TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Implementace komunikačního protokolu Faulhaber RS-232

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Miroslav Hlávka**
Vedoucí práce: Ing. Leoš Beran, Ph.D.

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Implementation of communication protocol Faulhaber RS-232

Bachelor thesis

Study programme: B2646 – Information technology
Study branch: 1802R007 – Information technology

Author: **Miroslav Hlávka**
Supervisor: Ing. Leoš Beran, Ph.D.

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Miroslav Hlávka**
Osobní číslo:	**M12000131**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Implementace komunikačního protokolu Faulhaber RS232**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s komunikačním protokolem společnosti Faulhaber.
2. Navrhněte funkční bloky pro ovládání jednotek Faulhaber v jazyku ST.
3. K funkčním blokům vytvořte nápovědu dle zvyklostí.
4. Návrh a realizaci ověřte na dostupném hardwaru MCBL 3006 S RS.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

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

Seznam odborné literatury:

[1] **AUTOMATION, B&R. Controls - training text. Austria: [s.n.], 2008. 205 s.**

[2] **FAULHABER. Communication and Function Manual [online]. 2012 [cit. 2012-10-05]. Dostupné z: http://www.faulhaber.com/uploadpk/EN_7000_05029.pdf**

[3] **JOHN, Kharl-Heinz; TIEGELKAMP, Michael. IEC 61131-3 Programming Industrial Automation Systems : Concepts and Programming Languages, Requirements for Programming Systems, Decision - Making Aids. 2nd Edition. NewYork : Springer, 2010. 390 s. ISBN 978-3-642-12015-2**

Vedoucí bakalářské práce: **Ing. Leoš Beran, Ph.D.**
Ústav mechatroniky a technické informatiky

Konzultant bakalářské práce: **Ing. Martin Diblík, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2014**

Termín odevzdání bakalářské práce: **15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2014

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 14. 5. 2015

Podpis:

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářské práce Ing. Leošovi Beranovi, Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce.

Abstrakt

Práce obsahuje seznámení se s komunikačním protokolem společnosti Faulhaber, který je využíván pro řízení pohonných systémů. Navržení softwarových funkčních bloků pro zařízení PLC, určených pro ovládání jednotek Faulhaber, pomocí zmiňovaného protokolu, v jazyce Strukturovaný text. Dále k těmto funkčním blokům je vytvořena nápověda, která bude sloužit uživatelům k snazší práci s těmito funkčními bloky.

Hlavní přínos práce spočívá ve vytvoření knihovny pro PLC zařízení, která umožní ovládat jednotky Faulhaber za pomoci komunikačního rozhraní RS-232.

Klíčová slova:

protokol Faulhaber, PLC, softwárové funkční bloky, ovládání pohonu Faulhaber, nápověda

Abstract

This work includes familiarization with Faulhaber's communication protocol, which is used for drive systems. Design function blocks for PLC devices, used for controlling Faulhaber's units in programming language Structured text. Create for these function blocks Help. Help will serve for easier understanding of using library and each individual function blocks in it.

The main contribution consist of create a library for PLC devices which allow control Faulhaber's units with communication interface RS-232.

Klíčová slova:

protocol Faulhaber, PLC, software function blocks, controlling Faulhaber's drive systems, help

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

B&R	Bernecker a Rainer, firma vyrábějící PLC
PLC	Programovatelný logický automat
FB	Funkční blok
ST	Programovací jazyk Strukturovaný text
RS-232	Sériové komunikační rozhraní
ANSI C	Standardizovaný programovací jazyk C
WYSIWYG	What you see is what you get, co vidíš, to dostaneš
PDF	Portable Document Format, přenositelný formát dokumentu
CHM	Microsoft Compiled HTML Help
HTML	HyperText Markup Language, značkovací jazyk pro tvorbu hypertextových dokumentů

Úvod

Důvodem pro výběr tohoto tématu byla možnost prohloubit si své znalosti ve vývojovém prostředí Automation Studio a vyzkoušet si práci s pohonnou jednotkou firmy Faulhaber.

Hlavním cílem této bakalářské práce bylo nejprve se seznámit s komunikačním protokolem, který využívá společnost Faulhaber pro řízení svých pohonných systémů. Navrhnout knihovnu funkčních bloků v jazyce Strukturovaný text, která umožní ovládat pohonné jednotky firmy Faulhaber po sériovém komunikačním rozhraní RS–232. K této knihovně a jejím funkčním blokům poté vytvořit nápovědu dle zvyklostí pro usnadnění a zpřehlednění práce s vytvořenou knihovnou. Dále návrh a realizaci ověřit na dostupném hardwaru MCBL 3006 S RS. Pro vytvoření samotné knihovny a jejích funkčních bloků jsem použil vývojové prostředí Automation Studio od firmy B&R. V knihovně jsem vytvořil pět základních funkčních bloků pro ovládání jednotek Faulhaber. K této knihovně jsem vytvořil nápovědu za pomocí programu HelpNDoc 4. K dosažení tohoto cíle jsem se musel seznámit s programovatelnými logickými automaty, vývojovým prostředím Automation Studio, tvorbou knihoven a funkčních bloků, komunikačním protokolem a pohonnými jednotkami firmy Faulhaber, obecnou tvorbou nápověd a s programem HelpNDoc pro jejich tvorbu.

1 Použité technologie

První kapitola seznámí čtenáře s použitými technologiemi. Je zde seznámení s vývojovým prostředím Automation Studio, programovacím jazykem Strukturovaný text, knihovnou DVFrame, programem ARsim a HelpNDoc.

1.1 Automation Studio

Automation Studio je vývojové prostředí určené pro zařízení od firmy B&R. Je velmi robustní a má mnoho funkcí. Podporuje psaní programů v nejrůznějších programovacích jazycích například ANSI C, Sekvenční funkční diagram, Ladder Diagram, Strukturovaný text a mnoho dalších. Automation Studio obsahuje 3 základní pohledy. Prvním je konfigurace, umožňuje přidávat a nastavovat jednotlivé konfigurace pro dané PLC. Druhý, fyzický pohled, slouží pro přidávání virtuálního hardwaru jako je klávesnice a display. Logický pohled umožňuje spravovat jednotlivé soubory, knihovny a vizualizaci. Automation Studio obsahuje nástroj Build pro sestavení projektu, Transfer pro odeslání sestaveného projektu do PLC, funkci Monitor pro sledování stavu proměnných za běhu PLC, nástroj Debugger pro ladění programu a mnoho dalších.

1.2 Strukturovaný text

Tato práce byla vytvářena v programovacím jazyce Strukturovaný text. Je to vyšší programovací jazyk vycházející z jazyku Pascal. Syntaxe jazyka ST je dána povolenými výrazy a příkazy. Výraz se skládá z operátorů a operandů. Příkazy jsou pro přiřazení, volání funkcí, výběr a iteraci.

1.3 Knihovna DVFrame

Knihovna DVFrame je soubor funkčních bloků sloužící k výměně dat s externími zařízeními jako jsou tiskárny, monitory a podobně. Výměna dat se prování přijímáním a odesíláním takzvaných rámců. Z této knihovny jsou použity funkce xopen, gbuf, write, read, rbuf a close.

