Fakulta elektrotechnická
Katedra aplikované elektroniky a telekomunikací

Bakalářská práce

Úprava elektroniky točny kolejiště

Autor práce: Matěj Vaněk
Vedoucí práce: Ing. Luděk Elis

Plzeň 2014

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Matěj VANĚK**
Osobní číslo:	**E10B0358P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a telekomunikace**
Název tématu:	**Úprava elektroniky točny kolejiště**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Prostudujte současnou koncepci řídicí elektroniky točny kolejiště a zhodnoťte současné řešení.
2. Navrhněte vhodná vylepšení SW i HW.
3. Aplikujte navržená vylepšení.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah pracovní zprávy: **20 - 30 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

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

Vedoucí bakalářské práce: **Ing. Luděk Elis**
Katedra aplikované elektroniky a telekomunikací

Datum zadání bakalářské práce: **14. října 2013**

Termín odevzdání bakalářské práce: **9. června 2014**

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

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 14. října 2013

Abstrakt

Práce se věnuje popisu řídícího obvodu točny kolejiště, jejímu zhodnocení a návrhem možných vylepšení, a popisu použité točny kolejiště včetně jejího ovládání. Navržená vylepšení jsou aplikována v nové konstrukci řídícího obvodu točny. Dále se věnuje popisu nové konstrukce a vysvětlením aplikovaných vylepšení i změn. Stejným způsobem je zhodnocen i software pro ovládání točny a navržen nový s novou konstrukcí. Navržený software je v samostatné kapitole popsán a vysvětlen.

Klíčová slova

točna, CAN, konstrukce, software, komunikace

Abstract

Vaněk, Matěj. *Modification of railway turntable's electronic .* [*Úprava elektroniky točny kolejiště*]. Pilsen, 2014. Bachelor thesis (in Czech). University of West Bohemia. Faculty of Electrical Engineering. Department of Applied Electronics and Telecommunications. Supervisor: Ing. Luděk Elis

---

This thesis is devoted to the description of control electronic of railway turntable, its evaluation and suggestions for possible improvements. Also it is devoted to the description of the railway turntable including its control. The proposed improvements are applied in the new design of control electronic. It also deals with the description of the new design and explanation of applied improvements and changes. In the same way, software for control railway turntable is evaluated. New software designed and its description and explanation is in a separate chapter.

Keywords

railway turntable, CAN, design, software, communication

Prohlášení

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

Prohlašuji, že jsem svou závěrečnou práci vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 270 trestního zákona č. 40/2009 Sb.

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

V Plzni dne 8. června 2014

Matěj Vaněk

. . . . . .

Podpis

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů a zkratek

ACK	Acknowledge. Potvrzení správného příjmu.
A/D	Analog/Digital. Analogový/Digitální převodník.
BDM	Background Debug Mode. Rozhraní programátoru pro debugování.
$BKGD/MS$	Background/Mode Select.
BRP	Baud Rate Prescaler.
CAN	Controller Area Network.
CAN_H	Označení vodiče CAN sběrnice.
CAN_L	Označení vodiče CAN sběrnice.
CD	Compact disc. Kompaktní disk.
$CISC$	Complex Instruction Set Computer. Skupina procesorů s podobným návrhem instrukcí.
CRC	Cyclic Redundant Check. Cyklický redundantní součet.
DLC	Data Lenght Code. Informace o délce dat.
EOF	End Of Frame. Konec rámce zprávy.
FET	Field effect transistor. Unipolární tranzistor.
$FIFO$	Firs In First Out. Pamět fronty.
GND	Uzemnění.
HW	Hardware.
ID	Identifier. Identifikátor.
ISO	International Organization for Standardization. Mezinárodní organizace pro normalizaci.
LED	Light Emitting Diode. Světlo vyzařující dioda.
M	Motor.
MCG	Multipurpose Clock Generator. Víceúčelový hodinový generátor.
$PHS1$	PHase Segment 1.
$PHS2$	PHase Segment 2.
PRS	PRopagation time Segment.
$R\ [\Omega]$	Rezistor.
RTR	Remote Transmission Request. Bit označující typ zprávy.
$RS232$	Rozhraní pro sériovou komunikaci.
RxD	Receiver. Přijímač.
SCI	Serial Communications Interface. Sériové komunikační rozhraní.
SMD	Surface Mounted Device. Zařízení pro povrchovou montáž.
SOF	Start Of Frame. Začátek rámce zprávy.
SW	Software.
THT	Through hole technology. Technologie vývodových součástek.
TPM	Timer/PWM. Modul časovače/pulzně šířkové modulace.

TQ	Time Quantum.
TxD	Transceiver. Vysílač.
$UART$	Universal asynchronous receiver/transmitter. Univerzální asynchronní přijímač/vysílač.
V_{DDA}	Analog power supply. Zdroj analogového napětí.
$V_{OUT}[V]$	Výstupní napětí stejnosměrného měniče.
V_{REFH}	Vysoká úroveň referenčního napětí.
V_{REFL}	Nízká úroveň referenčního napětí.
V_{SSA}	Analog ground. Analogové uzemnění.

Kapitola 1

Úvod

Cílem této bakalářské práce je prostudovat současnou konstrukci řídící elektroniky točny kolejiště a zhodnotit současné řešení. Dále vytvořit potřebnou dokumentaci a navrhnout vhodná vylepšení softwaru i hardwaru. Tato vylepšení následně aplikovat.

V první části práce se zabývám popisem konstrukce řídící elektroniky s procesorem AT89C51CC03. K tomuto obvodu jsem dostal pouze soubory k desce z programu EAGLE a samotný zdrojový kód, který není řádně okomentován. Je proto složitější pochopit funkci programu.

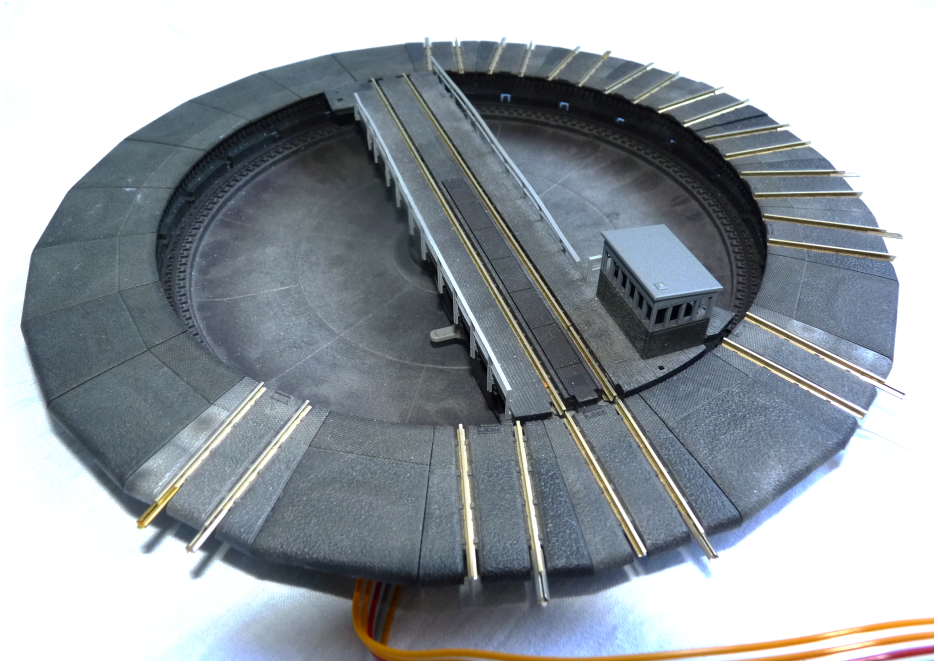
Z tohoto důvodu jsem se rozhodl vytvořit vlastní řídící obvod s novým typem procesoru od firmy Freescale. Na tomto obvodě jsem aplikoval navrhovaná vylepšení, která se převážně týkala záměny THT součástek za SMD a přidáním tří LED diod pro vizuální reakci činnosti obvodu i bez připojeného počítače. Díky tomu lze poznat případné chyby. Pro možné rychlé manuální ovládání jsem přidal dvě tlačítka. Také jsem musel poupravit schéma řídící elektroniky a zaměnit hodnoty součástek, jinak deska byla zcela nebo z části nefunkční. Podrobnější popis celé konstrukce je popsán v kapitole 4.

Protože program pro starší typ procesoru nemohu aplikovat do nového procesoru, napsal jsem program vlastní. Pro psaní programu jsem využil prostředí CodeWarrior. Pro naprogramování procesoru využívám USBDM programátor s jeho ovladači a nastavením pro CodeWarrior. Vyřešená CAN komunikace byla funkční, a tak jsem se zaměřil na ovládání točny pomocí sériové komunikace s počítačem. Kompletní projekt najdete v přiloženém CD.

Základní myšlenkou programu a jeho popisem se zabývám v kapitole 5, včetně rozboru funkcí.

