmware. V dnešní době tuto činnosti obstarávají nejčastěji speciální periferie osobních počítačů, tzv. programátory. Tato práce se zabývá návrhem a výrobou programátoru pro mikrokontroléry firmy Microchip.

Kapitola 2

Hardwarové řešení programátoru

Na začátku práce bylo nutné položit si zásadní otázku:

Navrhnout a vyrobit pouze další programátor pro mikrokontroléry Microchip, nebo se pokusit o návrh univerzálního řešení, které by přesáhlo hranice jedné architektury?

Aniž bych si uvědomil zrádnost slova „univerzální“,
zvolil jsem cestu složitějšího řešení...

Rovněž bylo zřejmé, že pokud má být programátor připraven pro spolupráci s velkým množstvím součástek, bude nutné ho vybavit kromě USB převodníku i mikrokontrolérem nebo hradlovým polem. Zpočátku vývoje byl jako řídící obvod uvažován mikrokontrolér PIC16F877 [8] ve spolupráci s USB převodníkem FTDI 2232 [4], který jsem zvolil proto, že ačkoli jsem pro mikrokontroléry nikdy žádný velký projekt nenapsal, měl jsem dostupný vývojový kit právě s tímto mikrokontrolérem.

Brzy se ovšem ukázalo, že osm let starý PIC16F877 nebude svými paramtery pro navrhovaný programátor stačit. Náhle jsem stál před problémem. Musím vybrat nový, pro mne zcela neznámý mikrokontrolér, který s největší pravděpodobností nebudu mít ani čím naprogramovat. Na druhou stranu mi tato situace dala možnost vybrat si mikrokontrolér „na míru“ podle toho, co budu při konstrukci potřebovat. Tento fakt dodal celé práci nový rozměr. Programátor měl být schopný prácovat bez připojení k počítači a proto byla navrhovaná deska rozšířena o velké množství periferií.

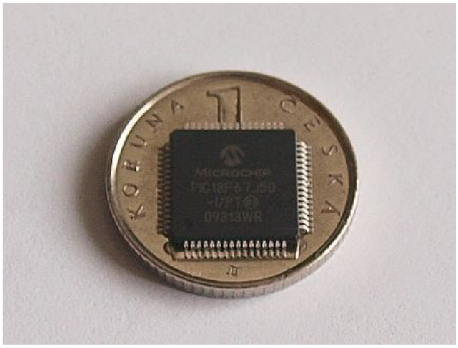
V následující kapitole jsou podrobně rozebrány jednotlivé funkční bloky navrhovaného programátoru, jejich zapojení a princip činnosti. Programátor je primárně určen pro programování dvou nejvyšších rodin osmibitových mikrokontrolérů PIC16F a PIC18F. Nicméně je pravděpodobné, že navržená konstrukce bude použitelná pro programování mnoha dalších součástek.

2.1 Řídící mikrokontrolér

Programátor je řízen mikrokontrolérem PIC18F67J50 taktovaným na frekvenci 48 MHz, s vnitřní pamětí o kapacitě 128 kB a integrovaným USB rozhraním 2.1. Tím odpadá nutnost použít pro komunikaci s počítačem samostatný převodník. Dále je mikrokontrolér vybaven dvěma nezávislými sériovými jednotkami (jedna nakonfigurována jako SPI, druhá jako I^2C), 10 bitovým A/D převodníkem a dalšími periferiemi. Protože bude v následujícím textu často

řeč o dvou mikrokontrolérech, je tento v dalším textu označován jako *řídící mikrokontrolér*, zatímco mikrokontrolér, který bude programován pomocí programátoru bude uváděn jako *cílová součástka*.

Každý mikrokontrolér potřebuje ke své činnosti hodinový signál. Moderní mikrokontroléry mají většinou velké množství možností, jak tento signál generovat. Jednotlivé varianty se liší frekvencí, spotřebou a v neposlední řadě také nutností použít v zapojení další součástky. V navrhovaném programátoru je pro generování hodinového signálu použit krystal o frekvenci 20 MHz a režim oscilátoru HS-PLL. Tato kombinace byla zvolena s ohledem na možnost provozovat vestavěný USB modul v režimu High-Speed, což při libovolné konfiguraci není možné. Pomocí fázového závěsu zabudované násobičky mikrokontrolér generuje frekvenci 96 MHz, která slouží jako výchozí bod pro generování všech ostatních kmitočtů nutných pro provoz periferií atd. Detailní popis lze nalézt v oficiální dokumentaci [10]. Mikrokontrolér je napájen napětím 3,3 V, o napájení samotného jádra napětím 2,5 V se stará zabudovaný stabilizátor.



Obrázek 2.1: Řídící mikrokontrolér PIC18F67J50

Většina pinů, které nesdílejí žádný analogový režim činnosti, je tolerantní vůči signálům 5 V logiky. Tolerantní vstupy jsou podstatnou vlastností při komunikaci s programovanou součástkou. Koneckonců je pro ni vyhrazena bezmála třetina pinů řídícího mikrokontroléru. Všechny vybrané piny mohou pracovat pouze v digitálních režimech, což umožňuje bezpečně komunikovat s cílovou součástkou, která pracuje s napětím 5 V. Aby bylo možné dosáhnout dostatečných napěťových úrovní při komunikaci směrem z řídícího mikrokontroléru do cílové součástky, jsou v zapojení použity dodatečné obvody. Popis jejich funkce je uveden v kapitole 2.2.2 Přepínání napěťové hladiny . Kompletní seznam vývodů řídícího mikrokontroléru a funkce, ke kterým slouží, jsou uvedeny v tabulce B.1.

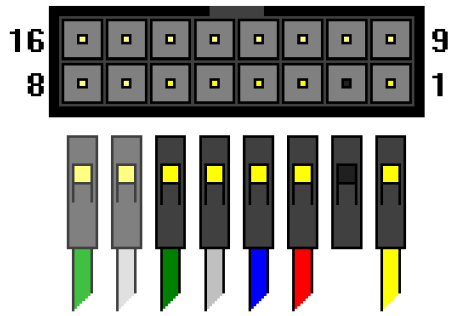
2.2 Komunikační rozhraní

Následující odstavce obsahují základní údaje o provedení rozhraní, které slouží pro komunikaci s cílovou součástkou. Součástky Microchip jsou programovány sériově pomocí protokolu ICSP, který bude detailně popsán v dalších kapitolách této práce.

Pro komunikaci s cílovou součástkou slouží šestnáctipinový konektor ve spodní části programátoru. Jeho zapojení je navrženo tak, aby bylo možné nadále používat kabeláž vyrobenou pro programátory české firmy Asix. Jediný rozdíl v zapojení konektoru ICSP mezi referenčním zapojením firmy Microchip a firmou Asix, je ve vložení klíčovacího pinu mezi V_{PP} a V_{CC}, který zabraňuje tomu, aby byl konektor zapojen naopak.

2.2.1 Vodiče komunikačního rozhraní

Samotné ICSP rozhraní vyžaduje celkem tři napájecí a dva signálové vodiče. Postupným rozšiřováním rozhraní došlo k navýšení datových vodičů z nutných dvou, na konečných 12. Zapojení konektoru je na obrázku 2.2, popis jednotlivých pinů v tabulce 2.1. Rozdělení pinů v konektoru je zachováno ve prospěch standardu Asix, ačkoli díky tomu je horní (doplňková) řada uspořádána ne zcela logickým způsobem. Hlavní (spodní) řada je stejně jako u programátoru Asix PRESTO doplněna o signály MISO a LVP, které dávají navrhovanému programátoru hardwarové předpoklady k pokrytí minimálně stejně široké součástkové základny.



Obrázek 2.2: Schéma konektoru programátoru

PIN	Název	Barva	Popis
1	Vpp	Žlutá	Programovací napětí 12 V nebo 13 V
2	–	–	Klíč, ochrana proti nesprávnému zapojení
3	Vcc	Červená	Napájecí napětí 3,3 V nebo 5,0 V
4	GND	Modrá	Zem
5	Data	Zelená	ICSP datový pin
6	Clock	Bílá	ICSP hodinový pin
7	MISO	Zelená	Vstupní datový vodič pro režim SPI
8	LVP	Bílá	Pin pro nízkonapěťově programování
9-14	I/O	–	Šest obousměrných pinů TTL/LVTTL
15-16	Input	–	Dva vstupní piny TTL/LVTTL

Tabulka 2.1: Popis konektoru programátoru

2.2.2 Přepínání napěťové hladiny

Různé cílové součástky vyžadují při programování různé napěťové hladiny. V současné době se setkáváme nejčastěji s programováním pomocí napěťových hladin 5 V, 3,3 V, 2,5 V a 1,8 V. Obecně platí pravidlo, že čím složitější a novější součástku programujeme, tím nižší je napěťová hladina. Navrhovaný programátor by proto měl být schopen sám přepínat tyto napěťové hladiny bez nutnosti používat nejrůznější převodníky nebo hardwarové úpravy.

Bohužel, zvolené součástky neumožnily podporu 1,8 V a 2,5 V, proto je navržený programátor schopen programovat cílovou součástku pouze pomocí signálů o napětí 3,3 V nebo

5 V. Přesto v podpoře 3,3 V programování vidím jednu z největších výhod celé konstrukce. Odpadá tím nepříjemný a čím dál častější problém s programováním „3,3 V only“ součástek, které bez speciálního převodníku ve většině programátorů jednoduše nelze naprogramovat. Například firma Asix tento problém u programátoru PRESTO řeší volitelným příslušenstvím, které ale znemožňuje programování součástek Microchip[1].

Všechny vstupy komunikačního rozhraní 2.3 jsou 5 V tolerantní bez ohledu na aktuálně zvolenou napěťovou hladinu.[2] Rozhraní je založeno na dvou obvodech řady VHC, jejichž výhodou jsou hlavně 7 V tolerantní vstupy a výstupní proud až 25 mA.

Prvním obvodem je oktalový transciever 74VHC245 [21], který zajišťuje obousměrnou komunikaci mezi 3,3 V logikou řídícího mikrokontroléru a navolenou napěťovou hladinou výstupního rozhraní. Při zápisu programu do cílové součástky pomocí 5 V napěťové hladiny obvod zvyšuje logickou úroveň z 3,3 V na 5 V. Při čtení programu slouží pouze jako mezičlánek mezi cílovou součástkou a řídícím mikrokontrolérem. Při komunikaci s cílovou součástkou na napěťové hladině 3,3 V slouží obvod pouze jako proudové posílení výstupních signálů, což snižuje namáhání vývodů řídícího mikrokontroléru. Tímto způsobem jsou zapojeny obousměrné piny komunikačního rozhraní, včetně datového signálu ICSP rozhraní.

Pro posílení hodinového signálu ICSP je použit integrovaný obvod 74VHC32[20]. Ten obsahuje čtyři dvouvstupová hradla typu OR, z nichž první, obdobným způsobem jako u obousměrných signálů, zvyšuje logickou úroveň z 3,3 V na 5 V. Zbylá hradla jsou zapojena jako univerzální vstupy a ošetřena proti zákmitům slabými pull-up rezistory. Klidové napětí na těchto vstupech odpovídá aktuálně nastavené napěťové hladině komunikačního rozhraní. Jeden z těchto vstupů se nachází, stejně jako na programátoru Asix PRESTO, vedle hodinového signálu a je připraven pro použití jako pin MISO pro softwarovou implementaci rozhraní SPI například pro programování součástek Atmel.

2.2.3 Napájení komunikačního rozhraní

Možnost přepínat napěťovou hladinu komunikačního rozhraní je založena na přítomnosti odděleného stabilizátoru napětí pro obvody komunikačního rozhraní 2.4. Stabilizátor LE33CD [22] navíc disponuje funkcí *inhibit*, které umožňuje komunikační rozhraní kompletně vypnout. Vnitřní konstrukce LE33CD navíc umožňuje bezpečně přemostit vlastní vstup a výstup, čehož je využito pro přivedení 5 V napájení na komunikační rozhraní. V zapojení se tak děje unipolárním P-FET tranzistorem s velmi nízkou rezistancí. Tento „spínač“ při sepnutí napájí komunikační rozhraní přímo 5 V získanými z portu USB.[3] Z uživatelského pohledu může být tedy komunikační rozhraní pracovat na napěťových hladinách 3,3 V a 5 V, nebo může být zcela vypnuto.