1.3.1 Xopen

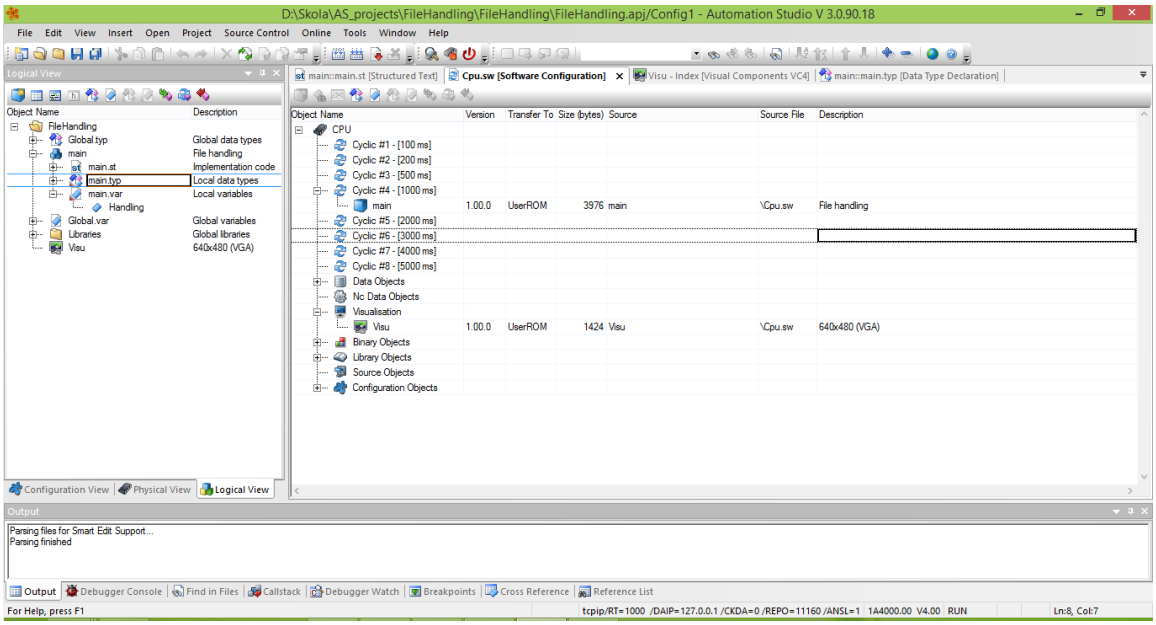
Funkční blok Xopen slouží pro inicializaci a otevření komunikace s externím zařízením. Funkční blok má čtyři vstupní parametry. Parametr enable, který je datového typu BOOL a slouží pro zapnutí samotného funkčního bloku. Parametr device je určený pro výběr rozhraní. Parametr mode nastavuje typ, přenosovou rychlost, paritu, datové bity a stop bity zvoleného rozhraní. Posledním vstupním parametrem je ukazatel na konfigurační strukturu, která definuje parametry odesílaného rámce. Tento funkční blok vrací parametr ident, kterým se jednoznačně identifikuje dané komunikační rozhraní. Parametr ident je vstupní parametr ve všech následujících funkčních blocích knihovny DVFrame.

1.3.2 Gbuf

Funkční blok Gbuf je určen pro inicializaci vyrovnávací paměti, do které se poté nahrají data k odeslání. Má dva parametry, enable a ident. Parametr enable je datového typu BOOL a slouží pro samotné vykonání funkčního bloku. Parametr ident pro identifikování komunikačního rozhraní.

1.3.3 Write

Funkční blok Write umožňuje odeslat data z vyrovnávací paměti. Funkční blok má tři parametry. Parametr enable pro vykonání samotného funkčního bloku. Parametr ident pro identifikaci komunikačního zařízeni a posledním vstupní parametrem je buffer, který určuje adresu vyrovnávací paměti, ve které jsou uložená data k odeslání.



Obrázek 1.1: Prostředí Automation Studio

```
WHILE Var1 <> 0 DO
    Var2 := Var2 * 2;
    Var1 := Var1 - 1;
END_WHILE;
```

Obrázek 1.2: While cuklus v ST

1.3.4 Read

Funkční blok Read čte odpovědi odeslané z externího zařízení a ukládá je do vyrovnávací paměti. Funkční blok má dva vstupní parametry. Parametr enable pro provedení samotného funkčního bloku a parametr ident pro jednoznačné identifikování komunikačního rozraní.

1.3.5 Rbuf

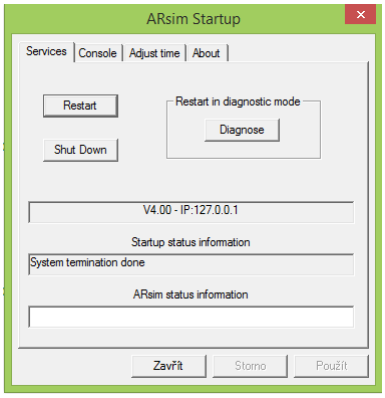
Po přečtení odpovědi je potřeba vymazat vyrovnávací paměť. K tomu slouží funkční blok Rbuf. Funkční blok má čtyři vstupní parametry. Parametr enable pro vykonání samotné funkce. Parametr ident pro identifikování komunikačního rozhraní. Parametr buffer, který ukazuje na vyrovnávací paměť určenou k smazání a parametr bufflng, který určuje velikost vyrovnávací paměti.

1.3.6 Close

Funkční blok Close slouží pro ukončení komunikace s externím zařízením. Funkční blok má dva vstupní parametry. Parametr enable, datového typu BOOL, pro vykonání funkčního bloku a parametr ident pro identifikaci komunikačního rozhraní.

1.4 ARsim

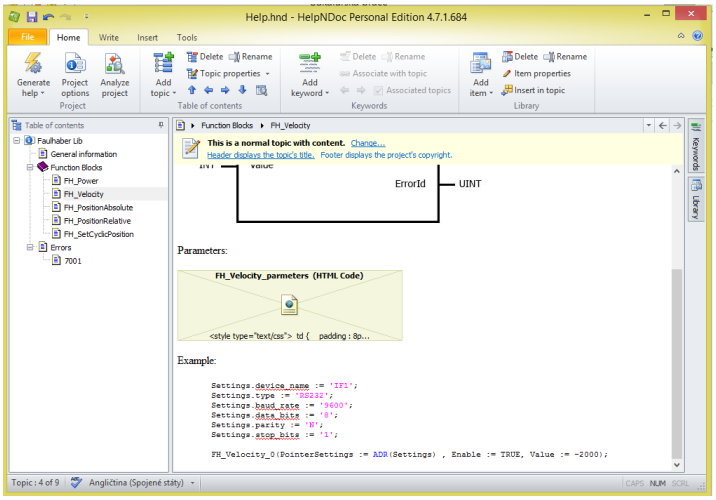
ARsim je program, který je součástí Automation Studia. Slouží pro virtualizaci programovatelného logického automatu na počítači . Díky tomuto programu není nutné mít při vývoji projektu u sebe fyzicky PLC zařízení. Program nám umožňuje toto virtuální PLC v určité míře ovládat. Můžeme například restartovat PLC, vypnout PLC, přečíst si hlášky, které PLC vypisuje na display, spustit PLC v diagnostickém režimu a podobně.



Obrázek 1.3: Program ARsim

1.5 HelpNDoc

HelpNDoc je program na vytváření HTML nápověd. Je založený na WYSIWYG a tudíž se podobá textovým editorům, jako jsou například Microsoft Word nebo OpenOffice Writer. Výslednou nápovědu lze poté uložit ve formátech HTML, PDF, CHM a mnoha dalších.



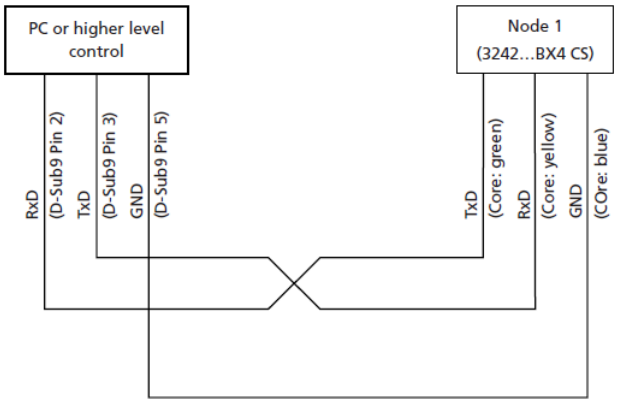
Obrázek 1.4: Program HelpNDoc4

2 Příprava pracoviště

Tato kapitola obsahuje popis a schéma zapojení pohonu MCS3268, popis komunikačního protokolu Faulhaber, postup vytváření a nastavení projektu a testování komunikace s pohonem za pomoci programu Motion Manager.

2.1 Zapojení motoru MCS3268

Z motoru MCS3268 je veden jeden sériový kabel. Dva vodiče tohoto kabelu jsou přivedeny do napájecího zdroje Power Supply PS1042 od firmy B&R. Modrý, který slouží jako uzemnění a růžový, který je určen pro napájecí napětí. Další dva vodiče jsou připojené ke konektoru RS-232. První vodič je zelené barvy a slouží pro přenos dat. Druhý vodič je žluté barvy a je určen pro příjem dat. Na RS-232 konektor je připojená USB redukce, která je poté připojená do počítače, ve kterém virtuální PLC komunikuje s motorem.



