VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

NÁVRH A REALIZACE LABORATORNÍHO PRACOVIŠTĚ S MIKROKONTROLÉREM RABBIT

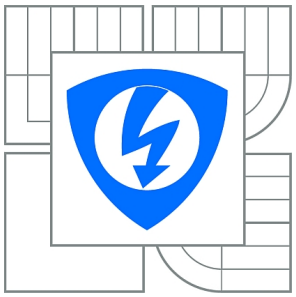
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. TOMÁŠ VÁŠA
AUTHOR

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

NÁVRH A REALIZACE LABORATORNÍHO PRACOVIŠTĚ S MIKROKONTROLÉREM RABBIT

LABORATORY STAND WITH RABBIT CPU

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

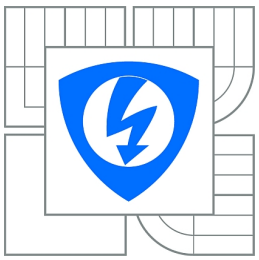
AUTOR PRÁCE — Bc. TOMÁŠ VÁŠA

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. ZDENĚK BRADÁČ, Ph.D.

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Tomáš Váša	***ID:***	98074
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Návrh a realizace laboratorního pracoviště s mikrokontrolérem RABBIT

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte koncepci laboratorní úlohy pro výuku programování mikrokontroléru RABBIT. Navrhněte koncepci periferních obvodů a komunikačních rozhraní. Vybavte mikrokontrolérem řady RABBIT. Navrhněte elektroniku laboratorní úlohy, navrhněte a realizujte nezbytné DPS, osaďte je a oživte elektroniku. Vybavte programovým vybavením a demonstrujte správnou funkci HW realizace i programového vybavení.

DOPORUČENÁ LITERATURA:

Pavel Herout: Učebnice jazyka C, KOPP, 2004, IV. přepracované vydání, ISBN 80-7232-220-6
Dle pokynů vedoucího práce.

Termín zadání:	7.2.2011	***Termín odevzdání:***	23.5.2011

Vedoucí práce: doc. Ing. Zdeněk Bradáč, Ph.D.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Diplomová práce se zabývá návrhem laboratorního pracoviště pro výuku programování jednoduchých mikrokontrolerů. Pracoviště je osazeno dvěma moduly od firmy Rabbit, které jsou vzájemně propojeny pomocí lokální sítě.
Navržené úlohy se soustředí na práci se základními periferiemi, jako jsou digitální vstupy a výstupy, generátor PWM signálu,hodiny reálného času a sériová komunikace. Dále se úlohy zabývají vyhodnocováním maticové klávesnice a komunikací s alfanumerickým LCD displejem. Díky možnosti jednoduchého přístupu k internetu jsou některé úlohy navrženy pro práci s tímto rozhraním.
Pro navržené úlohy byly také navrženy potřebné periferie. Díky těmto přípravkům je možné ovládání stejnosměrných motorů, krokových motorů, ovládání digitálních výstupů, ovládání LCD displeje.
Každá navržená úloha obsahuje úvod, zadání, návod a vývojový diagram realizované úlohy. Návody jsou sepsány tak, aby obsahovaly důležité kroky ke správnému zvládnutí úlohy. Jednotlivé úlohy jsou koncipovány se vzrůstávající obtížností, tak jak studenti získávají zkušenosti.

KLÍČOVÁ SLOVA

Rabbit, RCM3200, RCM2200, BL2600, OP7200, Ethernet, TCP/IP, webové rozhraní, web server

ABSTRACT

This master's thesis contains design of laboratory stand for learning programming microcontrollers. Stand is equipped with two Rabbit modules, which are connected together over local area network.
Exercises are focused on working with basic peripheries as digital inputs and outputs,PWM modulation, serial communication. Exercises are interested in evaluation of matrix keyboard and communication with alphanumeric display controller. With the ability to easy access the Internet some exercises are designed for work with this interface. To these exercises is made a preparation of external peripherals. Through these peripherals there can be can be controlled DC motors, step motor, binary outputs and inputs.
Each lab exercise contains an introduction, task, instructions and the flowchart of realized exercise. Instructions are written so that it contains important steps to properly manage assignments. Individual exercises are designed to gradually increasing complexity of exercises

KEYWORDS

Rabbit, RCM3200, RCM2200, BL2600, OP7200, Ethernet, TCP/IP, web interface, web server

VÁŠA Tomáš, *NÁVRH A REALIZACE LABORATORNÍHO PRACOVIŠTĚ S MIKROKONTROLÉREM RABBIT*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, ÚSTAV AUTOMATIZACE A MĚŘÍCÍ TECHNIKY, 2011. 95 s. Vedoucím práce byl doc. Ing. Zdeněk Bradáč, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „NÁVRH A REALIZACE LABORATORNÍHO PRACOVIŠTĚ S MIKROKONTROLÉREM RABBIT“ jsem 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 diplomové 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 části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

Poděkování

Chtěl bych poděkovat všem lidem, bez nichž by nemohla tato práce vzniknout, především svému vedoucímu práce Doc. Ing. Zdeňku Bradáčovi, Ph.D., za vedení v průběhu realizace laboratorních úloh a za poznámky k práci. Dále bych chtěl poděkovat rodině, zejména za jejich pomoc a podporu při mém studiu.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 ÚVOD

V dnešní době, kdy jsme obklopěni za všech stran nejrůznější elektronikou, se stále více setkáváme s mikrokontrolery. Proto je třeba studenty elektrotechnických oborů připravit tak, aby byli schopni tato zařízení programovat a nasazovat v praxi. Dříve velice oblíbené 8bitové mikrokontrolery řady x51 v dnešní době již nedostačují svým výkonem a možnostmi konektivity. Cílem této práce je naučit studenty základní programovací návyky zábavnou formou, která studenty zaujme a neodradí.

Zadaná problematika musí být dobře a srozumitelně vysvětlena, což klade vysoké nároky na zkušenosti vyučujícího. Navržené úlohy musí být obtížné tak, aby je bylo možné zvládnout v v hodinách počítačových cvičení.

Začít programovat mikrokontorolery lze nejdříve po pochopení jejich vnitřní struktury. Tato část by měla být vyučována na přednáškách, kde by student získal obecné znalosti o danné části použitého zařízení. Poté v počítačových cvičeních by měl získat konkrétní informace o použitém mikrokontroleru, které by měly být základem pro správné programování.

Na začátku práce jsou popsány základní vlastnoti použitých modulů od firmy Rabbit. Tato část je nezbytná pro získání obecného přehledu o možnostech těchto modulů. Jsou zde také popsány základní vlastnosti vývojového prostředí Dynamic C. Velká pozornost je zde věnována popisu funkcí spojených s možností připojit tyto moduly k ethernetovému rozhraní. V další části práce se popisují navržené periferie a jejich vlastnosti. Jsou zde uvedeny základní informace potřebné k jejich ovládání. Závěr práce se věnuje navrženým úlohám, které využívají integrované systémy umístěné na uvedených modulech a navržené periferie .

2 MODULY

Pro návrh byly použity dva moduly od firmy RABBIT. Jedním je jednodeskový počítač BL2600 a druhý je ovládací panel OP2700. Oba dva moduly jsou ovládány procesorem od stejné firmy. Jejich největší výhodou oproti podobným modulům je možnost přímého připojení k ethernetu. Moduly se programují v aplikaci Rabbit's Dynamic C speciálně určené pro programování výrobků firmy Rabbit. Tato aplikace v sobě zahrnuje editor kódu, kompilátor a debbuger. Jsou s ní dodávány vzorové kódy, ve kterých je ukázáno jak se správně používají periferie modulů. Napájení modulů je řešeno síťovými adaptéry, které dávají výstupní napětí 12 V a proud až 2 A.

2.1 BL2600

BL2600 je vysoce výkonný jednodeskový počítač, který obsahuje digitální a analogové vstupně/výstupní porty a možnost připojení k ethernetu. Je řízen modulem RCM 3200, který je vyráběn stejnou firmou.Tento modul se programuje a debuguje přes sériovou linku. Výstupy a vstupy počítače jsou vyvedeny na dva druhy svorek. Jsou použity standardní konektory se zámkem a průchozí IDC soket, díky kterému je možné připojit zařízení zespoda desky.

Mikroprocesor:	Rabbit 3000 pracující frekvencí 44,2 MHz
Paměť programu:	512 kB
Paměť dat:	256 kB
Flash paměť	512 kB
Připojitelnost	36 digitálních vstupů/výstupů
	4 výkonové výstupy
	osm 11bitových analogových vstupů
	čtyři 12bitové analogové výstupy
	Rabbit net, ethernet, tři sériové porty

2.1.1 Digitální vstupy a výstupy

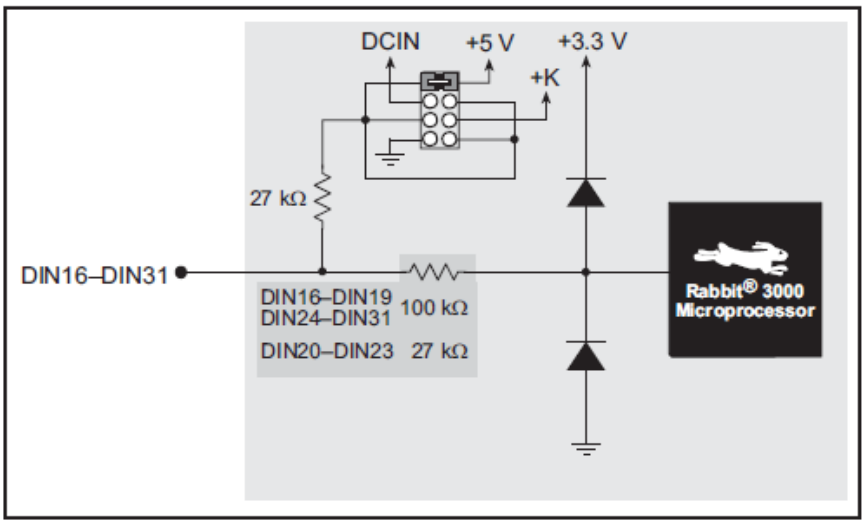
Digitální vstupy

BL2600 má 16 digitálních vstupů, z nichž některé mají speciální funkci. Tyto funkce a využití pinů ukazuje Tab. 2.1.

Pin	Přerušení	Input Capture	Qadraturní dekodér	PWM výstup
DIN16	X			
DIN17	X			
DIN18			X	
DIN19		X	X	
DIN20			X	X
DIN21		X	X	X
DIN22			X	X
DIN23		X	X	X

Tab. 2.1: Speciální funkce digitálních vstupů

Pokud budeme piny používat jako vstupy, mohou být připojeny k napájecímu napětí, k napětí +5 V nebo na zem. Schéma vstupního pinu je na Obr. 2.1. Volba připojení se volí propojkami JP3, JP4 a JP5 umístěnými na desce počítače.



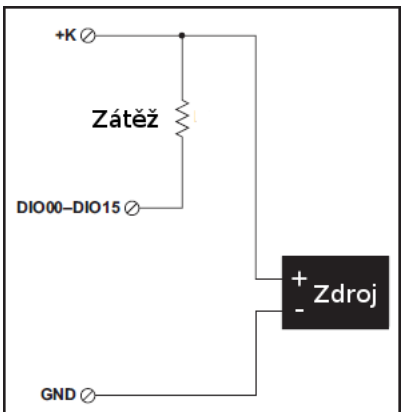
Obr. 2.1: Zapojení vstupních pinů DIN16 až DIN23.

Pokud je vstupní napětí menší než 1,4 V je vyhodnoceno jako log. „0“. Pokud je vyšší je vyhodnoceno jako log. 1. Na vstupy se může připojit napětí o velikosti až -36 V až +36 V.

Vstupně/Výstupní piny

U 16 pinů (DIN00 - DIN15) je možné zvolit jestli jsou vstupní nebo výstupní. Volby se provádí softwarově zavoláním příslušných funkcí. Piny nastavené jako vstup mají

stejné vlastnosti jako vstupy DIN16 - DIN23. Výstupní piny mohou spínat zátěž pouze k zemi. Způsob zapojení ovládané zátěže je na Obr. 2.2.



Obr. 2.2: Zapojení výstupního pinu.

Přes takto zapojenou zátěž může protékat až 200 mA při 40 V aniž by se výstupní pin poškodil.

Výkonové výstupy

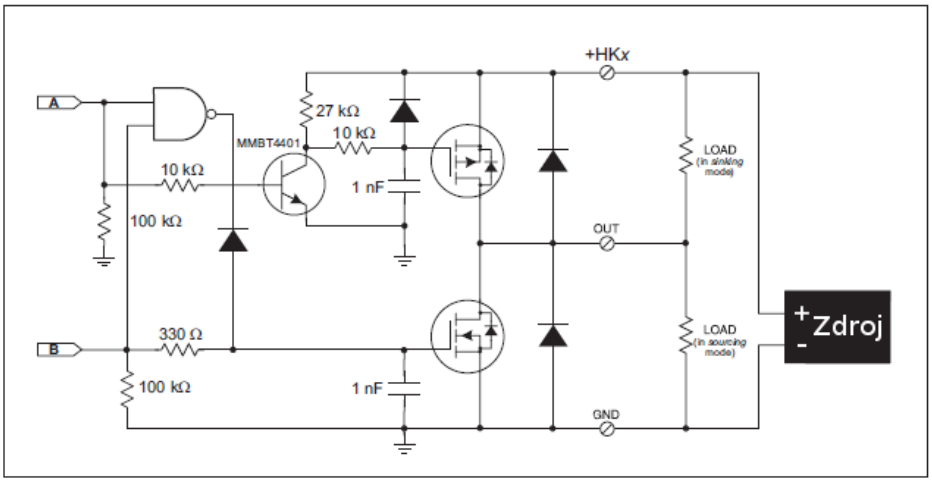
BL2600 disponuje čtyřmi výstupy, které dokáží spínat proud až 2 A. Zátěž mohou připojovat k zemi nebo k napájecímu napětí. Dají se použít jako H- můstek nebo výkonový spínač. Po restartu zařízení se výstupy nacházejí ve stavu vysoké impedance. Jejich funkce se přepíná softwarově pomocí speciálních funkcí. Pokud budeme používat pro tyto výstupy svorkovnici IDC je potřeba použít oba vyvedené konektory, protože jedním může protékat pouze proud 1 A. Schéma výstupu je na Obr. 2.3.

Analogové vstupy

Zařízení disponuje A/D převodníkem s přesností 12 bitů (11 bitů se používá pro hodnotu a jeden bit pro polaritu napětí). Vstup A/D převodníku je multiplexován na 8 vyvedených vstupů. Každému vstupu se dá nastavit vlastní zesílení a filtrace. Vstupy se dají používat jako unipolární a nebo bipolární.

Analogové výstupy

Můžeme používat D/A převodník. Tento převodník může generovat čtyři výstupní napětí v rozsahu 0 až 10 V nebo čtyři proudy 4 - 20 mA. Pokud na výstupu AV0 nastavíme výstupní napětí + 10 V, výstupní proud na výstupu AI0 bude 20 mA. To je způsobeno tím, že pro napěťový i proudový výstup je použit jeden D/A převodník.



Obr. 2.3: Schéma výkonového výstupu.

2.1.2 Sériová komunikace

BL2600 může komunikovat s okolím pomocí několika sériových rozhraní, které se dají nakonfigurovat podle potřeby. Možnosti konfigurace jsou ukázány v Tab. 2.2. Všechny sériové linky komunikují v asynchronním módu. Maximální přenosová rychlost je 5,525 Mbps.

Mód			
	C	D	E
0	RS-232, 3 vodiče	RS-232, 3 vodiče	RS-232, 3 vodiče
1	RS-232, 3 vodiče	RS-485	RS-232, 3 vodiče
2	RS-232, 5 vodičů	RS-232, 3 vodiče	CTS/RTS
3	RS-232, 5 vodičů	RS-485	CTS/RTS

Tab. 2.2: Možnosti sériové komunikace

2.1.3 Připojení k ethernetu

Zařízení se může připojit k ethernetu pomocí standardního síťového konektoru RJ - 45. Jsou podporovány přenosové rychlosti 10 Mb/s a 100 Mb/s. Díky těmto vlastnostem může zařízení lehce komunikovat po standardní počítačové síti.

2.1.4 Programovací konektor

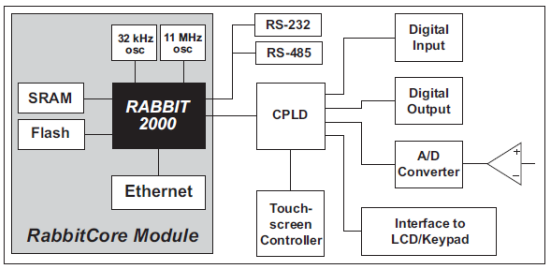
Je to 10 pinový konektor, na který se připojuje kabel připojený do počítače. Umožňuje programování a debugování zařízení. Používá sériovou linku na portu A.

2.2 OP2700

Jedná se o inteligentní operátorský panel vybavený grafickým dotykovým displejem. S okolím může být propojen přes sériové linky RS - 232 nebo RS - 485. Také pomocí ethernetu a Rabbbit netu. Panel obsahuje také 24 digitálních vstupů/výstupů a 9 tlačítek, bateriově zálohované hodiny reálného času.

Mikroprocesor:	Rabbit 2000 pracující frekvencí 22,1 MHz
Paměť flash:	256 kB
Paměť RAM:	128 kB
Displej	VGA 320 x 240 pixelů s podsvícením
Připojitelnost	24 digitálních vstupů/výstupů
	osm 11bitových analogových vstupů
	Rabbit net, ethernet, tři sériové porty

Všechny výstupy a vstupy jsou vyvedeny na svorkovnici, umístěné po stranách hlavní desky. Na Obr. 2.4 je zobrazeno blokové schéma ovládacího panelu OP2700.

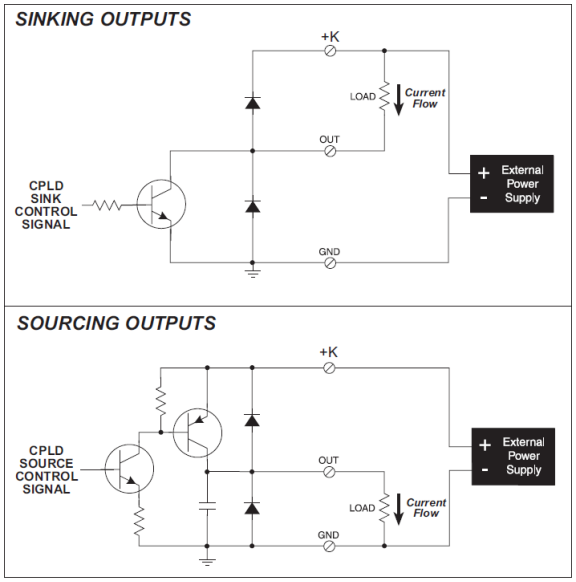


Obr. 2.4: Blokové schéma OP2700.

2.2.1 Digitální vstupy a výstupy

Vstupy a výstupy jsou až na několik vyjímek zapojeny stejně jako konfigurovatelné vstupy/výstupy u modulu BL2600. Vnitřní struktura vstupů je na Obr. 2.1. Výstupní piny na rozdíl od BL2600 mohou fungovat ve zdrojovém režimu, kdy do zátěže dodávají proud. V tomto režimu mohou do zátěže dodat proud až 250 mA.

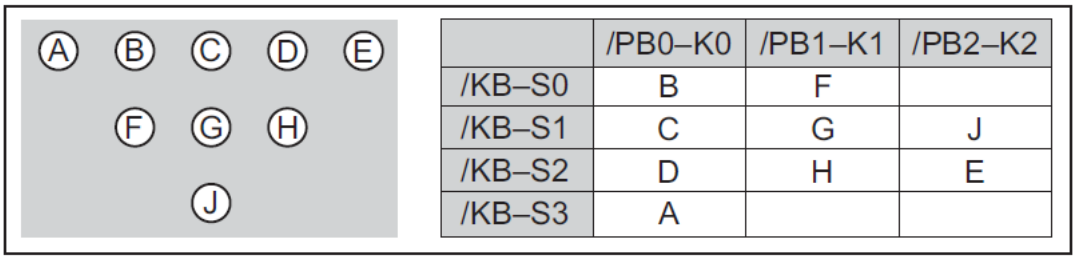
A nebo ve spotřebičovém režimu, kdy spínacím tranzistorem může protéct až 350 mA. Obě dvě zapojení jsou na Obr. 2.5.



Obr. 2.5: Zapojení výstupů OP2700.

2.2.2 Klávesnice

K panelu je připojena maticová klávesnice s devíti tlačítky. Klávesnice je vyvedená na přední panel pod dotykový display. Na Obr. 2.6 je zobrazeno zapojení klávesnice.



	/PB0–K0	/PB1–K1	/PB2–K2
/KB–S0	B	F	
/KB–S1	C	G	J
/KB–S2	D	H	E
/KB–S3	A		

Obr. 2.6: Zapojení maticové klávesnice OP2700.

2.2.3 Dotykový displej

OP2700 je vybaven monochromatickým dotykovým displejem s rozlišením 320 x 270 bodů a bílým podsvětlením. Rozlišení dotykového rozhraní je 4096 x 4096 bodů. Vyhodnocování pozice dotyku je řešeno rezistivní metodou.

3 VÝVOJOVÉ PROSTŘEDÍ DYNAMIC C

Dynamic C je vývojové prostředí dodávané firmou Rabbit Semiconductors. Je navržené specialně pro práci s mikrokontrolery tohoto výrobce. Obsahuje v sobě přehledný editor, kompilátor a debbuger.

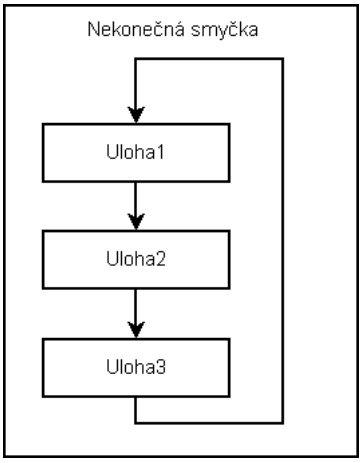
Debugging probíhá prostřednictvím programovacího kabelu připojeného k programovanému zařízení. Kód je možno psát ve standardním jazyku C, v assembleru a nebo jejich kombinací.

Dynamic C má v sobě integrované speciální funkce, které programátorovi usnaďnují práci s mikrokontrolery Rabbit. Programátoři nejvíce ocenní širokou nabídku již předpřipravených knihoven, které umožňují rychlé a jednoduché programování. Knihovny obsahují funkce pro ovládání všech periferií obsažených na desce BL2600 a OP2700.