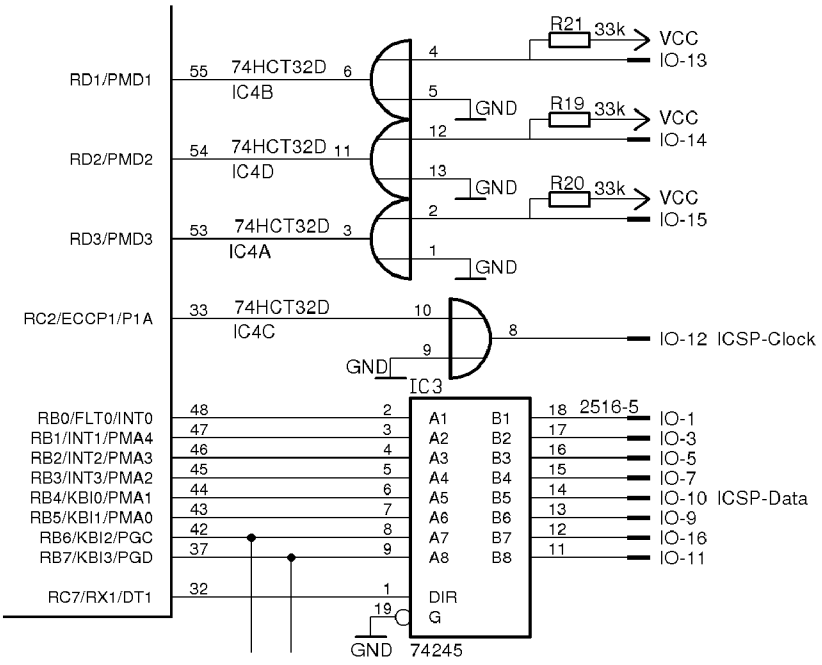
2.3 Generátor programovacího napětí

Odpovídající napěťová hladina programovacích signálů je bezesporu nutnou podmínkou pro bezpečné naprogramování cílové součástky. Mnoho součástek ovšem k programování potřebuje ještě zdroj tzv. Programovacího napětí – V_{PP}. Toto napětí bývá nejčastěji 13 V

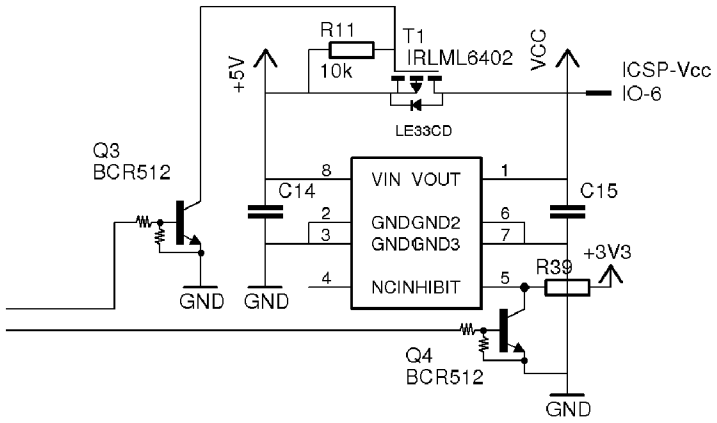
[1] Převodníky firmy Asix jsou pouze *jednosměrné* díky čemu je nelze využít pro programování součástek vyžadujících obousměrnou komunikaci po stejných vodičích – bohužel např. ICSP.

[2] **Nikdy** nepřipojujte 5 V vstupní signály, pokud je současně připojena „3,3 V only“ součástka. Dostatečně silný vstupní signál by mohl způsobit nárůst napětí na ostatních výstupních signálech a součástku poškodit!

[3] Současné sepnutí obou napěťových hladin logicky nedává smysl, z hlediska hardware je však zcela bezpečné. Na výstupu bude v takovém případě 5 V.



Obrázek 2.3: Schéma zapojení komunikačního rozhraní



Obrázek 2.4: Modul přepínače logických úrovní komunikačního rozhraní

a historicky sloužilo k napájení programovací matice paměti EPROM. Dnešní mikrokontroléry vyžadují jeho přítomnost jako podmínku pro přechod do *programovacího režimu*. Na každý pád musí být programátor schopen zajistit zdroj tohoto napětí.

2.3.1 Proudové nároky programovaných součástek

Jak bylo právě zmíněno, nové součástky s pamětí flash ve většině případů využívají programovací napětí pouze k detekci a přechodu do programovacího režimu a samotné přepisování

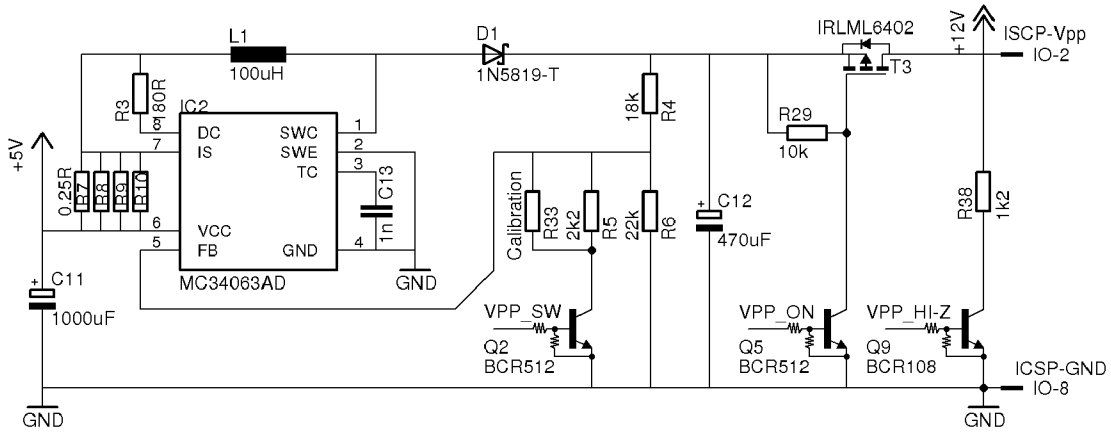
flash paměti se stará zabudovaná napěťová pumpa. Nicméně starší a OTP[4] součástky, používají programovací napětí k napájení paměťového pole a „napálení“ programu do paměti. Odběr během programování je pro mikrokontroléry PIC16C asi 35 mA. U pamětí EPROM se požadavky liší dle konkrétního typu a stáří. Programátor je schopen bezpečně a trvale dodávat až 100 mA při napětí 12 V nebo 13 V.

2.3.2 Nízkonapěťové programování

Výrobci jsou si nepříjemností se zajištěním programovacího napětí a jeho izolací vůči ostatním částem obvodu vědomi. Proto většina dnešních mikrokontrolérů umožňuje tzv. nízkonapěťové programování. V drtivé většině případů je tato možnost z výroby zapnuta. Pokud ji ale uživatel přepsáním hodnoty v příslušném registru zruší, lze součástku opět naprogramovat pouze za použití standardního programovací procesu. Tento fakt je dalším důvodem, proč musí být programátor schopen toto napětí zajistit. Samotná podpora nízkonapěťového programování navíc znamená rozšířit konektor ICSP o další vstupně-výstupní vodič – LVP. Pin pro LVP je na programátoru fyzicky přítomen, ačkoli se práce touto formou programovaní nezabývá.

2.3.3 Konstrukce DC/DC měniče

Nutnost zajistit programovací napětí, aniž by bylo nutné připojovat k programátoru síťový adaptér, vedla k osazení programátoru DC/DC měničem 2.5. Protože se velikost programovacího napětí liší podle cílové součástky, lze dle typu cílové součástky zvolit hodnotu 12 V nebo 13 V. Výchozím bodem pro konstrukci se stal článek v časopise Amatérské rádio [18], který byl upraven pro potřeby programátoru. Měnič je řízen integrovaným obvodem MC34063 [15], který je běžně dostupný. Návrh vykazuje účinnost okolo 75 %, malé zvlnění a přesnost do 2 % v celém pracovním rozsahu bez ohledu na zatížení. V obou režimech může trvale dodávat proud až 100 mA.

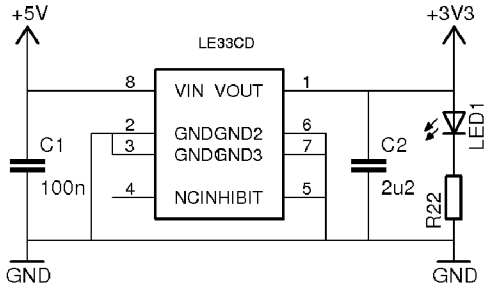


Obrázek 2.5: Schéma zapojení generátoru programovacího napětí

[4]Součástky které lze naprogramovat pouze jednou

2.4 Požadavky na napájení

V předcházejících podkapitolách byly rozebrány nároky na napájení cílové součástky. Následující text popisuje požadavky na napájení z hlediska programátoru. Ke svému chodu zařízení vyžaduje dostatečně silný zdroj stejnosměrného napětí 5 V. Jako primární zdroj je uvažován USB port osobního počítače. Z tohoto předpokladu vychází požadavek na maximální spotřebu zařízení, která nesmí v žádném stavu přesáhnout povolený limit 500 mA na jeden USB port. Nejnáročnějším spotřebičem je generátor programovacího napětí, který může při garantovaných podmínkách odebírat až 1,5 W, což je 60 % dostupných zdrojů. Podsvícení LCD a indikační diody jsou napájeny přímo 5 V a spínány pomocí digitálních tranzistorů. Hlavním důvodem pro toto zapojení je zajištění stability 3,3 V napájení pro řídící mikrokontrolér, LCD a paměťovou kartu. V celém programátoru se celkem 3× vyskytuje 3,3 V stabilizátor LE33CD [22]. Jeden slouží pro napájení obvodů výstupního rozhraní při zvolené napěťové hladině 3,3 V. Druhý napájí řídící mikrokontrolér, LCD, paměťovou kartu a slouží jako napěťová reference pro A/D převodník. Poslední je vyhrazen pro neosazený dotykový snímač Atmel AT42QT2160 který pro svou činnost vyžaduje velmi čisté napájecí napětí bez napěťových rázů. Typické zapojení stabilizátoru je na obrázku 2.6.



Obrázek 2.6: Modul stabilizátoru 3,3 V

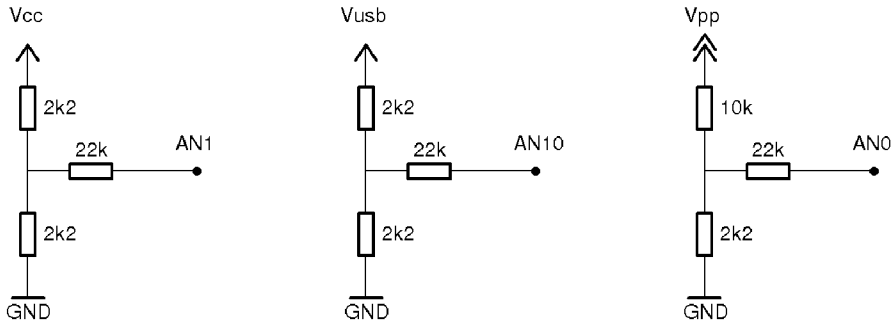
2.5 Měření aktuálních napětí

Aby bylo zajištěno spolehlivé naprogramování cílové součástky, je nutné garantovat stabilitu napájecího i programovacího napětí. Proto je programátor osazen třemi napěťovými děliči, které umožňují monitorovat aktuální napětí na portu USB – V_{USB}, stejně jako napětí V_{PP} a V_{CC} na komunikačním rozhraní. Poměry děličů jsou zvoleny s ohledem na jednoduchost zapojení. V případě V_{USB} a V_{CC} je dělící poměr zvolen 1:1, u V_{PP} je 10 : 2,2 jak je zobrazeno na 2.7. Rozlišení A/D převodníku je 10 bitů na kanál, maximální měřená napětí jsou omezena napájecím napětím procesoru a poměrem děličů. Pro V_{USB} a V_{CC} je maximální přípustná hodnota napětí 6,6 V, pro V_{PP} 15,0 V. Aktuální napětí je získáno jako výsledek A/D převodu vynásobený *magickou konstantou*, která odpovídá rozsahu děliče přizpůsobenému na míru použitým rezistorům[5].

Pokud dochází k velkému kolísání napájecího napětí, odpojte zařízení a použijte jiný (kvalitnější) kabel. Vyhněte se použití dlouhých kabelů nebo USB prodloužení. Rovněž přední USB konektory levných počítačových skříní bývají připojeny extrémně tenkými

[5]Výrobcem udávaná třída přesnosti je u použitých rezistorů 5 %.

a nekvalitními vodiči. Použití takových konektorů nelze doporučit. Pokud je dlouhé vedení nezbytné, připojte programátor do rozbočovače s vlastním napájením.

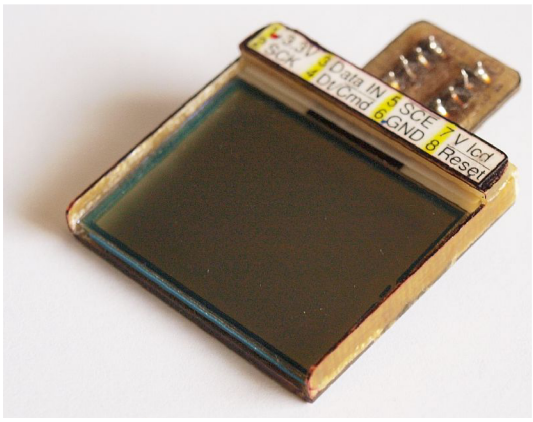


Obrázek 2.7: Zapojení děličů napětí

2.6 LCD displej

Pro zobrazování aktuálního stavu je programátor osazen grafickým LCD displejem z mobilního telefonu Nokia 3410 s fyzickým rozlišením 96 × 65 bodů a technologií FSTN 2.8. Přímo na skle displeje je osazen řadič Philips OM6206 [16], který umožňuje komunikaci s displejem po SPI rozhraní. OM6206 umožňuje řízení matice bodů o maximálních rozměrech 102 × 65. Při psaní firmware pro obsluhu LCD je nutné na tuto skutečnost pamatovat a šest chybějících sloupců přeskočit. Displej sdílí sběrnici SPI společně s paměťovou kartou, jejíž zapojení je na obrázku 2.10. Proto musí řídící mikrokontrolér před zahájením komunikace uvést pin `chip_enable` požadované periferie do logické nuly.

Řadič obsahuje zabudovanou nábojovou pumpu pro napájení zobrazovací matice. Pomocí komunikačního protokolu lze nastavit generované napětí v rozsahu od 5 V do 9 V, což má mimo jiné vliv kontrast displeje. Pumpu lze případně úplně vypnout a napájet zobrazovací matici z vnějšího zdroje (pin č.7). V opačném případě je nutné vložit mezi piny 6 a 7 kondenzátor 2,2μF .



