Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Programové vybavení jednotky pro řízení elektrických motorů

Plzeň, 2012 Michal Mourek

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 23. července 2012

Michal Mourek

Abstract

This thesis is aimed on software methods for controling of DC and stepping engines. One of goals is also dedicated to hardware solutions of such control. It will also introduce you to parameters and circuit design of board V850_M1.
Probably the most important goal was implementation of protocol that can control connected engines. Communication protocol should use human-friendly commands. Final defined protocol can be found in chapter 4.
Testing of such protocol required standalone application which implements basic functions for operation with V850_M1. These routines defines startup procedure, main loop and communication interfaces. Protocol service is interrupt-based and operates with commands queue.
Implemented firmware uses NEC microcontroller from V850 series model called IG3 which is determined for such purposes (engine control). Basic protocol is architectural independent and can be ported for any other platform.

Obsah

Seznam obrázků **iv**

Seznam tabulek **v**

Poděkování **1**

1 Úvod **2**

2 Motory **3**

2.1 Typy motorů 3

2.2 Stejnosměrné motory 3

2.2.1 Obecně 3

2.2.2 Řízení motorů 3

2.2.3 Hardwarové řešení řízení motorů 4

2.3 Krokové motory 5

2.3.1 Obecně 5

2.3.2 Řízení motorů 5

3 Deska V850_M1 **7**

3.1 Procesor NEC V850-IG3 7

3.1.1 Obecně 7

3.1.2 Speciální funkce 7

3.1.3 Porty 8

3.2 Rozhraní 8

3.2.1 I2C 8

3.2.2 RS232 . . . 8
3.2.3 SPI . . . 9
3.2.4 USB . . . 9
3.3 CPLD . . . 10
3.3.1 JTAG . . . 10
3.4 Porty desky . . . 11
3.5 Motory . . . 11

4 Protokol **12**
4.1 Paket . . . 12
4.1.1 Návrh a popis paketu . . . 12
4.2 Popis a seznam příkazů . . . 13
4.2.1 Inicializační příkazy . . . 13
4.2.2 Normální příkazy . . . 14
4.2.3 Urgentní příkazy . . . 15
4.3 Popis a seznam odpovědí . . . 15
4.4 Popis a seznam chyb . . . 17
4.5 Komunikace a zpracování příkazů . . . 18
4.5.1 Zpracování příkazů . . . 18

5 Programové vybavení **19**
5.1 Procesor . . . 19
5.2 Pulzně šířková modulace . . . 19
5.2.1 Nastavení čítače TAA2 . . . 19
5.3 Motory . . . 23
5.4 Datový model . . . 25
5.5 Průchod příkazu - Běh programu . . . 26
5.6 Popis modulů a funkcí . . . 27
5.6.1 Modul low_level_initialization.c . . . 27
5.6.2 Modul UART.c . . . 28
5.6.3 Modul I2C.c . . . 28

5.6.4 Modul Engines.c 29
5.6.5 Modul Commands.c 29
5.6.6 Modul TAA2.c 30
5.6.7 Modul main.c 30

6 Závěr **31**

Symboly a zkratky **32**

Literatura **33**

A Rozsáhlé tabulky **34**

Seznam obrázků

2.1 PWM modulace 4
2.2 H-můstek [3] 5

5.1 Blokové schéma čítače TAA2 [6] 20
5.2 Běh programu 26
5.3 Obsluha přerušení komunikačních rozhraní 27

Seznam tabulek

4.1 Paket 12
4.2 Adresa 13
4.3 Příkaz 13
4.4 Priority příkazů 13

5.1 Registr TAA2CTL0 20
5.2 Registr TAA2CTL1 20
5.3 Registr TAA2IOC0 21
5.4 Registr TAA2IOC1 21
5.5 Registr TAA2IOC2 21
5.6 Registr TAA2OTP0 22
5.7 Registr TAA2CCR0 22
5.8 Registr TAA2CCR1 22
5.9 Registr TA2OVIC 22
5.10 Registr TAA2CCIC0 23
5.11 Registr TAA2CCIC1 23
5.12 Bity registrů vývodu P01 23
5.13 Číslování motorů 24
5.14 Značení motorů 24

A.1 Seznam příkazů 34
A.2 Seznam odpovědí 35
A.3 Seznam chybových zpráv 36

Poděkování

Rád bych poděkoval svému vedoucímu práce panu Dr. Ing. Karlovi Dudáčkovi za rady a velkou trpělivost.
Též chtěl poděkovat své rodině, která i přes můj počáteční neúspěch mne dokázala dál podporovat. V neposlední řadě chci poděkovat i Ing. Jiřímu Novotnému za velice kvalitní komentáře.

1 Úvod

Tato bakalářská práce se zabývá metodami řízení stejnosměrných a krokových motorů, dále také hardwarovém řešení ovládání. Popisuje též obvodové řešení a parametry desky *V850_M1*, která je navržena jako univerzální prostředek pro řízení různých druhů motorů. Rovněž je v ní návrh protokolu, který bude deska používat pro komunikaci s nadřazeným systémem. Následuje implementace programového vybavení procesoru *NEC V850-IG3*.

Cílem bakalářské práce bylo navržení a vývoj programového vybavení desky. Aniž by jsme si to uvědomovali, tak různé motory jsou ve všech možných domácích spotřebičích (např. šlehače, mixéry, kartáček na zuby, ad.)

Hlavní přínos je v tom, že se jedná o univerzální firmware, který je platformě nezávislý a umožňuje naráz ovládat nejenom více motorů, ale také různé typy. Jeho všestrannost jej předurčuje k vysokému procentu použití. Příkladem může být moje naportování na univerzální desku V850_M1.

Programové vybavení desky, umožňuje komunikovat přes několik rozhraní, které jsou např. v PC (RS232 a USB), tak i v rámci mikropočítačových systémů (I2C a UART). Přednost je ovládání jednotlivých motorů pomocí příkazů, které jsou pro člověka dostatečně intuitivní.

2 Motory

2.1 Typy motorů

V praxi se setkáváme s různými typy motorů, které dělíme podle různých kritérií. Základním dělením je:

Stejnosměrný

Krokový

Třífázový

Servo

V dalších podkapitolách se zaměříme na stejnosměrný a krokový motor. Hlavním důvodem je jejich použití v této práci. S dalšími typy motorů je možné se seznámit např. v [1].

2.2 Stejnosměrné motory

2.2.1 Obecně

Jedná se o točivý elektromotor napájený stejnosměrným proudem, využívající magnetické pole. Motor má vnitřní a vnější vinutí. Pomocí komutátoru vnitřní pole mění svou polaritu, tím chvilku přitahuje a chvilku odpuzuje. Při přesném načasování dochází k točivému pohybu. Možná nevýhoda je nutnost použití další elektroniky (encoderů) při zjišťování polohy. Používají se, kde je důležitá konstantní rychlost, na kterou jsou nastaveny např. ventilátory, pohánění magnetické hlavy u videorekordéru. [2]

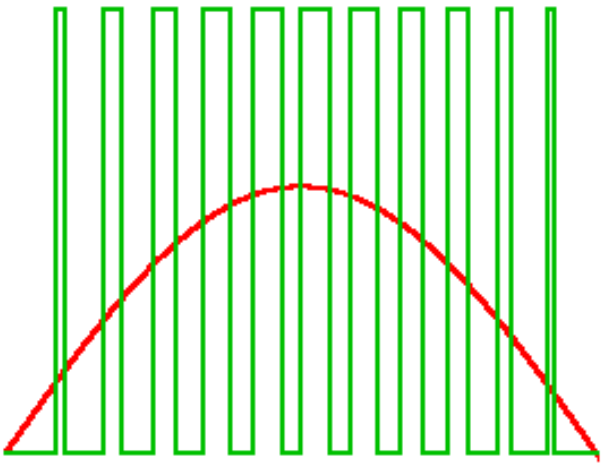
2.2.2 Řízení motorů

ON/OFF
Jedná se o nejjednodušších metodu na řízení motorů, ke které není potřeba žádných složitých externích obvodů. Bohužel její jednoduchost je i její hlavní nevýhoda. Metoda spočívá v připojení konstantního napětí na svorky motoru, polarita nám určuje

směr otáčení. Rychlost otáčení je přímo úměrná velikosti přiloženého napětí. S principu tedy nelze nijak kontrolovat rychlost otáčení motoru. Velký problém je taky v proudových rázech, které při stavu *ON* nutně nastávají. Motor běží jen na stejných otáčkách. Tato metoda lze snadno vylepšit manipulací se vstupním napájením. [2]

Pulzně šířková modulace

Tato metoda se v literatuře nazývá *PWM*, což vychází z anglického *Pulse Width Modulation*. Princip spočívá v generování analogového signálu pomocí diskrétního (obdélníkového) signálu viz Obrázek 2.1. To je důvodem vysoké frekvence zdroje *PWM* a dlouhé setrvačnosti řízeného systému. Šířkou pulzu měníme střední hodnotu generovaného signálu a díky tomu můžeme regulovat rychlost otáčení motoru. Velikou výhodou je přesnější řízení otáček motoru a velice jemná plynulá regulace. V některých případech může být nevýhodou použití speciální elektroniky (např. *I2C PWM* generátor). [2]



Obrázek 2.1: PWM modulace

2.2.3 Hardwarové řešení řízení motorů

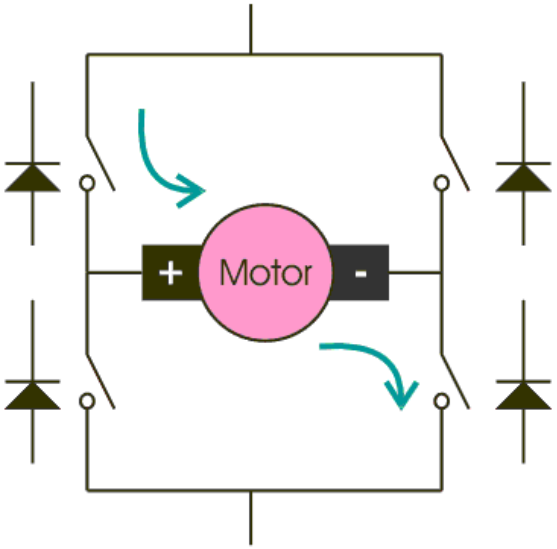
H-můstky

Velice dobrým řešením na ovládání motorů se jeví H-můstek. Umožňuje i s nízko proudovou řídící elektronikou ovládat velice energeticky náročné motory. Směr otáčení motoru je dáno polaritou napětí, které je na motoru. Změnou polarity napětí se změní i směr otáčení. H-můstek je tvořen čtyřmi spínači, kde se otevřou vždy jen dva protilehlé, tím zajistí průchod proudu jen jedním směrem a můžeme daný směr měnit viz Obrázek 2.2. [3]

Speciální zdroje

Toto řešení spočívá v tom, že motory jsou připojeny k vlastnímu zdroji napájení.