Další výhodnou vlastností Dynamic C je podpora nepreemtivního multitaskingu. Díky tomu může procesor zpracovávat víc úloh současně.

3.1 Multitasking v Dynamic C

Dynamic C podporuje tři typy multitaskingu. Nejvýkonnější verze multitaskingu, kterou je možné použít s mikrokontrolery Rabbit, se nazývá uC/OS-II. Tato verze je dostupná pouze v premiové verzi prostředí Dynamic C. Další dvě verze multitaskingu jsou dostupné i v normální verzi Dynamic C a zabývají se kooperativním a preemtivním multitaskingem. Na Obr. 3.1 je zobrazen průběh programu při použití multitaskingu.



Obr. 3.1: Průběh programu při použití kooperativniho multitaskingu [13]

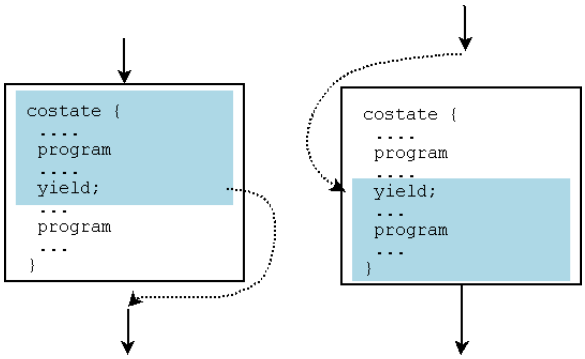
3.1.1 Kooperativní multitasking

V tomto případě každá spuštěná úloha ovládá procesor, dokud se této kontroly sama nevzdá. Ve skutečnosti počítač, který používá tento multitasking, funguje jako stavový automat. V prostředí Dynamic C se jednotlivé stavy uzavírají do bloku uvedeného definicí `costate`. Tyto bloky se dají nořit do sebe a poté jsou postupně zpracovávány.

Pokud se v bloku `costate` narazí na slova `yield,abort` a nebo `waitfor`. Prováděná úloha se ukončí a začne se zpracovávat další.

yield

Po volání tohoto příkazu je okamžitě ukončeno zpracovávání aktivní úlohy a začne se zpracovávat úloha následující. Po dokončení všech ostatních úloh se opět začne zpracovávat úloha, která volala yield, v místě následujícím ihned za příkazem yield. Postup zpracování programu je rozkreslen na Obr. 3.2.



(a) Po prvním průchodu (b) Po druhém průchodu

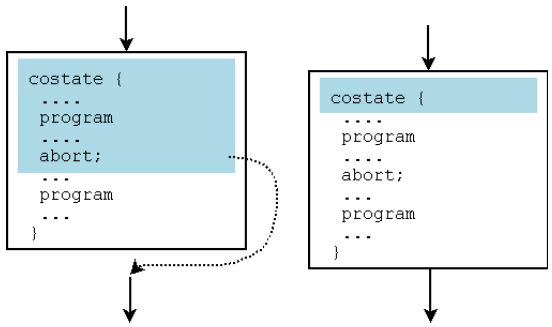
Obr. 3.2: Průběh programu při použití `yield` [13]

abort

Úloha okamžitě odevzdává kontrolu nad CPU a po zpracování všech ostatních úloh, už nepokračuje za tímto voláním, jako v případě yield. Postup zpracování programu je rozkreslen na Obr. 3.3.

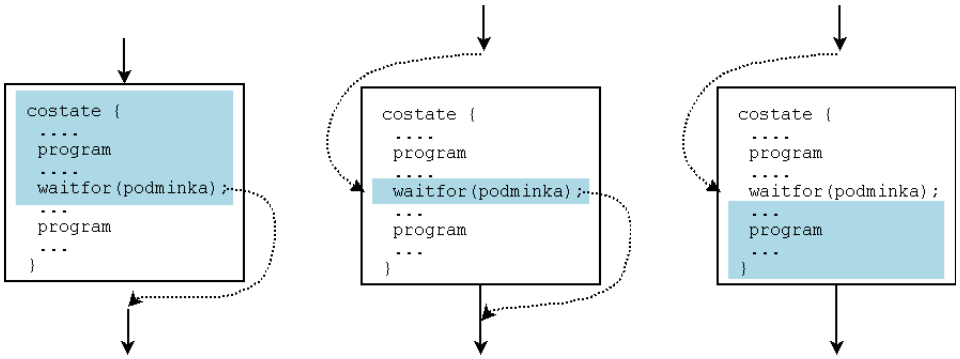
waitfor(podmínka)

Prováděná úloha bude přeskočena do té doby, než bude splněna podmínka uvedená v parametru příkazu `waitfor`. V dalším průchodu programem se opět otestuje zda je splněna podmínka. Pokud ano, je vykonán zbytek instrukcí v úloze. Postup zpracování programu je rozkreslen na Obr. 3.4.



(a) Po prvním průchodu (b) Po druhém průchodu

Obr. 3.3: Průběh programu při použití `abort` [13]



(a) Po prvním průchodu (b) Po druhém průchodu (c) Dokončení události po splnění podmínky. (Po třetím průchodu)

Obr. 3.4: Průběh programu při použití `waitfor` [13]

```
while(1){ // hlavni nekonecna smycka

        costate {
                ... uloha c.1 ... // telo prvni ulohy
                yield;
        }
        costate {
                ... uloha c.2 ... // telo druhe ulohy
                waitfor(DelaySec(1));
        }
}
```

V tomto případě první úloha ukončí svoje provádění vždy, když narazí na `yield` a začne se provádět úloha druhá, která odevzdá kontrolu nad CPU na 1 vteřinu.

3.1.2 Preemtivní multitasking

V tomto případě přiděluje kontrolu nad CPU výhradně operační systém. V prostředí Dynamic C se nastavuje jak dlouho může jedna úloha kontrolovat CPU. Nevýhodou tohoto multitaskingu v Dynamic C je, že způsobuje potíže při komunikaci po sériové lince.

Preemtivní multitasking se zde vytváří voláním funkce `slice`.

slice

Každá úloha vytvořená pomocí `slice` má přidělen svůj prostor pro ukládání proměnných hodnot (tzv. "stack") a dobu po kterou se má daná úloha vykonávat. Tvar zápisu je následující:

```
while(1)
{
slice(100,20) {
        ... uloha c.1 ... // telo prvni ulohy
        }
slice(100,40) {
        ... uloha c.2 ... // telo druhe ulohy
        }
}
```

Tímto zápisem udáváme, že první úloha má k dispozici stack o velikosti 100 bytů a má se protádět 20 ticků (1 tcik je roven 1/2048 sekundy). Druhá úloha má také pridělený stack o velikosti 100 bytů, ale bude se provádět 40 ticků.

Podrobnější popis multitaskingu s názornými příklady je uveden v [12] a v dokumentaci k prostředí Dynamic C [13].

3.2 Adresování vstupů a výstupů

Dynamic C umožňuje číst a nebo nastavovat buďto jednotlivé piny a nebo celé osmice pinů, tak zvané porty. Pro čtení/nastavování jednotlivých pinů se používají funkce `digIn` a nebo `digOut`. Podle toho, zdali chceme číst a nebo zapisovat z/na daný pin. Jako parametr se zadává číslo pinu, se kterým chceme pracovat a hodnota, kterou chceme zapsat. Práce s celými porty se provádí pomocí funkcí `digInBank` a `digOutBank`. Parametrem těchto funkcí je adresa banky a hodnota, kterou na port chceme zapsat.

Prototypy funkcí jsou zapsány níže.

```
void digOut(int channel, int state);
// funkce nastavi na pinu s cislem "channel" hodnotu "state"
void digOutBank(char bank, char data);
// funkce nastavi na piny patricim do portu
// oznacenem hodnotou "bank" hodnoty podle "data"
int digIn(int channel);
// vrati hodnotu prectenou z pinu oznacenem cislem "channel"
char digInBank(int bank);
// vrati cislo odpovidajici hodnotam jednotlivych pinu v portu
// oznacenem hodnotou "bank"
```

V Tab. 3.1 a Tab. 3.2 je rozepsáno, do které banky patří jaký pin.

bank / Pin	D7	D6	D5	D4	D3	D2	D1	D0
0	DIO07	DIO06	DIO05	DIO04	DIO03	DIO02	DIO01	DIO00
1	DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO09	DIO08

Tab. 3.1: Rozdělení výstupních pinů do portů

bank / Pin	D7	D6	D5	D4	D3	D2	D1	D0
0	DIO07	DIO06	DIO05	DIO04	DIO03	DIO02	DIO01	DIO00
1	DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO09	DIO08
2	DIN23	DIN22	DIN21	DIN20	DIN19	DIN18	DIN17	DIN16
3	DIN31	DIN30	DIN29	DIN28	DIN27	DIN26	DIN25	DIN24

Tab. 3.2: Rozdělení vstupních pinů do portů

3.3 Podpora protokolu TCP/IP

Podporu protokolu TCP/IP zajišťuje v Dynamic C několik dodávaných knihoven. Protokol TCP/IP je podporován ve verzi 4. Podrobnější informace jsou uvedeny v [14] a [15]. Programy a nastavení se pro moduly BL2600 a OP7200 neliší.

3.3.1 Podporované rozhraní a jejich inicializace

- Ethernet
- PPP (Point-to-Poit Protocol)over serial link
- PPP over Ethernet (PPPoE)

Výběr fyzického rozhraní, které bude podporované TCP/IP zásobníkem se provádí pomocí následujících maker.

3.3.2 Inicializace TCP/IP zásobníku

Inicializce se provádí v bloku hlavního programu před použitím funkcí, které využívají TCP/IP zásobník. Provádí se voláním funkce `sock_init();`. Příklad použití je uveden níže.

```
void main(){
        sock_init();
        ...
}
```

Zavoláním funkce `sock_init()` provedeme následující operace:

- Inicializace subsystému pro ARP,TCP,UDP a DNS
- Zkontroluje se, zdali nebyla funkce `sock_init()` už zavolána, pokud ne jsou vykonány další kroky.
- Inicializuje se packetový řadič. To znamená, že se resetuje hardwarová část a vymaže se přijímací buffer
- Vymažou se směrovací tabulky
- Rozhraní je inicializováno podle nastavení uvedeného v předdefinovaných makrech.

Pokud všechny kroky proběhnou bez problému, je ethernetové rozhraní připravené k práci.

3.3.3 Konfigurace rozhraní

TCP/IP stack získává nastavení z jednoho z těchto zdrojů:

- předdefinované konfigurace z knihovny `tcp_config.lib`
- definice pomocí maker před derektivou `#use dcrtcp.lib`
- konfigurace pomocí protokolů BOOTHP a DHCP
- voláním funkce `ifconfig()`
- konfigurace přes konzoli `zconsole.lib`

Pro praktické použití je vhodnější použít dynamické nastavení rozhraní místo statického. Statické nastavení se nejvíce používá při testování zařízení.

Předdefinovaná konfigurace

Pro rychlé a jednoduché nastavení rozhraní se může použít již předdefinované makro TCPCONFIG. Použití je velice jednoduché jeho příklad je ukázán níže.

```
#define TCPCONFIG cislo_nastaveni
#use "dcrtcp.lib"
```

`cislo_nastaveni` udává jaké nastavení bude použito. Nejčastěji používané možnosti jsou uvedeny v Tab. 3.3.

Číslo_nastaveni	Ethernet	PPP	DHCP	RUNTIME
1	Ano	Ne	Ne	Ne
2	Ne	Ano	Ne	Ne
3	Ano	Ne	Ano	Ne
4	Ano	Ano	Ne	Ne
5	Ano	Ne	Ano	Ne
6	Ano	Ne	Ne	Ano

Tab. 3.3: Nastavení rozhraní pro jednotlivé hodnoty makra TCPCONFIG

Pokud zvolíme `číslo_nastaveni 3`, makro `TCPCONFIG` se převede na následující kód.

```
#if TCPCONFIG == 3
 #define USE_ETHERNET 1
 #define USE_DHCP
 #define DHCP_NUM_SMTP 1
 #define DHCP_CLASS_ID "Rabbit-TCPIP:Z-World:DHCP-Test:1.0.0"
 #define DHCP_CLIENT_ID_MAC
 #define IFCONFIG_ETH0 \
 IFS_DHCP, 1, \
 IFS_UP
#endif
```

Statická konfigurace

Jedná se o nejjednodušší možnost nastavení, bohužel je většinou vhodné jen pro testovací účely. Nastavení se provádí použitím maker uvedených v příkladu níže.

```
#define MY_IP_ADDRESS '10.10.6.100' // definice IP adresy zarizeni
#define MY_NETMASK '255.255.255.0'  // maska podsite
#define MY_GATEWAY '10.10.6.1'      // IP adresy vychozi brany
```

Tento způsob zápisu lze stále používat, ale je zachován pouze z důvodu zpětné kompability. Doporučuje se používat způsob zápisu pomocí makra IFCONFIG_*, kde * udává jméno rozhraní, které má být použito. Příklad zápisu je uveden níže.

```
#define IFCONFIG_ETH0 \
IFS_IPADDR, aton('IP adresa zarizeni'),\
IFS_NETMASK, aton('maska podsite'),\
IFS_ROUTER_SET, aton('IP adresa vychozi brany'), \
IFS_UP
```

Výhodou tohoto zápisu je, že rozhraní lze v průběhu programu zapínat a nebo vypínat podle potřeby. Standardně je rozhraní vypnuto. Toto přepínání je možné díky parametru `IFS_UP`
Ostatní možnosti nastavení konfigurace jsou popsány v [14] a [15].

3.4 HTTP server

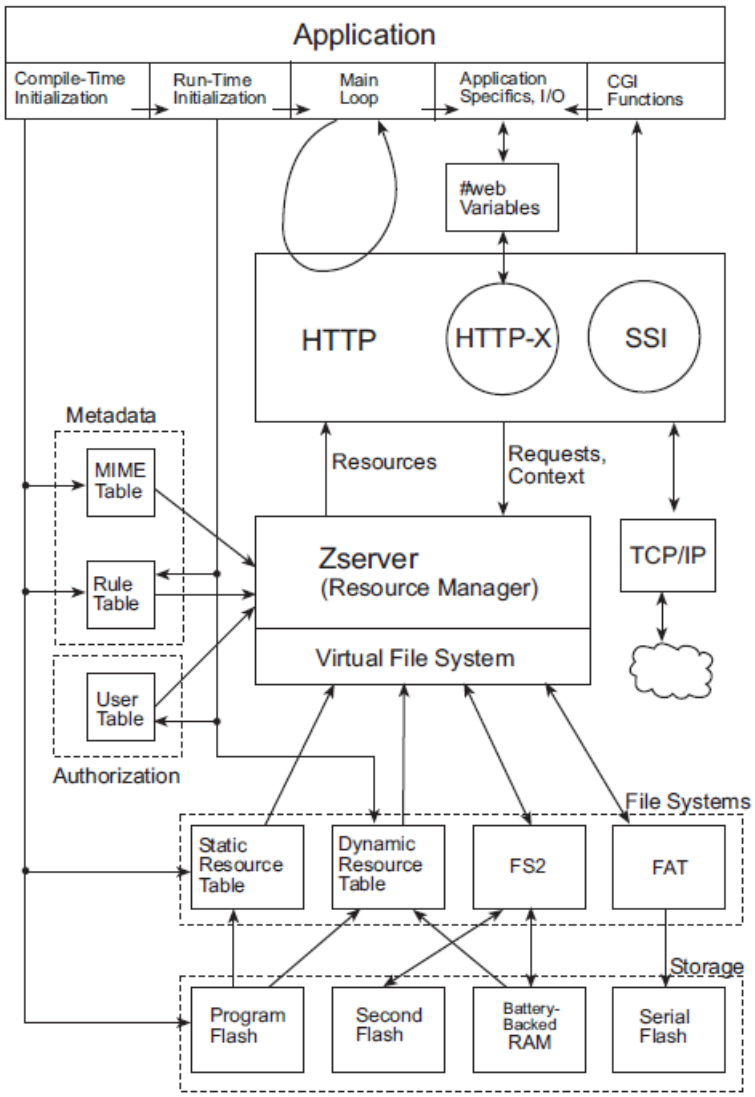
Http server umožňuje ovládání modulu přes HTTP rozhraní a přehledné grafické zobrazení stavu zařízení v internetovém prohlížeči. Prostředí Dynamic C je vybaveno velkým množstvým knihoven potřebných pro jeho zprovoznění. Samotný HTTP server se skládá ze dvou částí: samotný zpracovávaný kód, který obsluhuje připojené periferie k modulu a kód v jazyce HTML nebo ZHTML, tvořící uživatelské rozhraní zobrazované v internetovém prohlížeči.

3.4.1 Architektura HTTP serveru

Na Obr. 3.5 je struktura HTTP serveru. Obsahuje všechny důležité části pro obsluhu webové aplikace. Na Obr. 3.5 je zobrazena maximální možná konfiurace. Pro většinu aplikací není třeba využívat všechny prvky. Podrobný popis všech vlastností je popsán v [15]

Blok Aplication

Tento blok obsahuje dalších pět menších bločků, které reprezentují vytvořený kód. Všechny ostatní funkce pod tímto blokem zajišťují knihovní funkce, u kterých je



Obr. 3.5: Struktura HTTP serveru [15]

možné nastavovat parametry. Jednotlivé podbloky jsou popsány níže. Obsahuje moduly knihoven, inicializaci datových struktur a tabulek, výběr vhodné síťové konfigurace a vložení statických zdrojů pomocí direktiv `ximport` a nebo `zimport`. V průběhu této inicializace se nastavují následující tabulky.

- Tabulka MIME určuje internetovému prohlížeči jaký obsah zobrazuje uživateli.
- Tabulka pravidel, která je potřebná pouze pokud je potřeba přidělovat přístupová práva uživatelů k jednotlivým souborům.
- Tabulka statických zdrojů obsahuje seznam souborů, které aplikace používá.
- Programová pamět FLASH. Pomocí direktiv `ximporta` nebo `zimport` jsou do ní nahrávány zdrojové soubory.

Runtime Inicialization

Tato inicializace probíhá voláním funkcí z hlavní funkce `main()`.

- `sock_init()`. Tato funkce se musí volat vždy. Inicializuje TCP/IP komunikační rozhraní.
- `http_init()`. Inicializuje HTTP server.
- Dále se volají funkce pro nastavení tabulek uživatelů,pravidel a dynamických zdrojů.

Main loop

Nekonečná smyčka uvnitř hlavního programu `main()` opakovaně volá funkci `http_handler()`, která zpracovává příchozí události.

Aplication Specific,I/O

Díky tomuto bloku je HTTP server schopen komunikovat s připojenými periferiemi. Nejčastěji se využívají proměnné za direktivou `#web`,

CGI functions

Obstarává zpracování jazyka CGI, který umožňuje dynamické generování HTML stránek.

HTTP Block

Tento blok zpracovává požadavky přijaté přes ethernet. Jeho práci lze rozdělit do třech kroků. Nejprve analyzuje přijatý požadavek a získá název zdroje, ke kterému chce přistupovat. Poté ověří zdali má uživatel dostatečné oprávnění k přístupu. Nakonec, pokud je vše v pořádku, odešle zpět odpověď (HTML stranka, obrázek ...).

Vnitřní kruh označený HTTP-X je subkomponenta RabbitWeb, což je rozšíření jazyka C, které zjednodušuje prezentaci objektů v internetovém prohlížeči. Rabbit Web umožňuje zpracovávat stránky HTML se speciálními skriptovacími značkami, které jsou při zpracování nahrazeny proměnnými označenými direktivou `#web`. Jedná se o zpracování skriptů na straně serveru.

Toto rozšíření se aktivuje direktivou `#define USE_RABBITWEB 1`.

Druhý kruh označený SSI reprezentuje klasickou cestu generování dynamického obsahu. SSI generuje stejný výstup jako RabbitWeb, ale na rozdíl od něj dokáže zpracovávat skripty CGI.

ZSERVER Block

Nachází se pod blokem HTTP a je nazýván manažer zdrojů. Řídí přístup k většině uvedených bloků. Protože poskytuje rozhraní pro různé druhy zdrojů, může obsluhovat například i FTP server.

Kontrola přístupu

Pokud je požadována kontrola přístupu k jednotlivým zdrojům, nejjednodušší řešení je přidělit jim přístupová práva, definovat skupiny uživatelů a pomocí autentifikace určovat, zda uživatel má nebo nemá k danému zdroji přístup.

V Dynamic C se přístup nastavuje pomocí dvou tabulek.

Tabulka uživatelů

Tato tabulka obsahuje seznam uživatelů a jejich autentifikační informace. Dále je určeno do jaké skupiny patří. Můžeme používat až 16 jednotlivých skupin. Vytvoření uživatele se provede následujím kódem.

- uživatele vytvoříme funkcí
  `userid = sauth_adduser( "jmeno", "heslo", server)`, kde proměnná `server` označuje, ke kterým serverům bude mít uživatel přístup (`SERVER_HTTP`- pouze HTTP server, `SERVER_ANY`- bude mít přístup ke všem serverům).
- přidělení do skupiny uživatelů se provede příkazem
  `sauth_setusermask( userid, skupina_uzivatelov,doplnkove_data)`

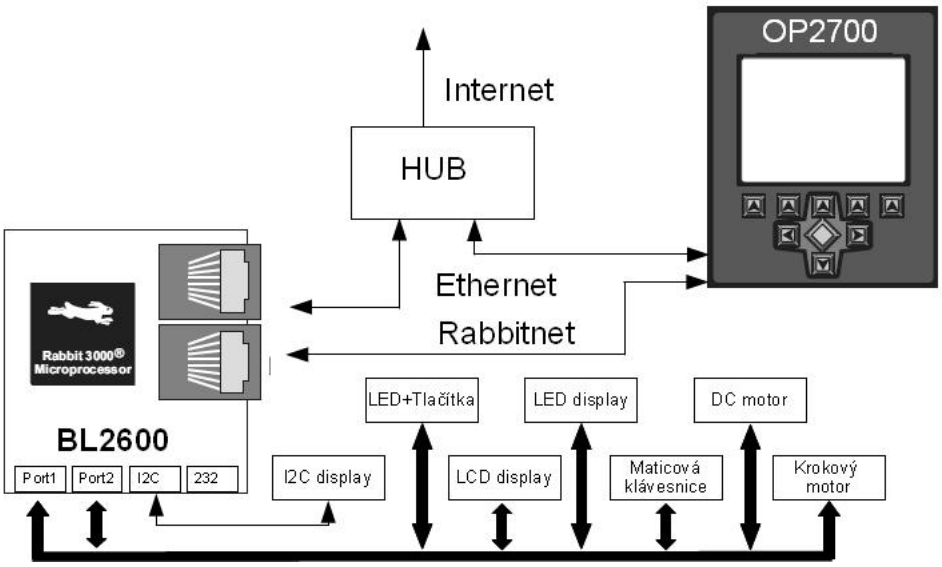
Tabulka pravidel

Tabulka pravidel obsahuje přístupová práva ke zdrojům. Ke každému zdroji jsou v této tabulce uvedeny informace o skupinách, které mají povoleno pouze čtení nebo zápis. Vytvoření pravidla pro přístup k souboru může vypadat následovně
`sspec_addrule( "/index.html", "realm", skupina_ro,skupina_rw, server, metoda_autentifikace,doplnkove_data)`

4 KONSTRUKCE

4.1 Blokové schéma pracoviště

Na Obr. 4.1 je zobrazeno blokové schéma navrženého pracoviště. BL2600 zde plní funkci hlavního počítače, ke kterému se připojují všechny ostatní prvky. Panel OP2700 slouží jako zobrazovací jednotka, která je s BL2600 připojena přes ethernet a pomocí sběrnice rabbit net. Ethernet je rozdělován pomocí switche Linksys přes který je možné přistupovat z obou modulů k internetu.