Obrázek 2.8: LCD z telefonu Nokia 3410 v testovací patici

PIN	Název	Popis
1	V_{cc}	Napájecí napětí 3,3 V
2	SCK	Hodinový signál sběrnice SPI
3	DataIN	Vstupní data - MOSI
4	D / $\overline{C}$	Význam přijímaných dat 1 = Data, 0 = $\overline{\text{Command}}$
5	$\overline{SCE}$	$\overline{\text{ChipEnabled}}$ – umožňuje zapojit víc zařízení na SPI sběrnici
6	GND	Zem
7	V_{lcd}	Externí napájení zobrazovací matice
8	$\overline{\text{Reset}}$	Uvede řadič displeje do výchozího stavu

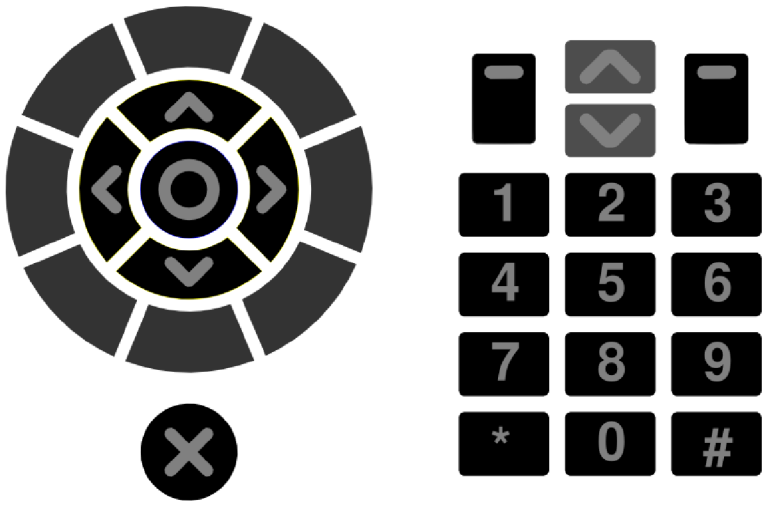
Tabulka 2.2: Popis pinů LCD displeje

2.7 Dotykové ovládání

Původním záměrem bylo osadit programátor dotykovým ovládáním se snímači umístěnými pod sklem displeje LCD a demonstrovat tak možnost elegantně vytvořit kapacitní dotykový displej. Jako problém se ukázala přítomnost slabé kovové vrstvy na spodní straně LCD.

Protože se problém nepodařilo odstranit, není v současném stavu programátor vybaven žádnými ovládacími prvky. Celý programovací proces je řízen z obslužné aplikace v PC. Nový návrh počítá s dodatečným osazením ovládacích prvků. Na vrchní desce plošných spojů jsou k tomuto účelu vyvedeny vodiče univerzálního modulu MSSP, které mohou sloužit jako sběrnice SPI nebo I^2C. Také jsou vyvedena všechna napájecí napětí včetně V_{PP} a V_{CC}.

Navrhovaný obvod Atmel AT42QT2160 [2] komunikuje po sběrnici I^2C a umožňuje připojení až 16 dotykových ploch o rozměrech od 6 × 6 do 100 × 100 milimetrů při maximální tloušťce krycího panelu 3 mm. (Tloušťka kritické části použitého LCD byla pouze 1,75 mm.) Výhledově se počítá s použitím výše uvedeného obvodu pro realizaci klasické dotykové klávesnice se dvěma možnými variantami rozložení 2.9.



Obrázek 2.9: Uvažovaná rozložení dotykových snímačů

První varianta vychází z rozložení starších mobilních telefonů Nokia, resp. modelu 7110 s rolovacím kolečkem. Základ tvoří číselná matice 4 × 3 doplněná o dvě navigační plochy pod

displejem. Poslední dvě plochy slouží pro posuv v menu. Jejich zapojení v režimu „slider“ umožňuje získat celkem ze dvou ploch celkem 4 hodnoty v závisloti na poloze prstu. Uživatel pak posuv ovládá podobným způsobem, jako by otáčel kolečkem myši.

Druhá varianta vychází z ovládacích prvků používaných na hudebních přehrávačích Apple iPod. Základem je 8 ploch v režimu „slider“ uspořádaných do kruhu, středový navigační kříž a plocha zpět, která je umístěná dole, mimo kruhové uspořádání. Finální volba varianty bude záviset na spolehlivosti a složitosti konstrukce.

2.8 Rozhraní paměťové karty Secure Digital

Jedním z nadstandardních rozšíření programátoru má být možnost práce v terénu bez připojení k PC. To bohužel v současné době kvůli absenci dotykových snímačů není možné. Nicméně toto rozšíření s sebou přináší problém s ukládáním zkopilovaných HEX souborů. Jako úložiště byla v raných fázích vývoje uvažována sériová paměť EEPROM o kapacitě 512 kB. Nakonec byl programátor, na doporučení vedoucího práce, osazen slotem pro paměťovou kartu. Návrh je připraven pro spolupráci s paměťovými kartami Secure Digital a MultiMediaCard včetně jejich derivátů, které podporují režim komunikace po sběrnici SPI 2.3. Použitý slot typu „push-push“ je vybaven kontakty pro detekci vložení paměťové karty a uzamčení proti zápisu. Oba kontakty jsou aktivní v logické nule a proti zákmitům vybaveny slabými pull-up rezistory. Povedené shrnutí užitečných informací o používání SD a MMC karet v režimu SPI lze nalézt v [3].

Typ karty	**Podpora SPI**
SD	Ano
SDIO	Ano
MiniSD	Ano
MicroSD	Volitelně
MMC	Volitelně
RS–MMC	Volitelně
MMC Plus	Volitelně
Secure MMC	Ano

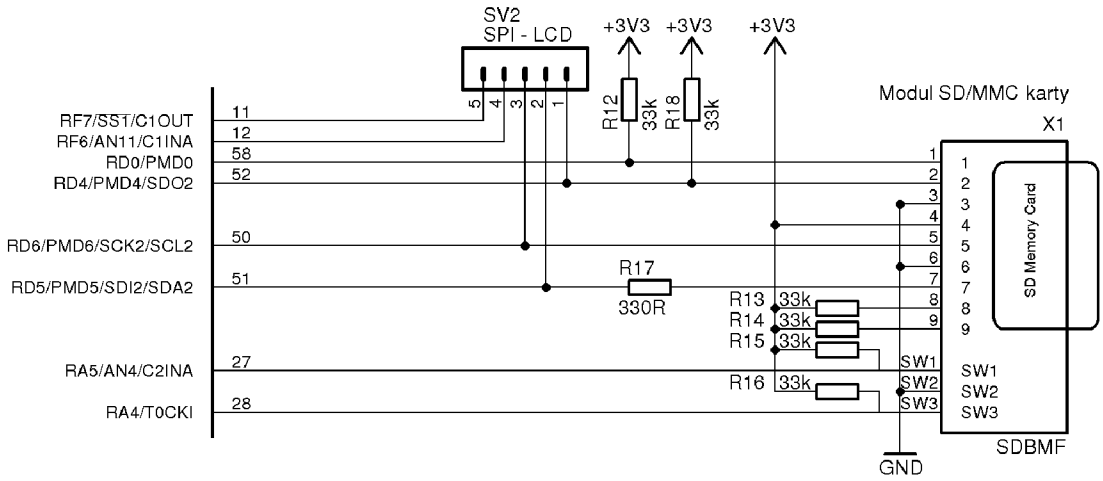
Tabulka 2.3: Podpora SPI u SD a MMC karet [25]

2.8.1 Zapojení sběrnice SPI

Řídící mikrokontrolér programátoru obsahuje dva univerzální moduly pro sériovou komunikaci – MSSP. Protože je programátor vybaven periferiemi, které používají sběrnice I^2C i SPI, sdílí paměťová karta a LCD displej společnou sběrnici. S tímto faktem je spojená nutnost zvolit periferii, se kterou chceme komunikovat. Výběr periferie se děje pomocí uvedení jejího tzv. *Chip-enabled* pinu do logické nuly. Zapojení sběrnice včetně pull-up rezistorů a konektoru pro LCD displej je znázorněno na obrázku 2.10.

2.9 Programování samotného programátoru

Vzhledem k faktu, že je programátor řízen mikrokontrolérem, dochází zde k problému „slepice – vejce“, jinými slovy, řídící mikrokontrolér programátoru je potřeba naprogramovat



Obrázek 2.10: Schéma zapojení sběrnice SPI a paměťové karty

jiným programátorem. K tomuto účelu v průběhu vývoje sloužil zapůjčený programátor Asix PRESTO.

2.9.1 ICSP rozhraní řídícího mikrokontroléru

Protože řídící mikrokontrolér PIC18F67J50 podporuje pouze 3,3 V programování, bylo nutné tento fakt zohlednit při návrhu desky plošných spojů. Značné množství programátorů totiž neumožňuje obousměrné programování na této napěťové hladině. Maximální přípustné hodnoty pro řídící mikrokontrolér, jsou společně se stavem po úpravě (U_{new}) uvedeny v tabulce 2.4. Provedená úprava je popsána v sekci 2.11 a vychází z předpokladu, že uživatel nemá k dispozici programátor schopný obousměrné komunikace při 3,3 V.

PIN	Funkce	U_{min}	U_{max}	U_{new}	Popis
1	$\overline{\text{MCLR}}$	0,8	3,6	5,5	Vstup do programovacího režimu, RESET
2	–	–	–	–	Klíč
3	VDD	3,0	3,6	5,5	Napájecí napětí
4	GND	0,0	0,0	0,0	Zem
5	PGD	3,0	3,6	5,5	ICSP Data
6	PGC	3,0	3,6	5,5	ICSP Clock

Tabulka 2.4: Maximální napětí pro programování programátoru

2.9.2 Napěťové omezení pro 5 V programátory

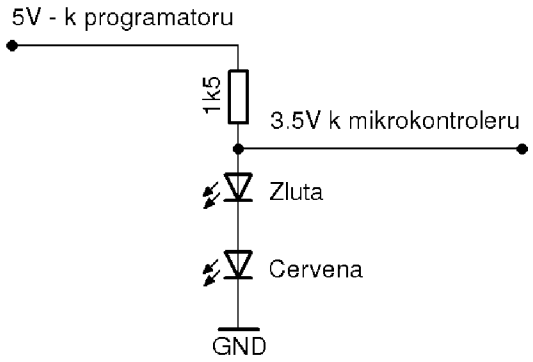
Původní návrh počítal s využitím sériového zapojení Zenerovy diody s rezistorem, nevedl ale použitelnému řešení. Zenerovy diody se závěrným napětím 3,3 V i 3,6 V se projevily jako nedostatečně stabilní napěťová reference a v obou případech pohltily přijímaný signál. nehledě na fakt, že nárůst napětí na pinech mikrokontroléru dosahovalo už při vstupním napětí 5,2 V kritických hodnot $\geq$3,6 V.

Použité zapojení napěťového omezení je založeno na úbytku napětí na LED diodách, které se liší podle jejich barev. Empiricky bylo zjištěno, že sériové zapojení červené a žluté diody vyžaduje k rozsvícení napětí v rozsahu 3,49 V – 3,55 V. (Toto napětí se liší pár od páru, z většího množství diod však není problém vybrat vhodné páry.) Stabilizátor LE33CD, který napájí mikronotrolér stabilizuje napětí na 3,32 V. Tento stav umožňuje programovat programátor pomocí 5 V signálů většiny dostupných programátorů.

Pozor, pin $\overline{\text{MCLR}}$ řídícího mikrokontroléru **nepoužívá napětí 13 V, ale pouze 3,3 V**. Připojení 13 V na $\overline{\text{MCLR}}$ by mělo fatální následky! Toto se týká všech nových mikrokontrolérů PIC18F**J**. Jádrem úpravy jsou tři celky skládající se vždy z dvojice LED diod v sériovém zapojení společně s rezistorem 1,5 kΩ.

2.9.3 Princip činnosti

Jak je patrné ze schématu 2.11, při komunikaci směrem do programátoru (zápis dat) je 5 V signál přiveden na ICSP rozhraní programátoru. Velikost napětí je dostatečná k rozsvícení dvojice diod na kterých vznikne úbytek 3,52 V ± 0,05 V. Toto napětí je použito k reprezentaci logické úrovně pro signály vstupující do mikrokontroléru. Probíhá-li komunikace v opačném směru (čtení dat), je signál z mikrokontroléru o napětí 3,30 V spojen přes rezistor s pomocným programátorem. Napětí 3,30 V k rozsvícení LED diod nestačí, ale je dostatečné pro reprezentaci logické úrovně **1** i v případě 5 V logiky. Řešení je bez problémů použitelné do frekvence 250 kHz, při vyšší frekvenci začíná docházet k deformaci náběžné hrany signálu. Přes svou jednoduchost však zapojení s LED vykazuje nejlepší parametry ze všech testovaných zapojení.

