

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

LABORATORNÍ UKÁZKA ŘÍZENÍ ASYNCHRONNÍHO MOTORU

LABORATORY EXAMPLE OF SYNCHRONOUS MOTOR CONTROL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. VÁCLAV VRTAL
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. ZDENĚK NĚMEC, CSc.
SUPERVISOR

BRNO 2010

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Václav Vrtal

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Laboratorní ukázka řízení asynchronního motoru

v anglickém jazyce:

Laboratory example of synchronous motor control

Stručná charakteristika problematiky úkolu:

Jde o laboratorní ukázku řízení otáček asynchronního třífázového motoru moderními prostředky. K řízení bude použito frekvenčního měniče, programovatelného automatu a operátorského panelu.

Cíle diplomové práce:

1. Osvojit si práci s daným třífázovým motorem o výkonu 0,25 kW a s frekvenčním měničem Sinamics G110.
2. K řízení uvažujte programovatelný automat Simatic S7-224XP a operátorský panel TP 177 micro.
3. Navrhnout a realizovat uživatelské programové vybavení, které by demonstrovalo možnosti regulací pro ukázky v laboratorních cvičeních.
4. Vyhodnotit dosažené výsledky a shrnout poznatky z uvádění do provozu.

Seznam odborné literatury:

[1] Švarc, I.; Šeda, M.; Vítečková, M. Automatické řízení. Brno: CERM, 2007. 324 s. ISBN 978-80-214-3491-2.
[2] Firemní literatura k použitým přístrojům.

Vedoucí diplomové práce: doc. Ing. Zdeněk Němec, CSc.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2009/2010.

V Brně, dne 23.11.2009

L.S.

Ing. Jan Roupec, Ph.D.	prof. RNDr. Miroslav Doupovec, CSc.
Ředitel ústavu	Děkan fakulty

Abstrakt

Tato diplomová práce se zabývá řízením otáček asynchronního třífázového motoru frekvenčním měničem Sinamics G110 pomocí programovatelného automatu SIMATIC S7-224XP. Ovládání programovatelného automatu je realizováno prostřednictvím operátorského panelu TP 177 micro. Komunikace mezi automatem a frekvenčním měničem je realizována pomocí USS protokolu po sběrnici RS-485.

Hlavním cílem bylo navrhnout a realizovat programové vybavení, které by umožňovalo obsluze volit různé varianty časových průběhů otáček (změny sinusové, skokové, pilové, atd.), ale také možnost regulaci polohy.

Dalším cílem bylo vytvoření grafického prostředí pro operátorský panel, umožňující obsluze řízení, editaci a sledovaní stavu otáček motoru.

Posledním cílem bylo vyhodnocení dosažených výsledků a shrnutí poznatků při uvádění do provozu.

Abstract

This thesis deals with the speed control three-phase asynchronous motor revolutions by a frequency converter Sinamics G110 by way of the programmable automat SIMATIC S7-224XP. Control of programmable automat is realized the operator panel TP 177 micro. Communication between the PLC and frequency converter is realized by USS protocol on bus RS-485.

The main goal was to design and implement software, which would allow operation to choose different variations of speed time histories (sinusoid, saw, jump course, etc.), but also the possibility to control the position.

Another goal was to create graphical environment for the operator panel that allows manual control, editing and monitoring of engine speed.

The last goal was to evaluate the results and summary of findings during commissioning.

PODĚKOVÁNÍ

Touto cestou bych chtěl poděkovat vedoucímu diplomové práce Ing. Zdeňku Němcovi, CSc., za odborné vedení, cenné rady a připomínky při vypracování této diplomové práce.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, dle pokynů vedoucího diplomové práce s využitím materiálu uvedených v seznamu použité literatury.

V Brně dne 26. 5. 2010

……...

podpis

Obsah

Zadání diplomové práce ... 3

Abstrakt ... 5

ČESTNÉ PROHLÁŠENÍ ... 7

1 Úvod ... 13

2 Popis jednotlivého hardwaru ... 15

2.1 Frekvenční měnič SINAMICS G110 ... 15

2.1.1 Základní popis frekvenčního měniče ... 15

2.1.2 Technické údaje frekvenčního měniče SINAMICS G110 ... 15

2.1.3 Blokové schéma měniče a jeho popis ... 17

2.1.4 Popis sběrnice RS-485 ... 18

2.1.5 Zapojení sběrnice USS ke svorkám frekvenčního měniče a jejich popis ... 19

2.1.6 DIP přepínač ... 20

2.1.7 Popis tlačítek BOP panelu ... 20

2.1.8 Uvedení frekvenčního měniče do provozu ... 22

2.2 PLC SIMATIC S7 224 XP ... 23

2.2.1 Základní charakteristika ... 23

2.2.2 Propojení automatu s frekvenčním měničem ... 24

2.2.3 Princip vykonávání řídícího programu ... 25

2.2.4 Vytvoření programu pomocí Step 7 – Micro/WIN ... 25

2.3 Operátorský panel SIMATIC TP 177 micro ... 27

2.3.1 Základní charakteristika ... 27

2.3.2 SIMATIC WinCC flexible 2008 Micro ... 27

2.4 Napájecí zdroj LOGO! Power 24 V ... 30

2.4.1 Základní charakteristika ... 30

2.4.2 Technická data zdroje LOGO!Power 24 V ... 30

2.5 Asynchronní motor s inkrementálním snímačem ... 31

2.5.1 Základní charakteristika asynchronního motoru 31

2.5.2 Technické specifikace trojfázového asynchronního motoru SIEMENS 31

2.5.3 Inkrementální snímač 32

2.5.4 Ukázka zapojení celého setu 33

Obr. 2.18 Ukázka zapojení všech komponent v provozním režimu. 33

3 Knihovna USS 35

3.1 K čemu slouží knihovna USS? 35

3.2 Požadavky a omezení při použití knihovny USS 35

3.3 Použití instrukcí USS 36

3.4 Instrukce USS4_INIT 36

3.5 Instrukce USS4_DRV_CTRL 38

3.6 USS4_RPM_x 40

3.7 Vytvoření knihovny USS 42

4 Sestavení hardwaru pro laboratorní úlohu 45

4.1 Blokové schéma laboratorní úlohy 45

4.2 Propojení a nastavení jednotlivých prvků řídící soustavy 45

4.2.1 Propojení automatu s operátorským panelem 45

4.2.2 Propojení automatu s PC 46

4.2.3 Nastavení komunikace mezi automatem a PC 46

4.2.4 Nastavení komunikace mezi automatem a operátorským panelem 48

4.2.5 Propojení a nastavení komunikace automatu s frekvenčním měničem 49

4.3 Připojení inkrementálního snímače na vstupy automatu 50

5 Návrh regulátoru polohy a jeho nastavení 51

5.1 Návrh regulátoru polohy 51

5.1.1 První varianta regulátoru 51

5.1.2 Druhá varianta regulátoru 52

5.1.3 Finální varianta regulátoru 53

5.2 Nastavení regulátoru 54
5.2.1 Nastavení regulátoru rychlosti s dopředným řízením 54
5.2.2 Nastavení regulátoru polohy 56
6 Programové vybavení automatu 59
6.1 Vývojový diagram 59
6.2 Podrobný popis programu 61
6.2.1 Hlavní program Main 61
6.2.2 Podprogram počáteční nastavení SBR19 62
6.2.3 Program generování pilových průběhů SBR0 64
6.2.4 Program na generování obdélníkových průběhů 66
6.2.5 Řízení otáček motoru pomocí sinusoidy 67
6.2.6 Řízení otáček pomocí rampy 69
6.2.7 Program pro ruční řízení motoru 72
6.2.8 Program pro řízení instrukcí pro čtení parametrů z frekvenčního měniče 72
6.2.9 Časované přerušení 1 (INT5) 74
6.2.10 Časované přerušení 0 (INT0) 77
6.2.11 Podprogram krokování 79
7 Uživatelské prostředí realizované pomocí OP 81
7.1 Struktura menu uživatelského prostředí 81
7.2 Popis a ukázky vybraných obrazovek uživatelského prostředí 82
7.2.1 Hlavní menu 82
7.2.2 Úlohy na řízení otáček motoru 83
7.2.3 Ruční řízení otáček motoru 83
7.2.4 Řízení motoru pomocí sinusoidy 84
7.2.5 Graf generované sinusoidy 85
7.2.6 Výpis základních údajů 86
8 Závěr a shrnutí poznatků z uvádění do provozu 87

SEZNAM POUŽITÉ LITERATURY 89

SEZNAM PŘÍLOH 91

1 Úvod

V této práci se budu zabývat řízením otáček asynchronního třífázového motoru frekvenčním měničem Simanics G110 pomocí programovatelného automatu SIMATICS S7-224XP. Obsluha programovatelného automatu je realizována prostřednictvím operátorského panelu TP 177 micro. Hlavním cílem bylo navrhnout a realizovat programové vybavení, které by umožňovalo obsluze volit různé varianty časových průběhů otáček (změny sinusové, skokové, pilové, atd.), ale také možnost regulaci polohy. Dalším cílem bylo vytvoření grafického prostředí pro operátorský panel, umožňující obsluze řízení, editaci a sledovaní stavu otáček motoru.

Důvodem proč jsem si vybral tuto diplomovou práci, bylo zamyšlení nad tím, jaké bych chtěl budoucí zaměstnání. Jelikož trend v dnešní době je takový, že je snaha o zvyšování produktivity práce, která nelze zajistit bez modernizace, rekonstrukce a automatizace výrobního procesu. Dochází tedy nahrazování pracovníka automatem všude tam, kde se jedná o monotónní práci nebo práci ve škodlivém prostředí. Automatizace nahrazuje pracovníka i tam, kde zvýšená automatizace výroby přinese i zvýšenou kvalitu výroby. Z tohoto důvodu jsem si vybral tuto práci, která mi umožnila se podrobně seznámit s různým hardwarem a programovacími metodami uplatňující se při realizaci automatizačního procesu.

Prvním kapitole se budeme věnovat seznámením s jednotlivým hardwarem. K tomuto účelu mi posloužila technická dokumentace firmy Siemens, která je výrobcem veškerého hardwaru používaného v této diplomové práci. Po nastudování technické dokumentace jsem provedl propojení veškerého hardwaru. Pro propojení programovatelného automatu s frekvenčním měničem bylo nutné zhotovit komunikační kabel, který nebyl součásti setu. V další kapitole se seznámíme s komunikačním protokolem USS. Jedná se o speciální knihovnu, která umožňuje jednoduše řídit frekvenční měnič pomocí programovatelného automatu po sběrnici RS-485. V následující kapitole se budeme zabývat návrhem a nastavením regulátoru polohy. V dalších kapitolách si podrobně popíšeme programové vybavení laboratorních úloh pro programovatelný automat a operátorský panel. Na závěr provedeme zhodnocení dosažených výsledků a shrnutí poznatků při uvádění do provozu.

2 Popis jednotlivého hardwaru

2.1 Frekvenční měnič SINAMICS G110

2.1.1 Základní popis frekvenčního měniče

SINAMICS G110 je frekvenční měnič vybavený základními funkcemi pro regulaci otáček asynchronních motorů v rozmanitých průmyslových aplikacích. Kompaktní měnič SINAMICS G110 je napájen jednofázovým střídavým napětím (200 až 240V) a reguluje otáčky motoru změnou napětí a frekvencí výstupního třífázového napětí (3 x 230 V). Patří do skupiny měničů SINAMICS, která je vhodná pro řízení motorů malých výkonů. SINAMICS G110 se uplatní všude tam, kde je potřeba řízení otáček asynchronních motorů malých výkonů, především je určen pro řízení otáček ventilátorů, čerpadel, různých dopravníku a mnohých aplikací v průmyslu.

Jeho předností je jednoduchá instalace, nastavení a široké možnosti ovládání. Je odolný vůči elektromagnetickému rušení. Obsahuje dostatečné množství parametrů k nastavení v širokém spektru aplikací. Varianta s analogovým nebo USS řízením předurčuje měnič pro konkrétní aplikaci. Volitelně lze pořídit panel BOP, který umožňuje ovládání a nastavování parametrů frekvenčního měniče, bez potřeby dalších prvků (PC). BOP panel umožňuje také zobrazení stavových a výstražných hlášení frekvenčního měniče. BOP panel umožňuje klonování parametrů, což zkracuje čas potřebný k nastavení více měničů. Frekvenční měnič obsahuje ochranné funkce, k ochraně samotného měniče, tak i motoru. Patří sem například odpojení motoru při překročení maximální povolené teploty motoru, při překročení maximálního proudu, při předpětí nebo zkratu. Stav měniče se indikuje pomocí informační LED. Frekvenční měnič může pracovat na různých frekvencích napájecího napětí, a to na 50 nebo 60 Hz. Volba pracovní frekvence se provádí DIP přepínačem na čele přístroje. Zabudovaný interface umožňuje komunikaci měniče s ostatními systémy řízení po sériové lince RS485 (USS verze). Zakončení sběrnice RS485 (USS verze) se provádí pomocí DIP přepínač.

2.1.2 Technické údaje frekvenčního měniče SINAMICS G110

Jednotlivé varianty měniče řady G110 se od sebe odlišují především pracovním výkonem v rozmezí 0,12 ÷ 3 kW a způsobem řízení měniče. Mezi další rozdíly patří rozměry měniče, volba ze dvou chladičů (normální varianta a varianta s plochým chladičem) a volba měniče s filtrem nebo bez filtru. Naše varianta frekvenčního měniče může pracovat s maximálním výkonem 0,25 kW a komunikuje s ostatními systémy pomocí RS485 (USS), rozměrově patří k nejmenším, používá normální velikost chladiče a neobsahuje filtr. Ostatní parametry se od jednotlivých variant neodlišují. Následující výběr parametrů odpovídá našemu typu měniče s označením SINAMICS G110 CPM110 USS.

Technické údaje:

- vstupní napětí: 200 ÷ 240 V ± 10% 1AC
- výstupní napětí: 3 x 230 V
- maximální výkon motoru: 0,25 kW
- vstupní proud: 4,5 A (pro napájecí napětí 230V)
- výstupní proud: 1,7 A

- vstupní frekvence: 47 ÷ 63 Hz
- výstupní frekvence: 0 ÷ 650 Hz
- přetížitelnost 1,5 x výstupní proud (150%) po dobu 60 s, pak 0,85 x výstupní proud (85%) po dobu 240 s, délka cyklu je 300 s.
- účinnost měniče:
  - u výkonů < 750W je 90 ÷ 94 %
  - u výkonů ≥ 750W je ≥ 95 %
- modulační frekvence: 8 kHz (standardně) nebo 2 ÷ 16 kHz (s krokem 2kHz)
- rozlišení nastavení frekvence 0,01 Hz
- 3 programovatelné digitální vstupy, neizolované, typ PNP, kompatibilní se SIMATIC
- 1 digitální výstup, optočlen (24VDC, 50mA, odporová zátěž, NPN)
- sériový interface RS485, komunikace protokolem USS
- stejnosměrné brzdění
- teplotní rozsah:
  - Provozní: -10 ÷ +40°C
  - Skladovací: -40 ÷ +70°C
- maximální relativní vlhkost je 95%
- ochrany: Podpětí, přepětí, zkrat na zem, zkrat mezi fázemi, chod bez zátěže, ochrana motoru I2t, přehřátí motoru i měniče
- krytí IP20

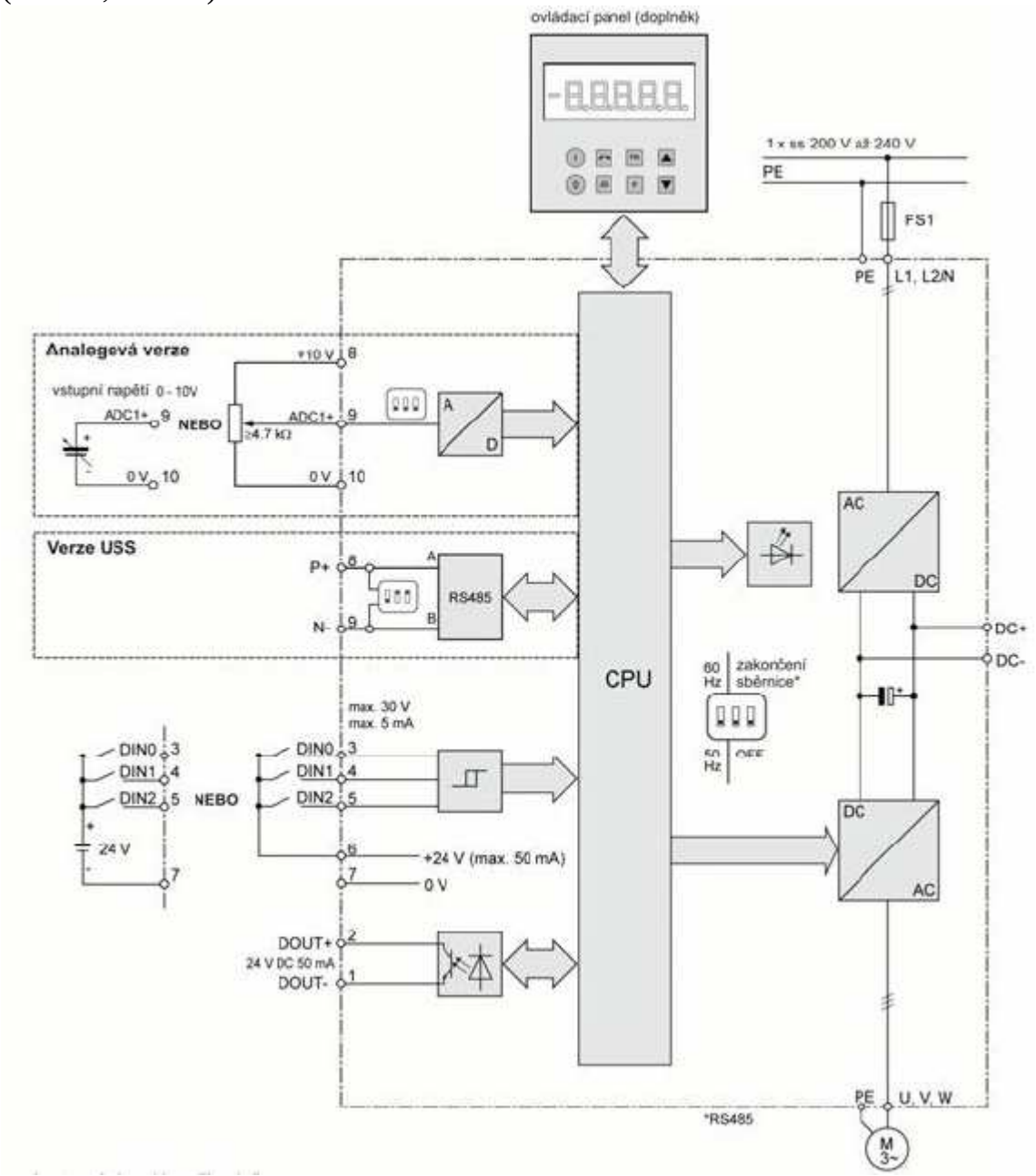
Pro úplnost připojuji snímek štítku měniče – viz obr. 2.1. Štítek měniče obsahuje přesné označení měniče, objednací a sériové číslo a všechny důležité parametry pro bezpečné provozování měniče.



Obr. 2.1 Technické údaje na štítku měniče.

2.1.3 Blokové schéma měniče a jeho popis

Na obrázku 2.2 je zobrazeno blokové schéma měniče. Jsou zde zobrazeny nejdůležitější bloky měniče. Blokové schéma na pravé straně zobrazuje jednotlivé vstupy a výstupy frekvenčního měniče, které jsou vyvedeny na řídící svorkovnici. Jak je patrné ze schématu, tak u analogové verze nastavujeme výstupní frekvenci, za pomoci potenciometru zapojeného mezi svorky 8, 9 a 10 nebo změnou stejnosměrného napětí v rozsahu 0 – 10 V přiváděného mezi svorky 9 a 10. Varianta USS využívá k řízení výstupní frekvence měniče sběrnici RS485, která využívá svorky 8 a 9, jako u analogové varianty. Popis sběrnice RS485 bude následovat v další kapitole, nyní se zaměříme na to, že komunikace probíhá po dvou vodičích (kanál A, kanál B).

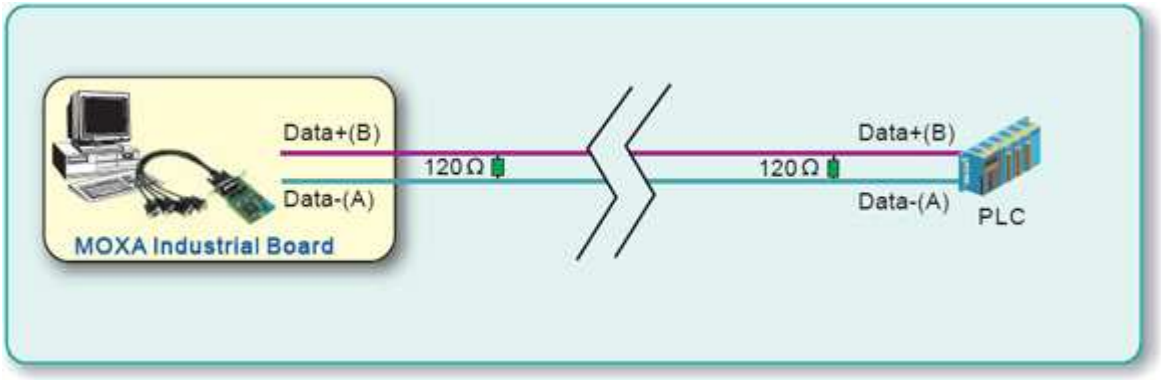


Obr. 2.2 Blokové schéma frekvenčního měniče SINAMICS G110

Dalším způsobem ovládání měniče je přes svorkovnici, jedná se o svorky 1 – 7. Svorky 3, 4 a 5 představují digitální vstupy a svorky 1 a 2 jsou digitální výstupy. Pomocí těchto vstupů a výstupu můžeme provádět jednoduché řízení. Funkce jednotlivých vstupů a výstupu se nastavuje ve frekvenčním měniči, za pomocí BOP panelu nebo propojení měniče s PC. Při tomto režimu řízení nastavujeme tři pevné kmitočty a jejich kombinaci (součtem) dostáváme výstupní frekvenci. Pevné kmitočty se opět nastavují pomocí BOP panelu nebo PC. Poslední způsob ovládání motoru je pomocí BOP panelu. Pomocí panelu můžeme zapnout/vypnout motor, měnit požadovanou frekvenci, provádět změnu směru rotace motoru a neposledně provádět krokování motoru. Tímto způsobem, můžeme pohodlně ovládat motor a nastavovat parametry měniče, bez zapotřebí dalšího příslušenství (PLC, PC). Volba způsobu řízení měniče se nastavuje pomocí parametrů (BOP panel, PC).

2.1.4 Popis sběrnice RS-485

Jedná se o sériovou průmyslovou sběrnici, pomocí které může komunikovat maximálně 32 vysílačů a 32 přijímačů. Funkčnost sběrnice je zaručena díky tomu, že všechny přijímače i neaktivní vysílače se v klidu musí nacházet ve stavu vysoké impedance, tj. nijak neovlivňují komunikující zařízení. Pouze jedno zařízení na sběrnici může v daném čase pracovat jako řadič (vysílač), ovšem veškeré řízení přenosu i arbitráž sběrnice je ponechána na protokolu vyšší. U této sběrnice se při vysílání používá diferenciálního kódování dat – jedna polarita představuje logickou jedničku, obrácená polarita pak logickou nulu. Rozdíl mezi oběma napěťovými potenciály musí dosahovat hodnoty minimálně 0,2 Voltů, typicky se však používají mnohem vyšší rozdíly, například 5 V, 7 V či 12 V. Na obrázku 2.3 si můžeme všimnout rezistorů na obou koncích přenosové linky. Jejich odpor by měl odpovídat impedanci vedení, hodnota 100 Ω až 120 Ω odpovídá kroucené dvojlince.

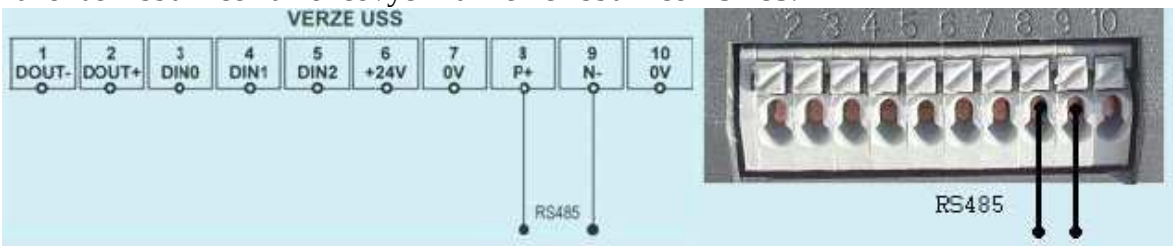


Obr 2.3: Základní způsob zapojení dvou zařízení na sběrnici RS-485.

Díky použití kroucené dvojlinky a diferenciálního kódování je možné data přenášet i na poměrně velkou vzdálenost. Jako hranice dosažitelná za běžných podmínek (průmysl) se uvádí cca 1200 metrů, přičemž přenosová rychlost může dosahovat hodnot až 10 Mbit/s. Ovšem přenosová linka musí být správně zapojena, což v tomto případě znamená nutnost připojit na oba konce linky rezistory (terminátory) s odporem cca 120 Ω. Kromě těchto rezistorů se obě diferenciální linky v klidu nastavují na nějaké napětí odlišné od 0 V.

2.1.5 Zapojení sběrnice USS ke svorkám frek. měniče a jejich popis

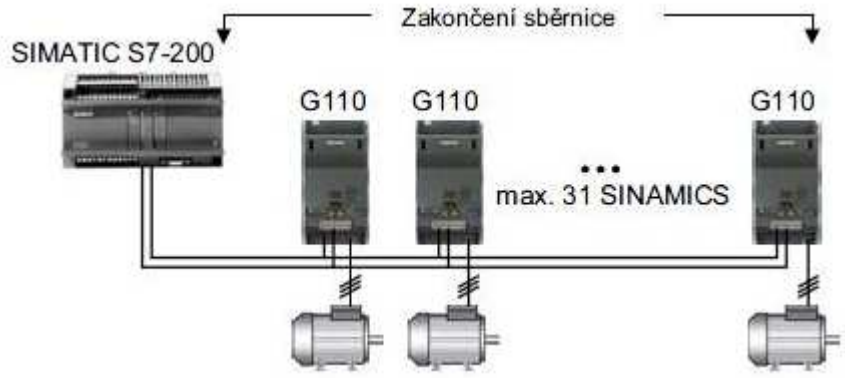
Připojení sběrnice RS485 na svorky měniče je zobrazeno na obrázku 2.4. Levá strana obrázku, zobrazuje značení jednotlivých svorek a prává strana zobrazuje skutečnou svorku na frekvenčním měniči. Popis jednotlivých svorek je v tabulce 2.1. Obrázek 2.5 znázorňuje zakončení sběrnice na koncových zařízeních sběrnice RS 485.



Obr. 2.4 Připojení sběrnice RS485 na svorky měniče.

Svorka	Popis	Funkce	
1	DOUT-	Digitální výstup (-)	
2	DOUT+	Digitální výstup (+)	
3	DIN0	Digitální vstup 0	
4	DIN1	Digitální vstup 1	
5	DIN2	Digitální vstup 2	
6	-	Izolovaný výstup +24 V / 50mA	
7	-	Výstup 0 V	
Varianta		**Analogová**	**USS**
8	-	Výstup + 10 V	RS485 P+ (A)
9	ADC	Analogový vstup	RS485 N- (B)
10	-	Výstup 0 V	

Tab. 2.1 Popis svorkovnice frekvenčního měniče.



Obr. 2.5 Ukázka zakončení sběrnice RS 485.

2.1.6 DIP přepínač

Na měniči SINAMICS G110 je standardně frekvence sítě 50 Hz. Pro motory navržené pro práci v síti s frekvencí 60 Hz lze změnu standardně nastavené frekvence měniče provést pomocí přepínače DIP na přední stěně měniče.

Poslední měnič na síťové sběrnici je třeba zakončit. Zakončení se provádí nastavením DIP přepínačů pro zakončení sběrnice na přední stěně měniče (DIP přepínače 2 a 3) do polohy „Bus termination“ (poloha ON - ZAPNUTO). Všechna zařízení na sběrnici USS je nutné propojit společným vodičem o napětí 0 V. K tomuto účelu lze použít svorku č. 10 na řídicí svorkovnici.



Obr. 2.5 DIP přepínač pro nastavení kmitočtu sítě a zakončení sběrnice

2.1.7 Popis tlačítek BOP panelu

BOP panel obsahuje display, který zobrazuje stav frekvenčního měniče a ovládací tlačítka viz obr. 2.6.



Obr. 2.6 BOP panel.

Popis funkce jednotlivých tlačítek na BOP panelu je vypsána v následující tab. 2.2.

	Tímto tlačítkem provedeme zapnutí měniče, tlačítko je při počátečním nastavení vypnuto. Tlačítko aktivujeme nastavením parametru: P0700 =1
	Stisknutím tohoto tlačítka způsobí plynule zastaveni motoru, podle nastavené doběhové rampy. Toto tlačítko je z výroby vypnuto. Tlačítko aktivujeme nastavením parametru: P0700 = 1
	Stisknutím tohoto tlačítka provedeme změnu směru otáčení motoru. Opačný směr je indikován znaménkem minus (-). Funkce je z výroby vypnuta. Tlačítko aktivujeme nastavením parametru: P0700 = 1
	Po stisknutí tlačítka při stojícím pohonu se motor začne rozbíhat v závislosti na nastavených hodnotách parametrů. Po uvolnění tlačítka dojde k zastavení motoru. Stisknutí tlačítka při běžícím pohonu nemá žádný účinek.
	Tlačítko slouží k zobrazení dalších informací. Při stisknuti a podrženi tlačítka se zobrazí následující parametry: 1. hodnota napětí stejnosměrného meziobvodu (signalizace zobrazením d – jednotky napětí) 2. výstupní kmitočet (Hz) 3. hodnota výstupního napětí (signalizace o – jednotky napětí). 4. hodnota určena parametrem P0005 (pokud je parametr P0005 nastaven na zobrazování některé z výše uvedených hodnot (1–3) nezobrazí se nic). **Kvitování** Při poruše nebo poplašném hlášení lze krátkým stisknutím tlačítka Fn kvitovat poruchový stav. **Přepínání** Krátkým stisknutím tlačítka Fn lze ze kteréhokoli parametru (rXXXX nebo PXXXX) přepnout na r0000, a jiný parametr pak podle potřeby změnit. Po návratu k r0000 se pak stisknutím tlačítka Fn vrátíme k výchozímu parametru.
	Tímto tlačítkem se dostaneme do menu s parametry. Pomocí parametrů měníme vlastnosti frekvenčního měniče.
	Tlačítko slouží ke zvětšení zobrazené hodnoty. Slouží k listování v menu parametrů, ke zvětšování hodnoty parametrů a v aktivním režimu ke zvyšování požadované frekvence.
	Tlačítko slouží ke zmenšení zobrazené hodnoty. Slouží k listování v menu parametrů, ke snížení hodnoty parametrů a v aktivním režimu ke snížení požadované frekvence.