Obr. 4.1: Blokové schéma pracoviště

Pro každou periferii je navržen vlastní plošný spoj. Veškeré diskrétní součástky jsou použity v pouzdrech SMD, integrované obvody jsou zasazeny do patic, kvůli případné výměně. Jednotlivé periferie se s rozvodnou deskou propojují pomocí vícežilového plochého kabelu. Ten je na obou stranách ukončen samořeznou zásuvkou, která zapadá do konektorů umístěných na deskách. Konektory mají zámky, které při odemčení zásuvku z konektoru vytlačí a tím jdou lehce odpojit. Mělo by se tím zabránit přetrhávání kabelů.

4.2 Rozvodná deska

Jedná se o hlavní součást celého zařízení. Prostřednictvím této desky se k BL2600 připojují ostatní navržené moduly. Rozvodná deska je s BL2600 propojena oboustrannými kolíky, které zapadají do průchozích konektorů osazených na BL2600.

Schéma zapojení je na Obr. A.2 a deska plošného spoje je na Obr. A.3 a Obr. A.4. Plošný spoj je realizován jako oboustranný.

Další moduly se připojují pomocí plochých kabelů zakončených konektorem se zámkem, aby nemohly samovolně vypadnout. Při odpojování je nutné uvolnit zámek umístěný na zásuvce konektoru, čímž se omezí vytrhávání plochých kabelů z konektorů.

4.2.1 Rozvod napájení

Rozvodná deska rozděluje napájecí napětí na všechny porty a také poskytuje napájení samotnému BL2600. Jsou zde umístěny dva konektory pro připojení napájecího napětí. První je pro připojení napájecího adaptéru a druhý konektor je šroubovací svorkovnice, díky které je možné připojit libovolný napájecí zdroj. Oba dva přívody jsou chráněny proti nechtěnému přepólování napájecího napětí. Použití dvou zdrojů bylo nutné zejména kvůli modulům s DC motorem a krokovým motorem. Při použití větších motorků by se mohlo stát, že napájecí adaptér nebude moci dodat dostatečný proud pro jejich napájení. Volba napájecího zdroje se provádí přepnutím posuvného přepínače, který je umístěn na rozvodné desce. BL2600 je vždy napájen přes připojený adaptér.

Je zde také umístěn pomocný zdroj vytvářející ze vstupního napětí, napětí o hodnotě +5 V. Toto napětí je přivedeno na oba vstupně/výstupní porty a je jím napájen obvod pro převod úrovní sériové komunikace MAX232.

K rozvodné desce může být připojeno stejnosměrné napájecí napětí o maximální velikosti 35 V. Při překročení této hodnoty by se mohlo poškodit připojené zařízení.

4.2.2 Výstupní a vstupní porty

Na desce jsou umístěny dva téměř identické vstupně/výstupní porty SV1 a SV2, na které se připojují přídavné moduly. Jediný rozdíl je, že na konektor SV1 obsahuje vstupy DIN16 až DIN23, které se dají přepnout do speciálních funkcí. Tyto funkce jsou využívány pro řízení DC motoru.

V Tab. 4.1 je popis zapojení jednotlivých pinů vstupně/výstupních portů. Každý tento port má osm vstupních pinů a osm dalších pinů, které mohou být softwarově nastaveny jako vstupní nebo jako výstupní. Toto nastavení je možné měnit pouze pro celou osmici pinů.

Kde DCPREP je napájecí napětí, které je vybráno přepínačem. Více je o něm napsáno v kapitole 4.2.1.

Port SV1 (SV2)			
Pin	Připojen	Pin	Připojen
1	DCPREP	2	DCPREP
3	DCPREP	4	DCPREP
5	+5V	6	+5V
7	DIO0 (DI08)	8	DIN16 (DIN24)
9	DIO1 (DI09)	8	DIN17 (DIN25)
11	DIO2 (DI010)	12	DIN18 (DIN26)
13	DIO3 (DI011)	14	DIN19 (DIN27)
15	DIO4 (DI012)	16	DIN20 (DIN28)
17	DIO5 (DI013)	18	DIN21 (DIN29)
19	DIO6 (DI014)	20	DIN22 (DIN30)
21	DIO7 (DI015)	22	DIN23 (DIN31)
23	GND	24	GND
25	GND	26	GND

Tab. 4.1: Zapojení vstupně / výstupních portů

4.2.3 Výkonové výstupy

Na rozvodné desce jsou také vyvedeny všechny čtyři výkonové výstupy, kterými BL2600 disponuje. Jsou umístěny v samostatném konektoru SV3, ke kterému se připojuje modul s krokovým motorem. V Tab. 4.2 je popsáno zapojení jednotlivých pinů.

Každý z těchto výstupů může spínat proud o maximální velikosti 2 A.

4.2.4 Analogové vstupy a výstupy

Všechny analogové vstupy a výstupy, které BL2600 obsahuje jsou připojeny nakonektor SV6

4.2.5 Sériová komunikace

BL2600 disponuje třemi sériovými linkami RS-232. Jedna z nich je použita pro programování a debugování mikrokontroleru, proto není nikde vyvedena. Zbylé dvě sériové linky jsou vyvedeny na konektory. Jelikož standardně komunikují na úrovních RS232 není nutné je dále upravovat a je možné je přímo vyvést na konektory pro připojení sériové linky X1 a X2 typu DSUB9.

Port SV3			
Pin	Připojen	Pin	Připojen
1	+5V	2	+5V
3	HOUT3	4	HOUT3
5	HOUT2	6	HOUT2
7	HOUT1	8	HOUT1
9	HOUT0	8	HOUT0
11	DCPREP	12	DCPREP
13	DCPREP	14	DCPREP
15	GND	16	GND

Tab. 4.2: Zapojení výkonového portu

4.3 Modul LED diod a tlačítek

Modul slouží jako základní vstupně výstupní periferie. Led diody a jednotlivé segmenty zobrazovače se rozsvěcují nastavením úrovně log. „0“ na výstupních portech BL2600. Pomocí tohoto modulu by se studenti měli naučit základní ovládání vstupně - výstupních pinů BL2600.

Na desce je osazena zkratovací propojka, pomocí které můžeme volit zdali bude fungovat sedmisegmentový zobrazovač, nebo čtveřice dvoubarevných led diod. Schéma zapojení je na Obr. A.6, deska plošného spoje je na Obr. A.7 a osazovací výkres je zobrazen na Obr. A.8.

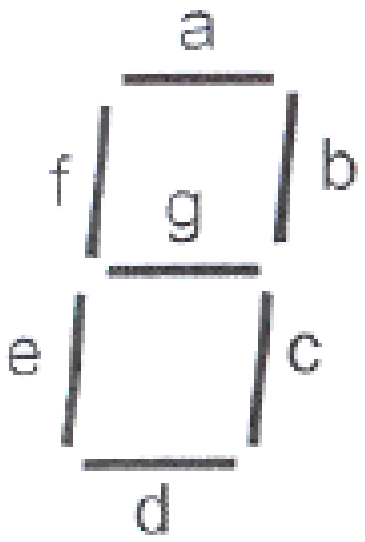
Po osazení všech součástek byl modul připojen plochým kabelem k rozvaděčové desce a jeho funkčnost byla vyzkoušena programem, který je k dispozici jako demonstrační pro ověřování a nastavování vstupů a výstupů desky BL2600.

Modul se dá připojit k oběma vstupně/výstupním portům na rozvaděčové desce.

V Tab. 4.3 je zobrazen sedmisegmentový dekodér. Pro zobrazení jednotlivých znaků je potřeba na výstupní porty nastavit hexadecimalní hodnotu napsanou u každého znaku. Na Obr. 4.2 je popis jednotlivých segmentů dipleje.

4.4 Modul DC motoru

Díky tomuto modulu se dají k BL2600 připojit dva stejnosměrné motory, u kterých je možné řídit směr jejich otáčení a jejich rychlost otáčení. Směr otáčení se řídí nastavením vhodných kombinací výstupů DIO0 až DIO3. Rychlost se řídí pomocí vstupů DIN22 a DIN23.



Obr. 4.2: Sedmisegmentový displej

Znak	h	g	f	e	d	c	b	a	Hex
0	1	1	0	0	0	0	0	0	0xC0
1	1	1	1	1	1	0	0	1	0xF9
2	1	0	1	0	0	1	0	0	0xA4
3	1	0	1	1	0	0	0	0	0xB0
4	1	0	0	1	1	0	1	1	0x9C
5	1	0	0	1	0	0	1	0	0x92
6	1	1	0	0	0	0	0	0	0x82
7	1	1	1	1	1	0	0	0	0xF8
8	1	0	0	0	0	0	0	0	0x80
9	1	0	0	1	1	0	0	0	0x98
A	1	0	0	0	1	0	0	0	0x88
B	1	0	0	0	0	0	1	1	0x83
C	1	1	0	0	0	1	1	0	0xC6
D	1	0	1	0	0	0	0	1	0xA1
E	1	0	0	0	0	1	1	0	0x86
F	1	0	0	0	1	1	1	0	0x8E

Tab. 4.3: Sedmisegmentový dekodér

Pro správnou funkci je nutné, aby tento modul byl připojen k vstupně/výstupnímu portu SV1, protože obsahuje piny DIN22 a DIN23. Jedině na těchto pinech lze hardwarově generovat signál PWM, kterým se řídí rychlost otáčení.

Také je nutné odstranit zkratovací propojku JP4 na desce BL2600.

Schéma zapojení modulu je na Obr. A.15, plošný spoj a osazovací výkres jsou na Obr. A.16 a Obr. A.17.

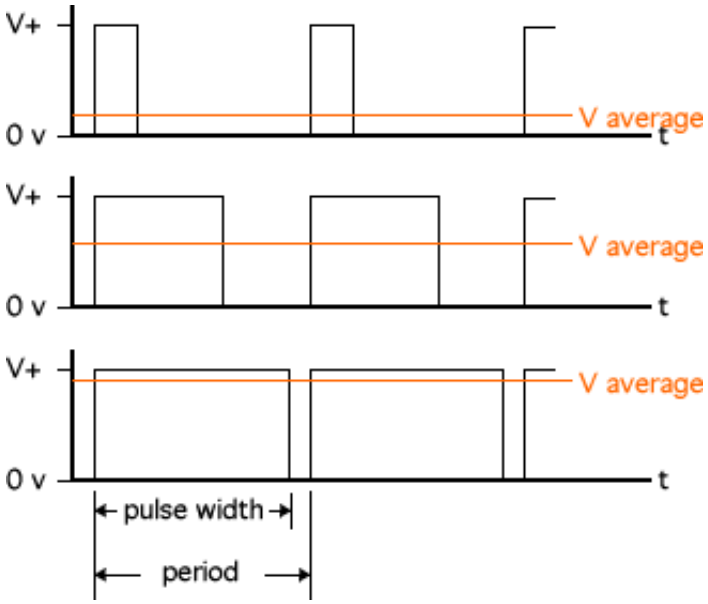
4.4.1 Řízení stejnosměrného motoru

Směr otáčení je dán kombinací výstupů DIO0 a DIO1 nebo DIO2 a DIO3. V Tab. 4.4 jsou zobrazeny kombinace výstupních pinů a jejich funkce.

DIN23 DIN22	DIO0 DIO2	DIO1 DIO3	Funkce
H	L	H	Otáčení vlevo (vpravo)
H	H	L	Otáčení vpravo (vlevo)
H	L	L	Rychlé zastavení motoru
H	H	H	Rychlé zastavení motoru
L	X	X	Rychlé zastavení motoru

Tab. 4.4: Kombinace výstupních pinů a jejich funkce

Rychlost otáčení je dána střídou pulsů signálu PWM, který je vyveden na pinech DIN22 a DIN23. Velikostí střídy se dá regulovat velikost výstupního napětí na motoru v rozsahu 0 až 100 %. Na Obr. 4.3 je nakreslen princip PWM.



Obr. 4.3: Princip PWM řízení

Softwarové řešení

V programu pro ovládání motoru jsou potřebné následující funkce:
`brdinit();` - inicializace BL2600
`digOutConfig(0xFFFF);` - nastavení portů DIO jako výstupy

Podle požadovaného směru otáčení nastavíme kombinaci pinů DIO2 a DIO3.
`digOut(3, 1);` - nastavení pinu DIO3 na hodnotu „1“

`digOut(2, 0);` - nastavení pinu DIO2 na hodnotu „0“

`freq = pwm_init(10000ul);` - inicializace PWM generátoru na frekvenci 10kHz. Funkce vrací hodnotu skutečné frekvence, na které generátor funguje.

`pwm_set(0, pwm, PWM0_OPTION);` - tato funkce nastavuje střídu PWM signálu na určeném kanálu, první parametr je číslo kanálu, který se má nastavit (0 až 3), druhý parametr je hodnota střídy (0 až 1024) a třetí parametr umožňuje nastavení dalších možností.

Pro úplnou funkci je v programu potřeba zajistit možnost, měnit střídu signálu za běhu programu. To je umožněno přes debugovací okno.

4.4.2 Napájení DC motorů

Připojené motory lze napájet ze dvou zdrojů. Volba se provádí přepínačem na rozvodné desce. Prvním zdrojem je napájecí adaptér, který napájí desku BL2600. Druhý zdroj se dá připojit ke svorkovnici umístěné na rozvodové desce.

Napájecí napětí je tudíž rovno hodnotě napětí adaptéru, což je 12 V a nebo velikosti připojeného externího napájení.

Proud, který mohou motory odebírat, je omezen maximálním proudem dodávaným zdrojem. Pokud použijeme napájecí adaptér, odebíraný proud by neměl přesáhnout pro oba motory 1 A. Pokud použijeme externí zdroj, jsme omezeni maximálním povoleným proudem, který může protékat obvodem L293. Tento proud je maximálně 600 mA pro každý motor.

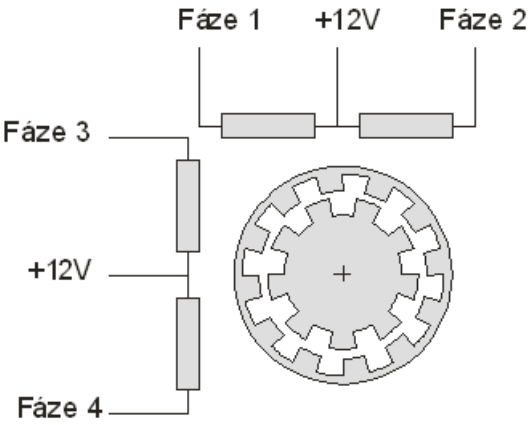
4.5 Modul krokového motoru

Prostřednictvím tohoto modulu se k BL2600 dá připojit jeden unipolární krokový motor.

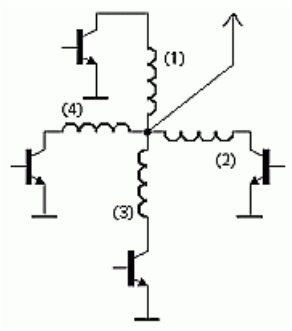
Krokový motor je speciální případ synchroních motorů, který se využíva především tam, kde je potřebné řídit nejen otáčky, ale také konkrétní polohu rotoru.

Pro ovládání krokového motoru jsou použity výkonové výstupy obsažené na desce BL2600. Na rozvodné desce mají umístěn svůj vlastní konektor SV3. Je jimi možno řídit dvoufázový krokový motor v unipolárním zapojení. Schéma takového motoru je ukázáno na Obr. 4.4.

Ukázka zapojení unipolárního dvoufázového krokového motoru je na Obr. 4.5. Kde A a B jsou cívky krokového motoru. Vodiče označené +12 V jsou připojeny na



Obr. 4.4: Schéma krokového motoru



Obr. 4.5: Schéma unipolárního řízení krokového motoru

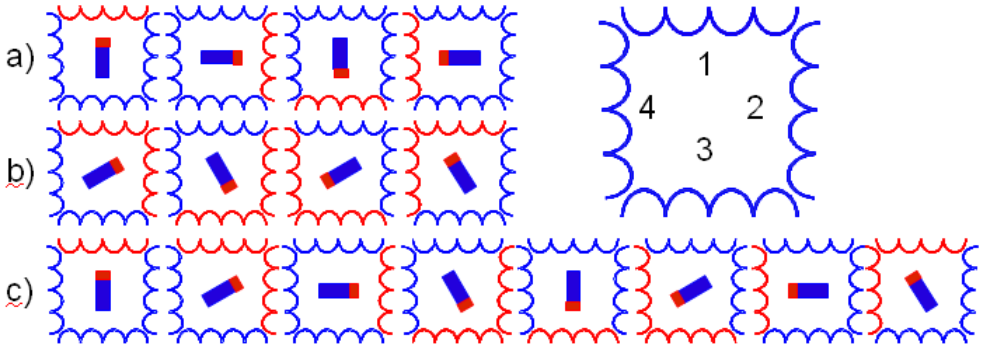
napájecí napětí a vodiče Fáze 1 až Fáze 4 jsou připojeny na výkonové výstupy HK0 až HK3, které připojují zátěž k zemi.

Schéma zapojení je na Obr. A.18 a deska plošného spoje je na Obr. A.19.

4.5.1 Řízení unipolárního krokového motoru

Unipolární krokové motory se dají řídit třemi způsoby. Jednofázově s plným krokem, dvoufázově s plným krokem a dvoufázově s polovičním krokem. Jednofázové a dvoufázové řízení se mezi sebou liší ve velikosti kroutícího momentu. Jelikož při jednofázovém řízení vytváří magnetické pole pouze jedna cívka, má oproti dvoufázovému řízení menší kroutící moment. Řízení s polovičním krokem dokáže za jednu otáčku hřídele udělat dvakrát více kroků než řízení s plným krokem. Na Obr. 4.6 je zobrazen princip těchto tří typů řízení. Červená cívka znamená, že jí protéká proud a tím budí magnetické pole.

V Tab. 4.5 a Tab. 4.6 je popsán postup spínání jednotlivých cívek při různých typech řízení. Jejich číslování je zobrazeno na Obr. 4.6. Jednička zapsaná u jednotlivých cívek znamená, že cívkou protéká proud a tudíž vytváří magnetické pole. Nula znamená, že cívkou proud neprotéká.



Obr. 4.6: Princip řízení unipolárního krokového motoru. a) jednofázové řízení b) dvoufázové řízení s plným krokem c) dvoufázové řízení s polovičním krokem

	Jednofázové řízení					Dvoufázové řízení			
Cívka	1. krok	2. krok	3.krok	4.krok		1. krok	2. krok	3.krok	4.krok
1.	1	0	0	0		1	0	0	1
2.	0	1	0	0		1	1	0	0
3.	0	0	1	0		0	1	1	0
4.	0	0	0	1		0	0	1	1

Tab. 4.5: Postup zapínání cívek u jednofázového a dvoufázového řízení krokových motorů

	Dvoufázové řízení s polovičním krokem							
Cívka	1. krok	2. krok	3.krok	4.krok	5. krok	6. krok	7.krok	8.krok
1.	1	1	0	0	0	0	0	1
2.	0	1	1	1	0	0	0	0
3.	0	0	0	1	1	1	0	0
4.	0	0	0	0	0	1	1	1

Tab. 4.6: Postup zapínání cívek u dvoufázového řízení s polovičním krokem

Softwarové řešení

Pro ovládání krokového motoru budou potřeba následující funkce:
`brdinit();` - inicializace BL2600
`digHoutConfig(0x00);` - nastavení všech výkonových výstupů do režimu spínání k zemi.
`digHout(0, 1);` - nastavení výstupu HOUT0 do hodnoty „0“. Tímto způsobem se nastavují i ostatní výkonové výstupy.

Přepínáním výstupů podle tabulek Tab. 4.5 a nebo Tab. 4.6 se začne rotor motoru otáčet. Rychlost přepínání těchto výstupů je rovna rychlosti otáčení rotoru. Změnou zpoždění mezi jednotlivými přepnutími můžeme regulovat rychlost otáčení. Směr otáčení se mění změnou směru přepínání výstupů.

4.5.2 Napájení krokových motorů

Napájení krokového motoru je řešeno stejně jako napájení u stejnosměrného motoru. Modul je možné napájet ze dvou zdrojů, mezi kterými se přepíná pomocí přepínače na rozvodové desce.

Krokové motory je třeba vybírat podle proudu, protékajícího všemi najednou sepnutými cívkami tak, aby tento proud nepřekročil maximální velikost proudu, který je vybraný zdroj schopen dodat a nebo výkonové tranzistory sepnout.

4.6 Modul maticové klávesnice

Jedná se o modul, prostřednictvím kterého je možné připojit maticovou klávesnici k BL2600. Obsahuje také čtyři LED diody a čtyři obyčejná tlačítka.

Výhoda maticové klávesnice je, že potřebuje méně vstupních pinů než kdybychom připojovali každé tlačítko zvlášť. Toto platí pouze tehdy, připojujeme-li více než čtyři tlačítka.

Po osazení této desky bylo zjištěno, že poloha montážních otvorů na klávesnici se neshoduje s otvory vyvrtanými na desce modulu. Proto bylo nutno do desky vyvrtat otvory na správných místech. Bylo to možné díky tomu, že vodiče na spodní straně desky nevedly těmito místy.