Kapitola 2

Točna kolejiště

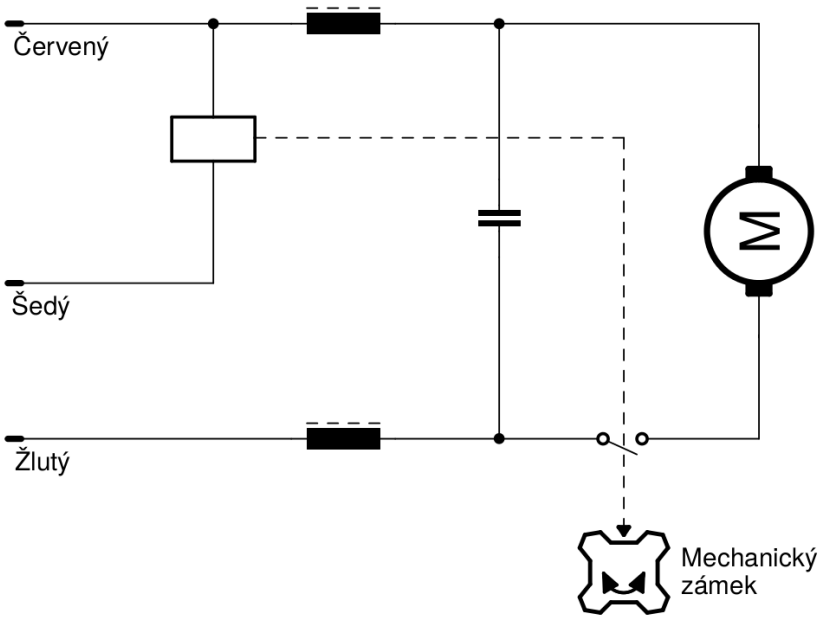


Obrázek 2.1: Reálné zobrazení točny.

Samotná točna má 24 pozic a mechanický zámkový mechanizmus, který umožňuje přesné nastavení mostu pro každou pozici. Pro otočení točny je důležité přivést správnou polaritu napětí (volíme směr otáčení) a pomocí relé uvolnit mechanický zámek. Při otáčení mostu zůstává páka v dané poloze a není nutné přivádět budící proud do relé. Jakmile se most nastaví do jedné z pozic, páka automaticky vypne motor. Tímto způsobem lze snadno otočit mostem točny o jednu pozici zvoleným směrem.

Na skutečném zobrazení točny (viz. 2.1) je vidět rozmístění kolejí pouze do 12 pozic. Pozice jsou rozmístěné tak, jak je v plánu točnu implementovat do stávajícího kolejiště na fakultě elektrotechnické. Bohužel nemám k dispozici plán rozmístění kolejí k točně.

Na obrázku 2.2 je vidět vnitřní zapojení točny. K řídící elektronice je nutné pouze připájet přívodní dráty a softwarově vyřešit napájení. Schéma neobsahuje přidělanou Hallovu sondu pro označení počáteční pozice. Sonda se nachází pod vybranou pozicí. Pro detekci polohy je na straně mostu bez kabinky přidělaný magnet. Ten vytváří magnetické pole, které je snímané Hallovou sondou.



Obrázek 2.2: Vnitřní schéma zapojení točny.

Princip ovládání točny vychází z vnitřního zapojení točny, kdy na přívodní kabely přivedeme napětí podle tabulky 2.1. Těmito kabely je točna zároveň napájena 12 V. Pro otočení pouze o jednu pozici musíme na šedý přívod přivést krátký impulz (cca 600 ms). Tímto impulzem sepneme relé, které odblokuje mechanický zámek a točna se začne otáčet.

	Po směru hod. ručiček	**Proti směru hod. ručiček**
ČERVENÝ	0 V	12 V
ŠEDÝ	12 V	0 V
ŽLUTÝ	12 V	0 V

Tabulka 2.1: Hodnoty napětí pro volbu směru otočení točny.

Kapitola 3

Konstrukce s procesorem AT89C51CC03

3.1 Popis zapojení

Pro lepší orientaci v obvodu najdete v příloze A.1 kompletní schéma zapojení řídící elektroniky točny.

Jádrem celé řídící jednotky je procesor AT89C51CC03. Tento procesor byl zvolen hlavně ze dvou důvodů. Za prvé kvůli rychlosti vykonání instrukcí. Díky vnitřní konstrukci (a SW navolení) se kmitočet oscilátoru upraví tak, že většinu instrukcí je možné vykonat během 6 taktů hodin (nikoli 12 taktů hodin oproti původnímu návrhu). Za druhé kvůli zabudovanému CAN řadiči v procesoru, čímž se ušetří práce při zřizování komunikace s kolejištěm. Určitým vodítkem ve volbě procesoru také mohla být skutečnost použití procesoru se stejným jádrem jako zbytek kolejiště.

Ovšem pro fyzické propojení CAN řadiče procesoru se sběrnicí je nutné mít další prvek. Tuto funkci vykonává rozhraní CAN řadiče PCA82C250. Díky tomuto rozhraní lze mít různou přenosovou rychlost pro komunikaci se sběrnicí a komunikaci s procesorem. Také je plně kompatibilní se standardem ISO 11898 a je primárně určen pro vysokorychlostní aplikace.

Napájení obvodu je z jednotného zdroje kolejiště o 12 V. Na desce plošného spoje je dále upravováno stabilizátorem a stejnosměrným měničem (viz 3.4).

Pro připojení k počítači slouží sériové rozhraní UART, které je přímo vyvedeno z procesoru na piny.

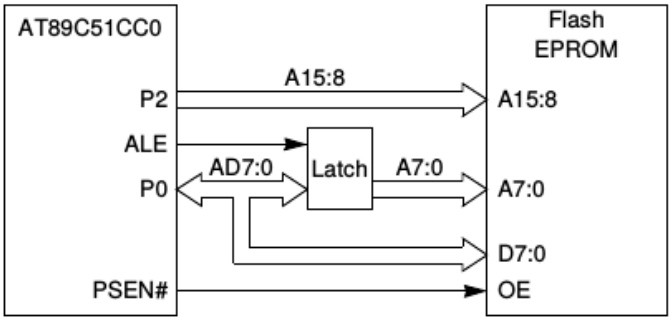
K procesoru je dále připojený vývod pro Hallovu sondu. Tento vstup se využívá pro natočení točny do počáteční pozice, ze které vychází hodnota softwarově definované polohy.

3.2 Procesor AT89C51CC03

Procesor je 8 bitový a obsahuje 256 B RAM, 64 kB FLASH a 2 kB EEPROM paměti. Dále obsahuje tři 16 bitové čítače/časovače, plně duplexní UART sériovou komunikaci, 10 bitový A/D převodník s 8 vstupy, CAN řadič. Všechny podrobnosti k procesoru najdete v datasheetu [3].

3.2.1 Programování procesoru

Program se ukládá do programové paměti FLASH. Paměť FLASH umožňuje tzv. In-System Programming (ISP). Připojení k paměti probíhá přes externí sběrnici (brány 0 a 2) a zahrnuje také řídící signály (PSEN a ALE). Na obrázku 3.1 je znázorněna vnitřní struktura rozhraní pro nahrání programu do paměti FLASH.



Obrázek 3.1: Vnitřní struktura rozhraní pro nahrání programu. Převzato z [3]

Po připojení procesoru k programátoru, lze nahrát program do paměti. Je k tomu nutný software, který bude s programátorem komunikovat.

3.2.2 Hodiny

Pro chod procesoru jsou vytvořeny vnitřní hodinové impulzy pomocí připojeného 24 MHz krystalu na vývody XTAL1 a XTAL2.

Sám procesor má speciální vlastnost. Softwarově si můžeme nastavit vstup pro krystal, tak aby jeho kmitočet byl dělen dvěma. Díky této vlastnosti jádro procesoru potřebuje pouze 6 taktů hodin pro vykonání instrukce (namísto původních 12 taktů hodin). Tím se zrychlí činnost procesoru. Více najdete v datasheetu [3].

3.3 CAN

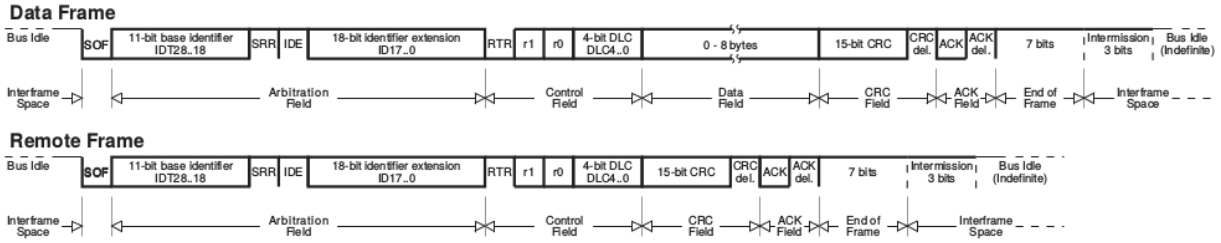
U procesoru AT89C51CC03 je výhodou již zabudovaný CAN řadič. Jako rozhraní CAN řadiče pro fyzickou komunikaci se sběrnicí je použit obvod PCA82C250.

Komunikace mezi obvody a řídícím počítačem probíhá po CAN sběrnici, která je detailně popsáná v Diplomové práci *Návrh systému počítačového řízení modelového kolejiště*[2].