Tab. 2.2 Popis jednotlivých tlačítek ovládacího panelu.

2.1.8 Uvedení frekvenčního měniče do provozu

Řízení frekvenčního měniče je nastaveno od výroby pomocí USS protokolu (parametr P0700 = 5). Frekvenční měnič budeme řídit pomocí USS protokolu, ale pro počáteční ověření funkčnosti měniče nastavíme řízení měniče přes BOP panel (P0700 = 1). Postup změny parametru P0700 je následující:

1. po stisknutí tlačítka P (parametr) v pohotovostním režimu měniče (obr. 1) se dostaneme do režimu změny parametrů (obr. 2).
2. Pomocí šipek se dostaneme na požadovaný parametr (obr. 3 – P0700) a po opětovném stisknutí tlačítka P potvrdíme volbu požadovaného parametru. Parametr P0700 = 5 (obr. 4), postupným stisknutím tlačítka pro snížení hodnoty nastavíme parametr P0700 = 1.
3. Po nastavení požadované hodnoty, potvrdíme hodnotu pomocí tlačítka P (obr. 5).
4. K vrácení do pohotovostního režimu, se provádí parametrem 0000 (obr. 2). K tomuto parametru, se dostaneme pomocí tlačítka FN, nebo pomocí šipek. Potvrzením tlačítkem P se vrátíme do pohotovostního režimu. Tímto způsobem nastavíme výběr zdroje žádané hodnoty na BOP panel (parametr P1000 = 1).



Obr. 2.7 Postup při změně parametru P0700 (volba řízení měniče pomocí BOP panelu)

Nyní máme měnič nastaven na řízení pomocí BOP panelu. Ještě než spustíme motor je třeba přizpůsobit měnič na konkrétní motor (jmenovitý výkon, prou, napětí, frekvence…). Pro zobrazení všech parametrů, které můžeme měnit je třeba nastavit parametr P0003 na hodnotu 3 (tovární nastavení P0003 = 1 – standardní uživatelská přístupová práva, 3 – pro odborníky). Po nastavení všech důležitých parametrů můžeme vyzkoušet řízení motoru pomocí BOP panelu. Po úspěšném odzkoušení měniče nastavíme parametry P0700 a P1000 na původní hodnotu (P0700 a P1000 = 5 = USS komunikace). Při použití řízení měniče pomocí USS je třeba nastavit přenosovou rychlost sběrnice a nastavení unikátní adresy měniče. Přenosová rychlost sběrnice a adresa musí být stejně nastavena na straně PLC (knihovna USS). Pro náš frekvenční měnič je strop 38400 bps.

P2010	**Přenosová rychlost USS**	**Možná nastavení:**
		3 - 1200 bps
		4 - 2400 bps
		5 - 4800 bps
		6 - 9600 bps
		7 - 19200 bps
		8 - 38400 bps
		9 - 57600 bps

Tab. 2.3 Možná nastavení přenosové rychlosti USS

2.2 PLC SIMATIC S7 224 XP

2.2.1 Základní charakteristika

SIMATIC S7-200 je řada malých programovatelných automatů (mikro-PLC) určených k řízení v jednodušších automatizačních aplikacích. S7-200 sleduje stav vstupů a podle uživatelského programu řídí výstupy. Uživatelský program může obsahovat Booleovu logiku, čítače, časovače, složité matematické operace a komunikaci s jinými inteligentními zařízeními. Kompaktní design, flexibilní konfigurace, výkonný instrukční soubor jsou důvody, proč je S7-200 výborným řešením pro řízení široké škály automatizačních aplikací.



Obr. 2.8 PLC S7 224 XP

Hlavní přednosti zvoleného automatu:

- malý a kompaktní design
- výkonná instrukční sada, pro všechny modely CPU
- systém časových přerušení a přerušení od událostí
- vysokofrekvenční čítače (do 200 kHz) a pulzní výstupy (do 100 kHz)
- možnost rozšíření o další moduly (vstupy/výstupy, funkce)

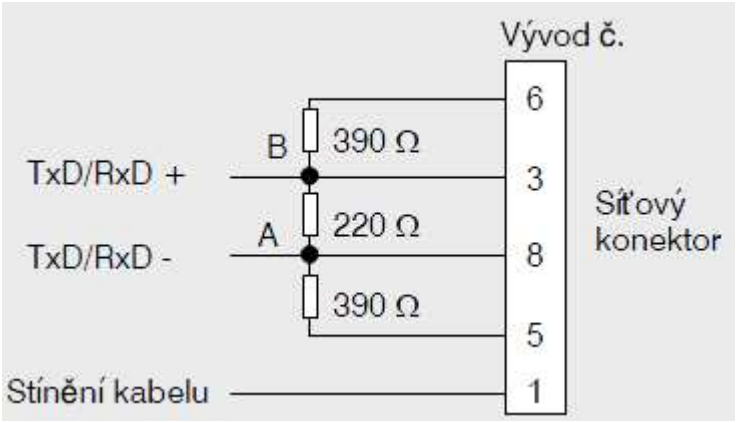
Technické údaje Simatic S7 224 XP:

- integrované digitální vstupy/výstupy: 14/10
- maximální počet digitálních vstupů/výstupů: 94/82
- integrované analogové vstupy/výstupy: 2/1
- maximální počet analogových vstupů/výstupů: 30/15
- paměť pro program: 12 KB a pro data: 10 KB
- vysokorychlostní čítač: celkem 6 čítačů z toho 2 čítače při 200 kHz
- komunikační porty RS 485: 2x
- podporované protokoly: PPI master-slave/MPI slave/Freeport

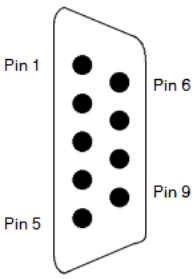
Pro naše použití automatu je nejdůležitější, že obsahuje dva komunikační porty RS 485. Jeden port využijeme pro propojení frekvenčního měniče s automatem a druhý propojení automatu s operátorským panelem nebo při programování s PC. Dalším důležitým prvkem je vysokofrekvenční čítač (200 kHz), který využijeme pro měření otáček a polohy motoru. Motor obsahuje inkrementální čítač s 1024 pulsy na jednu otáčku, proto je potřeba vysoká frekvence vzorkování.

2.2.2 Propojení automatu s frekvenčním měničem

Pro propojení automatu s frekvenčním měničem, bylo zapotřebí zhotovit kabel, protože nebyl součástí setu. Jednalo se o běžný 9 pinový CAN konektor typu samec. Koncová zařízení na sběrnici RS 485 musí být zakončena odporovou zátěží. Zakončení na straně měniče se provádí pomocí přepínače a na straně automatu se zakončení provádí zapojením konektoru podle obrázku 2.9. Takto zakončená sběrnice může měřit až 1200 m. V našem případě, kdy máme propojená jenom dvě zařízení a délka kabeláže je jen několik desítek centimetrů, můžeme zakončovací odpory vynechat. Vynechání zakončovacích odporů na straně automatu, nemá vliv na funkci sběrnice (v našem případě).



Obr. 2.9 Zapojení sběrnice RS 485 na straně automatu.

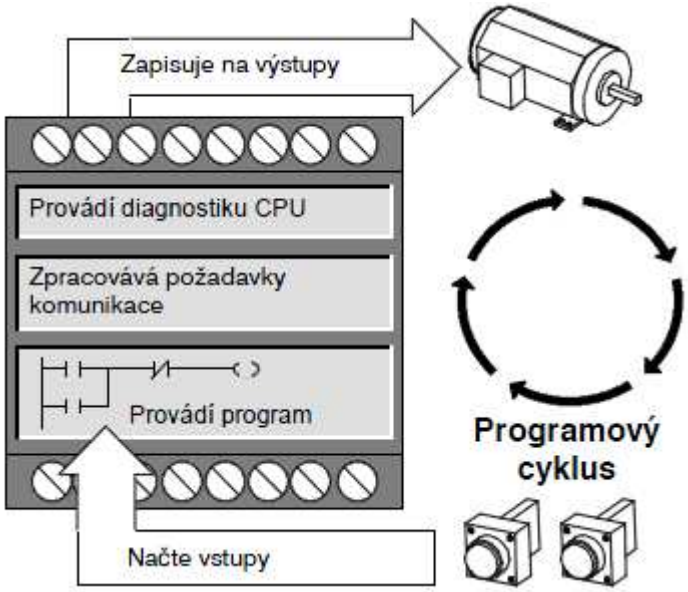
Konektor	Číslo vývodu	PROFIBUS Signal	Port 0/Port 1
	1	Stínění	Uzemnění rozvaděče
	2	24 V zpětný	Logický společný
	3	RS-485 signál B	RS-485 signál B
	4	Výzva k vysílání	RTS (TTL)
	5	5 V zpětný	Logický společný
	6	+5 V	+5 V, 100 Ω sériový odpor
	7	+24 V	+24 V
	8	RS-485 signál A	RS-485 signál A
	9	Nepoužito	10bitová volba protokolu (vstup)
	Pouzdro konektoru	Stínění	Uzemnění rozvaděče

Tab. 2.4 Popis jednotlivých pinů konektoru na straně automatu.

2.2.3 Princip vykonávání řídícího programu

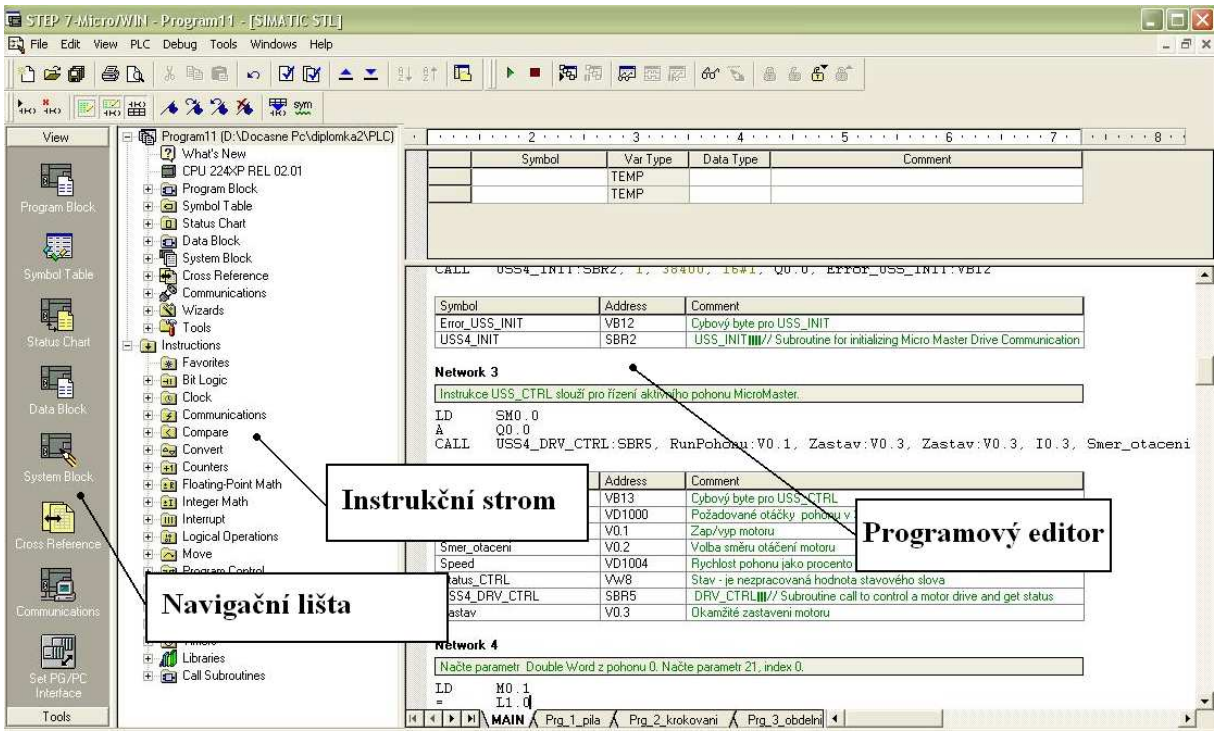
Programovatelný automat cyklicky provádí uložený řídicí program, čte a zapisuje data. S7-200 cyklicky zpracovává řadu úloh. Toto cyklické provádění jednotlivých úloh se nazývá programový cyklus. Jak je znázorněno na obrázku 2.5, S7-200 provádí během programového cyklu všechny následující úlohy nebo jejich větší část:

- Čtení vstupů: S7-200 kopíruje stav fyzických vstupů do registru obrazu vstupů.
- Provádění řídící logiky programu: automat provede instrukce programu a hodnoty uloží do různých oblastí paměti.
- Zpracování požadavků komunikace: automat provede všechny úlohy požadované pro komunikaci.
- Provádění autodiagnostiky CPU: S7-200 kontroluje, zda firmware, paměť pro program pracují správně.
- Zapisování na výstupy: hodnoty uložené v registru obrazu výstupů jsou zapsány na fyzické výstupy.



Obr. 2.5 Programový cyklus automatu.

2.2.4 Vytvoření programu pomocí Step 7 – Micro/WIN



Obr. 2.6 Programové prostředí Step 7 – Micro/WIN

Okno programu Step 7 – Micro/WIN nabízí přehledné prostředí pro tvorbu programů. Horní část programu odpovídá běžnému rozložení aplikací ve Windows. Najdeme zde běžnou nabídku příkazů, které po rozkliknutí nabídnou seznam všech příkazů v dané menu.

Nástrojové lišty obsahují tlačítka pro zkratky k často používaným příkazům menu. Všechny nástrojové lišty můžete zobrazit nebo skrýt. Prostředí pro tvorbu programu je rozděleno na tři části. Jedná se o navigační lištu, instrukční strom a programový editor.

Navigační lišta obsahuje ikony k jednotlivým programovacím prvkům, patří sem například blok programu, tabulka symbolů, datový blok, systémový blok nebo položka pro nastavení komunikace automatu s okolím.

Strom s instrukcemi zobrazuje všechny objekty projektu a instrukce pro tvorbu řídicího programu. Jednotlivé instrukce můžeme do programu vložit pomocí přetáhnutím myší na dané místo nebo můžeme na instrukci dvakrát kliknout, čímž ji vložíte na současnou pozici kurzoru v programovém editoru.

Programový editor obsahuje program a tabulku lokálních proměnných, ve které můžeme přiřadit symbolické názvy dočasným lokálním proměnným. Podprogramy a přerušení jsou zobrazeny jako záložky ve spodní části okna programového editoru. Přecházení mezi hlavním programem, přerušeními a podprogramy, se provádí kliknutím na příslušnou záložku

STEP 7 - Micro/WIN obsahuje tři editory pro vytváření uživatelského programu: kontaktní schémata (LAD), výpis příkazů (STL) a funkční bloky (FBD). S určitými omezeními mohou být programy, psané v kterémkoliv z těchto programových editorů, prohlíženy a editovány ostatními programovými editory.

Popis editoru STL

Editor STL zobrazuje program jako znakově orientovaný programovací jazyk. Umožňuje vytvářet řídicí programy vkládáním textových instrukcí. Editor STL také umožňuje tvorbu programů, které by pomocí editoru LAD nebo FBD nešly vytvořit. Je to proto, že v STL programujete v jazyku S7-200 a nikoli v jazyku grafického editoru, kde platí určitá omezení, aby byly diagramy správně nakresleny. Jak je vidět na obrázku 2.7 je tato znakově orientovaná koncepce velmi podobná programování ve strojovém kódu. S7-200 provádí každou instrukci v pořadí určeném programem shora dolů, pak začne opět odshora. Pro tvorbu programu jsem se rozhodl využít STL editor díky jeho univerzálností a protože prostředí programu je obdobou prostředí u jiných vývojových nástrojů, se kterými jsem již dříve pracoval.

```
//urceni hodnoty e=w-y (regulacni odchylka)
LD      SM0.0
-R      #Y1:LD0, #W1:LD4              // x = w-y
MOVR    #W1:LD4, #E1:LD8              // e := x
```

Obr. 2.7 Ukázka kódu v editoru STL.

Hlavní charakteristiky STL editoru:

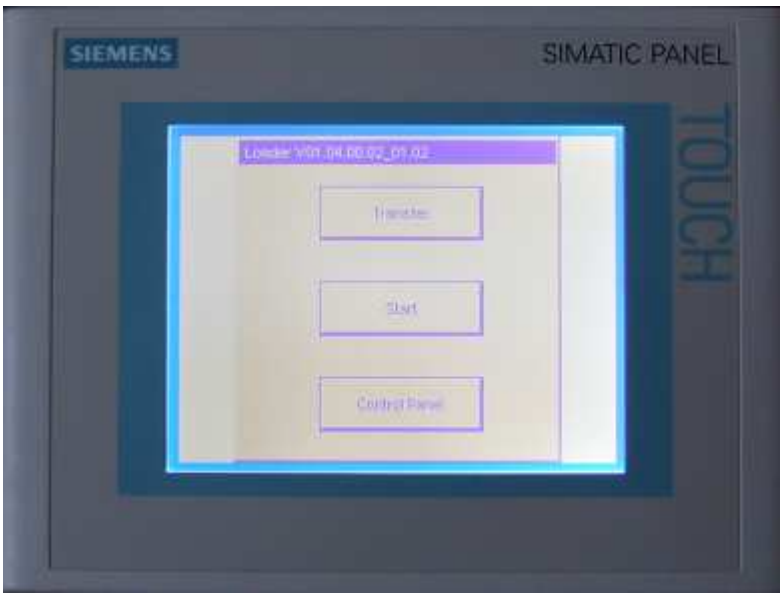
- STL je nejvhodnější pro zkušené programátory.
- STL někdy umožní řešit problémy, které se nedají snadno řešit pomocí editoru LAD a FBD.
- Editor STL můžeme používat pouze s instrukčním souborem SIMATIC.
- Editor STL můžeme vždy použít na prohlížení nebo editaci programu, který byl vytvořen pomocí editorů LAD nebo FBD, naopak to ale není vždy možné.

2.3 Operátorský panel SIMATIC TP 177 micro

2.3.1 Základní charakteristika

Grafický operátorský panel SIMATIC TP 177 micro (obr. 2.10) představuje nejvýkonnější model v řadě micropanelů. Je vybaven dotykovým displejem (odstíny modré) o velikosti 5,7“ a má rozlišení 320 x 240 bodů. Nově tento panel umožňuje využití vektorové grafiky. Panel nabízí uživateli praktické funkce pro všechny požadované úlohy. Systém hlášení umožňuje volně definovat jednotlivá hlášení, zvolit pro ně typ zobrazení a nutnost potvrzení. Panel je nástupcem panelu SIMATIC TP 170 micro. Aplikace pro tento panel se vytvářejí pomocí softwaru WinCC flexible micro.

- Napájecí napětí: 24 V DC.
- Přípustný rozsah: +20.4 V. +28.8 V DC.
- Jmenovitý proud: 0,24 A.
- Paměť:Flash, 256 KB použitelné paměti pro uživatelská data.
- Možnost propojeni: 1 x RS485 (max. 187.5 Mbit/s).
- Podporované sběrnice: RS422, RS 485, RS 232.

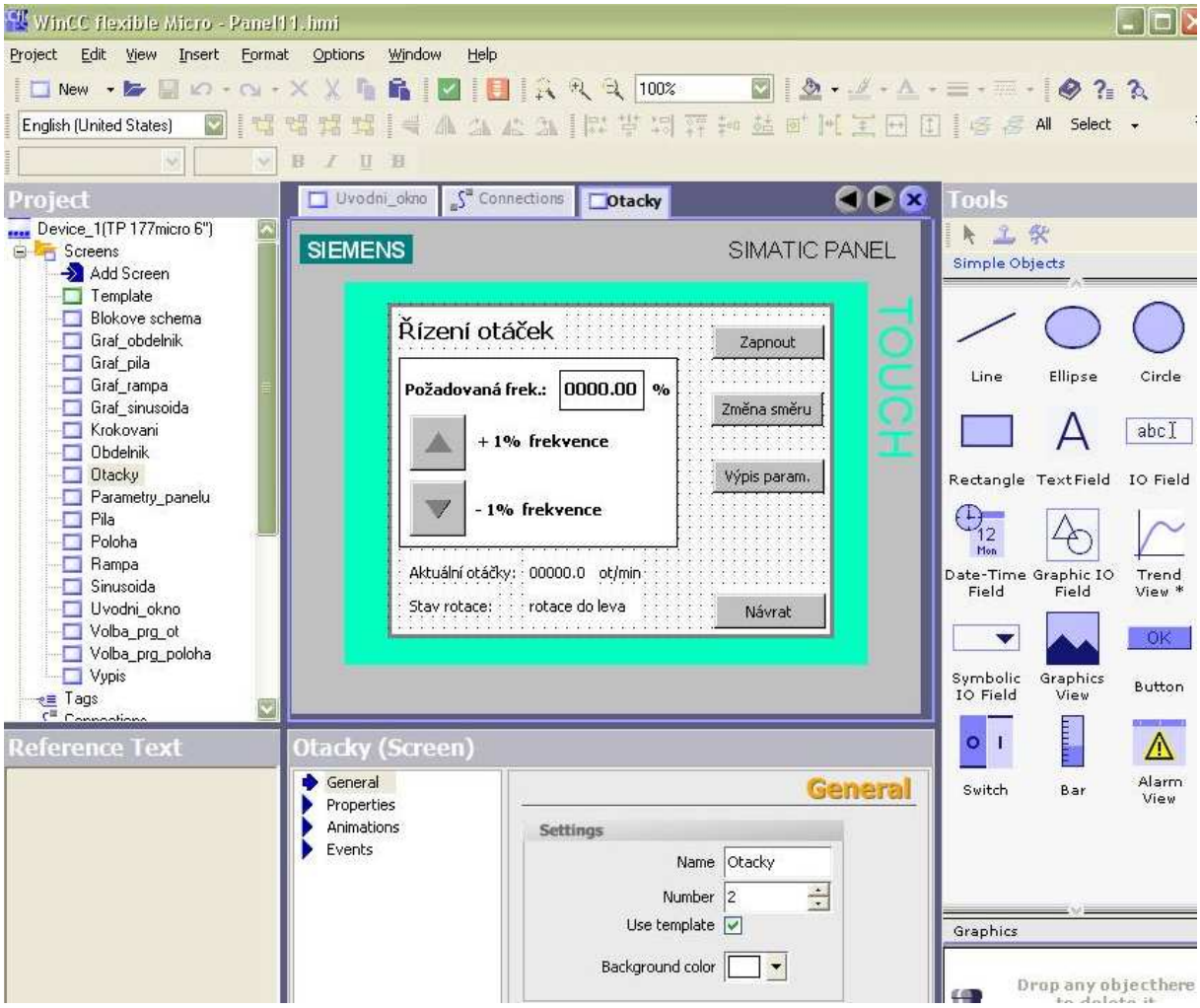


Obr. 2.10 Operátorský panel SIMATIC TP 177 micro.

2.3.2 SIMATIC WinCC flexible 2008 Micro

Simatic WinCC flexible je ideální software pro operátorská rozhraní (Human-Machine Interface – HMI) všude, kde operátoři musí sledovat a řídit technologické procesy. Nezáleží na tom, zda se jedná o nespojitou výrobu nebo automatizaci spojitých procesů. Není také omezen na určité odvětví průmyslu. Je to univerzální otevřený inženýrský software, jenž je určen pro ovládací panely souboru Simatic HMI a monitorovací zařízení – od nejmenších mikropanelů po PC. Verze micro je určena pro mikropanely, komunikující s programovatelnými automaty S7-200. Verzi micro lze upradovat na vyšší verze WinCC Flexible.

Při prvním spuštěním softwaru WinCC flexible Micro se nám objeví nabídka pro vytvoření nového projektu nebo otevření již vytvořeného projektu. Při volbě nového projektu musíme zadat typ panelu, pro který chceme vytvářet aplikaci. My volíme panel TP177micro 6“, který lze provozovat s orientací displeje horizontálně nebo vertikálně (v našem případě budeme panel provozovat horizontálně). Po volbě panelu se nám otevře samotné vývojové prostředí (obr. 2.11). Na levé části programu je umístěn správce objektu,



Obr. 2.11 WinCC flexible Micro, zobrazující již vytvořený projekt.

který obsahuje seznam všech obrazovek panelu, seznam proměnných (Tagy), nastavení režimu komunikace s PLC zařízením, připojených alarmů, jazykových a systémových nastavení. Uprostřed programu se nachází pracovní plocha zobrazující panel. Na tuto pracovní plochu vkládáme jednotlivé objekty (tlačítka, přepínače, grafy, text…). Jednotlivé objekty, které můžeme vkládat na pracovní plochu, najdeme ve správci nástrojů na pravé straně programu. Správce nástrojů obsahuje základní nebo rozšířené objekty (jedná se o různé ikony, značky a obrázky průmyslových zařízení,…). Je zde i možnost naimportovat vlastních grafických objektů (s omezeními pro daný displej). Pod pracovní plochou je umístěn správce vlastností a funkcí objektů. Po kliknutí na daný objekt se objeví možnosti objektu, které můžeme nastavovat. Můžeme např. pojmenovávat objekty, nastavovat text písma, nastavovat pozadí objektu, nebo nastavit funkci pro nějakou událost objektu (kliknutí na tlačítko, přepnutí přepínače,…). Samotná obrazovka pro vkládání objektů, je tvořena 32 vrstvami. Můžeme nastavovat, ve které vrstvě bude daný objekt zobrazen a tak dosáhnout překrývání objektů, které se podle potřeby mohou zobrazit nebo skrýt. Speciální typem obrazovky je Template, která objekty na této obrazovce umožňuje zobrazit na všech obrazovkách, které mají povolené použití obrazovky Template. Na obrazovce Template můžeme např. vložit rámeček, který je pak zobrazen ve všech obrazovkách.

Proměnné – Tagy:

WinCC Flexible využívá systém proměnných, které se nazývají tagy. Tag může být bud' interní, vytvořený objektem a fungující jen v rámci uživatelského rozhraní, nebo externí, který je závislý na komunikaci s PLC. Nejdůležitější vlastností tagu je jeho hodnota. Externí tagy mohou být všech datových typů, které nabízí PLC. Jsou to:

- Char
- Byte
- Integer
- Word
- Double Integer
- Double Word
- Real
- Boolean
- StringChar

Objekty

WinCC Flexible využívá při tvorbě programu objektový přístup. Všechny komponenty umístěné ve správci nástrojů mají své vlastnosti, jak už jsem se zmiňoval. Objekty lze rozdělit na statické a dynamické. Statické se v průběhu vykonávání programu nemění, jedná se například o objekty typu čára, kružnice, elipsa, obdélník nebo textové pole. Tyto objekty mají jen základní vlastnosti jako název, vzhled, vrstvu. Dynamické objekty umožňují zobrazovat hodnoty, vykreslovat průběh hodnoty, upravovat svůj vzhled. Dále umožňuje reagovat na různé události, které vyvolá projekt nebo samotný proces (např. událost na kliknutí tlačítka, kdy dojde k přepnutí obrazovky na jinou). Ve svém projektu jsem používál následující objekty:

- I/O pole – slouží k zobrazování a zadávání požadovaných dat (např. požadované otáčky motoru). Můžeme je nastavit na vstupní, výstupní nebo vstupně/výstupní kdy je umožněno zadávat požadované data a současně je číst.
- Symbolické I/O pole – umožňuje výběr položky ze seznamu textů, nebo její zobrazení v závislosti na stavu procesu.
- Tlačítka – při stisku se aktivuje a po uvolnění deaktivuje.
- Přepínač – umožňuje přepínat mezi dvěma stavy
- Graf – umožňuje vykreslovat průběh číselné proměnné.

2.4 Napájecí zdroj LOGO! Power 24 V

2.4.1 Základní charakteristika

Jedná se o napájecí zdroj rodiny LOGO!, která představuje zdroje kompaktních rozměrů s volitelnými výstupními napětí a proudů. Řada zdrojů Power 24V je nabízena ve třech variantách, které se od sebe odlišují výstupním proudem (1,3A, 2,5A, 4A) a rozměry. Náš konkrétní zdroj pracuje s výstupním proudem 1,3A, jedná se tedy o nejnižší variantu, která je však pro naše použití plně dostačující. Vstupní napětí zdroje lze provozovat v rozsahu od 85 do 264 V, jsou tedy vybaveny pro každou síť na světě. Na přední části zdroje je umístěna LED dioda signalizující, zda je výstupní napětí v pořádku. Při přetížení a zkratu dodávají primární spínací regulátory konstantní proud, to znamená bez pokusů o opakovaný start. Stupňovitý profil stavební konstrukce, minimální montážní hloubka a montáž na normovanou profilovou lištu umožňují bezproblémovou montáž do instalačních malých rozdělovačů, ale lze je univerzálně použít také v oblasti průmyslu.

2.4.2 Technická data zdroje LOGO!Power 24 V

Rozměry:	**(Š x V x H) 54 x 90 x 55 mm**
Číslo výrobce:	6EP1331-1SH02
Počet výstupů:	1
Výstupní výkon:	32 W
Výstupní napětí:	24V/DC
Výstupní proud:	1,3 A
Rozsah vstupního napětí:	85 - 264 V/AC
Hmotnost:	170 g

Tab. 2.5 Technické data zdroje LOGO!Power 24 V.

2.5 Asynchronní motor s inkrementálním snímačem

2.5.1 Základní charakteristika asynchronního motoru

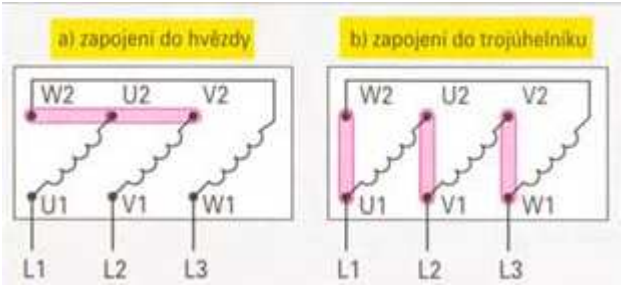
Asynchronní motor je točivý elektrický stroj (elektromotor) pracující na střídavý proud. Je to nejrozšířenější pohon v elektrotechnice vůbec. Tok energie mezi hlavními částmi motoru (stator a rotor) je realizován výhradně pomocí elektromagnetické indukce, proto se často tento motor označuje jako motor indukční. Výhodou asynchronního motoru je vysoká spolehlivost, jednoduchá konstrukce a napájení z běžné střídavé sítě. Napájecí napětí může být jednofázové nebo trojfázové, přičemž trojfázové je výrazně používanější. V našem případě používáme trojfázový motor od společnosti SIEMENS.

2.5.2 Technické specifikace trojfázového asynchronního motoru SIEMENS

Jmenovité elektrické a mechanické údaje motoru jsou uvedeny na výkonovém štítku, který je zobrazen na obrázku 2.12. Napájecí napětí a jmenovitý proud závisí na tom, zdali je motor zapojen do hvězdy, nebo do trojúhelníku. Motor může pracovat na různých kmitočtech sítě (50/60 Hz). Náš motor je zapojen do trojúhelníku se jmenovitým kmitočtem 50 Hz (V = 3x 230V AC, I = 1,34 A).