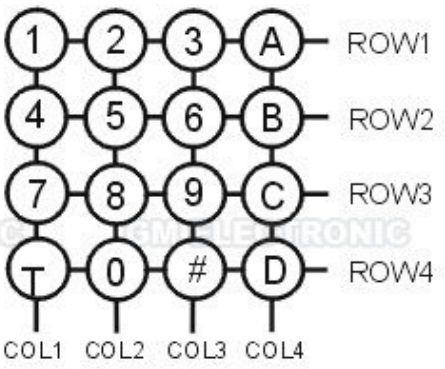
Schéma zapojení je na Obr. A.9,deska plošného spoje a osazovací výkres jsou na Obr. A.10 a Obr. A.11.

Modul se dá připojit k oběma vstupně/výstupním portům na rozvaděčové desce.

4.6.1 Čtení kláves maticové klávesnice

Pro ověření funkčnosti modulu byl napsán jednoduchý program pro čtení stisknutých kláves, který používal metodu postupující nuly.
Tato metoda funguje následovně:
Nastaví se hodnota log. „0“ na prvním sloupci a poté se čte na jakém vstupním portu se objeví log. „0“. Pokud se neobjeví, nastaví se nula na dalším sloupci a opět se čtou hodnoty na vstupních portech. Když se na některém vstupním portu objeví log. „0“ znamená to, že stisknuté tlačítko leží na spojnici sloupce s nastavenou log. “0“ a vstupního portu, kde se objevila logická „0“. Vývojový diagram je nakreslen na Obr. 4.8.

Na Obr. 4.7 je zobrazeno vnitřní zapojení maticové klávesnice. V Tab. 4.7 je ukázáno připojení jednotlivých pinů maticové klávesnice k vstupním a výstupním portům BL2600.



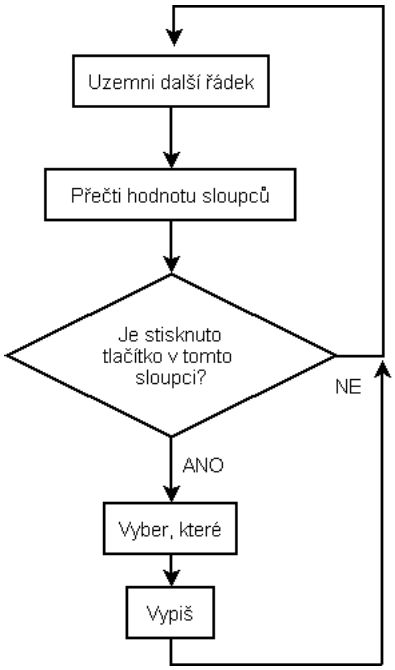
Obr. 4.7: Zapojení maticové klávesnice

Pin	Označení	Připojení
1	COL1	DIO12
2	COL2	DIO13
3	COL3	DIO14
4	COL4	DIO15
5	ROW1	DIN31
6	ROW2	DIN30
7	ROW3	DIN29
8	ROW4	DIN28

Tab. 4.7: Připojení maticové klávesnice

Softwarové řešení

Pro ovládání maticové klávesnice budeme potřebovat následující funkce:
`brdinit();` - inicializace BL2600
`digOutConfig(0xFFFF);` - nastavení portů DIO jako výstupy
`digOut(12,1);` - nastavování jednotlivých sloupců na hodnotu „0“ nebo „1“ tak, jak je potřeba u metody postupující nuly. Vždy nastavujeme výstupy tak, aby v jednom sloupci byla „0“ a v ostatních „1“.
`reading = digIn(28);` - čtení hodnoty na vstupu DIN28. Postupným čtením všech vstupů DIN28 až DIN31 zjistíme jaké tlačítko bylo stisknuto.



Obr. 4.8: Postup vyhodnocování maticové klávesnice

4.7 Modul LCD displeje

Modul LCD displeje slouží k připojení LCD displeje s řadičem k BL2600. Použil jsem displej MC2004B, který obsahuje řadič S6A0069. Jde o řadič kompatibilní s hojně používaným řadičem HD4770.

Modul se dá připojit k oběma vstupně/výstupním portům na rozvaděčové desce.

4.7.1 Vlastnosti displeje

Komunikace může probíhat čtyřbitově nebo osmibitově. To znamená, že data k displeji jsou přiváděna po paralelní osmivodičové nebo čtyřvodičové sběrnici. U navrženého modulu byla použita čtyřvodičová sběrnice, z důvodu úspory výstupních pinů. Rozdíl mezi těmito dvěma způsoby je pouze v tom, že u čtyřvodičového zapojení jsou data přenášena ve dvou cyklech oproti osmivodičové sběrnici. Z toho vyplývá, že komunikace probíhá dvakrát pomaleji.

Displej je čtyřřádkový s dvaceti znaky na řádku. Každý znak má rozlišení 5x8 bodů. České znaky se dají nahrát do vyhrazené paměti na začátku paměťového prostoru řadiče.

Displej má připojeno podsvícení, které se dá zkratovací propojkou vypnout nebo zapnout. Také se dá nastavit kontrast displeje pomocí trimru umístěného na plošném

spoji modulu. Samotný displej je k plošnému spoji modulu připojen přes konektorovou lištu a připevněn čtyřmi distančními sloupky.

V Tab. 4.8 je zobrazeno připojení jednotlivých pinů displeje k BL2600.

Pin	Označení	Popis	Připojen k
1.	Vss	Napájení 0V	GND
2.	Vdd	Napájení 5V	+5V
3.	V0	Ovládání kontrastu	+5V
4.	R/S	Data / Instrukce	DIO9
5.	R/W	Čtení / Zápis	DIO10
6.	E	Povolení	DIO11
7.	D0	Data	GND
8.	D1	Data	GND
9.	D2	Data	GND
10.	D3	Data	GND
11.	D4	Data	DIO12
12.	D5	Data	DIO13
13.	D6	Data	DIO14
14.	D7	Data	DIO15
15.	A	Anoda osvětlení	+5V
16.	K	Katoda osvětlení	GND

Tab. 4.8: Zapojení LCD displeje

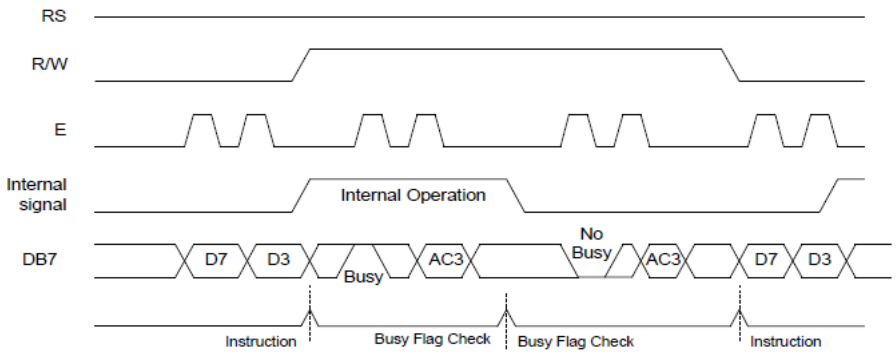
4.7.2 Komunikace s displejem

Jak bylo napsáno výše, komunikace probíhá čtyřbitově. To znamená, že se používají pro přenos dat pouze vyšší čtyři komunikační linky a data se po nich přenášejí nadvakrát. Příklad komunikace s displejem je na Obr. 4.9 [6].

Komunikace probíhá následovně:
Na sběrnici se vystaví nižší 4 bity dat a potvrdí se signálem EN. Poté se na sběrnici vystaví vyšší 4 bity dat a opět se potvrdí změnou úrovně signálu EN. Signálem R/W se určuje, jestli data z uvedené adresy budeme číst nebo zapisovat. Signál RS určuje, zda jsou na sběrnici vystavena data nebo kód instrukce.

Inicializace:

Pro správnou funkci displeje je nejprve potřeba provést jeho inicializaci. Jednotlivé kroky inicializace jsou popsány v Obr. A.1a nebo v [6]. Inicializace probíhá ve čtyřech



Obr. 4.9: Čtyřbitová komunikace s displejem [6]

krocích.

V prvním kroku se určuje, jakým způsobem se s displejem bude komunikovat,jestli čtyřbitově nebo osmibitově. Dále se určí na kolika řádcích se má text zobrazovat a zapne se displej.

V druhém kroku se nastavují vlastnosti kurzoru. Ten může být vidět a nebo být skrytý. Také se určí zda má blikat nebo být stále viditelný.

V třetím kroku se celý displej smaže a kurzor se nastaví na první pozici.

Ve čtvrtém kroku určíme má-li se kurzor po zapsání znaku posunovat doleva nebo doprava a jak se má displej chovat po zapsání všech znaků.

Po úspěšně dokončené inicializaci můžeme displej používat jako zobrazovač.

Zápis:

Znaky na displeji se zobrazí tak, že jejich hodnotu zapíšeme do paměti DDRAM řadiče, na kterou odkazuje ukazatel na paměť. Po každém zapsaném znaku se ukazatel posune na další paměťovou buňku. Každé pozici na displeji odpovídá určitá adresa DDRAM.

Postup zápisu je zobrazen na Obr. 4.9. Nastavíme signál R/W do „0“ a na sběrnici se vystaví čtyři nižší bity dat. Poté je zápis potvrzen pulsem signálu E. To samé se opakuje pro vyšší čtveřici bitů dat. Po zapsaní vyšších čtyřech bitů je kontrolován signál „Busy“, který signalizuje nemožnost posílání dalších dat. Pokud se signál „Busy“ nachází v úrovni log. „0“, je řadič připraven pro příjem dalších dat.

Softwarové řešení

Pro vypisování na displej budeme potřebovat následující funkce:
`brdinit();` - inicializace BL2600
`digOutConfig(0xFFFF);` - nastavení portů DIO jako výstupy

`digOut(9,1);` - nastavováním jednotlivých signálů DIO09 až DIO15 na „0“ nebo „1“ tak jak je popsáno na Obr. 4.9 může být řízena funkce displeje. Pro jednodušší ovládání byla vytvořena funkce, která podle hexadecimálního čísla nastavuje jednotlivé piny na „1“ nebo „0“, tak jak je tomu u programování osmibitových kontrolerů.

4.8 Modul I2C

Tento modul slouží k propojení BL2600 s dalšími integrovanými obvody prostřednictvým sběrnice I2C. Při osazování desky plošného spoje bylo zjištěno, že při návrhu byla použita špatná knihovna. Z tohoto důvodu byly vývody u maticových displejů umístěny blíže než ve skutečnosti. Proto musela být tato deska předělána. Její schéma a plošný spoj jsou na Obr. A.21, Obr. A.23 a Obr. A.22.

Modul obsahuje dva integrované obvody MAX6953, které se starají o ovládání osmi maticových displejů o velikosti 5x7 bodů. Dále jsou zde umístěny hodiny reálného času PCF8583P, které jsou bateriově zálohované. Je zde osazen také osmibitový expandér PCF8574 a také dvě paměti. PCF8570 je paměť typu SRAM o velikosti 256B a AT24C04, která je paměť typu EEPROM o velikosti 4kB.

Jelikož komunikace I2C je řešena softwarově pouze na některých pinech, je potřeba aby tento modul byl připojen ke vstupně / výstupnímu portu SV2.

4.8.1 Obvod MAX6953

Jedná se o řadič maticových displejů komunikující po sběrnici I2C. Jeden tento obvod může řídit až čtyři maticové displeje o velikosti 5x7 led diod. Je k němu možné připojit displeje, kde v řádku jsou umístěny katody led diod a ve sloupci jejich anody.

Adresa obvodu se nastavuje přepojováním zkratovacích propojek JP2 a JP3. Těchto kombinací může být až 16. Z toho vyplývá, že na jedné sériové lince může být zapojeno až 16 těchto obvodů. Na našem modulu jsou použity pouze dva.

4.8.2 Obvod PCF8583

Jedná se o hodiny reálného času komunikující po sběrnici I2C. Při odpojení napájení jsou zálohované osazenou baterií CR2430, která postačuje obvodu pro správnou funkci a k udržení dat v jeho paměti. Obvod disponuje také pamětí RAM o velikosti 240 B. Na vstup DIN26 je připojeno přerušení od tohoto obvodu, které se vyvolá v čase na který byl nastaven alarm. Adresa se nastavuje zkratovací propojkou JP14.

4.8.3 Obvod PCF8574

Obvod PCF8574 je osmibitový expandér, který komunikuje po sběrnici I2C. Přes sběrnici I2C můžeme libovolně nastavovat hodnoty na jeho pinech a nebo tyto hodnoty číst. Pokud se změní hodnoty na jeho vstupech, může se vyvolat přerušení. To je signalizováno na vstupu DIN26 změnou logické úrovně. Adresa se nastavuje zkratovacími propojkami JP6 až JP8.

4.8.4 Obvod PCF8570

Jde o paměť typu RAM komunikují po sběrnici I2C. Její velikost je 265 B. Adresa se nastavuje zkratovacími propojkami JP9 až JP11.

4.8.5 Obvod AT24C

Tento obvod je paměť typu EEPROM o velikosti 4kB komunikující po sběrnici I2C. Paměť se dá zamknout proti zápisu pomocí propojky JP1. Adresa se nastavuje zkratovacími propojkami JP15 až JP17.

Všechny použité obvody mají nastavitelný alespoň jeden bit z adresy. Při nastavování adres je potřeba dát pozor na to, abychom dvěma zařízením nenastavili stejnou adresu. Potom by správně nekomunikovalo ani jedno z nich.

Jelikož se deska musela předělat, nebylo možné ověřit její funkčnost.

5 ŘEŠENÉ ÚKOLY

Cílem práce bylo také navrhnout několik úloh, které by měli studenti řešit v hodinách počítačových cvičení. Pro každou úlohu je napsáno podrobné zadání, její řešení a je přiložena vývojový diagram.

5.1 Ovládání digitálních vstupů a výstupů

5.1.1 Úvod

Jednodeskový počítač BL2600 obsahuje 16 digitálních vstupů a 16 pinů, u kterých lze nastavit jejich funkci. To znamená, že mohou plnit jak funkci digitálních vstupů tak i funkci digitálních výstupů. Na Obr. 2.1 je znázorněno zapojení digitálních vstupů a na Obr. 2.2 je zobrazeno zapojení digitálních výstupů. Jak je ze zapojení vidět, je možné měnit velikost napětí při výstupu/vstupu v hodnotě log. „1“. Tato úroveň může být rovna velikosti napájecího napětí, napětí připojeného na svorku +K a nebo hodnotě +5 V. Podrobný popis je popsán v kapitole 2.1.1

První úloha slouží k seznámení s vývojovým prostředím Dynamic C a funkcí vstupních a výstupních pinů.

Pro demonstraci úlohy bude potřeba připojit modul LED diod a tlačítek. Jeho podrobný popis je v kapitole 4.3.

5.1.2 Zadání

1. Vytvořte program, který bude při stisku tlačítka náhodně zobrazovat na sedmisegmentovém displeji čísla od 0h do Fh. Opětovné losování se spustí stiskem druhého tlačítka.
2. Upravte předchozí program tak, že maximální velikost losovaného čísla se rovná binární kombinaci přepínačů 1 až 4 umístěných na používaném modulu.

5.1.3 Přehled funkcí

```
void brdInit(void);
```

Inicilizace modulu BL2600

```
void digOutConfig(int configuration);
```

Nastavení vstupně výstupních pinů. `configuration` je 16-ti bitové číslo určující, který pin zůstane vstupní, který se přenastavý jako výstupní.

```
char digInBank(int bank);
```

Vrací hodnotu typu `char`, která odpovídá kombinaci hodnot nastavených na vstupních pinech. `bank` určuje jaký port se bude číst. Podrobné zobrazení je v Tab. 3.2.

```
int digIn(int channel);
```

Funkce vrací „1“ nebo „0“ podle stavu pinu určeného parametrem `channel`.

```
void digOutbank(char bank, char data);
```

Nastaví na výstupní porty vybrané pomocí parametru `bank` kombinaci hodnot odpovídající hodnotě `data`. Podrobný popis je v Tab. 3.1.

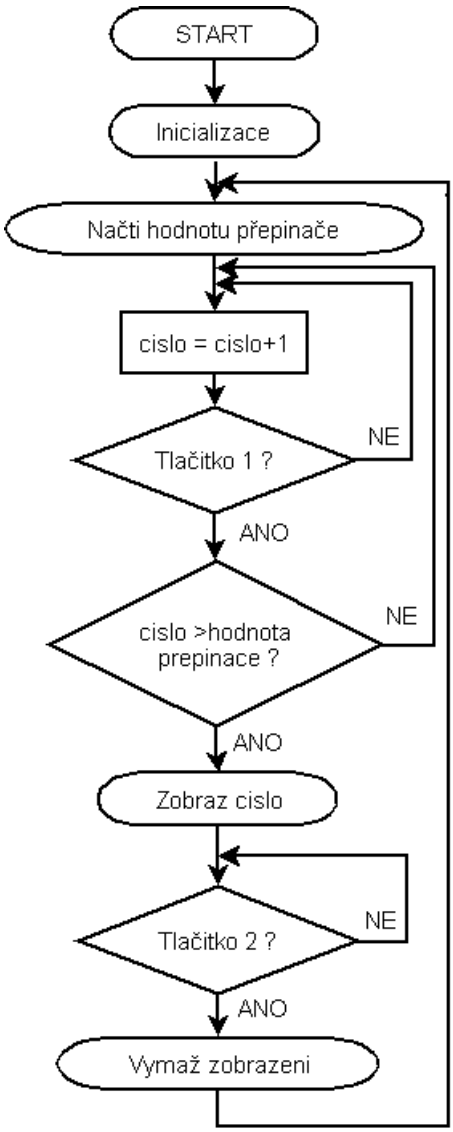
```
void digOut(int channel, int state);
```

Nasataví pin označený parametrem `channel` do stavu log.“0“ nebo log,“1“, podle hodnoty druhého parametru `state`.

5.1.4 Řešení

Nejdříve musíme inicializovat modul BL2600. Poté je potřeba nastavit vstupně výstupní piny DIO jako výstupní. Toho je docíleno zavoláním funkce `digOutConfig` s parametrem 0xFFFF. Dále v nekonečné smyčce vždy načteme hodnotu vstupů a určíme v jakém stavu jsou jednotlivé přepínače. Jejich stav je určen horními čtyřmi bity načtené hodnoty. Hodnotu, kterou nesmí losované číslo překročit jednoduše získáme bitovým posunem o čtyři bity doprava. Poté losujeme číslo, které může nabývat hodnot od nuly do dané maximální hodnoty. Losování probíhá tak dlouho dokuď není stisknuto první tlačítko. Po stisknutí prvního tlačítka se zobrazí hodnota na sedmisegmentovém displeji. Zobrazení je provedeno přes sedmisegmentový dekodér, který je vytvořen ve funkci `Zobraz();`. Dekodér je realizován přepínačem `switch`, který pro každou možnou hodnotu vylosovaného čísla, nastaví na výstup takovou hodnotu aby se číslo korektně zobrazilo na sedmisegmentovém displeji. Jednotlivé výstupní hodnoty jsou pro všechny čísla napsány v Tab. 4.3. Číslo je zobrazeno do té doby, než je stisknuto druhé tlačítko. Poté se celý cyklus opakuje. Vývojový diagram celého programu je zobrazen na Obr. 5.1

5.1.5 Realizace



Obr. 5.1: Vývojový diagram programu pro ovládání led diod

5.2 Ovládání maticové klávesnice

5.2.1 Úvod

Hlavní výhodou maticové klávesnice je, že pro její připojení je potřeba méně vstupů, než kdybychom jednotlivá tlačítka připojovali samostaně. Tento způsob zapojení tlačítek je vhodný použít při potřebě obsluhovat více než 5 tlačítek.

Modul maticové klávesnice připojte k jednomu ze dvou portů pomocí plochého kabelu. Nejčastější způsob čtení maticové klávesnice je použití metody postupující nuly. Tento algoritmus je popsán v kapitole 4.6 věnující se tomuto modulu a vývojový diagram je zobrazen na Obr. 4.8.

5.2.2 Zadání

1. Vytvořte program, který bude vyhodnocovat stisknuté tlačítko na maticové klávesnici.
2. Vypisujte přečtené znaky do stdio okna aplikace Dynamic C.

Obsluha maticové klávesnice

Na Obr. 4.8 je ve vývojovém diagramu zobrazen postup vyhodnocování stisknutých tlačítek.

5.2.3 Přehled funkcí

```
int printf ( const char * format, ... );
```

Jedná se o funkci pro standardní výstup. V prostředí Dynamic C se výpisy provádějí do okna Stdio, které je součástí vývojového prostředí.

`MS_TIMER`

Proměnná, která je automaticky každou milisekundu inkrementuje. Nejčastěji se používá pro vytváření zpoždění. Pokud by nevyhovovala tato proměnná, z důvodu potřeby přesnějšího měření času je možné stejně používat TICK_TIMER, který se intrementuje 1024-krát za vteřinu.

5.2.4 Řešení

Na začátku programu nejdříve inicializujeme modul BL2600 a vstupně-výstupní piny. Poté v nekonečné smyčce opakujeme následující postup. Na prvním sloupci klávesnice se nastaví hodnota log. "0". Ostatní sloupce se nastaví do log. „1". Poté se čte jestli se na nějakém řádku objevila hodnota log. „0". Pokud ano, určíme podle kombinace sloupce a řádku o jaké tlačítko se jedná. Pokud ne, nastaví se hodnota log. „0" na dalším sloupci a opět se kontroluje, zdali je na některém řádku hodnota log. "0". Toto se periodicky opakuje. Znak stisklého tlačítka vypíšeme pomocí funkce `prinft();` do okna stdio. Tento znak získáme z dvojrozměrného pole, které jsme si na začátku vyplnili.

5.3 Ovládání krokového motoru

5.3.1 Úvod

K modulu krokového motoru se připojuje unipolární krokový motor. Rychlost jeho otáčení je úměrná rychlosti pulsů generovaných spínáním jednotlivých výstupů. Směr otáčení závisí na posloupnosti spínání jednotlivých cívek motoru. Podrobně jsou vlastnosti unipolárních motorů a způsoby jejich řízení popsány v kapitole 4.5.

Tento jediný modul využívá výkonové výstupy umístěné na rozvodové desce. Zapojení těchto pinů je popsáno v kapitole 2.1.1. Jednotlivé piny dokáží spínat proud až 2 A, ale ve skutečnosti je tento proud omezen použitým napájecím zdrojem. Rozvod napájecího napětí je zobrazen a popsán v kapitole 4.2.1.