Výstup může být velice přesný a vysoce proudově zatížitelný. Nastavení takového zdroje se provádí změnou referenčního napětí (např. I2C DAC nebo PWM) nebo změnou odporu ve zpětné vazbě. Výhoda těchto zdrojů spočívá v robustnosti, výkonu a též přesnosti. Závažnou nevýhodu tohoto řešení je velké množství okolních součástek, což zvyšuje ceny výsledného řešení, zároveň návrh takových zdrojů, velmi obtížný a časově náročný. Z těchto důvodů se nevyužívá v jednoduchých řešeních s malý výkonem.



Obrázek 2.2: H-můstek [3]

2.3 Krokové motory

2.3.1 Obecně

Motor je tvořen permanentním magnetem a dvěma páry nástavců, přičemž každý obsahuje jednu cívku. Motor má přesně stanovené polohy rotoru, díky tomu je při správném řízení zaručen jemný a čistý chod. Pro různé druhy točivých pohybů máme různé metody ovládání motorů. Pro plynulý pohyb by měl stator obsahovat více párů nástavců a tím pádem více poloh rotoru. Motory se využívají, kde je nutná přesnost polohy a proměnná rychlost např. tiskárny, CD-ROM, ... [4]

2.3.2 Řízení motorů

Celé kroky

Při protékajícím proudu cívkou se na každé straně vytvoří opačný pól a rotor se zafixuje. Když do druhé cívky pustíme proud a první cívce proud vypneme rotor se

nám pootočí o jeden krok, abychom postupovali ve stejném směru musíme do první cívky pustit proud v opačném směru a v druhé znovu vypnout, tím se pootočíme o další krok a tak pokračujeme stále dokola. Tím dochází k točivému pohybu mezi definovanými polohami. Tyto polohy se označují jako kroky, od toho tedy krokový motor. Plynulost otáčení a regulace polohy je přímo závislá na počtu kroků. [4]

Mezikroky

Abychom docílili menších kroků (plynulejšího pohybu) na stejném motoru. Postupujeme podobně jako v předchozím případě. Jen s tím rozdílem, že vždy do dvou sousedních cívek necháme téci stejný proud a tím zajistíme, aby rotor byl mezi póly. Tímto způsobem zdvojnásobíme počet poloh rotoru. Hlavním důsledkem je ještě větší plynulost a jemnost pohybu.

Některé typy krokových motorů máji takzvané *půl vinutí*. Tento typ vinutí umožňuje jednodušší ovládání tím, že cívky jsou propojené dohromady. Použitím tohoto principu ovládáme obě cívky naráz jen jedním vodičem přičemž neměníme složitost ovládací elektroniky. [4]

3 Deska V850_M1

Deska byla navržena pro řízení stejnosměrných, krokových a třífázových motorů pomocí procesoru *NEC V850-IG3*. Má dvě rozhraní na nízko odběrové motory, kde se dají zapojit 4 stejnosměrné motory nebo 2 krokové motory. Další tři rozhraní umožňují připojit motory s vyšším odběrem a to buď 4 stejnosměrné, 2 krokové motory a nebo dokonce jeden třífázový. *V850_M1* má čtyři komunikační rozhraní a to *USB*, *RS232*, *UART*, *SPI* a *I2C*. Dále je deska osazena programovatelným obvodem *CPLD* na koncové spínače a H-můstky na ovládání motorů s měrnými odpory na měření proudu a napětí. Číslicová část deska může být napájena pomocí *USB* nebo externím zdrojem. [5]

3.1 Procesor NEC V850-IG3

3.1.1 Obecně

Jedná se o 32 bitový mikrokontrolér o taktu 64 MHz s architekturou *RISC*. Procesor je výhradně určen na řízení motorů a má k tomu i speciální funkce. Velikost paměti je 256kB *ROM* a 12kB *RAM*. Velkou výhodou je jeho osazení na vývojové desce takzvaném *Target boardu*. Na tomto plošném spoji jsou již některé komponenty (např. *LED*, krystal, ad.) zároveň jsou přístupné všechny vývody a programovací konektor. Jeho použitím se výsledný návrh velmi zjednoduší. [6]

3.1.2 Speciální funkce

- A/D převodníky
- Rozhraní pro spojení s PC
- Čítače/časovače
- Sériová rozhraní
- Řadiče přerušení
- DMA řadič
- Jednotka pro řízení motorů

3.1.3 Porty

Mikrokontrolér je vybaven celou řadou osmi a šestnácti bitových portů. K některým z nich lze přistupovat v jednobitovém režimu, jiné sdílí funkce s ostatními periférie mi. Nastavení portů se provádí celou řadou registrů, určuje se typ, pull-up odpory ad.

3.2 Rozhraní

3.2.1 I2C

Inter-Integrated Circuit častěji se v literatuře označuje IIC nebo I2C a jedná se o synchronní *Multi-Master* sériovou sběrnici vyvinutá firmou Philips (dnes NXP). Rozhraní umožňuje obousměrný přenos, který je poloduplexní. Přes sběrnici se posílají osmi bitové rámce. Data můžeme přenášet ve dvou režimech a to režim broadcast nebo adresní.
Adresní režim umožňuje pomocí adresy poslat data jen určitému zařízení. Signálové úrovně jsou kompatibilní s běžně používanou logikou, ovšem mezi na různými napěťovými úrovněmi na jedné sběrnici je nutné vkládat převodníky tzv. Level-Shifters. Protokol je většinou přímo hardwarově podporován.
Na linku se dá připojit až 128 zařízení. Rozhraní využívá tři vodiče *SDA (Serial Data)*, *SCL (Serial Clock)* a *GND*. Proces přenosu dat se uskutečňuje mezi vedoucím (Master) a podřízeným zařízením (Slave). Vedoucí zařízení může vysílat i přijímat, podřízené může přijímat nebo vysílat jen na vyzvání od vedoucího. Jelikož je podporován režim *(Multi-Master)* musí se využít arbitrace. Priorita jednotlivých zařízení je dána jejich adresou.
Rozhraní má tři režimy rychlostí, standardní *S – Standard Mode* 100kb/s, rychlý *F – Fast Mode* 400kb/s a vysoký *Hs – High speed* 3,4Mb/s. Protokol též umožňuje na jedné lince byla připojena zařízení s různou rychlostí rozhraní a to díky možnosti zpožďování hodin na sběrnici. [7]
Desce *V850_M1* využívá toto rozhraní, které je implementováno přímo v použitém procesoru *V850 IG3*. Na námi požadované nastavení (přenosová rychlost, adresa atd.) se využívají speciální registry procesoru. Při komunikaci zadáme adresu podřízeného zařízení, nastavíme bity dle čtení/zápis a po příznaku odpovědi přečteme přijatá data v příslušných registrech a nebo posíláme další data. V našem případě je na rozhraní *I2C* připojena *EEPROM* paměť, která bude obsahovat nastavení procesoru a ovládací program. [5]

3.2.2 RS232

Název vychází z RFC Recommanded Standard 232, které je známe spíše pod názvem *port COM*, poskytující sériový přenos. Jedná se univerzální rozhraní, které je založené na asynchronním přenosu (UART). Tato specifikace využívá jiných napěťových úrovní než TTL a CMOS. Implementace portu je zaručena integrovanými obvody *UART*. Rozhraní se také specifické tím, že využívá signály jiného typu než je *TTL*. Datová přenosová rychlost, která je obvykle implementována dosahuje až 115 200 b/s, ale můžeme se setkat i s vyššími

rychlostmi (460kb/s). Rozhraní se používalo k připojení periférií k počítači např. myš, tiskárna, modem a nebo pro přímé propojení dvou počítačů. Data se dají vysílat a přijímat současně, což toto rozhraní řadí mezi plně duplexní. Rozhraní má několik signálů, většina je potřeba při připojení modemu, ale na jednoduché spojení stačí jen tři. Základní je uzemnění a pro přenos dat jsou využity signály *TXD* pro vysílání a *RXD* pro příjem. Komunikace specifikuje dvě role, *DCE* se vyskytuje modemu a *DTE* koncové zařízení, což může být počítač, tiskárna ... Pro propojení dvou koncových zařízení (např. *PC*) je použito takzvaného kabelu nulového modemu, který kříží vysílací a přijímací signál a tím umožňuje komunikaci. Takového křížení se využívá i mezi integrovanými obvody (např. MCU FTDI). [7]
Toto rozhraní je využito pro komunikaci s nadřazeným systémem. Na desce *V850_M1* je rozhraní ve dvou variantách, které se liší napěťovými úrovněmi. Implementovány jsou jak *TTL* tak i *RS232* a obě jsou vyvedeny na vlastní komunikační port. Použitý procesor má implementován jen 5-ti voltovou a o připojení druhé k procesoru se stará obvod *MAX232* od firmy *MAXIM*, který upravuje úrovně. Při komunikaci lze používat jen jedna. [5]