Obrázek 2.1: Schéma zapojení motoru MCS3268

2.2 Komunikace protokolu Faulhaber

Komunikační protokol firmy Faulhaber umožňuje komunikovat přes sériové rozhraní RS-232 s řídícími jednotkami Faulhaber za pomoci ASCII příkazů. Jednotlivé příkazy se odesílají v rámcích.

Obrázek 2.2: Motion Control System 3268 od firmy Faulhaber

Příkazový rámec je tvořen dvěma povinnými a dvěma nepovinnými částmi. První část příkazového rámce je určená pro nepovinný Node Number, který umožňuje řídit více zařízení na jednom rozhraní. Za ním následuje povinná část Command, která je tvořena ASCII řetězcem a je určena pro samotný příkaz. Jako další je nepovinná část s názvem Argument, která je potřeba jen u některých příkazů, například nastavení otáček a podobně. Poslední část rámce tvoří netisknutelný ASCII znak Carry Return s decimálním kódem 13. Rámec s odpovědí je tvořen třemi částmi. První část tvoří odpověď ve formě ASCII řetězce. Za ní následuje netisknutelný ASCII znak Carry Return a Line Feed.

2.3 Založení a nastavení projektu

Před samotným vytvářením knihovny bylo potřeba nejprve založit a správně nastavit projekt ve vývojovém prostředí Automation Studia. Vytváření projektů v Automation Studiu je velmi intuitivní. Pro vytvoření projektu klikneme v horním panelu na File - New Project. Otevřel se průvodce pro vytváření projektu. Vyplníme název projektu, jeho cestu, zaškrtneme možnost používání virtuálního PLC pomocí Runtime ARSim a dokončíme vytváření projektu tlačítkem Finish.

Po vytvoření projektu bylo potřeba nastavit sériový komunikační port, na kterém budeme komunikovat s pohonnou jednotkou společnosti Faulhaber. Přepnutím do fyzického pohledu, pravým kliknutím na komunikační port IF1 a zvolením možnosti Configuration se dostaneme do nastavení sériového portu, který budeme využívat při komunikaci. Typ zařízení jsme zvolili RS232, Bauld rate na hodnotu 9600, parametr Parity jsme nastavili na hodnotu None a vše ostatní jsme ponechali v původním nastavení.

2.4 Testování komunikace s pohonnou jednotkou

Nejjednodušší způsob ověření správnosti zapojení a vyzkoušení jednoduché komunikace s pohonnou jednotkou MCS3268 bylo použití softwaru Motion Manager od firmy Faulhaber, který umožňuje navázat spojení s připojenou pohonnou jednotkou a posílat příkazové rámce. Tím, že program identifikoval samotnou pohonnou jednotku jsme si ověřili, že zapojení motoru, podle technického manuálu, je správné.

Dalším krokem bylo vyzkoušet si jednotlivé příkazy jako zapnutí, nastavení otáček, nastavení pozice a podobně. Po osvojení těchto příkazů a jejich odpovědí jsme mohli přejít k vytvoření vlastní komunikace ve vývojovém prostředí Automation studia

3 Tvorba knihovny

V této kapitole se pojednává o vytváření knihovny, její datové struktuře o jednotlivých funkčních blocích a jejich vytváření. Dále o způsobu exportování knihoven ve vývojovém prostředí Automation Studio.

3.1 Založení knihovny

Po vyzkoušení jednoduché komunikace, vytvoření samotného projektu a nastavení potřebné konfigurace pro komunikaci bylo potřeba vytvořit samotnou knihovnu, která bude obsahovat funkční bloky pro ovládání pohonných jednotek Faulhaber. V námi vytvořeném projektu ve vývojovém prostředí Automation studia se přepneme do logického pohledu. V tomto pohledu poté pravým kliknutím vyvoláme menu, ve kterém zvolíme možnost Add object. Zobrazí se vyskakovací okno, kde z kategorie vybereme možnost Library a poté zvolíme New Library. Poté je nutné nastavit název knihovny a jazyk knihovny. Knihovnu jsme pojmenovali FHLib a jazyk knihovny IEC Library. Dokončíme průvodce kliknutím na tlačítko Next a přiřazením nového objektu k aktivnímu CPU.

3.2 Datová struktura

Při vytváření nové knihovny, kterou jsem nazval FHLib, jsme zaškrtli možnost vytvořit soubor pro deklaraci datových struktur. V Logickém pohledu jsme si rozbalili vytvořenou knihovnu FHLib, která v sobě obsahuje tři soubory, FHLib.fun, FHLib.var a FHLib.typ. Datová struktura se vytváří v souboru s příponou .typ.

Otevřením tohoto souboru se v hlavním okně programu zobrazí tabulka se čtyřmi sloupci:název, typ, reference a popis. Hodnoty ve sloupci název reprezentují názvy datových typů používaných v proměnných, které jsou poté používány v celém kódu knihovny. Sloupec typ určuje datový typ proměnné jako je například boolean, integer anebo složité datové typy jako jsou funkční bloky nebo vlastní struktury. Sloupec popis slouží pro vložení vlastního komentáře k danému datovému typu. Tento komentář nebude nikde v kódu vidět a slouží pouze pro zpřehlednění definice datových typů. Veškeré datové typy, které v programu budeme později potřebovat, jsme se snažili logicky rozdělit a zapouzdřit do vlastních datových struktur. Rozhodli jsme

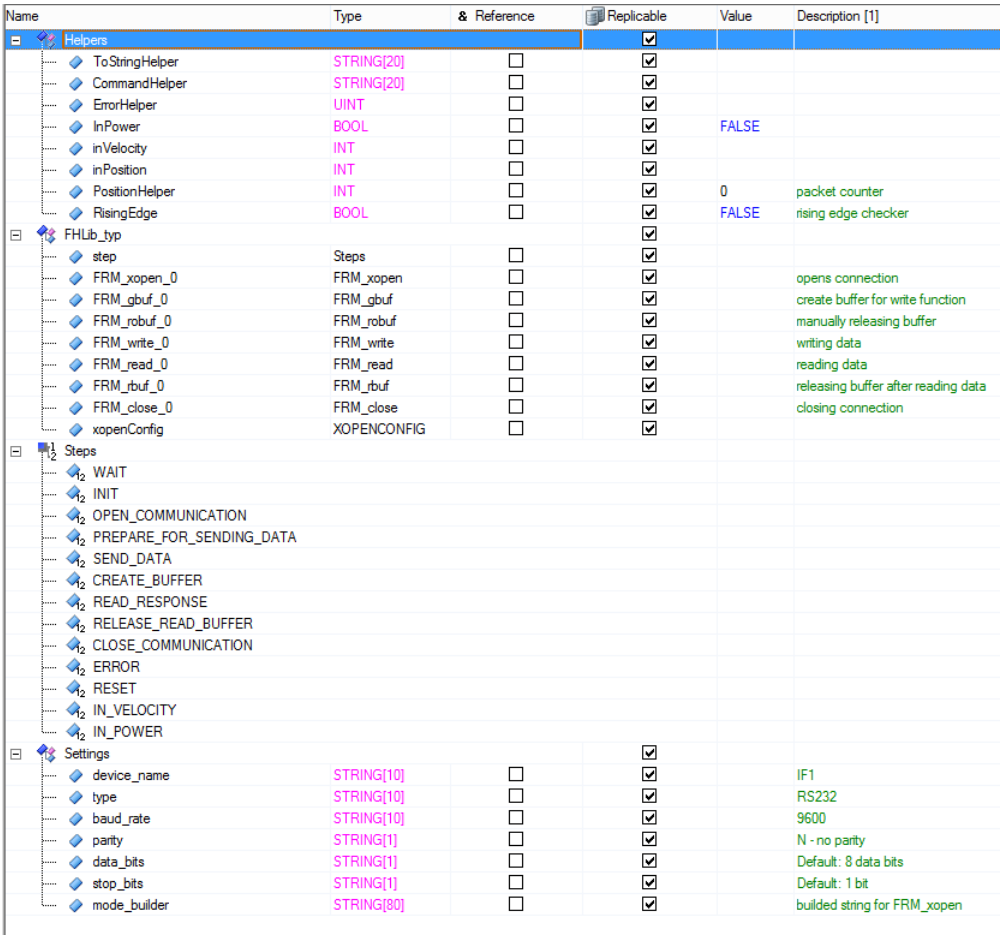
se datové typy rozdělit do čtyř základních struktur, FHLib_typ, Settings, Steps a Helpers.