5.3.2 Zadání

1. Vytvořte program, kterým bude možno ovládat směr otáčení a rychlost otáčení krokového motoru. Ovládání provádějte přes STDIO okno prostředí Dynamic C.
2. Upravte program tak, aby bylo možné přepínat mezi různými druhy řízení. Použijte všechny typy řízení uvedené v kapitole 4.5.

5.3.3 Přehled funkcí

```
int kbhit( void );
```

Tato funkce detekuje stisknuté klávesy v stdio okně v prostředí Dynamic C. Pokud nebyla stisknuta žádná klávesa funkce vrací hodnotu 0. Jinak vrací hodnoty různé od 0.

```
char getchar( void );
```

Načte znak zapsaný do stdio okna.

```
void digHoutconfig( char configuration);
```

nastaví výkonové výstupy jako sinking nebo sourcing výstup, kde hodnota 1 znamená sourcingovy typ výstupu a hodnota 0 výstup typu sinking.

```
void digHout( int channel, int state );
```

Nastavý na výkonovém pinu označeném parametrem channel hodnotu state. Hodnoty mohou být buď log.„0“, log.“1“ a nebo stav vysoké impedance.

5.3.4 Řešení

V navrženém programu je vytvořeno menu, které je vypisováno do stdio okna. Jednotlivé volby se zadávají pomocí čísel. V menu můžeme volit rychlost otáčení, způsob řízení a směr otáčení motoru. Samotný program je rozdělen do dvou tasků, kde první úloha obstarává obsluhu menu a druhá úloha se stará o generování pulsů pro řízení motoru.

Na začátku programu byly vytvořeny tři pole prvků typu char. Každé pole je určeno pro jeden typ řízení. Jsou v nich zapsány hodnoty výkonových výstupů tak, jak jsou ukázány v Tab. 4.5 a Tab. 4.6.

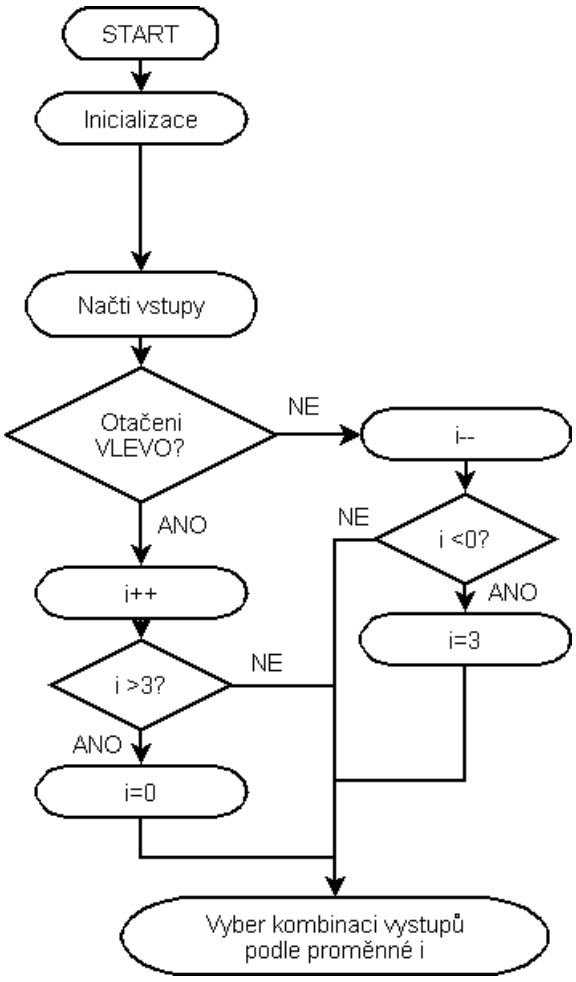
První úloha obstarává obsluhu menu a podle zadaných voleb řídí otáčení motoru. Jednotlivé volby se snímají funkcí `getchar();` a poté jsou pomocí přepínače zpracovány. Když se mění způsob řízení motorů je pouze změněn ukazatel, který ukazuje na dříve vytvořená pole kroků.

Druhá úloha generuje pulsy podle hodnot uložených ve vytvořených polích. V každém průběhu programu se inkrementuje ukazatel na pole a tím ukazuje na další prvek, který je poté vystaven na výstup. Vždy když se dojde na konec pole je ukazatel přesměrován opět na jeho začátek.

Rychlost otáčení se reguluje změnou zpoždění mezi jednotlivým přepínáním výstupů. Směr otáčení je závisí na směru procházení pole kroků. Pokud budeme procházet pole od prvního do posledníhoprvku a tyto prvky dávat na výstup, bude se motor otáčet vpravo. Pokud budeme postupovat z opačné strany, bude se motor otáčet opačně.

Vývojový diagram programu je zobrazen na Obr. 5.2.

5.3.5 Realizace



Obr. 5.2: Vývojový diagram programu pro ovládání krokového motoru

5.4 Ovládání DC motoru

5.4.1 Úvod

Rychlost stejnosměrného motoru se dá jednoduše řídit změnou jeho napájecího napětí. Směr otáčení se změní přepólováním napájecího napětí.

Připojovaný DC motor je ovládán pomocí můstku L293, který je schopen spínat proud až do velikosti 1,2 A. Díky tomuto obvodu lze ovládat dva stejnosměrné motory. A to jak jejich směr tak i rychlost jejich otáčení. Podrobně je řizeni DC motorů popsáno v kapitole 4.4.1. Tento modul je možné připojovat pouze k portu SV1, protože pouze ten obsahuje piny DIN22 a DIN23, kterými lze generovat signál

PWM. Tento signál se používá k regulování rychlosti otáčení. Směr otáčení se nastavuje vhodnou kombinací urovní na pinech DIO0 (DIO2) a DIO1 (DIO3).

Pro správnou funkci je potřeba na BL2600 odstranit zkratovaci propojku JP4.

5.4.2 Zadání

1. Vytvořte program, kterým bude možné ovládat směr otáčení a rychlost otáčení DC motoru. Ovládání provádějte přes STDIO okno prostředí Dynamic C.

5.4.3 Přehled funkcí

```
unsigned long pwm_init( unsigned long frequency );
```

Nastaví základní frekvenci pro PWM signál u všech čtyř PWM výstupů. Parametrem `frequency` se zadává požadovaná frekvence PWM signálu. Funkce vrací hodnotu skutečně nastavené frekvence.

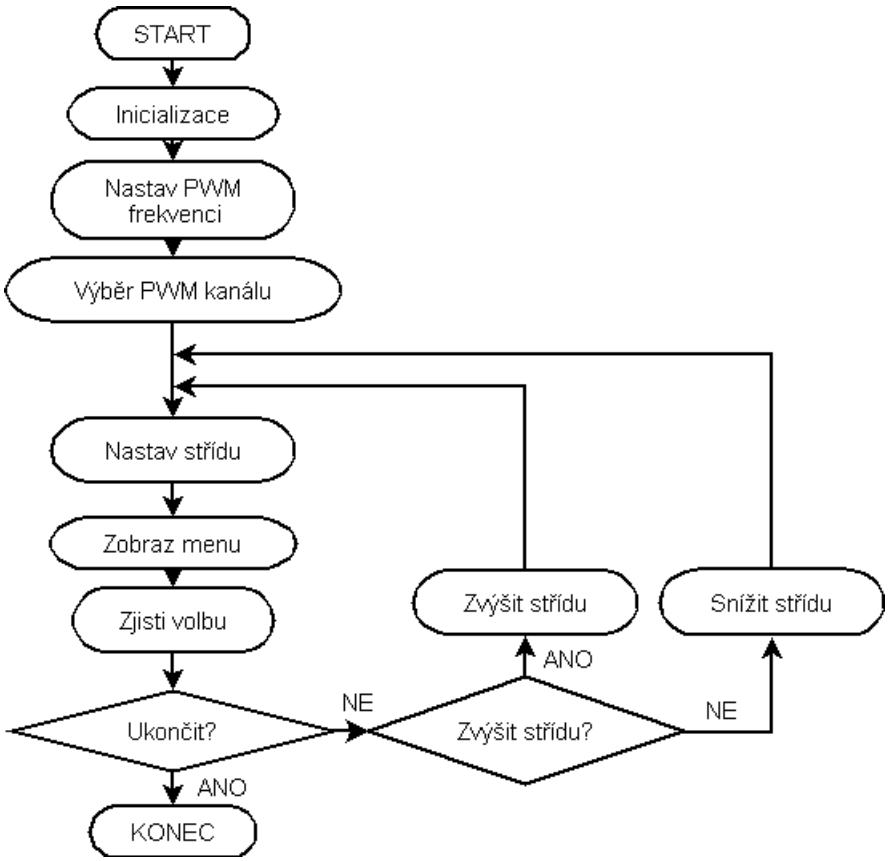
```
int pwm_set( int channel, int duty_cycle, int options );
```

Funkce nastavuje střídu PWM signálu a tím i velikost výstupního napětí. Parametrem `channel` vybíráme PWM výstup, druhý prametr nastavuje střídu signálu. Ta může nabývat hodnoty od 0 do 1024 (0 - 100%). Třetím parametrem se dají upravit vlastnosti PWM výstupu.

5.4.4 Řešení

Program je řešem podobně jako program pro ovládání krokového motoru. Pomocí stdio okna volíme směr a rychlost otáčení stejnosměrného motoru. Nejdříve je potřeba inicializovat modul BL2600 a výstupní porty, které jsou používány k řízení obvodu LM293. Dále je potřeba nastavit funkcí `pwm_init` požadovanou frekvenci PWM signálu. Poté už jen volíme rychlost otáčení změnou střídy signálu. Tato změna je prováděna voláním funkce `set_pwm` se spávně zadanými parametry. Směr otáčení se volí vhodným nastavením výstupních pinů, tak jak je napsáno v Tab. 4.4. Vývojový diagram programu je zobrazen na Obr. 5.3.

5.4.5 Realizace



Obr. 5.3: Vývojový diagram programu pro ovládání DC motoru

5.5 Ovládání LCD displeje

5.5.1 Úvod

K modulu BL2600 se připojuje LCD displej MC2004B, který obsahuje řadič S6A0069. Jde o řadič kompatibilní s hojně používaným řadičem HD4770. S tímto displejem se komunikuje po osmivodičové sběrnici, kde čtyři vodiče přenášejí data a zbylé čtyři slouží k řízení displeje. Protože pro přenos dat mámé k dispozici pouze čtyři vodiče, musí se přenášet ve dvou částech. První se posílají čtyři vyšší bity dat a poté následují zbylé čtyři bity.

Modul je možno připojit na oba porty na rozvodné desce. Pro změnu kontrastu je na modulu umístěn trimr. Je zde také umístěna zkratovací propojka pro vypnutí/zapnutí podsvětlení displeje.

5.5.2 Zadání

1. Vytvořte program, který bude komunikovat s připojeným LDC displejem. Vytvořte funkce pro jeho inicializaci, řízení, mazání a pro zobrazování znaků.
2. Napsaný program vylepšete o funkci, která dokáže zobrazovat na displeji celé řetězce znaků postupně po řádcích.

5.5.3 Přehled funkcí

Pro ovládání displeje jsme vytvořili několik funkcí.

```
void WriteCommand(int prikaz);
```

Tato funkce složí pro zasílání jednotlivých příkazů displeji.

```
void inicializace (void);
```

Jedná se o funkci, která provede inicializaci displeje a připraví ho pro zobrazování přijatých dat.

```
void WriteData(int data);
```

Slouží k posílání dat na displej.

```
void ClearDisp(void);
```

Po zavolání této funkce se celý displej smaže a kurzor se přesune na výchozí pozici.Využívá funkci `WriteCommand(int prikaz);`.

```
void gotoaddress(int address);
```

Přesune kurzor na zadanou adresu.

```
void printLine( char* line);
```

Funkce zobrazí na displeji celý řetězec znaků označený ukazatelem `line`. Pro řetězce delší než jeden řádek správně přesouvá kurzor tak, aby se text vypisoval na správných místech displeje(adresách).

5.5.4 Řešení

Nejprve je provedena inicializace BL2600 a výstupních portů. Nejprve je nutné displej inicializovat. To znamená, že určíme jakým způsobem se s displjem bude komunikovat a kolik řádků má použitý displej, poté můžeme nastavit zda chceme, aby byl kurzor vidět, aby blikal a na jako stranu se má posouvat po zobrazení znaku. Po úspěšně provedené inicializaci už můžeme na displej posílat data, která se zobrazí.

Největší problém při inicializaci je, že po zapnutí displeje se s ním komunikuje jakoby osmibitově, a až poté je přepnut do čtyřvodičového režimu. Zde nastává problém v tom, že po restartu BL2600 a nevypnutí displeje, se opět začíná komunikovat osmibitově, ale displej je stále nastaven ve čtyřbitovém módu (nebyl vypnut, vypnul se pouze modul BL2600). Abychom toto napravili je na začátku inicializace displeji čtyřbitovou formou komunikace nařízeno, aby se přepnul do osmibitového módu a poté je provedena celá inicializace. To znamená přepni se do čtyřbitového režimu, nastav dvouřádkový displej a nastav vlastnosti kurzoru. Celý postup inicializace je rozkreslen v Obr. A.1.

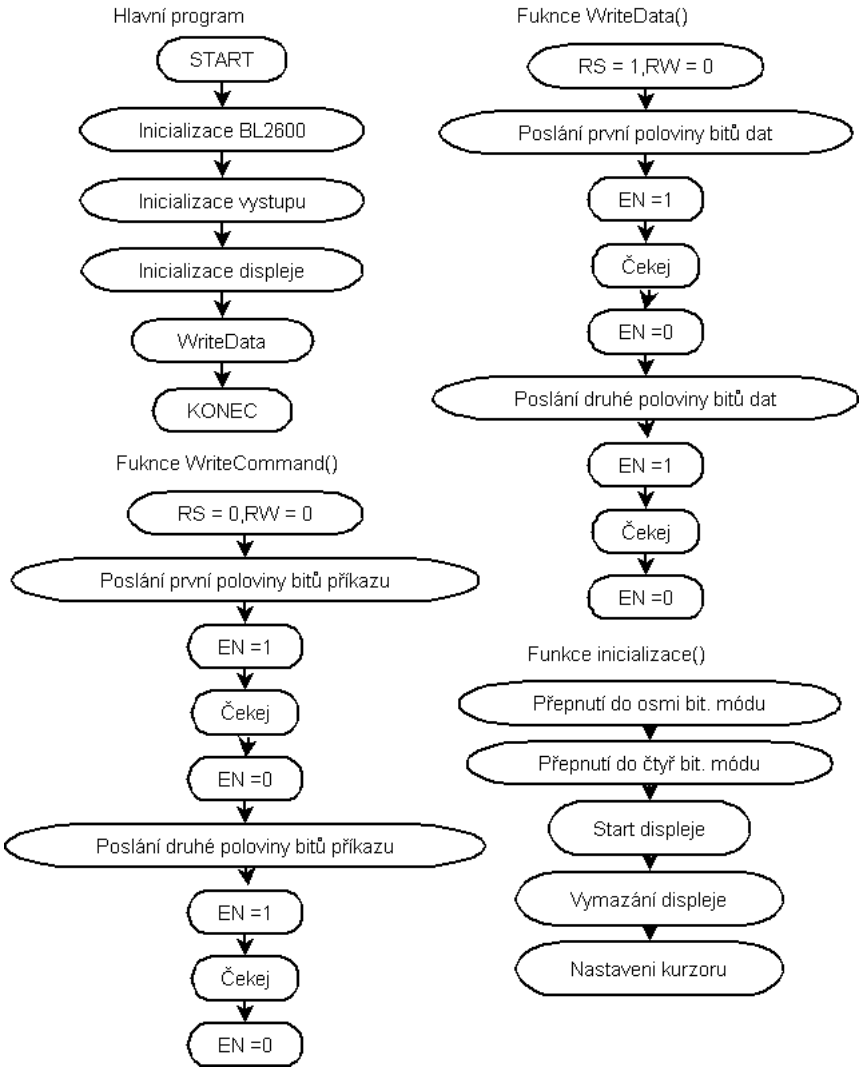
Dalším problémem je, že řadič displeje je ve skutečnosti dvouřádkový, kde každý řádek má 40 znaků. U použitého displeje je tedy každý řádek rozdělen na dvě poloviny. V Tab. 5.1 jsou zapsány adresy jednotlivých pozic znaků. Jak je z tabulky vidět, tak druhá polovina prvního řádku nezačíná hned pod první polovinou prvního řádku, ale až pod první polovinou druhého řádku. To ztěžuje práci při zobrazování dlouhých řetězců, které přesahují délku jednoho řádku. Proto je nutné provádět opravu pomocí funkce `gotoaddress(int address)`.

Průběh programu pro ovládání LCD displeje je na Obr. 5.4.

Pozice	1	2	3	4	5	6	.....	20
1/2 prvního řádku	00	01	02	03	04	05	.....	13
1/2 druhého řádku	40	41	42	43	44	45	.....	53
2/2 prvního řádku	14	15	16	17	18	19	.....	27
2/2 druhého řádku	54	55	56	57	58	59	.....	67

Tab. 5.1: Adresy a rozdělení řádků u LCD displeje

5.5.5 Realizace



Obr. 5.4: Vývojový diagram programu pro ovládání LCD displeje

5.6 Ovládání dotykového displeje OP7200

5.6.1 Úvod

Stejně jako u modul BL2600 je k OP7200 dodáváno mnoho knihoven, které programátorovi zjednodušují práci. V tomto programu budou nejvíce používány funkce pro vykreslování na displej, čtení polohy dotyku na displeji a vyhodnocování stisknutých kláves.

Modul OP7200 obsahuje dotykový grafický displej. Dotyková vrsta je složena z plošek tvořící matici o velikosti 4096 x 4096 plošek. Jak dotyková vsrtva tak i LCD displej pod ní mají počátek souřadnic umístěn v levém horním rohu. To znamená, že pokud se dotkneme v levém horním rohu, funkce vyhodnocující dotyk nám vrátí hodnotu souřadnic x = 0 a y = 0. A také pokud budeme vykreslovat bod do souřadnic x = 0 a y = 0, bod se objeví v levém horním rohu.

Klávesnice na modulu OP7200 je řešena jako maticová, její uspořádání je na Obr. 2.6.

5.6.2 Zadání

1. Vytvořte program, který bude vykreslovat na displeji OP7200 geometrické útvary. Jednotlivé tvary se budou vykreslovat v místě dotyku prstu na displeji.
2. Dále vytvořte jednoduche menu, umožňující volbu velikosti, tvaru a způsobu vykreslení objektu. Menu se ovládá prostřednictvím horní řady tlačítek umístěných pod displejem OP7200.

5.6.3 Přehled funkcí

```
void brdInit (void);
```

Tato funkce se musí volat na začátku programu, inicializuje modul OP7200.

```
void keyInit (void);
```

Bez této funkce by nebylo možné číst jednotlivé stisky na maticové klávesnici.

```
void glInit (void);
```

Tato funkce inicializuje grafický LCD dispelej. Bez zavolání této funkce by nebylo možné vykreslovat na displej.

```
void glBackLight (int onOff);
```

Zapíná nebo vypíná podsvětlení displeje. 1 = zapnuto, 0 = vypnuto

```
void glSetContrast (int contrast);
```

Nastavuje kontrast displeje (contrast = 0 až 31(0 = nejmenší, 31 = největší)).

```
void TsScanState (void);
```

Tato funkce zpracuje informace z dotykové vrstvy a uloží je do paměti. Hodnota může být vyčtena funkcí `TsXYBuffer()`.

```
long TsXYBuffer (void);
```

Vrací 32 bitovou hodnotu, kde vyšších osm bitů reprezentuje souřadnici x a nižších osm bitů reprezentuje souřadnici y.

```
void glSetBrushType (int type);
```

Nastavuje způsob vykreslování na diplej. Parameter `type` může nabívat hodnot `PIXBLACK, PIXWHITE, PIXXOR`.

```
void glFillScreen (int pattern);
```

Vyplní celý displej barvou určenou parametrem `pattern`. Pro `pattern = 0x00` se plocha vyplní bílou barvou, pro `pattern = 0xFF` se plocha vyplní černou barvou.

```
void glPlotline (int x0,int y0, int x, int y);
```

Vykreslí úsečku z bodu (x0,y0) do bodu (x,y).

```
void glBlock (int x, int y, int bmWidth, int bmHeight);
```

Vykeslí obdelník, který má levý horní roh v bodu (x,y) a má rozměry bmWidth,bmHeight.

```
void glPlotPolygon (int n,int x1, int y1,int x2, int y2, ..... );
```

Vykreslí polygon o n částech s jednotlivými body (x1,y1),(x2,y2)...

```
void glPlotCyrcle (int x,int y, int radius);
```

Vykreslí kruh se středem v bodě (x,y) o poloměru radius.

```
void glPrintf (int x,int y, fontInfo *pInfo, char *fmt);
```

Vypíše řetězec `fmt` na souřadnice (x,y) písmem specifikovaným `pInfo`.

```
void keyProcess (void);
```

Zaznamená stav maticové klávesnice.

```
char keyGet (void);
```

Vrací hodnotu stisknuté klávesy na klávesnici a nebo 0 pokud nebylo stisknuto žádné tlačítko.

5.6.4 Řešení

V programu jsme si vytvořili proměnou `int klavesy[] = {1,1,1,1};`, která bude informovat kolikrát byla jaká klávesa stisknuta a tudíž jaké parametry mají být použity pro vykreslování a jaké hodnoty se mají vepsat do menu.

Menu je tvořeno čtyřmi nápisy umístěnými na spodním okraji displeje nad tlačítky, kterými se mění vlastnosti vykreslování. Prvním tlačítkem volíme jaký tvar se bude vykreslovat. Máme na výběr z bodu, úsečky, čtverce, kruhu a trojuhelníku. Druhým tlačítkem se volí velikost jednotlivých objektů. Třetím tlačítkem volíme způsob vykreslování(černě, bíle, XOR) a čtvrtým tlačítkem smažeme celou plochu. Vždy po stisknutí tlačítka se překreslí menu a na pozici stisknutého tlačítka se napíše další volba. Po opětovném stisku tlačítka se opět přepíše menu a objeví se další možnost.

Program je rozdělen pomocí funkce `costate` na dvě samostatné úlohy, které si mezi sebou předávají parametry přes vytvořené pole `klavesy`. První úloha se stará o snímání dotyku na displeji a o vykreslování objektů s parametry získanými v poli `klavesy`, druhá úloha obstarává obsluhu maticové klávesnice a podle stisknutých tlačítek vyplňuje pole `klavesy`.