3.2.3 SPI

Rozhraní *SPI* (Serial Peripheral Interface) v jednodušší variantě též známé pod názvem *Microwire*. Jde o synchronní rozhraní, které umožňuje pomocí tří vodičů samostatně posílat vstupní a výstupní data. Na toto rozhraní lze připojit více zařízení arbitrace mezi zařízeními je dána dalším signálem označovaným Slave select, který se prakticky implementuje na GPIO. V některých případech lze použít i k programování cílových zařízení (In-System Programming, ISP) například pamětí. Na rozdíl od rozhraní *I2C* poskytuje vyšší přenosovou rychlost (až 50Mb/s). Takt přenosové rychlosti může dosahovat až 50 MHz. Navíc lze během každého hodinového impulzu přijímat i vysílat data, což znamená že se jedná o plně duplexní rozhraní. V některých případech je implementován režim slave na mikrokontroléru v takových případech lze vytvořit multimaster sběrnici. Rozhraní má čtyři vodiče, které přenáší synchronizační signál (*SCK*), datový výstup vedoucího zařízení (*MOSI*), datový vstup vedoucího zařízení (*MISO*) a uzemnění (*GND*). Pro výběr podřízeného zařízení se využívá další signalizační vodič (*SS*) nebo (*CS*). Tím vedoucí zařízení zajistí, že komunikuje jen s konkrétním podřízeným zařízením. [7]
Rozhraní na desce se využívá ke komunikaci s nadřazeným systémem, který přes něj bude posílat příkazy na ovládání motorů. Rozhraní je přímo implementováno v použitém procesoru *V850 IG3*.[5]

3.2.4 USB

Též se jedná o univerzální rozhraní, které je nástupcem všech rozhraní pro připojení periférií např. *RS232*, *LPT*, *PS2*. V dnešní době velice rozšířené, zejména kvůli své jednoduchosti pro cílové uživatele. Veškeré komplikace s vývojem jsou přeneseny na designéra a programátora. Jde o sériové rozhraní, které umožňuje připojení až 127 zařízení přes takzvaný *USB hub*. Stejně jako *RS232* se k němu dá připojit myš, klávesnice, tis-

kárna, ... Další jeho nespornou výhodou je napájení připojeného zařízení a to až do 500 mA. Taktéž podporuje technologii *Plug and Play*, která povoluje připojení a odpojení za chodu. Přenosová rychlost rozhraní je opravdu veliká v nejnovější specifikaci *USB 3.0* může dosahovat až 5 Gbit/s. Všechny dřívější specifikace využívají stejný konektor a též jsou i zpětně kompatibilní. Na rozdíl od svých předchůdců (rozhraní) využívá centralizované řízení a tak stačí na všechny zařízení jen jeden hostitelský řadič. [7]
Procesor *V850 IG3* nemá implementované rozhraní *USB*, proto byl využit externí obvod *FTDI FT232BM*. Tento obvod zajišťuje propojení rozhraní *RS232* na procesoru s *USB* na nadřazeném systému. Na počítači se *USB* zobrazuje jako *port COM*, protože použitý obvod rozhraní jen emuluje. Výhoda je, že se pomocí tohoto rozhraní dá napájet číslicová část desky.[5]

3.3 CPLD

Programovatelný logický obvod je použit pro rozšíření funkcí. Může implementovat jednoduchou propojovací matici mezi H-můstky a procesorem nebo složitý koprocesor sloužící k samostatnému řízení motorů. Je umístěn mezi procesor a H-můstky, které se starají o velké motory. Je zde použit kvůli připojení rozdílných typů motorů.
Nezanedbatelná funkce je též při spojení s koncovými spínači. Na koncové spínače se dají připojit různá čidla (např. mikrotlačítko, infračervený senzor ...). Čidla dají signál programovatelnému obvodu, který v základní konfiguraci, zastaví motor a pošle signál procesoru. Zastavení motoru se uskuteční bez účasti procesoru, tím je zajištěna rychlá reakce.

3.3.1 JTAG

Rozhraní *JTAG* (Joint Test Action Group) je synchronní sériové. Jedná se o specifické rozhraní, které není určeno pro přenos dat. Účelem je testování komplexních logických obvodů, programování zařízení nebo trvalé paměti mikrokontroléru. Rozhraní je řízeno jen jedním řadičem, který umožňuje připojit několik testovacích zařízení. Všechny standardní signály rozhraní tvoří port *TAP* (*Test Access Port*), který je využit k propojení testovaného zařízení s řadičem. Přes *TAP* se připojují všechny testovací zařízení do řetězce. Standardizovaný logický formát, tak umožní řadiči nezávislou komunikaci se zařízeními. Úkolem testovacího přístroje je generování testovacích signálů, které zasílá testovací program vytvořený vývojářem zařízení. Výhoda *JTAG* je v tom, že nerozhoduje složitost zařízení, k testování se používají vždy jen čtyři signály. Testovací logika obsahuje čtyři prvky, přístupový port testu (*TAP*, čtyři signály rozhraní), řadič registru *TAP*, registr instrukcí a testovací datové registry. Instrukční a datové registry jsou posuvné spojené paralelně. Řadič *TAP* je synchronní konečný automat.[7]
Deska *V850_M1* využívá rozhraní *JTAG* k programování obvodu *CPLD*, který je využit jako mezičlánek mezi procesorem *V850 IG3* a H-můstky. Jelikož je rozhraní umístěno přímo na desce, tak můžeme daný obvod naprogramovat přímo, bez jakékoli další složité manipulaci. Samozřejmostí je i opakované naprogramování. [5]

3.4 Porty desky

H1, H2, H3, H7, H16
Tyto porty jsou určeny ke komunikaci, ať již s nadřazeným systémem (H1, H2, H3, H16) a nebo k programováni obvodu *CPLD* (H7).

H4
Jedná se o pomocný port, který rozšiřuje desku o další možné funkce. Jsou na něj vyvedeny nepoužité vodiče procesoru (např. připojení senzorů)

H5
K tomuto portu je připojen sedmý port procesoru a rozšiřuje funkčnost desky.

H6
Na tomto portu jsou umístěné koncové spínače, můžeme sem zapojit, jak čidla, tak i spínače, které nám budou určovat krajní polohu.

H8, H9, H10, H11, H12
Motory s nízkým odběrem se můžou připojit k portům H8 a H9, na zbývající tři porty se dají připojit motory s větším odběrem. Další možnost na portech H10, H11 je připojení inkrementálních čidel.

H13, H14, H15
Jedná o napájecí konektory pro jednotlivé části desky ať už se jedná o analogovou, číslicovou a nebo výkonovou část.

3.5 Motory

Deska umožňuje řízení několika typů motoru (*stejnosměrný*, *krokový*, *třífázový*). Dále pak poskytuje ovládání i různě energeticky náročným motorům. Používá k tomu dva typy H–můstků. Pro nízko odběrové *L293D* a pro vysoko odběrové *L298*. Motory mohou mít napájeny, jednotlivě a nebo společné. [5]

Větší motory
Ovládání větších motorů je zajištěno pomocí H-můstků L298, který zvládá větší proudový odběr. Tento obvod je spojen s programovatelným obvodem typ *CPLD*, který bude složit k propojení procesoru, koncových spínačů a H–můstků. Koncové spínače budou sloužit k okamžitému zablokování motorů, bez ohledu na příkazy procesoru, například z důvodu dosažení krajní polohy. U motorů bude též možno měřit odběr proudu pomocí měřících odporů, které budou spojené s A/D převodníkem procesoru. Získané hodnoty lze použít například k určení otáček daného motoru. [5]

Malé motory
Rozhraní pro ovládání menších motorů jsou použity H–můstky L293D, které jsou spojeny přímo s procesorem a ten je může jednoduše ovládat, pomocí modulace *PWM*. [5]

4 Protokol

Řídící jednotka čeká na příjem paketu, který je poté ověřen a případně zpracován. Odpovědi na běžné příkazy (priorita 1 a nižší) jsou generovány do odchozí fronty. Na ostatní zprávy se reaguje okamžitě, tedy nedochází k zařazení do fronty. Na některé pakety nelze reagovat ihned např. *go time, go distance, ad.*, jelikož se odpověď generuje až po jejich dokončení. Po úspěšném vykonání je uložena do odchozí fronty jako u normálních příkazů. Přístup nadřazeného systému ke frontě odpovědí je závislý na použitém rozhraní a to buď synchronní např. *SPI* nebo asynchronní např. *I2C*. Je to dáno tím, zda komunikační linka umožňuje plně duplexní přenos. Průběh komunikace je znázorněn na obrázkách 5.2 a 5.3. Komunikace musí být zahájena inicializačními příkazy, které nastaví *řídící jednotka*. V tomto případě se jedné o procesor *V850E/IG3* a desku *V850_M1*. Musí se nastavit typ motoru a na jakém portu je připojen, dále jeho maximální rychlost, výchozí rychlost a průměr. Na pořadí inicializačních paketů nezáleží. Pro správnou funkčnost je ovšem třeba nastavit všechny připojené resp. používané motory. Při přepojení nebo výměně za jiný typ, je důležité provést novou inicializaci tak, aby ovládací program věděl, jak je může ovládat. Po správném nastavení parametrů už mohou být tyto motory použity, dle jejich vlastností a typu.