Sběrnici tvoří dva vodiče s označením CAN_H a CAN_L. Je třeba věnovat pozornost impedančnímu přizpůsobení vzhledem k vysokým přenosovým rychlostem. Přenášené bity jsou kódovány rozdílem napětí mezi vodiči. Hodnotě log. 1 přísluší shodné napětí v obou vodičích (tzv. recesivní stav) a hodnotě log. 0 přísluší rozdílné napětí mezi vodiči vyšší než 1,2V (tzv. dominantní stav). Takto je zajištěna priorita zpráv s nižším ID. Také nedochází k rušení souběžného vysílání více obvodů.

Obvody spolu komunikují předáváním zpráv. Tato zpráva v sobě obsahuje identifikátor, datovou informaci a chybovou funkci. Zprávy je možné zasílat i více obvodům a jen pokud je sběrnice volná.

Rámec začíná bitem SOF (Start of frame), který udává začátek zprávy. Následuje pole identifikátoru zprávy. Identifikátor může nabývat délky 11 bitů nebo 29 bitů podle vybraného rámce. Bitem RTR se rozlišuje mezi datovou zprávou a žádostí o data. Následující kontrolní pole obsahuje 4 bity DLC (Data Length Code). Bity v sobě obsahují počet přenášených datových bytů. Datové pole může být až 8 bytů dlouhé. Následuje 15 bitů CRC (Cyclic Redundant Check) sloužící pro detekci chyb. Bit ACK (Acknowledge) je vysílán jako recesivní a v případě správného přijmu zprávy je přepsán dominantním stavem. Zpráva musí být takto ukončena, jinak se vysílání bude opakovat. Rámec je ukončem 7 bity EOF (End Of Frame). Před vysíláním další správy musí být sběrnice alespoň 3 bity volná.



Obrázek 3.2: Rozšířený rámec CAN 2.0B. Převzato z [3]

3.3.1 CAN protokol

CAN protocol je definovaný podle normy ISO 11898 určený především pro vysokorychlostní aplikaci. Protokol podporuje a využívá specifikaci rozšířeného rámce identifikátoru CAN 2.0B s délkou 29 bitů. Graficky znázorněný identifikátor je vidět na obrázku 3.2. Pro komunikaci je k dispozici 15 nezávislých zpráv. Každá zpráva je programovatelná na vysílání nebo příjem.

Díky vhodnému návrhu sběrnice kolejiště je zaručena automatická priorita zpráv s nižším ID. Složení identifikátoru zpráv[1] je uvedeno v tabulce 3.1.

1b	8b	12b	8b
0	ID příkazu	adresa modulu	adresa jednotky v rámci modulu

Tabulka 3.1: Struktura identifikátoru zpráv.

První bit je pro rozšíření komunikačního protokolu. Dalších 8 b slouží jako identifikátor zprávy, díky kterému se určuje typ dat (viz. tab. 3.2; definováno v CAN.h). Rozsah 12 b pro adresu modulu je dostatečně předimenzovaný než co jsou schopny zvládnout použité budiče CAN. Využití posledních 8 b je ponecháno na konkrétních modulech.

3.3.2 Nastavení CAN protokolu

Nastavení parametrů komunikačního protokolu se provádí v inicializaci CANu (viz CAN.c) softwarově nastavením vhodných registrů. V následující tab. 3.3 jsou popsány hodnoty nastavení. Takto uvedené hodnoty jsou určené pro rychlost dat 250 kb/s, frekvenci oscilátoru 24 MHz a bodu vzorkování v 66.7 %.

[1]Sunek, Petr. Návrh systému počítačového řízení modelového kolejiště. Str. 21, Tab.3. [2]

ID [dec]	význam
0	Informace o stavu kolejových úseků
21	Informace o poloze točny
22	Informace o směru otáčení točny
121	Nastavení nové polohy točny
122	Nastavení směru otáčení točny

Tabulka 3.2: Identifikátory zpráv.

BRP	1
PRS	6
PHS1	7
PHS2	7
TQ	24

Tabulka 3.3: Hodnoty pro nastavení časování CAN řadiče.

3.4 Napájení

Hlavní napájení celé desky je vyvedeno přes napájecí konektor ARK. Velikost vstupního napětí není definovaná na přesnou hodnotu, nesmí však překročit 12 V. Toto omezení vzniká kvůli stejnosměrnému měniči, který je zapojený v napětí zvyšujícím módu. Na vstupní konektor je připojen stabilizátor na 5 V sloužící pro napájení procesoru, tlačítek, LED a konektorů. Stejnosměrný měnič napájí FET tranzistory a vývody k točně.

3.4.1 Napájení procesoru

Napájení procesoru je zajištěno stabilizátorem napětí, který zajišťuje konstantní napětí 5 V. Pro správnou funkci je nutné zajistit, aby vstupní napětí na stabilizátoru bylo alespoň o 2 V vyšší. Důvodem je úbytek napětí na stabilizátoru.

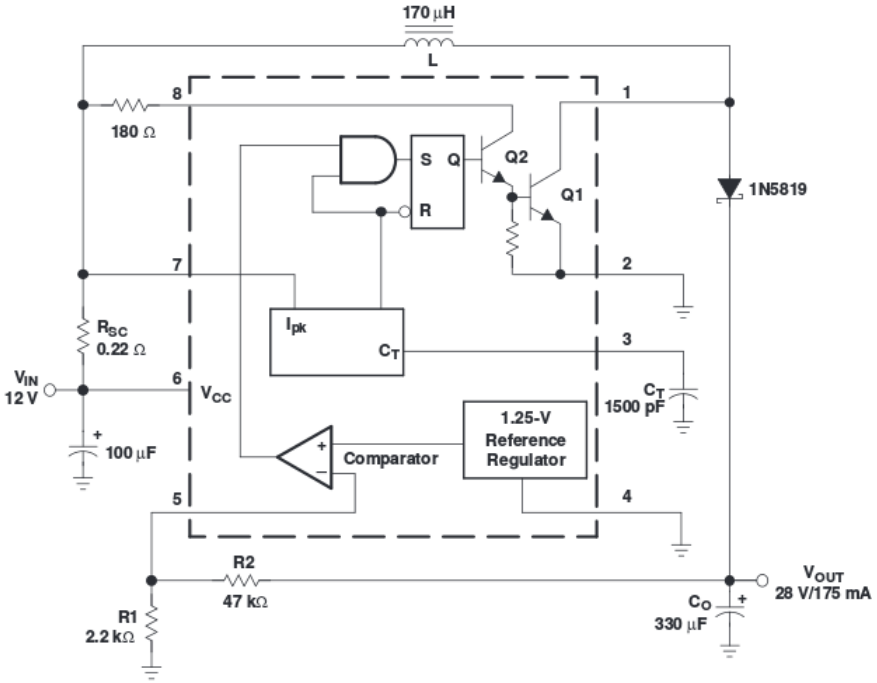
3.4.2 Napájení točny

Napájení FET tranzistorů pro ovládání točny zajišťuje stejnosměrný měnič MC34063AD. Měnič je zapojen v napětí zvyšujícím módu (Step-Up convertor). Zapojení je vidět na obrázku 3.3. Vstupní napětí měniče nesmí překročit 12 V, neboť by napětí nebylo možné snížit a zároveň musí být vyšší než 3 V. Výstupní napětí se volí poměrem rezistorů R1 a R2, které jsou na skutečné desce nahrazeny trimrem. Díky tomu je možné výstupní napětí dle potřeby doladit. Pro ovládání točny je důležité, aby výstupní napětí bylo 12 V. Konstrukce měniče zajišťuje maximální výstupní proud 1.5 A. Pokud bychom chtěli větší proud, museli bychom na výstup připojit zesilovač. Výpočet pro výstupní napětí je

$$V_{OUT} = 1.25 \left(1 + \frac{R2}{R1}\right). \tag{3.1}$$

3.5 Software

Pro napsání programu byl zvolen programovací jazyk C. Celý projekt je rozdělen do několika bloků pro různou činnost programu. Hlavní program se jmenuje *RailCon.c*. Točna



Obrázek 3.3: Step-Up converter. Převzato z [5]

je ovládaná pomocí přijatých zpráv z řadiče CAN, které se dekódují v souboru *CAN.c*. Struktura programu a návaznost bloků na sebe je zobrazena níže.

RailCon.c Hlavní program. Počáteční nastavení. Obsluha točny.
- CAN.h
- PortDefinitions.h
- Globals.h
- HHarCommon.h

CAN.c Obsluha CAN komunikace. De/kódování zpráv.
- CAN.h
- PortDefinitions.h
- Globals.h
- Config.h

Globals.c Globální funkce.
- Globals.h
- CAN.h
- PortDefinitions.h

HHarCommon.c Druhy zpoždění.

PortDefinitions.h Definování bran procesoru.

Kapitola 4

Konstrukce s procesorem MC9S08DZ96

Pro vytvoření nového řídícího obvodu jsem se rozhodl z několika důvodů. Prvním důvodem bylo použití starého CISC procesoru s jádrem 8051. Tento procesor je nahrazen novým, taktéž 8 bitovým, procesorem MC9S08DZ96 od firmy Freescale Semiconductor. Díky tomu mám mnohem více možností pro jeho programování a připojení periférií.

Dalším důvodem byla samotná konstrukce, použití vývodových součástek a nepřítomnost manuálního ovládání točny. Pro svou konstrukci jsem zvolil výhradně SMD součástky s pár výjimkami. Důvodem použití vývodových součástek byla snadná možnost ovládání (tlačítka, LED), připojení konektorů a typ součástky (velkokapacitní kondenzátory).