Obsluha klavesnice

Funkcí `keyProcess()`získáme informace o stavu klávesnice a pokud bude splněna následující podmínka

```
waitfor (( key = keyGet()) > 0 );
```

začne se vykonávat druhá úloha. Podle hodnoty proměné `key` se inkrementuje příslušný prvek pole `klavesy` a překreslí se menu už se změněnými popisy. Jednotlivé popisy se vykreslují funkcí `glPrintf(10,220,&fi10x12,"KRUH");`. Pokud by po inkrementaci byla hodnota prvku z pole `klavesy` vyšší než je počet možností je prvek vynulován a tím se začíná opět od první nabídky v menu.

Vývojový diagram je zobrazen na Obr. 5.5.

Obsluha dotykového displeje

Funkcí `TsScanState();` se uloží do paměti stav dotykového displeje. Poté je tento stav vyčten použitím funkce `TsXYBuffer();` a získaná hodnota je použita v následující podmínce.

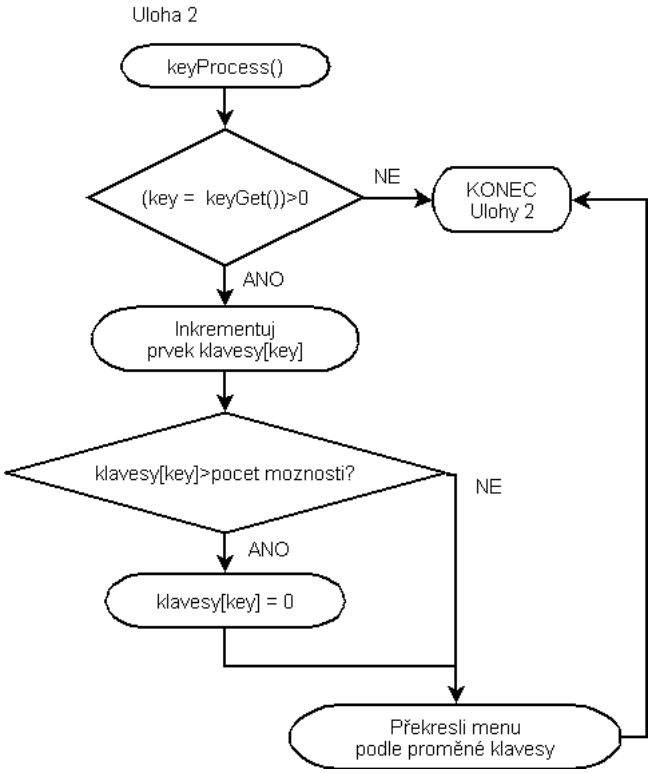
```
waitfor((TouchscreenXY = TsXYBuffer()) != -1 &&

&& !(TouchscreenXY & BTNRELEASE));
```

pokud hodnota přečtená funkcí `TsXYBuffer()` bude různá od -1 a `BTNRELEASE` bude různý od nuly začne se provádět první úloha. `BTNRELEASE` je různé od nuly pokud se nasnímané souřadnice už nemění, to znamená, že zadaný bod je konečný.

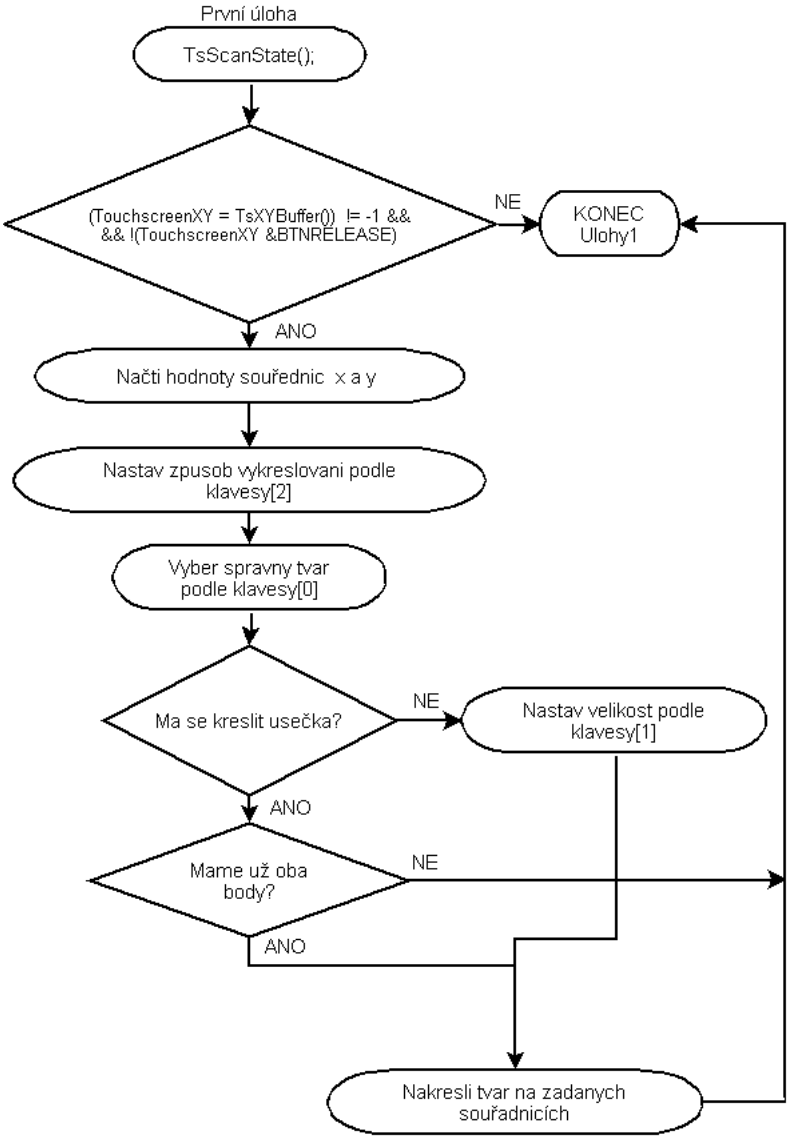
Poté podle hodnot v proměnné `klavesy` jsou vykresleny příslušné objekty se zadanými vlastnostmi. Dále je ošetřen případ úsečky, kdy je potřeba pro její vykreslení mít k dispozici dva body. Při prvním průchodu se uloží načtené souřadnice a podruhé se vykreslí úsečka.

Vývojový diagram je zobrazen na Obr. 5.6.

5.6.5 Realizace



Obr. 5.5: Vývojový diagram druhé úlohy pro ovládání dotykového displeje OP7200



Obr. 5.6: Vývojový diagram první úlohy pro ovládání dotykového displeje OP7200

5.7 Komunikace po sériové lince RS232

5.7.1 Úvod

Modul BL2600 obsahuje tři sériové porty, které lze použít pro komunikaci s okolím. V Tab. 2.2 jsou uvedeny jednotlivé možnosti nastavení, pro každý sériový port. Přímo na modulu BL2600 je osazen budič sériového rozhraní, což znamená, že pro připojení k PC není potřeba používat převodník úrovní ze signálu TTL na úrovně RS232. Maximánlní vzdálenost, na kterou lze komunikovat s nejvyšší přenosou rychlostí je 15 m.

5.7.2 Zadání

1. Napište program, který bude monitorovat, hodnoty digitálních vstupů modulu LED diod a přes sériovou linku je bude posílat informace do PC, kde se příchozí data budou číst prostrednictvim programu realterm. Příchozí data budou zobrazena v binárním tvaru.

5.7.3 Přehled funkcí

U následujících funkcí se místo písmena X dosazuje písmeno používané sériové linky. Může tedy být nahrazeno písmeny A až F. Jelikož u modulu BL2600 jsou vyvedeny jen některé linky můžeme místo X dosazovat pouze písmena C,E a F.

```
#define XINBUFSIZE  15
```

Tímto makrem určíme velikost buffru pro příchozí zprávy.

```
#define XOUTBUFSIZE 15
```

Tímto makrem určíme velikost buffru pro odchozí zprávy. U obou těchto buffru se velikost může nastavovat jako mocniny dvou - 1. To znamená, že mohou nabývat hodnot 15,31,63 a 127.

```
serMode(0);
```

Volíme funkci jednotlivých seriových linek. Parametr volime podle požadované kombinaci. Všechny kombinace jsou uvedeny v Tab. 2.2.

```
int serXopen( long baud );
```

Otevře sériovou linku označenou X.

```
void serXdatabits ( state );
```

Nastavuje formát přenášených dat. Ten může být sedmibitový(kdy do parametru dosadíme `PARAM_7BIT`) a nebo osmibitový (parametr = `PARAM_8BIT`).

```
void serXrdFlush( void );
```

Vyprázdní příchozí buffer.

```
void serXwrFlush( void );
```

Vyprázdní odchozí buffer.

```
int serXputc( char c );
```

Pošle přes sériovou linku znak.

```
int serXputs( char * s );
```

Odešle přes sériovou linku celý řetězec znaků, na který odkazuje parametr.

```
void serXclose( void );
```

Uzavře sériovou komunikaci.

```
int serXgetc( void );
```

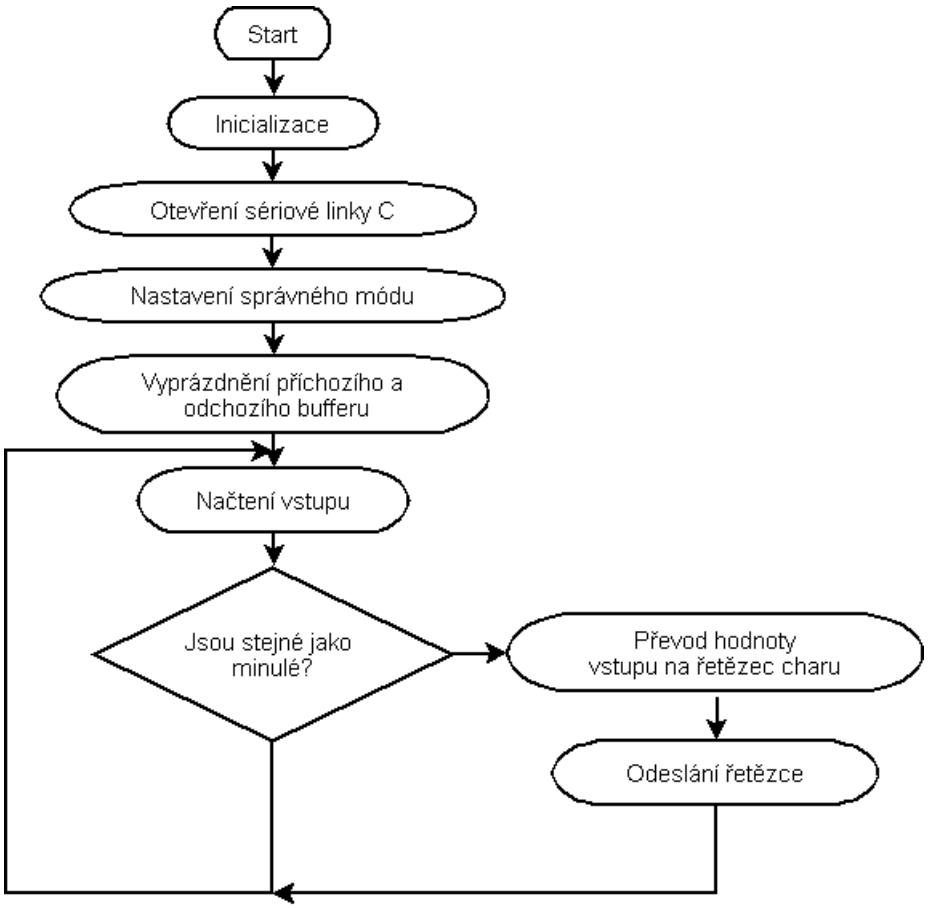
Načte přijatý znak z příchozího bufferu.

5.7.4 Řešení

Tento program je velice jednoduchý. Budeme s počítačem komunikovat přes sériovou linku C. Nejdříve inicializujeme modul BL2600. Poté funkcí `serCopen(9600L);` otevřeme sériovou komunikaci na lince C. Nastavíme funkci seriové linky C jako tří vodičovou sériovou linku bez řídících signálů. Toho dosáhneme zavoláním funkce serMod(0). Jak vidime z Tab. 2.2 tak i zbylé dvě sériové linky jsou nastaveny v tomto režimu. Dále vyprázdníme příchozí i odchozí buffer funkcema `serCwrFlush();` a `serCrdFlush();`.

V hlavní smyčce vždy přečteme hodnotu vstupů, porovnáme je s minulým stavem a pokud jsou od sebe různé odešleme je přes sériovou linku do PC. Zde přijaté hodnoty můžeme zobrazovat pomocí programu RealTerm nebo Hyperterminálu. Pro správné zobrazování je nutné, aby byly tyto programy správně nastaveny. Na Obr. 5.7 je zobrazen vývojový graf programu.

5.7.5 Realizace



Obr. 5.7: Vývojový diagram programu pro komunikaci po sériové lince

5.8 Jednoduchý webový http server

5.8.1 Úvod

Pro jednoduché a přehledné ovládání zařízení je možné vytvořit jednoduchý http server, který potřebné údaje zobrazuje v okně internetového prohlížeče. Návrh tohoto serveru se skládá ze dvou částí. Nejprve vytvoříme jednoduchou internetovou stránku ve formátu HTML. Tato stránka se poté nahraje při kompilaci hlavního programu do paměti modulu a z té se poté odesílá podle potřeb uživatelům, kteří se k serveru připojují přes ethernetové připojení. Podrobný popis fukncí webového serveru je popsán v kapitole 3.4.

5.8.2 Zadání

1. Napište program, který vytvoří webový server. IP adresa se bude přidělovat dynamicky, proto ji po načtení, zobrazte v stdio okně aplikace Dynamic C. Na serveru budou nahrány dvě internetové stránky, mezi kterými půjde přepínat. Umístěte na ně alespoň jeden obrázek.

5.8.3 Přehled funkcí

```
#define HTTP_MAXSERVERS 2
```

Omezuje maximální počet http serverů poslouchajících na portu 80.

```
MAX_TCP_SOCKET_BUFFERS 2
```

Počet bufferu pro navázaná spojení

```
#memmap xmem
```

Nastaví ukládání proměnných do externí paměti

```
#ximport "pages/index.html"    index_html
```

Při kompilaci nahraje soubor index.html do externí paměti a vytvoří na něj ukazatel index_htlm.

```
SSPEC_RESOURCETABLE_START
        SSPEC_RESOURCE_XMEMFILE("/index.html", index_html),
        SSPEC_RESOURCE_XMEMFILE("/index2.html", index2_html),
        SSPEC_RESOURCE_XMEMFILE("/VUT_FEKT.jpg", VUT_jpg),
        SSPEC_RESOURCE_XMEMFILE("/elektro.jpg", elektro_jpg)
SSPEC_RESOURCETABLE_END
```

Tento zápis říká, že pokud přijde požadavek například na soubor `/index.html` mají se začít posílat data z paměti, na kterou odkazuje `index_html`

```
SSPEC_MIMETABLE_START
        SSPEC_MIME(".html", "text/html"),
        SSPEC_MIME(".jpg", "image/jpg")
SSPEC_MIMETABLE_END
```

Tato tabulka určuje internetovému prohlížeči, jaký obsah zobrazuje uživatelovi.

`sock_init();`

Inicializace ethernetového připojení.

`http_init();`

Inicializace HTTP serveru.

`inet_ntoa(buffer, gethostid())`

Tento zápis získá IP adresu zařízení a převede ji do dekadické podoby, kde jednotlivé části jsou odděleny tečkami.

`http_handler();`

Volání této funkce se aktualizuje HTTP server

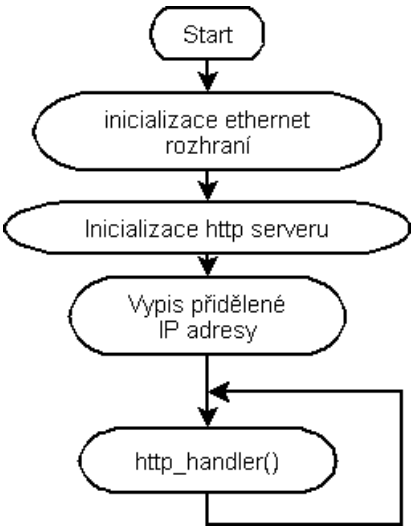
5.8.4 Řešení

Nejprve je potřeba vytvořit dvě jednoduché HTML stránky, které na sebe navzájem ukazují. Stránky jsou formátované standardními HTLM tagy.

V programu pro mikrokontroler nejdříve direktivou `#define TCPCONFIG 3` nastavíme, že IP adresa se bude přidělovat dynamicky. Poté omezíme maximální možný počet připojení zápisem
`#define HTTP_MAXSERVERS 2` a `#define MAX_TCP_SOCKET_BUFFERS 2`. Abychom mohli nahrát vytvořené stránky do externí paměti musíme zapsat direktivu `#memmap xmem`. Tím říkáme, že nahrávané soubory se uloží do externí paměti. Dále načteme knihovny `#use "dcrtcp.lib"` a `#use "http.lib"`. Následně už můžeme nahrát jednotlivé soubory pomocí zápisu
`#ximport "pages/index.html"index_html`. Takto nahrajeme veškeré soubory, které se v HTML stránkách vyskytují. Také musíme specfikova MIME tabulku a tabulku zdrojů. Zápis těchto dvou tabulek je uveden v 5.8.3.

V hlavní smyčce programu inicializujeme ethernetové připojení a spustíme HTTP server. Dále vypíšeme přidělenou IP adresu a nakonec v nekončné smyčce voláme funkci `http_handler();`.

5.8.5 Realizace



Obr. 5.8: Vývojový diagram jednoduchého http serveru

5.9 Zobrazení a nastavení RTC přes http rozhraní

5.9.1 Úvod

Jelikož statické internetové stránky neumožňují zobrazovat například aktuálně naměřené hodnoty. Je potřeba do stránek vkládat proměnné, které se po aktualizaci stránky změní. K tomu prostředí Dynamic C obsahuje nástroj nazvaný RabbitWeb. Funguje velice podobně jako skriptovací jazyk PHP. To znamená, že než se klientovi odešle požadovaná stránka, server zpracuje obsažené skripty a na jejich místo dosadí výsledek. Skripty se vkládají mezi standardní HTML značky. Stejně jako u předchozího příkladu je nejdříve potřeba vytvořit ZHTLM stránku, která se bude zobrazovat v prohlížeči. Písmenko Z před slovem HTLM značí, že stránka obsahuje skripty, které dokáže zpracovat RabbitWeb.

Na zobrazené stránce budeme zobrazovat čas poskytnutý hodinami reálného času (RTC), které jsou implementovány přímo na desce modulu. Tyto hodiny jsou zálohovány baterií, takže i po výpadku napájení si uchovávají svoji funkci a samostaně se aktualizují. Hodiny jsou uvnitř uloženy jako počet sekund od 01.01.1980.

Prostředí Dynamic C opět obsahuje rozsáhlé knihovny, které usnadňují práci s těmito zařízeními.

5.9.2 Zadání

- Napište program, který umožní zobrazení času uloženého v RTC internetový prohlížeč.
- Upravte program tak, aby bylo možné zobrazený čas změnit a uložit zpět do RTC. Nastavení ethernetového rozhraní nastavte tak, aby se IP adresa přidělovala dynamicky pomocí DHCP protokolu. Přidělenou IP adresu vypište do stdio okna.

5.9.3 Přehled funkcí

```
#define USE_RABBITWEB 1
```

Povolí používání RabbitWebu

```
struct tm rtc;
```

Struktutra typu tm je navržena pro ukládání získaných dat z obvodu RTC. Její struktura je následující:

```
struct tm {
        char tm_sec; // sekundy 0-59
        char tm_min; // minuty 0-59
        char tm_hour; // hodiny 0-23
        char tm_mday; // dny v mesici1-31
        char tm_mon; // mesice 1-12
        char tm_year; // roky 80-147 (1980-2047)
        char tm_wday; // dny v tydnu 0-6 0==nedele
};
```

Pro převod počtu sekund na jednotlivé prvky výše uvedené struktury existuje několik funkcí uvedených níže.

```
unsigned long mktime( struct tm * timeptr );
```

Funkce vrátí 32 bitové číslo, které reprezentuje počet sekund od 1.1.1980 do datumu, které je uloženo ve struktuře, na kterou ukazuje ukazatel `timeptr`.

```
unsigned int mktm( struct tm * timeptr, unsigned long time );
```

Převede 32 bitovou hodnotu `time` na jednotlivé prvky datumu a ty uloží do struktury `timeptr`.

```
unsigned long read_rtc( void );
```

Funkce vrací aktuální hodnotu paměti RTC.

```
void write_rtc( unsigned long int time );
```

Funkce zapíše `time` do paměti RTC.

```
int tm_wr( struct tm * t );
```

Funkce zapíše strukturu `tm` do paměti RTC.

```
int tm_rd( struct tm * t );
```

Funkce zapíše do struktury `tm` obsah paměti RTC.

```
#web chkboxTime
```

tímto způsobem je určeno, že proměnná za direktivou `#web` bude spolupracovat s blokem RabbitWeb.

```
#web_update foo, bar, baz user_callback
```

Tímto zápisem říkáme, že pokud bude změněna, aspoň jedna proměnná (bar,foo,baz) bude spuštěna funkce `user_callback`,která může podle změn v internetovém prohlížeči přenastavit proměnné uvnitř programu.

5.9.4 Řešení

Nejdříve vytvoříme ZHTML stránku, kterou poté nahrajeme do externí paměti. Jednotlivé ZHTML skripty jsou od zbytku HTML kódu odděleny následovně.

```
<?z statement ?>
```

Aby bylo možné zpracovávat informace z HTML stránky je nejvhodnější použít na stránce formuláře. Příklad vstupního pole formuláře vypadá následovně

```
<input type="text" name="<?z varname($rtc2.tm_hour) ?>" size="2"
        value="<?z printf( "%02d", $rtc2.tm_hour) ?>">
```

kde makro `printf("%02d", $rtc2.tm_hour)`má stejnou syntaxi a formátování jako standardní funkce v C. Funkce `varname(&proměnná)` správně překládá názvy proměnných.