Vytváření datových struktur jsme prováděli kliknutím na prázdný řádek pravým tlačítkem myši a zvolením Add Structure Type. Jako první jsme si vytvořili hlavní strukturu s názvem FHLib_typ, která bude obsahovat veškeré potřebné funkční bloky pro komunikaci přes rozhraní RS-232 z knihovny DVFrame. V této struktuře jsme poté vytvořili členy FRM_xopen_0 s datovým typem FRM_xopen pro inicializaci komunikace, FRM_gbuf_0 s datovým typem FRM_gbuf pro vytvoření vyrovnávací paměti, FRM_write_0 s datovým typem FRM_write pro odesílání rámců, FRM_robuf_0 s datovým typem FRM_robuf pro uvolňování vyrovnávací paměti po odeslání rámce, FRM_read_0 s datovým typem FRM_read pro čtení odpovědi, FRM_rbuf_0 s datovým typem FRM_rbuf pro uvolnění vyrovnávací paměti po přečtení přijmutích dat, xopenConfig s datovým typem XOPENCONFIG pro nastavení rozšiřujících parametrů pro komunikaci, posledním členem struktury FHLib_typ je step s datovým typem Steps, který určuje v jakém stavu je automat zrovna nachází.

Následně jsme vytvořili výčtový typ s názvem Steps, který obsahuje jedenáct členů. Je určen pro rozdělení programu každého funkčního bloku do jednotlivých stavů stavového automatu. Prvním a základním stavem je WAIT, který má za úkol čekat na povel k vykonání funkčního bloku od uživatele. Následujícím stavem je INIT, který má za úkol shromáždit veškeré vstupní parametry od uživatele a připravit je pro zahájení komunikace. Dalším stavem je OPEN_COMMUNICATION, který otevře komunikaci. Následuje stav CREATE_BUFFER pro vytvoření vyrovnávací paměti určené pro odesílání dat. Za tímto stavem následuje stav PREPARE_FOR_SENDING_DATA, který má za úkol naformátovat data k odeslání a uložit je do vyrovnávací paměti. Následujícím stavem je stav SEND_DATA, ve kterém se odešle rámec. Dalším stavem je READ_RESPONCE, který se stará o čtení odpovědi. Následuje stav RELEASE_READ_BUFFER, který uvolňuje vyrovnávací paměť určenou pro čtení odpovědi. Dalším stavem je CLOSE_COMMUNICATION, který má za úkol uzavřít komunikaci. Předposledním stavem je stav ERROR, který má za úkol informovat uživatele o vzniku chyby a pozastavit svojí činnost. Posledním stavem ve výčtovém typu Steps je RESET, který má za úkol vynulovat nastavení v případě, že se funkční blok dostane do stavu ERROR.

Další datovou strukturu jsme nazvali Setting a slouží pro nastavení komunikace. Veškeré členy této struktury jsou datového typu STRING. Prvním členem je device_name, který slouží pro zvolení názvu sériového portu. Dalším členem je type, který slouží pro zvolení typu rozhraní, v našem případě RS232. Následuje parametr baud_rate, pro nastavení šířky pásma. Za ním se nachází člen parity, data_bits, stop_bits a mode_builder, který má za úkol sestavit tyto parametry do jednoho řetězce.

Poslední datovou strukturou je struktura Helpers, která obsahuje pomocné proměnné používané v jednotlivých funkčních blocích. Prvními členy jsou ToStringHelper a CommandHelper s datovými typy STRING, které jsou používány při sestavování příkazů k odeslání. Následuje člen ErrorHelper, který má datový typ UINT, který se používá nastane-li ve funkčním bloku chyba. Dále následují členy inPower, inVelocity a inPosition, které slouží pro uložení hodnot, ve kterých se motor nachází. Následuje PositionHelper, který je datového typu INT a počítá kolik paketů zbývá ještě odeslat před dokončením komunikace. Posledním členem datové struktury Helpers je člen RisingEdge, která hlídá, zda se nezměnila náběžná hrana na parametru Enable u jednotlivých funkčních bloků.



Name	Type	& Reference	Replicable	Value	Description [1]
Helpers			☑		
ToStringHelper	STRING[20]	☐	☑		
CommandHelper	STRING[20]	☐	☑		
ErrorHelper	UINT	☐	☑		
InPower	BOOL	☐	☑	FALSE	
inVelocity	INT	☐	☑		
inPosition	INT	☐	☑		
PositionHelper	INT	☐	☑	0	packet counter
RisingEdge	BOOL	☐	☑	FALSE	rising edge checker
FHLib_typ			☑		
step	Steps	☐	☑		
FRM_xopen_0	FRM_xopen	☐	☑		opens connection
FRM_gbuf_0	FRM_gbuf	☐	☑		create buffer for write function
FRM_robuf_0	FRM_robuf	☐	☑		manually releasing buffer
FRM_write_0	FRM_write	☐	☑		writing data
FRM_read_0	FRM_read	☐	☑		reading data
FRM_rbuf_0	FRM_rbuf	☐	☑		releasing buffer after reading data
FRM_close_0	FRM_close	☐	☑		closing connection
xopenConfig	XOPENCONFIG	☐	☑		
Steps					
WAIT					
INIT					
OPEN_COMMUNICATION					
PREPARE_FOR_SENDING_DATA					
SEND_DATA					
CREATE_BUFFER					
READ_RESPONSE					
RELEASE_READ_BUFFER					
CLOSE_COMMUNICATION					
ERROR					
RESET					
IN_VELOCITY					
IN_POWER					
Settings			☑		
device_name	STRING[10]	☐	☑		IF1
type	STRING[10]	☐	☑		RS232
baud_rate	STRING[10]	☐	☑		9600
parity	STRING[1]	☐	☑		N - no parity
data_bits	STRING[1]	☐	☑		Default: 8 data bits
stop_bits	STRING[1]	☐	☑		Default: 1 bit
mode_builder	STRING[80]	☐	☑		builded string for FRM_xopen

Obrázek 3.1: Datová struktura knihovny

3.3 Vytváření funkčních bloků

Po vytvoření samotné knihovny musíme vytvořit jednotlivé funkční bloky. Ty vytvoříme v logickém pohledu pravým kliknutím na knihovnu a zvolením možnosti Add Object. Otevře se dialogové okno, kde z kategorie Function Block zvolíme jedinou dostupnou možnost a tou je New Function or Function Block. Zadáme název

funkčního bloku, zvolíme programovací jazyk Strukturovaný text a tlačítkem Next pokračujeme dále v průvodci. V následujícím dialogu zmáčkneme tlačítko Add a tím přidáme nový parametr do funkčního bloku. Každý parametr má svůj datový typ a rozsah. Rozsah určuje, zda je parametr vstupní, výstupní nebo interní. Rozhodli jsme se pro knihovny vytvořit pět základních funkcí pro ovládání pohonných jednotek. Funkci Power, Velocity, PositionAbsolute, PositionRelative a SetCyclicPosition.

3.4 Funkční bloky

S každým vytvořením funkčního bloku se nám do souboru FHLib.fun přidal samotný funkční blok s jeho parametry a vytvořil se nám další soubor s jeho názvem a příponou jazyka Strukturovaný text. Každý takovýto soubor začíná příkazem FUNCTION_BLOCK, který deklaruje, že se jedná o funkční blok. Pro větvení programu jsem použil příkaz CASE, který větví program podle zadané podmínky, který je uveden hned za klíčovým slovem CASE. Tuto podmínku poté porovnáváme s jednotlivými stavy, pokud je shoda, tak se vykoná část kódu v tomto stavu. Jako podmínku pro větvení ve všech funkčních blocích jsme zvolili proměnou step, která je zapouzdřená v datové struktuře FHLib. Do tohoto přepínače jsme umístili veškeré stavy výčtového typu Steps, tedy WAIT, INIT, OPEN_COMMUNICATION, PREPARE_FOR_SENDING_DATA, SEND_DATA, CREATE_BUFFER, READ_RESPONSE, RELEASE_READ_BUFFER, CLOSE_COMMUNICATION, ERROR a RESET. Pokud se tedy proměnná step bude rovnat jednomu z těchto stavů, vykoná se příslušný kód pro tento stav.