Posledním důvodem byl fakt o výpadku napájení celého kolejiště při prvotním spuštění. Docházelo k tomu kvůli velkému odběru proudu pro točnu i do zbylých obvodů kolejiště. Řešením je zařazení vstupního zpoždění do programu po připojení napájení.

4.1 Popis zapojení

Samotné zapojení se příliš neliší od předchozího. Procesor MC9S08DZ96 je také napájen ze stabilizátoru napětí na 5 V. Napájení točny využívá totožný stejnosměrný měnič MC34063A v napětí zvyšujícím módu a pro ovládání točny slouží FET tranzistory, které jsou spínány z vývodů procesoru. Je proto důležité, aby vstupní napětí nepřekročilo 12 V. Pakliže by vstupní napětí mělo být vyšší, stejnosměrný měnič by se musel napojit na výstup stabilizátoru napětí.

Pro komunikaci s CAN sběrnicí je k procesoru (CAN řadiči zabudovanému uvnitř struktury) připojeno rozhraní PCA82C250. Rozdíl s předchozím zapojení spočívá v přidání přizpůsobovacího odporu. Pro připojení kabelu sběrnice slouží konektor CANON s 9 piny.

Další konektor je využíván pro připojení Hallovy sondy kvůli indikaci počátečního stavu točny. K procesoru je připojena přímo spolu s napájením sondy. Z procesoru je vyveden UART, který slouží pro sériovou komunikaci rozhraním RS232 a terminálem. Pomocí posílání příkazů z terminálu je možné ovládat točnu. Program pro ovládání točny přes terminál naleznete v přiloženém CD. Velmi důležitý BKGD/MS konektor sloužící pro programování a debugování procesoru a je přímo spojen se vstupem $\overline{RESET}$ a $BKGD/MS$. Poslední samostatné piny, které jsou připojeny k výstupům FET tranzistorů, slouží pro ovládání točny.

Lepší indikace stavu procesoru nebo jeho činnosti je zajištěna třemi červenými LED diodami, které jsou zapojeny tak, že se rozsvítí, pokud na výstup přivedeme log. 0. Díky tomu můžou korespondovat s připojenými tlačítky, které prioritně slouží jako manuální vstup při ladění programu.

Z důvodu možné kontroly a softwarové ochrany řídícího obvodu je vstupní napětí obvodu a výstupní napětí stejnosměrného měniče připojeno na analogový vstup procesoru s A/D převodníkem.

Taktování procesoru je zajištěno připojeným 12 MHz krystalem připojeným ke vstupům XTAL a EXTAL.

4.2 Návrh plošného spoje

Pro vytvoření schématu a plošného spoje jsem využil volně šiřitelnou verzi editoru EAGLE. Úpravy schématu byly následující:

- Použití procesoru MC9S08DZ96 se 48 vývodovým pouzdrem
- Záměna THT součástek za SMD
- Změna frekvence krystalu na 12 MHz
- Změna hodnot vybraných součástek z důvodu správné funkce zapojení
- Připojení vstupního napětí obvodu a výstupního napětí měniče na vstup A/D převodníku
- Připojení LED diod pro vizuální signalizaci
- Připojení tlačítek pro manuální ovládání točny
- Přidání BKGD/MS konektoru pro programování a debugování procesoru

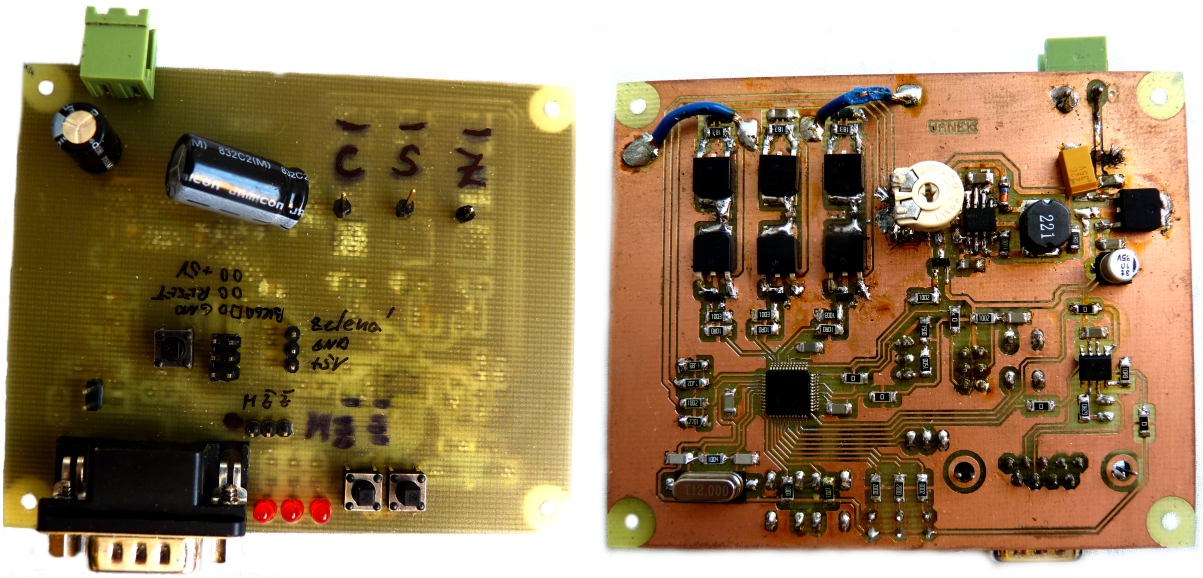
Desku jsem navrhoval pomocí principů tvorby desek plošných spojů, které jsou popsány v knize [1]. Výsledek mé práce je vidět na obrázku 4.1 nebo v přílohách A.2 a B.2. Výstupy pro frézu naleznete na přiloženém CD.

4.3 Procesor MC9S08DZ96

O řízení celého obvodu se stará 8 bitový procesor MC9S08DZ96 s 96 kB FLASH, 2 kB EEPROM a 6 kB RAM pamětí. Dále obsahuje 16 kanálový A/D převodník, MSCAN, SCI, 2 časovače (6 kanálový TPM1, 2 kanálový TPM2), BKGD/MS a 40 volitelně nastavovaných vývodů. Procesor má 48 vývodový LQFP pouzdro. Veškeré nezmíněné podrobnosti naleznete v datasheetu [4].

4.3.1 Vývod Background/Mode select

Pokud je aktivní reset (v log. 0), vývod BKGD/MS má funkci jako mode select. Ihned po náběhu resetu, funkce vývodu se změní na background a může být použit pro debugování. Vývod obsahuje vnitřní pull-up zařízení, vstupní hysterezi a standardní výstupní ovladač.



Obrázek 4.1: Vyobrazení hotového plošného spoje.

Pokud k vývodu není nic připojeno, procesor nastaví normální provozní režim s náběžnou hranou resetu. Pokud je připojen debugovací systém k 6 pinovému standardnímu background konektoru, udrží to BKGD na nízké úrovni při náběžné hraně resetu. Procesor je tak nucen přepnout do debugovacího background módu.

4.3.2 A/D převodník

Převodník u procesoru je využit pro snímání vstupního napětí obvodu a výstupního napětí stejnosměrného měniče. To vede k softwarové ochraně a indikaci v případě klesajícího/zvyšujícího napětí, což by mohlo zničit součástky.

Převodník je 12 bitový a jako referenci napětí používá vstupy vysoké úrovně referenčního napětí V_{REFH} a nízké úrovně referenčního napětí V_{REFL}. Referenční vstupy jsou vnitřně spojeny se vstupy V_{DDA} a V_{SSA}, tzn. referenční napětí je totožné s napájecím napětím procesoru (5 V). Maximální připojitelné napětí na převodník je 5.5 V. Kvůli tomu je na vstup převodníků připojen dělič napětí, neboť měřené napětí nabývá hodnoty až 12 V.

4.3.3 Sériová komunikace

K připojení k počítači je vyvedeno rozhraní UART (vývody TxD2, RxD2 a GND), které jsou připojeny k RS232. Propojení zajišťují 3 dráty na jednom konci připojení k 9 pinovému konektoru CANON a na druhém k jednoduchému počítačovému konektoru 1x3. Jako software pro komunikaci využívám volně stažitelný sériový terminál Hercules.

Jako přenosovou rychlost jsem zvolil 9600 bit/s. Formát dat je 8 bitový, což umožňuje neshodu v přenosové rychlosti až 4.5 %.

Je důležité nepřekročit maximální hladinu napětí 5.5 V při odesílání a vysílání zpráv.

4.3.4 Taktovací kmitočet

O výběr zdroje taktovacích impulzů se stará víceúčelový hodinový generátor (MCG) poskytující výběr několika zdrojů pro procesor. Dále řídí externí oscilátor. Pro taktovací

kmitočet procesoru je připojen externí oscilátor (krystal s frekvencí 12 MHz) na vývody XTAL a EXTAL.