Po zpracování tohoto kódu se v internetovém prohlížeči objeví následující kód.

```
<input type="text" name="rtc2.tm_hour" size="2"
        value="18">:
```

Jak je vidět po zpracování skriptu se místo skriptu

```
<?z printf( "%02d", $rtc2.tm_hour) ?>"
```

dosadí aktuální hodnota proměnné

```
$rtc2.tm_hour} = 18.
```

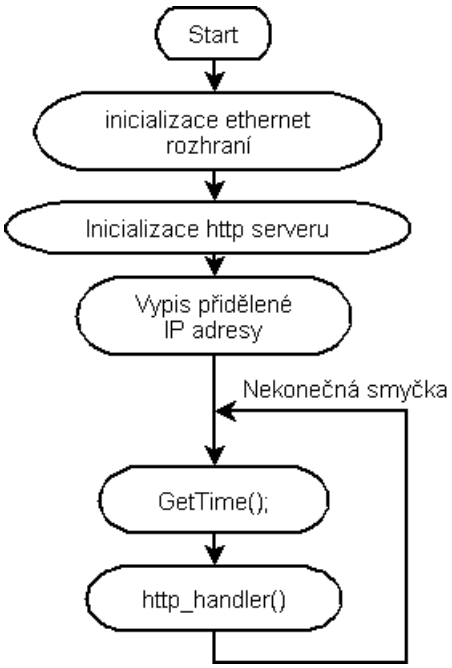
Více podrobností o tomto skriptovacím jazyku je napsáno v [15].

Samotný program je velice podobný předchozímu příkladu. Nejprve direktivou `#define TCPCONFIG 3` nastavíme, že IP adresa se bude přidělovat dynamicky. Také direktivou `#define USE_RABBITWEB 1` povolíme používání RabbitWebu. Poté stejně jako minule povolíme požívání externí paměti a použijeme knihovny `#use "dcrtcp.lib"` a `#use "http.lib"`. Nahrajeme vytvořenou stránku příkazem `#ximport "cas.zhtml"cas_zhtml`. Vyplníme tabulku MIME typů a tabulku zdrojů. Dále deklarujeme proměnné, které budou spolupracovat se skriptovacím jazykem. To se provádí následujícím zápisem `#web rtc2.tm_year (($rtc2.tm_year >= 1980) && ($rtc2.tm_year <= 2047))`, kde v závorkách je možné uvést omezení zadávaných dat.

Také musíme direktivou `#web_update rtc2.tm_year, rtc2.tm_mon, rtc2.tm_mday, rtc2.tm_hour, rtc2.tm_min, rtc2.tm_sec putTime` určit, že po změně některé proměnné se zavolá funkce pro uložení nastaveného času do paměti obvodu RTC.

V hlavním programu inicializujeme ethernetové připojení a spustíme HTTP server. Dále vypíšeme přidělenou IP adresu a nakonec v nekonečné smyčce voláme funkci `http_handler();` a funkci pro čtení dat z obvodu RTC.

5.9.5 Realizace



Obr. 5.9: Vývojový diagram http serveru čtením času z RTC

6 ZÁVĚR

Tato diplomová práce se zabývá návrhem laboratorních úloh s mikrokontorlerem Rabbit.

Aby bylo možné realizovat navržené úlohy, byly vytvořeny externí periferie nutné pro názorné vyzkoušení zadaných úkolů. Jednotlivé periferie se připojují k mirkokontroleru přes navrženou rozvodnou desku, která má na sobě vyvedeny všechny potřebné konektory. Bylo navrženo celkem šest obvodů, které jsou v úlohách používány.

Úlohy jsou koncipovány tak, aby studentovi přiblížily problematiku programování mikrokontrolerů a práci s jeho periferiemi. Každá úloha má svoje zadání, úvod, návod pro vypracování a vývojový diagram programu. V návodu jsou popsány důležité kroky, které pomáhají studentům zdárně splnit zadání. Úlohy jsou vypracovány v prostředí Dynamic C. Zdrojové kódy všech úloh jsou uloženy na přiloženém CD včetně veškerých potřebných podkladů pro výrobu plošných spojů navržených přípravků.

První úloha se zabývá ovládáním digitálních vstupů a výstupů mikrokontroleru. Ovládání vstupně výstupních portů mikrokontroleru patří k základním dovednostem, bez kterých nelze pokračovat k dalším úlohám. Druhá úloha se zabývá čtením stisků tlačítek maticové klávesnice. V třetí úloze se řeší ovládání krokového motoru pomocí výkonových výstupů modulu BL2600. Studenti si vyzkouší různé druhy řízení unipolárných krokových motorů. Čtvrtá úloha se zabývá generováním signálu PWM, kterým je poté ovládána rychlost otáčení stejnosměrného motoru. Pátá úloha se zabývá ovládáním alfanumerického displeje. Displej disponuje řadičem kompatiblním s hojně požívaným typem HD44780. Šestá úloha je navržena pro operátorský panel OP7200. Studenti se v ní seznámí s ovládáním dotykového displeje a klávesnice, které jsou umístěny na tomto modulu. Sedmá úloha studenty seznámí s komunikací po sériové lince. V osmé úloze je navržen jednoduchý http server, který klientovi posílá dvě statické HTML stránky, mezi kterými lze přepínat. Poslední devátá úloha opět obsahuje http server, který dynamicky generuje HTML stránky a umožňuje na nich zobrazovat načtený čas hodin reálného času. Prostřednictvým formuláře se tyto hodiny mohou libovolne nastavovat.

Jednotlivé navržené úlohy na sebe navazují a jejich obtížnost se postupně zvyšuje tak, jak student získává zkušenosti. Ovšem jejich konečné sestavení a vytvoření časového harmonogramu jednotlivých úloh záleží na zkušenostech vyučujícího. Úlohy musejí být sestaveny tak, aby probíraná látka korespondovala s přednáškami předmětu.

Kombinací navržených úloh vzniknou nové, u kterých se může libovolně měnit jejich obtížnost, podle toho jak studenti získávají zkušenosti s programováním.

LITERATURA

[1] Rabbit Semiconductor; *BL2600 Single-Board Computer User's Manual* [online]. 2008, poslední aktualizace 22. 04. 2010 [cit. 10. 11. 2010]. Dostupné z URL: <http://www.rabbit.com/hottag/index.php?ht=/documentation/docs/manuals/BL2600/BL2600UM.pdf>.

[2] Rabbit Semiconductor; *Rabbit 3000 Microprocessor User's Manual* [online]. 2008, poslední aktualizace 26. 08. 2010 [cit. 10. 11. 2010]. Dostupné z URL: <http://www.rabbit.com/hottag/index.php?ht=/documentation/docs/manuals/Rabbit3000/UsersManual/R3000UM.pdf>.

[3] Rabbit Semiconductor; *BL2600 Single-Board Computer Schematic* [online]. 2008, poslední aktualizace 28. 05. 2009 [cit. 10. 11. 2010]. Dostupné z URL: <http://www.rabbit.com/documentation/schemat/090-0195.pdf>.

[4] Rabbit Semiconductor; *OP7200 eDisplay User's Manual* [online]. 2008, poslední aktualizace 12. 06. 2009 [cit. 10. 11. 2010]. Dostupné z URL: <http://www.rabbit.com/hottag/index.php?ht=/documentation/docs/manuals/OP7200/OP7200UM.pdf>.

[5] Rabbit Semiconductor; *OP7200 Schematic* [online]. 2008, poslední aktualizace 31. 03. 2009 [cit. 10. 11. 2010]. Dostupné z URL: <http://www.rabbit.com/documentation/schemat/090-0138.pdf>.

[6] Samsung Electronic; *S6A0069 40SEG/16 COM DRIVER & CONTROLLER FOR DOT MATRIX LCD* [online]. 2000, [cit. 10. 11. 2010]. Dostupné z URL: <http://www.allshore.com/pdf/Samsung_KS0066_(s6a0069).pdf>.

[7] Atmel ; *Two-wire Serial EEPROM AT24C04B* [online]. 5/2007, poslední aktualizace 31. 07. 2008 [cit. 10. 11. 2010]. Dostupné z URL: <http://www.atmel.com/dyn/resources/prod_documents/doc0180.pdf>.

[8] Philips semiconductors ; *Remote 8-bit I/O expander for I2C-bus PCF8574* [online]. 2007, poslední aktualizace 14. 05. 2007 [cit. 10. 11. 2010]. Dostupné z URL: <http://www.datasheetcatalog.org/datasheet/philips/PCF8574_4.pdf>.

[9] Philips semiconductors ; *256 ´ 8-bit static low-voltage RAM with I2C-bus interface PCF8570* [online]. 2007, poslední aktualizace 06. 01. 2009 [cit. 10. 11. 2010]. Dostupné z URL: <http://www.datasheetcatalog.org/datasheet/philips/PCF8570P.pdf>.

[10] Philips semiconductors ; *Clock/calendar with 240 x 8-bit RAM PCF8583* [online]. 2007, poslední aktualizace 15. 06. 2007 [cit. 10. 11. 2010]. Dostupné z URL: <http://www.datasheetcatalog.org/datasheet/philips/PCF8583_5.pdf>.

[11] MAXIM - Dallas Semiconductor ; *2-Wire Interfaced Matrix LED Display Driver MAX6953* [online]. 2004, poslední aktualizace 09. 03. 2004 [cit. 10. 11. 2010]. Dostupné z URL: <http://www.datasheetcatalog.org/datasheet/maxim/MAX6953.pdf>.

[12] L. PEARSON, Randy; *Embedded Systems Programming Course* / [online]. 2007,Poslední aktualizace 04.03.2007 [cit. 2011-05-13]. Multi-tasking and Dynamic C. Dostupné z URL: <<http://www.bamafolks.com/randy/students/embedded/dynamicC_mtask.html.>.

[13] Rabbit Semiconductor; *Dynamic C Users Manual* [online]. 2008, poslední aktualizace 26. 08. 2010 [cit. 10. 11. 2010]. Dostupné z URL: <<http://ftp1.digi.com/support/documentation/0190125_j.pdf>.

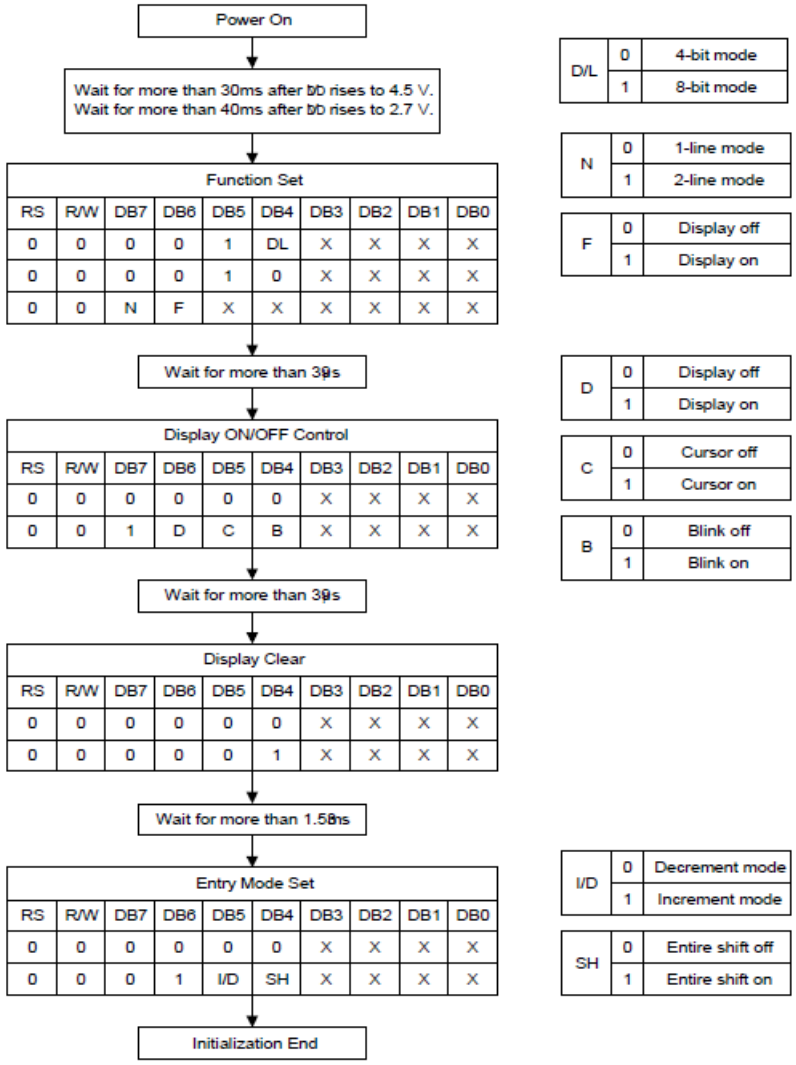
[14] Rabbit Semiconductor; *Dynamic C TCP/IP User's Manual Volume 1* [online]. 2009, poslední aktualizace 20. 11. 2009 [cit. 10. 11. 2010]. Dostupné z URL: <<http://ftp1.digi.com/support/documentation/0190143_h.pdf>.

[15] Rabbit Semiconductor; *Dynamic C TCP/IP User's Manual Volume 2* [online]. 2010, poslední aktualizace 30. 10. 2010 [cit. 10. 11. 2010]. Dostupné z URL: <<http://ftp1.digi.com/support/documentation/019-0144_G.pdf>.

SEZNAM PŘÍLOH

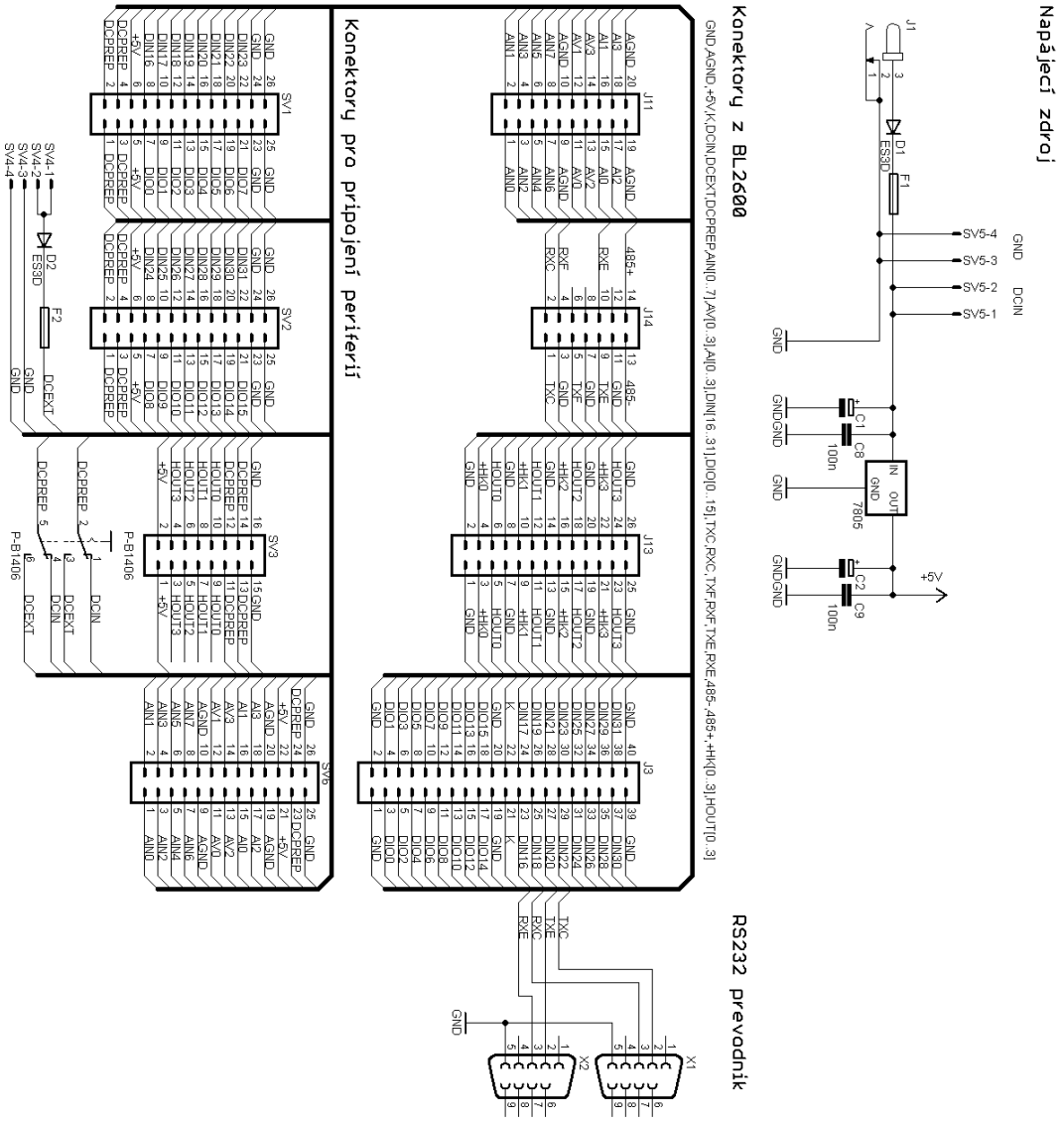
A PŘÍLOHY

A.1 Postup inicializace LCD displeje



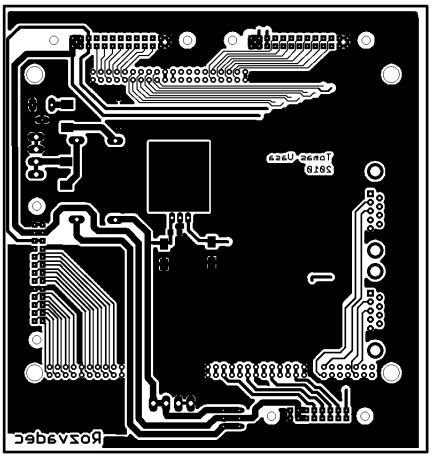
Obr. A.1: Postup inicializace displeje [6]

A.2 Schéma zapojení rozvaděče

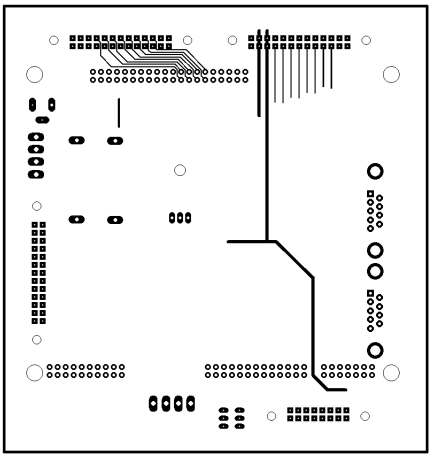


Obr. A.2: Schéma zapojení rozvaděče

A.3 Plošný spoj rozvaděčové desky

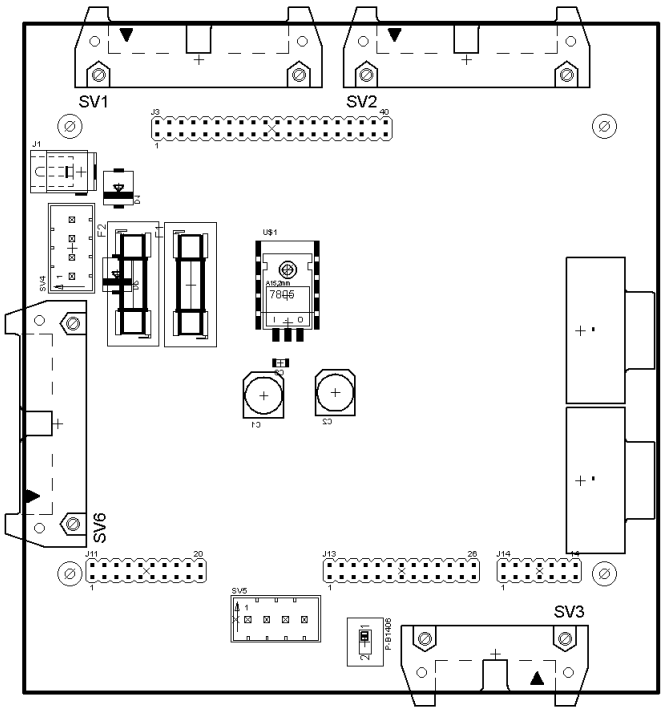


Obr. A.3: Plošný spoj rozvaděčové desky BOTTOM



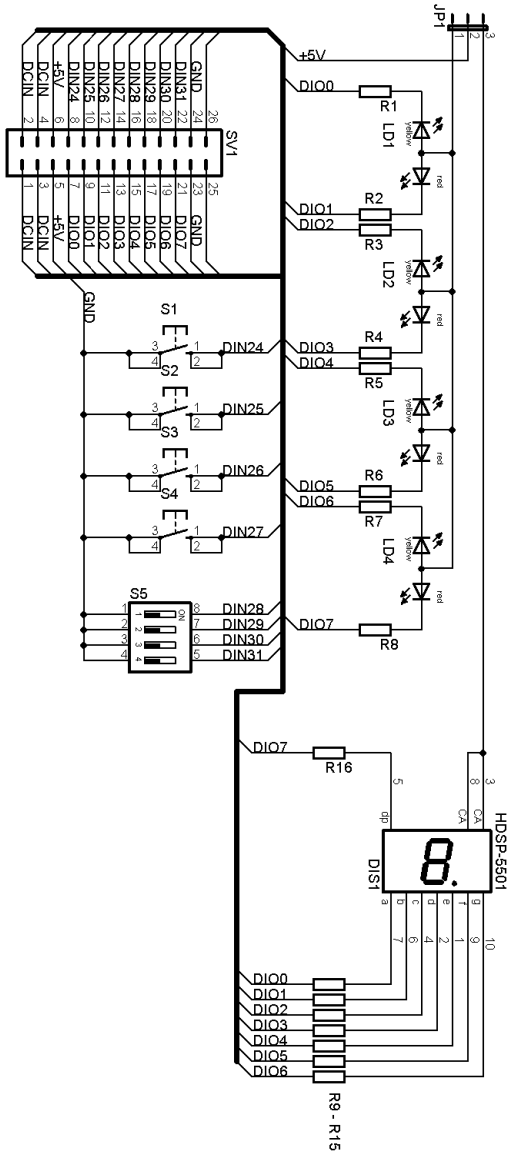
Obr. A.4: Plošný spoj rozvaděčové desky TOP

A.4 Osazovací výkres rozvaděčové desky



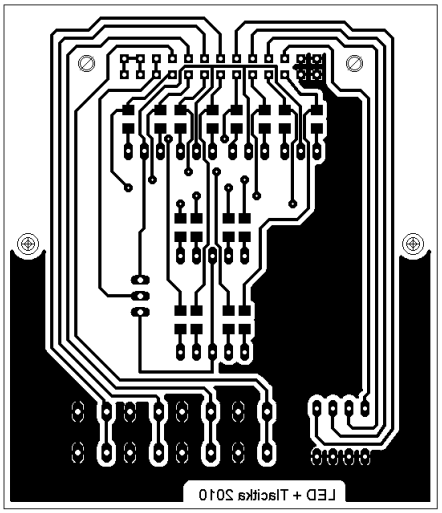
Obr. A.5: Osazovací výkres rozvaděčové desky