Name	Type	& Reference	Scope	Constant	Retain	Replicable	R...	Value	Description [1]
FH_Power									TODO: Enable Drive
Enable	BOOL		VAR_INPUT						
PointerSettings	UDINT		VAR_INPUT						Pointer to structure of settings
Response	STRING[20]		VAR_OUTPUT						
Done	BOOL		VAR_OUTPUT						
Error	BOOL		VAR_OUTPUT						
ErrorID	UINT		VAR_OUTPUT					0	
Settings	Settings		VAR						
FHLib	FHLib_typ		VAR						
Internals	Helpers		VAR						
FH_Velocity									TODO: Sets velocity
Enable	BOOL		VAR_INPUT						
PointerSettings	UDINT		VAR_INPUT						Pointer to structure of settings
Value	INT		VAR_INPUT						
Response	STRING[20]		VAR_OUTPUT						
Done	BOOL		VAR_OUTPUT						
Error	BOOL		VAR_OUTPUT						
ErrorID	UINT		VAR_OUTPUT						
Settings	Settings		VAR						
FHLib	FHLib_typ		VAR						
Internals	Helpers		VAR						
FH_PositionAbsolute									TODO: Set Absolute position
Enable	BOOL		VAR_INPUT						
PointerSettings	UDINT		VAR_INPUT						Pointer to structure of settings
Acceleration	UINT		VAR_INPUT						Value between 0 and 30 000
Deceleration	UINT		VAR_INPUT						Value between 0 and 30 000
MaximumSpeed	UINT		VAR_INPUT						Value between 0 and 30 000
Value	INT		VAR_INPUT						Value of position
Response	STRING[20]		VAR_OUTPUT						
Done	BOOL		VAR_OUTPUT						
Busy	BOOL		VAR_OUTPUT						
Error	BOOL		VAR_OUTPUT						
ErrorID	UINT		VAR_OUTPUT						
Settings	Settings		VAR						
FHLib	FHLib_typ		VAR						
Internals	Helpers		VAR						
FH_PositionRelative									TODO: Set relative position
Enable	BOOL		VAR_INPUT						
PointerSettings	UDINT		VAR_INPUT						Pointer to structure of settings
Acceleration	UINT		VAR_INPUT						Value between 0 and 30 000
Deceleration	UINT		VAR_INPUT						Value between 0 and 30 000
MaximumSpeed	UINT		VAR_INPUT						Value between 0 and 30 000
Value	INT		VAR_INPUT						Value of position
Response	STRING[20]		VAR_OUTPUT						
Done	BOOL		VAR_OUTPUT						
Busy	BOOL		VAR_OUTPUT						
Error	BOOL		VAR_OUTPUT						
ErrorID	UINT		VAR_OUTPUT						

Obrázek 3.2: Datová struktura funkčních bloků

3.5 Export knihovny

Po zkompletování knihovny poté bude potřeba samotnou knihovnu vyexportovat do binární podoby tak, jak je zvykem u běžných knihoven. Nejprve však musíme přidat závislosti na použité knihovny. Přidání závislostí se provádí pravým kliknutím na knihovnu v Logickém pohledu, zvolíme nabídku Properties. Otevře se dialogové okno a v záložce Dependencies musíme přidat knihovny, které jsme použili při tvorbě knihovny FHLib. Těmi jsou knihovny dvframe, aastring a standard. Výslednou knihovnu lze exportovat označením knihovny v Logickém pohledu a vybráním možnosti Export Library v horní záložce File. Následně se otevře dialogové okno, kde zvolíme cestu pro uložení knihovny.

4 Princip fungování FB v knihovně FHLib

V této kapitole je podrobně rozepsáno obecné fungování funkčního bloku v knihovně FHLib. Jsou zde uvedené jednotlivé stavy, do kterých se funkční blok může dostat, jejich popis a do jakých následujících stavů může přejít.

4.0.1 WAIT

V tomto stavu se čeká, dokud uživatel nedá pokyn k zapnutí funkčního bloku pomocí vstupního parametru Enable. V takovém případě se uloží do proměnné steps hodnota INIT a tím se přejde do tohoto stavu.

4.0.2 INIT

Tento stav slouží pro přípravu, před otevřením samotné komunikace a inicializací výstupních parametrů. Výstupní parametr Responce se nastaví na prázdnou hodnotu a výstupní parametr Done se nastaví na False. Tím zajistíme správnou inicializaci samotného funkčního bloku. Dále zkopírujeme data z vstupní struktury PointerSettings do interní proměnné Settings. Za pomoci této proměnné poté sestavíme nastavení pro komunikaci v podobě řetězce do proměnné mode_builder. V tomto nastavení je obsažen typ rozhraní, šířka pásma, parita a stop bit. Poté provedení těchto kroků se nastaví proměnná step na hodnotu OPEN_COMMUNICATION.

4.0.3 OPEN_COMMUNICATION

Stav OPEN_COMMUNICATION slouží pro navázání spojení s připojenou pohonnou jednotkou za použití funkčních bloků xopen a xopenConfig z knihovny DVFrame. Funkčnímu bloku xopen nastavíme parametr device z interní proměnné Settings.name a do parametru mode nahrajeme adresu řetězce z proměnné mode_builder, kterou jsme si připravili v kroku INIT. Dále pomocí funkčního bloku xopenConfig nastavíme idle time na hodnotu 10, delimc na hodnotu 10, tx_cnt na hodnotu 3, rx_cnt na hodnotu 3, tx_len na hodnotu 256, rx_len také na 256 a zavoláme samotný funkční blok xopen. Funkční blok xopen vrací ve výstupním parametru status číslo, které označuje stav provádění samotného funkčního bloku. Pokud je hodnota status rovna nule, nastaví se stav CREATE_BUFFER. Pokud je hodnota status 65535 funkční blok je zaneprázdněn a musíme ho zavolat znovu. Jakmile je hodnota status jiná, nastala chyba, nastaví se do struktury Internals chybové číslo a přejde se do stavu ERROR.

4.0.4 CREATE_BUFFER

Tento stav má za úkol vytvořit vyrovnávací paměť, kde se budou ukládat data určená pro odeslání do pohonné jednoty. Využívá funkční blok gbuf z knihovny DVFrame. Vstupním parametrem tohoto funkčního bloku je ident, který slouží pro identifikaci zařízení. Ident získáme z výstupního parametru funkčního bloku xopen. Funkční blok gbuf vrací ve výstupním parametru status číslo, které označuje stav provádění samotného funkčního bloku. Pokud je hodnota status rovna nule, nastaví se stav PREPARE_FOR_SENDING_DATA. Pokud je hodnota status 65535 funkční blok je zaneprázdněn a musíme ho zavolat znovu. Jakmile je hodnota status jiná, nastala chyba, nastaví se do struktury Internals chybové číslo a přejde se do stavu ERROR.

4.0.5 PREPARE_FOR_SENDING_DATA

Následující stav slouží pro připravení dat a vytvoření odesílacích rámců. Každý funkční blok se zde liší, a proto bude obsah tohoto stavu popsán dále v podkapitolách jednotlivých funkčních bloků.

4.0.6 SEND_DATA

Stav SEND_DATA využívá funkčního bloku Write z knihovny DVFrame. Nastaví se vstupní parametr ident, předá se odkaz na vyrovnávací paměť s uloženými daty a nastaví se délka dat. Poté se zavolá samotný funkční blok write. Funkční blok write vrací ve výstupním parametru status číslo, které označuje stav provádění samotného funkčního bloku. Pokud je hodnota status rovna nule, nastaví se stav READ_RESPONSE. Pokud je hodnota status 65535 funkční blok je zaneprázdněn a musíme ho zavolat znovu. Jakmile je hodnota status jiná, nastala chyba, nastaví se do struktury Internals chybové číslo a přejde se do stavu ERROR.

4.0.7 READ_RESPONSE

Stav READ_RESPONSE má za úkol přečíst odpověď z pohonné jednotky. Využívá k tomu funkční blok read z knihovny DVFrame. Nastavíme vstupní parametr ident a zavoláme samotný funkční blok read. Po úspěšném přečtení dat nakopírujeme přečtená data do proměnné Response příkazem memcpy. Pokud odpověď z pohonné jednotky byla v pořádku, přejde se do stavu RELEASE_READ_BUFFER.