4.1 Paket

4.1.1 Návrh a popis paketu

Paket má velikost šest oktetů, kde první oktet je adresa rozhraní, druhý a třetí oktet je horní resp. dolní část 16-ti bitového čísla paketu, čtvrtý oktet je priorita příkazu a operační kód příkazu, pátý a šestý oktet jsou parametry příkazu viz Tabulka 4.1.

Paket					
adresa	číslo paketu H	číslo paketu L	příkaz	parametr 1	parametr 2

Tabulka 4.1: Paket

Adresa, která je v prvním oktetu znázorňuje rozhraní, ze kterého byl paket přijat resp. na které byl poslán viz Tabulka 4.2. Adresace využívá jen 7 horních bitů. Poslední bit je využit pouze u *I2C* pro definici směru komunikace, na ostatních rozhraních je ignorován.

Čtvrtý oktet paketu nazvaný příkaz se dále dělí na dva a šest bitů, horní dva bity určují prioritu příkazu a zbylých šest bitů je operační kód příkazu viz Tabulka 4.3.

rozhraní	adresa
RS232	0x02
USB	0x04
SPI	0x06
I2C	0x08

Tabulka 4.2: Adresa

Příkaz							
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
priorita příkazu		operační kód příkazu					

Tabulka 4.3: Příkaz

Protokol definuje pět priorit příkazů, kterým odpovídá bitové označení viz Tabulka 4.4. Odpověď nemá své bitové označení, jelikož se použije to, co obsahuje paket, na který se má reagovat. Tedy příkaz a jeho odpověď má stejné pořadové číslo i prioritu. Navíc tím ušetříme další bit (a jeho kombinace), který se dá využít na operační kód dalších příkazů.

Priorita příkazu		
	bit 7	bit 6
inicializační	0	0
normální	0	1
urgentní	1	0
error	1	1
odpověď	-	-

Tabulka 4.4: Priority příkazů

4.2 Popis a seznam příkazů

Příkazy jsou rozděleny na tři typy *inicializační*, *normální* a *urgentní*. V kapitolách 4.2.1, 4.2.2 a 4.2.3 jsou jednotlivé příkazy popsány. Parametry, priority a operační kódy příkazů jsou podrobně popsány viz Tabulka A.1. Číslování motoru viz Tabulka 5.13 a typ motoru mají značení viz Tabulka 5.14.

4.2.1 Inicializační příkazy

Type of engine - nastavení typ motoru na určitý port na desce *V850_M1*

Set max speed - nastavení maximální rychlosti[1]

[1]maximální rychlost odpovídá maximálnímu napětí daného motoru

Set default speed - nastavení výchozí rychlosti[2]

Set distance - nastavení vzdálenosti v milimetrech, nastavení je důležité, aby se dal provádět přesně příkaz *Go distance*, u krokového motoru je vzdálenost jeden krok motoru, u stejnosměrného vzdálenost, která motor ujede mezi dvěma pulzy.

4.2.2 Normální příkazy

Acceleration/Decaleration - určuje o kolik se rychlost[3] motoru zvýší resp. sníží

Speed - nastavuje rychlost[4] motoru. Znaménko rychlosti určuje směr otáčení. + pravotočivé - levotočivé. Tato rychlost je použita při aktivaci.

Condition - dotazový příkaz. V odpovědi je uvedena aktuální rychlost a směr otáčení daného motoru.

Run at max speed - motor se rozjede maximální rychlostí, kterou jsme nastavili u daného motoru

Go to the end point[5] - řídící jednotka čeká na signál čidla, poté dojde k zastavení motoru.

Start counter - začne čítání v čítači procesoru, čítač je 16-bitový

Reset counter - hodnota v čítači se vynuluje

Return to zero - motor pojede opačným směrem do té doby, dokud nebude hodnota v čítači nula (hodnota čítače se postupně odčítá)

Go time - motor pojede výchozí rychlostí zadaný čas

Go distance - motor pojede výchozí rychlostí zadanou vzdálenost.

Stop - okamžité zastavení motoru

Disconnect - odpojení motoru od napájení, motor se může dotáčet setrvačností

Change direction of rotation - změna směru otáčení

Turn one step - otočení *krokového* motoru o jeden krok, druhý parametr určuje směr otáčení + pravotočivé - levotočivé

Turn n steps - otočení *krokového* motoru o zadaný počet kroků, znaménko určuje směr otáčení + pravotočivé - levotočivé

[2] výchozí rychlost daného motoru (ideálně $\frac{2}{3}$ maximálního napětí, protože motor by neměl být dlouhodobě plně zatížen)

[3] zpomalení resp. zrychlení se udává v hodnotách -20 až 20, tato hodnota se odečte resp. přičte k rychlosti

[4] je zadávána v bezrozměrných jednotkách, Např. rychlost 127 je maximální rychlost a pomocí přepočtu a *PWM modulace*, bude na motoru maximální napětí, které jste nastavili

[5] příkaz se dá použít jen pro motory 1 až 6

Turn one turn - otočení *krokového* motoru o jednu otáčku druhý parametr určuje směr otáčení + pravotočivé - levotočivé

Turn n turns - otočení *krokového* motoru o zadaný počet otáček, znaménko určuje směr otáčení + pravotočivé - levotočivé

Measure current[6] - měření proudu na vinutí motoru, druhý parametr udává četnost měření a zasílaní zpráv *nadřazenému systému*, periodicky *[0xAA]*, jednorázově *[0x55]* nebo vypnutí zasílání *[0xFF]*

Measure voltage[7] - měření napětí na vinutí motoru, druhý parametr udává četnost měření a zasílaní zpráv *nadřazenému systému*, periodicky *[0xAA]*, jednorázově *[0x55]* nebo vypnutí zasílání *[0xFF]*

Stop all engines - okamžité zastavení motorů

Disconnect all engines - odpojení motorů od napájení, motory se mohou dotáčet setrvačností

4.2.3 Urgentní příkazy

Stop all engines - okamžité zastavení motorů,

Disconnect all engines - odpojení motorů od napájení, motory se mohou dotáčet setrvačností

Delete queue of commands - vymaže frontu příkazů

Delete queue of responses - vymaže frontu odpovědí

Responses - dotazový příkaz k zjištění počtu odpovědí, které čekají ve frontě, následuje zaslání všech čekajících odpovědí

4.3 Popis a seznam odpovědí

Na všechny příkazy nejsou odpovědi, jelikož by zbytečné zatěžovaly sběrnici (zejména *I2C*) viz Tabulka A.2. Odpovědi nemají svou prioritu příkazu, jelikož to není nutné. Využívají prioritu příkazu a číslo paketu, na který odpovídají. Tím *nadřazený systém* pozná nejen na jaký příkaz *řídící jednotka* odpovídá, ale také pozná konkrétní paket.

Condition - druhý parametr bude rychlost motoru, ze které je patrná rychlost a směr otáčení

[6] příkaz se dá použít jen pro motory 1 až 6
[7] příkaz se dá použít jen pro motory 1 až 6

Go to the end point - odpoví motor zastaven

Return to zero - odpoví motor stojí

Go time - odpoví motor stojí

Go distance - odpoví motor stojí

Turn one step - odpoví motor stojí

Turn n steps - odpoví motor stojí

Turn one turn - odpoví motor stojí

Turn n turns - odpoví motor stojí

Measure current - pošle hodnotu proudu v *[mA]*

Measure voltage - pošle hodnotu napětí ve *[V]*

Stop all engine - odpoví motory zastaveny

Disconnect all engine - odpoví motory odpojeny

Stop all engines - odpoví motor zastaveny

Disconnect all engines - odpoví motor odpojeny

Delete queue of commands - fronta vymazána

Delete queue of responses - fronta vymazána

Responses - počet příkazů, dále následují odpovědi čekajíc ve frontě

4.4 Popis a seznam chyb

Protokol definuje celou sadu zpětnovazebních chyb. Tyto chyby mají nejvyšší prioritu v protokolovém zásobníku. Jednotka poskytuje základní chybu "neznámý příkaz", dále pak přiřazuje chybovou odpověď každému implementovanému příkazu viz Tabulka A.3. Touto strukturou umožňuje rychlejší analýzu nadřazenému systému. Operační kód a číslo paketu je stejné jako u příkazu, na který chyba reaguje. Volné pole parametru umožňuje další hierarchizaci chyb pro stejný operační kód.