Obr. 2.12 Štítek trojfázového asynchronního motoru SIEMENS.



Obr. 2.13 Zapojení asynchronního motoru Δ/Y.

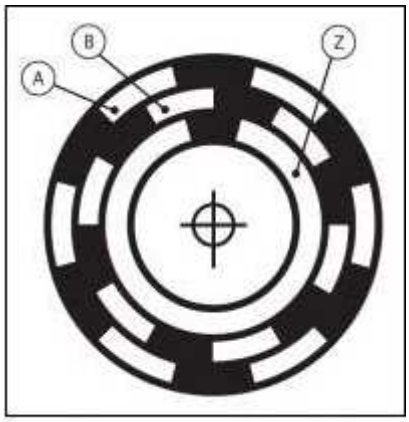
2.5.3 Inkrementální snímač

Jádrem snímače je otočný optický disk, který je mechanicky spojen s připojovací hřídelí snímače (obr. 2.14). Infračervené světlo vysílané světelným zdrojem (LED), jenž je umístěn před diskem může procházet pouze průhlednými okénky; zbývající část disku je neprůhledná a světlo pohlcuje. Světlo dále prochází clonkami, které mají stejnou rozteč jako okénka na disku. Světelné impulsy dopadají na optický detektor, který je převádí na elektrické impulsy. Pilový elektrický signál je dále zesílen a elektronicky tvarován na obdélníkový. Pro zlepšení kvality a stability výstupních signálů se snímá vždy v diferenčním režimu, kdy se porovnávají dva totožné signály s opačnou fází (fázový rozdíl je 180° el.). Diferenční způsob snímání umožňuje eliminovat vliv rušení.



Obr. 2.14 Princip činnosti snímače.

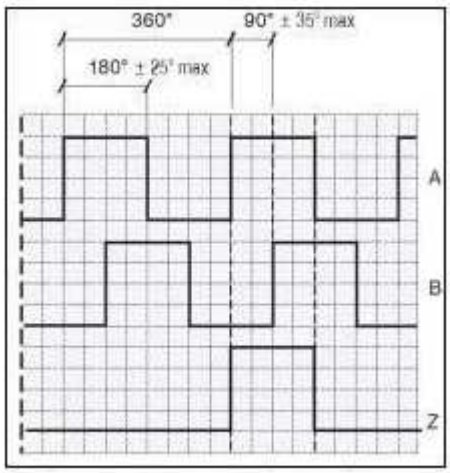
Pro rozlišení směru otáčení jsou na pevné části dvě soustavy clonek vzájemně posunuté o 90° (el.). Popsaným způsobem tedy lze získat dva obdélníkové signály posunuté o polovinu periody: kanál A a kanál B (obr. 2.15). Zpracováním samotného kanálu A je možné získat informaci o rychlosti otáčení, pomocí druhého kanálu lze rozeznat i směr otáčení. K dispozici je ještě další signál, nazývaný Z nebo také nulový (referenční) kanál, který udává nulovou (referenční) polohu hřídele snímače. Tento signál je obdélníkový a je ve fázi s kanálem A. Pro naše použití nám postačují signály A a B. Kdy za pomocí těchto signálů jsem schopni určit rychlost i směr otáčení motoru. Signál Z zůstane nevyužit. Výstupní signál snímače při otáčení hřídele ve směru hodinových ručiček je zobrazen na obrázku 2.17.



Obr. 2.15 Optický disk.

Technické specifikace snímače 1XP8001-1:

- 1024 pulsů na jednu otáčku hřídele.
- HTL logika
- Vstupní napětí od 10V do 30V
- 3 výstupní signály (A, B, Z)



Obr. 2.17 Výstupní signály A, B a Z.

2.5.4 Ukázka zapojení celého setu

Na závěr této kapitoly přidávám fotografii všech komponent, které už jsou vzájemně zapojené a jsou v provozním režimu.



Obr. 2.18 Ukázka zapojení všech komponent v provozním režimu.

3 Knihovna USS

3.1 K čemu slouží knihovna USS?

Knihovny instrukcí STEP 7-Micro/WIN usnadňují řízení pohonu MICROMASTER, protože obsahují již nakonfigurované podprogramy přerušení a jiné podprogramy, které jsou speciálně navrženy pro použití USS protokolu při komunikaci s pohonem. Pomocí instrukcí USS můžeme řídit fyzický pohon a číst, popřípadě zapisovat jeho parametry.

Knihovna USS není součástí programovacího prostředí STEP 7-Micro/WIN a je tedy nutné zakoupit knihovnu zvlášť. Škola vlastní knihovnu USS4, která je určená pro frekvenční měniče MICROMASTER řady 4. Z tohoto důvodu jsem musel použít knihovnu USS4, ale nejdřív jsem musel ověřit, zdali bude spolupracovat s frekvenčním měničem SINAMICS G110. Po komunikaci s technickou podporou SIEMENS mi bylo sděleno, že danou knihovnu můžu použít pro daný frekvenční měnič, jenom musím provést změny v nastavení.

3.2 Požadavky a omezení při použití knihovny USS

Knihovny instrukcí ve STEP 7 – Micro/WIN obsahují 14 podprogramů, 3 podprogramy přerušení a 8 instrukcí, které podporují USS protokol. Instrukce USS používají v automatu následující prostředky:

- Inicializace USS protokolu vyhradí port 0 pro komunikace USS.
  Instrukce USS_INIT se používá pro volbu USS nebo PPI pro port 0. Poté co zvolíme pro komunikaci s pohony použití USS protokolu nemůžeme port 0 používat pro žádný jiný účel včetně komunikace se STEP 7 – Micro/WIN.

- Instrukce USS ovlivňují všechna místa paměti SM spojená s komunikací Freeport na portu 0.

- Díky instrukcím USS se zvětšuje část paměti potřebná pro uživatelský program. V závislosti na konkrétních použitých instrukcích USS mohou podpůrné programy pro tyto instrukce zvýšit režii pro řídící program nejméně o 2300 bytů, až na 3600 bytů.

- Proměnné pro instrukce USS potřebují 400 bytový blok v paměti V. Počáteční adresa tohoto bloku je přidělena uživatelem a je rezervována pro proměnné USS.

- Některé z instrukcí USS také vyžadují 16 bytový komunikační zásobník. Uživatel přiděluje počáteční adresu v paměti V pro tento zásobník jako parametr instrukce. Doporučuje se přiřadit vlastní zásobník každé instrukci USS.

- Při provádění výpočtů používají instrukce USS akumulátory AC0 až AC3. V našem programu můžeme také použít akumulátory, ale hodnoty v akumulátorech budou instrukcemi USS změněny.

- Instrukce USS není možné použít v podprogramu přerušení

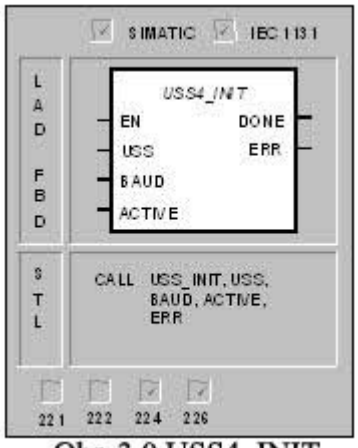
3.3 Použití instrukcí USS

Pro použití instrukcí USS protokolu v programu programovatelného automatu S7 – 200 je třeba provést tyto kroky:

1. Vložíme do programu instrukci USS_INIT, kterou provádíme jen po dobu jednoho programového cyklu. Instrukci USS_INIT lze použít buď pro inicializaci nebo pro změnění parametru komunikace USS. Po vložení instrukce USS_INIT se k našemu programu přidá automaticky několik skrytých podprogramů a podprogramů přerušení.
2. Pro každý aktivní pohon vložíme do svého programu pouze jednu instrukci USS_CTRL. Instrukce USS_RPM_x a USS_WPM_x můžeme vložit tolikrát, kolikrát potřebujeme, ale pouze jedna z nich může být v určitou dobu aktivní.
3. Přidělíme paměť V pro knihovnu instrukcí.
4. Nakonfigurujeme parametry pohonu tak, aby odpovídaly přenosové rychlosti a adrese použité programem.
5. Zapojení komunikačního kabelu mezi automatem S7 – 200 a pohonem. Zařízení se mezi sebou propojují pomocí krátkého silného kabelu se stejným uzemněním nebo nulovým bodem obou zařízení.

3.4 Instrukce USS4_INIT

Instrukce USS_INIT (obr. 3.0) se používá pro povolení a inicializaci nebo pro ukončení komunikace s pohony MicroMaster. Instrukce USS4_INIT musí být dokončena bez chyb před použitím jakékoliv jiné instrukce USS. Dříve než může být provedena další instrukce musí být prováděná instrukce ukončena a nastaven bit ”DONE”.



Obr. 3.0 USS4_INIT

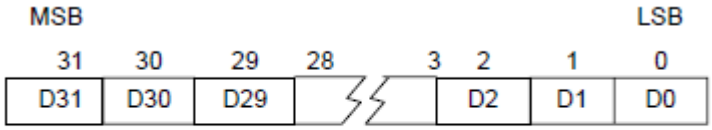
Tato instrukce je provedena v každém programovém cyklu, když je zapnutý vstup EN. Instrukci USS4_INIT provádíme pro každou změnu ve stavu komunikace pouze jednou. Proto by vstup EN měl být zapojený pulzně přes vyhodnocující náběžnou hranu. Chceme-li změnit parametr inicializace, provedeme novou instrukci USS4_INIT.

Parametr ”USS” volí komunikační protokol: vstupní hodnota 1 přiřazuje port 0 USS protokolu a aktivuje protokol, vstupní hodnota 0 přiřazuje port 0 protokolu PPI a blokuje USS protokol. ”BAUD” nastavuje přenosovou rychlost na 1200, 2400, 4800, 9600, 19200 nebo 38400. ”ACTIVE” indikuje, které pohony jsou aktivní. Některé pohony podporují pouze adresy 0 až 30.

Vstupy/Výstupy	Operandy	Typ dat
USS	VB, IB, QB, MB, SB, SMB, LB, AC, konstanta, *VD, *AC, *LD	BYTE
BAUD	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, konstanta, AC *VD, *AC, *LD	WORD
ACTIVE	VD, ID, QD, MD, SD, SMD, LD, AC, konstanta, *VD, *AC, *LD	DWORD
DONE	I, Q, M, S, SM, T, C, V, L	BOOL
ERR	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE

Tab. 3.0 Parametry instrukce USS4_INIT

Obrázek 3.1 ukazuje popis a formát vstupu aktivního pohonu**.** Kterýkoliv pohon označený jako ”ACTIVE” je automaticky dotazován na pozadí tak, aby řídil pohon, shromažďoval stav a zabránil časovému odpojení sériové linky pohonu.



D0 Bit aktivity pro pohon 0; 0 – pohon není aktivní, 1 – pohon aktivní
D1 Bit aktivity pro pohon 1; 0 – pohon není aktivní, 1 – pohon aktivní

Obr. 3.1 Formát parametru aktivního pohonu.

Když je dokončena instrukce USS4_INIT, zapne se výstup ”DONE”. Výstupní byte ”ERR” obsahuje výsledek provádění této instrukce. Tabulka 3.1 definuje chyby, které mohou vyplynout z provedení této instrukce.

Network 2

Instrukce USS4_INIT slouží pro povolení a inicializaci nebo pro ukončení komunikace s pohony MicroMaster.

```
LD      SM0.1
CALL    USS4_INIT:SBR2, 1, 38400, 16#1, Q0.0, Error_USS_INIT:VB12
```

Obr. 3.1 Volání instrukce USS4_INIT v STL editoru.

Popis instrukce:

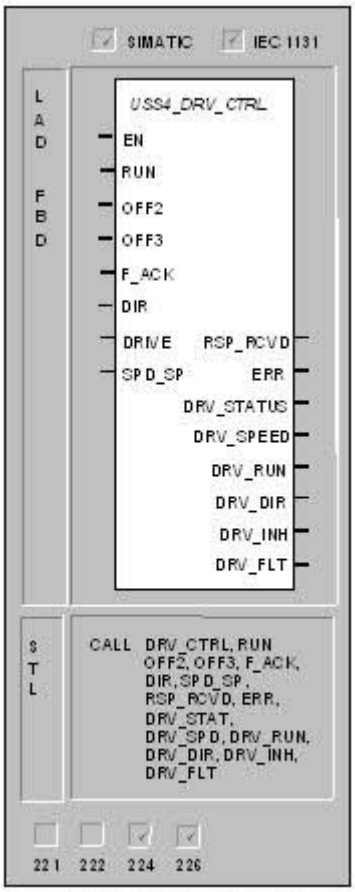
USS = 1 - přiřadí port 0 USS protokolu
BAUD = 38400 - nastaví přenosovou rychlost na 38400 bps
ACTIVE = 16#1 – pohon 0 je aktivní
DONE = Q0.0 – výstup Q0.0 automatu signalizuje dokončení instrukce USS_INIT
ERR = VB12 – paměti VB12 obsahuje chyby, které mohou nastat při provádění této instrukce

Na co si dát pozor při provádění instrukce USS4_INIT:

Při konfiguraci USS_INIT instrukce je důležité mít nastavenou správnou přenosovou rychlost. Rychlost musí odpovídat nastavené rychlosti na frekvenčním měniči. Současně musí být podporována hardwarem (automat, frek. měnič). V našem případě je maximum 38400 bps, kdy jsme omezení USS4 instrukcí, protože frekvenční měnič zvládá přenosovou rychlost až 57600 bps. Dalším důležitým parametrem je „Active“, kde je potřeba zadat číslo pohonu, který má být aktivní. Číslo pohonu v USS_INIT instrukci musí být stejné jaké je nastavené na straně frekvenčního měniče a současně musí být zadáno ve správném formátu. Já jsem při nastavení tohoto parametru udělal chybu, když jsem nastavil „Active“ na nulu (aktivní pohon 0). S tímto nastavením se instrukce USS_INIT provedla a potom i další instrukce. Vše fungovalo správně, až na to, že odezva frekvenčního měniče na příkaz z automatu byla příliš velká (pohybovala se v rozsahu 1 až 2 s). Taková odezva byla nevyhovující pro rozumné řízení. Při hledání chyby jsem zkoušel nastavovat různé přenosové rychlosti, měnit délku procesních dat PZD sériové linky USS, nastavování komunikace na straně operátorského panelu, nastavování podílu času pro komunikaci, ale nic tento problém neodstranilo. Po opětovném pročtení instrukce USS_INIT, jsem našel chybu na straně parametru „Active“. Po nastavení aktivního pohonu 0 ve správném formátu, se odezva pohybuje v řádu několika desítek ms.

3.5 Instrukce USS4_DRV_CTRL

Instrukce USS4_DRV_CTRL (obrázek 3.2) se používá na řízení aktivního pohonu MicroMaster. Tato instrukce uloží vybrané příkazy do komunikačního zásobníku, který je pak odeslán adresovanému pohonu (parametr ”Drive” - pohon), pokud byl tento pohon vybrán v parametru ”Active” instrukce USS4_INIT. Každému pohonu by měla být přiřazena pouze jedna instrukce USS4_DRV_CTRL.



Obr. 3.2 USS4_DRV_CTRL

Aby byla povolena instrukce USS4_DRV_CTRL, musí být zapnutý bit EN. Tato instrukce by měla být vždy povolena.
RUN (RUN/STOP) indikuje, zda je pohon zapnutý (1) nebo vypnutý (0). Když je bit RUN zapnutý, dostane pohon MicroMaster příkaz, aby se rozběhl se zadanou rychlostí a směrem. Aby byl pohon uveden do chodu, musí platit následující:

- Pohon musí být vybrán jako ”Active” v USS4_INIT.
- OFF2 a OFF3 musí být nastaveny na 0.
- ”DRV_FLT” (Porucha) a ”DRV_INH” (Blokování) musí být 0.

Když je bit RUN vypnutý, je pohonu MicroMaster poslán příkaz, aby zastavil se zpomalením. Bit OFF2 se používá, aby umožnil pohonu MicroMaster volně doběhnout až do zastavení. Bit OFF3 se používá, jestliže chceme pohonu MicroMaster přikázat, aby zastavil okamžitě.

Bit RSP_RCVD (Odpověď přijata) potvrzuje odpověď pohonu. Všechny aktivní pohony jsou cyklicky dotazovány na aktuální informace o stavu pohonu. Pokaždé, když automat S7-200 dostane odpověď od pohonu, zapne se bit RSP_RCVD po dobu jednoho programového cyklu a všechny hodnoty jsou aktualizovány.

Bit F_ACK (Potvrzení poruchy) se používá pro potvrzení poruchy pohonu. Pohon vymaže poruchu (Fault), když F_ACK přejde z 0 na 1.

Bit DIR (Směr) ukazuje, kterým směrem by se měl pohon pohybovat.

Vstupy/Výstupy	Operandy	Typ dat
RUN, OFF2, OFF3, F_ACK, DIR	I, Q, M, S, SM, T, C, V, L, signálový tok	BOOL
DRIVE	VB, IB, QB, MB, SB, SMB, LB, AC, konstanta, *VD, *AC, *LD	BYTE
SPD_SP	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD, konstanta	REAL
RSP_RCVD	I, Q, M, S, SM, T, C, V, L	BOOL
ERR	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE
DRV_STATUS	VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD	WORD
DRV_SPEED	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD	REAL
DRV_RUN, DRV_DIR, DRV_INH, DRV_FLT	I, Q, M, S, SM, T, C, V, L	BOOL

Tab. 3.2 Parametry instrukce USS4_DRV_CTRL

Vstup ”Drive” (Adresa pohonu) je adresa pohonu MicroMaster, na kterou má být poslán příkaz USS4_DRV_CTRL. Platné adresy: 0 až 31.

“SPD_SP” (Nastavená hodnota rychlosti) je rychlost motoru jako procento plných otáček. Záporné hodnoty Speed_SP způsobí, že se pohon bude pohybovat v opačném směru. Dosah: -200,0 % až 200,0 %.

”ERR” (Chyba) je chybový byte, který obsahuje výsledek posledního požadavku na komunikaci s modulem. Tabulka 3.1 definuje chyby, které mohou vyplynout z provedení této instrukce.

”DRV_STATUS” (Stav) je nezpracovaná hodnota stavového slova odeslaného zpět pohonem.

”DRV_SPEED” (Rychlost) je rychlost pohonu jako procento plných otáček. Dosah: -200,0 % až 200,0 %.

” DRV_RUN” (Povolení RUN) indikuje, zda je pohon v chodu (1), nebo vypnutý (0).

” DRV_DIR” indikuje směr pohybu pohonu.

”DRV_INH” indikuje stav blokovacího bitu pro pohon (0 - není blokován, 1 - blokován). Chceme-li vymazat blokovací bit, musí být vypnutý bit ”Fault”; vstupy RUN, OFF2 a OFF3 musí být také vypnuté.

”DRV_FLT” indikuje stav bitu pro poruchu (0 - bez poruchy, 1 - porucha). Poruchový kód odráží stav pohonu (Viz příručku pro pohon). Pro vymazání bitu ”DRV_FLT”, je nutné opravit příčinu poruchy a nastavte bit F_ACK. Výpis všech parametrů a jejich datové typy najdeme v tabulce 3.2.

Ukázka instrukce USS4_DRV_CTRL:

```
LD    SM0.0
A     Q0.0
CALL  USS4_DRV_CTRL:SBR5, V0.1, V0.3, V3.0, I0.3, V0.2, 0, VD1000, M5.0, VB13,
                    VW8, VD1004, Q0.1, Q0.2, Q0.3, Q0.4
```

Popis instrukce:

Tato instrukce je aktivní, až je dokončena instrukce USS4_INIT. Výstup Q0.0 signalizuje dokončení instrukce USS4_INIT.
V0.1 (RUN) = log.1 – pohon se rozeběhne požadovanými otáčkami a směrem.
V0.3 (OFF2) – volné doběhnutí motoru (ve svém programu nevyužívám).
V3.0 (OFF3) – okamžité zastavení motoru (ve svém programu nevyužívám).
I0.3 (F_ACK) – přechod z log. 0 do log. 1 vymaže poruchu (ve svém programu nevyužívám).
V0.2 (DIR) – provádí změnu rotace motoru.
0 – adresa pohonu (DRIVE)
VD1000 (SPD_SP) – obsahuje požadovanou rychlosti otáčení motoru.
M5.0 (RSP_RCVD) potvrzuje odpověď pohonu (ve svém programu nevyužívám).
VB13 (ERR) – obsahuje chybu, která může nastat pro tuto instrukci (využíval při ladění programu).
VW8 (DRV_STATUS) – stavové slovo odeslané zpět pohonem (ve svém programu nevyužívám)
VD1004 (DRV_SPEED) – aktuální rychlost otáčení motoru.
Q0.1 (DRV_RUN) – indikace chodu motoru.
Q0.2 (DRV_DIR) – indikace směru pohonu.
Q0.3 (DRV_INH) – indikace stavu blokace motoru (0 – není blokován, 1 - blokován).
Q0.4 (DRV_FLT) – indikace poruchy (0 – bez poruchy, 1 - porucha).

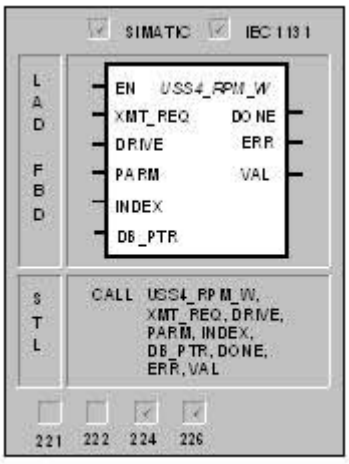
Na co si dát pozor při provádění instrukce USS4_DRV_CTRL:

Při zprovozňování této instrukce je třeba si dát pozor na zadání správného čísla pohonu a na zadání požadovaných otáček motoru ve správném tvaru (např. 50% otáčky se zadávají jako 50.0). Jinak tato instrukce obsahuje spoustu parametrů, které není potřeba využívat. Problém který může nastat, je při využívání oficiálního manuálu: manual_s7_200_2004_cz, kde jsou popsané instrukce USS, které se od instrukcí USS4 trochu odlišují. Např. instrukce USS_CTRL má o jeden parametr navíc než instrukce USS4_DRV_CTRL. Jedná se o parametr “Type”, kde volíme o jaký typ pohonu se jedná. Jelikož knihovna USS4 je prvotně určena pro pohon MicroMaster 4 je tento parametr vynechán. Při použití univerzální USS knihovny parametr “Type” může obsahovat (je to moje domněnka).

3.6 USS4_RPM_x

V protokolu USS existují tři instrukce pro čtení:

- Instrukce USS_RPM_W (obr. 3.3) čte parametr word bez znaménka.
- Instrukce USS_RPM_D čte parametr double word bez znaménka.
- Instrukce USS_RPM_R čte parametr s plovoucí desetinnou čárkou.



Obr. 3.3 USS4_RPM_W

V jednu dobu může být aktivní pouze jedna instrukce pro čtení (USS4_RPM_x) nebo zápis (USS4_WPM_x).
Transakce USS4_RPM_x je ukončena, když pohon MicroMaster potvrdí příjem příkazu, nebo když je ohlášena chyba. Programový cyklus se nadále provádí i v době, kdy tento proces čeká na odezvu. Aby byl povolen přenos požadavku, musí být bit EN zapnutý a musí zůstat zapnutý do nastavení bitu ”Done”, který signalizuje ukončení procesu. Například požadavek USS4_RPM_x je vysílán pohonu MicroMaster v každém programovém cyklu, kdy je zapnutý vstup XMT_REQ. Proto by měl být vstup XMT_REQ spouštěn pulzem od náběžné hrany, aby byl požadavek vyslaný pouze jednou v každém kladném přechodu vstupu EN.

Vstup ”DRIVE” je adresa pohonu MicroMaster, na kterou má být poslán příkaz USS4_RPM. Platné adresy jednotlivých pohonů jsou 0 až 31.

”PARM” je číslo parametru. ”INDEX” je indexová hodnota parametru, který má být načten. ”VAL” je vrácená hodnota parametru. Adresa 16bytového zásobníku musí být zadána na vstup DB_PTR. Tento zásobník používá instrukce USS4_RPM_x na uložení výsledku příkazu daného pohonu MicroMaster.
Po ukončení instrukce USS4_RPM_x se zapne výstup ”DONE” a výstupní byte ”ERR” a výstup ”VAL” obsahují výsledky provedení instrukce. Tabulka 3.1 definuje chyby, které mohou vyplynout z provedení této instrukce. Výstupy ”ERR” a ”VAL” nejsou platné dokud se nezapne výstup ”DONE”. Přehledný výpis všech parametrů a jejich platné operandy zobrazuje tabulka 3.3.

Vstupy/Výstupy	Operandy	Typ dat
XMT_REQ	I, Q, M, S, SM, T, C, V, L, signálový tok podmíněné detekcí náběžné hrany	BOOL
DRIVE	VB, IB, QB, MB, SB, SMB, LB, AC, konstanta, *VD, *AC, *LD	BYTE
PARM	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, konstanta, AC *VD, *AC, *LD	WORD
INDEX	VW, IW, QW, MW, SW, SMW, LW, T, C, AIW, konstanta, AC *VD, *AC, *LD	WORD
DB_PTR	&VB	DWORD
DONE	I, Q, M, S, SM, T, C, V, L	BOOL
ERR	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD	BYTE
VAL	VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD	WORD, DWORD, REAL

Tab. 3.3 Operandy platné pro USS4_RPM_x

Ukázka instrukce USS4_RPM_R:

```
Network 4
Načte parametr Double Word z pohonu 0. Načte parametr 21, index 0.
LD      M0.1
=       L1.0
LD      M0.1
EU
=       L2.0
LD      L1.0
CALL    USS4_RPM_R:SBR10, L2.0, 0, 21, 0, &VB1300, Q1.1, VB14, VD1102
```

Obr. 3.4 Ukázka instrukce USS4_RPM_R s prostředí STEP 7-Micro/WIN v STL editoru.

Popis instrukce:
Pro povolení instrukce musí být bit L1.0 (EN) zapnutý a to doby, než Q1.1 (DONE) signalizuje dokončení procesu čtení. Vstup L2.0 (XMT_REQ), je aktivní jen když vstup M0.1 přejde s log. 0 do log. 1 (aktivní s náběžnou hranou). Stav M0.1 řídíme v podprogramu pro čtení parametrů, který si později popíšeme. Čtení se provádí s pohonu 0 (Parametr DRIVE = 0) a čteme skutečnou hodnotu kmitočtu (PARM = 21). Indexová hodnota parametru je nastavena na nulu (INDEX = 0). VB1300 představuje počáteční adresu 16 bytového zásobníku (DB_PTR = &VB1300). Do proměnné VB14 se ukládá chyba, která může při vykonávání této instrukce vyplynout (ERR = VB14). Hodnota přečteného parametru se uloží do proměnné VD1102 (VAL = VD1102).

Na co si dát pozor při provádění instrukce USS4_RPM_x:

Při provádění instrukce pro čtení je zapotřebí, aby před zavoláním další instrukce pro čtení byla dokončena předcházející. Pokud bychom volali funkci v krátkých intervalech (řádově několika ms), tak by nastala chyba s číslem 8. Je tedy nutné kontrolovat dokončení instrukce pomocí bitu “DONE”, nebo nastavení doby intervalu volání funkce větší, než je doba trvání samotné funkce. Tímto zajistíme bezproblémový chod funkce pro čtení. Dále si musíme dávat pozor na zadání správného parametru pro čtení “PARM” a správnou adresu pohonu.

Obdoba instrukce pro čtení (USS4_RPM_x) existuje instrukce pro zápis (USS4_WPM_x) do frekvenčního měniče. Tuto instrukci ve svých programech nevyužívám, tak proto ji nebudu popisovat.

1.1 Chybové kódy pro provádění USS

Chybové kódy	Popis
0	Bez chyby
1	Pohon neodpověděl
2	Byla zjištěna chyba kontrolního součtu v odpovědi pohonu
3	Byla zjištěna chyba parity v odpovědi pohonu
4	Chyba způsobená zásahem uživatelského programu
5	Pokus o nepřípustný příkaz
6	Zadána nepřípustná adresa pohonu
7	Komunikační port nebyl nastaven pro USS protokol
8	Komunikační port je zaneprázdněn zpracováváním instrukce
9	Vstupní rychlost pohonu je mimo rozsah
10	Nesprávná délka odpovědi pohonu
11	Nesprávný první znak v odpovědi pohonu
12	Znak délky v odpovědi pohonu není podporován instrukcemi USS
13	Odpověděl nesprávný pohon
14	Zadaná adresa DB_Ptr je nesprávná
15	Zadané číslo parametru je nesprávné
16	Byl vybrán neplatný protokol
17	USS je aktivní; změna není povolena
18	Byla specifikována neplatná přenosová rychlost
19	Nekomunikuje se: pohon není AKTIVNÍ
20	Parametr nebo hodnota v odpovědi pohonu jsou nesprávné nebo obsahují chybový kód
21	Namísto požadované hodnoty word byla vrácena hodnota double word
22	Namísto požadované hodnoty double word byla vrácena hodnota word

Tab. 3.1 Chybové kódy při běhu instrukcí USS.

3.7 Vytvoření knihovny USS

Ještě bych chtěl poznamenat, že knihovna USS4 nebyla ve formě knihovny, ale ve formě programu, proto bylo nutné vytvořit z tohoto programu knihovnu. Postup pro vytvoření knihovny je následující:

1. Napíšeme program jako standardní projekt STEP 7-Micro/WIN a funkci, která má být zahrnuta do knihovny, vložíme do podprogramů, nebo přerušení.
2. Zkontrolujeme, že všechna místa paměti V, v pod-programech nebo přerušeních mají přiřazen symbolický název. Abychom minimalizovali velikost paměti V, kterou knihovna potřebuje, použijeme za sebou jdoucí místa paměti V.
3. Přejmenujeme podprogramy a přerušení tak, aby měly názvy, které chceme mít v knihovně instrukcí.



Obr. 3.5 USS knihovna

Pro kompilování nové knihovny instrukcí použijeme příkaz menu **Soubor > Vytvořit knihovnu**.

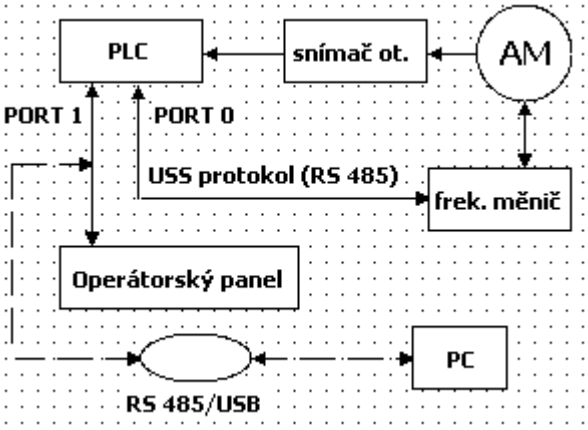
Postup pro připojení knihovny a její následné přiřazení paměti je následující:

1. Pomocí příkazu menu **Soubor > Přidat knihovnu** přidáme adresář ”Knihovny” do instrukčního stromu. Vybereme příslušnou instrukci a vložte ji do programu (stejně jako jakoukoliv standardní instrukci). Na obrázku 3.5 je zobrazena část připojené knihovny USS.
2. Ještě je třeba přiřadit pro knihovnu část paměti V, STEP 7- Micro/WIN nás po kompilaci projektu vyzve k přiřazení paměťového bloku. Pro přiřazení paměťových bloků použijeme dialogové okno ”Přidělení paměti pro knihovnu”.