4.3.5 Časovače

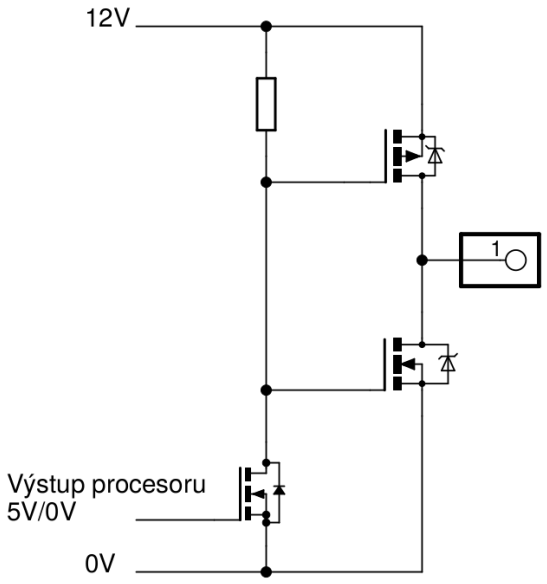
V programu využívám časovače TPM1 a TPM2 pouze pro vnitřní vyvolání přerušení. Výstupy časovačů zůstaly nevyužity pro činnost obvodu. Zdroj hodinových impulzů je stejný jako pro sběrnici procesoru.

4.4 Periférie

4.4.1 Připojení točny

Zapojení a ovládání točny je popsáno v kapitole 2. Točna je k řídícímu obvodu připojena přívodními kabely na výstupy FET tranzistorů, které jsou vhodným způsobem spínané. Tím se přivede příslušná polarita napětí na motor točny a krátký impulz na relé pro uvolnění mechanického zámku, otočení točny.

Na obrázku 4.2 je vidět způsob zapojení FET tranzistorů a jejich spínání. Pokud nastavíme výstup z procesoru do log. 0., FET tranzistor připojený na vstup výkonových tranzistorů se neotevře a na bráně G bude napětí 12 V. Tím výkonový FET transistor s P-kanálem zůstane zavřený a s N-kanálem se otevře. Na výstupu k točně bude 0 V. Pokud výstup z procesoru nastavíme do log. 1, děj proběhne obráceně. FET tranzistor na vstupu výkonových transistorů se sepne. Na bráně G bude 0 V. Tak se otevře tranzistor s P-kanálem a s N-kanálem zůstane zavřený. Na výstupu k točně bude 12 V.

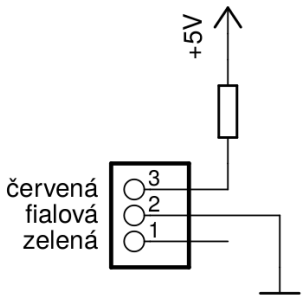


Obrázek 4.2: Zapojení jedné větve FET tranzistorů pro ovládání točny.

4.4.2 Připojení Hallovy sondy

Hallova sonda je připojena k točně na vybranou pozici a slouží pro počáteční nastavení točny, ze které se softwarově inkrementuje pozice točny. Pro zjištění správného natočení

mostu je na jeden konec přilepen magnet, který vytváří magnetické pole pro snímání Hallovou sondou. Pokud sonda indikuje magnetické pole magnetu, nastaví svůj výstup na opačnou hodnotu, ve které se předtím nacházel.

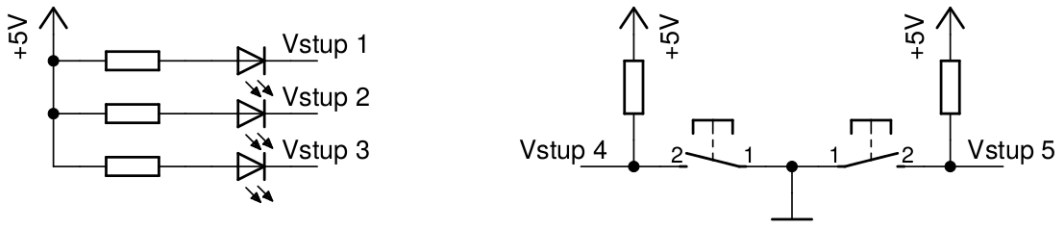


Obrázek 4.3: Zapojení Hallovy sondy.

4.4.3 Tlačítka a LED

Připojení tlačítek a LED diod je vidět na obrázku 4.4. Jejich aktivní úroveň je log. 0. Ke tlačítkům jsou připojeny pull-up rezistory, které zvedají logickou úroveň na vstupu procesoru pro lepší rozeznání, jestli je tlačítko stisknuto nebo ne.

LED diody slouží pro vizuální zpětnou vazbu stisku tlačítek a reakcí točny. Tlačítka ve funkčním programu slouží pro manuální ovládání točny, posun točny o jednu pozici vpravo nebo vlevo.



Obrázek 4.4: Zapojení LED diod a tlačítek.

4.4.4 USBDM Programátor

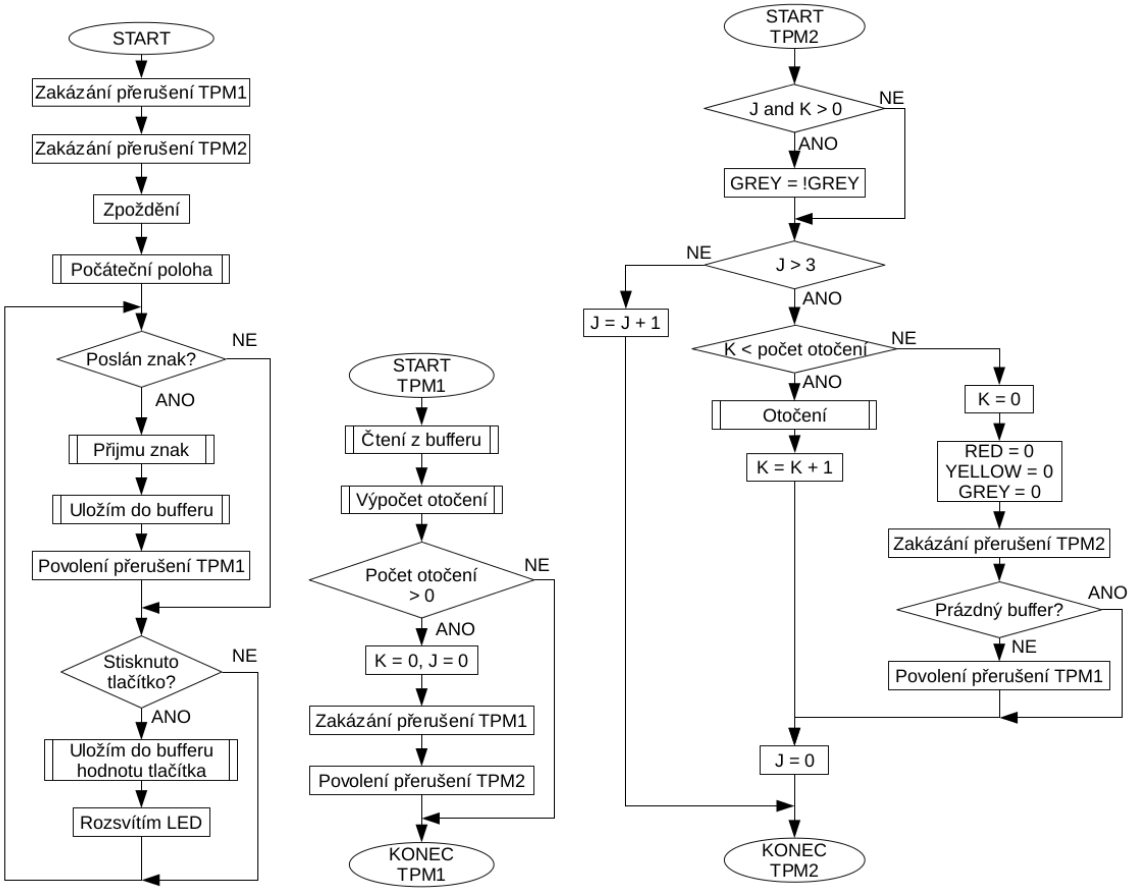
K nahrání programu do procesoru a debugování byl využit programátor USBDM JM V1.6. Připojuje se na background konektor. Spolu s instalací ovladače pro programátor se instaluje aplikace programátoru, přes kterou se nahrává program (soubor typu *.abs) do paměti FLASH. Také je možné upgradovat nastavení CodeWarrioru přidáním možnosti debugování pomocí BDM. Takto nastavený CodeWarrior bude komunikovat s programátorem a lze snadno debugovat program. Zároveň programátor napájí procesor po dobu jeho připojení.

Kapitola 5

Software

V této kapitole se zabývám samotným programem. Zaměření je na sériovou komunikaci a logiku ovládání točny. V další částech kapitoly postupně probírám důležité body programu a vysvětluji jejich funkci. Kompletní funkční program naleznete v přiloženém CD. Soubory jsou uložené ve složce *Tocna_1_0/Sources*. Pro psaní programu jsem využil prostředí od firmy Freescale Semiconductor CodeWarrior v10.5 a programovací jazyk C. Vytvořený projekt využívá nástroje Processor-expert.

5.1 Vývojový diagram



Obrázek 5.1: Vývojový diagram činnosti programu.

5.2 Hlavní program

Předtím než začnu vysvětlovat vytvořený program, uvedu důležité hardwarové nastavení procesoru.