Type of engine - daný typ motoru neexistuje

Set max speed - rychlost mimo interval

Set default speed - rychlost mimo interval

Set distance - vzdálenost mimo interval

Acceleration/Decaleration - zrychlení resp. zpomalení mimo interval

Speed - nebyla nastavena maximalní rychlost

Run at max speed - nebyla nastavena maximální rychlost

Go to the end point - špatné číslo motoru, tento motor to neumožňuje

Start counter - čítač již čítá

Reset counter - čítač je již v nule

Return to zero - čítač je již v nule

Go time - nebyla zadána výchozí rychlost

Go distance - nebyla nastavena výchozí rychlost nebo vzdálenost

Stop - motor stojí

Disconnect - motor stojí

Change direction of rotation - motor se netočí

Turn one step - motor není připojen

Turn n steps - motor není připojen

Turn one turn - motor není připojen

Turn n turns - motor není připojen

Measure current - špatné číslo motoru, tento motor neumožňuje měření

Measure voltage - špatné číslo motoru, tento motor neumožňuje měření

Stop all engines - motor stojí

Disconnect all engines - motor stojí

Stop all engines - motor stojí

Disconnect all engines - motor stojí

Delete queue of commands - fronta je prázdná

Delete queue of responses - fronta je prázdná

Responses - fronta je prázdná

Unknown command - přijatý příkaz je neznámý

4.5 Komunikace a zpracování příkazů

4.5.1 Zpracování příkazů

Po přijetí paketu od *nadřazeného systému*, se provede analýza zda takový příkaz existuje a zda jsou jeho parametry v pořádku. Pokud během toho zpracování dojde ke zjištění chyby je *nadřazený systém* upozorněn zaslání chybové zprávy. Pokud je příkaz v pořádku je buď obsloužen a nebo zařazen do fronty příkazů. Jakmile se *řídicí jednotka* dostane dostane k tomuto příkazu, je využito stavového automatu k postupnému zpracování paketu. Základní schéma obsluhy komunikace je na obrázku 5.2.
Po zpracování předního příkazu z fronty, se vytvoří odezva. Typ odpovědi je daná na základě provedeného příkazu a je uložena do řady reakcí. Odezvy se posílají *nadřazenému systému* uložením do fronty odpovědí. U rozhraní *I2C*, ale tomu tak není. Reakce se zašlou *nadřazenému systému*, až o ně projeví zájem.

5 Programové vybavení

5.1 Procesor

Nejprve se u procesoru musí nastavit frekvence, na které procesor běží. Základní frekvence procesoru je 8 MHz. Takt procesoru se nastavuje registrem *PCC* (Processor clock control register). Můžeme zde zvolit z hodnot (8MHz, 4MHz, 2MHz a 1MHz). Při potřebě vyšší frekvence je nutné zapnout též závěsník a určit násobek zvolené frekvence. Závěsník se ovládá pomocí registru *PLLCTL* (PLL control register). Kvůli rychlosti zpracování instrukcí byla zvolena nejvyšší možná frekvence (64 MHz) a též je procesor ve stále aktivním režimu. Z důvodů vysoko odběrových periférií není potřeba řešit spotřebu procesoru, a proto nebyl využit Standby mod, který se ovládá registrem *PSC* (Power save control).

5.2 Pulzně šířková modulace

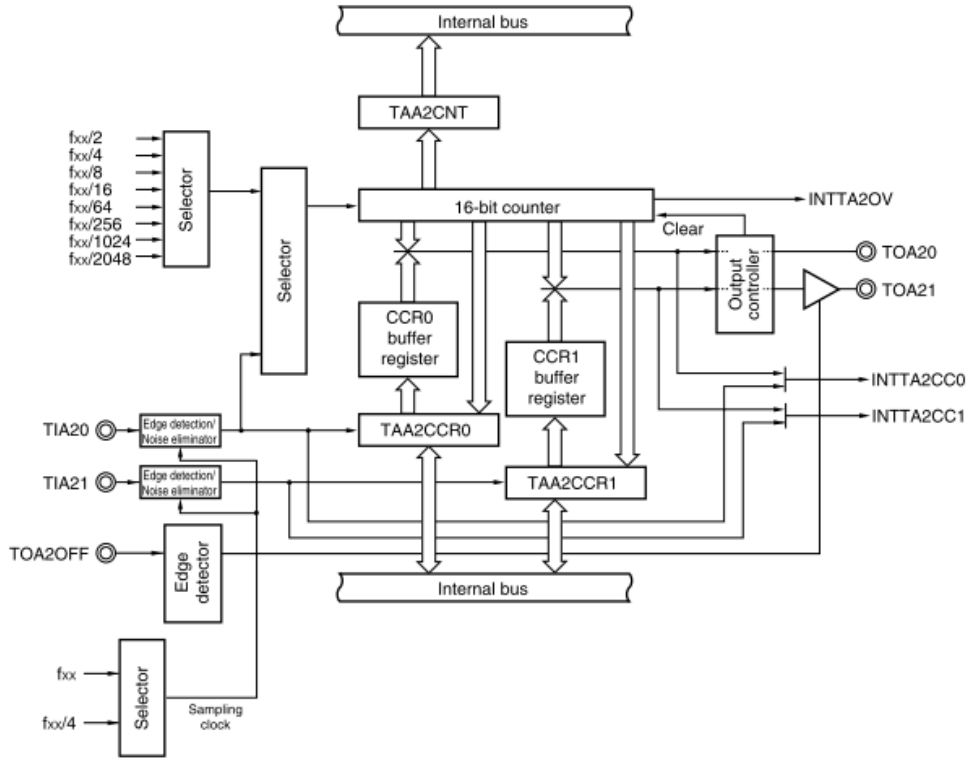
Pulzně šířková modulace je jednu z nejdůležitějších částí ovládacího programu . Bylo k tomu použito několik čítačů *TAB1*, *TAB2*, *TAA2*, *TAA3*, *TAA4*. Kvůli tomu, že většina čítačů je velice podobná a tím i nastavení je shodné, bude zde ukázáno jen nastavení jednoho z čítačů.

5.2.1 Nastavení čítače TAA2

Blokové schéma čítače viz Obrázek 5.1. Můžeme zde vidět děličku frekvence (Selector), komparátor, čítací registr, signály přerušení.

Registr TTA2CTL0

Nejprve je důležité zajistit nastavení frekvence *CLK*, které se provádí pomocí třech bitů (TAA2CKS2, TAA2CKS1, TAA2CKS0) v registru *TTA2CTL0*. Po experimentálním zkoušení se jako ideální frekvence ukázala 1 MHz. Frekvence nemůže být vyšší, než námi zvolená, protože H–můstky se nedokážou zavírat a tím je motor neustále pod plným napětím.
Registr obsahuje ještě bit *TAA2CE*, který spouští čítání. Spuštění čítání se provede až po celkovém nastavení *PWM*. Zvolené hodnoty bitů registru viz Tabulka 5.1.



Obrázek 5.1: Blokové schéma čítače TAA2 [6]

Registr TAA2CTL0							
TAA2CE	0	0	0	0	TAA2CKS2	TAA2CKS1	TAA2CKS0
0/1	0	0	0	0	1	1	1

Tabulka 5.1: Registr TAA2CTL0

Registr TTA2CTL1

Tento registr určuje funkci čítače. Jelikož nepoužijeme externí signál na čítání, je registr *TTA2EEE* nastaven na hodnotu 0. Dále musíme zvolit funkci čítače. Ačkoli je možno přímo nastavit funkci *PWM* není toto ideální řešení, jelikož využívá dva čítače pro generovaní fázově posunutých *PWM*. Proto byla zvolena funkce *Free-running timer mode*, která tuto nevýhodu obchází generováním *PWM* na *GPIO* v přerušeních. Zvolené hodnoty registru viz Tabulka 5.2.

Registr TAA2CTL1							
TAA2SYE	TAA2EST	TAA2EEE	0	0	TAA2MD2	TAA2MD1	TAA2MD0
0	0	0	0	0	1	0	1

Tabulka 5.2: Registr TAA2CTL1

Registr TTA2IOC0

Kontrolní registr, který umožňuje řízení výstupů z časovače. Bity *TAA2OE1* a *TAA2OE0* určují úroveň výstupu (nízká – 0, vysoká – 1). Bity *TAA2OL1* a *TAA2OL0* nastavují povolení (povoleno – 1, zakázáno – 0) výstupního signálu z čítače. Použité hodnoty v bitech viz Tabulka 5.3.

Registr TAA2IOC0							
0	0	0	0	TAA2OE1	TAA2OL1	TAA2OE0	TAA2OL0
0	0	0	0	1	1	1	1

Tabulka 5.3: Registr TAA2IOC0

Registr TTA2IOC1

Poněvadž jsme externí signál nevyužili, tak hodnoty všech bitů jsou 0 viz Tabulka 5.4. Bity určují na jakou hranu čítač bude reagovat.

Registr TAA2IOC1							
0	0	0	0	TAA2IS3	TAA2IS2	TAA2IS1	TAA2IS0
0	0	0	0	0	0	0	0

Tabulka 5.4: Registr TAA2IOC1

Registr TTA2IOC2

Jelikož jsme externí spouštění a čítaní nepoužili, tak všechny hodnoty jsou dány na 0 viz Tabulka 5.5. Bity nám určují na jakou hranu bude čítač reagovat.

Registr TAA2IOC2							
0	0	0	0	TAA2EES1	TAA2ES0	TAA2ETS1	TAA2ETS0
0	0	0	0	0	0	0	0

Tabulka 5.5: Registr TAA2IOC2

Registr TTA2OTP0

Bity *TAA2CCS1* a *TAA2CCS0* dávají na výběr porovnávání (0) a zachytávání (1) hodnoty v registru *TAA2CCR0* s registrem čítače. Pro funkci *PWM* jsem zvolil porovnávání. Bit *TAA2OVF* je v 1, když přeteče registr v čítači. Nastavené hodnoty v bitech viz Tabulka 5.6.

Registr TAA2OTP0							
0	0	TAA2CCS1	TAA2CCS0	0	0	0	TAA2OVF
0	0	0	0	0	0	0	0

Tabulka 5.6: Registr TAA2OTP0

Registr TTA2CCR0

Porovnávací 16–ti bitový registr slouží k nastavování střídy na prvním vývodu z čítače. Mnou nastavená hodnota *0x7FFF* nastavuje střídu 50%:50% viz Tabulka 5.7.