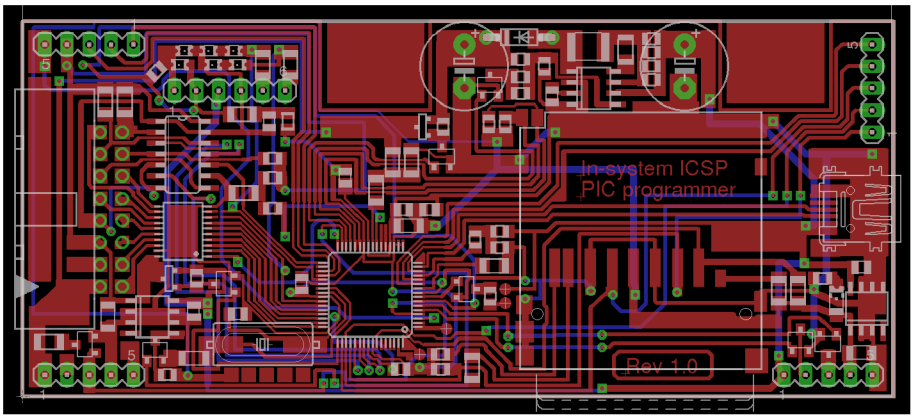


Obrázek 2.11: Zapojení převodníku pro naprogramování 5 V signály

2.10 Návrh desek plošných spojů

Obě desky plošných spojů programátoru jsou navrženy v programu Eagle. Jejich rozměry jsou 100 × 55 milimetrů a ačkoli jsou navrženy jako dvouvrstvé, projevila se zde snaha o maximální využití pouze jedné vrstvy 2.12. Při návrhu bylo potřeba rozšířit základní sadu knihoven programu Eagle o některé součástky. Dodatečně byla použita knihovna obvodu MC34063 od P. Szramowskeho [23]. Na jejím základu byla vytvořena knihovna pro obvod LE33CD, který se vyrábí ve stejných pouzdrech. Součástky jako řídící mikrokontrolér, slot pro paměťovou kartu nebo miniUSB konektor jsou přítomny v knihovnách dodaných spo-

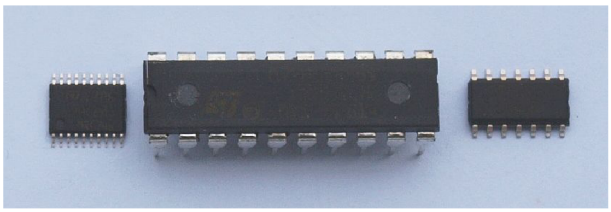
lečně s instalací. Pouze u obvodu 74xx245 bylo nutné doplnit pouzdro TSSOP 20 viz. 2.13 a v knihovně součástek Microchip byl upraven vzhled schématické značky řídícího mikrokontroléru pro potřeby schématu. Všechny upravené knihovny jsou společně se schématy dostupné na přiloženém CD.



Obrázek 2.12: Návrh řídící desky v programu Eagle

V průběhu návrhu desky došlo k problémům s napájením jádra mikrokontroléru, neboť jsem si nesprávně vyložil funkci dvojice pinů ENVREG a V_{CAP}. Tento problém pomohly vyřešit stránky bulharské firmy Olimex [13] která se zabývá výrobou vývojových kitů. Kit s označením PIC-3310 [14] obsahuje stejný mikrokontrolér jako navrhovaný programátor, slot na paměťovou kartu a podobné LCD, ovšem s jiným řadičem.

Po opakované konfrontaci obou zapojení s katalogovým listem řídícího mikrokontroléru byla chyba odstraněna a deska programátoru po vzoru uvedeného kitu doplněna o vyhlazovací $100nF$ kondenzátory umístěné v blízkosti napájecích pinů mikrokontroléru a pull-up rezistory i pro nepoužívané piny paměťové karty.



Obrázek 2.13: Velikosti pouzder TSSOP 20, DIP 20 a SO 16

Kapitola 3

Obslužný firmware programátoru

V této kapitole je popsán návrh firmwaru pro řídící mikrokontrolér. Hlavním úkolem firmware programátoru je převod přijatých dat na řízení programovacího procesu pomocí signálů ICSP Clock a ICSP Data. Základní popis protokolu ICSP, který slouží k tomuto účelu, lze nalézt v dokumentu AN910 [19]. Tento dokument je pro vývoj firmware nepochybně nejdůležitější, neboť shrnuje popis protokolu ICSP, který běžný programátor při práci s mikrokontroléry PIC nepotřebuje. Pro programování například nelze použít generátor hodinového signálu s pevnou frekvencí, protože je v průběhu procesu nutno hodiny pozastavovat.

3.1 Microchip C 18 compiler LITE

Firma Microchip dodává pro své produkty vývojové prostředí MPLAB® které umožňuje vývoj firmware pro všechny produkty. Program je provázán s některými programátory a dává možnost používat jak assembler, tak kompilátory jazyka C včetně distribucí třetích stran. Firmware pro řídící mikrokontrolér je napsán v jazyce C pro překladač Microchip C 18 (dále jen MCC18), který je ve verzi LITE volně k dispozici. Výhodou jsou dostupné knihovny a velké množství konkrétně zaměřených příkladů. Pro základní nastavení prostředí MPLAB a překladače MCC18 jsem využil článku [1]. Výchozím bodem pro implementaci se staly zdrojové kódy firmwaru pro emulaci sériového portu, které jsou součástí dokumentu AN956 [17]. Ty obsahují několik příkladů komunikace se zaměřením na vývojové kity firmy Microchip. Tyto zdrojové kódy, jejichž autorem je stejně jako v případě dokumentu Rawin Rojvanit, jsou využity například i v demonstrační aplikaci firmy Olimex [13] pro vývojový kit [14].

Pro potřeby programátoru byly zdrojové kódy značně odlehčeny. Byla odstraněna podpora vývojových kitů a všechny příklady. Nastavení napájení a zapínání USB modulu bylo přizpůsobeno řídící desce programátoru, která není vybavena konektorem pro externí napájení a není proto nutné detekovat připojení konektoru USB. V části identifikace zařízení byly nahrazeny názvy produktu a výrobce, identifikátory vendor ID a product ID ale zůstaly zachovány kvůli možnosti použít jako ovladače INF soubory dodané společně s firmwarem.

3.1.1 Hlavní smyčka aplikace

Základem firmwaru je jako u většiny vestavěných systémů nekonečná hlavní smyčka aplikace. Do té firmware vstoupí po jednorázové inicializaci desky, během které jsou nakonfigurovány všechny periferie a aktivován USB modul. Protože není možné bezpečně zajistit,

aby příjem dat po sériové lince probíhal pouze v případě, kdy je volána odpovídající funkce, je všechna funkcionalita programátoru implementována jako neblokující a po dokončení libovolné činnosti jsou nastaveny příslušné flagy. Vstupní buffer pro příjem dat má kapacitu 64 znaků. Funkce, které nejsou schopny dokončit svoji činnost v rámci jedné datové zprávy[1], udržují svůj stav pomocí statických proměnných. Tento systém umožňuje periodicky obsluhovat všechny periferie programátoru aniž by z uživatelského pohledu docházelo k „zamrzání“, nicméně způsobuje problémy při implementaci funkcionality citlivé na přesné dodržení časových intervalů.

3.1.2 Obsluha USB modulu

Na rozdíl od obvodů FTDI, které lze jednoduše nastavit pro emulaci nejčastějších rozhraní, je vestavěný USB modul mikrokontroléru PIC18F67J50 před naprogramováním zcela „hloupý“. Tento fakt s sebou nese nevýhodu v podobě složitější práce při vývoji firmware. Na druhou stranu ale dává programátorovi plnou kontrolu nad USB rozhraním, včetně detailů jako nastavení maximálního odběru, typu napájení, verze rozhraní a rychlosti. Tabulka poskytovaných informací je součástí přílohy B.3. Dle standardu USB nesmí zařízení připojené ke sběrnici v klidovém stavu odebírat z datových vodičů proud. Proto ze USB modul vybaven vestavěnými pull-up rezistory, které je možné nastavením příslušných registrů ovládat.

3.2 Formát Intel HEX

Data pro cílovou součástku jsou uložena v PC ve formátu Intel HEX. Jde o textový ASCII soubor, kde jednotlivé dvojice znaků reprezentují 1 bajt dat v hexadecimální soustavě.
Každý řádek je uvozen znakem `':'`, za kterým následuje délka dat, offset adresy, typ záznamu a samotná data doplněná o kontrolní součet. Protože programátor sám řídí programovací proces, je zdrojový kód přenášen po sériové lince na vyžádání. Po inicializaci a zahájení programovacího procesu programátor posílá do PC požadavek o zaslání dalšího řádku souboru. Jednotlivé řádky jsou zasílány formou 64 bytových zpráv, jejichž formát je popsán v části 4.3. Ukázka zkompilovaného kódu ve formátu Intel HEX je v tabulce 3.1.
Význam přenášených dat určuje pole typ. Při programování mikrokontrolérů PIC se setkáváme se čtyřmi typy záznamů. Záznam **00** obsahuje data, záznam **01** označuje konec souboru, segmentový záznam má hodnotu **02** a rozšířený adresový záznam **04**. Nejvíce významné bity jsou uvedeny jako první (MSb first). Detailní popis protokolu lze nalézt v [6]. Některé programy pro práci se soubory Intel HEX přidávají do obsahu souboru různé poznámky, identifikaci cílové architektury, typ součástky atp. Z tohoto důvodu jsou všechny řádky, které nezačínají dvojtečkou přeskočeny. Toto chování není upraveno žádnou normou, nicméně je implicitní pro většinu editorů.

3.3 Protokol ICSP

Pochopení a implementace programovacího protokolu ICSP je nejdůležitější částí této práce. Přestože by měl být základ protokolu napříč spektrem osmibitových mikrokontrolérů podobný, existuje v době psaní této práce celkem 47 skupin, které se vzájemně liší programovacími specifikacemi. Detailní popis je vždy součástí dokumentu „Programming Specifi-

[1] Typicky všechny operace, které souvisí s programováním cílové součástky.

Délka	Offset	Typ	Data	Kontrolní součet
:02	00 00	**04**	00 00 **FA**	
:10	00 00	**00**	83 01 8B 01 9F 01 83 16 C8 30 81 00 47 30 9F 00	**18**
:10	00 10	**00**	08 11 88 10 83 12 08 15 1F 20 08 11 10 20 0B 28	**C2**
:10	00 20	**00**	A0 01 A1 01 0A 30 A2 00 A0 0B 14 28 FF 30 A0 00	**FB**
:10	00 30	**00**	A1 0B 14 28 FF 30 A1 00 A2 0B 14 28 08 00 A0 01	**76**
:10	00 40	**00**	A1 01 0A 30 A2 00 88 14 A0 0B 23 28 7F 30 A0 00	**51**
:10	00 50	**00**	88 10 A0 0B 29 28 7F 30 A0 00 88 10 A1 0B 23 28	**2E**
:0A	00 60	**00**	FF 30 A1 00 A2 0B 23 28 08 00	**C6**
:04	30 00	**00**	00 34 00 34	**64**
:02	40 0E	**00**	02 2F **7F**	←*Konfigurační slovo*
:00	00 00	**01**	**FF**	

Tabulka 3.1: Ukázková aplikace pro PIC16F877A ve formátu Intel HEX

cation“ daného mikrokontroléru. Rozdělení do skupin je uvedeno v příloze B.2 Naštěstí jsou dokumeny s programovacími specifikacemi (např. [11]) psány a řazeny podobným způsobem, a tak je možné potřebné informace nalézt poměrně rychle. Přehledově jsou informace rozděleny do pěti kapitol. (Detailně viz. strana 9 v [19].)

- **Overview** – Rozložení vývodů součástky, požadavky na napájení, použitý programovací protokol (většinou ICSP).
- **Program Memory Programming and Verification** – Uspořádání paměti, sada příkazů a specifických algoritmů pro programování.
- **Configuration Word, Device IDs and ID Locations** – Umístění konfiguračního slova ID zařízení a ID součástky v paměti. Tyto adresy nejsou běžně dostupné.
- **Code Protection and Checksum** – Popis funkce pojistek proti zcizení kódu a princip výpočtu kontrolního součtu.
- **AC/DC Specifications** – Požadavky pro napájení při programování, podporované příkazy a časování protokolu.

Z výše uvedených skutečností vyplývá, že podpora kompletního spektra součástek rodin 16F a 18F by vyžadovalo nastudování rozdílů všech těchto mikrokontrolérů, jejich zakoupení a ověření funkčnosti programovacího procesu. Proto se dále budeme zabývat implementací ICSP rozhraní se zaměřením na mikrokontrolér PIC16F877A který bude součástí prezentované demo aplikace 3.3. Kapitola vychází z aplikačních poznámek AN910 [19] a katalogového listu zmíněného mikrokontroléru [9]. Uvedené diagramy 3.1, 3.2 jsou převzaty z těchto dokumentů. Nejprve však popíšeme odlišné způsoby programování pro jádra 16F a 18F.

3.3.1 Přechod do programovacího režimu

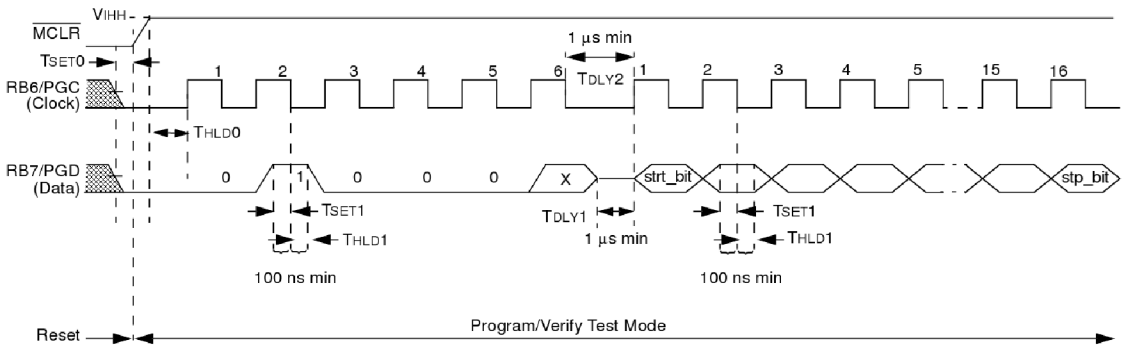
Způsob, jakým součástka pozná, že má místo zahájení provádění programu přejít do programovacího režimu se skládá ze tří kroků, které musí být provedeny v následujícím pořadí:

1. Připojení napájení na všechny piny V_{SS} a V_{DD}.
2. Zvýšení programovacího napětí V_{PP} pinu $\overline{\text{MCLR}}$ na úroveň V_{IHH}.
3. Uvedení pinů PGC a PGD do logické nuly po specifikovanou dobu.