A.5 Schéma zapojení LED diod a tlačítek



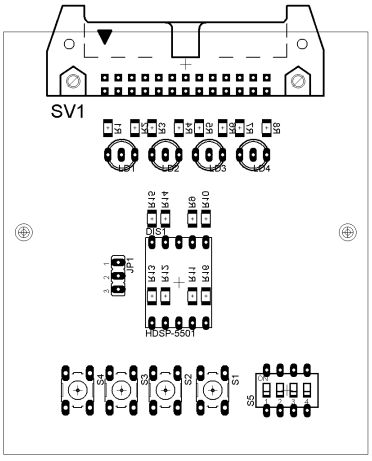
Obr. A.6: Schéma zapojení LED diod a tlačítek

A.6 Deska plošného spoje LED diod a tlačítek



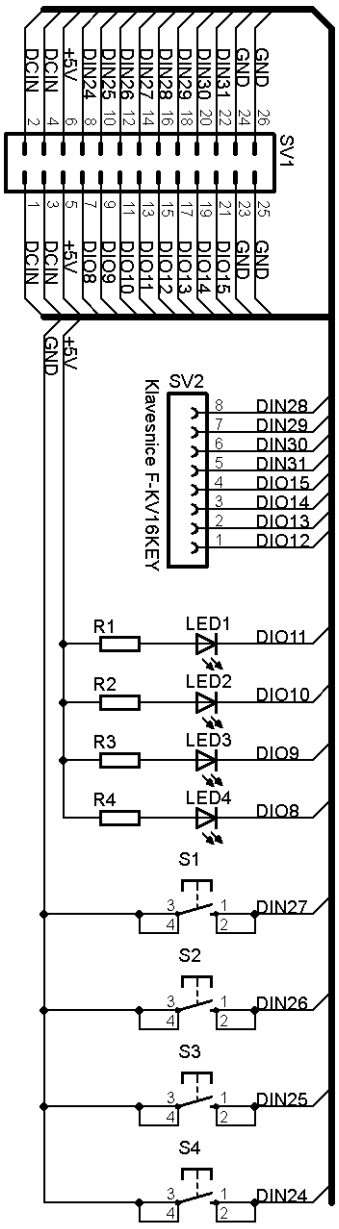
Obr. A.7: Deska plošného spoje LED diod a tlačítek

A.7 Osazovací výkres modulu LED diod a tlačítek



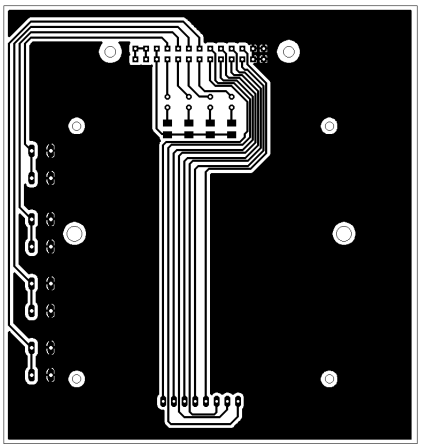
Obr. A.8: Osazovací výkres modulu LED diod a tlačítek

A.8 Schéma zapojení maticové klávesnice



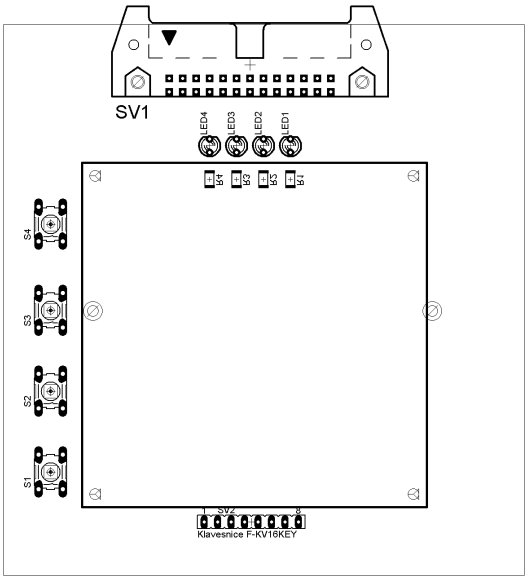
Obr. A.9: Schéma zapojení maticové klávesnice

A.9 Deska plošného spoje maticové klávesnice



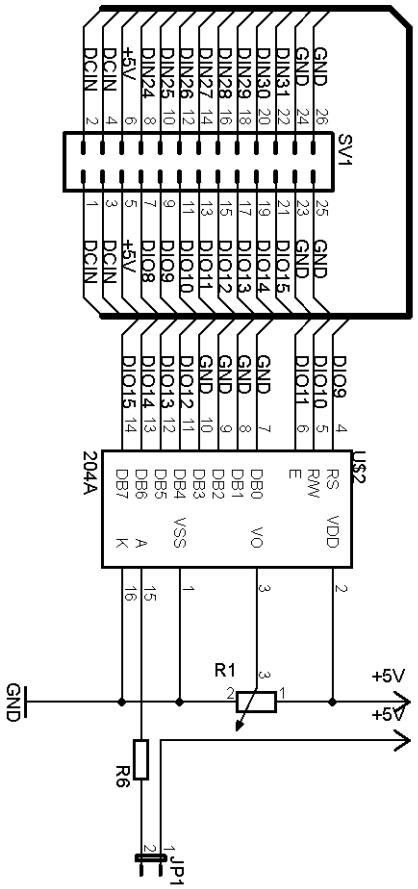
Obr. A.10: Deska plošného spoje maticové klávesnice

A.10 Osazovací výkres modulu maticové klávesnice



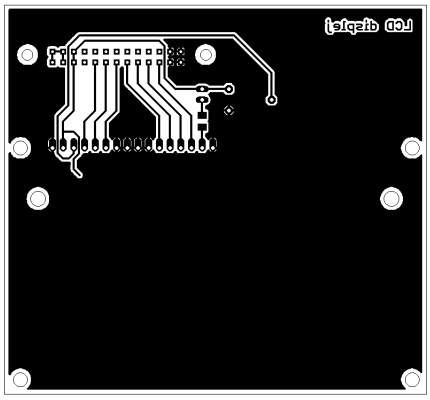
Obr. A.11: Osazovací výkres modulu maticové klávesnice

A.11 Schéma zapojení LCD displeje



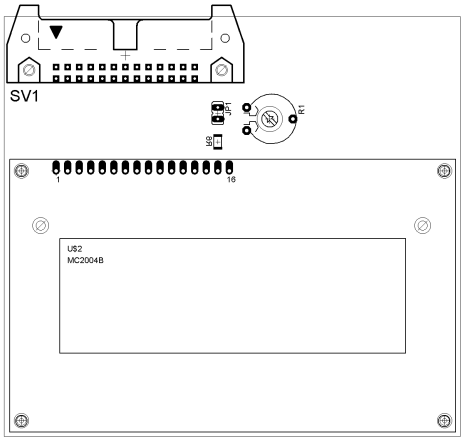
Obr. A.12: Schéma zapojení LCD displeje

A.12 Deska plošného spoje LCD displeje



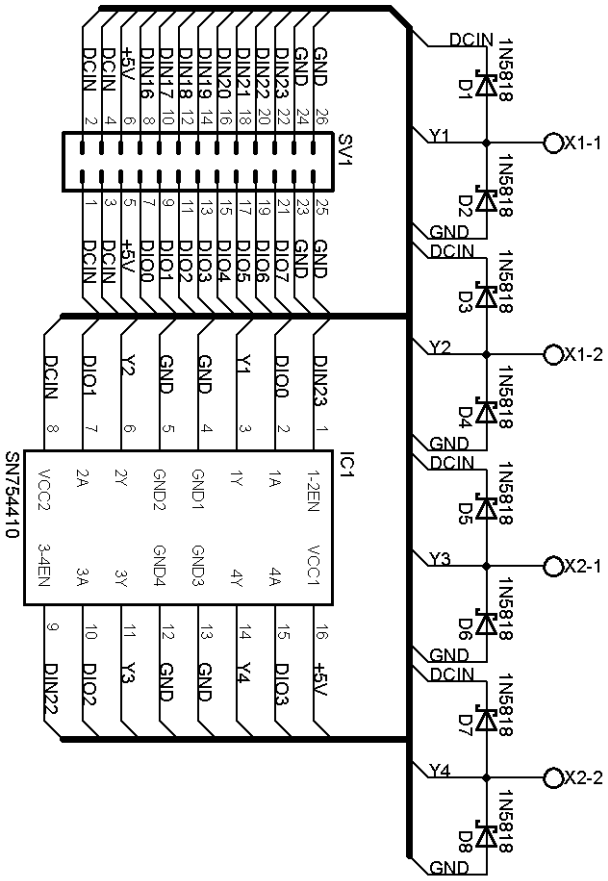
Obr. A.13: Deska plošného spoje LCD displeje

A.13 Osazovací výkres modulu LCD displeje



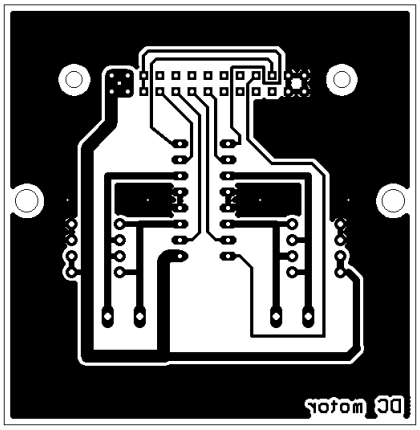
Obr. A.14: Osazovací výkres modulu LCD displeje

A.14 Schéma zapojení DC motoru



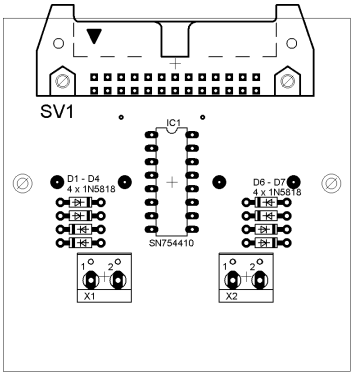
Obr. A.15: Schéma zapojení DC motoru

A.15 Deska plošného spoje DC motoru



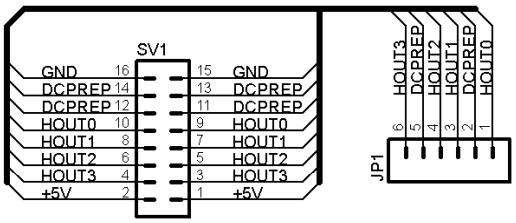
Obr. A.16: Deska plošného spoje DC motoru

A.16 Osazovací výkres modulu DC motoru



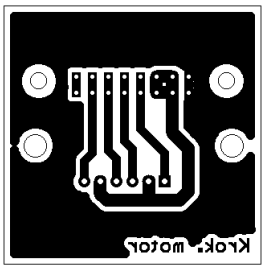
Obr. A.17: Osazovací výkres modulu DC motoru

A.17 Schéma zapojení krokového motoru



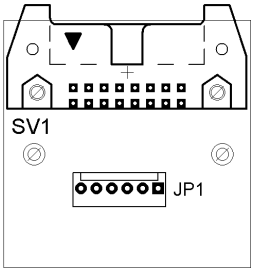
Obr. A.18: Schéma zapojení krokového motoru

A.18 Deska plošného spoje krokového motoru



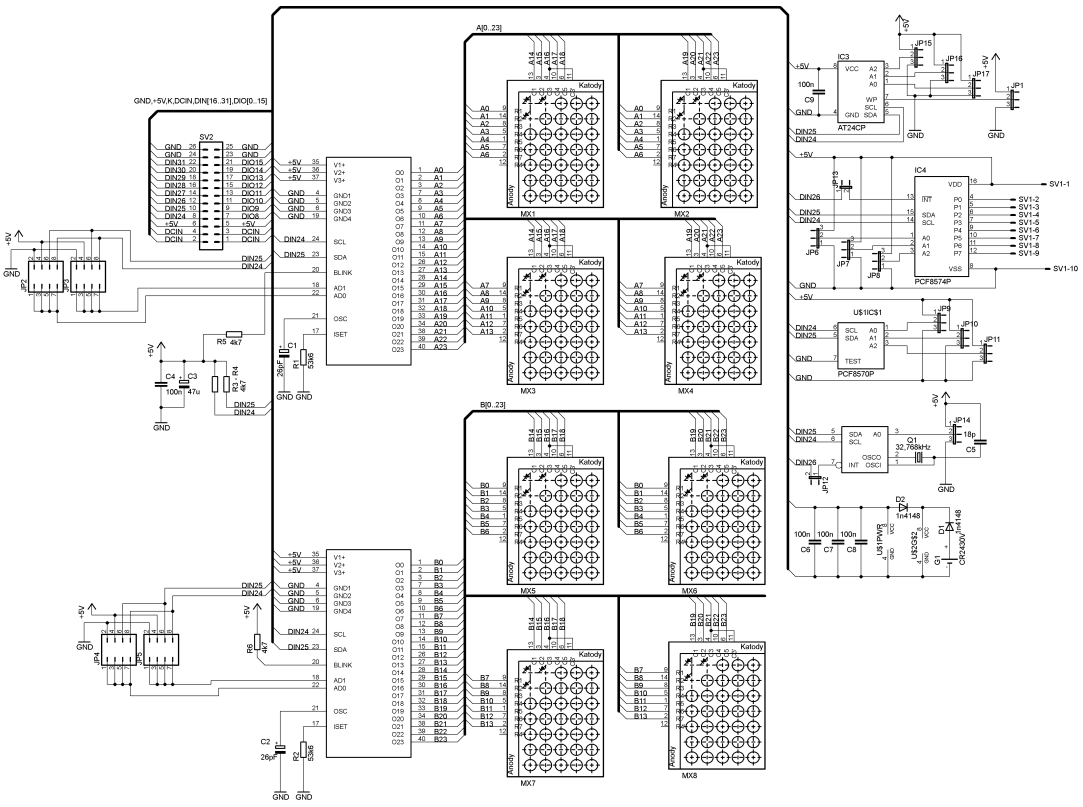
Obr. A.19: Deska plošného spoje krokového motoru

A.19 Osazovací výkres modulu krokového motoru



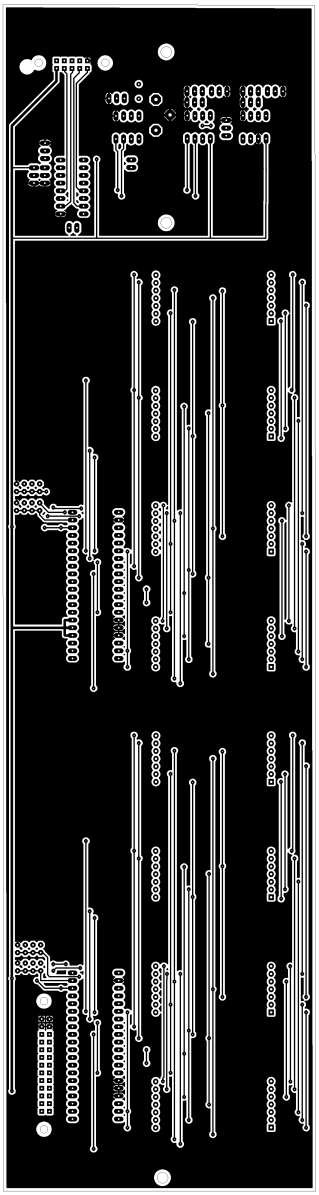
Obr. A.20: Osazovací výkres modulu krokového motoru

A.20 Schéma zapojení I2C modulu

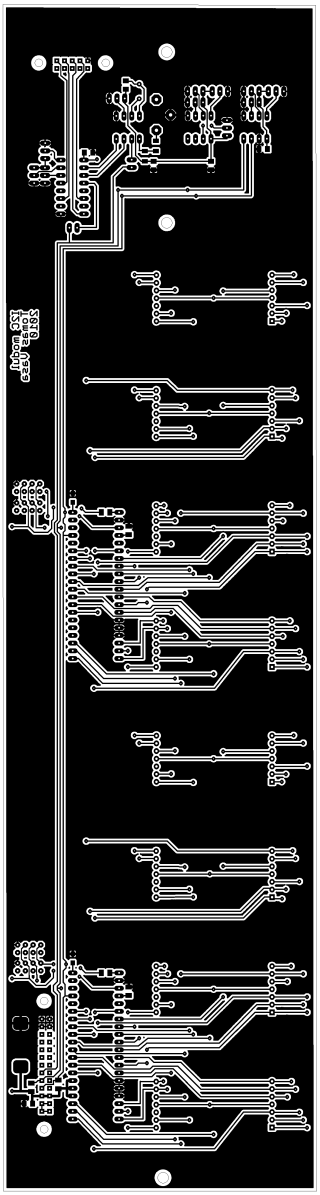


Obr. A.21: Schéma zapojení I2C modulu

A.21 Plošný spoj modulu I2C

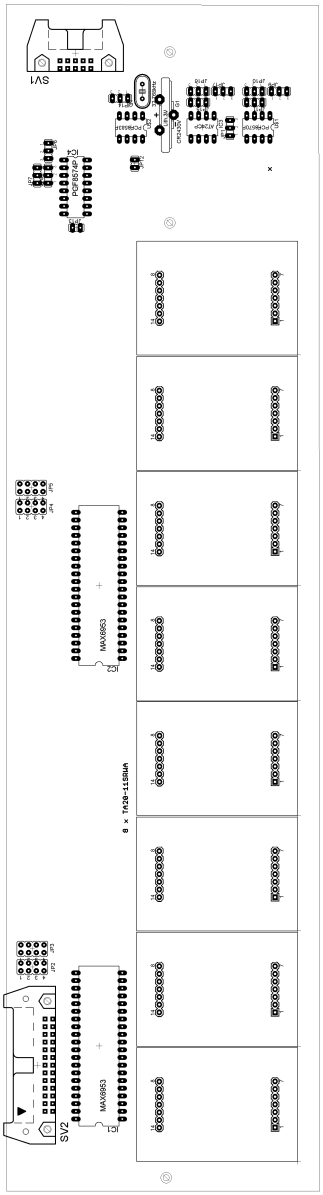


Obr. A.22: Plošný spoj I2C TOP

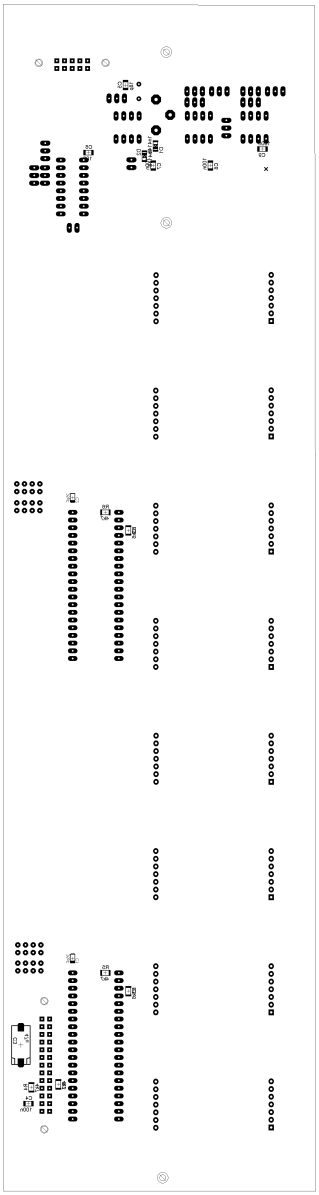


Obr. A.23: Plošný spoj I2C BOTTOM

A.22 Osazovací výkres modulu I2C

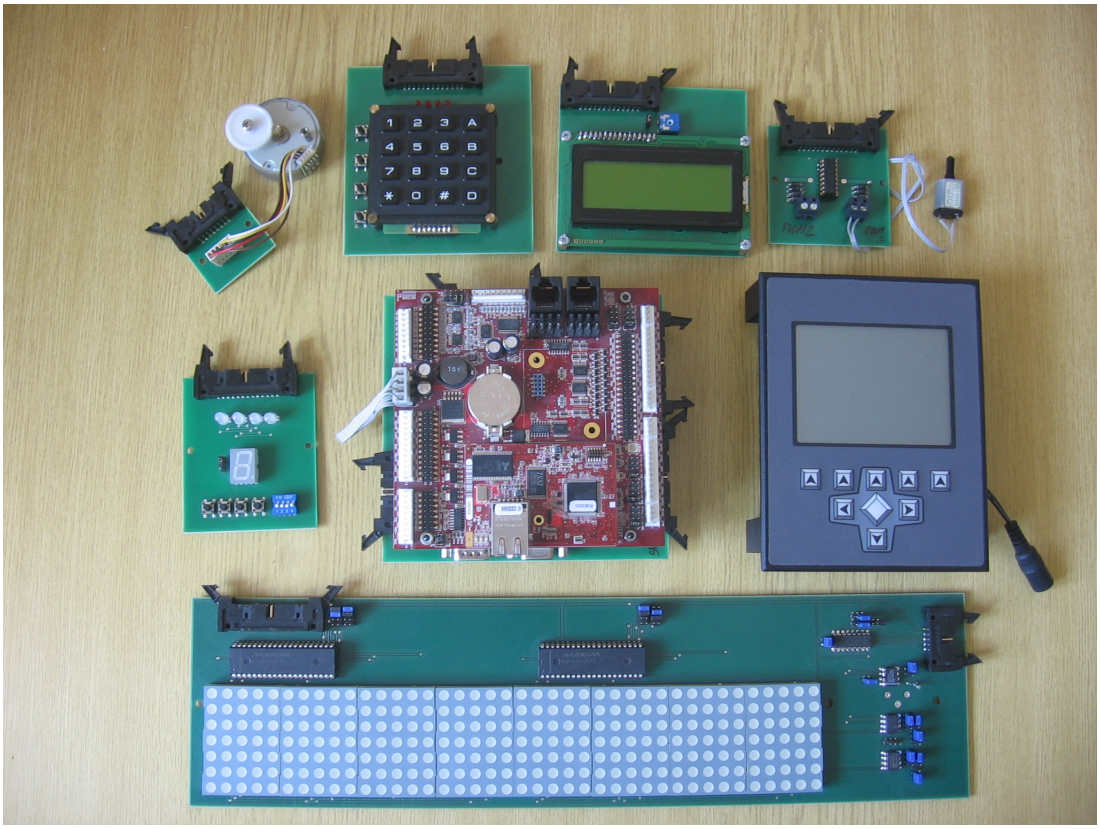


Obr. A.24: Osazovací výkres TOP



Obr. A.25: Osazovací výkres BOTTOM

A.23 Fotografie použitých modulů



Obr. A.26: Fotografie použitých modulů