4.0.8 RELEASE_READ_BUFFER

Tento stav slouží pro uvolnění vyrovnávací paměti určené ke čtení. Je použit funkční blok rbuf z knihovny DVFrame. Funkčnímu bloku nastavíme ident, odkaz na vyrovnávací paměť a její délku. Pokud se uvolnění vyrovnávací paměti provede v pořádku a jsou odeslané veškeré rámce, přejde se do stavu CLOSE_COMMUNICATION. Pokud se neodeslali veškeré potřebné rámce, přejde se do stavu CREATE_BUFFER. V případě chyby se přejde do stavu ERROR.

4.0.9 CLOSE_COMMUNICATION

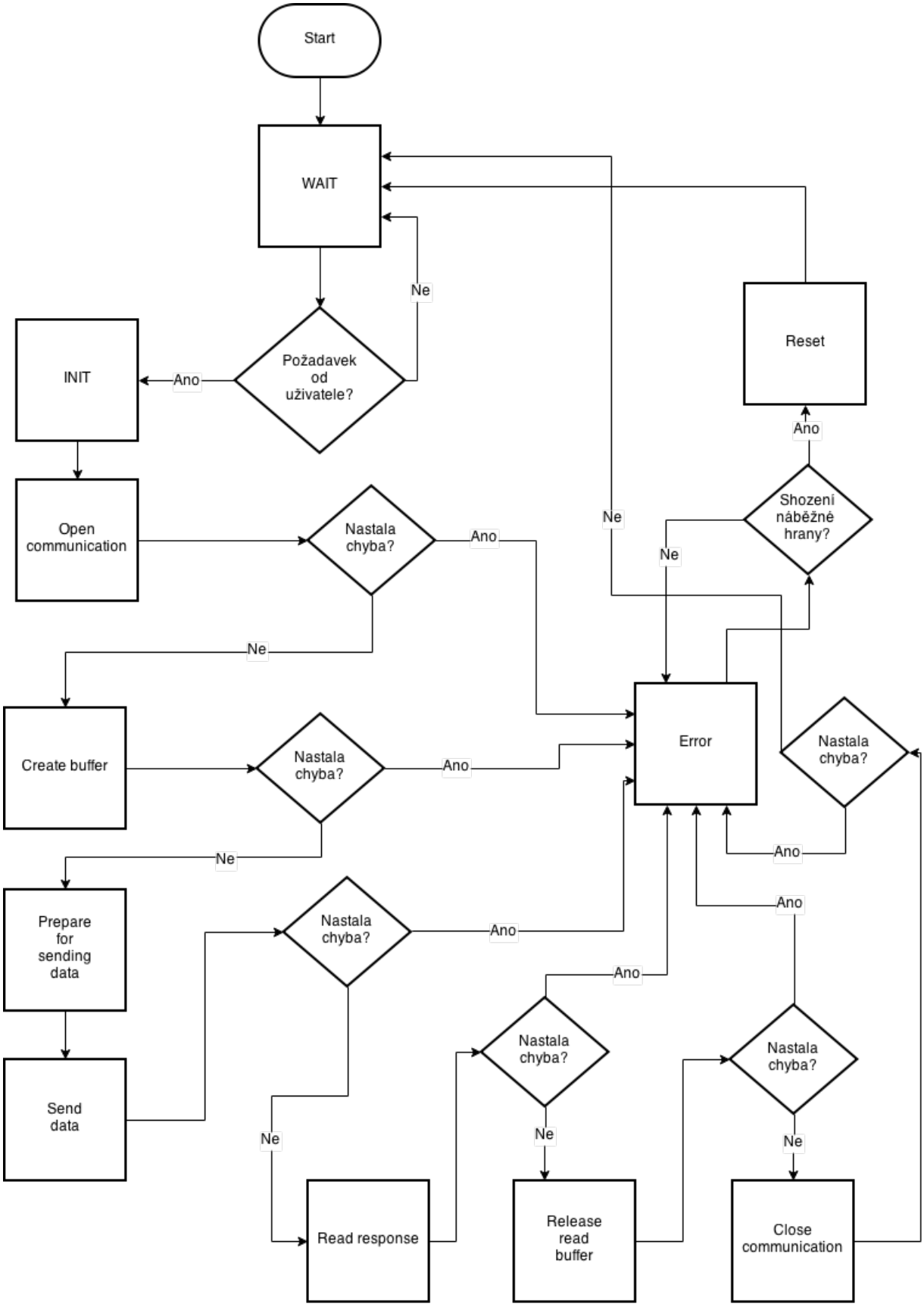
Stav CLOSE_COMMUNCATION má za úkol uzavřít spojení s pohonnou jednotkou za použití funkčního bloku close z knihovny DVFrame. Jako vstupní parametr postačí předat ident. Zavoláme funkční blok. Pokud uzavření spojení proběhlo v pořádku, nastavíme výstupní parametr Done funkčního bloku na hodnotu TRUE a přejdeme do stavu WAIT. V opačném případě přejdeme do stavu ERROR.

4.0.10 ERROR

V tomto stavu funkčního bloku FH_ PositionAbsolute se nastaví výstupní parametr Error na hodnotu TRUE a ErrorID na hodnotu příslušné chyby. Funkční blok přetrvá v tomto stavu do té doby, než uživatel nenastaví sestupnou hranu na vstupním parametru Enable. Pokud tak učiní, přejde se do stavu RESET.

4.0.11 RESET

Stav RESET má za úkol nastavit veškeré proměnné na výchozí hodnoty a zotavit funkční blok z chybového stavu. Nastaví se hodnoty struktury Settings do výchozích hodnot. Výstupní parametr Response se nastaví na prázdný řetězec. Výstupnímu parametru ErrorID se nastaví hodnota 0, Error a Done se nastaví na hodnotu FALSE. Poté se přejde do stavu WAIT.



Obrázek 4.1: Vývojový diagram funkčního bloku knihovny FHLib

5 Jednotlivé funkční bloky

Tato kapitola obsahuje popis činnosti jednotlivých funkčních bloků, jejich parametry a rozdíly mezi nimi. Parametry jsou uvedeny pomocí tabulky.

5.1 Funkční blok FH_Power

Funkční blok FH_Power je první a nejzákladnější funkční blok z celé knihovny FHLib. Zpřístupňuje ovládání pohonné jednotky. Funkční blok FH_Power je prvním blokem, který musí uživatel zavolat. Dokud je funkční blok aktivní lze řídit pohonnou jednotku ostatními funkčními bloky z knihovny FHLib.

5.1.1 Parametry

Parametry funkčního blok FH_Power jsou zobrazeny v tabulce 5.1.

Tabulka 5.1: Parametry FH_Power

I/O	Parametry	Datový typ	Popis
IN	Enable	BOOL	Zapne funkční blok s náběžnou hranou
IN	PointerSettings	UDINT	Ukazatel na datovou strukturu s nastavením komunikace
OUT	Done	BOOL	Indikuje stav dokončení FB
OUT	Response	STRING	Odpověď po dokončení FB
OUT	Error	BOOL	Indikuje chybu FB
OUT	ErrorID	UINT	Číslo chyby

5.1.2 Rozdíly

V tomto funkčním bloku se pracuje s proměnnou CommandHelper. Do vyrovnávací paměti nahrajeme příkaz uložený v proměnné CommandHelper pomocí příkazu strcpy, který se poté dosadí do těla rámce pro odeslání.

5.2 Funkční blok FH_Velocity

Druhým základním funkčním blokem je FH_Velocity. Umožňuje uživateli roztočit pohonnou jednotku na uživatelem definované otáčky. Uživatel může hodnotu otáček

zadat jak kladnou, tak i zápornou a tím si vybrat směr otáčení pohonné jednotky.

5.2.1 Parametry

Parametry funkčního blok FH_Velocity jsou zobrazeny v tabulce 5.2.

Tabulka 5.2: Parametry FH_Velocity

I/O	Parametry	Datový typ	Popis
IN	Enable	BOOL	Zapne funkční blok s náběžnou hranou
IN	PointerSettings	UDINT	Ukazatel na datovou strukturu s nastavením komunikace
IN	Value	INT	Udává rychlost otáček (kladné i záporné číslo)
OUT	Done	BOOL	Indikuje stav dokončení FB
OUT	Response	STRING	Odpověď po dokončení FB
OUT	Error	BOOL	Indikuje chybu FB
OUT	ErrorID	UINT	Číslo chyby

5.2.2 Rozdíly

V tomto funkčním bloku se pracuje s proměnnou CommandHelper a Value. Do vyrovnávací paměti nahrajeme příkaz uložený v proměnné CommandHelper a hodnotu otáček uloženou v proměnné Value pomocí příkazu strcpy. Tyto hodnoty poté vytvoří tělo rámce pro odeslání.