Tato sekvence způsobí, že je místo vykonávání kódu aktivován konečný automat, který slouží pro řízení programování součástky. Adresovatelný prostor paměti je zvětšen na dvojnásobek, což zpřístupní konfigurační registry a další speciální vlastnosti součástky, které jsou zde namapovány. Dále dojde k nastavení programového čítače na hodnotu 0000 a součástka očekává příjem instrukce.

3.3.2 Microchip PIC12F a PIC16F

Rodiny 12F a 16F jsou založeny na 14 bitovém jádře a používají k programování speciální instrukce s 6 bitovým operačním kódem, které nejsou v běžném režimu dostupné. Po odeslání instrukce následuje prodleva v hodinovém signálu, typicky 1 μs. Jejím účelem je oddělit instrukci od datových bitů. Těch je přenášeno vždy 16 bez ohledu na architekturu jádra. Přebytek bitů je řešen obalením 14 bitového slova nulami, přičemž platí pravidlo jedné počáteční nuly následované daty doplněnými o zbývající počet nul. Pokud je tedy programována vestavěná paměť EEPROM, je za počáteční nulou odesláno 8 datových bitů a 7 nul, v případě 14 bitové paměti programu je na konci nula pouze jedna. Logická hodnota je čtena ze signálu PGD při *sestupné* hraně hodinového signálu PGC. V případě nedodržení tohoto pravidla hrozí nechtěné posunutí významu přenášených dat o bit vlevo se všemi důsledky, které z takového chování mohou vyplynout. Například uzamčení součástky pomocí pojistek CP a CPD. Tabulka instrukcí 3.2 platí pro mikrokontroléry s flash pamětí, pro OTP součástky se význam instrukcí liší!



Obrázek 3.1: Příklad 6 bitové instrukce zápisu paměti programu u PIC16F87XA [19]

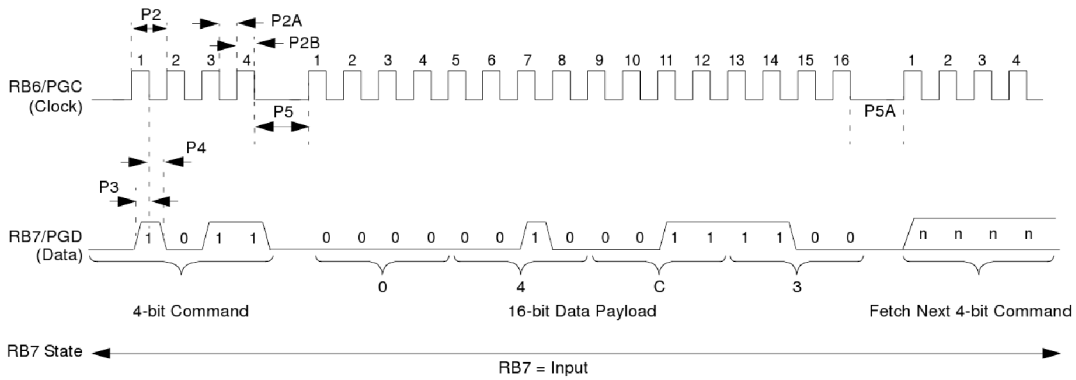
Instrukce	Operační kód						Formát následujících dat
Zápis konfiguračního slova	0	0	0	0	0	0	0,(data (0:13)),0
Zápis paměti programu	0	0	0	0	1	0	0,(data (0:13)),0
Čtení paměti programu	0	0	0	1	0	0	0,(data (0:13)),0
Inkrementace adresy	0	0	0	1	1	0	–
Start mazání a pak programování	0	0	1	0	0	0	–
Start programování (pouze)	0	1	1	0	0	0	–
Zápis paměti EEPROM	x	x	0	0	1	1	0,(data (0:7)),0000000
Čtení paměti EEPROM	x	x	0	1	0	1	–
Vymazání paměti programu	x	x	1	0	0	1	–
Vymazání paměti EEPROM	x	x	1	0	1	1	–
Konec programováním	0	0	1	1	1	0	–
Vymazání součástky	x	1	1	1	1	1	–

Tabulka 3.2: Operační kódy 6 bitových programovacích instrukcí [19]

3.3.3 Microchip PIC18F

Programování mikrokontrolérů řady PIC18F probíhá zcela odlišným způsobem. Instrukce používají 4 bitový operační kód. Pro čtení i zápis se používají klasické instrukce `TABLE READ` a `TABLE WRITE` které umožňují přímý přístup k paměti programu. Instrukce čtení a zápisu jsou dostupné i ve verzích s možností inkrementovat nebo dekrementovat ukazatel paměti, jak je naznačeno v tabulce 3.3 pomocí syntaxe jazyka C.

Instrukce `NOP` má v programovacím režimu speciální význam. Data zaslaná za operačním kódem instrukce `NOP` jsou interpretována jako libovolná instrukce z celé instrukční sady a okamžitě provedena. K zápisu dat do daného místa v paměti je potřeba 16 hodinových cyklů. Naproti tomu k přečtení paměťového místa stačí pouze 8 cyklů! Je to způsobeno rozdílným chováním flash paměti při čtení a zápisu. Data jsou stejně jako v případě rodiny 16F čtena i zapisována při sestupné hraně hodinového signálu.



Obrázek 3.2: Příklad 4 bitové instrukce pro mikrokontroléry PIC18Fxx20 [19]

Instrukce	Operační kód				Formát následujících dat
Vynucení instrukce `NOP`	0	0	0	0	instrukce (0:15)
Vysunutí hodnoty `TABLAT`	0	0	1	0	data (0:15)
`TABLE READ`	1	0	0	0	data (0:15)
`TABLE READ` , pointer++	1	0	0	1	data (0:15)
`TABLE READ` , pointer−−	1	0	1	0	data (0:15)
`TABLE READ` , ++pointer	1	0	1	1	data (0:15)
`TABLE WRITE`	1	1	0	0	data (0:15)
`TABLE WRITE`, pointer++	1	1	0	1	data (0:15)
`TABLE WRITE`, pointer−−	1	1	1	0	data (0:15)
`TABLE WRITE`, ++pointer	1	1	1	1	data (0:15)

Tabulka 3.3: Operační kódy 4 bitových programovacích instrukcí [19]

3.4 Přidání nové součástky – PIC16F877A

V této části práce je popsán postup přidání podpory pro programování nové součástky. Jako příklad je zvolen mikrokontrolér PIC16F877A který je použitý v ukázkové aplikaci 3.3. Prvním krokem nutným pro rozšíření podpory je shromáždění informací potřebných k návrhu konečného automatu pro řízení protokolu. Výchozí chování protokolu popsaného v kapitole USB komunikace je založeno na odesílání zdrojového souboru ve formátu Intel HEX řádek po řádku do vstupního bufferu programátoru. Zde jsou data interpretována a použita k řízení programovacího procesu. Programovací proces se liší dle typu součástky, typu její paměti a organizace. Pokud zanedbáme drobné rozdíly, lze mikrokontroléry PIC rozdělit na tři skupiny. První z nich jsou OTP součástky s pamětí EPROM. U těchto součástek, které lze naprogramovat pouze jednou, je časování zápisu do paměti řízené z externího zdroje. Součástky vybavené pamětí flash jsou obvykle doplněny o mechanismus „self-timing array“, který sám řídí časování zápisu jednotlivých buněk a programátor je zodpovědný pouze za správné naplnění vstupního bufferu a dodržení čekacích intervalů. Poslední skupinou jsou mikrokontroléry s pamětí flash, které ovšem nejsou mechanismem automatického časování vybaveny.

3.4.1 Parametry nové součástky

Pro přidání nové součástky je nutné dodržet všechna napájecí a komunikační hlediska. Katalogový list součástky [9] většinou neobsahuje potřebné údaje. Ty jsou zveřejňovány ve zvláštních dokumentech nazvaných „Programming Specifications“. Tento dokument [11] shrnuje elektrické požadavky programovacího procesu, umístění konfiguračního slova, prostor pro zápis ID, soupis programovacích instrukcí, požadavky na časové prodlevy hodinového signálu a další parametry, který je nutné dodržet.

3.4.2 Parametry PIC16F87XA

Výběr nejpodstatnějších parametrů je shrnut v následujícím výčtu.

- PIC16F – 14 bitové jádro
- Při programování musí být součáska napájena napětím v rozmezí 4,5 V – 5,5 V.
- Programovací napětí V_{PP} je 13 V.

- Pro uživatelská ID jsou vyhrazeny adresy 2000h – 2003h, pro ID zařízení 2006h.
- Konfigurační slovo leží na adrese 2007h, jeho výchozí hodnota je 3FFFh.
- Data jsou čtena při sestupné hraně hodinového signálu.
- Prodleva pro zápis slova $T_{Prog1} = 1\,ms$, zápis probíhá po osmi slovech

V tabulce 3.2 byly uvedeny operační kódy programovacích instrukcí. Tabulka je sestavena pro obecný mikrokontrolér rodiny PIC16, proto jsou některá pole označena písmenem `'x'`. Kompletní tabulka je vždy součástí dokumentu „Programming specifications" příslušného mikrokontroléru. V tabulce jsou rovněž uvedeny rozsahy napětí pro jednotlivé instrukce. Tabulka 3.4, platná pro PIC16F877A byla převzata z [11].

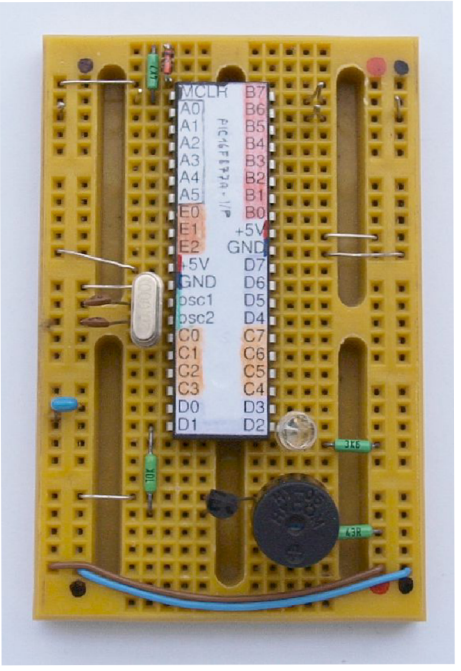
Instrukce	**Operační kód**					**Data**	**Rozsah napětí**
Zápis konfiguračního slova	0	0	0	0	0	0,(data (0:13)),0	2,2 V – 5,5 V
Zápis paměti programu	0	0	0	1	0	0,(data (0:13)),0	2,2 V – 5,5 V
Čtení paměti programu	0	0	1	0	0	0,(data (0:13)),0	2,2 V – 5,5 V
Inkrementace adresy	0	0	1	1	0	–	2,2 V – 5,5 V
Start mazání a pak programování	0	1	0	0	0	*čekej 4* ms_{int}	2,2 V – 5,5 V
Start programování (pouze)	1	1	0	0	0	*čekej 1* ms_{ext}	4,5 V – 5,5 V
Vymazání paměti programu	0	1	0	0	1	– $_{int}$	4,5 V – 5,5 V
Vymazání paměti EEPROM	0	1	0	1	1	– $_{int}$	4,5 V – 5,5 V
Vymazání součástky	1	1	1	1	1	*čekej 4* ms_{int}	4,5 V – 5,5 V
Zápis paměti EEPROM	0	0	0	1	1	0,(data (0:13)),0	2,2 V – 5,5 V
Čtení paměti EEPROM	0	0	1	0	1	0,(data (0:13)),0	2,2 V – 5,5 V
Konec programováním	1	0	1	1	1	–	–

Tabulka 3.4: Operační kódy mikrokontrolérů PIC16F87XA [11]

Uvedené operační kódy instrukcí uvedené v tabulce jsou ve svém důsledku pouze 5 bitové. Zleva první chybějící, tj. v pořadí šestý odesílaný bit instrukce je v případě všech instrukcí tzv. „don't care", jeho hodnota nemá na průběh programování vliv a proto není v tabulce uveden. U některých instrukcí jsou navíc uvedeny indexy *int* a *ext* které značí, zda jsou následující hodinové signály pro instrukci generovány vnitřně nebo je třeba je generovat ze strany programátoru.

3.4.3 Adresový prostor

V normálním režimu je adresový prostor mikrokontroléru 8192 14 bitových slov. To odpovídá adresám 0000h – 1FFFh. V programovacím režimu je adresový prostor dvojnásobný. Do nově přístupných adres 2000h – 3FFFh je namapováno konfigurační slovo, všechna ID, ale i 256 bajtů paměťi EEPROM (2100h – 21FFh), ke které lze za běhu programu přistupovat pouze jako k periferii.