- Po resetu jsou přerušení časovačů zakázána.
- Časovače jsou nastaveny na hodnoty
  - TPM1 250 ms
  - TPM2 720 ms
- Nastavení vstupů/výstupů sériové komunikace
  - Vysílač (TxD) - neinvertovaný
  - Přijímač (RxD) - invertovaný
- Taktovací kmitočet je 8 MHz

Při spuštění programu po připojení napájení nebo po resetu je nutné zamezit okamžitému velkému odběru proudu, který je potřeba pro otáčení točny. To je zařízeno vložením čekací smyčky. Tuto smyčku lze pozorovat blikající LED diodou.

Aby byla zaručena správná inkrementace pozice, řídící obvod točnu natočí do počáteční polohy po každém resetu. Natočení do počáteční polohy je provedeno v podprogramu, který otáčí točnou do doby, než dostane impulz z Hallovy sondy. Výstup z Hallovy sondy nabývá hodnot logických úrovní. Po resetování obvodu je výstup nastaven na takovou logickou úroveň, v jaké se nacházel před resetem. Proto nedokážeme říct, jestli se točna před i po resetu nachází v žádané poloze, ani nevíme s jakou počáteční úrovní ze sondy máme pracovat. Zároveň také záleží z jaké strany se točna k sondě otočila, neboť to ovlivní výběr počáteční pozice. Zvolil jsem si otáčet točnou po směru hodinových ručiček.

V okamžiku, kdy je točna natočena, spustí se nekonečný cyklus. V této nekonečné smyčce se čeká na přijetí znaku ze sériového kanálu. Pokud znak byl přijat, přečte se a je poslán do funkce *buffer_write()*, kde se do kruhového bufferu uloží pozice kolejiště příslušící přijatému znaku. Funkce bufferu bude vysvětlena dále.

V okamžiku, kdy potřebujeme posunout točnu jen o jednu pozici (nebo pár pozic), můžeme využít tlačítek. Každé tlačítko otočí točnou o jednu pozici opačným směrem. Snímání stisku tlačítek probíhá pouze jednou za definovanou dobu, a to z toho důvodu, aby se nenaplnil celý buffer jedním stiskem tlačítka.

5.3 Volba pozice

Komunikace s procesorem probíhá v textovém terminálu jednoduchém principu. Uživatel vysílá znaky abecedy od A do X. Těmto písmenům přísluší číselná hodnota od 1 do 24, která je ukládaná do kruhového bufferu. Označení postupuje proti směru hodinových ručiček od počáteční polohy. Program rozlišuje i velikost písmen. Využití velkých a malých písmen dává možnost výběru, jak bude točna k žádané pozici natočena. Jestli se most natočí kabinkou k pozici, nebo druhým koncem mostu. Záleží jakým směrem vlak pojede.

5.4 Funkce pro sériovou komunikaci

Přijetí znaku přes sériovou komunikaci vykonává procesor. Program pouze hlídá, jestli byl znak přijat a následně ho vyčte z registru, kam ho procesor uložil. Tuto činnost provádí funkce *uart_recv()* v souboru *Tocna_1_0/Sources/serial.c*. Následující funkce se nacházejí ve stejném souboru.

Protože otočení točny na žádanou pozici trvá několik sekund, nemůžeme každou žádost vykonat hned po přijetí. A aby se data neztrácela, jsou ukládaná do kruhového bufferu, což je zásobníková paměť FIFO. Maximální počet znaků je 15, ale dá se jednoduše přepsáním hodnoty rozšířit.

Když je do bufferu uložena hodnota, povolí se přerušení časovače TPM1, který má za úkol číst postupně data z bufferu a vypočítat, o kolik pozic a jakým směrem se točna bude otáčet. Pak zakáže vlastní přerušení, aby nedocházelo ke ztrátě dat při předčasném čtení z bufferu a možným chybám při ovládání točny. Zároveň se povolí přerušení časovače TPM2, který slouží pro vydávání pokynů točně.

Přerušení časovače TPM1 se povoluje tak dlouho, dokud jsou platná nepřečtená data v bufferu. Data se čtou teprve, až se točna dostane do žádané pozice.

Logika bufferu je zabezpečena proti přepisování nepřečtených dat a čtení již jednou přečtených. Úryvek kódu znázorňuje funkci bufferu.

Důležitým aspektem funkce je signalizace prázdného (Buffer_EMPTY) nebo plného (Buffer_FULL) bufferu. Dále se kontrolují indexy zápisu (WR) a čtení (RD). Indexy se inkrementují a platí u nich jednoduché pravidlo. Index čtení nesmí předběhnout index zápisu a index zápisu se nemůže inkrementovat vícekrát než je maximální velikost bufferu. Pozice indexů určují stav bufferu.

```
void buffer_write(unsigned char byte)
{
        if(!Buffer_FULL)
        {
                switch(byte)
                  {
                          case 'A': Buffer_tocna[WR++] = 1;break;
                          case 'B': Buffer_tocna[WR++] = 2;break;
                          case 'C': Buffer_tocna[WR++] = 3;break;
                                    ...
                          default: return;
                  }

                Buffer_EMPTY = FALSE;
                if(TPM2SC_TOIE == 0)
                        TPM1SC_TOIE = 1;
                if(WR >= BUFFER)
                        WR = 0;
                if((WR == RD) && !Buffer_EMPTY)
                         Buffer_FULL = TRUE;
        }
}

unsigned char buffer_read(void)
{
        volatile unsigned char byte;

        if(!Buffer_EMPTY)
        {
                byte = Buffer_tocna[RD++];
                Read_last = byte;
                Buffer_FULL = FALSE;
                if(RD >= BUFFER)
                        RD = 0;
```

```
                if(RD == WR)
                {
                        Buffer_EMPTY = TRUE;
                        TPM1SC_TOIE = 0;
                }

                return (byte);
        }
}
```

5.5 Ovládání točny

Funkce pro ovládání točny jsou uloženy v souboru *Tocna_1_0/Sources/tocna.c*.

Pokud přijatý znak byl uložen do bufferu a bylo povoleno přerušení časovače TPM1, vypočte se směr a hodnota, o kolik se má točna otočit (resp. kolikrát se má otočit o jednu pozici daným směrem). Výpočet pohybu točny je systematičtější. Funkce vybírá tu nejkratší cestu k žádané pozici. Úryvek kódu toto znázorňuje pro jeden směr.

```
if(byte > last)
{
        navrat = byte - last;

        if((24 - navrat) < navrat)
        {
                navrat = 24 - navrat;
                smer = 'L';
        }
        else
                smer = 'R';
        last = byte;
        return(navrat);
}
```

Pokud žádáme natočení točny na žádanou pozici druhým koncem mostu (ten bez kabinky), výpočet pro otočení se provede stejným způsobem. To jaký konec mostu se otočení na žádanou pozici se definuje již při zápisu do bufferu. Při přijmu malého znaku abecedy se do bufferu zapíše hodnota, která odpovídá protilehlé hodnotě žádané pozice, např. žádáme otočení točny koncem bez kabinky do pozice 12. Procesor přijme znak 'l' ('L' odpovídá hodnotě 12) a do bufferu uloží hodnotu 24.

Jakmile je hodnota vypočtena, povolí se přerušení časovače TPM2, a zároveň zakáže přerušení časovače TPM1. Je to z toho důvodu, aby nedocházelo k chybám. V přerušení časovače TPM2 dochází k pootočení točny o jednu pozici vypočteným směrem. Tento děj se vykonává tolikrát, dokud nedojede na žádanou pozici. Pohyb točny se signaliuje blikáním LED diod, každá pro jeden směr pohybu. Využití časovače je hlavně z toho důvodu, aby bylo vůbec možné natočit točnu do žádané pozice. Chybí měření absolutní pozice (například několika Hallovými sondami). Časovač je proto nastaven na dostatečně velký časový úsek, kdy je točna schopná se posunout o jednu pozici, na které na chvíli stojí než dostane nový impulz posunu. Jakmile se dokončí otočení točny a buffer není prázdný, povolí se přerušení časovače TPM1 a čtou se další data. Celý děj se opakuje.

Otočení točny se vynutí sepnutím FET tranzistorů podle tabulky 2.1. Úryvek kódu znázorňuje časování pootočení o jednu pozici. Aby se točna na další pozici zastavila, musíme na šedý přívodní vodič přivést krátký impulz. Točna vyžaduje necelé 3 s, aby

se otočila o jednu pozici (čas se mírně liší pro oba směry). Časovač lze nastavit na maximální hodnotu 1 s. Proto se děj v přerušení vykonává pouze jednou za 4 periody časovače. Z toho impulz na šedý přívodní vodič trvá pouze 1 periodu. Po dokončení otáčení se přestane napájet točna, a tím dojde k poklesu odebíraného proudu.

```
ISR(ISR_TPM2)
{
        (void)TPM2SC;
        TPM2SC_TOF = 0;

        if((j < 1) && (k > 0))
        {
                GREY = !GREY;
                LED1 = 1;
                LED3 = 1;
        }

        if((j++ >= 3) || (k == 0))
        {
                if(k++ < pocet_otoceni)
                {
                        otoceni_tocny();
                }
                else
                {
                        k = 0;
                        RED = 0;
                        YELLOW = 0;
                        GREY = 0;

                        LED1 = 1;
                        LED3 = 1;

                        TPM2SC_TOIE = 0;
                        overeni_bufferu();
                }

                j=0;
        }
}
```

Kapitola 6

Závěr

Při prostudování konstrukce řídícího obvodu s procesorem AT89C51CC03 jsem zjistil, že obvod je funkční a s kolejištěm komunikuje přes CAN sběrnici. Také jsem zjistil, že po náběhu napájení celého kolejiště točna začne odebírat příliš velký proud, kdy tento odběr zdroj nedokáže utáhnout. Tento problém se vyřešil zařazením dostatečně velkého zpoždění na začátek programu, kdy hodnota odebíraného proudu točnou nepřesáhne 90 mA.