4 Sestavení hardwaru pro laboratorní úlohu

4.1 Blokové schéma laboratorní úlohy

Na obrázku 4.1 je zobrazeno blokové schéma našeho hardwarového setu, kde obdélníky znázorňují řídící soustavu (PC, automat, operátorský panel, frekvenční měnič a snímač otáček) a řízený prvek (asynchronní motor) je zobrazen kružnicí. Plné šipky znázorňují propojení jednotlivých prvků řídící soustavy. Přerušované šipky znázorňují propojení PC s automatem a operátorským panelem, při nahrávání programů do těchto prvků.



Obr. 4.1 Blokové schéma laboratorní úlohy.

4.2 Propojení a nastavení jednotlivých prvků řídící soustavy

4.2.1 Propojení automatu s operátorským panelem

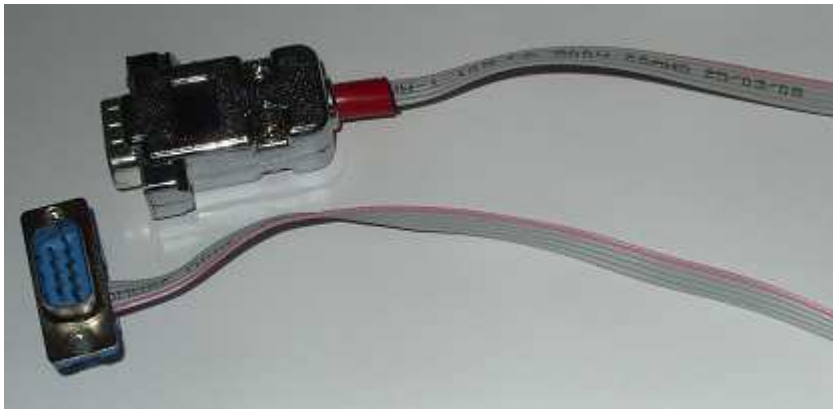
Programovatelný automat, má dva porty pro komunikaci s okolím. Porty mají označení PORT 0 a PORT 1, kde se jedná o standardní CAN 9 konektor typu samice. Tento stejný konektor je i na straně operátorského panelu (obrázek 4.2). PORT 0 na straně automatu je přiřazen USS knihovně. Komunikace automatu s operátorským panelem proto musí probíhat na PORTu 1. Na straně operátorského panelu je jenom jeden komunikační konektor, tak zde nelze udělat chybu při zapojení.



Obr. 4.2 Datový a napájecí konektor panelu.

Součástí operátorského panelu je i komunikační kabel, který slouží k propojení automatu a panelu. Tento kabel je však zbytečně masivní a dlouhý, pro naše využití nevyhovující. Proto bylo třeba zhotovit jednoduší variantu tohoto kabelu (obrázek 4.3). Původní kabel měl jenom 5 pinů, proto stačí pro komunikaci 5-žilový kabel, což přináší první zjednodušení. Kabel je dobře ohebný o délce přibližně 40 cm. Na straně panelu je konektor

otočen o 90° a na straně automatu je použit přímý konektor. Volba těchto konektorů vycházela z polohy umístění konektorů na straně jednotlivých zařízení.



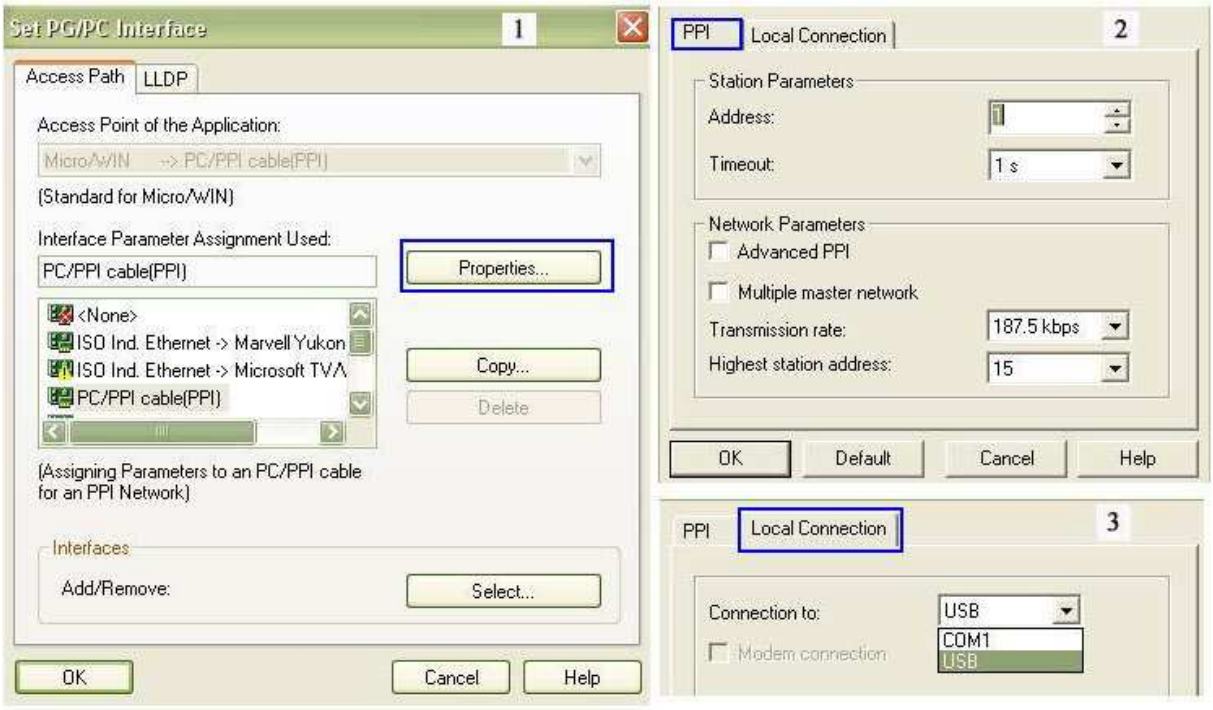
Obr. 4.3 Zhotovená jednodušší varianta komunikačního kabelu.

4.2.2 Propojení automatu s PC

Pro komunikaci mezi automatem a osobním počítačem použijeme standardní kabel USB/PPI Multi-Master Cable. Jak už napovídá označení kabelu, tak na straně PC je USB konektor a na straně automatu je CAN 9 konektor. Součástí kabelu je převodník, který zajišťuje komunikaci mezi těmito různými rozhraními. Pro komunikaci s PC využijeme jako u panelu PORT 1. Je tedy zřejmé, že při nahrávání a především při ladění programu, nelze současně využívat operátorského panelu, pro zadávání příkazů. Je to jisté omezení, které ztěžuje sledování a ladění programu.

4.2.3 Nastavení komunikace mezi automatem a PC

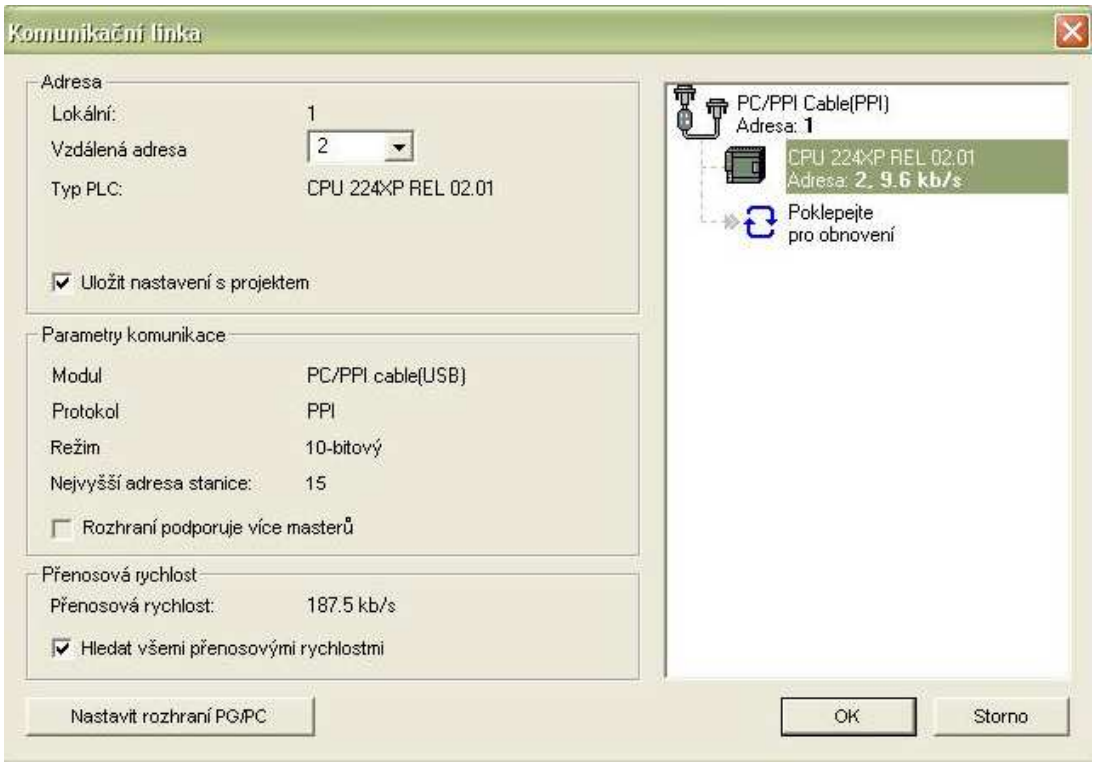
V prostředí STEP 7-Micro/WIN klikneme v navigační liště na položku "Nastavení rozhraní PG/PC" (Set PG/PC Interface), kdy se nám otevře okno pro nastavení komunikaci



Obr. 4.4 Nastavení komunikace mezi automatem a Pc.

mezi automatem a různými rozhraními (obrázek 4.4-1). Vybereme rozhraní PC/PPI cable(PPI) a klikneme na tlačítko “Properties”, otevře se nám okno pro nastavení adresy automatu, přenosové rychlosti (obrázek 4.4-2). Nastavíme adresu automatu na 1 a přenosovou rychlost na 187.5 kbps a nejvyšší adresu zařízení v síti nastavíme na 15, ostatní položky necháme ve výchozím stavu. Po kliknutí na položku “Local Connection” se nám objeví nabídka pro volbu komunikačního rozhraní na straně PC (obrázek 4.4-3). V našem případě nastavíme USB kabel. Potvrdíme všechny volby tlačítkem “OK“.

Po nastavení rozhraní PG/PC klikneme v navigační liště na položku “Komunikace” (Communication), kdy se nám otevře komunikační okno (obrázek 4.5), které zobrazuje nastavené parametry komunikace a nastavenou přenosovou rychlost.

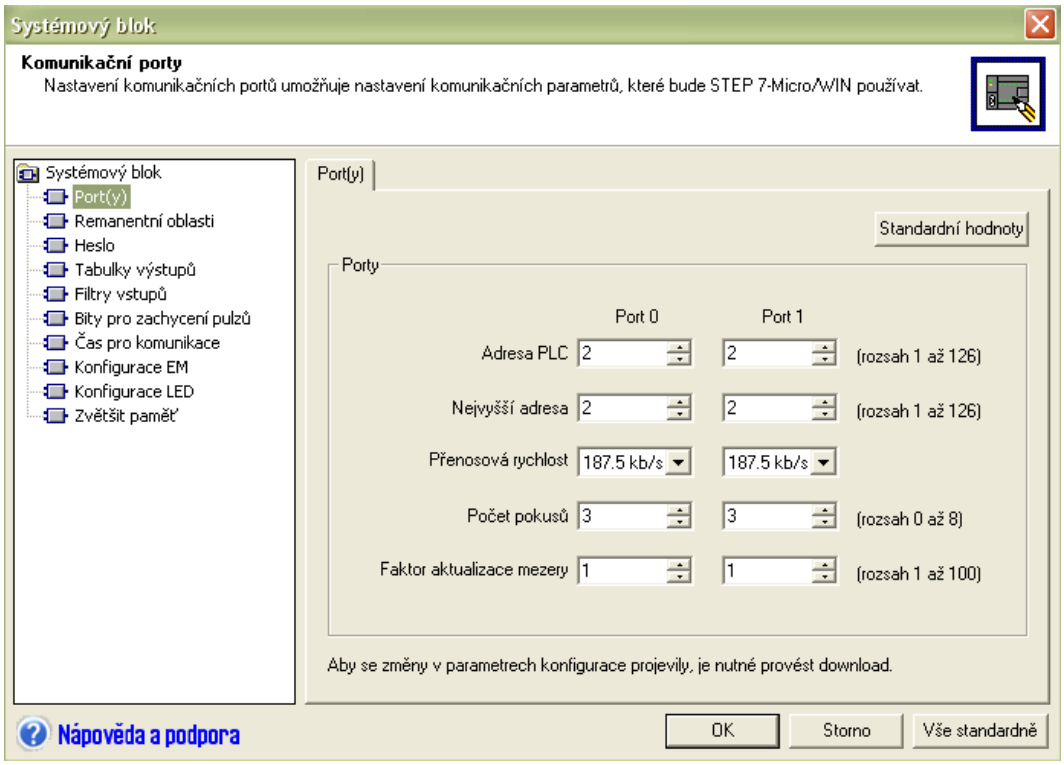


Obr. 4.5 Aktuálně nastavené parametry komunikace a přenosové rychlosti.

V levé části okna je zobrazeno zařízení (automat), které bylo detekováno aplikací STEP 7-Micro/WIN. Je zde informace o přesném označení zařízení, jeho adrese a nastavené přenosové rychlosti. Přenosová rychlost se od požadované odlišuje, proto je nutné nastavit přenosovou rychlost i na straně automatu, to se provede po kliknutí na položku “Systémový blok” (System block) v navigační liště. Systémový blok (obrázek 4.6) obsahuje celou řadu nastavení automatu, nás zajímá první položka “Port(y)”. Je zde možné nastavení parametrů obou portů automatu (PORT0 a PORT 1). Nastavení portů je shodné a je zobrazeno na obrázku 4.5, kdy adresa automatu a nejvyšší adresa zařízení v síti, je nastavena na hodnotu 2. Přenosová rychlost je nastavena na 187,5 kb/s. Ostatní parametry jsou ponechány ve výchozím stavu. Aby se nastavené parametry projevily na straně automatu, je třeba provést download. Po úspěšném nahrání parametrů do PLC a provedení obnovení načtení zařízení se nám objeví požadovaná rychlost přenosu (187.5 kb/s). Nastavení PORTu 1 slouží jak pro komunikaci mezi automatem a PC, tak i mezi automatem a operátorským panelem. Tímto máme nastavenou komunikaci pro obě tyto



zařízení. PORT 0 je nastaven pro komunikaci mezi automatem a frekvenčním měničem pomocí USS protokolu.



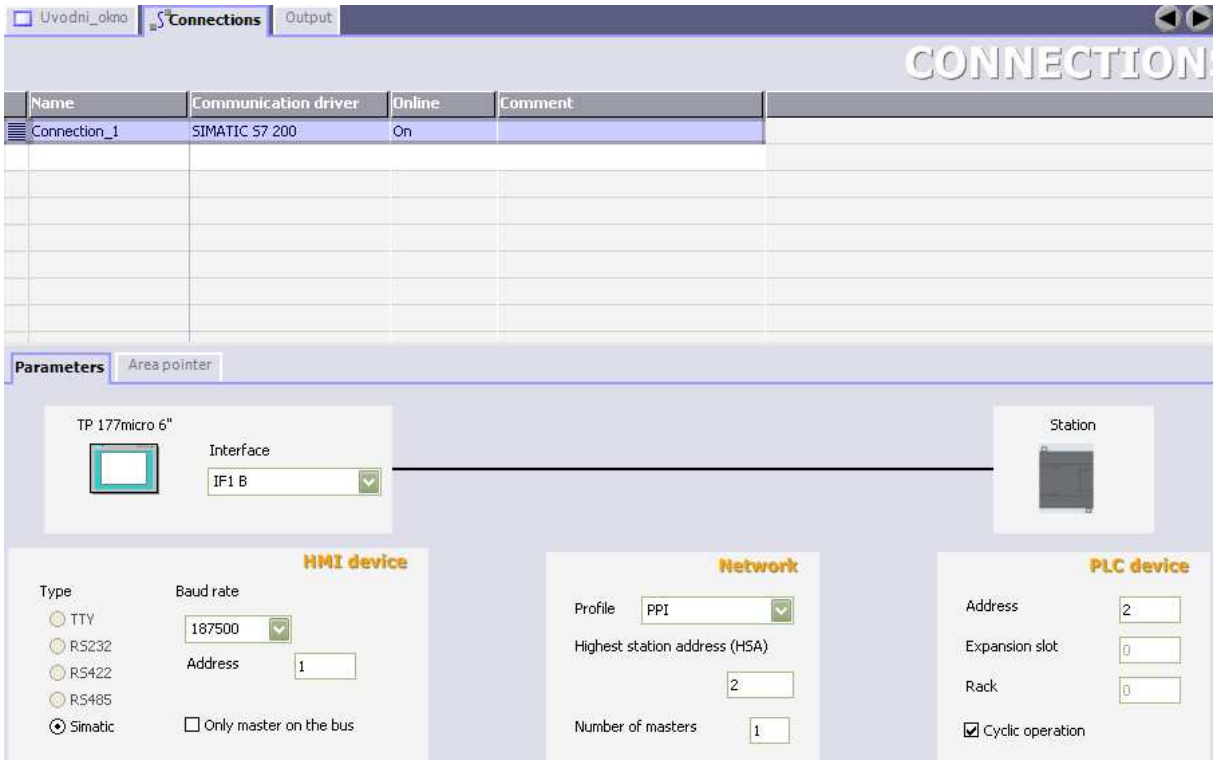
Obr. 4.6 Nastavení komunikačních portů, které bude STEP 7-Micro/WIN používat.

4.2.4 Nastavení komunikace mezi automatem a operátorským panelem

Nastavení komunikace na straně operátorského panelu se provede pomocí programovacího prostředí WinCC flexible Micro. Po kliknutí na položku Connections ve správci objektů, se nám otevře nabídka pro nastavení propojení panelu s automatem (obrázek 4.7). Program WinCC flexible Micro slouží pro nastavení a vytváření aplikací pro panely micro, které se propojují s automaty řady S7 – 200. Proto je zde jenom možnost propojení s automatem řady S7 – 200 (položka Communication driver). Ve spodní části máme zobrazený panel, automat a síť. Na straně panelu (HMI device) nastavujeme přenosovou rychlost na hodnotu 187500 bps a adresu panelu, která je defaultně nastavená na 1 (Address = 1). Panel zde pracuje jako master, který dává příkazy automatu (zařízení typu slave). Na straně automatu je nejdůležitější nastavení adresy automatu (addres = 2). Ostatní parametry necháme nastavené v defaultním stavu. Pro komunikaci použijeme PPI protokol, který se používá v případě, když je panel připojen k automatu přímo bez dalších zařízení na síti a současně panel pracuje jako master. Nejvyšší adresu zařízení jsme nastavili na 2 (HSA = 2) a číslo zařízení master je 1 (Number of masters = 1). HSA jsem nastavil na hodnotu 2 z důvodu, aby nedocházelo ke zbytečnému skenování sítě (zvyšování režii sítě), když tam jiná zařízení nejsou. Číslo zařízení master určuje, které zařízení bude dávat příkazy (operátorský panel).

Nastavení komunikace na straně automatu už jsme provedli v předchozí kapitole. Po zapnutí obou zařízení (automat a panel) a propojení příslušným kabelem se nám na displeji operátorského panelu objeví hláška s informací o navázání spojení s automatem. Tímto máme

úspěšně nastavenou komunikaci mezi automatem a operátorským panelem a můžeme pokračovat v dalších nastaveních.



Obr. 4.7 Možnosti nastavení komunikace mezi panelem a PLC.

4.2.5 Propojení a nastavení komunikace automatu s frekvenčním měničem

Propojení automatu s frekvenčním měničem bylo popsáno v kapitole věnující popisu samotného hardwaru.

Nastavení komunikace na straně automatu bylo popsáno v kapitole věnující se nastavením komunikace mezi automatem a PC. Bylo zde popsáno nastavení PORTu 1, tak i PORTu 0, který je využíván pro komunikaci s frekvenčním měničem (obrázek 4.6).

Postup pro změnu parametrů měniče jsme si popsali v kapitole věnující se frekvenčnímu měniči. Pro možnost editovat některých parametrů (např. parametry motoru), je třeba spustit režim “Rychlé uvedení do provozu”, který se aktivuje nastavením parametru P0010 = 1. Nyní postupně zadáváme všechny důležité parametry, které jsou vypsané v tabulce 4.0. Ostatní parametry jsou ponechány ve výchozím nastavení. Po nastavení všech parametrů provedeme jejich uložení. Uložení provedeme nastavením parametru P3900 = 2. Po uložení je ještě dobré překontrolovat všechny nastavované parametry, zdali se provedlo uložení u všech parametrů.

Číslo parametru	Název parametru	Nastaveno
P0003	Přístupová práva	3
P0100	Evropa / Severní Amerika	0
P0304	Jmenovité napětí motoru	230
P0305	Jmenovitý proud motoru	1.34
P0307	Jmenovitý výkon motoru	0.25
P0310	Jmenovitý kmitočet motoru	50
P0311	Jmenovité otáčky motoru	1350
P0610	Chování měniče při přetížení motoru	1
P0700	Způsob ovládání měniče	5
P1000	Výběr zdroje žádané hodnoty	5
P1080	Minimální hodnota výstupního kmitočtu	0.0
P1082	Maximální kmitočet	100
P1120	Doba rozběhu motoru	0.2
P1121	Doba doběhu motoru	0.2
P2010	Rychlost přenosu dat sériové komunikace USS	8
P2011	Adresa měniče na sériové lince USS	0

Tab. 4.0 Seznam parametrů měniče, které se odlišují od výchozích hodnot.

4.3 Připojení inkrementálního snímače na vstupy automatu

Zapojení výstupů inkrementálního snímače na vstupy automatu není libovolné, ale záleží, jaký vysokofrekvenční čítač používáme. Programovatelný automat S7 -224 XP má 6 vysokofrekvenčních čítačů, kde každý čítač využívá jiné vstupy. My jsme zvolili čítač HSC4, který využívá vstupy I0.3, I0.4, I0.5. Popis výstupů z inkrementálního snímače a jeho připojení k automatu je popsáno v tabulce 4.1. Čítač HSC4 byl zvolen pro jeho vzorkovací frekvenci 100 kHz pro dvě fáze a schopnost pracovat jako A/B fázový kvadraturní čítač.

vodič	signál	popis	Vstup automatu
bílý	Ua2	kanál B	I0.4
zelený	Ua1	kanál A	I0.3
černý	UaS	detekce chyby signálu	I0.5
žlutý	Ua0	detekce otočení o 360°	I0.6
červený	Up	napájení +	L+
modrý	0 V	napájení 0 V	M

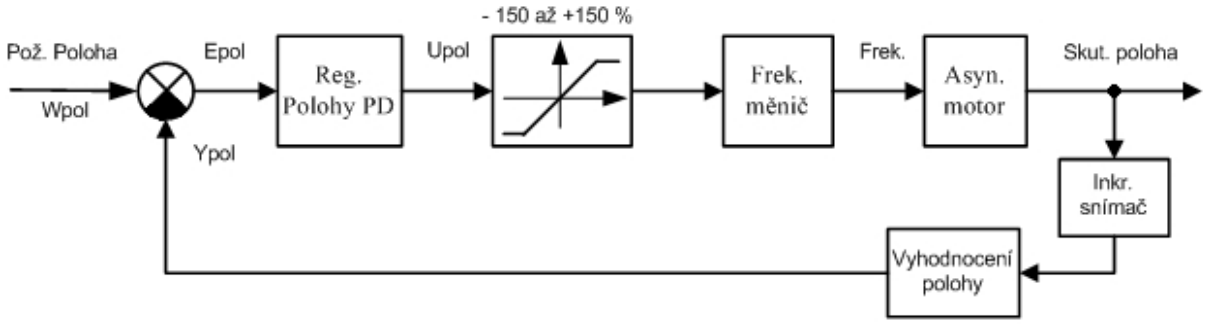
Tab. 4.1 Popis vodičů inkrementální snímače.

5 Návrh regulátoru polohy a jeho nastavení

5.1 Návrh regulátoru polohy

5.1.1 První varianta regulátoru

První navrhovaný regulátor pro řízení polohy je použití PD regulátoru, který je zobrazen na obrázku 5.0. Požadovaná veličina vstupuje do rozdílového členu jako požadovaná poloha (natočení motoru), kterou zadává obsluha prostřednictvím OP. Skutečná hodnota polohy je počítána na základě impulsů z inkrementálního snímače. Regulační odchylka vstupuje do PD regulátoru a jeho akční zásah je omezen v rozsahu -150 až + 150. Omezení je zde nutné, protože požadovaná poloha natočení může být libovolná. Tímto můžou vzniknout vysoké hodnoty akčního zásahu, které by frekvenční měnič odmítl akceptovat. Akční zásah se předává frekvenčnímu měniči prostřednictvím USS knihovny, kdy hodnota je v procentech. 100 % = 50 Hz. Motor je schopný pracovat i na vyšší frekvenci než je 50 Hz, proto je zde omezení nastaveno na 150%, kdy frekvence je 75 Hz. Tímto je zajištěn kratší čas potřebný k požadovanému počtu otočení hřídele motoru.



Obr. 5.0 První varianta regulátoru polohy – PD regulátor.

Nejdůležitějším blokem celé regulační smyčky je PD regulátor, který zesiluje a derivuje vstupní regulační odchylku. Jelikož regulátor je realizovaný softwarově pomocí programovatelného automatu, pracujeme s nespojitými hodnotami, které nemůžeme derivovat popřípadě integrovat. Tímto je derivace nahrazena diferencí a integrace je nahrazena sumací. Existují dva základní typy algoritmu pro výpočet hodnot akčního zásahu pro číslicové regulátory a to „polohový " a „přírůstkový". V případě, že nepoužíváme sumační složku, je pro náš výpočet akčního zásahu vhodný polohový algoritmus (5.0).

$$\mathrm{u(k)} = \mathrm{r}_0\left\{e(k) + \frac{T}{T_i}\sum_{i=1}^{k} e(i) + \frac{T_d}{T}[e(k) - e(k-1)]\right\} \qquad (5.0)$$

Kde r0 představuje proporcionální konstantu neboli zesílení PID regulátoru, v praxi se však většinou označuje jako Kp. Po vynulování integrační složky dostáváme vztah 5.1.

$$u(k) = r_0\left\{e(k) + \frac{T_d}{T}[e(k) - e(k-1)]\right\} \qquad (5.1)$$

Vztah 5.1 můžeme ještě upravit na vztah 5.2, dosazením Kp za r0 a použitím derivační konstanty Cd (5.3).

$$u(k) = K_p e(k) + C_d\left[e(k) - e(k-1)\right] \qquad (5.2)$$

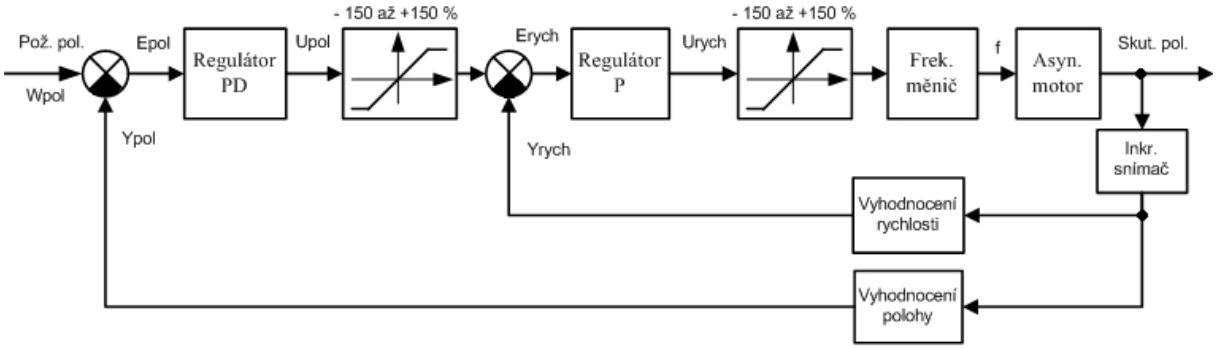
$$\text{Kde } C_d = K_p\frac{T_d}{T} \qquad (5.3)$$

V rovnici (5.2) je u (k) akční veličina, kterou budeme v každém cyklu programu počítat. Kp značí proporcionální konstantu, která bude volena na základě Ziegler - Nicholsonovi metody stejně jako Td, která značí derivační časovou konstantu. Za proměnnou e (k) budeme dosazovat regulační odchylku, která bude rozdílem žádané hodnoty w (k) a hodnoty skutečné y (k). Proměnná e (k-1) značí minulou regulační odchylku (vypočítanou v předchozím programovém cyklu).

Základem při návrhu tohoto regulátoru bylo použití proporcionální regulátoru, který zajistí dostatečně rychlou regulaci a stabilitu. Pro zlepšení vlastností regulace byl regulátor doplněn o derivační složku, která zajistí urychlení regulace. Integrační složku pro odstranění trvalé regulační odchylky nebyla použita z důvodu velkých regulačních odchylek, kdy by došlo naintegrovaní (k sumaci) velkých hodnot. Naintegrování by způsobilo prodloužení doby pro ustálení regulátoru.

5.1.2 Druhá varianta regulátoru

Jedná se o předcházející regulátor polohy, který je doplněn o vnitřní smyčku regulace rychlosti (obrázek 5.1). Tato vnitřní smyčka zajistí zlepšení dynamiky regulátoru polohy.



Obr. 5.1 Regulátor polohy doplněn o vnitřní smyčku regulace rychlosti.