Obrázek 3.3: Ukázková aplikace s PIC16F877A

Kapitola 4

Uživatelský software pro PC

4.1 Obslužná aplikace

Obslužná aplikace slouží k fyzickému nahrání zkompilovaných souborů ve formátu Intel HEX do cílové součástky. Aplikace je napsána pomocí více-platformního frameworku Qt4 [12]. Samotné použití tohoto frameworku ovšem není schopno zajistit přenositelnost aplikace. Je to způsobeno rozdílným přístupem k sériovým portům na jednotlivých platformách. V projektu proto byla použita třída QextSerialPort [7], která zajišťuje jednotný přístup k sériové komunikaci jak v systémech splňujících standard POSIX[1], tak v systémech Microsoft Windows. Verze knihovny 1.1, vyvíjená pro Qt 4.1 a zveřejněná 25. března 2007, vyžadovala drobné úpravy v názvech tříd a pojmenování hlavičkových souborů. Tento jev byl patrně způsoben novou (přísnější) verzí překladače GCC.

Aplikace umožňuje otevření více zdrojových souborů pomocí záložek. V horní liště jsou dostupné základní operace pro práci se soubory, editaci a obsluhu programovacího procesu. V panelu na pravé straně jsou zobrazována aktuální napětí na klíčových místech programátoru a možnost ručního ovládání napájecích pinů mimo programovací proces. Data jsou do programátoru přenášena v textové podobě řádek po řádku. Typ zpráv a řízení komunikace bude popsáno detailněji.

4.2 Podporované systémy

Pro zajištění podpory napříč operačními systémy je u vyvíjeného zařízení vhodné použít standardní komunikační protokol. Proto využívá vestavěný USB modul mikrokontroléru třídu CDC (communications device class). Tato třída, jejíž specifikaci vydalo USB Implementers Forum [24], definuje jednotný přístup pro zařízení, která využívají USB rozhraní pro emulaci sériové komunikace. Dodržení tohoto standardu je dobrým předpokladem pro bezproblémovou podporu v drtivé většině operačních systémů.

[1] Zkratka pro „Portable Operating System Interface“. Třída QextSerialPort oficiálně podporuje následující posixové systémy: Linux, FreeBSD, SunOS/Solaris, HP-UX, SGI/IRIX a Digital UNIX.

4.2.1 Linux

Ovladače pro CDC jsou součástí linuxového jádra od verze 2.6.0. Programátor byl ve všech testovaných systémech[2] automaticky detekován, nakonfigurován a okamžitě připraven k použití. V systému se objeví nové znakové zařízení, typicky `/dev/ttyACMx`, kde `x` je číslo od 0 do N. Pokud je programátor odpojen v průběhu komunikace, je při opětovném připojení vytvořeno nové zařízení. Základní informace o konfiguraci lze zjistit příkazy uvedenými v 4.1, detailní výpis konfigurace (`lshal`) je uveden v příloze B.3.

```
pidi@natalka:/$ uname -a
Linux natalka 2.6.31-21-generic #x86_64 GNU/Linux

pidi@natalka:/$ lsusb | grep Microchip
Bus 004 Device 051: ID 04d8:000a Microchip Technology, Inc.

pidi@natalka:/$ ls -hl /dev/ | grep ttyACM
crw-rw----  1 root dialout   166,   0 2010-04-23 18:52 ttyACM0
```

Obrázek 4.1: Identifikace programátoru v systému Linux

4.2.2 Microsoft Windows

Ovladače potřebné k podpoře CDC jsou standardní součástí operačního systému Windows XP, Vista i 7. Kvůli rozdílným identifikátorům zařízení je nutné poskytnout systému upravený INF soubor. Microchip poskytuje INF soubory pro 32 bitové verze systémů XP a Vista, přičemž ovladače pro Vistu jsou funkční i v systému Windows 7. Programátor je systémem interpretován jako virtuální komunikační port `COMx` kde `x` je číslo od 0 do 255 přiřazené v závislosti na konkrétním portu USB a přítomnosti fyzických portů COM v systému.

4.2.3 Mac OS X

Programátor byl otestován v emulovaném prostředí systému Snow Leopard 10.6.3. Po připojení byl programátor rozeznán jako USB modem a nabídnuta možnost konfigurace nového síťového zařízení. (Tato skutečnost nemusí být nutně špatně, původním záměrem CDC je právě podpora modemů komunikujících po sériové lince.) V systému se objevilo nové znakové zařízení `/dev/tty.USBmodem221`. Komunikaci s programátorem se však nepodařilo navázat. Tato skutečnost však mohla být způsobena nedostatkem zkušeností s touto platformou.

4.3 Příkazová řádka a komunikační protokol

Pro přenos dat a stavových informací po sériové lince bylo nutné navrhnout a implementovat komunikační protokol mezi programátorem a aplikací v PC. Protokol musí být schopen v jednom cyklu dodat dostatečné množství dat pro zápis alespoň jednoho bloku paměti cílové součástky a dostatečně často informovat aplikaci o svém stavu.

[2] Testováno na systémech Ubuntu 9.04 / 9.10 / 10.04, Debian 5.0.3 a Gentoo v 32 bitových i 64 bitových verzích

Po problémech s vyrovnávacími buffery na straně programátoru padla volba na protokol typu Otázka – Odpověď s výměnou zpráv o pevné délce 64 bajtů. Prvních 8 bajtů identifikuje příkaz, následujících 53 bajtů složí pro data, zakončení zprávy pomocí `\r\n\0` je nepovinné, ale obecně vhodné. Protokol je navržen tak, aby byla jeho textová podoba srozumitelná i zvídavému uživateli. Jednotlivé zprávy jsou rozepsány v následujících tabulkách 4.1 a 4.2. Příklad komunikace je uveden v tabulce 4.3.

Příkaz	**Data** **celkem 64 bajtů**
DISCOVER	:\r\n\0
SELECTIC	:PIC16F877A\r\n\0
VOLTAGES	:Voltage set: Vcc=5 Vpp=12 \r\n\0
WRITE-IC	:\r\n\0
READ--IC	:\r\n\0
VERIFYIC	:\r\n\0
ERASE-IC	:\r\n\0
DATA----	:1000000083018B019F018316C830810047309F0018\r\n\0

Tabulka 4.1: Příkazy komunikačního protokolu

Příkaz	**Data** **celkem 64 bajtů**
+OK-----	:In-Circuit ICSP PIC Programmer Rev:0.5 READY!\r\n\0
+OK-----	:Selected device: PIC16F877A\r\n\0
+OK-----	:Voltage status: Vusb=5143mV Vcc=5122mV Vpp=12002mV\r\n\0
+OK-----	:Entering programming mode, waiting for data...\r\n\0
+OK-----	:Entering reading mode, get ready for data...\r\n\0
+OK-----	:Entering verify mode, waiting for data...\r\n\0
+OK-----	:Chip Erase comleted sucessfully...\r\n\0
+OK-----	:1000000083018B019F018316C830810047309F0018\r\n\0

Tabulka 4.2: Odpovědi komunikačního protokolu

Při komunikaci s programátorem pomocí terminálu je nutné dodržet délku a formát zprávy. Zpráva je ke zpracování předána jako celek ve chvíli, kdy je dosaženo počtu 64 odeslaných znaků. Pokud je počet odeslaných znaků vyšší, jsou nadbytečné znaky interpretovány jako začátek další zprávy, případně ztraceny vlivem příliš rychlé komunikace, která způsobí přetečení vstupního bufferu. Získávání údajů ze zaslané zprávy je založeno na jejich přesné pozici, proto je při ručním komunikaci nutné tuto skutečnost dodržet.

Příkaz	**Data** **celkem 64 bajtů**
DISCOVER	:Datová část některých příkazů může být libovolná.\r\n\0
+OK-----	:In-Circuit ICSP PIC Programmer Rev:0.5 READY!\r\n\0
SELECTIC	:PIC16F628A\r\n\0
-ERR----	:Sorry, device not supported yet.\r\n\0
VOLTAGES	:Voltage set: Vcc=3 Vpp=0 \r\n\0
+OK-----	:Voltage status: Vusb=4754mV Vcc=3320mV Vpp=17 mV\r\n\0

Tabulka 4.3: Příklad komunikace mezi PC a programátorem

4.4 Online aktualizace knihovny součástek

Jako možné rozšíření aplikace byla zvažována možnost online aktualizace knihovny podporovaných součástek. Jednalo by se o zpřístupnění dokumentu XML nebo CSV s popisem programovacích vlastností na veřejně dostupném (S)FTP serveru. V současné podobě je však tento krok obtížně proveditelný. Je to způsobeno hlavně velkou rozmanitostí programovacích specifikací ICSP rozhraní a s tím spojená nutnost upravit nejen obslužnou aplikaci, ale i firmware řídícího mikrokontroléru. Pokud by bylo dosaženo možnosti aktualizovat firmware přes USB rozhraní „za běhu“, tedy pomocí instrukcí `TABLE READ` a `TABLE WRITE` podobně jako je tomu u programovacího procesu řady PIC18F, je podpora online aktualizace pouze otázkou vyhrazení části paměti mikrokontroléru pro parametry a časové prodlevy protokolu součástek.

4.5 Shrnutí programovacího procesu

Přenesení zdrojového kódu do cílové součástky není triviální proces. V případě mikrokontrolérů Microchip PIC je zkompilovaný kód uložen ve formátu Intel HEX. Kód je do programátoru přenášen řádek po řádku v textové podobě pomocí emulace sériové linky. Řídící mikrokontrolér z přijatých dat nejprve určí délku a typ načteného záznamu (datový záznam, segmentový záznam ...) a provede příslušné operace. V případě datového záznamu je to zejména ne zcela triviální překódování přijatých 16 bitových záznamů do formátu pro cílovou součástku.

Obsahem zdrojového souboru jsou nejen data pro paměť programu, ale i konfigurační registry, uživatelská ID a periferie jako například paměť EEPROM (pokud ji cílová součástka obsahuje). Aby bylo možné naprogramovat tyto specifické vlastnosti a zároveň dodržet standardní formát Intel HEX, jsou tyto speciální vlastnosti namapovány do adresového prostoru, který není za běhu programu přístupný.

Přejde-li součástka do programovacího režimu, převezme její řízení vestavěný konečný automat a šířka adresy vzroste o jeden bit. Tím dojde k nárůstu adresovatelného prostoru na dvojnásobek a zpřístupnění konfiguračních registrů a periferií.

Dle rodiny mikrokontroléru se liší i délka a sada instrukcí pro řízení konečného automatu součástky. Uvedením do programovacího režimu dojde k nastavení programového čítače (PC) na hodnotu nula a součástka očekává naplnění bufferu daty. Před zahájením zápisu dat je zpravidla ještě přečtena hodnota *Device ID* a porovnána s hodnotou uvedenou v kódu. To zabraňuje nechtěnému zápisu firmwaru do jiného typu součástky.

Množství dat, jejich délka a případné obalení nulami závisí na typu součástky, stejně jako nutnost iniciovat zápis nebo ručně inkrementovat programový čítač. Jakmile je buffer naplněn, může být zavolána instrukce pro zahájení zápisu bufferu do paměti součástky. Po té následuje obvykle časová prodleva, během které vestavěný automat zapisuje data na příslušnou adresu. Tento proces je opakován, dokud nejsou zapsána všechna požadovaná data.

Po zapsání všech dat následuje opětovné čtení a porovnání na shodu – verifikace. Pokud nejsou načtená data v rozporu se zdrojovým kódem, programovací proces končí a součástka je resetována pomocí uvedení pinu $\overline{\text{MCLR}}$ co logické nuly.

Kapitola 5

Závěr

5.1 Dosažené výsledky

Cílem práce bylo vytvořit programátor pro mikrokontroléry Microchip PIC 16F a 18F, což se podařilo splnit. Téma jsem zvolil z důvodu, že se mi zdálo blízké. S postupným rozšiřováním projektu o nové periferie jsem však zjistil, že stojím před problémem, o kterém skoro nic nevím. Nový řídící mikrokontrolér z programátoru udělal bez velké nadsázky kapesní vývojový kit s periferiemi nejen pro programování mikrokontrolérů.

Volba zcela nového mikrokontroléru řady PIC18F**J** se ukázala jako šťastná i nešťastná zároveň. Je zcela správné, že je programátor osazen jedním z nejmodernějších mikrokontrolérů této řady, z mé strany to však znamenalo vložit do vývoje mnohem více úsilí. Místy jsem si připadal jako bych začínal úplně znovu a od nuly, což se projevilo na časové tísni, do které jsem se dostal v posledních čtyřech týdnech psaní této práce. Navzdory faktu, že jsem podklady začal shromažďovat již před Vánoci a s prací začal v průběhu ledna.

Je zřejmé, že bez dostupnosti příkladů, hlavně pro komunikaci s USB modulem, by se práci nepodařilo dokončit včas. Jeho obsluha je včetně řízení sběrnice a psaní ovladačů pro PC plně v rukou uživatele. Bez využití demonstračního příkladu zaměřeného na komunikaci po virtuální sériové lince, ke kterému Microchip příslušné ovladače dodává, by bylo zvládnutí komunikace s programátorem velmi obtížné.

Práce mi přinesla velmi mnoho nových zkušeností. Vlastně jsem byl donucen pochopit a naučit se programovat novou řadu mikrokontrolérů. Nově použitý PIC18F67J50 má s řadou PIC16F společného méně než bych si býval přál. Koneckonců i překladače jazyka C pro osmibitové mikrokontroléry Microchip jsou rozděleny na dvě skupiny. První zahrnuje řady 10F, 12F a 16F, druhá pouze řadu 18F. Poprvé jsem se na konkrétním problému setkal s použitím sběrnic SPI a I^2C, konstrukcí převodníků napěťových úrovní a v neposlední řadě k hlubšímu pochopení fungování sběrnice USB.