Registr TAA2CCR0			
0	0	0	0
0	0	F	F

Tabulka 5.7: Registr TAA2CCR0

Registr TTA2CCR1

Porovnávací 16–ti bitový registr slouží k nastavování střídy na druhém vývodu z čítače. Mnou nastavená hodnota *0x3FFF* nastavuje střídu 25%:75% viz Tabulka 5.8.

Registr TAA2CCR1			
0	0	0	0
0	F	F	F

Tabulka 5.8: Registr TAA2CCR1

Registr TA2OVIC

Jedná se o vektor přerušení přetečení čítače. Bit *TA2OVIF* (zakázáno – 1, povoleno – 0) napíná vlajku přerušení a bit *TA2OVMK* (zakázaná – 1, povoleno – 0) nastavuje masku. Bity *TA2OVPR2*, *TA2OVPR1* a *TA2OVPR0* určují prioritu (nejvyšší – 0, nejnižší – 7). Nastavené hodnoty viz Tabulka 5.9

Registr TA2OVICP							
TA2OVIF	TA2OVMK	0	0	0	TA2OVPR2	TA2OVPR1	TA2OVPR0
0	0	0	0	0	0	0	0

Tabulka 5.9: Registr TA2OVIC

Registr TA2CCIC0

Jedná se o vektor přerušení při shodném porovnání. Bit *TA2CCICF0* (zakázáno – 1, povoleno – 0) napíná vlajku přerušení a bit *TA2CCMK0* (zakázaná – 1, povoleno – 0) nastavuje masku. Bity *TA2CCPR02*, *TA2CCPR01* a *TA2CCPR00* určují prioritu (nejvyšší – 0, nejnižší – 7). Zvolené hodnoty viz Tabulka 5.10

Registr TAA2CCIC0							
TA2CCIF0	TA2CCMK0	0	0	0	TA2CCPR02	TA2CCPR01	TA2CCPR00
0	0	0	0	0	0	0	0

Tabulka 5.10: Registr TAA2CCIC0

Registr TA2CCIC1

Jedná se o vektor přerušení při shodném porovnání. Bit *TA2CCICF1* (zakázáno – 1, povoleno – 0) napíná vlajku přerušení a bit *TA2CCMK1* (zakázaná – 1, povoleno – 0) nastavuje masku. Bity *TA2CCPR12*, *TA2CCPR11* a *TA2CCPR10* určují prioritu (nejvyšší – 0, nejnižší – 7). Zvolené hodnoty viz Tabulka 5.11

Registr TAA2CCIC1							
TA2CCIF0	TA2CCMK0	0	0	0	TA2CCPR02	TA2CCPR01	TA2CCPR00
0	0	0	0	0	0	0	0

Tabulka 5.11: Registr TAA2CCIC1

Důležité je také nastavení vývodů procesoru, aby byly v režimu OUT viz Tabulka 5.12. Tím můžeme na vývody generovat *PWM*.

Bity registrů	hodnota
P01	0
PM01	0
PMC01	0
PFCE01	0
PFC01	0
PU01	0

Tabulka 5.12: Bity registrů vývodu P01

5.3 Motory

Motory jsou číslovány podle čísla můstků, ale hlavně z důvodu urychlení zpracovávání příkazu viz Tabulka 5.13.

Značení typu motorů je z důvodu, aby podřízený systém věděl, o jaky typ se jedná a podle toho mohl daný motor řídit viz Tabulka 5.14 Každý motor má svoji datovou strukturu, která o něm obsahuje informace

Typ motoru

Maximální rychlost

Standardní rychlost

Rychlost

Vzdálenost

Status

Hodnota status umožňuje zjištění jestli všechny inicializační informace jsou nastaveny. Některé položky mají jen motory číslo $\langle 1 - 6 \rangle$. Těmi údaji jsou

Napětí

Číslování motorů			
Můstek	Port	Číslo motoru	Typ motoru
1	H12	1	stejnosměrný
1	H12	2	stejnosměrný
1	H12	3	krokový, třífázový
2	H10	4	stejnosměrný
2	H11	5	stejnosměrný
2	H10, H11	6	krokový
3	H9	7	stejnosměrný
3	H9	8	stejnosměrný
3	H9	9	krokový
4	H8	10	stejnosměrný
4	H8	11	stejnosměrný
4	H8	12	krokový

Tabulka 5.13: Číslování motorů

Značení motorů	
Značení	Typ motoru
1	stejnosměrný
2	krokový
3	třífázový

Tabulka 5.14: Značení motorů

Proud

Hodnota čítače

Tyto hodnoty jsou cyklicky obnovovány pomocí přerušení čítače. Je to z důvodu urychlení zpracování.

5.4 Datový model

Už z principu věci byl z nabídky modelů (fronta, zásobník, strom, hash tabulka) vybrán model fronty. Jelikož příkazy mají různou priority jedná se o frontu prioritní. Zvolená předloha byla použita pro frontu příkazů a frontu odpovědí, která byla využita jen pro rozhraní *I2C*. Rozhodovalo nad realizací modelu seznamem nebo polem, též bylo nutné řešit implementaci jednotlivých položek mezi strukturami a nedefinovanými objekty.

Pole Velký klad jsou staticky alokované bloky paměti, které jsou přiřazeny při startu programu. Díky překladači lze vynutit celistvost alokované paměti. Ovšem tato výhoda zároveň určuje i nevýhodu, kterou je velká náročnosti na paměť. Jak bylo zmíněno alokace se provádí při startu a paměť je tedy stále blokovaná. Vzhledem ke kruhové implementaci fronty lze přistupovat k již vyřazeným položkám, které jsou při nesprávném implementaci potencionální hrozbou. Náročnost na paměť se může omezit výpočtem nebo též experimentálním měřením, které nám určí optimální počet položek v poli.

Seznam Hlavní výhodou je možnosti dynamického zvětšování paměti pro položky. Nezanedbatelné je též zjednodušení zřetězení (např. přidávaní, odebíraní prvků), v našem případě by se dalo využít pro zařazovaní urgentních položek. Vzhledem k tomu, že uživatel nemá ucelený přehled o vnitřním obsazení paměti, nemůže tedy tak snadno přistupovat uvolněným položkám. Bohužel přednosti dynamického rozšiřování, zda nemůže být využito, protože vyžaduje přiřazování paměti za běhu. Z toho důvodu by bylo potřeba implementovat funkce *malloc* a *free*. Později ušetřená paměť datová by byla vykoupena obsazenou pamětí programovou. což by mělo za následek nižší spotřebu paměti.

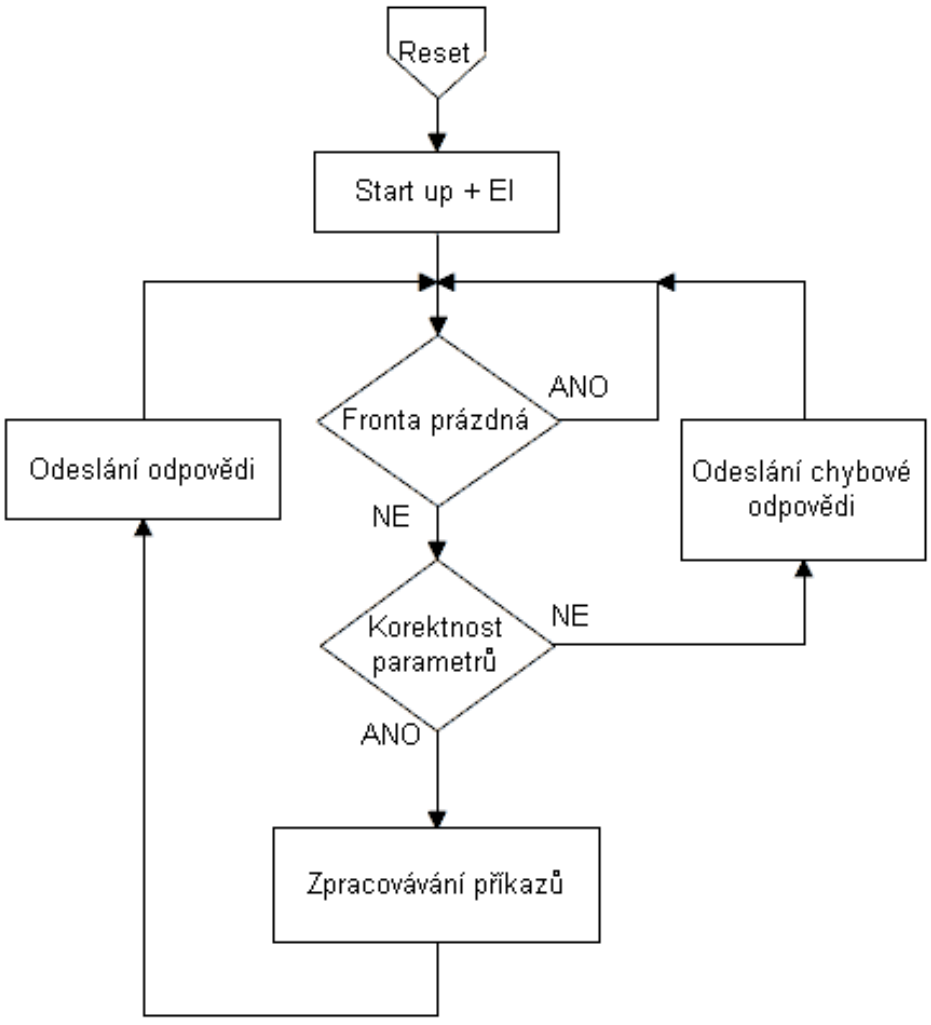
Nedefinované Sice nenáročný na vytvoření, ale o to složitější na údržbu a další rozšiřování. Bohužel u použitých dat je při každém přidání či ubrání parametru vyžadován velký zásah do zdrojového kódu, hlavně se jedná změnu indexace. Jelikož byl protokol navržen co nejuniverzálnější a nejrozšířitelnější vede to k zavržení tohoto modelu.