Dále jsem měl k dispozici zdrojové kódy řídícího obvodu bez jakékoli jiné dokumentace. Musel jsem se tedy orientovat pouze pro mě zcela neznámým kódem, který nebyl řádně okomentován. Pochopení programu bylo o to složitější.

Zdrojový kód je postaven na přijímání zpráv z CAN sběrnice, které se musí dekódovat a podle toho se určí činnost točny. Z důvodu chybějících komentářů, nezjištění podoby zpráv pro točnu a nemožnost aplikovat vylepšení komunikace jsem se tímto problémem nezabýval.

Ovšem při návrhu nového řídícího obvodu jsem připojil CAN rozhraní k procesoru. Díky tomu je obvod připraven pro budoucí využití komunikace s CAN sběrnicí.

Kvůli chybějící dokumentaci a tudíž lepší orientaci v programu jsem se rozhodl vyrobit vlastní řídící obvod s aplikovanými vylepšeními. Změny jsou následující.

- Použití procesoru MC9S08DZ96 se 48 vývodovým pouzdrem
- Záměna THT součástek za SMD
- Změna frekvence krystalu na 12 MHz
- Změna hodnot vybraných součástek z důvodu správné funkce zapojení
- Připojení vstupního napětí obvodu a výstupního napětí měniče na vstup A/D převodníku
- Připojení LED diod pro vizuální signalizaci
- Připojení tlačítek pro manuální ovládání točny
- Přidání BKGD/MS konektoru pro programování a debugování procesoru

Navržený obvod jsem zapájel a oživil.

Vlastní program jsem psal taktéž v jazyce C a použil prostředí CodeWarrior, který mi umožnil debugovat program a odstranit všechny chyby.

V programu se zabývám sériovou komunikací přes rozhraní UART. Komunikace s počítačem probíhá v textovém terminálu a je jednosměrná. Procesor pouze přijímá znaky vysílané uživatelem. Do procesoru posílám znaky abecedy, které odpovídají jednotlivým pozicím točny. Jakmile procesor přijme znak, vydá povel točně. Z terminálu jde poslat několik znaků za sebou, i když se točna ještě nedotočila do žádané pozice. Přijaté znaky se ukládají do kruhového bufferu a tím nejsou ztracena data. Jako další vylepšení je možnost rozlišovat konce mostu točny. Tím si uživatel může zvolit, jak se točna přesně natočí, tedy jestli lokomotiva bude stát přední částí nebo zadní částí k výjezdové koleji. Volba konce mostu je definovaná velikostí vyslaného znaku abecedy. Točna je ovládaná inteligentně, tzn. že se otáčí nejkratší cestou k žádané pozici. Tím se čas na otočení zkrátí na minimum.

Literatura

[1] Záhlava, Vít. *Návrh a konstrukce desek plošných spojů.* Praha: Nakladatelství BEN - technická literatura, 2010 1. vydání. ISBN 9788073002664

[2] Sunek, Petr. *Návrh systému počítačového řízení modelového kolejiště (Diplomová práce).* Plzeň: Západočeská univerzita, FEL, 2006. 55 stran, 5 příloh. Vedoucí práce Ing. Hloušek Petr Ph.D.

[3] *AT89C51CC03 Datasheet.* Atmel corporation 2006

[4] *MC9S08DZ96 Datasheet.* Freescale Semiconductor 2008

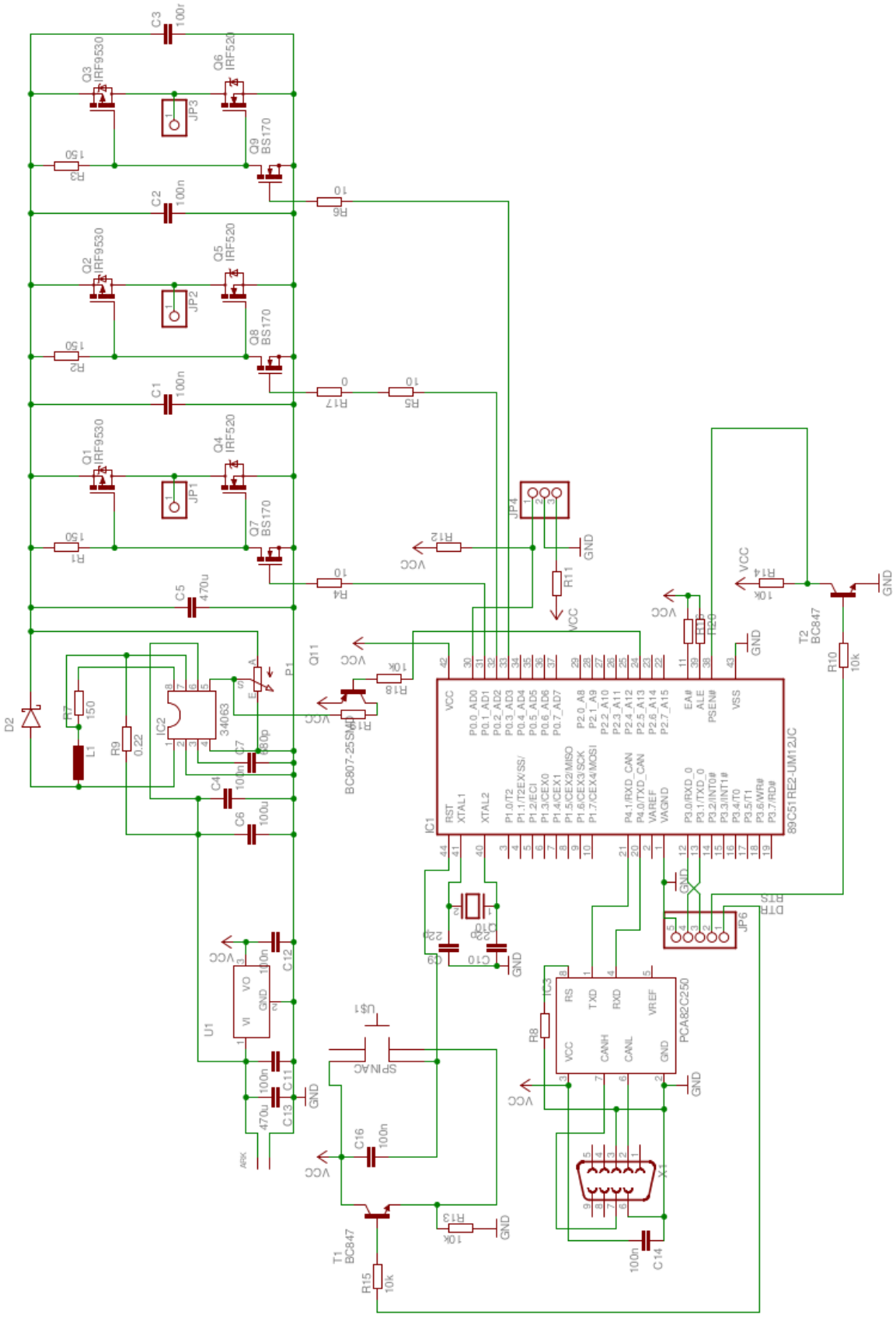
[5] *MC34063A Datasheet* Texas Instruments 2011