5.2 Přínos a možnosti dalšího vývoje

Největší plus celé práce vidím v její rozšiřitelnosti. Z hardwarového hlediska je programátor až na dotykové ovládání hotový a jeho výbava je dostatečná pro programování nejen mikrokontrolérů Microchip, ale i součástek většiny ostatních výrobců. Výkon mikrokontroléru (48 MHz) dává dostatečný prostor k implementaci celé řady netriviálních programovacích a testovacích protokolů, jako je například JTAG.

Pokud bych se do podobné práce pouštěl podruhé, zaměřil bych se více na softwarovou

stránku problematiky, kde vidím v současném stavu velké rezervy. Nad rámec zadání práce je programátor vybaven čtečkou SD karet, podpora formátu FAT32 nebo Ext2 ale chybí. O něco lepší je situace u LCD, ovšem ani tu nelze označit za vyhovující. Změny by se samozřejmě nevyhnuly ani hardwarové části, ale byly by spíše kosmetického charakteru. Jednalo by se pouze o přepracování komunikačního rozhraní kvůli podpoře 1,8 V a 2,5 V programování a vestavbu baterie, která by byla užitečná pro práci v terénu.

Literatura

[1] Hello World Program in C. http://www.pic18f.com/18f4550-c-tutorial/2009/11/16/tutorial-4-hello-world-program-in-c/, 2009-11-16 – [cit. 2010-03-27].

[2] Atmel: QSlide, 16-key QMatrix Sensor IC AT42QT2160. http://www.atmel.com/dyn/resources/prod_documents/AT42QT2160_DS.pdf, 2008-07 Rev. A [cit. 2010-03-15].

[3] ChaN: How to Use MMC/SDC. http://elm-chan.org/docs/mmc/mmc_e.html, 2008-06-11 – [cit. 2010-02-22].

[4] FTDI: FT2232H Dual High Speed USB to Multipurpose UART/FIFO IC. http://www.ftdichip.com/Documents/DataSheets/DS_FT2232H.pdf, 2010 [cit. 2010-04-28].

[5] Hamerling, R.: JAL V2 libraries: PICs sharing the same programming specifications. http://code.google.com/p/jallib/source/browse/wiki/PicPgmGroups.wiki, r1783 – [cit. 2010-04-21].

[6] Intel: Hexadecimal Object File Format Specification. http://www.microsym.com/content/index.php?pid=4&id=25, 1988-01-06 Rev. A [cit. 2010-05-05].

[7] Michal Policht, B. F.: QextSerialPort – a cross-platform serial port class. http://sourceforge.net/projects/qextserialport/, 2007-03-27 [cit. 2010-07-05].

[8] Microchip: Katalogový list rodiny mikrokontroléru PIC16F877. http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf, 2001-30-01 [cit. 2009-12-27].

[9] Microchip: Katalogový list rodiny mikrokontroléru PIC16F877A. http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf, 2003-28-07 [cit. 2010-04-28].

[10] Microchip: Katalogový list rodiny mikrokontroléru PIC18F67J50. http://ww1.microchip.com/downloads/en/DeviceDoc/39775c.pdf, 2009-26-09 [cit. 2010-02-10].

[11] Microchip: Flash Memory Programming Specifications PIC16F87XA. http://ww1.microchip.com/downloads/en/DeviceDoc/39589C.pdf, 2010-19-01 [cit. 2010-05-10].

[12] Nokia: Qt – Cross-platform application and UI framework. http://qt.nokia.com, [cit. 2010-11-05].

[13] Olimex: Development boards and Tools. http://www.olimex.com/dev/index.html, [cit. 2010-02-12].

[14] Olimex: PIC-LCD3310: Development board with PIC18F67J50 3-axis accelerometer and Nokia3310 LCD display. http://www.olimex.com/dev/pic-lcd3310.html, [cit. 2010-02-12].

[15] OnSemiconductor: 1.5 A, Step-Up/Down/Inverting Switching Regulator. http://www.onsemi.com/pub/Collateral/MC34063A-D.PDF, 2009-12 [cit. 2010-28-04].

[16] Philips: OM6206 65 × 102 pixels matrix LCD driver. http://pdf1.alldatasheet.com/datasheet-pdf/view/85628/PHILIPS/OM6206.html, 2001-14-11 [cit. 2010-03-16].

[17] Rojvanit, R.: Migrating Applications to USB from RS-232 UART with Minimal Impact on PC Software. http://ww1.microchip.com/downloads/en/AppNotes/00956b.pdf, 2004-03-12 [cit. 2010-03-17].

[18] Rádio, A.: Zajímavé obvody. ročník 8, č. 4, Duben 2003: s. 14–15, iSSN 1211-3557.

[19] Somerville, E.: PICmicro®Device Programming: What You Always Wanted to Know (But Didn't Know Who to Ask). http://ww1.microchip.com/downloads/en/AppNotes/00910a.pdf, 2005-23-11 [cit. 2010-03-15].

[20] STMicroelectronics: Quad 2-Input OR Gate. http://www.st.com/stonline/products/literature/ds/5743/74vhc32.pdf, 2004-03-12 [cit. 2010-03-17].

[21] STMicroelectronics: Octal Bus Transciever (3-State). http://www.st.com/stonline/products/literature/ds/5742/74vhc245.pdf, 2004 [cit. 2010-04-28].

[22] STMicroelectronics: Very low drop voltage regulators with inhibit. http://www.st.com/stonline/products/literature/ds/2573/le33c.pdf, 2008 [cit. 2010-03-17].

[23] Szramowski, P.: MC34063: 1.5 A, Step-Up/Down/Inverting Switching Regulator. ftp://ftp.cadsoft.de/eagle/userfiles/libraries/mc34063.lbr, [cit. 2010-05-11].

[24] USB Implementers Forum, I.: Approved Class Specification Documents. http://www.usb.org/developers/devclass_docs, [cit. 2010-05-05].

[25] Wikipedia: Secure Digital. http://es.wikipedia.org/wiki/Secure_Digital, [cit. 2010-03-29].

Dodatek A

Obsah CD

A.1 Seznam adresářů

Stromové zobrazení adresářové struktury na přiloženém CD.

- **Aplikace**
  - **Bin** – spustitelná verze aplikace pro Linux a Windows
  - **Source** – zdrojové kódy aplikace v C++ (Qt 4.6.1)
- **Firmware**
  - **Hex** – zkompilovaný firmware pro řídící mikrokontrolér
  - **Source** – zdrojové kódy firmwaru v MCC18 (MPLAB 8.4.3)
- **Literatura**
  - **Dokumentace** – použité katalogové listy
- **Schémata**
  - **MCUboard** – schéma a PCB řídící desky programátoru
  - **LCDboard** – schéma a PCB desky s displayem
  - **Knihovny** – upravené knihovny programu Eagle 5.7.0
- **Text**
  - **PDF** – technická zpráva ve formátu PDF
  - **Latex**– zdrojové kódy technické zprávy pro LaTeX

Dodatek B

Tabulky

B.1 Zapojení řídícího mikrokontroléru

Následující tabulka přiřazuje funkce jednotlivým pinům řídícího mikrokontroléru. Protože má většina pinů více možných režimů činnosti, je v seznamu uvedeno vždy číslo pinu a zvolená funkce.

PIN	Název	Typ	Popis
1	RE1	I/O	Náhradní pin
2	RE0	–	–
3	ECCP3	O	Řízení podsvícení LCD
4	RG1	O	Uvedení pinu $\overline{\text{MCLR}}$ do stavu vysoké impedance
5	RG2	O	Indikační LED RX
6	RG3	O	Indikační LED TX
7	$\overline{\text{MCLR}}$	I	Detekce programovacího napětí a RESET
8	RG4	O	Spínání pro piezoreproduktor
9	VSS	Pwr	Zem
10	VCAP	Pwr	Napájecí napětí jádra MCU, připojen kondenzátor
11	RF6	O	$\overline{\text{Chip} - \text{Select}}$ signál pro LCD displej
12	RF7	O	Přepínač Data/$\overline{\text{Command}}$ pro LCD displej
13	AN10	A/D	Měření napájecího napětí Vusb
14	USB D+	I/O	Rozhraní USB
15	USB D-	I/O	Rozhraní USB
16	AN7	–	–
17	VUSB	I	Detekce napětí na USB portu, připojeno trvale k VDD
18	ENVREG	Pwr	Napájení 2,5 V stabilizátoru pro jádro MCU
19	AVDD	Pwr	Napájecí napětí pro analogovou část, +3,3 V, filtrováno
20	AVSS	Pwr	Zem pro analogovou část
21	VREF+	I	Vnější napěťová reference – nepřipojeno
22	VREF-	I	Vnější napěťová reference – nepřipojeno
23	AN1	A/D	Měření napětí komunikačního rozhraní Vcc
24	AN0	A/D	Měření programovací napětí Vpp
25	VSS	Pwr	Zem
26	VDD	Pwr	Napájecí napětí +3,3 V
27	RA5	I	Detekce vložení paměťové karty
28	RA4	I	Detekce ochrany proti zápisu na paměťovou kartu

PIN	Název	Typ	Popis
29	RC1	O	Přepínač napěťové hladiny Vpp 12 V / 13 V
30	RC0	O	Spínač programovacího napětí Vpp
31	RC6	O	Spínač 3,3 V logiky komunikačního rozhraní
32	RC7	O	Volba směru toku dat pro obousměrné piny rozhraní
33	ECCP1	O	ICSP Clock, hodinový signál komunikančího rozhraní
34	SCK1	O	SPI – hodinový signál, I^2C hodinový signál
35	SDI1	I	SPI – MISO, Sběrnice I^2C Data
36	SDO1	O	SPI – MOSI, volný digitální pin
37	RB7	I/O	Obousměrný pin komunikačního rozhraní
38	VDD	Pwr	Napájecí napětí +3,3 V
39	OSC1	I	Vstup oscilátoru – hodinový signál pro MCU
40	OSC2	O	Buzení oscilátoru
41	VSS	Pwr	Zem
42	RB6	I/O	Obousměrný pin komunikačního rozhraní
43	RB5	I/O	Obousměrný pin komunikačního rozhraní
44	RB4	I/O	Obousměrný pin komunikačního rozhraní
45	RB3	I/O	Obousměrný pin komunikačního rozhraní
46	RB2	I/O	Obousměrný pin komunikačního rozhraní
47	RB1	I/O	Obousměrný pin komunikačního rozhraní
48	RB0	I/O	Obousměrný pin komunikačního rozhraní
49	RD7	O	Spínač 5 V logiky komunikačního rozhraní
50	SCK2	O	SPI – Clock, pro LCD a paměťovou kartu
51	SDI2	I	SPI – MISO, pro LCD a paměťovou kartu
52	SDO2	O	SPI – MOSI, pro LCD a paměťovou kartu
53	RD3	I	Vstupní pin komunikačního rozhraní
54	RD2	I	Vstupní pin komunikačního rozhraní
55	RD1	I	Vstupní pin komunikačního rozhraní
56	VSS	Pwr	Zem
57	VDD	Pwr	Napájecí napětí +3,3 V
58	RD0	0	$\overline{\text{Chip} - \text{Select}}$ signál pro paměťovou kartu
59	RE7	–	–
60	RE6	–	–
61	RE5	–	–
62	RE4	–	–
63	RE3	–	–
64	RE2	I/O	Náhradní pin

Tabulka B.1: Popis pinů řídícího mikrokontroléru

B.2 Skupiny mikrokontrolérů se shodným protokolem ICSP

Specifikace ICSP protokolu se liší nejen v rámci jednotlivých rodin mikrokontrolérů, ale i v rámci mikrokontrolérů samotných. Následující tabulka je součástí projektu JAL V2 libraries [5] a rozděluje osmibitové mikrokontroléry Microchip do skupin podle programovacích specifikací.