Struktury Tento model položky je snadno rozšířitelný, není žádný problém přidávat další a další parametry, aniž by se program musel složitě upravovat. Jednou z výhod je možné pojmenování položek, což zpřehledňuje zdrojový kód.

Po posouzení a vyhodnocení kladů a záporů jednotlivých datových struktur, též z důvodů velké náročnosti pro implementaci funkcí, které se starají o správu paměti, byl vybrán model fronty implementovaný polem, jehož prvky jsou tvořeny strukturami.

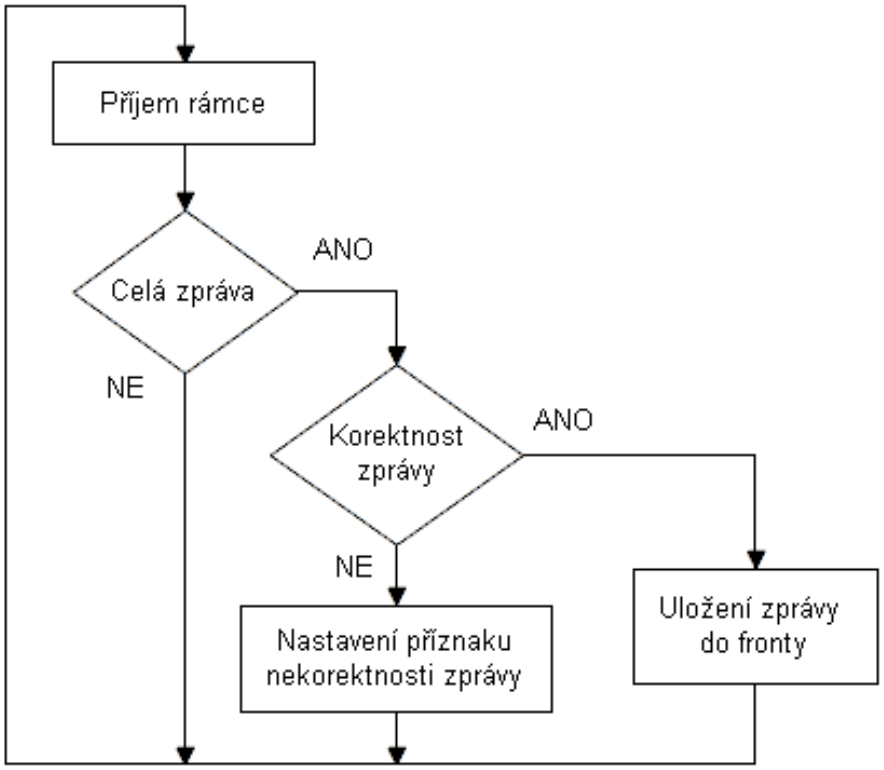
5.5 Průchod příkazu - Běh programu

Při resetu procesoru se spustí obslužný program, který nastaví jednotlivé části procesoru (Start up) a zároveň povolí přerušení. Hlavní smyčka programu kontroluje prázdnost fronty příkazů. O ukládání příkazů do fronty se starají obslužné procedury rozhraní viz dále. Při zjištění špatných parametrů ve zprávě je zaslána chybová odpověď. Jinak se zpráva dál zpracovává dle jejích informacích, které nese. Základní model je znázorněn na obrázku 5.2. Jak již bylo řečeno příkaz může být přijat ze čtyř rozhraní (I2C, USB, UART a SPI). Pří-



Obrázek 5.2: Běh programu

jem jednotlivých paketů potažmo celých rámců je u všech rozhraní obdobný. Byly využity jednotlivá přerušení komunikačních rozhraní, které následně volají obslužnou funkci. Ta je zobrazena na obrázku 5.3. Nejprve se jednotlivé pakety složí v jednu zprávu, která se otestuje a když bude v pořádku uloží se dle priority do fronty příkazů. Při nekorektnosti zprávy se zašle chybová odpověď.



Obrázek 5.3: Obsluha přerušení komunikačních rozhraní

5.6 Popis modulů a funkcí

5.6.1 Modul low_level_initialization.c

Jedná se o modul, který se stará o základní nastavení *CPU*, též čítačů a časovačů, rozhraní, *PWM*.

unsigned char __low_level_init (void);
Deklarace funkce, která nastavuje registry. Tyto registry jsou potřeba pro správný chod jednotlivých částí procesoru.

5.6.2 Modul UART.c

V modulu *UART* řeší nastavení samotného rozhraní. Dále jsou zde vektory přerušení na příjímání, odesílaní a též i chybový příjem. Těmto vektorům se přiřazují funkce, které se dál starají o příjímání nebo odesílaní zpráv.

#pragma vector=INTUA2RE_vector __flat __interrupt void UA2RE_isr(void);

Jedná se definici obsluhy přerušení při chybném přijmu na rozhraní *UART*.

#pragma vector=INTUA2R_vector __flat __interrupt void UA2R_isr(void);

Definice obsluhy přerušení při přijmu na rozhraní.

#pragma vector= INTUA2T_vector __flat __interrupt void UA2T_isr(void);

Definice obsluhy přerušení při odesílání na rozhraní.

void UART_registr_setting();

Specifické nastavování rozhraní.

void UART_transmission(UINT data);

Stará se o příjem dat na rozhraní.

UCHAR UART_reception();

Stará se o zasílání dat na rozhraní.

void UART_close();

Ukončuje práci s rozhraním.

5.6.3 Modul I2C.c

Modul, který nastavuje a spravuje rozhraní *I2C*. Obsahuje obslužné metody přerušení.

#pragma vector=INTIIC_vector __flat __interrupt void IIC_isr(void);

Definice obsluhy přerušení, která obstarává příjem a odesílání zpráv.

void set_port_i2c();

Funkce, která nastavuje registry ovládající *I2C*.

void set_clock_i2c(void);

Ovládání hodin rozhraní.

void set_local_address(unsigned char address);

Nastavení lokální adresy.

void set_cond_setting();

Nastavení přerušení a vlajek přerušení.

void set_control_i2c();

Řídí počet přerušení a kdy mají nastat.

void stop_i2c();

Vygenerování impulzu stop.

void start_i2c();

Vygenerování impulzu start.

5.6.4 Modul Engines.c

Vytvoření struktur motorů a jejich ovládání (zapisování, mazání hodnot).

5.6.5 Modul Commands.c

Realizace jednotlivých příkazů, ovládání motorů.

5.6.6 Modul TAA2.c

Nastavení a spouštění čítače. Obsluhuje a řídí *PWM*. Nastavuje frekvenci a periodu. Podobně jsou nastaveny i ostatní čítače.

#pragma vector=INTTA2OV_vector __flat __interrupt void TA2OV_isr(void);

Obsluha přerušení, která se spouští při přetečení čítače.

#pragma vector=INTTA2CC0_vector __flat __interrupt void TA2CC0_isr(void);

Obsluha přerušení, která se spustí při shodě hodnot v registru *TAA2CC0* a hodnoty v čítači.

#pragma vector=INTTA2CC1_vector __flat __interrupt void TA2CC1_isr(void);

Obsluha přerušení, která se spustí při shodě hodnot v registru *TAA2CC1* a hodnoty v čítači.

void taa2_start_pwm();

Zapnutí čítání.

void taa2_set_freq_clock(int CSK2, int CSK1, int CSK0);

Nastavení frekvence čítače.

void taa2_set_compare_register_TAA2CRR0(int value);

Umožní nastavit do registru *TAA2CRR0* hodnotu, která určuje periodu *modulace*.

void taa2_set_compare_register_TAA2CRR1(int value);

Umožní nastavit do registru *TAA2CRR1* hodnotu, která určuje periodu *modulace*.

5.6.7 Modul main.c

Spouští program, ovládá základní povely a přijímá zprávy.

6 Závěr

Úkolem práce bylo nastudovat metody řízení stejnosměrných a krovových motorů, též navržení komunikačního protokolu a v neposlední řadě programového vybavení, kde se implementovaly získané znalosti na desku *V850_M1*.

Komunikační protokol byl navržen, co nejuniverzálnější, aby ho bylo možné jednoduše upravovat a rozšiřovat. To umožňuje snadné využití protokolu i na jiných deskách. I když je napsán jen pro čtyři rozhraní, tak stačí přidat další adresu a rozhraní může být použito. Též i příkazy, lze snadno rozšiřovat. Chybová hlášení se můžou dále specifikovat dle nastavené hierarchie, kdy se pomocí parametrů detailněji popíše nastalá chyba.

Programové vybavení desky bylo naprogramováno zcela dle zadání, byly využity všechny možnosti, které umožňuje navržený protokol a použitá deska, ať už zasílaní chybových hlášek, čí využití všech rozhraní desky. Bohužel z důvodu neosazení části desky, nebylo možno všechny navržené funkce odzkoušet.