5.3 Funkční blok FH_PositionAbsolute

Funkční blok FH_PositionAbsolute umožňuje uživateli nastavit polohu rotoru absolutně, tedy od počátku. Hodnota pozice může být jak kladná, tak i záporná.

5.3.1 Parametry

Parametry funkčního blok FH_PositionAbsolute jsou zobrazeny v tabulce 5.3.

5.3.2 Rozdíly

V tomto funkčním bloku je potřeba odeslat pět rámců s jednotlivými příkazy. K počítání již odeslaných rámců nám slouží vnitřní proměnná PositionHelper. Podle její hodnoty se určuje, jaký příkaz se do pohonné jednotky pošle. Jako první se odesílá rámec s příkazem pro nastavení zrychlení. Poté se posílá rámec s hodnotou zpomalení. Jako další se posílá rámec s maximální rychlostí posunu. Předposledním rámcem je hodnota samotného posunu. Poslední rámec obsahuje příkaz pro vykonání posunu.

Tabulka 5.3: Parametry FH_PositionAbsolute

I/O	Parametry	Datový typ	Popis
IN	Enable	BOOL	Zapne funkční blok s náběžnou hranou
IN	PointerSettings	UDINT	Ukazatel na datovou strukturu s nastavením komunikace
IN	Acceleration	UINT	Hodnota zrychlení
IN	Deceleration	UINT	Hodnota zpomalení
IN	MaximumSpeed	UINT	Maximální rychlost v otáčkách za minutu
IN	Value	INT	Pozice (kladné i záporné číslo)
OUT	Done	BOOL	Indikuje stav dokončení FB
OUT	Busy	BOOL	Indikuje vykonávání FB
OUT	Response	STRING	Odpověď po dokončení FB
OUT	Error	BOOL	Indikuje chybu FB
OUT	ErrorID	UINT	Číslo chyby

5.4 Funkční blok FH_PositionRelative

Funkční blok FH_ PositionRelative umožňuje uživateli nastavit polohu rotoru relativně, tedy posun rotoru o danou hodnotu vzhledem k jeho aktuální pozici. Hodnota pozice může být jak kladná, tak i záporná.

5.4.1 Parametry

Parametry funkčního blok FH_PositionRelative jsou zobrazeny v tabulce 5.4.

Tabulka 5.4: Parametry FH_PositionRelative

I/O	Parametry	Datový typ	Popis
IN	Enable	BOOL	Zapne funkční blok s náběžnou hranou
IN	PointerSettings	UDINT	Ukazatel na datovou strukturu s nastavením komunikace
IN	Acceleration	UINT	Hodnota zrychlení
IN	Deceleration	UINT	Hodnota zpomalení
IN	MaximumSpeed	UINT	Maximální rychlost v otáčkách za minutu
IN	Value	INT	Pozice (kladné i záporné číslo)
OUT	Done	BOOL	Indikuje stav dokončení FB
OUT	Busy	BOOL	Indikuje vykonávání FB
OUT	Response	STRING	Odpověď po dokončení FB
OUT	Error	BOOL	Indikuje chybu FB
OUT	ErrorID	UINT	Číslo chyby

5.4.2 Rozdíly

Tento funkční blok je velmi podobný funkčnímu bloku FH_PositionAbsolute. Je zde také potřeba odeslat pět rámců s jednotlivými příkazy. K počítání již odeslaných rámců nám slouží vnitřní proměnná PositionHelper. Podle její hodnoty se určuje, jaký příkaz se do pohonné jednotky pošle. Jako první se odesílá rámec s příkazem pro nastavení zrychlení. Poté se posílá rámec s hodnotou zpomalení. Jako další se posílá rámec s maximální rychlostí posunu. Předposledním rámcem je hodnota samotného posunu. Poslední rámec obsahuje příkaz pro vykonání posunu.

5.5 Funkční blok FH_SetCyclicPosition

Funkční blok FH_SetCyclicPosition umožňuje uživateli nastavit pozici rotoru. V případě, že chce uživatel pozici změnit, nemusí celý funkční blok volat znovu, ale stačí pouze změnit hodnotu pozice.

5.5.1 Parametry

Parametry funkčního blok FH_SetCyclicPosition jsou zobrazeny v tabulce 5.5.

Tabulka 5.5: Parametry FH_SetCyclicPosition

I/O	Parametry	Datový typ	Popis
IN	Enable	BOOL	Zapne funkční blok s náběžnou hranou
IN	PointerSettings	UDINT	Ukazatel na datovou strukturu s nastavením komunikace
IN	Value	INT	Nastavuje pozici (kladné i záporné číslo)
OUT	Done	BOOL	Indikuje stav dokončení FB
OUT	Response	STRING	Odpověď po dokončení FB
OUT	Error	BOOL	Indikuje chybu FB
OUT	ErrorID	UINT	Číslo chyby

5.5.2 Rozdíly

U tohoto funkčního bloku je potřeba odeslat dva rámce. K počítání již odeslaných rámců nám slouží vnitřní proměnná PositionHelper. Podle její hodnoty se určuje, jaký příkaz se do pohonné jednotky pošle. Pokud je hodnota PositionHelperu rovna 0 do vyrovnávací paměti se nahraje příkaz uložený v proměnné CommandHelper a požadovaná pozice rotoru uložená v proměnné Value. Pokud je hodnota proměnné PositionHelper rovna 1 odešle se rámec pro vykonání pohybu.

6 Uživatelský návod knihovny FHLib

Tato kapitola obsahuje podrobný popis, jak importovat knihovnu do vlastního projektu. Jaké je potřeba vytvořit datové struktury, proměnné a jak zavolat jednotlivé funkční bloky knihovny.

6.1 Import knihovny

Jakmile si uživatel vytvoří vlastní projekt, je potřeba, aby naimportoval knihovnu. Přepnutím se do Logického pohledu a pravým kliknutím na jakékoli místo se vyvolá nabídka, kde vybereme možnost Add Object. Z kategorie Library vybereme možnost Existing Library. Výběrem se otevře dialog pro výběr knihovny. Zvolíme knihovnu FHLib a dokončíme průvodce.

6.2 Datová struktura

Pro použití knihovny je potřeba nejdříve připravit si jednoduchou datovou strukturu, ve které bude uloženo nastavení pro samotnou komunikaci. Toto nastavení se poté bude předávat každému funkčnímu bloku knihovny FHLib. Datová struktura se bude jmenovat Settings a bude mít 7 parametrů s datovým typem STRING. Těmito parametry jsou device_name, type, baud_rate, parity,data_bits, stop_bits a builder_string.

6.3 Proměnné

Je potřeba vytvořit proměnnou pro každý funkční blok z knihovny FHLib, který chceme použít. Dále je potřeba vytvořit proměnnou pro datovou strukturu Settings. Pokud tedy budeme chtít použít funkční blok FH_Power, budeme muset vytvořit proměnnou Settings_0 a nastavit jí datový typ Settings. Dále musíme vytvořit proměnnou FH_Power_0 s datovým typem FH_Power.

6.4 Volání funkčního bloku v programu

Jakmile máme importovanou knihovnu, vytvořenou datovou struktur pro nastavení komunikace a vytvořené proměnné, můžeme se přesunout do samotného programu. V programu je potřeba nastavit parametry datové struktury Settings. Parametr device_name nastavíme na hodnotu IF1. Parametr type nastavíme na hodnotu ‘RS232‘. Baud_rate nastavíme na hodnotu ‘9600‘. Parametr data_bits nastavíme na 8. Parametr Parity nastavíme na hodnotu N. Parametr stop_bits nastavíme na hodnotu 1. Po nastavení této struktury můžeme již zavolat jednotlivé funkční bloky. Například funkční blok FH_Power zavoláme příkazem FH_Power_0(PointerSettings:= ADR(Settings), Enable:= TRUE);

7 Tvorba nápovědy

V této kapitole je popsána tvorba nápověd v programu HelpNDoc4. Dále je zde popsána tvorba uživatelské nápovědy ke knihovně FHLib, která obsahuje základní informace, popis jednotlivých funkčních bloků a chybové stavy.