Kat. list	Mikrokontroléry se shodně navrženým protokolem ICSP
41228E	10F200 10F202 10F204 10F206
41266C	10F220 10F222
41227E	12F508 12F509
41257B	12F510 16F506
41316B	12F519
41396A	12F609 12F615 12F617 12HV609 12HV615 16F610 16F616 16HV610 16HV616
41191D	12F629 12F675 16F630 16F676
41204G	12F635 12F683 16F631 16F636 16F639 16F677 16F684 16F685 16F687 16F688 16F689 16F690
41226F	16F505
41317B	16F526
41207D	16F54
41208C	16F57
41243B	16F59
30034D	16F627 16F628
41196G	16F627A 16F628A 16F648A
40245B	16F716
39588A	16F72
41332B	16F722 16F723 16F724 16F726 16F727 16LF722 16LF723 16LF724 16LF726 16LF727
30324B	16F73 16F74 16F76 16F77
30492B	16F737 16F747 16F767 16F777
41237C	16F785 16HV785
39603C	16F818 16F819
30262E	16F83 16F84 16F84A
39607B	16F87 16F88
39025F	16F870 16F871 16F872 16F873 16F874 16F876 16F877
39589B	16F873A 16F874A 16F876A 16F877A
41287C	16F882 16F883 16F884 16F886 16F887
41244E	16F913 16F914 16F916 16F917 16F946
30467A	16HV540
39583B	18F1220 18F1320 18F2220 18F2320 18F4220 18F4320 18F6520 18F6620 18F6720 18F8520 18F8620 18F8720
39752B	18F1230 18F1330
41357B	18F13K22 18F14K22 18LF13K22 18LF14K22
41342D	18F13K50 18F14K50 18LF13K50 18LF14K50

Kat. list	Mikrokontroléry se shodně navrženým protokolem ICSP
39622K	18F2221 18F2321 18F2410 18F2420 18F2450 18F2455 18F2458 18F2480 18F2510 18F2515 18F2520 18F2525 18F2550 18F2553 18F2580 18F2585 18F2610 18F2620 18F2680 18F2682 18F2685 18F4221 18F4321 18F4410 18F4420 18F4455 18F4458 18F4480 18F4510 18F4515 18F4520 18F4525 18F4550 18F4553 18F4580 18F4585 18F4610 18F4620 18F4680 18F4682 18F4685
30500A	18F2331 18F2431 18F4331 18F4431
41297F	18F23K20 18F24K20 18F25K20 18F26K20 18F43K20 18F4450 18F44K20 18F45K20 18F46K20
39576B	18F242 18F248 18F252 18F258 18F442 18F448 18F452 18F458
39759A	18F2423 18F2523 18F4423 18F4523
30480B	18F2439 18F2539 18F4439 18F4539
39687E	18F24J10 18F24J11 18F24J50 18F25J10 18F25J11 18F25J50 18F26J11 18F26J50 18F44J10 18F44J11 18F44J50 18F45J10 18F45J11 18F45J50 18F46J11 18F46J50 18LF24J10 18LF24J11 18LF24J50 18LF25J10 18LF25J11 18LF25J50 18LF26J11 18LF26J50 18LF44J10 18LF44J11 18LF44J50 18LF45J10 18LF45J11 18LF45J50 18LF46J11 18LF46J50
39624C	18F6310 18F6390 18F6393 18F6410 18F6490 18F6493 18F8310 18F8390 18F8393 18F8410 18F8490 18F8493
39644K	18F63J11 18F63J90 18F64J11 18F64J90 18F65J10 18F65J11 18F65J15 18F65J50 18F65J90 18F66J10 18F66J11 18F66J15 18F66J16 18F66J50 18F66J55 18F66J90 18F66J93 18F67J10 18F67J11 18F67J50 18F67J90 18F67J93 18F83J11 18F83J90 18F84J11 18F84J90 18F85J10 18F85J11 18F85J15 18F85J50 18F85J90 18F86J10 18F86J11 18F86J15 18F86J16 18F86J50 18F86J55 18F86J90 18F86J93 18F87J10 18F87J11 18F87J50 18F87J90 18F87J93
30499B	18F6525 18F6621 18F8525 18F8621
39643B	18F6527 18F6622 18F6627 18F6628 18F6722 18F6723 18F8527 18F8622 18F8627 18F8628 18F8722 18F8723
39606D	18F6585 18F6680 18F8585 18F8680
39688D	18F66J60 18F66J65 18F67J60 18F86J60 18F86J65 18F87J60 18F96J60 18F96J65 18F97J60
Nezařazeno	12F520 16F707 16LF707 18F13K20 18F14K20 18F25K22 18F26J13 18F26J53 18F27J13 18F27J53 18F45K22 18F46J13 18F46J53 18F47J13 18F47J53 18F65K22 18F65K90 18F66K22 18F66K27 18F66K90 18F66K95 18F67K22 18F67K90 18F85K22 18F85K90 18F86J72 18F86K22 18F86K27 18F86K90 18F86K95 18F87J72 18F87K22 18F87K90 18F96J72 18F97J72 18LF25K22 18LF26J13 18LF26J53 18LF27J13 18LF27J53 18LF45K22 18LF46J13 18LF46J53 18LF47J13 18LF47J53

Tabulka B.2: Skupiny mikrokontrolérů se shodně impelentovaným ICSP protokolem [5]

B.3 Identifikace v rámci HAL

Následující tabulka obsahuje informace, které jsou dostupné operačnímu systému, když je programátor připojen k počítači. Kritická jsou zejména nastavení Vendor ID a Product ID, která určují, jaké ovladače budou pro zařízení instalovány.

```
usb_device.bus_number = 4  (0x4)  (int)
usb_device.can_wake_up = false  (bool)
usb_device.device_class = 2  (0x2)  (int)
usb_device.device_protocol = 0  (0x0)  (int)
usb_device.device_revision_bcd = 1  (0x1)  (int)
usb_device.device_subclass = 0  (0x0)  (int)
usb_device.is_self_powered = true  (bool)
usb_device.linux.device_number = 4  (0x4)  (int)
usb_device.linux.sysfs_path = '/sys/devices/.../usb4/4-2'  (string)
usb_device.max_power = 500  (0x1f4)  (int)
usb_device.num_configurations = 1  (0x1)  (int)
usb_device.num_ports = 0  (0x0)  (int)
usb_device.product = 'In-System ICSP Programmer'  (string)
usb_device.product_id = 10  (0xa)  (int)
usb_device.speed = 12.0 (12) (double)
usb_device.vendor = 'Microchip Technology, Inc.'  (string)
usb_device.vendor_id = 1240  (0x4d8)  (int)
usb_device.version = 2.0 (2) (double)
```

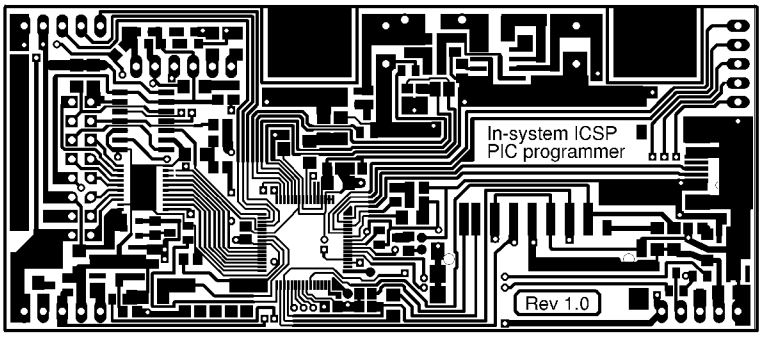
Tabulka B.3: Identifikace programátoru vůči HAL

Dodatek C

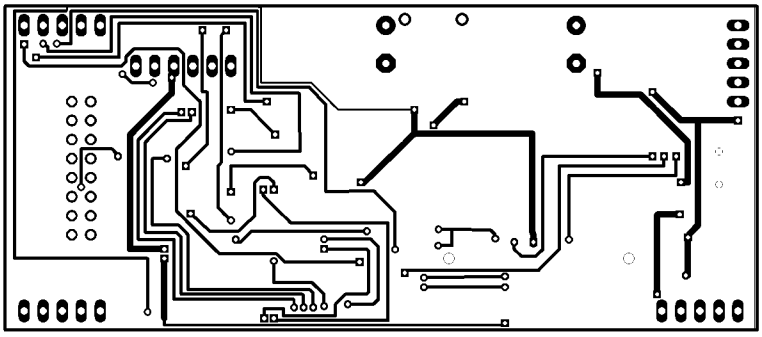
Schémata a PCB

Desky plošných spojů v měřítku 1:1 vhodné jako matrice pro leptání plošného spoje. Zdrojové soubory ve formátu Eagle 5.7.0 včetně upravených knihoven naleznete na přiloženém CD

C.1 Řídící deska

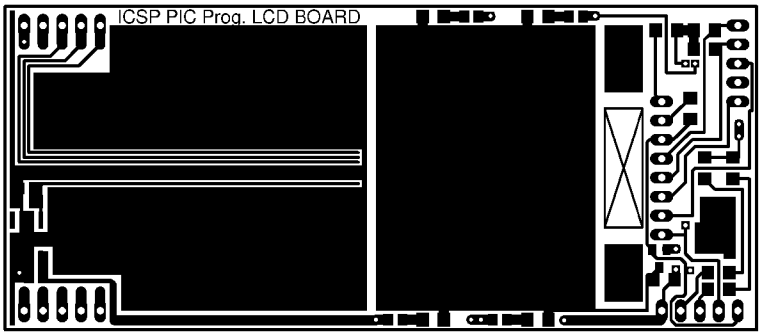


Obrázek C.1: Řídící deska – strana součástek

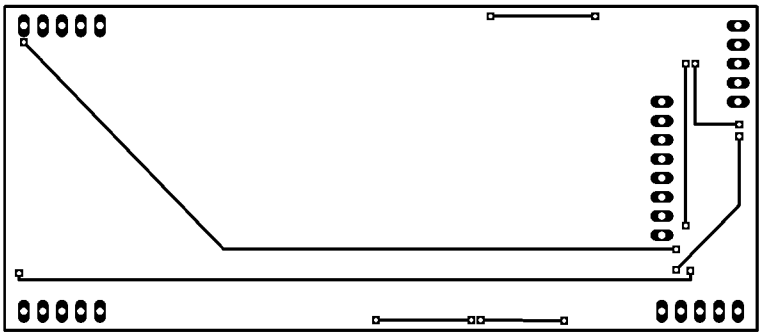


Obrázek C.2: Řídící deska – strana spojů

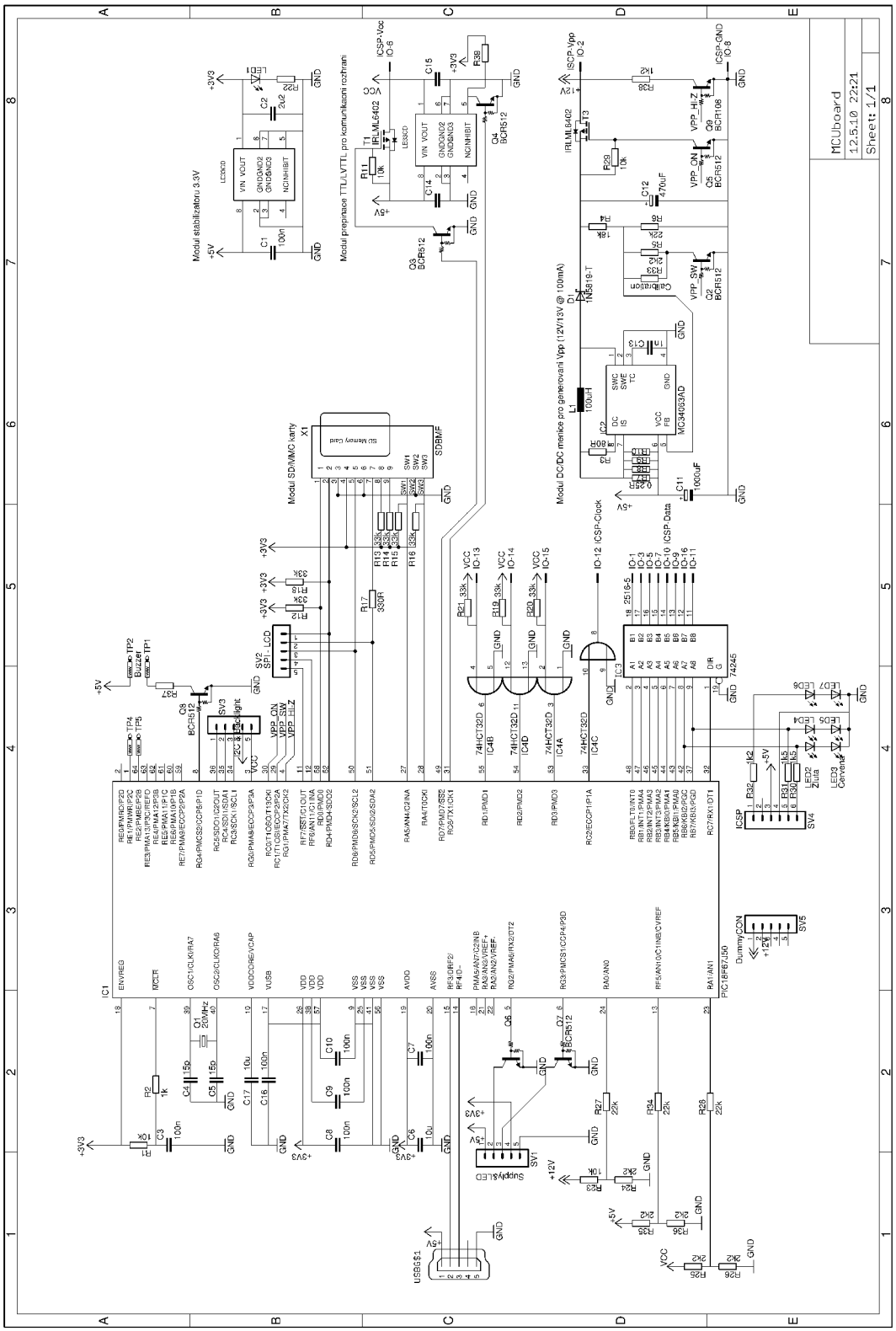
C.2 Deska displeje



Obrázek C.3: Deska displeje – strana součástek



Obrázek C.4: Deska displeje – strana spojů


MCUboard
12.5.10 22:21
Sheet: 1/1
Modul stabilizátoru 3.3V
Modul přepínače TTL/LVTTL pro komunikaci rozhraní
Modul SD/MMC karty
Modul DC/DC menice pro generování Vpp (12V/13V @ 100mA)
ICSP-Vcc
ICSP-Vpp
ICSP-GND
ICSP-Clock
ICSP-Data
VPP_ON
VPP_HI/Z
VPP_SW
Backlight
SupplyLED
LED1 Zluta
LED2 Zluta
LED3 Cervena
USBG$1
ICSP
SV1
SV2 LCD
SV3
SV4
SV5
SDBMF
IC1
PIC18F67J50
74HCT32D
74245
MIC94063AD
IRLML6402
1N5819-T
BCR512
BCH108