Pro odzkoušení řízení motorů bylo použito pásové vozítko *ROBO EXPLORER*, které bylo propojeno s bakalářskou prací mého kolegy, který se zabýval detekcí překážek a v neposlední řadě též spojen pomocí technologie Bluetooth a mobilním telefonem, který dokázal toto vozítko řídit.

Možným vylepšením programového vybavení by mohla být portace nějakého operačního systému reálného času, tím by se procesoru umožnilo využití paralelních procesů. Též protokol může být rozšířen o další rozhraní, příkazy, chybová hlášení. To vše je umožněno univerzálností firmwaru i navrženého protokolu. Možnosti užití vidím například v řízení modelů (aut, letadel, lodí ad.), či robotických ruk.

Symboly a zkratky

CLK	–	Clock
CMOS	–	Complementary Metal–Oxide–Semiconductor
COM	–	Serial port
CPLD	–	Complex programmable logic device
CS	–	Chip Select
DAC	–	Digital-to-analog converter
DC	–	Direct current
DCE	–	Data communications equipment
DMA	–	Direct Memory Access
DTE	–	Data terminal equipment
EEPROM	–	Electrically Erasable Programmable Read-Only Memory
GND	–	Ground (electricity)
GPIO	–	General Purpose Input/Output
I2C	–	Inter-Integrated Circuit
JTAG	–	Joint test action group
LED	–	Light-emitting diode
MCU	–	Microcontroller
MISO	–	Master input, Slave output
MOSI	–	Master output, Slave input
PLLCTL	–	PPL control registr
PPC	–	Processor clock control registr
PWM	–	Pulse-width modulation
RFC	–	Request For Comment
RISC	–	Reduced instruction set computing
RS232	–	Serial link
RXD	–	Receive data
SCL	–	Synchronous Clock
SDA	–	Synchronous Data
SPI	–	Serial Peripheral Interface
SS	–	Slave Select
TAP	–	Test access port
TTL	–	Ttransistor-transistor-logic
TXD	–	Transmit data

Literatura

[1] RNDr. Ivo Novák, Ph.D., *Emotor.cz*
[online] http://www.emotor.cz/
Ostravská univerzita v Ostravě
20.5.2002

[2] UZIMEX PRAHA, spol. s r.o., *Malé stejnosměrné motory MAXON*
[online] http://www.uzimex.cz/soubory/20070103_maxon_serial.pdf
18.7.2002

[3] Chuck McManis, *Krokové motory a jejich řízení*
[online] http://www.mcmanis.com/chuck/robotics/projects/esc2/hbridge_spiking.html
20.5.2002

[4] Ing. Pavel Rydlo, *Voltage Spikes in FET based H-bridges*
[online] http://www.mti.tul.cz/files/ats/krok2.pdf
Liberec 2000

[5] Karel Dudáček, *Jednotka pro řízení stejnosměrných a krokových motorů*
[online] https://portal.zcu.cz/wps/PA_StagPortletsJSR168/
KvalifPraceDownloadServlet?typ=1&adipidno=31453
Západočeská univerzita v Plzni
11.6.2009

[6] NEC Electronics Corporation, *User's manual V850E/IG3*
Japonsko
Březen 2010

[7] Michael Gook, *Hardwarová rozhraní: Průvodce programátora*
Computer Press, ISBN: 80-251-1019-2,
Czech Republic 2006

Pozn.: Pokud u jednotlivých citací chybí nějaký údaj, nepodařilo se mi jej zjistit.

A Rozsáhlé tabulky

Seznam příkazů					
Type of commands	Names of commands	Bits [hex]	1. Parameters	2. Parameters	Type of engines
initialization	type of engine	0x01	number of engine ⟨1;12⟩	type of engine ⟨1;2⟩	all
initialization	set max speed	0x02	number of engine ⟨1;12⟩	max of speed[V] ⟨1;12⟩	all
initialization	set default speed	0x03	number of engine ⟨1;12⟩	default speed[V] ⟨1;12⟩	all
initialization	set distance	0x04	number of engine ⟨1;12⟩	distance[cm] ⟨1;255⟩	all
normal	acceleration/decaleration	0x33	number of engine ⟨1;12⟩	accelerace/decelerace ⟨-20;20⟩	all
normal	speed	0x34	number of engine ⟨1;12⟩	speed ⟨-127;127⟩	all
normal	condition	0x35	number of engine ⟨1;12⟩	-	all
normal	run at max speed	0x36	number of engine ⟨1;12⟩	-	all
normal	go to the end point	0x37	number of engine ⟨1;6⟩	-	all
normal	start counter	0x38	number of engine ⟨4;6⟩	-	all
normal	reset counter	0x39	number of engine ⟨4;6⟩	-	all
normal	return to zero	0x3A	number of engine ⟨4;6⟩	-	all
normal	go time	0x3B	number of engine ⟨1;12⟩	time[s]	all
normal	go distance	0x3C	number of engine ⟨1;12⟩	distance[cm]	all
normal	stop	0x3D	number of engine ⟨1;12⟩	-	all
normal	disconnect	0x3E	number of engine ⟨1;12⟩	-	all
normal	change direction of rotation	0x3F	number of engine ⟨1;12⟩	-	all
normal	turn one step	0x40	number of engine ⟨3;6;9;12⟩	left[0x80] or right[0x00]	stepper
normal	turn n steps	0x41	number of engine ⟨3;6;9;12⟩	number of steps ⟨-127;127⟩	stepper
normal	turn one turn	0x42	number of engine ⟨3;6;9;12⟩	left[0x80] or right[0x00]	stepper
normal	turn n turns	0x43	number of engine ⟨3;6;9;12⟩	number of turns ⟨-127;127⟩	stepper
normal	measure current	0x44	number of engine ⟨1;6⟩	periodic[0xAA], once[0x55] or disable[0xFF]	all
normal	measure voltage	0x45	number of engine ⟨1;6⟩	periodic[0xAA], once[0x55] or disable[0xFF]	all
normal	stop all engine	0x46	-	-	all
normal	disconnect all engine	0x47	-	-	all
urgent	stop all engines	0x81	-	-	all
urgent	disconnect all engines	0x82	-	-	all
urgent	delete queue of commands	0x83	-	-	all
urgent	delete queue of responses	0x84	-	-	all
urgent	responses	0x85	-	-	all

Tabulka A.1: Seznam příkazů

Seznam odpovědí					
Type of commands	Names of commands	Bits [hex]	1. Parameters	2. Parameters	Type of engines
normal	condition	0x35	number of engine ⟨1;12⟩	speed	all
normal	go to the end point	0x37	number of engine ⟨1;6⟩	-	all
normal	return to zero	0x3A	number of engine ⟨1;6⟩	-	all
normal	go time	0x3B	number of engine ⟨1;12⟩	-	all
normal	go distance	0x3D	number of engine ⟨1;12⟩	-	all
normal	turn one step	0x40	number of engine ⟨3;6;9;12⟩	-	stepper
normal	turn n steps	0x41	number of engine ⟨3;6;9;12⟩	-	stepper
normal	turn one turn	0x42	number of engine ⟨3;6;9;12⟩	-	stepper
normal	turn n turns	0x43	number of engine ⟨3;6;9;12⟩	-	stepper
normal	measure current	0x44	number of engine ⟨1;6⟩	mA	all
normal	measure voltage	0x45	number of engine ⟨1;6⟩	V	all
normal	stop all engine	0x46	-	-	all
normal	disconnect all engine	0x47	-	-	all
urgent	stop all engines	0x81	-	-	all
urgent	disconnect all engines	0x82	-	-	all
urgent	delete queue of commands	0x83	-	-	all
urgent	delete queue of responses	0x84	-	-	all
urgent	responses	0x85	number of responses	-	all

Tabulka A.2: Seznam odpovědí

Seznam chybových zpráv					
Type of commands	Names of commands	Bits [hex]	1. Parameters	2. Parameters	Type of engines
error	type of engine	0xC1	number of engine	Type of engine	all
error	set max speed	0xC2	number of engine	max of speed[V]	all
error	set default speed	0xC3	number of engine	default speed	all
error	set diameter	0xC4	number of engine	diameter	all
error	acceleration/decaleration	0xC5	number of engine	accelerace/decelerace	all
error	speed	0xC6	number of engine	speed	all
error	condition	0xC7	number of engine	-	all
error	run at max speed	0xC8	number of engine	-	all
error	go to the end point	0xC9	number of engine	-	all
error	start counter	0xCA	number of engine	-	all
error	reset counter	0xCB	number of engine	-	all
error	return to zero	0xCC	number of engine	-	all
error	go time	0xCD	number of engine	time	all
error	go distance	0xCE	number of engine	distance	all
error	stop	0xCF	number of engine	-	all
error	disconnect	0xD0	number of engine	-	all
error	change direction of rotation	0xD1	number of engine	-	all
error	turn to one step	0xD2	number of engine	-	Stepper
error	turn to n step	0xD3	number of engine	number of steps	Stepper
error	turn to one turn	0xD4	number of engine	-	Stepper
error	turn to n turn	0xD5	number of engine	number of turns	Stepper
error	measure current	0xD6	number of engine	periodic or once	all
error	measure voltage	0xD7	number of engine	periodic or once	all
error	stop all engine	0xD8	-	-	all
error	disconnect	0xD9	-	-	all
error	stop all engine	0xDA	-	-	all
error	disconnect all engine	0xDB	-	-	all
error	delete queue of commands	0xDC	-	-	all
error	delete queue of responses	0xDD	-	-	all
error	responses	0xDF	-	-	all
error	unknown command	0xE0	-	-	all

Tabulka A.3: Seznam chybových zpráv