7.1 Popis programu HelpNDoc4

Grafické rozhraní programu HelpNDoc4 se skládá ze tří částí. První částí jsou jednotlivé karty, které slouží pro ovládání funkcí, jako jsou generování nápovědy, přidání tématu do nápovědy, formátování textu, vkládání obrázků a podobně. Další částí grafického rozhraní je zobrazení struktury nápovědy. Poslední, hlavní část programu HelpNDoc4 obsahuje okno, ve kterém píšeme samotnou nápovědu.

7.2 Tvorba samotné nápovědy

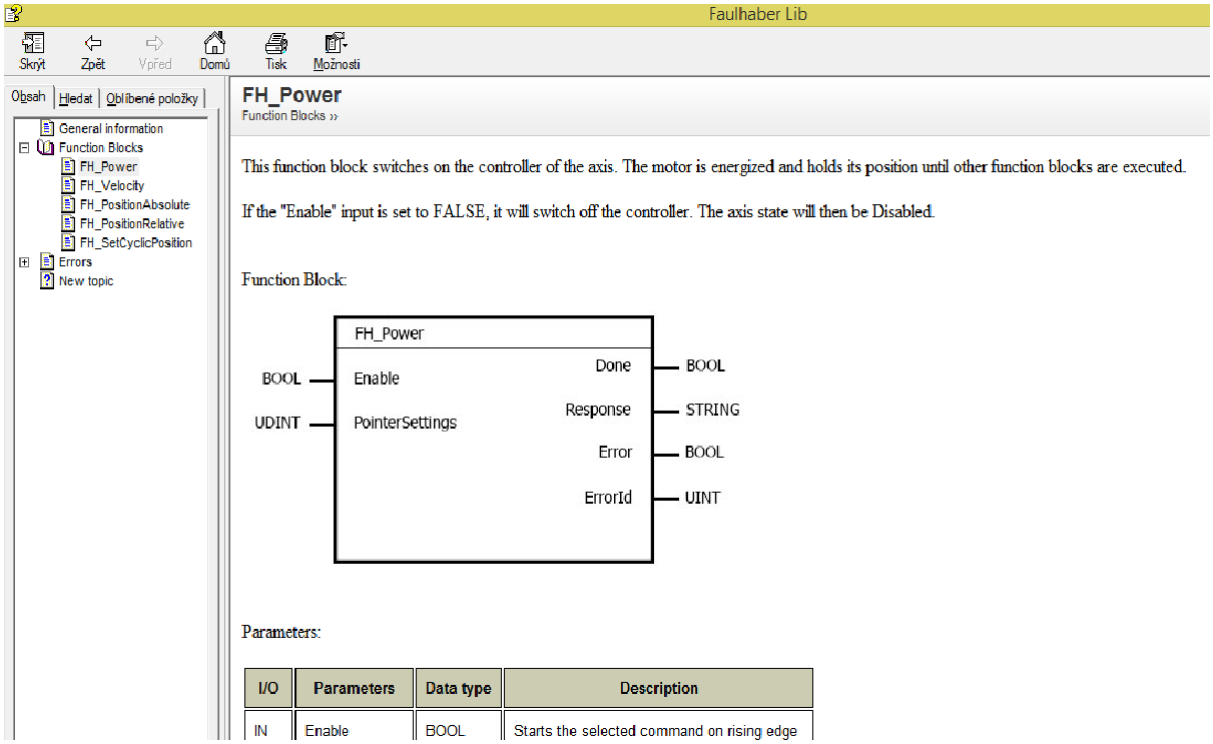
Nápovědu jsme se rozhodli rozdělit na tři části. První část obsahuje základní informace o samotné knihovně a odkazy na jednotlivé funkční bloky v ní obsažené. Dále pro každý funkční blok vytvoříme samostatnou stránku obsahující popis, diagram, parametry a ukázku použití. Poslední částí nápovědy jsou chybové stavy. Je zde seznam všech chybových stavů a pro každý chybový stav je zde uveden jeho popis, jak chybu opravit a v jakých funkčních blocích se chyba může vyskytovat.

7.2.1 Základní informace

Nejprve jsme vytvořili téma s názvem General infromation, kliknutím na tlačítko Add topic v kartě Home. Toto téma obsahuje obecné informace o samotné knihovně a seznam všech funkčních bloků. Položky v tomto seznamu jsou hypertextové odkazy na jednotlivá témata samotných funkčních bloků.

7.2.2 Funkční bloky

Dále jsme vytvořili téma, které obsahuje pět podtémat. Pro každý funkční blok jedno. V těchto podtématech je vždy uvedeno k čemu funkční blok slouží. Dále je zde diagram funkčního bloku, který graficky znázorňuje jaké má vstupní a výstupní parametry a jejich datové typy. Poté je zde tabulka všech parametrů a jejich datových typů a popisů vytvořená pomocí HTML kódu. Tuto tabulku jsme vytvořili



Obrázek 7.1: Ukázka nápovědy

kliknutím na Insert HTML code v kartě Insert. Poslední část, kterou každá nápověda k funkčnímu bloku obsahuje je ukázka zavolání tohoto bloku.

7.2.3 Chybové stavy

Jako poslední jsme vytvořili téma s názvem Errors, které obsahuje seznam všech chybových čísel a hypertextové odkazy na ně. Každé chybové číslo obsahuje popis chyby, řešení, jak chybu odstranit a v jakých funkčních blocích se chyba může vyskytovat.

Závěr

Seznámili jsme se s pohony Faulhaber a jejich komunikačním protokolem. Na základě těchto znalostí, jsme vytvořili knihovnu, obsahující funkční bloky pro ovládání jednotek Faulhaber v jazyku ST. Za pomocí těchto funkčních bloků lze jednotku zapnout, nastavit ji absolutně, nebo relativně pozici rotoru, nastavit ji rychlost a směr otáčení a cyklicky nastavovat pozici rotoru. K této knihovně funkčních bloků jsme poté vytvořili uživatelskou nápovědu. V této nápovědě se uživatel stručně a přehledně dozví jak knihovnu používat.

Práce obsahuje popis jednotlivých technologií, které byly použity. Obsahuje také popis zapojení pohonu Faulhaber a vyzkoušení komunikace přes software Motion Manager od firmy Faulhaber. Dále také podrobný popis vytváření knihovny a jejich funkčních bloků. V poslední řadě práce obsahuje návod jak knihovnu naimplementovat a popis tvorby uživatelské nápovědy.

Knihovnu je možné dále rozvíjet. Jeden z hlavních směrů je rozšířit ji o možnost ovládání více pohonů zároveň. Dále je zde prostor pro vytváření dalších funkčních bloků pro jeho ovládání a předávání informací o stavu pohonu uživateli.

Literatura

[1] AUTOMATION, B&R. Controls - training text. Austria : [s.n.], 2008. 205 s.

[2] FAULHABER. Communication and Function Manual [online]. 2012 [cit. 2015-03-02]. Dostupné z: http://www.faulhaber.com/uploadpk/EN_7000_05029.pdf

[3] JOHN, Kharl-Heinz; TIEGELKAMP, Michael. IEC 61131-3 Programming Industrial Automation Systems : Concepts and Programming Languages, Requirements for Programming Systems, Decision - Making Aids. 2nd Edition. NewYork : Springer, 2010. 390 s. ISBN 978-3-642-12015-2

[4] BERNECKER + RAINER INDUSTRIE-ELEKTRONIK GES.M. B. H. B&R Help: Libraries. 2013. vyd. 2013.

[5] BERNECKER + RAINER INDUSTRIE-ELEKTRONIK GES.M. B. H. B&R Help: DVFrame. 2013. vyd. 2013.

A Obsah přiloženého CD

V přiloženém CD je obsažen text bakalářské práce, zdrojové kódy knihovny, exportovaná knihovna, uživatelská nápověda a použité obrázky.

- Text bakalářské práce
  - bakalarska_prace_2015_Miroslav_Hlavka.pdf
  - bakalarska_prace_2015_Miroslav_Hlavka.tex
  - kopie_zadani_bakalarska_prace_2015_Miroslav_Hlavka.pdf
- Knihovna
  - Zdrojové soubory
  - Exportovaná knihovna
- Uživatelská nápověda
  - Help.chm
- img - použité obrázky pro ilustraci textu bakalářské práce