Akční veličina PD regulátoru představuje požadovanou veličinu pro P regulátor rychlosti. Skutečnou veličinou jsou otáčky motoru za minutu (ot/min), které se pohybují v rozmezí od -1500 do 1500 ot/min, kde záporná hodnota značí změnu směru rotace. Tuto hodnotu musíme přizpůsobit rozsahu požadované veličině (-150 až + 150) a proto ji podělíme hodnotou 10. Po výpočtu regulační odchylky vnitřní smyčky, je předána regulátoru rychlosti (P reg.), který vypočítá akční zásah, který opětovně omezíme. Vnitřní smyčka může mít horší vlastnosti, než vnější, a proto dostačuje použití samostatného P regulátoru. Vztah pro výpočet akčního zásahu P regulátoru je následující:

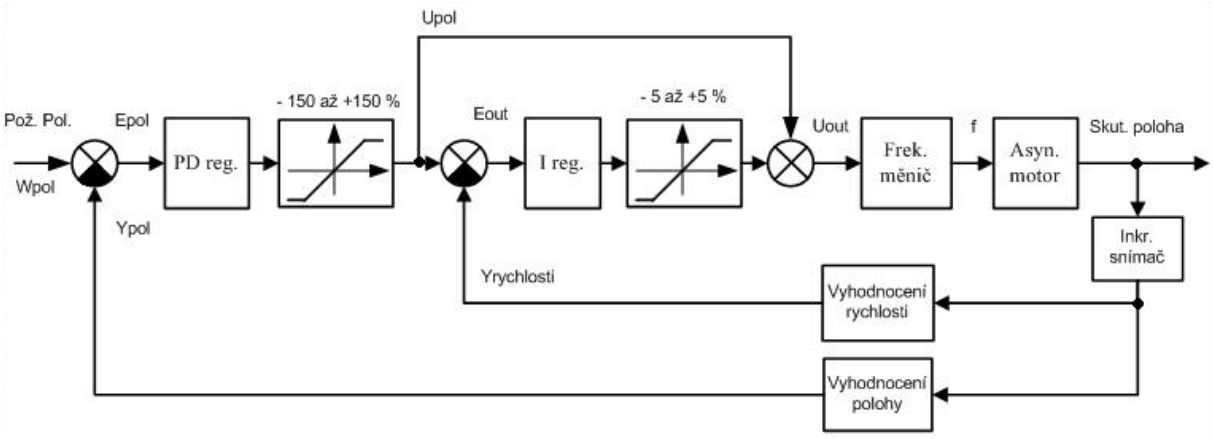
$$u(k) = K_p e(k) \qquad (5.4)$$

Tato varianta by zlepšila vlastnosti regulátoru polohy, ale bohužel nastal problém při seřízení P regulátoru. Problém spočíval v tom, že regulátor nešel seřídit tak, aby dosáhl požadované hodnoty (otáček motoru). Pokud bylo zesílení regulátoru menší než jedna, tak vznikala velká regulační odchylka a pokud bylo zesílení 1, nebo větší byl už regulátor nestabilní. Tento projev regulátoru naznačoval, že soustava trpí dopravním zpožděním. Tato domněnka se nakonec potvrdila při experimentálním měření, kdy zpoždění se pohybovalo kolem 60 ms. Dopravní zpoždění je způsobeno hlavně komunikací mezi automatem a frekvenčním měničem pomocí USS protokolu a pak samotným frekvenčním měničem. Tímto

jsem musel tuto variantu regulace opustit a navrhnout jiné vylepšení základního navrhovaného regulátoru polohy.

5.1.3 Finální varianta regulátoru

Finální varianta regulátoru vychází z koncepce regulátoru polohy s vnitřní smyčkou regulace rychlosti, ke které přidává dopředné řízení a proporcionální regulátor nahrazuje integračním. Jedná se tedy o regulátor polohy s pomocným regulátorem rychlosti s dopředným řízením (obrázek 5.2). Princip tohoto regulátoru je následující. Akční zásah PD



Obr. 5.2 Regulátor polohy doplněn o regulátor rychlosti s dopředným řízením.

regulátoru se přivádí na součtový člen a zároveň vstupuje jako požadovaná hodnota vnitřní smyčky I regulátoru. Skutečnou hodnotu vnitřní smyčky představují otáčky motoru za minutu, které jsou přizpůsobeny požadované hodnotě, jako v předešlé variantě regulátoru. Akční zásah I regulátoru je omezena v rozsahu -5 až + 5%, protože regulátor zde slouží pro kompenzaci skluzu motoru, který je cca do 5%. Akční zásah I regulátoru je přičten k akčnímu zásahu PD regulátoru. Tímto máme zajištěno, že otáčky nezatíženého, ale i zatíženého motoru budou odpovídat požadovaným otáčkám. Volba I regulátoru pro vnitřní smyčku regulace polohy, byla zvolena na základě požadavku na nulovou regulační odchylku. Vztah pro výpočet akčního zásahu I regulátoru:

$$u(k) = \left[u(k-1) + C_i e(k)\right] \qquad (5.5)$$

$$\text{Kde } C_i = K_p \frac{T}{T_i} \text{ a } K_p = 1 \qquad (5.6)$$

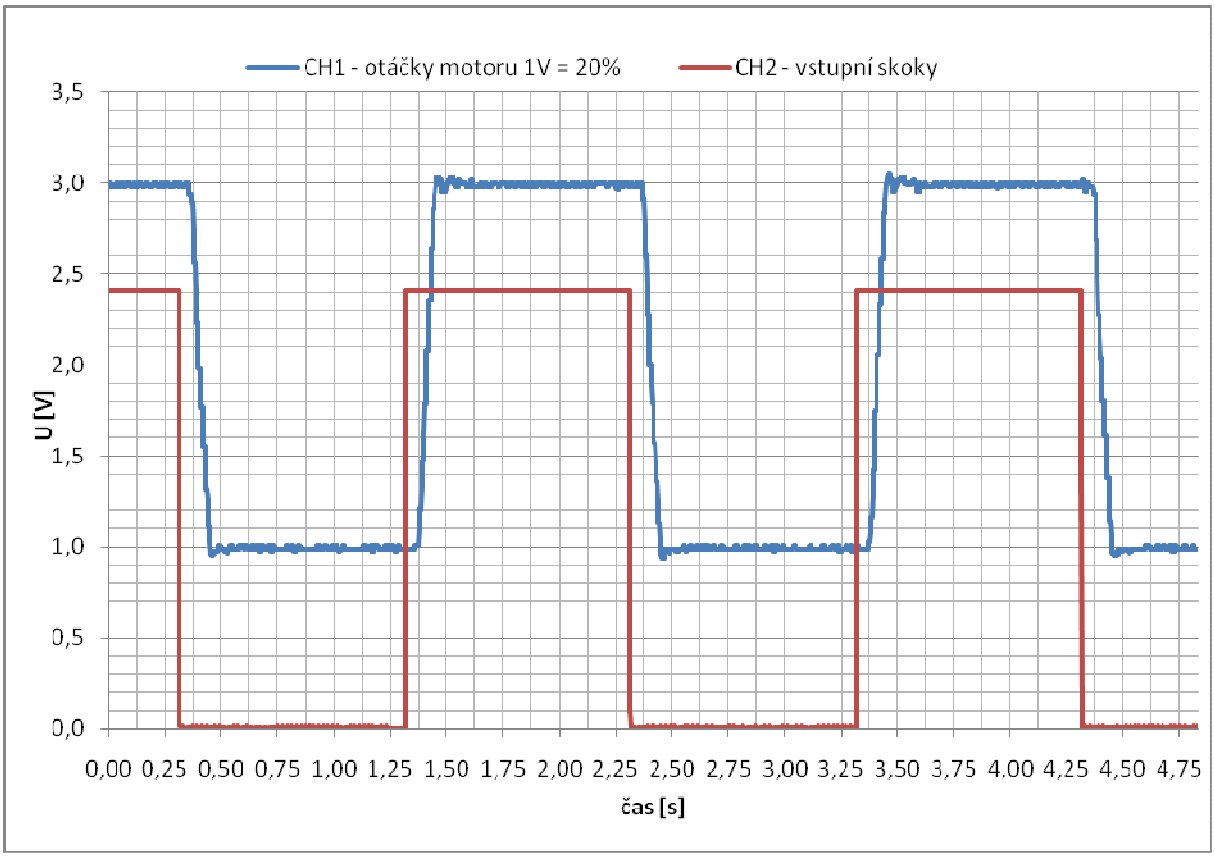
V rovnici 5.5 je u (k) akční zásah I regulátoru, T je jako u derivační složky vzorkovací perioda a T_i je integrační časová konstanta. Hodnotu T_i určíme pomocí Ziegler-Nicholsonovi metody a zesílení je v tomto případě rovno 1. Hodnota u (k-1) bude odpovídat velikosti minulé akční veličiny.

5.2 Nastavení regulátoru

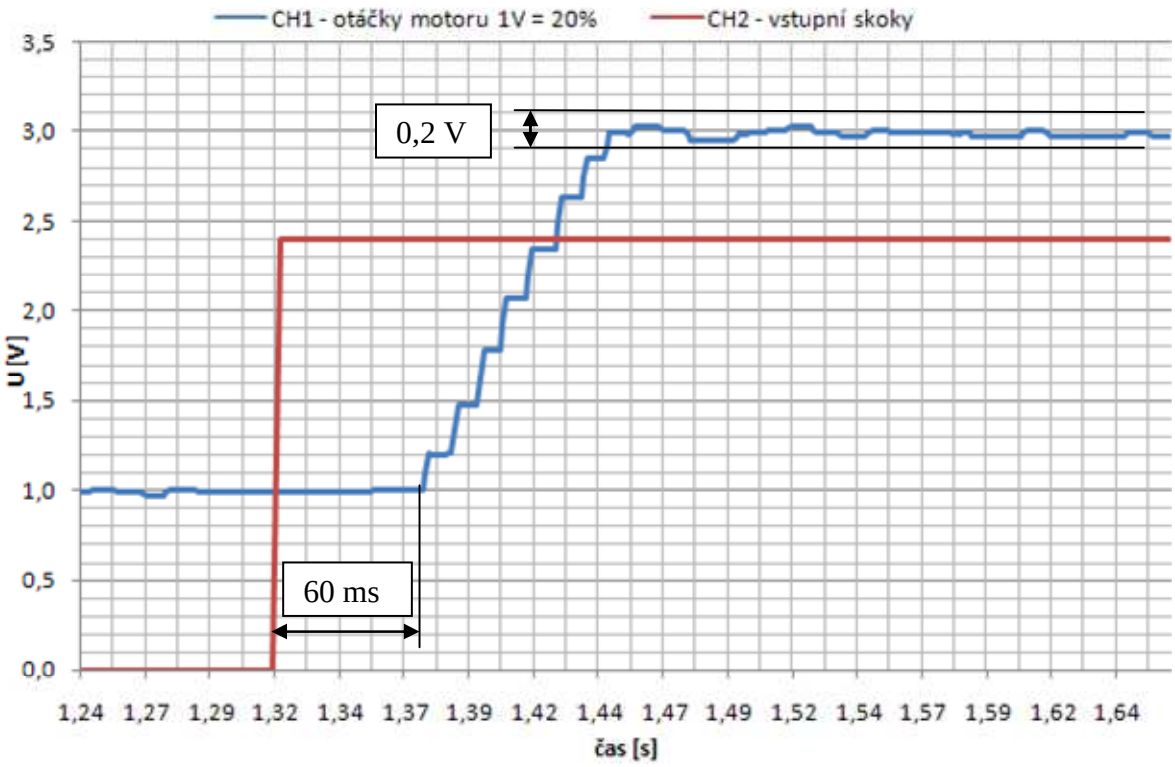
Aby celá laboratorní úloha pracovala co nejpřesněji a bez sebemenších problémů, musí dojít k seřízení regulátorů polohy a rychlosti. Seřízení parametrů regulátoru budeme provádět pomocí metody Zigler – Nicholse. Základní myšlenkou metody Ziegler - Nichols je přivést regulační obvod na hranici stability. Za kritické nastavení považujeme takové, při němž je derivační a integrační složka vyřazena. Postupnou změnou zesílení r_0 je regulační obvod přiveden na hranici stability. Zesílení r_0 , kterým jsme obvod dostali na hranici stability, se nazývá kritické zesílení r_{ok}. Na hranici stability kmitá obvod netlumenými kmity o konstantní amplitudě a důležité je změřit právě periodu těchto kmitů, a to je tzv. kritická perioda. T_k. U integračního regulátoru se obvod dostane do kritického stavu změnou integrační konstanty regulátoru T_i, přičemž tuto kritickou hodnotu označíme T_{ik}. Z této hodnoty odvodíme optimální nastavení I regulátoru.

5.2.1 Nastavení regulátoru rychlosti s dopředným řízením

Nejdříve bylo nutné seřídit rychlostní smyčku regulátoru. Regulátor rychlosti je realizovaný pomocí integračního regulátoru. Seřízení regulátoru probíhalo podle výše popsaného postupu. Kritická hodnota integrační konstanty byla stanovena na hodnotu T_{ik} = 0,039 s. Při použití samotného I regulátoru se T_i rovná dvojnásobku T_{ik}, tímto dostáváme T_i = 0,078 s. Po nastavení regulátoru bylo provedeno změření přechodové charakteristiky regulátoru polohy, která je zobrazena na obrázku 5.3. Červeně vyznačený průběh nepředstavuje požadovanou hodnotu otáček motoru, ale signalizuje požadavek na změnu otáček motoru. Jedná se tedy o skokové změny požadovaných otáček. Každou sekundu je provedena změna požadovaných otáček. Logické úrovně nula odpovídá 300 ot/min (požadované otáčky jsou zadány jako 20% jmenovitých otáček motoru) a logické úrovně jedna odpovídá 900 ot/min (60%). Změna skutečných otáček je vyznačena průběhem modré barvy, kde 1V odpovídá 300 ot/min. Z grafu je patrné časové zpoždění mezi požadavkem na změnu otáček a počátkem provádění změny otáček motoru. Dále si můžeme všimnout na průběhu skutečných otáček malého překmitu, který se vyskytuje jak při zvýšení, tak pří snížení otáček motoru. Detail tohoto překmitu a dopravního zpoždění je zobrazen na obrázku 5.4. Podle tohoto detailu je patrné, že překmit se pohybuje v rozmezí ± 0,1 V, což odpovídá hodnotě ± 2%. Jedná se tedy o malý překmit, který potvrzuje správné nastavení regulátoru. Časové zpoždění odpovídá hodnotě 60 ms, které je způsobenou časovou prodlevou komunikace mezi automatem a frekvenčním měničem při využití USS protokolu a zpožděním na frekvenčním měniči.



Obr. 5.3 Časový průběh skokových změn otáček motoru regulátoru rychlosti.

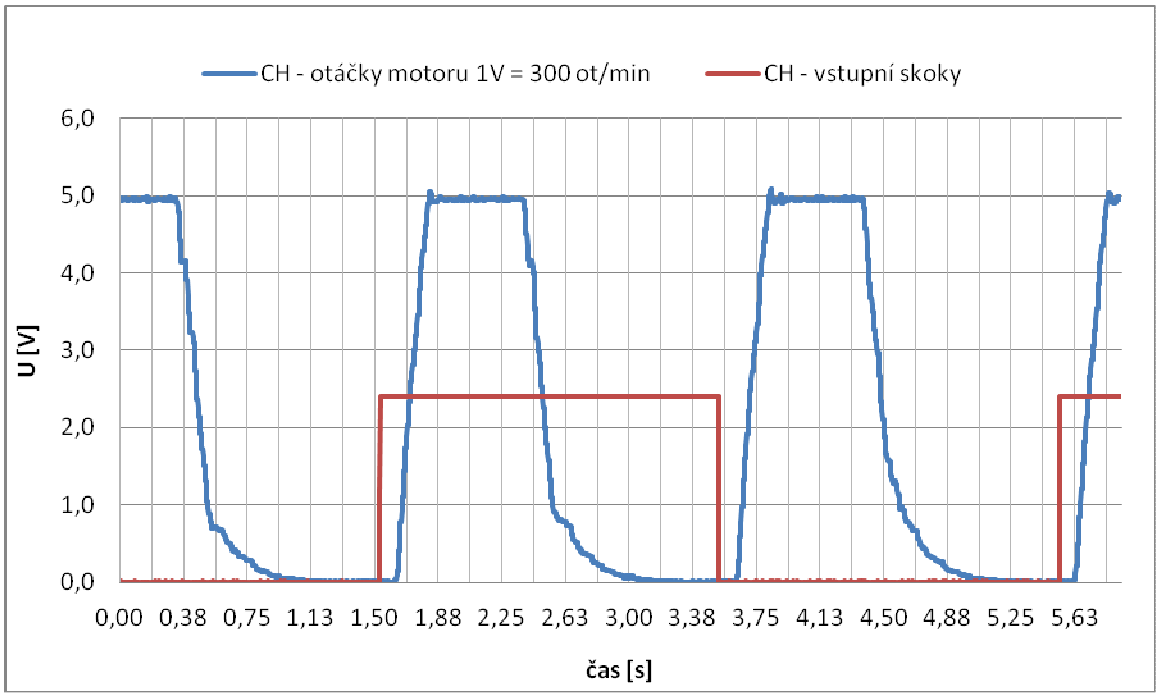


Obr. 5.4 Detail přechodové charakteristiky otáček motoru.

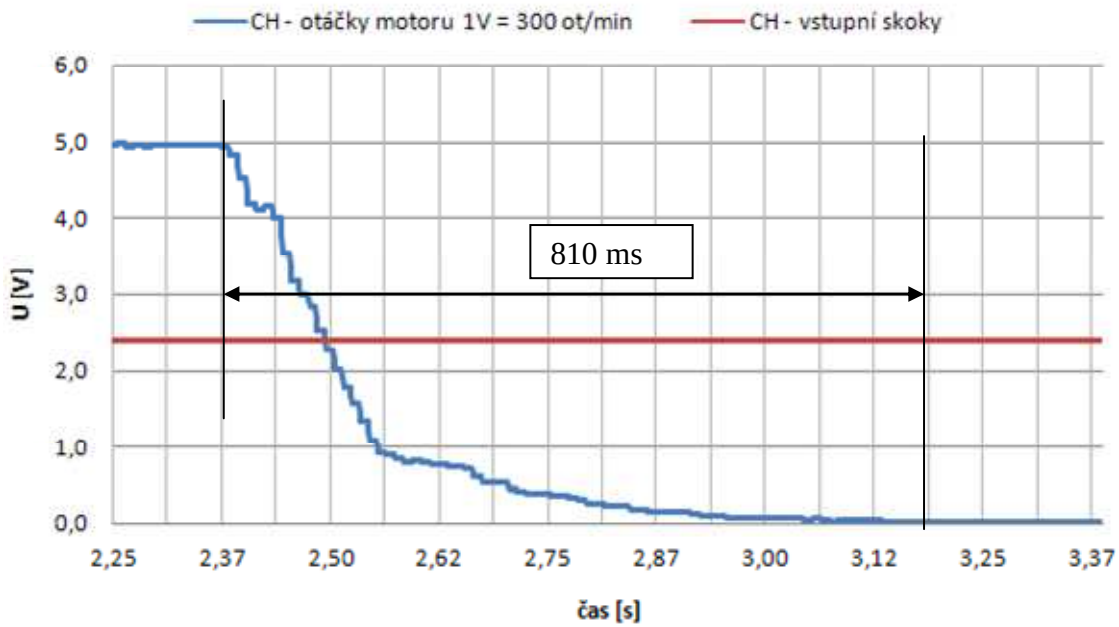
5.2.2 Nastavení regulátoru polohy

Pro regulaci polohy natočení hřídele motoru byl použít PD regulátor. Nastavení PD regulátoru polohy probíhalo podle výše popsaného postupu. Byla vyřazena D složka a bylo zvyšováno zesílení až na mez stability. Kritické zesílení je v ten moment 67 a kritická perioda T_k přibližně 400 ms. Doba vzorkování T je u regulátoru polohy nastavena na 15 ms. Všechny přechodové charakteristiky byly měřeny pro skokové změny žádané hodnoty polohy od 0 do 20, v časovém intervalu dvou vteřin. Po určení kritického zesílení a kritické periody bylo vypočteno konkrétní nastavení podle metody Ziegler - Nicholse. Pro PD regulátor bylo výpočtem stanoveno zesílení 26,8 a derivační časová konstanta 20 ms.

Průběh přechodové charakteristiky takto nastaveného regulátoru polohy je na obrázku 5.5. Červeně vyznačený průběh signalizuje požadavek na změnu polohy. Průběh otáček je vyznačen modrou barvou. Při požadavku na změnu polohy se motor postupně roztočí na maximální otáčky, na kterých setrvá a jakmile se hodnota skutečné polohy přiblíží k hodnotě požadované polohy provede se snižování otáček. Snižování otáček má zpočátku strmý spád, ale pak se strmost klesání otáček snižuje. Detail tohoto přechodu je zobrazen na obrázku 5.6. Jak je patrné, tak doba regulace je přibližně 800 ms, což je vysoké číslo. Doba regulace by šla zmenšit zvýšením zesílením proporcionální složky, což by mělo za následek vzniku překmitů v regulačním pochodu, což je u regulace polohy nežádoucí jev. Dalším výrazným zkrácením doby průběhu přechodové charakteristiky by bylo rozšíření regulátoru polohy o vnitřní smyčku regulace rychlosti s P regulátorem (druhá varianta navrhovaného regulátoru). Z důvodu dopravního zpoždění nelze tuto variantu regulátoru použít. Přestože je doba přechodové charakteristiky takto vysoká, dosahuje regulace polohy uspokojivých výsledků.



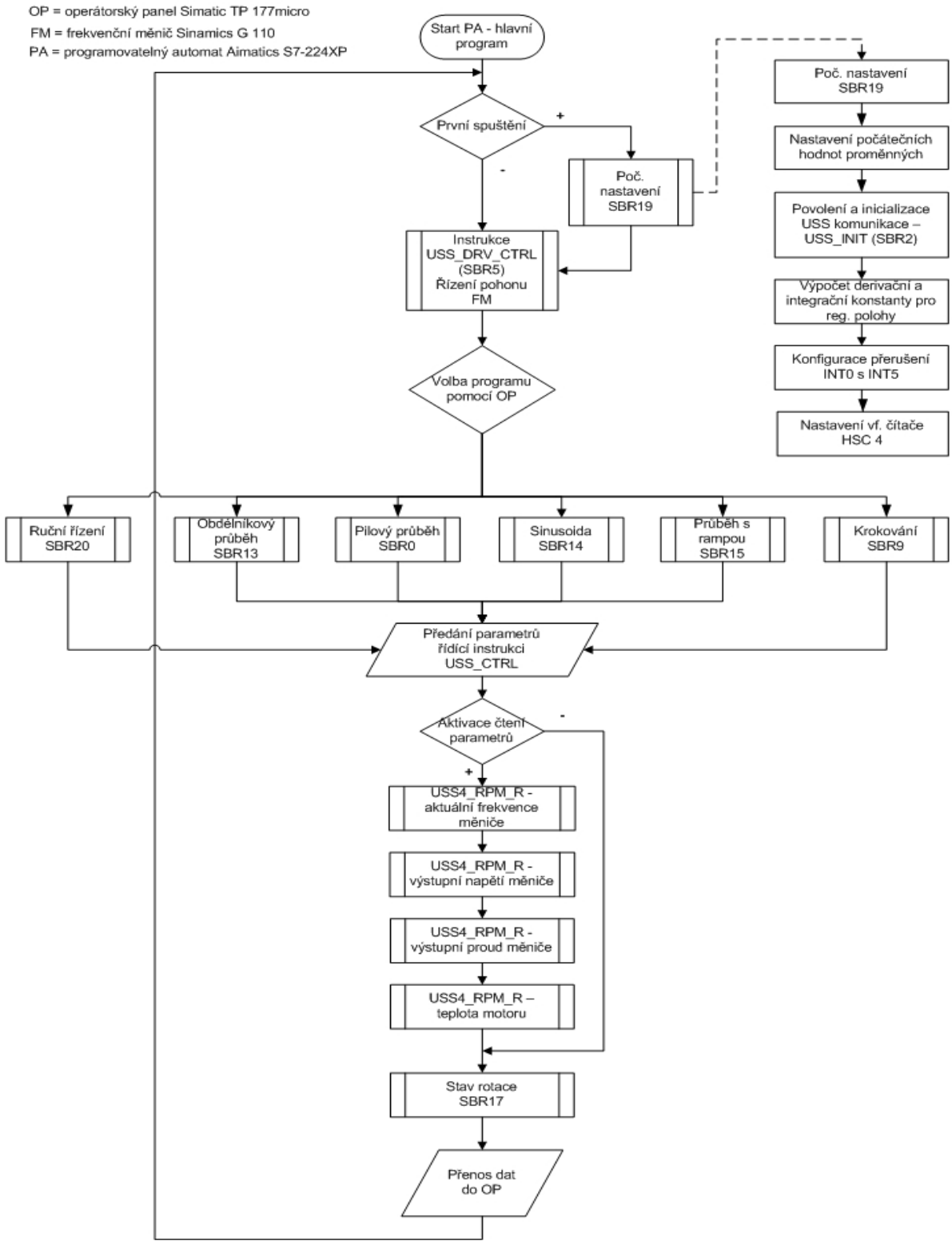
Obr. 5.5 Průběh přechodové charakteristiky otáček motoru regulátoru polohy.



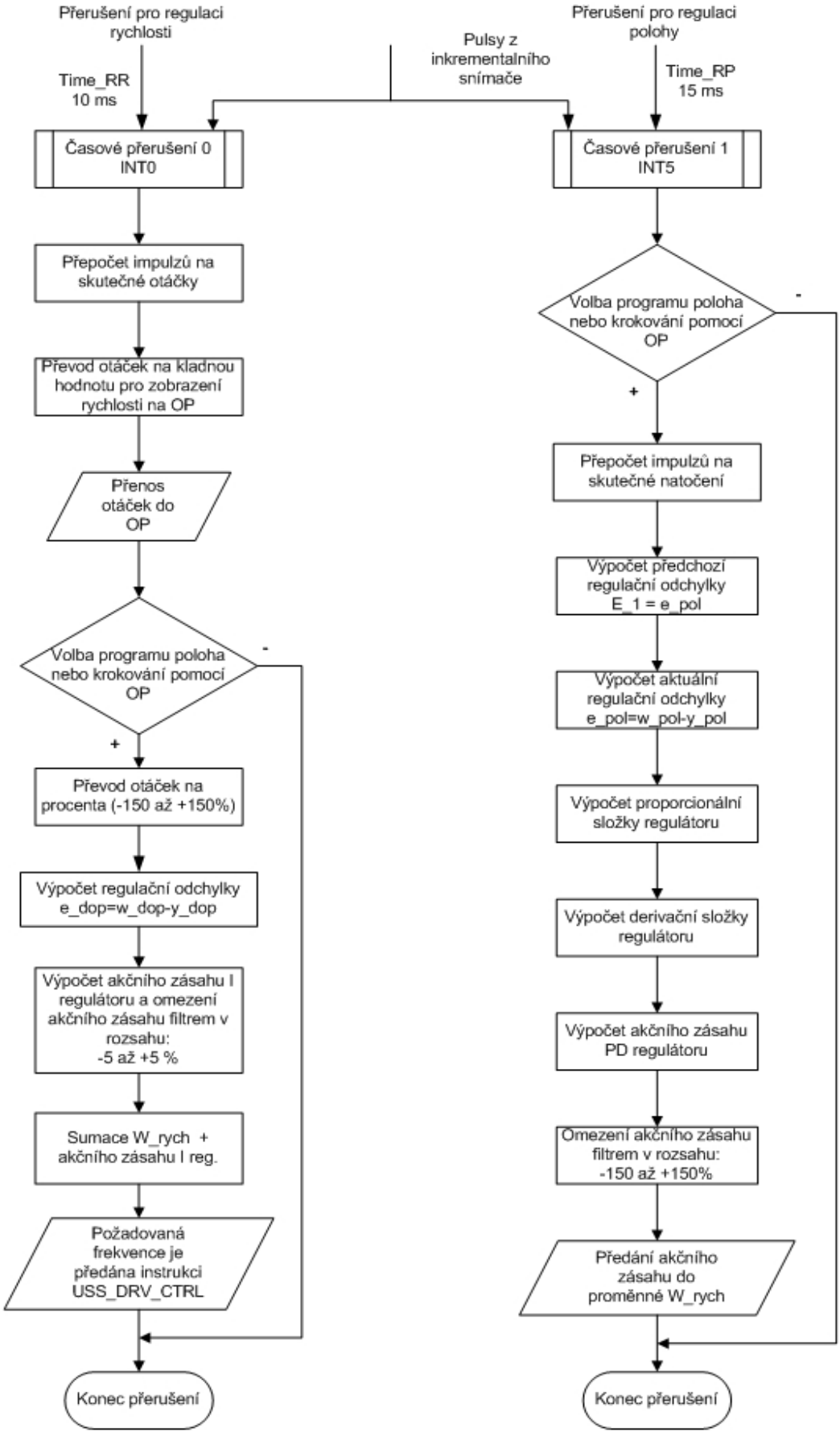
Obr. 5.6 Detail přechodové charakteristiky otáček motoru regulátoru polohy.

6 Programové vybavení automatu

6.1 Vývojový diagram



Obr. 6.0 Vývojový diagram hlavního programu (main) a jednotlivých podprogramů.



Obr. 6.1 Část vývojového diagramu zobrazující časové přerušení 0 a 1.

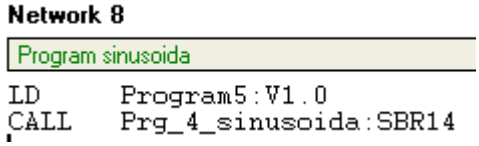
Režie zpracovávání programu v STL módu vytvořeného aplikací STEP 7-Micro/WIN je následující. Spustí se hlavní program (main), ve kterém se postupně od shora dolů zpracovávají instrukce, po dokončení poslední instrukce se program vrátí opět na začátek hlavního programu (cyklické volání). K hlavnímu programu můžeme vytvářet podprogramy a různá přerušení. Pokud proces při zpracovávání hlavního programu narazí na volání podprogramu, odskočí proces do podprogramu, kde opětovně zpracovává instrukce od shora dolů. Po dokončení poslední instrukce podprogramu, se proces vrátí opět do hlavního programu, kde pokračuje ve zpracovávání instrukcí. Podprogramy se mohou volat v každém cyklu hlavního programu nebo např. při splnění určitých podmínek, anebo se volají jenom v prvním cyklu hlavního programu. Doba trvaní jednoho cyklu programu závisí na typu instrukcí, které se zpracovávají a na tom jaké podprogramy jsou volány. Čas potřebný pro vykonání jednoho cyklu programu se může pro stejný program měnit. Většinou se pohybuje v oblasti od minimálního do maximálního času pro cyklus. Pokud potřebujeme provádět nějakou část programu v přesně stanovených intervalech (např. vzorkovací perioda regulátoru), použijeme instrukci časového přerušení. Jedná se o podprogram, který je volán v přesně časově stanovených cyklech. Kromě časového přerušení existuje celá řada přerušení, které se od sebe odlišují událostí, pro kterou se volají.

Strukturu našeho řídícího programu zobrazuje blokové schémata na obrázku 6.0 a 6.1. První blokové schéma představuje chod hlavního programu s voláním jednotlivých podprogramů. Na druhém obrázku je vývojový diagram zobrazující běh programu v časových přerušeních. Nejprve si popíšeme část programu zobrazující na prvním vývojovém diagramu (obr. 6.0) a potom část programu realizující časovým přerušením. Při popisu programu budu vkládat fragmenty kódu vytvořené v STL módu. Celý program bude umístěn v příloze na CD.