Příloha A

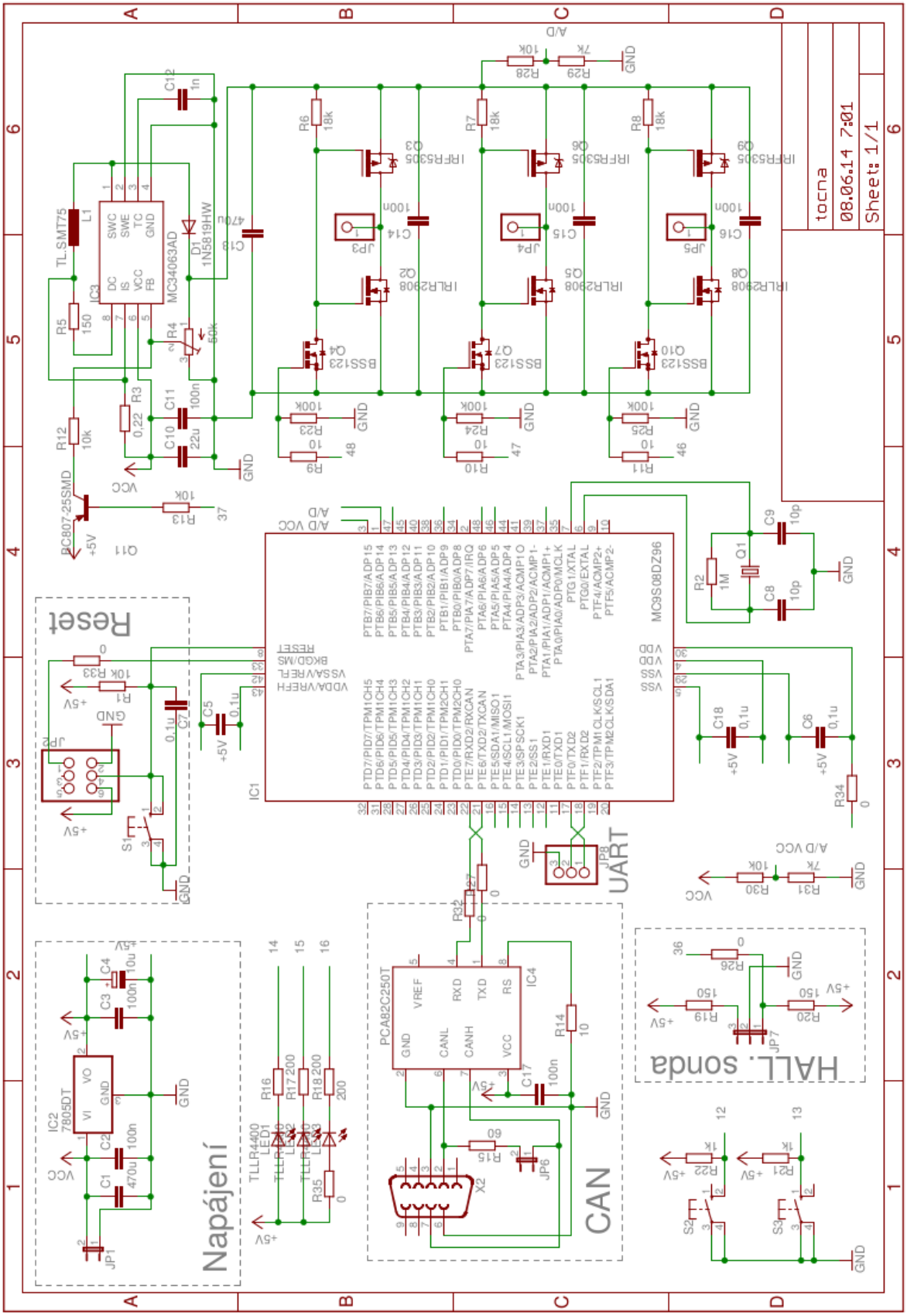
Schémata zapojení

A.1 Schéma řídící elektroniky s 8051

A.2 Schéma řídící elektroniky s Freescale



Obrázek A.1: Schéma řídící elektroniky točny s procesorem 89C51CC03VA



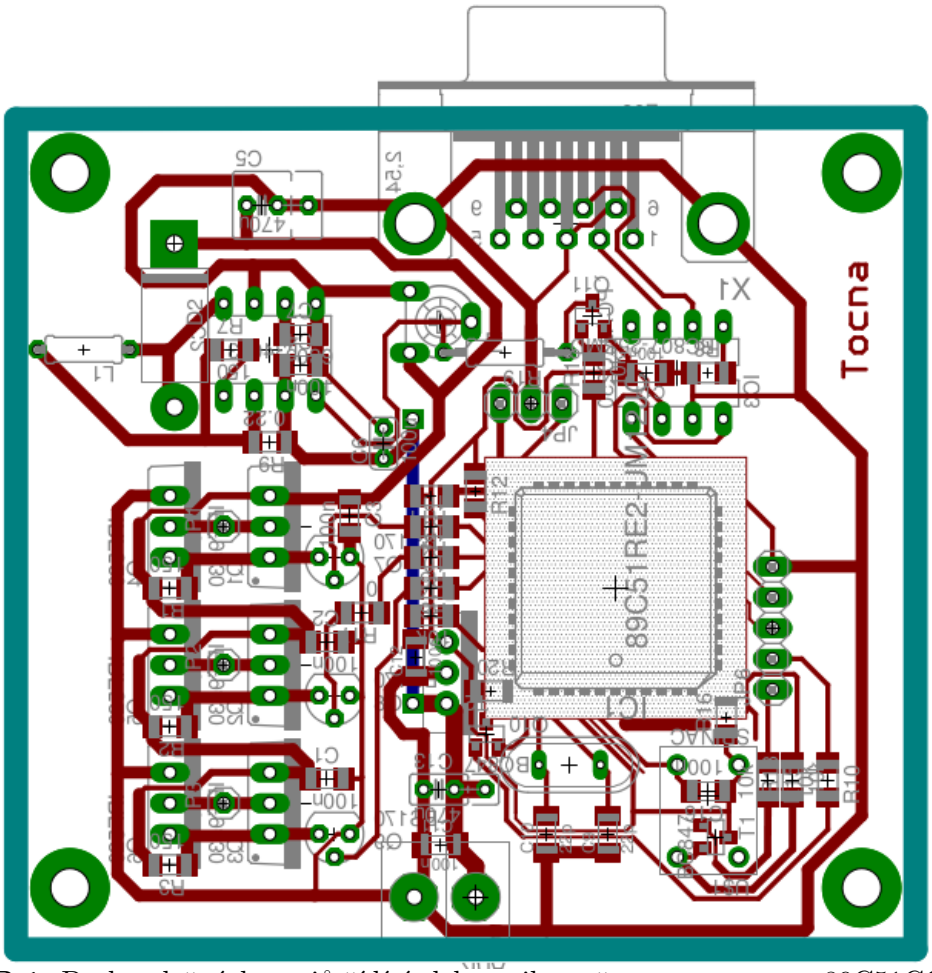
Obrázek A.2: Schéma řídící elektroniky točny s procesorem MC9S08DZ96

Příloha B

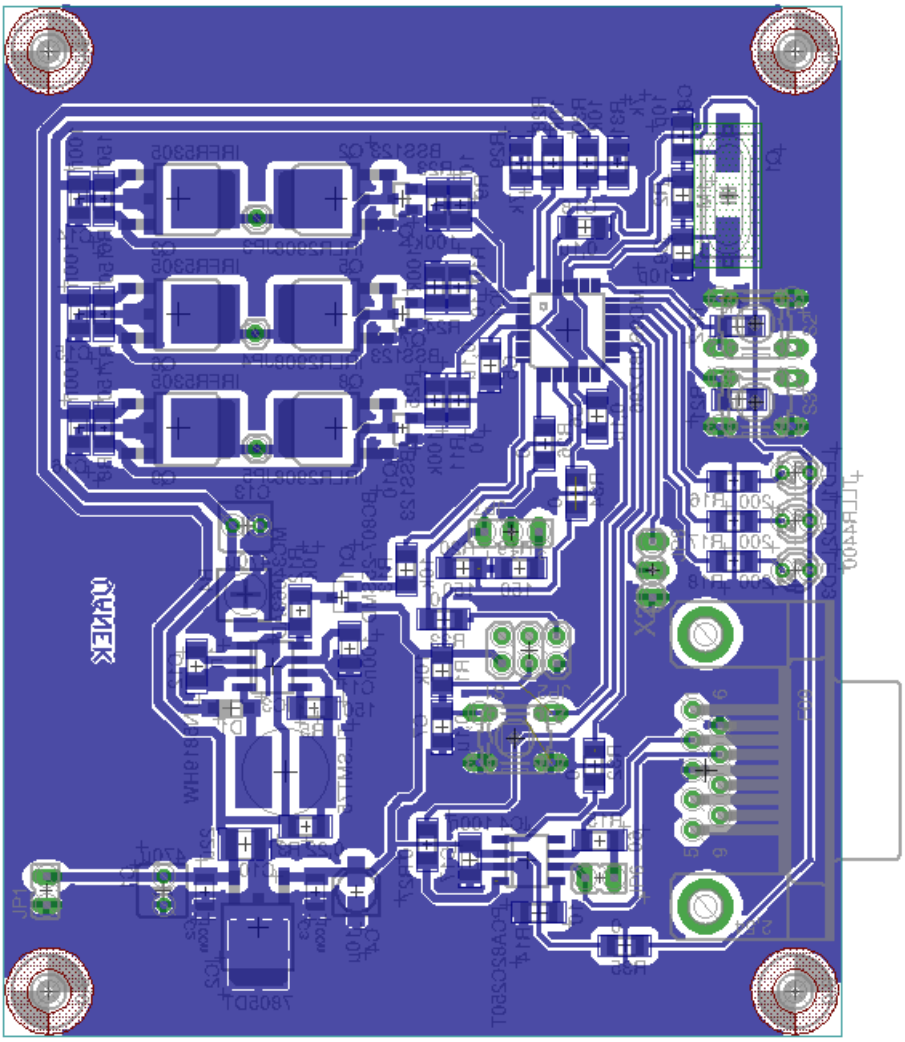
Desky plošných spojů

B.1 Modul řídící elektroniky s 8051

B.2 Modul řídící elektroniky s 8051



Obrázek B.1: Deska plošných spojů řídící elektroniky točny s procesorem 89C51CC03VA



Obrázek B.2: Deska plošných spojů řídící elektroniky točny s procesorem MC9S08DZ96

Obsah CD

- Bakalářská_práce
- Zadání práce
- Konstrukce_8051
  - Zdrojové kódy
  - Návrh DPS
  - Datasheet procesoru
- Konstrukce_Freescale
  - Zdrojové kódy
  - Návrh DPS
  - Odkazované datasheety
  - Ovladače a aplikace pro programátor
  - Terminál pro sériovou komunikaci
  - Obrázky točny a řídícího obvodu