6.2 Podrobný popis programu

6.2.1 Hlavní program Main

Jak už zobrazoval vývojový diagram, tak v prvním cyklu programu se volá podprogram “Pocatecni_nastaveni:SBR19”, který se spustí jenom v prvním cyklu. Spuštění podprogramu jenom v prvním cyklu nám zajistí speciální stavový bit SM0.1, který je v log. 1 právě v prvním programovém cyklu. V každém programovém cyklu se nám spouští instrukce USS_DRV_CTRL, která byla popsaná v kapitole věnující se instrukcím USS. O spouštění této instrukce se postará stavový bit SM0.0, který má vždy log. 1. Po instrukci následuje řada podprogramu, které se volají na základě volby obsluhy. Jedná se o jednotlivé programy demonstrující ukázky a možnosti řízení asynchronního motoru pomocí frekvenčního měniče a automatu. Volba programu se provádí prostřednictvím operátorského panelu. Předávání povelů a zadávání požadovaných hodnot pomocí panelu, je realizováno za pomocí globálních proměnných. Ukázka volání podprogramu “Prg_4_sinusoida:SBR14” zobrazuje obrázek 6.2.



Obr. 6.2 Ukázka volání podprogramu.

Pokud bude bit “Program5:V1.0” v log. 1, tak se provede zavolání “CALL” podprogramu sinusoida. Jak si můžeme všimnout, tak datové proměnné si lze pojmenovat pro lepší

přehlednost kódu. Přiřazení názvu a komentář k proměnné se provádí v tabulce symbolů. Malá ukázka takto pojmenovaných a okomentovaných proměnných je na obrázku 6.3.

4			Smer_otaceni	V0.2	Volba směru otáčení motoru
5			Zastav	V0.3	Okamžité zastaveni motoru
6			Program1	V0.4	Volba programu 1
7			Program2	V0.5	Volba programu 2

Obr. 6.3 Možnosti pojmenování a okomentování datových proměnných.

Jednotlivé programy představující ukázky řízení asynchronního motoru si postupně popíšeme. Po zvolení uživatelem některého z nabízených programů můžeme provádět čtení některých základních údajů z měniče. Pro čtení parametrů slouží instrukce USS_RPM_x, která byla popsaná v kapitole věnující se USS instrukcím. Podmínka pro USS instrukce je, že musí být volány z hlavního programu. Další podmínkou v případě instrukce pro čtení, je že může být současně aktivní jenom jedna instrukce pro čtení. V našem programu provádíme čtení čtyř parametrů: aktuální frekvence měniče, výstupní napětí, výstupní proud a teplotu motoru. Proto je potřeba čtyř instrukcí pro čtení, které se od sebe odlišují parametrem, který budou číst. Ukázka instrukce pro čtení výstupního proudu, je zobrazena na obrázku 6.4.

```
Provádí čtení výstupního proudu.
LD      M0.3
=       L1.2
LD      M0.3
EU
=       L2.2
LD      L1.2
CALL    USS4_RPM_R:SBR10, L2.2, 0, 27, 0, &VB1340, Q1.1, Error_USS_RPM:VB14, OUT_I:VD1110
```

Obr. 6.4 Instrukce pro čtení parametrů, nastavená pro čtení výstupního proudu frek. měniče.

Aby bylo zajištěno, že v jednom okamžiku, bude spuštěna vždy jen jedna instrukce pro čtení, bylo třeba vytvořit podprogram, který bude provádět postupné volání instrukcí pro čtení. Tento podprogram má označení “Cteni_par:SBR18” a je spouštěn uživatelem pomocí operátorského panelu. Při volbě jednoho z programů se dostaneme k možnosti volby “Výpis parametrů”, která aktivuje tento podprogram. Přepínání je realizováno pomoví bitů M0.x, který má každá instrukce pro čtení rozdílný. Popis tohoto podprogramu si později popíšeme i s ukázkou zdrojového kódu. Posledním volaným podprogramem v hlavním programu je podprogram “Stav_rotace:SBR17”, který slouží pro zobrazení stavu rotace motoru na operátorském panelu. Máme tři stavy: zastaveno, rotace doleva a rotace doprava. Tento podprogram je spouštěn v každém programovém cyklu.

6.2.2 Podprogram počáteční nastavení SBR19

První co se provede při spuštění tohoto podprogramu je nastavení a vynulování počátečních hodnot globálních proměnných, které v programu využíváme. Část kódu prvního networku je zobrazen na obrázku 6.5. Další network obsahuje volání instrukce USS_INIT, která provede inicializaci komunikace s frekvenčním měničem pomocí USS knihovny.

```
Nastavení a vynulování datových proměnných.
LD      SM0.0                       // log.1 pouze v prvnim cyklu
MOVR    10.0, Poz_ot_motoru:VD1000  // nastavení počáteční frekvence na 25Hz
MOVR    0.0, Pocet_skut_ot:VD1050   // vynulování počítadla otáček
MOVR    0.0, Pocet_poz_ot:VD1054    // nastavení počátečních otáček
R       Program1:V0.4, 1            // provede resetovaní bitu pro spuštění programu1
R       Program2:V0.5, 1            // provede resetovaní bitu pro spuštění programu2
R       Program3:V0.6, 1            // provede resetovaní bitu pro spuštění programu3
R       Program4:V0.7, 1            // provede resetovaní bitu pro spuštění programu4
```

Obr. 6.5 Počáteční nastavení a vynulování používaných proměnných.

Součástí ukázkových programů je i program na regulaci, který využívá PD regulátor pro regulaci polohy a I regulátor pro regulaci rychlosti s dopředným řízením bylo třeba vypočítat derivační složku PD regulátoru a integrační složku I regulátoru. Jelikož je derivační a integrační složka konstantní, bylo výhodné provést výpočet v tomto podprogramu, který se provede jenom jednou. Obrázek 6.6 zobrazuje algoritmus pro výpočet

```
Výpočet derivační složky pro PD regulátor Cd = Kpp*(Td/Ts)
LD      SM0.0                           // log.1
MOVR    0.0, Cd:VD1162                  // vynulujeme promennou Cd
MOVR    Td:VD1134, Cd:VD1162            // do Cd vlozime Td (Cd=Td)
/R      Ts:VD1154, Cd:VD1162            // Cd = Td/Ts
*R      Kpp:VD1078, Cd:VD1162           // Cd = Kp*(Td/Ts)
```

Obr. 6.6 Výpočet derivační složky pro PD regulátor.

derivační složky. Druhý řádek provede vložení nuly do proměnné Cd (vynulování proměnné). Po vynulování vložíme do proměnné Cd hodnotu derivační časové konstanty Td. Následuje podělení vzorkovací periodou Ts a vynásobení proporcionální konstantou Kpp regulátoru polohy. Tímto algoritmem jsme vypočítali derivační složku PD regulátoru. Samotný výpočet PD regulátoru je v časovém přerušení INT5, který si popíšeme později. Obdobně provedeme výpočet integrační složky.

```
Network 5
LD      SM0.0
MOVB    Time_RR:VB15, SMB34             // nastaveni intervalu preruseni na 10 ms
ATCH    Dopredne_rizeni:INT0, 10

Network 6
LD      SM0.0
MOVB    Time_RP:VB10, SMB35             // nastaveni intervalu preruseni na 15 ms
ATCH    Regulace_polohy:INT5, 11
ENI                                     // Povolení všech přerušení.
```

Obr. 6.7 Nastavení časových přerušení 0 a 1.

Dalším krokem je nastavení a povolení časových přerušení (obrázek 5.7). Časová přerušení se mohou volatv rozsahu od 1 ms 255 ms. Pro cyklické přerušení 0 (INT0) se čas zapisuje do SMB34 a pro cyklické přerušení 1 (INT5) do SMB35. V našem případě nezadáváme čas do SMB34 přímo, ale prostřednictvím proměnné Time_RR a Time_RP (SMB35). Hodnoty proměnných Time_RR a Time_RP jsou uloženy v datovém bloku jako konstanty (obrázek 6.8). V našem

```
// vzorkovaci perioda v ms pro regulaci polohy
Time_RP:VB10 15

// vzorkovaci perioda v ms pro regulaci rychlosti
Time_RR:VB15 10
```

Obr. 6.8 Ukázka konstant uložené v datovém bloku.

případě byla zvolena časová konstanta 10 ms pro vnitřní smyčku regulace rychlosti a 15 ms pro vnější smyčku regulace polohy. Instrukce ATCH označuje přerušení a číslovka na konci řádku značí událost, na kterou bude přerušení reagovat (10, 11 = časová přerušení). Instrukce ENI provádí povolení všech přerušení, která se budou volat.

Posledním networkem v podprogramu počátečního nastavení je instrukce vysokorychlostního čítače. Nastavení vysokofrekvenčního čítače je zobrazeno na obrázku 6.9. Speciální proměnná SMB147 představuje konfigurační bity vf. čítače. Vysokofrekvenční čítač má nastaven startovací a nulovací vstup na aktivní úroveň a je zvolen 1x režim čítání

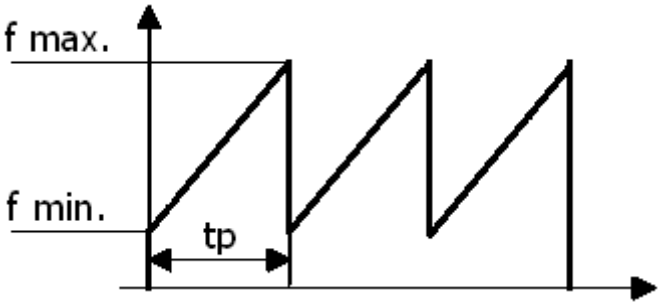
(SMB147 = FC). Definice vysokorychlostního čítače se provádí instrukcí HDEF. První parametr instrukce udává číslo čítače a druhý parametr definuje režim, ve kterém bude čítač pracovat. Čítač HSC4 pracuje jako A/B fázový kvadraturní čítač, který podporuje 3 vstupy. Vstupy jsou: hodiny A, hodiny B a nulovací vstup, který není v režimu 9 podporován. Speciální proměnná SMD148 představuje aktuální hodnotu čítače, která je při počáteční konfiguraci čítače vynulována. Poslední řádek (HSC 4), provede naprogramování čítače HSC4. Toto je poslední instrukce, která se provádí v podprogramu počátečního nastavení a tím tedy opouštíme popis tohoto podprogramu.

```
Počáteční nastavení vysokorychlostního čítače
LD      SM0.0                   // log. 1
MOVB    16#FC, SMB147           // Načtení řídicích bitů pro HSC4.
HDEF    4, 9                    // HSC4, režim 9 - A/B fázový kvadraturní čítač
MOVD    +0, SMD148              // Vynulování aktuální hodnoty čítače HSC4.
HSC     4                       // povolení čítače HSC 4
```

Obr. 6.9 Nastavení vysokofrekvenčního čítače HSC4 v režimu 9.

6.2.3 Program generování pilových průběhů SBR0

Jak už napovídá název programu, tento podprogram řídí frekvenci měniče (otáčky motoru) pomocí pilového průběhu. Jedná se tedy o cyklické generování pilových průběhů (obrázek 6.10). Tvar pily můžeme měnit pomocí tří parametrů, jedná se o nastavení minimální a maximální frekvence a času rozběhu tp. Čas tp se zadává v sekundách a představuje dobu, za kterou se pila dostane s minimální frekvence na maximální frekvenci. Frekvence min. a max. se nezadává v Hz, ale v procentech, kdy platí 100% = 50 Hz.



Obr. 6.10 Ukázka generování pilových průběhů.

Obecný popis tohoto programu máme za sebou, teď se můžeme věnovat popisu kódu vytvořeného v STEP7-Micro/WIN. Začneme popis od druhého networku (obrázek 6.11), protože v prvním je jenom počáteční nastavení. První co provedeme je přesun proměnné f_max do lokální proměnné f_roz. Poté provede odečtení f_min od f_max a dostaneme tak rozdíl f_roz. Další řádek provede přesunutí globální proměnné “Rozbeh_time:VW6”, kterou zadává obsluha prostřednictvím OP do lokální proměnné “#Cas_roz:LW8”. V lokální proměnné máme nyní čas v sekundách, za který se musí změnit frekvence z minimální na maximální hodnotu. Pro zvyšování požadované frekvence se používá časovač zpožděného zapnutí “TON T97“, který je třetím networku (obrázek 6.12). Časovač T97 má základní časovou jednotku 10 ms a číslice za instrukcí časovače udává kolikrát je časová jednotka vynásobena (v našem případě to je 5 x 10 ms = 50ms). To znamená, že po uplynutí 50 ms dojde k sepnutí výstupu časovače. Zvyšování frekvence se tedy provádí každých 50 ms.

```
Network 2
LD      SM0.0                          // log. 1
MOVR    f_max:VD1016, #f_roz:LD0       // přesun f_max do f_roz
-R      f_min:VD1012, #f_roz:LD0       // f_roz = f_max - f_min
MOVW    Rozbeh_time:VW6, #Cas_roz:LW8 // presun VW6 do LW8
*I      20, #Cas_roz:LW8               // převod s na 50ms
ITD     #Cas_roz:LW8, LD10             // převod integer na double integer
DTR     LD10, LD10                     // převod double integer na real
/R      LD10, #f_roz:LD0               // LD0 = LD0/LD10
```

Obr. 6.11 Network 2 podprogramu generování pilových průběhu.

Abychom zajistili, že po uplynutí stanovené doby, byla skutečná frekvence rovna maximální frekvenci musíme vypočítat přírůstek, o který se bude skutečná frekvence zvyšovat. Tímto se vracíme k pokračování popisu zbylých instrukcí druhého networku. Čas pro rozběh máme uložený v lokální proměnné LW8, kterou následně vynásobíme číslicí 20. Vynásobením času dostaneme počet cyklů, za kterých se musí skutečná frekvence rovnat maximální frekvenci při použití 50 ms časovače. Následně proměnnou LW8 převedeme na double integer a vložíme do proměnné LD10, kterou následovně převedeme na real. Poslední instrukce provede podělení f_roz s naším upraveným časem a výsledek uloží do f_roz. Tímto máme v proměnné f_roz uložen přírůstek frekvence pro časovač. Network 2 máme tedy popsaný a můžeme se vrátit k popisu třetího. Aby byl časovač aktivní, musí být splněny dvě podmínky: skutečná frekvence musí být menší než maximální a současně musí být zapnut pohon.

```
Network 3
LDR<    Poz_ot_motoru:VD1000, f_max:VD1016 // pokud pozadovaná frek. je m
A       RunPohonu:V0.1                 // log.1 když je pohon zapnut
TON     T97, 5                         // časovač 10 ms cyklus (5x10 = 50ms)
```

Obr. 6.12 Network 3 časovač zpožděného zapnutí.

Network 4 (obrázek 6.13) provádí každých 50 ms zvýšení skutečných otáček o vypočítaný přírůstek f_roz. Zvyšování se provádí do doby, kdy maximální frekvence se rovná skutečné frekvenci. Po zvýšení frekvence se provede reset čítače, aby se znovu napočítalo 50 ms, tím je zajištěna opakující se smyčka.

```
Network 4
LD      T97                                 // on T97
EU
+R      #f_roz:LD0, Poz_ot_motoru:VD1000
R       T97, 1
```

Obr. 6.13 Network 4 provádí zvyšování skutečné frekvence o vypočítaný přírůstek.

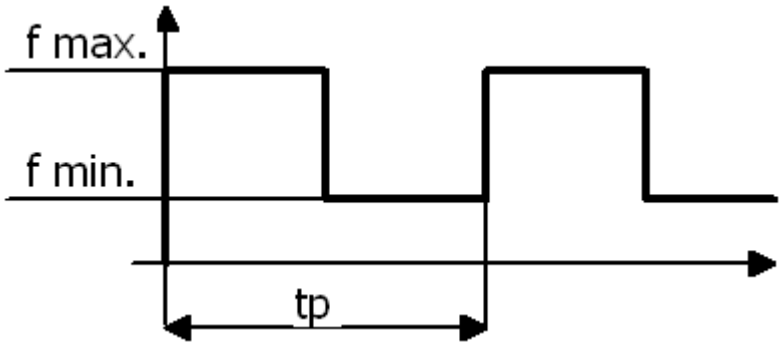
Poslední network tohoto podprogramu je zobrazen na obrázku 6.14. Tato část kódu provádí kontrolu, zdali skutečná frekvence je větší nebo rovna požadované maximální frekvenci. Pokud je podmínka splněna je do skutečné frekvence vložena hodnota s f min. Tímto proběhla jedna pila a celý proces se opakuje. Tímto byl podprogram, generování pilových průběhů popsán a přejdeme k popisu dalšího programu.

```
Network 5
LDR>=   Poz_ot_motoru:VD1000, f_max:VD1016 // pokud pož. frek = f max pak
MOVR    f_min:VD1012, Poz_ot_motoru:VD1000 // vloží min. frek. do pož. frek.
```

Obr. 6.14 Poslední network podprogramu generování pilových průběhů.

6.2.4 Program na generování obdélníkových průběhů

Jedná se o obdobu předchozího programu, s tím rozdílem, že tentokrát jsou otáčky motoru řízeny obdélníkovým průběhem (obrázek 6.15). Opětovně můžeme tvar obdélníkového průběhu ovlivnit pomocí tří parametrů. Jedná se o maximální, minimální frekvenci a periodu tp. Opět platí, že frekvence se zadává v procentech a perioda v sekundách. Obecné seznámení s tímto programem máme za sebou a teď si popíšeme zdrojový kód.



Obr. 6.15 Ukázka obdélníkového průběhu, pro řízení motoru.

Network 1 si popisovat nebudeme, protože je v něm jenom počáteční nastavení tohoto podprogramu. Network 2 (obrázek 6.16) je částečně shodný s druhým networkem předchozího podprogramu, ale tentokrát je mnohem jednoduší. První provedeme přesunutí globální proměnné "Rozbeh_time:VW6" do lokální proměnné "#Cas_per:LW4". Tímto máme v lokální proměnné uložen čas periody v sekundách, který vynásobíme hodnotou 10. Vynásobení hodnotou 10 provádíme z toho důvodu, že v tomto programu používáme časovače se základním rozlišením 100 ms. Tím dostaneme číslo, které udává, kolikrát se 100 ms vleze do požadované periody, která je v sekundách. Posledním krokem je podělení periody dvojkou, kdy dostaneme polovinu periody pro každý časovač. Tímto je druhý network popsaný a můžeme přejít na třetí.

```
Network 2
LD      SM0.0                          // log. 1
MOVW    Rozbeh_time:VW6, #Cas_per:LW4 // VW6 přesunuta do LW4
*I      10, #Cas_per:LW4               // převod sekundy na 100ms
/I      2, #Cas_per:LW4                // podelením dostaneme 1/2 periody
```

Obr. 6.16 Network 2 podprogramu na generování obdélníkových průběhů.

Činnost networku 3 (obrázek 6.17) je následující: pokud je spuštěný pohon V0.1 provede se načtení maximální frekvence do požadované frekvence. Současně se spustí časovač zpožděného zapnutí (TON T39), který odpočítává čas poloviny periody. Po odpočítání tohoto času se aktivuje network 4 (obrázek 6.18).

```
Network 3
LD      RunPohonu:V0.1
MOVR    f_max:VD1016, Poz_ot_motoru:VD1000
TON     T39, #Cas_per:LW4              // 100 ms časovač x 1/2 periody
```

Obr. 6.17 Aktivace 100ms časovače zpožděného zapnutí T39.

Po odpočítání poloviny periody časovačem T39 se provede přesun minimální frekvence do požadovaných otáček a spustí se odpočítávání poloviny periody časovačem zpožděného

zapnutí T40. Časovač T40 patří do kategorie časovačů se základním rozlišením 100 ms. Po odpočítání poloviny periody se aktivuje network 5 (obrázek 6.19).

```
Network 4
LD      T39
MOVR    f_min:VD1012, Poz_ot_motoru:VD1000
TON     T40, #Cas_per:LW4           // 100 ms časovač x 1/2 periody
```

Obr. 6.18 Aktivace časovače zpožděného zapnutí T40.

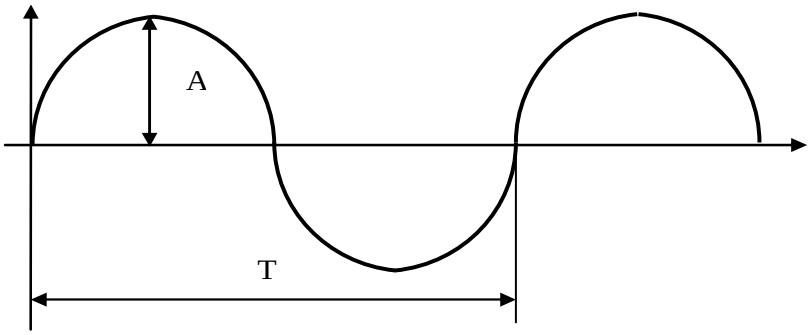
Po aktivaci networku 5 se provede reset časovačů T39 a T40. Tímto proběhla jedna perioda obdélníkového průběhu. Po resetu obou časovačů se program vrátí do networku 2 a celý cyklus se opakuje. Program bude cyklicky běžet do doby, kdy bude pohon aktivovaný a nedojde ke změně volby ukázkového programu.

```
Network 5
LD      T40                         // on T40
R       T39, 1                      // reset T39
R       T40, 1                      // reset T40
```

Obr. 6.19 Network 5 – provedení resetu obou čítačů.

6.2.5 Řízení otáček motoru pomocí sinusoidy

Tento program je založen na generování sinusoidy (obrázek 6.20), která se přenáší na otáčky motoru. Tvar sinusoidy generované programem můžeme měnit za pomocí dvou parametrů. Jedná se o změnu amplitudy A, kterou lze měnit v rozsahu 10 – 100% a dobu periody T, kterou lze měnit v rozmezí od 10 s do 300 s. Tímto jsme v jednoduchosti popsali chování tohoto programu a můžeme přejít na popis kódu programu.



Obr. 6.20 Ukázka sinusoidy a parametry, pomocí kterých můžeme ovlivňovat její tvar.

Opětovně při popisu vynecháme první network, který pro pochopení tohoto programu není důležitý. Začneme tedy s popisem od druhého networku (obrázek 6.21). Nejprve globální proměnnou VD1170, do které uživatel pomocí OP zapisuje požadovanou periodu v sekundách vložíme do lokální proměnné LD12, abychom s ní mohli dále pracovat. Tuto proměnnou převedeme s double wordu a na real a vložíme do pomocné proměnné LD22. Protože pracujeme s časovačem T96, jehož základní rozlišení je 1 ms a pracujeme v rozsahu 0 – 360 (tedy 360° na jednu sinusoidu). Musíme pomocnou proměnnou LD22 podělit číslicí 360 a současně vynásobit hodnotou 1000. Tímto dostaneme časovou periodu pro časovač T96. Tuto hodnotu musíme zaokrouhlit na celé číslo pomocí instrukce ROUND a výsledek vložíme do pomocné proměnné LD16. Tato proměnná je typu double word, ale parametr pro

časovač musí být typu word. STEP7-Micro/Win nepodporuje instrukci pro převod proměnné double word na word, takže musíme použít akumulátor AC0. Do akumulátoru vložíme proměnou typu double word a přesuneme ji s akumulátoru do proměnné "#Perioda:LW30" jako word. Tímto způsobem jsme vytvořili převod proměnné typu double word na word. Takto upravenou periodu vložíme do časovače. Tímto je network 2 popsán a můžeme přejít na popis třetího.

```
Network 2
LD      SM0.0
MOVD    Perioda:VD1170, #Poz_perioda:LD12 // Globální proměnnou perioda vlo
DTR     #Poz_perioda:LD12, #Pom2:LD22 // Prevedeme na realné číslo
/R      360.0, #Pom2:LD22             //     Podělíme 360 a dostaneme přírust
*R      1000.0, #Pom2:LD22            // Prirustky vynásobíme 1000 a dostanem
ROUND   #Pom2:LD22, #Pom3:LD16        // Zaokrouhlíme na celé číslo
MOVD    #Pom3:LD16, AC0               // Vložíme DW do AC0
MOVW    AC0, #Perioda:LW30            //Vložíme AC0 do LW26 = prevod DW na W

LD      RunPohonu:V0.1
TON     T96, #Perioda:LW30            // Vlozime upravenou periodu do čítače
```

Obr. 6.21 Network 2 výpočet periody pro časovač T96.

Při každém zapnutí časovače T96 (obrázek 6.22), se proměnná "#Stupne:LD26" zvětší o jedničku, což zajistí instrukce inkrementace double word INCD. Tato proměnná bude postupně nabývat hodnot od 0 do 360 (generování stupňů pro funkci sinu). Po zvětšení proměnné stupně o jedničku se provede vynulování časovače T96, který opětovně začne odpočítávat danou periodu. Tímto máme zajištěn cyklický chod programu.

```
Network 3
LD      T96                           // 1ms x perioda
EU
INCD    #Stupne:LD26                  // VD16 = VD16+1
MOVW    0, T96                        // vynulování časovače T32
```

Obr. 6.22 Postupné zvětšování proměnné stupně o jedničku.

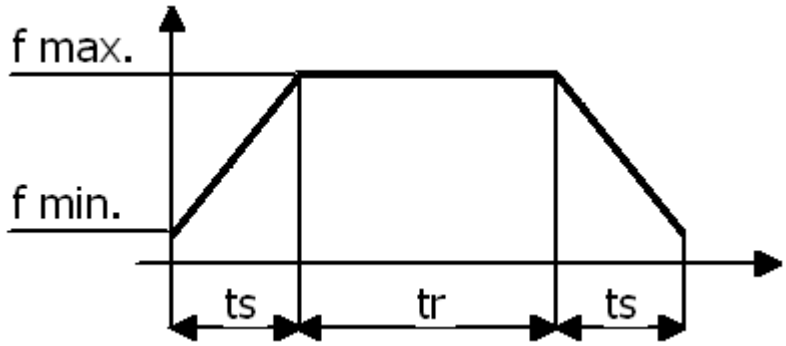
Poslední network (obrázek 6.23) provádí generování sinusoidy. Nejprve proměnou stupně přesuneme do lokální proměnné LD4, kterou převedeme na reálné číslo a uložíme do proměnné "#Radiany:LD0". Po vynásobení úhlu ve stupních převodovým koeficientem dostáváme úhel v radiánech. Tuto hodnotu vložíme do instrukce SIN, jako první parametr a výsledek této funkce je vložen v proměnné "#Sinus:LD8". Výsledkem je hodnota v rozmezí (-1 až +1). Po vynásobení této hodnoty amplitudou dostáváme výslednou frekvenci v procentech, která se při maximální amplitudě pohybuje v rozsahu -100 až +100%. Tuto výslednou frekvenci vložíme do proměnné požadované otáčky motoru VD1000. Jako poslední se provádí test, zda stupně dosáhly hodnoty 360 a pokud ano, je provedeno vynulování této proměnné. Tímto je program pro generování sinusoidy popsán a můžeme přejít na popis dalšího programu reprezentujícího možnosti řízení motoru pomocí frekvenčního měniče a programovatelného automatu.

```
Network 4
LD      SM0.0
MOVD    #Stupne:LD26, #Pom:LD4        // stupne vlozime do pomocne promenne
DTR     #Pom:LD4, #Radiany:LD0        // prevedeme na realné číslo
*R      0.01745329, #Radiany:LD0      //převod stupňů na radiany
SIN     #Radiany:LD0, #Sinus:LD8      // funkce sinu
*R      Amplituda:VD1174, #Sinus:LD8 // vynásobení hodnoty amplitudou v rc
MOVR    #Sinus:LD8, Poz_ot_motoru:VD1000 // výslednou hodnotu vložíme do \
LDD>=   #Stupne:LD26, 360             // Pokud je už 360 stupnu, tak se prov
MOVD    0, #Stupne:LD26
```

Obr. 6.23 Network4 – funkce sinu.

6.2.6 Řízení otáček pomocí rampy

Tato varianta programu je asi ze všech nejsložitější a to jak co do počtů parametrů kterými můžeme měnit tvar rampy, tak i při vytváření programového kódu v STEP 7. Mezi parametry, které můžeme měnit patří minimální a maximální frekvence, se kterými jsme se setkali již v minulých programech (kromě sinusoidy). Další možností je změna doby rozběhu/doběhu rampy ts a doby tr, při které se motor otáčí konstantními otáčkami f max. Oba časové parametry zadáváme v sekundách prostřednictvím OP. Tímto je stručný popis za námi a můžeme se věnovat popisu zdrojového kódu.



Obr. 6.24 Řízení otáček motoru pomocí rampy.

Začneme s popisem od druhého networku (obrázek 6.25). Nejprve provedeme přesun globálních proměnných VD1016, VW20, VW6 do lokálních proměnných LD0, LW6 a LW4. Jedná se tedy o maximální frekvenci, čas pro rampu a čas pro rozběh/doběh rampy.

Network 2

```
LD      SM0.0
MOVR    f_max:VD1016, #f_roz:LD0      // přesun VD1016 do LD0
MOVW    Rampa_time:VW20, #tr:LW6      // čas pro rampu
MOVW    Rozbeh_time:VW6, #ts:LW4      // čas pro rozběh/doběh rampy
-R      f_min:VD1012, #f_roz:LD0      // LD0 = f_max - f_min
*I      100, #ts:LW4                  // převod času roz/dob rampy s => 100 ms
ITD     #ts:LW4, LD14                 // převod integer na double integer
DTR     LD14, LD14                    // převod double integer na real
MOVR    #f_roz:LD0, #Prirustek:LD8    //přesun LD0, do LD8
/R      LD14, #Prirustek:LD8          // LD8 = LD8/LD14
*I      100, #tr:LW6                  // převod času rampy s => 100ms
```

Obr. 6.25 Network 2 programu pro generování rampy.

Dále vypočítáme rozdíl mezi maximální a minimální frekvencí, kterou uložíme do proměnné "#f_roz:LD0". Následuje úprava času pro časovač T99, který má základní rozlišení 10 ms, proto vynásobíme čas ts hodnotou 100. Následně provedeme dva převody, kdy čas ts typu integer převedeme na reálné číslo, které vložíme do lokální proměnné LD14. Dalším krokem je přesun proměnné LD0 do proměnné "#Prirustek:LD8", která představuje přírůstkovou hodnotu pro časovač. Po podělení proměnné LD8 časem uloženým v proměnné LD14, dostáváme daný přírůstek pro časovač. Posledním krokem je přizpůsobení času tr pro časovač zpožděného zapnutí T99. Popis tohoto network máme za sebou a můžeme přejít k popisu dalšího s číslem 3 (obrázek 6.26).

```
Network 3
LD      RunPohonu:V0.1                  // pohon musí být aktivní
EU                                      // náběžná hrana
MOVR    f_min:VD1012, Poz_ot_motoru:VD1000 // vložení min. frek do pož.

LD      RunPohonu:V0.1                  // pohon musí být aktivní
AN      On_Off_cit_rampa:M1.2           // zajišťuje automatické nulování čítače
TON     T99, 1                          // každých 10 ms se volá časovač
```

Obr. 6.26 Aktivace časovače zpožděného zapnutí T99.

Při přepnutí pohonu z neaktivního do aktivního pohonu se provede načtení nastavené minimální frekvence do požadované frekvence VD1000. Zajištění, že se frekvence načte jenom při přechodu z off do on pohonu zajišťuje instrukce náběžná hrana EU. Časovač T99 zajišťuje řízení jak doby ts, tak i tr. Časovač je aktivní, jestliže je pohon aktivní a současně bit M1.2 je v logické 0. Bit M1.2 slouží pro automatické nulování čítače. To je tak asi vše co se dá říct o této části kódu a proto přejdeme na network s číslem 4 (obrázek 6.27).

```
Network 4
LD      T99                             // časovač napočítal 10 ms
LDR>=   Poz_ot_motoru:VD1000, f_max:VD1016 // kontrola zdali pož. frek. je v
MOVR    f_max:VD1016, Poz_ot_motoru:VD1000 // pokud ano je f max přesunuto d
A       L30.0                           // musí být v log. 1
S       L30.1, 1                        // nastavení bitu L30.1
JMP     1                               // odskok do networku 5
LD      T99                             // časovač napočítal 10 ms
A       L30.0                           // současně je L30.0 v log. 1
EU                                      // naběžná hrana
+R      #Prirustek:LD8, Poz_ot_motoru:VD1000 // pož. frek zvětšit o přírust
R       L30.1, 1                        // vynulování bitu L30.1
```

Obr. 6.27 Kontrola zdali je požadovaná frek. větší nebo rovna f max, pokud ne je zvětšena.

Po té co časovač napočítá čas 10 ms je nastaven na log.1. Dalším krokem je kontrola, zdali skutečná frekvence měniče je větší nebo rovna nastavené maximální frekvenci. Pokud se skutečná rovná maximální a současně je řídící bit L30.0 aktivní je provedeno nastavení bitu L30.1 na log. 1 a současně se provede odskok do networku 5. Odskok se provede instrukcí JMP 1, kdy je odskočeno na instrukci LBL 1, která je první instrukcí v networku 5. Pro některé případy nastavení času a rozdílem frekvencí max. a min může nastat, že skutečná frekvence je o přírůstek větší než je maximální frekvence. Z tohoto důvodu se po splnění podmínky, že pož. frek. je větší nebo rovno max. frekvenci je hodnota s f max. vložena do pož. frekvenci, tím je zajištěna vždy přesná hodnota. Pokud není splněna zmiňovaná podmínka, jsou popisované instrukce přeskočeny a dostaneme se na další instrukci časovače T99. Kdy při každém přechodu časovače z off/on a současně při splnění podmínky L30.0 je v log. 1, dojde ke zvětšení požadované frekvenci o vypočítaný přírůstek LD8 a současně se provede reset bitu L30.1. Tímto máme zajištěno, že pokud není splněna podmínka, dochází k postupnému zvětšování skutečné frekvence až do doby kdy se skutečná a maximální frekvence rovnají. Po rovnosti je zastaveno zvětšování požadované hodnoty o přírůstek, protože je proveden odskok do networku 5 (obrázek 6.28).

Po odskoku na instrukci LBL 1 se pokračuje ve vykonávání následujících instrukcí. Máme zde instrukci časovače T99 a při každém přechodu časovače z off/on dojde ke zvětšení proměnné LW40 o jedničku. Nutnou podmínkou pro přičítání je nutnost aktivního bitu L30. Nejprve se provede porovnávání, zdali je LW40 větší nebo rovno upravené doby tr. Po splnění této podmínky nastane reset bitů L30.0 a L30.1 a vynulování proměnné LW40. Do této části kódu bylo zajištěno, že se nám hodnota postupně zvětšovala z min frekvence na maximální. Po dosažení max. frekvence zůstala po dobu tr frekvence na maximální hodnotě a teď nastává situace, kdy se bude hodnota postupně zmenšovat na hodnotu f minimální. Pokud

tedy je L30.0 v log. 0 a časovač je sepnut, nastane ověření podmínky, zdali min. frek. je rovna nebo větší než požadovaná frekvence. Pokud je podmínka splněna, je provedeno nastavení bitu L30.0 na log. 1 a provede se odskok do posledního networku. Naopak pokud není provedeno odečtení hodnoty přírůstku od požadované frekvence a současně se resetuje bit L30.0. Odečítání se provádí dokud není splněna předchozí podmínka.

```
Network 5
LBL     1                                   // odskok z networuku 4
LD      T99                                 // časovač napočítal 10 ms
A       L30.1                               // musí být v log. 1
EU                                          // náběžná hrana
INCW    LW40                                // LW40=LW40+1
LDW>=   LW40, #tr:LW6                       // Pokud LW40 >= tr pak
R       L30.0, 1                            // se provede reset L30.0
R       L30.1, 1                            // a reset L30.1
MOVW    0, LW40                             // vynulování LW40

LDN     L30.0                               // pokud L30.0 je v log. 0
A       T99                                 // a současně časovač napočítal 10 ms
LDR<=   Poz_ot_motoru:VD1000, f_min:VD1012 // podmínka f min >= pož. ot.
S       L30.0, 1                            // pokud ano je provedeno set L30.0
JMP     2                                   // odskok do networku 6

LDN     L30.0                               // pokud L30.0 je v log. 0
A       T99                                 // a současně časovač napočítal 10 ms
-R      #Prirustek:LD8, Poz_ot_motoru:VD1000 // provede se zmenšení pož. f
R       L30.0, 1                            // reset bitu L30.0
```

Obr. 6.28 Kód pro řízení časového průběhu tr a následné klesání frek. na min. frekvenci.

Po odskoku na LBL 2 se dostáváme do posledního networku (obrázek 6.29) tohoto podprogramu. Tato část kódu provádí resetování čítače každých 10 ms, po té, co se provedla celá část kódu. Když je časovač sepnut je současně bit “On_Off_cit_rampa:M1.2” nastaven na log. 1. Ve třetím networku tohoto programu je podmínka, že časovač je aktivní pokud je zapnut pohon a současně je bit M1.2 v log. 0. Tím, že se bit M1.2 přejde do log. 1 se provede vypnutí časovače a tedy jeho vynulování, ale tím se opětovně bit M1.2 resetuje a tak dojde k opětovnému zapnutí časovače. Tento proces se cyklicky opakuje.

```
Network 6
LBL     2                               // odskok z networku 5
LD      T99                             // pokud časovač napočítal 10 ms
=       On_Off_cit_rampa:M1.2           // pak se provede jeho reset
```

Obr. 6.29 Poslední část kódu, zajišťující resetování čítače každých 10 ms.

6.2.7 Program pro ruční řízení motoru

Tento program umožňuje uživateli ovládat motor manuálně. Uživatel prostřednictvím OP může měnit velikost požadované frekvence měniče, která se zadává v procentech. Dále může měnit směr otáčení a provádět zapnutí nebo vypnutí pohonu. Tyto základní úkony lze realizovat přímo prostřednictvím instrukce USS_DRV_CTRL, která je volána z hlavního programu, kdy pomocí OP přistupujeme k jejím parametrům. Zdálo by se, že je tedy zbytečné vytvářet podprogram pro manuální řízení, když přistupujeme přímo k instrukci USS. Program manuální řízení byl ale doplněn o dvě tlačítka, pomocí nichž provádíme zvýšení nebo snížení požadované frekvence o jeden procentní bod. Kód pro zvyšování frekvence je zobrazen na obrázku 6.30. Při stisknutí tlačítka přidat otáčky, které je provázáno s bitem "Pridat_ot:V1.2", se při každém sepnutí zvýší požadovaná frekvence o jedno procento. Následuje resetování stavu bitu V1.2, který při stisknutí tlačítka přidat otáčky zůstane v log. 1. Tím je zajištěno, že při zmáčknutí tlačítka "přidej otáčky", se frekvence zvětší o jeden procentní bod. Obdoba tohoto tlačítka je tlačítko "ubrat otáčky", které provede zmenšení frekvence o jeden procentní bod. Kód je podobný s tím rozdílem, že tam není instrukce součet +R, ale instrukce rozdíl -R. A tlačítko "ubrat otáčky" ovládá bit V1.3. Ještě pro doplnění, tak v programu pro manuální řízení můžeme na OP sledovat skutečné otáčky motoru (ot/min) a stav rotace.

```
Network 1
LD      Pridat_ot:V1.2
EU
+R      1.0, Poz_ot_motoru:VD1000
EU
R       Pridat_ot:V1.2, 1
```

Obr. 6.30 Tlačítko pro zvětšování frekvence.

6.2.8 Program pro řízení instrukcí pro čtení parametrů z frekvenčního měniče

Jak už bylo napsáno, tak instrukce pro čtení parametrů slouží pro řízení přepínání instrukcí pro čtení USS_RPM_R, které jsou v hlavním programu čtyři. Tento podprogram se stará, aby byla v jednu dobu pouze jedna aktivní instrukce pro čtení, jinak by docházelo k datovým kolizím, kdy by instrukce pro čtení generovala chybu. Každá instrukce pro čtení je aktivována pomocí bitu řady M0.x, kdy stav těchto bitů řídí podprogram pro čtení instrukce. Další funkcí tohoto podprogramu je odstranění vnitřních chyb instrukce pro čtení, které jsou nejspíš dány neodladěním této instrukce. První problém je, že když aktivujeme instrukci pro čtení a současně bit V0.2 pro směr otáčení je v log. 1, dochází při čtení parametrů z měniče k neustálému překlápění směru rotace motoru. Tento neduh lze odstranit tím, že bit pro směr rotace resetujeme na log. 0 před spuštěním instrukce pro čtení. Reset provádíme v případě kdy bit V0.2 je v log. 1. Tato negace proměnné V0.2 způsobuje, že při zapnutí výpisu parametrů z měniče, provede změnu směru motoru (v případě, že V0.2 je v log. 1). Tuto negaci řeší první dvě instrukce prvního networku (obrázek 6.31).

```
Network 1     Network Title
LD      Smer_otaceni:V0.2          // pokud je smer v log.1
R       Smer_otaceni:V0.2, 1       // tak provede změna směru rotace

LDN     On_Off_citac:M1.0          // zajišťuje automatické nulování čítače
TON     T100, 10                   // 10 x 10ms = 100ms
```

Obr. 6.31 Network 1 podprogramu pro čtení parametrů z frekvenčního měniče.

Dalším neduhem této instrukce je, že při provádění čtení parametrů se náhodně ukládá nulová hodnota do výsledných parametrů. Zkoušel jsem nastavovat různé adresy komunikačních zásobníků, kde každá instrukce má svůj vlastní. Dále jsem zkoušel měnit adresy proměnných, do kterých se ukládá výsledek instrukce pro čtení. Bohužel nic nepomohlo, tak jsem provedl ošetření této instrukce. Ošetření spočívá v tom, že vezmeme proměnnou, ve které je uložen výsledek instrukce a provedeme kontrolu, jestli je tato hodnota různá od nuly. Pokud je různá od nuly, vložíme ji do druhé globální proměnné, která slouží pro zobrazení údajů na OP. Pokud se tedy objeví nulová hodnota některého ze čtených parametrů při spuštěném pohonu, je tato hodnota ignorována a není přenášena do OP. Toto ošetření řeší část programu umístěné ve třetím networku (obrázek 6.33). Teoretický rozbor jsme už provedli a teď se můžeme věnovat popisu samotného kódu tohoto podprogramu.

V prvním networku je část kódu pro změnu rotace a aktivování časovače zpožděného zapnutí T100, který odpočítává dobu 100 ms pro jeho sepnutí. Reset čítače je řešeno pomocí bitu “On_Off_citac:M1.0”. To jsou všechny instrukce v prvním networku a můžeme přejít k popisu dalšího.

Network 2 (obrázek 6.32) obsahuje část kódu pro řízení přepínání instrukcí pro čtení. S každou náběžnou hranou čítače T100 se zvětší hodnota počítadla čítače o jedničku. To znamená, že každých 100 ms se tato hodnota počítadla zvětší. Na začátku kdy je hodnota nulová dojde k sepnutí bitu M0.1, který patří první instrukcí pro čtení (aktuální otáčky měniče). Po uplynutí času 100 ms se nám počítadlo zvětší na jedničku, tím dojde resetování bitu M0.1 a současně nastavení bitu M0.2 na log. 1. Tímto způsobem se postupně přepínají všechny bity M0.x a tím i jednotlivé instrukce pro čtení. Po provedení všech instrukcí se provede vymazání počítadla čítače. Tímto je program připraven na další cyklus. Kód pro řízení přepínání instrukcí pro čtení je popsán a můžeme přejít na další network s číslem 3.

```
Network 2
LD      T100                          // načteno 100 ms
EU                                    // náběžná hrana
INCW    Pocitadlo_citac:VW22          // počítadlo čítače
LDW=    Pocitadlo_citac:VW22, 0       // pokud je VW22 nula
S       M0.1, 1                       // set bitu M0.1 na log. 1
LDW=    Pocitadlo_citac:VW22, 1       // pokud je VW 22 jedna
R       M0.1, 1                       // reset bitu M0.1
LDW=    Pocitadlo_citac:VW22, 1       // celé se to opakuje
S       M0.2, 1                       //....
LDW=    Pocitadlo_citac:VW22, 2
R       M0.2, 1
LDW=    Pocitadlo_citac:VW22, 2
S       M0.3, 1
LDW=    Pocitadlo_citac:VW22, 3
R       M0.3, 1
LDW=    Pocitadlo_citac:VW22, 3
S       M0.4, 1
LDW=    Pocitadlo_citac:VW22, 4
R       M0.4, 1
MOVW    0, Pocitadlo_citac:VW22       // vynulování počítadla
```

Obr. 6.32 Část kódu pro řízení přepínání instrukcí pro čtení.

Network 3 se stará o odstranění druhého popisovaného neduhu instrukce pro čtení. První se provede porovnání proměnné “aktualni_ot_menic:VD1102”, zdali je různá od nuly. Pokud je hodnota různá od nuly je přesunuta do proměnné “ak_ot_mne_panel:VD1118”, která slouží pro zobrazení aktuálních otáček na OP. Jestliže je hodnota rovna nule neprovede se přesun a tím je zajištěno, že se na panelu nebude zobrazovat neplatná hodnota parametru aktuální otáčky. Tímto způsobem jsou ošetřeny i ostatní parametry. Jedná se výstupní napětí, výstupní proud a teplota motoru. Neposlední řadě musíme zajistit, aby po vypnutí pohonu se

první tři parametry vynulovaly, protože teplota motoru je nenulová i po vypnutí pohonu. Vynulování se provádí pomocí bitu V0.1, který pokud není aktivní, provádí nulování prvních tří parametrů.

```
Network 3
LDR<>  aktualni_ot_menic:VD1102, 0.0 // pokud je hod. VD1112 různá od nu
MOVR   aktualni_ot_menic:VD1102, ak_ot_mne_panel:VD1118 // presun VD1102
LDR<>  OUT_U:VD1106, 0.0           // obdoba
MOVR   OUT_U:VD1106, OUT_U_panel:VD1122 //...
LDR<>  OUT_I:VD1110, 0.0
MOVR   OUT_I:VD1110, OUT_I_panel:VD1126
LDR<>  OUT_Tep:VD1114, 0.0
MOVR   OUT_Tep:VD1114, OUT_Tep_panel:VD1130

LDN    RunPohonu:V0.1              // pokud není pohon aktivní pak
MOVR   0.0, ak_ot_mne_panel:VD1118 // vynuluje hodotu VD1118
MOVR   0.0, OUT_U_panel:VD1122     //...
MOVR   0.0, OUT_I_panel:VD1126
```

Obr. 6.33 Zdrojový kód pro ošetření vnitřní chyby instrukce pro čtení.

Poslední část podprogramu (obrázek 6.44) pro čtení parametrů provádí nulování čítače pomocí bitu M1.0. Princip je stejný jako u podprogramu řízení otáček pomocí rampy.

```
Network 4

LD     T100                    // načteno 100 ms
=      On_Off_citac:M1.0       // reset čítače
```

Obr. 6.44 Resetování časovače po uplynutí doby 100 ms.

Tímto jsme popsali všechny důležité podprogramy, které se týkají části pro řízení otáček motoru. Nyní přejdeme na popis programů věnující se úloze pro regulaci polohy motoru.

6.2.9 Časované přerušení 1 (INT5)

Jedná se o časové přerušení s nižší prioritou než má časované přerušení 0. Toto přerušení v našem programu má čas cyklu nastaven na 15 ms, který se předává časovému přerušení 1 vložením do speciální proměnné SMB35. Pomocí tohoto přerušení realizujeme vnější smyčku regulátoru polohy, která je realizována pomocí PD regulátoru. Popis regulátoru a jeho rovnic jsme si popsali už dříve a tak můžeme přejít na popis kódu, kterým je realizován PD regulátor.

Začneme popisem od prvního networku (obrázek 6.45). Tato část kódu ověřuje, zdali byla uživatelem zvolena volba regulace polohy nebo podprogram “Krokování”. Pokud byla zvolena jedna z voleb bude se dál vykonávat program, jinak dojde k odskoku pomocí instrukce JMP 3 na konec programu a všechny instrukce budou vynechány.

```
Network 1    NÁZEV NETWORKU (jeden řádek)

LD     Program2:V0.5               // je zapnut program poloha
O      Krokovani:V2.2              // nebo krokovani
=      On_PSD:M1.1                 // bit M1.1 je v log. 1

LDN    On_PSD:M1.1                 //Pokud není zvolena regulace polohy,
JMP    3                           // odskok na LBL 3 (konec podprogramu
```

Obr. 6.45 Rozhodování, zdali se bude pokračovat ve vykonávání programu.

Důležitou části kódu tohoto podprogramu je umístěn ve druhém a třetím networku (obrázek 6.46). Nejprve se provede přesun aktuální hodnoty vysokofrekvenčního čítače do proměnné “Pulsy:VD1082”, která bude obsahovat aktuální stav napočítaných pulsů vf. čítače. Proměnnou pulsy následně vložíme do lokální proměnné “#AktHodCit:LD20”, od které odečteme stav čítače v předchozím cyklu. Tímto máme v proměnné LD20 aktuální hodnotu pulsů za 15 ms. Posledním krokem je vložení aktuálního stavu čítače do proměnné VD1150, která představuje předchozí stav čítače v dalším cyklu. Dalším krokem je převod pulsů polohu natočení hřídele motoru. Tento převod provádí třetí network. Prvním krokem je převod aktuální hodnoty čítače na reálné číslo, kterou následně podělíme počtem pulsů na jedno otočení hřídele (1024). Posledním krokem této části programu je přičítání LD28 do proměnné “Pocet_skut_ot:VD1050”, která tak obsahuje počet otočení hřídele o 360°. Tímto jsme vypočítali potřebný parametr pro regulátor polohy, který představuje skutečnou veličinu.

```
Network 2
LD      SM0.0                          // log. 1
MOVD    HC4, Pulsy:VD1082              // presun stavu citace do VD1082
MOVD    Pulsy:VD1082, #AktHodCit:LD20 // pulsy přesunem do LD20
-D      PredHodCit:VD1150, #AktHodCit:LD20 // od LD20 odečteme VD1150
MOVD    Pulsy:VD1082, PredHodCit:VD1150 // počet. pulsu vložime do VD1150

Network 3
LD      SM0.0
DTR     #AktHodCit:LD20, #DWdoR:LD28 // Konvertuje hodnotu na reálné číslo
/R      1024.0, #DWdoR:LD28          // vydelime realne cislo 1024 a dostane
+R      #DWdoR:LD28, Pocet_skut_ot:VD1050 // Provadi sumaci otacek
```

Obr. 6.46 Výpočet počtu pulsů za cyklus, které se převádějí na polohu.

Samotný PD regulátor je realizovaný ve čtvrtém networku (obrázek 6.47). Nejdříve provedeme přesun proměnných počet skutečných a požadovaných otočení motoru do lokálních proměnných Ypol a Wpol. Dalším nezbytným krokem je určení předchozí regulační odchylky, kterou provádíme před výpočtem aktuální regulační odchylky. Nejprve provedeme vynulování proměnné “#Epol_1:LD28”, která představuje minulou regulační odchylku. Po vynulování do ní vložíme proměnnou “#Epol:LD32”, která představuje aktuální regulační odchylku. V prvním cyklu bude mít aktuální regulační nulovou hodnotu, protože její výpočet ještě neproběhl. Posledním krokem v části pro určení předchozí reg. odchylky je vynulování aktuální regulační odchylky. Výpočet aktuální reg. odchylky provedeme tak, že od požadované hodnoty Wpol odečteme skutečnou hodnotu počtu otočení Ypol. Výsledek přesuneme do proměnné Epol. Tímto máme vypočítanou aktuální a předchozí reg. odchylku.

```
Network 4
LD      SM0.0
MOVR    Pocet_skut_ot:VD1050, #Ypol:LD0 // přesun VD1050 do Ypol
MOVR    Pocet_poz_ot:VD1054, #Wpol:LD4 // přesun VD1054 do Wpol

// urceni hodnoty e_1 (predchozi regulacni odchylka)

LD      SM0.0                          // log. 1
MOVR    0.0, #Epol_1:LD28              // vynulování předchozí reg. odch.
MOVR    #Epol:LD32, #Epol_1:LD28       // e_1:= e
MOVR    0.0, #Epol:LD32                // e:= 0.0

//urceni hodnoty e=w-y (regulacni odchylka)

LD      SM0.0
-R      #Ypol:LD0, #Wpol:LD4           // x = w-y
MOVR    #Wpol:LD4, #Epol:LD32          // e := x
```

Obr. 6.47-1 Část programu pro výpočet aktuální a předchozí reg. odchylky.

Nejprve provedeme výpočet proporcionální složky PD regulátoru. První krok je vynulování proměnné UPn, která představuje akční zásah P složky. Do UPn vložíme aktuální regulační odchylku Epol, kde ji následně vynásobíme proporcionální konstantou regulátoru. Tímto jsme vypočítaly akční zásah P regulátoru a přejdeme na výpočet akčního zásahu derivační složky. Obdobně jako u P složky provedeme vynulování proměnné UDn, která představuje akční zásah D složky. Do UDn vložíme aktuální regulační odchylku Epol, od které odečteme předchozí regulační odchylku Epol_1. Nyní stačí proměnnou UDn vynásobit derivační konstantou Cd a dostáváme akční zásah D regulátoru. Posledním krokem je provést součet akčních zásahů proporcionální a derivační složky. Provedeme vynulování proměnné Ukn, která představuje akční zásah PD regulátoru v n-tém vzorku. Do proměnné Ukn přesuneme akční zásah UPn regulátoru P a následně k Ukn přičteme akční zásah UDn regulátoru D. Tímto dostáváme akční zásah PD regulátoru, u kterého je třeba omezit jeho maximální hodnotu. Na výstupu akčního zásahu je filtr, který zajistí, že maximální hodnota akčního zásahu PD regulátoru se pohybuje v rozsahu -150 až +150. Jednotkou akčního zásahu je procento, tedy -150% až + 150%. Volba tohoto rozsahu vychází z toho, že 100% odpovídá 50 Hz na výstupu frekvenčního měniče. Měnič je schopný dodávat i větší frekvenci než je 50 Hz. V našem případě máme měnič nastavený na maximální frekvenci 100 Hz = 200%. Z tohoto důvodu jsme zvolili hodnotu 150 %, která je kompromisem mezi maximální a provozní frekvencí, tím zajistíme při povelu natočení požadované polohy zkrácení času potřebný pro přesun. V případě, že bychom nechtěli frekvenci větší než 50 Hz, stačí zmenšit maximální hodnotu frekvence na straně měniče.

```
// výpočet proporcionální složky Upn = Kp * e(n)

LD      SM0.0                           // log. 1
MOVR    0.0, #UPn:LD8                   // vynulujeme akcni zasah P slozky
MOVR    #Epol:LD32, #UPn:LD8            // Upn:=e
*R      Kpp:VD1078, #UPn:LD8            // Upn = Kp*e


// výpočet derivační složky UDn = Cd * [e(n) - e(n-1)]

LD      SM0.0                           // log. 1
MOVR    0.0, #UDn:LD12                  // vynulujeme akční zásah D slozky
MOVR    #Epol:LD32, #UDn:LD12           // do UDn vložíme reg. odchylku
-R      #Epol_1:LD28, #UDn:LD12         // od reg. odch. odečtem před. reg. o
*R      Cd:VD1162, #UDn:LD12            // rozdíl vynásobíme derivační složko

// výpočet akčního zásahu PD regulátoru Ukn = UPn + UDn

LD      SM0.0                           // log. 1
MOVR    0.0, #Ukn:LD16                  // vynulování akcniho zasahu PD reg.
MOVR    #UPn:LD8, #Ukn:LD16             // přesun UPn do Ukn
+R      #UDn:LD12, #Ukn:LD16            // Ukn = UPn + UDn
LDR>    #Ukn:LD16, 150.0                // výstupní filtr
MOVR    150.0, #Ukn:LD16                // Ukn = 150
LDR<    #Ukn:LD16, -150.0               // výstupni filtr
MOVR    -150.0, #Ukn:LD16               // Ukn = -150
```

Obr. 6.47-2 Výpočet akčního zásahu PD regulátoru.

Posledním krokem tohoto přerušení je přesun akčního zásahu PD regulátoru do proměnné “W_rych:VD1178”, která představuje požadovanou hodnotu reg. vnitřní smyčky.

```
Network 5
LD      SM0.0                           // log. 1
MOVR    #Ukn:LD16, W_rych:VD1178        // předáme akční zasah vnitřní smyčce
LBL     3
```

Obr. 6.47-3 Přesun akčního zásahu PD reg. do požadované veličiny vnitřní smyčky reg.

6.2.10 Časované přerušení 0 (INT0)

Časované přerušení 0 má vyšší prioritu než má časované přerušení 1, proto pomocí tohoto přerušení realizujeme vnitřní smyčku regulátoru. Vzorkovací perioda vnitřní smyčky je nastavena na 10 ms. Pro předání vzorkovací periody časovači přerušení 1 slouží speciální proměnná SMB 34. Vnitřní smyčka představuje dopředné řízení, které bylo popsáno už dříve a tak se můžeme věnovat popisu samotného programu.

Prvním krokem jako u vnější smyčky regulátoru musíme určit počet pulsů v aktuální vzorkovací periodě. Tento výpočet provádí první část kódu n prvním networku (obrázek 6.48). Výsledkem je proměnná "AktHodCit:VD1146", ve které je uložen aktuální počet pulsů v dané vzorkovací periodě. Postup výpočtu je stejný jako u vnější smyčky. Dalším krokem je převod pulsů na otáčky za minutu. Jelikož pracujeme se vzorkovací periodou 10 ms a převádíme pulsy na 1 minutu = 60 s, je třeba počet pulsů vynásobit hodnotou 6000. Tímto dostaneme počet pulsů za minutu, kdy jedné otáčce odpovídá 1024 pulsů, a proto tuto hodnotu podělíme počtem pulsů na otáčku. Dostáváme tak ot/min, které jsou uloženy v proměnné VD1028, kterou ještě přesuneme do lokální proměnné LD4. Proměnná VD1028 slouží k zobrazení aktuálních otáček pomocí OP. Posledním krokem je kontrola zdali jsou otáčky větší než nula (mají kladnou hodnotu), pokud tomu tak není je hodnota otáček vynásobena mínus jedničkou. Tento úkon provádíme proto, aby se nám zobrazovali otáčky vždy v kladných hodnotách. Část programu obsahují druhý network (obrázek 6.49) jsme si už popsali v časovém přerušení 1. Provádí rozhodování, zdali se má skočit na konec programu nebo v jeho pokračování.

```
Network 1
LD      SM0.0                           //log. 1
MOVD    HC4, AktHodCit:VD1146           // presun stavu citace do VD1146
MOVD    AktHodCit:VD1146, #Pom2:LD0 // stav citace vlozime do LD0
-D      PredHodCitace:VD1186, AktHodCit:VD1146 // od aktuálního stavu od
MOVD    #Pom2:LD0, PredHodCitace:VD1186 // přesuneme LD0 do předchozího
MOVD    AktHodCit:VD1146, Pocet_pulsu_enkoder:VD1142 // pocet pulsů v to

LD      SM0.0                           // log. 1
DTR     AktHodCit:VD1146, aktualni_ot_snimac:VD1028 // Konvertuje hodnot
*R      6000.0, aktualni_ot_snimac:VD1028 // pocet pulsu za 1 min
/R      1024.0, aktualni_ot_snimac:VD1028 // pocet otoceni za 1 min  (1
MOVR    aktualni_ot_snimac:VD1028, #Akt_ot_neupravene:LD4 // VD1028 přes

LD      SM0.0                           // log.1
LDR<    aktualni_ot_snimac:VD1028, 0.0 // Pokud jsou otáčky zaáporných h
*R      -1.0, aktualni_ot_snimac:VD1028 // vynásobení otáček -1
```

Obr. 6.48 Network provádí výpočet otáček motoru za minutu.

```
Network 2
LD      Program2:V0.5               // je zapnut program poloha
O       Krokovani:V2.2              // nebo krokovani
=       On_PSD:M1.1                 // bit M1.1 je v log. 1

LDN     On_PSD:M1.1                 //Pokud není zvolena regulace polohy,
JMP     2
```

Obr. 6.49 Pokud byl zvolen program řízení polohy, nebo krokování není proveden odskok.

Nyní můžeme věnovat popisu samotného integračního regulátoru, který je realizovaný ve třetím networku (obrázek 6.50). Skutečnou veličinou jsou zde otáčky motoru, které jsou v rozsahu -1500 až +1500 ot/min. Jelikož požadovaná veličina je v rozsahu -150 až +150 %, je třeba skutečnou veličinu podělit hodnotou 10, kdy dojde ke sjednocení pracovního rozsahu skutečné a požadované veličiny. Tento převod provádí první část popisovaného networku.

Dalším krokem je výpočet regulační odchylky I regulátoru. Nejprve provedeme vynulování proměnné "#e_dop:LD12", která představuje regulační odchylku. Do této proměnné vložíme požadovanou hodnotu VD1178, od které následně odečteme skutečnou hodnotu otáček. Tímto jsme vypočítali reg. odchylku. Posledním krokem je výpočet akčního zásahu I regulátoru. Nejprve vložíme minulou akční veličinu do proměnné UIn_1. Následně vypočítáme akční zásah v tomto cyklu, kdy do proměnné UIn vložíme integrační složku regulátoru, kterou následně vynásobíme regulační odchylkou e_dop. K této vypočítané hodnotě přičteme minulou akční veličinu. Tímto jsme vypočítali akční veličinu v aktuálním cyklu, která je uložena v proměnné UIn. Integrační složka má za úkol kompenzovat skluz motoru, který je cca do 5%, proto je v programu výstupní filtr, který propouští hodnotu akčního zásahu I regulátoru v rozsahu -5 až +5 %. Tímto je část programu pro výpočet akčního zásahu I regulátoru za námi a můžeme přejít k popisu posledního networku (obrázek 6.51).

```
Network 3     NÁZEV NETWORKU (jeden řádek)
// Převedení otáček na procenta pro regulátor rychlosti

LD     SM0.0
MOVR   #Akt_ot_neupravene:LD4, #y_dop:LD8 // otáčky v rozmezí -1500 až 150
/R     10.0, #y_dop:LD8            // otáčky převedeny na -150 až +150 %

// Výpočet regulační odchylky  e_dop = w_dop - y_dop

LD     SM0.0                       // log.1
MOVR   0.0, #e_dop:LD12            // vynulujeme e_rych
MOVR   W_rych:VD1178, #e_dop:LD12  // do e_rych vložíme požadovanou hodnot
-R     #y_dop:LD8, #e_dop:LD12     // e_dop = w_dop - y_dop


// výpočet integrační složky UIn = UIn_1 + [Ci * e(n)]

LD     SM0.0
MOVR   #UIn:LD30, #UIn_1:LD34      //minulou integrační složku vložime do
MOVR   Ci:VD1158, #UIn:LD30        //Ci do UIn
*R     #e_dop:LD12, #UIn:LD30      // Ci * e(n)
+R     #UIn_1:LD34, #UIn:LD30      // [UIn = UIn_1 + [Ci * e(n)]
LDR>   #UIn:LD30, 5.0              // filtr I složky +
MOVR   5.0, #UIn:LD30              // ...
LDR<   #UIn:LD30, -5.0             // filtr I složky -
MOVR   -5.0, #UIn:LD30             // ...
```

Obr. 6.50 Část programu realizující výpočet akčního zásahu I regulátoru.

Vnitřní smyčka je realizovaná jako dopředné řízení, proto je třeba provést sumaci požadované veličiny a akčního zásahu integračního regulátoru. Nejprve tedy vložíme požadovanou hodnotu do proměnné "#Suma:LD38", ke které následně přičteme akční veličinu I reg. Výslednou sumu přesuneme do proměnné VD1000, která představuje parametr instrukce USS_DRV_CTRL, pro požadavek na výstupní frekvenci měniče. Poslední řádek představuje instrukci pro odskok od druhého networku. Tímto jsme si popsali jeden cyklus časovaného přerušení, které se opakuje každých 10 ms.

```
Network 4
LD     SM0.0                       // log. 1
MOVR   W_rych:VD1178, #Suma:LD38   // přesune požadovanou rychlost
+R     #UIn:LD30, #Suma:LD38       // k Sume přičte akční zásah I
MOVR   #Suma:LD38, Poz_ot_motoru:VD1000 // Sumu přesune do požadov.

LBL    2                           // skok od JMP 2
```

Obr. 6.51 Provedení sumace požadované veličiny a akčního zásahu I regulátoru.

6.2.11 Podprogram krokování

Tento podprogram je doplňkem programu pro řízení polohy, kdy umožňuje otáčet motor po volitelných krocích. Uživatel má na výběr mezi čtyřmi možnostmi, které se od sebe odlišují velikostí kroku. Velikosti kroku je následující: čtvrt, půl, třičtvrtě otáčky nebo jedna celá otáčka. Uživatel si tedy nejprve vybere požadovaný krok, pak zapne pohon a pomocí tlačítek přidat/ubrat krok provádí krokování motoru. Tímto jsme si obecně popsali funkci tohoto podprogramu a můžeme přejít na samotný popis kódu. Prvním krokem je ošetření stavu, kdy uživatel nezvolí žádný krok, tak se automaticky provede volba kroku "Krok_1:V2.6". Toto ošetření provádí první část programu v prvním networku (obrázek 6.52). Následně je provedeno přiřazení hodnoty kroku do proměnné "#Krok_l:LD0" podle zvoleného kroku. Tímto je ve stručnosti první network popsán a přejdeme na druhý (obrázek 6.53).

```
Network 1
LDN     Krok_025:V2.3               // pokud není zvolena žádný krok,
AN      Krok_05:V2.4
AN      Krok_075:V2.5
AN      Krok_1:V2.6
S       Krok_1:V2.6, 1

LD      Krok_025:V2.3
MOVR    0.25, #Krok_l:LD0           // 1 krok = 0,25 ot
LD      Krok_05:V2.4
MOVR    0.5, #Krok_l:LD0            // 1 krok = 0,5 ot
LD      Krok_075:V2.5
MOVR    0.75, #Krok_l:LD0           // 1 krok = 0,75 ot
LD      Krok_1:V2.6
MOVR    1.0, #Krok_l:LD0            // 1 krok = 1 ot
```

Obr. 6.52 Část programu pro volbu kroku.

Tato část kódu je obdoba programu ruční řízení motoru. Kde jsme měli dvě tlačítka, kdy jedním jsme zvyšovali a druhým snižovali frekvenci. V tomto programu přidáváme, nebo ubíráme krok. Kdy po stisku tlačítka "Ubrat_ot:V1.3" se provede odečtení hodnoty zvoleného kroku, od požadovaných otáček VD1054. Tímto nastane regulační odchylka a regulátor provede akční zásah, aby tuto odchylku odstranil. Obdobně je to u tlačítka "Pridat_ot:V1.2", které zvětší požadovanou hodnotu otáček o daný krok a regulátor opětovně provede akční zásah pro odstranění regulační odchylky.

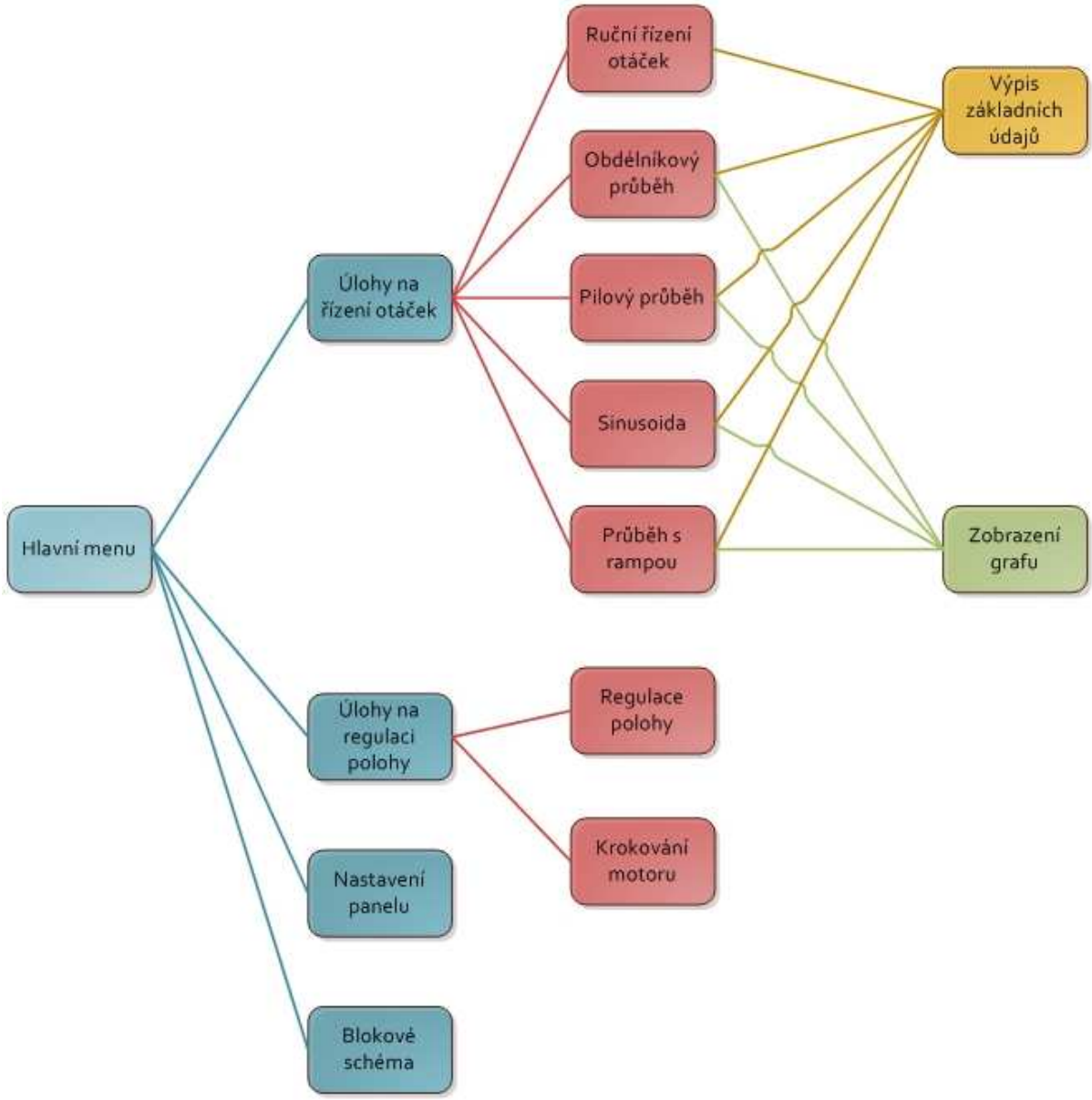
```
Network 2
LD      Ubrat_ot:V1.3
EU
-R      #Krok_l:LD0, Pocet_poz_ot:VD1054
EU
R       Ubrat_ot:V1.3, 1
```

Obr. 6.53 Řízení tlačítka pro přidání kroku

Tímto jsme u konce s popisem programů vytvořených v programovém prostředí STEP 7-Micro/WIN, pro programovatelný automat SIMATIC S7-224XP. Pomocí automatu realizujeme různé programy pro řízení otáček a regulaci polohy asynchronního motoru pomocí frekvenčního měniče G110. Jako vstupní výstupní periferie slouží operátorský panel TP 177 micro. Pomocí tohoto panelu jsme vytvořili uživatelské prostředí, které nám umožňuje číst, zapisovat proměnné, přepínat mezi jednotlivými programy a neposledně vykreslovat stav otáček motoru pomocí grafu. Tímto opouštíme tuto kapitoly a přejdeme na další věnující se jednotlivým obrazovkám programů zobrazovaných pomocí OP.

7 Uživatelské prostředí realizované pomocí OP

7.1 Struktura menu uživatelského prostředí



Obr. 7.0 Zobrazuje strukturu menu uživatelského prostředí.

Na obrázku 7.0 je zobrazená celá struktura obrazovek, mezi kterými může uživatel procházet. Jednotlivé úrovně vnoření jsou barevně rozlišený. Po zapnutí panelu a spuštění programu nás na obrazovce uvítá první okno programu, jedná se o “Hlavní menu”. Toto okno má nulovou úroveň vnoření a je vybarvenou světle modrou barvou. Z tohoto okna se můžeme dostat do další úrovně vnoření, která představuje čtyři obrazovky. Tyto obrazovky mají první úroveň vnoření a jsou vybarveny tmavší modrou barvou než u prvního okna. Okna “Nastavení panelu” a “Blokové schéma” neumožňují dalšího vnoření, jsou to tedy cílové obrazovky. Obrazovka “Úlohy na řízení otáček” umožňuje dalšího větvení o pět obrazovek a tedy dalšího vnoření. Tato úroveň je zvýrazněna červenou barvou. Tímto se dostáváme k jednotlivým programům pro řízení motoru, kde názvy obrazovek vystihují činnost jednotlivých programů.

Ze všech obrazovek programů se můžeme dostat do další úrovně vnoření. Při volbě výpisu parametrů, se nám zobrazí obrazovka “Výpis základních údajů”. Tato obrazovka je zvýrazněna žlutou barvou. Další možností vnoření je při volbě zobrazení grafu, který vykresluje průběh otáček motoru řízené daným programem. Toto vnoření můžeme provést ze všech programů, kromě programu “Ruční řízení otáček. Toto okno má označení “Zobrazení grafu”, které je zvýrazněno zelenou barvou. Okna pro zobrazení grafu a výpisu parametrů představují největší vnoření. Tímto se můžeme vrátit k oknu “Úlohy na regulaci polohy”, která umožňuje rozvětvení na dvě obrazovky “Regulace polohy” a “Krokování motoru”, které se dále nevětví. Tyto dvě okna jsou v úrovni vnoření, která je zvýrazněna červenou barvou.

7.2 Popis a ukázky vybraných obrazovek uživatelského prostředí

V této kapitole se budeme věnovat popisu vybraných obrazovek uživatelského prostředí. Nebudou zde popsány všechny, ale budou vybrány ty nedůležitější. Všechny obrazovky budou umístěny v příloze.

7.2.1 Hlavní menu

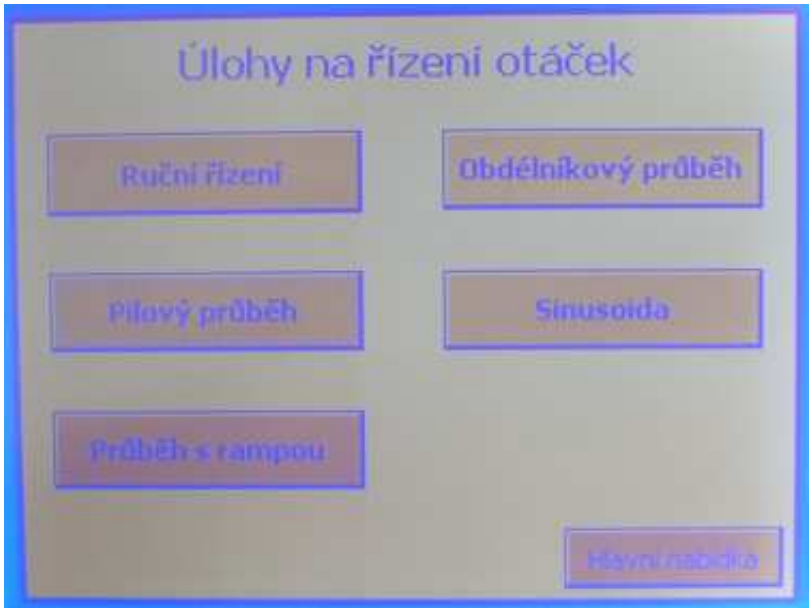
První popisovaná obrazovka (obrázek 7.1) představuje úvodní obrazovku uživatelského prostředí, která představuje hlavní menu. Toto okno obsahuje čtyři tlačítka pomocí kterých provádíme volbu další nabídky. Nejdůležitější jsou zde tlačítka “Řízení otáček” a “Řízení polohy”, které mají i větší rozměry než zbylé dvě tlačítka. Pomocí tlačítka „Řízení otáček“ se dostaneme na obrazovku “Úlohy na řízení otáček”, obdobně po stisku tlačítka „Řízení polohy“ se přepneme do obrazovky “Úlohy na regulaci polohy”. Pomocí tlačítka “Blokové schéma” zobrazíme blokové schéma, které slouží pro představu jak je celá soustava zapojena. Poslední tlačítko slouží pro volbu obrazovky “Nastavení panelu”, ve které můžeme provádět změnu kontrastu OP. Poslední doplňkovou informací v tomto okně je název fakulty, školy a jméno autora.



Obr. 7.1 Úvodní obrazovka uživatelského prostředí.

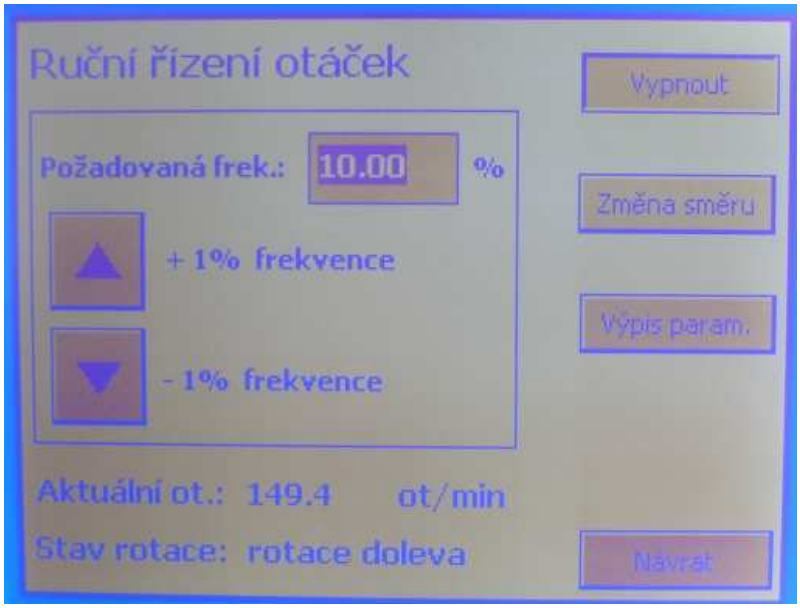
7.2.2 Úlohy na řízení otáček motoru

Po stisku tlačítka řízení otáček se objeví obrazovka nabízející volbu řízení otáček motoru (obrázek 7.2). Máme na výběr řízení motoru ručně, generování obdélníkových, pilových průběhů pomocí sinusoidy, nebo průběhu ve tvary rampy, které se přenášejí na otáčky motoru.



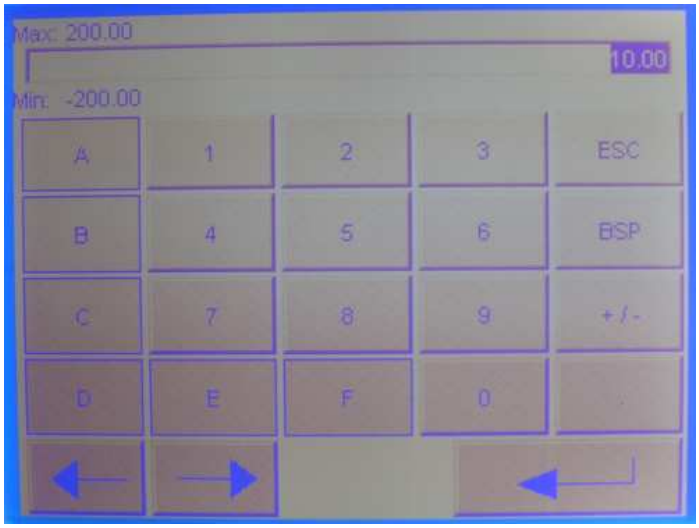
Obr. 7.2 Obrazovka zobrazující nabídku možností řízení otáček motoru.

7.2.3 Ruční řízení otáček motoru



Obr. 7.4 Obrazovka ručního řízení otáček motoru.

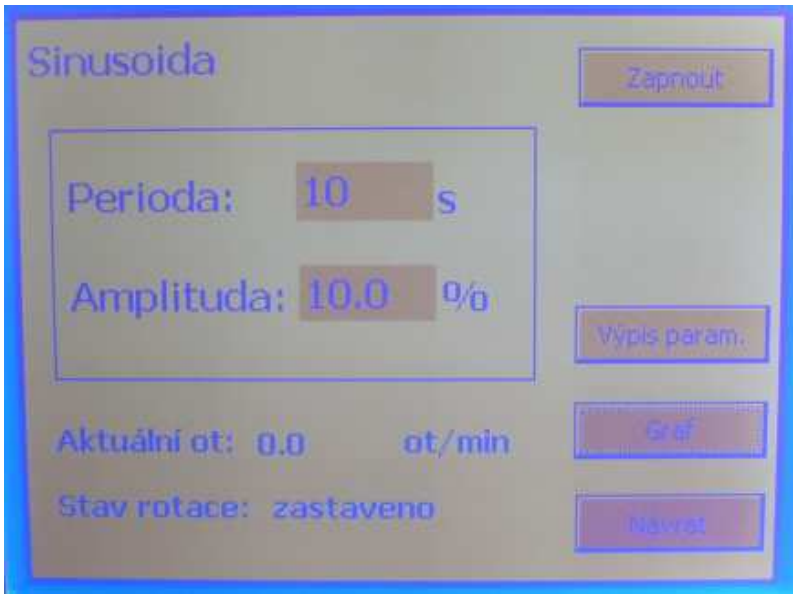
Při stisknutí tlačítka ruční řízení se přepneme do okna ruční řízení otáček, které je zobrazeno na obrázku 7.4. Na pravé straně obrazovky jsou pod sebou umístěny čtyři tlačítka. Tlačítko s nápisem “Vypnout” slouží pro zapnutí nebo vypnutí pohonu, nápis na tomto tlačítku se mění podle stavu motoru (spuštěn/vypnut). Toto tlačítko ve všech programech plní stejnou funkci. Dalším tlačítkem s nápisem “Změna směru” slouží, jak už název napovídá pro změnu směru rotace motoru. Tlačítko “Výpis parametrů” slouží pro přepnutí na obrazovku “Výpis základních údajů”, která podává informace o aktuálních otáčkách motoru, výstupním napětí a proudu frekvenčního měniče a teplotě motoru. Posledním tlačítkem ze zmiňované čtveřice slouží pro návrat do předcházející obrazovky. Velkou část obrazovky zabírá rámeček, ve kterém jsou umístěné dvě tlačítka a rámeček pro zadávání požadované výstupní frekvence měniče. Po kliknutí na rámeček s hodnotou požadované frekvence se přepneme do editačního okna, které je zobrazeno na obrázku 7.3. Po zadání požadované hodnoty pomocí numerické klávesnice a potvrzení této volby pomocí enteru, se vrátíme zpět na obrazovku ručního řízení otáček motoru. Změnu frekvence můžeme taky provádět pomocí dvou tlačítek, kdy jejich grafické symboly označují, zdali budeme přidávat nebo odebírat. Jedno stisknutí znamená přidání nebo ubrání jednoho procentního bodu požadované frekvence. Spodní část obrazovky je věnována informacím o stavu rotace motoru a jeho aktuálních otáčkách za minutu. Tímto máme obrazovku zobrazující možnosti ručního řízení za popsanou a můžeme přejít na další.



Obr. 7.3 Okno pro zadávání požadované hodnoty.

7.2.4 Řízení motoru pomocí sinusoidy

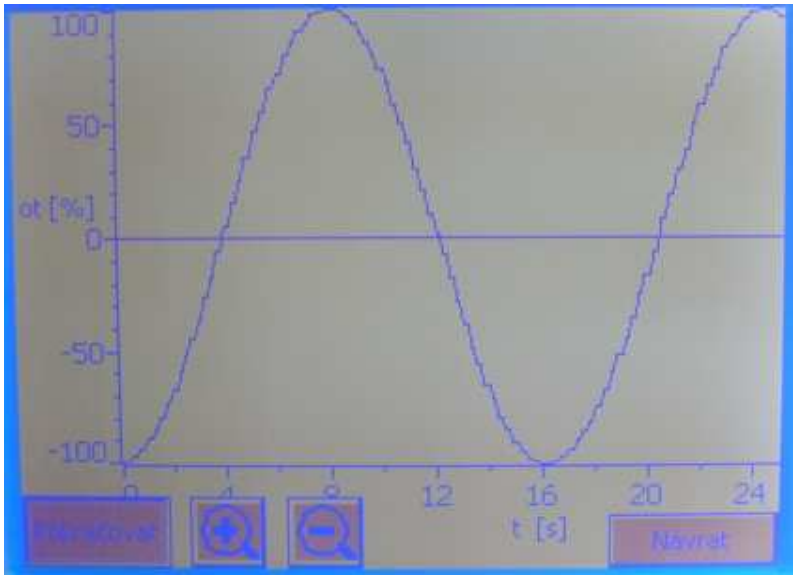
Při volbě programu řízení motoru na základě generované sinusoidy se nám spustí obrazovka sinusoida (obrázek 7.5). Na levé straně máme opětovně čtveřici tlačítek, které jsou stejné až na jedno jako u obrazovky ručního řízení otáček. V tomto případě zde není tlačítko pro změnu směru, které je nahrazeno tlačítkem pro zobrazení grafu generované funkce. Velkou část obrazovky zabírá obdélník, který sdružuje dva parametry, kterými ovlivňujeme tvar sinusoidy. Jedná se tedy o parametr perioda a amplituda sinusoidy. Po kliknutí na šedý rámeček reprezentující daný parametr se opětovně přepneme do editačního okna (obrázek 6.4), jako v předešlém případě. Po nastavení a potvrzení požadované hodnoty se opětovně vrátíme na obrazovku sinusoidy. Po zadání obou parametrů můžeme provést spuštění motoru pomocí tlačítka “Zapnou”. Na spodní části obrazovky máme opětovně zobrazen stav rotace a otáček motoru (ot/min).



Obr. 7.5 Obrazovka pro ovládání programu sinusoida.

7.2.5 Graf generované sinusoidy

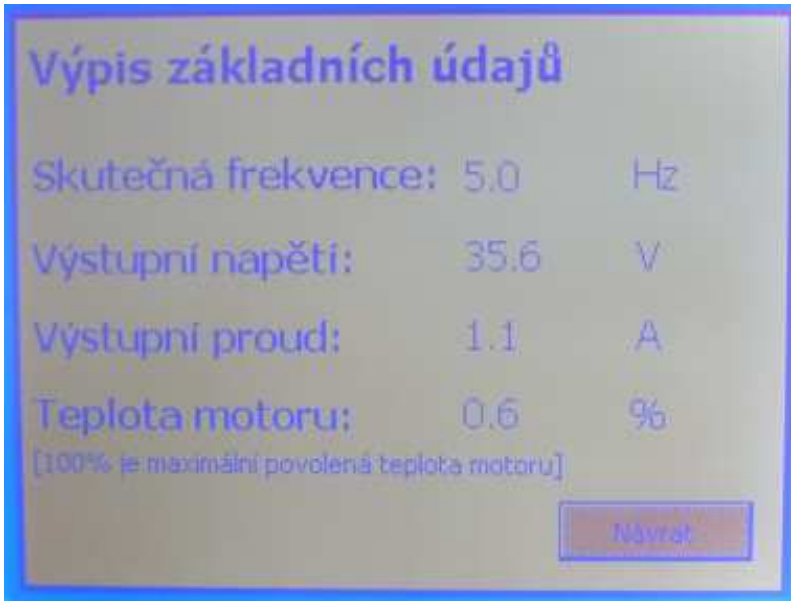
Při kliknutí na tlačítko “Graf” se přepneme na novou obrazovku, která slouží pro vykreslování průběhu otáček motoru ve zvoleném časovém úseku. V případě, že řídíme motor pomocí sinusoidy a při zapnutém motoru klikneme na tlačítko graf, zobrazí se nám obrazovka na obrázku 7.6. Téměř celou obrazovku zabírá samostatný graf jenom na spodní části najdeme čtyři tlačítka. V dolním pravém rohu je umístěno tlačítko pro návrat předchozí obrazovky. V dolním levém rohu je umístěno tlačítko pro zastavený vykreslování nových hodnot otáček. Po opětovném stisknutí se pokračuje ve vykreslování aktuální hodnoty otáček motoru. Poslední dvě tlačítka slouží pro úpravu časové osy (osa x), tím si můžeme provést detail vykreslovaného průběhu nebo naopak zobrazit delší časový úsek sledované veličiny. Pro větší detail (zmenšení časové osy) slouží tlačítko s plusem a pro zmenšení detailu tlačítko s mínusem uprostřed lupy.



Obr. 7.6 Graf generované sinusoidy programu Sinusoida.

7.2.6 Výpis základních údajů

Toto je poslední obrazovka, kterou si stručně popíšeme. Po stisknutí tlačítka pro výpis parametrů se přepneme do následující obrazovky (obrázek 7.7). Zobrazované údaje jsou zobrazovány s přesnosti na jedno desetinné místo. Všechny zobrazované veličiny jsou zobrazeny v základních jednotkách až na teplotu motoru, která je udávána v procentech. Teplotu motoru frekvenční měnič vyhodnocuje nepřímou metodou, kdy 100% odpovídá maximální povolené teplotě motoru. Po překročení této hodnoty, bude měnič snižovat proud procházejícím motorem, nebo v kritickém stavu dojde k odpojení motoru ze sítě.



Obr. 7.7 Obrazovka pro výpis základních údajů.

Tímto bych s popisem obrazovek uživatelského prostředí skončil, nepopsané obrazovky budou umístěny na konci této práce v příslušné příloze. Princip obsluhy ostatních obrazovek je obdobný jako u předešlých obrazovek.

8 Závěr a shrnutí poznatků z uvádění do provozu

Tato diplomová práce se zabývá řízením otáček asynchronního třífázového motoru moderními prostředky. Změnu otáček u asynchronního motoru lze provést několika způsoby. Můžeme měnit otáčky pomocí změny skluzu motoru, který se například provádí zařazením odporu do obvodu rotoru. Dalším způsobem je změnou počtů pólů, kdy dosáhneme skokové změny otáček motoru. Posledním způsobem je řízení motoru pomocí změny vstupního kmitočtu. Nejmodernějším ale také i nejsložitějším způsobem je řízení otáček asynchronního motoru pomocí změny frekvence napájecího napětí. Zařízení, které provádí změnu frekvence označujeme jako frekvenční měnič, který se považuje za moderní prostředek pro řízení otáček asynchronního třífázového motoru. V této diplomové práci jsme využívali model Sinamics G110, který řídíme pomocí programovatelného automatu SIMATICS S7-224XP. Obsluha automatu je realizována prostřednictvím operátorského panelu TP 177 micro. Tento hardware dodává firma Siemens jako set, spolu s motorem. Komunikace mezi automatem a frekvenčním měničem je realizovaná pomocí USS protokolu po sběrnici RS-485. Tato knihovna umožňuje jednoduše řídit frekvenční měnič pomocí programovatelného automatu po sběrnici RS-485. Tato instrukce zjednodušuje programování, kdy se nemusíme starat o samotnou komunikaci mezi automatem a měničem.

Hlavním cílem této práce bylo vytvoření ukázkovou laboratorní úlohu pro řízení otáček asynchronního motoru, která by demonstrovala možnosti, které nám nabízí používaný hardware. Laboratorní úloha umožňuje obsluze řídit otáčky motoru manuálně, automatickým řízením podle určitého časového schématu (skokové změny, pilový průběh, sinusoida) a neposledně umožňuje regulaci polohy.

Při řešení diplomové práce jsem narazil na několik problémů. Tyto problémy se týkali především USS knihovny. První problém spočívá v instrukci pro čtení parametrů z frekvenčního měniče. Pokud spustíme tuto instrukci a současně máme spuštěný pohon, který má směr rotace indikující na měniči v záporných hodnotách, tak dochází k neustálému přepínání směru rotace motoru. Tímto je tedy nutné před zavoláním instrukce pro čtení změnit stav požadovaných otáček na kladnou hodnotu. Tato změna způsobí, že motor při aktivaci čtení parametrů přepne směr rotace. Změna rotace se provede, jen tehdy pokud jsou indikované otáčky motoru v záporných hodnotách před spuštěním instrukce pro čtení. Dalším problémem u instrukce pro čtení je náhodný přepis přečteného parametru. Tato chyba se projevila, tak že zobrazované parametry na displeji OP byly náhodně přepisovány nulovou hodnotou. Tento problém jsem odstranil tak, že jsem proměnou ve které byl uložen přečtený parametr porovnával jestli je přečtená hodnota různá od nuly. Pokud byla hodnota nenulová uložila se do jiné proměnné, která je zobrazována na displeji OP. Tímto se chybová hodnota ignoruje a tak se nedostane na výstup. Posledním problémem s USS knihovnou spočívá v počátečním nastavení směru rotace motoru při zapnutí. V hlavní řídící knihovně je bit pro nastavení směru rotace, který je při spuštění vždy nastaven na log. 0. Při tomto nastavení by měl být indikovaný směr rotace při každém spuštění stejný ale ve skutečnosti má jednou hodnota nastavených otáček kladnou hodnotu a jindy zase zápornou. Proto je třeba při každém spuštění provést synchronizaci, aby klaný směr otáček odpovídal log. 0 a záporný směr log. 1 řídícího bitu. To je především důležité u regulace polohy, kdy při zadání požadovaných otáček se motor otáčel správným směrem, aby nedocházelo ke zvětšování regulační odchylky namísto zmenšování. Až na uvedené drobnosti pracuje USS knihovna spolehlivě. Laboratorní úloha je plně funkční a je připravená pro výuku.

SEZNAM POUŽITÉ LITERATURY

[1] ŠVARC, Ivan; ŠEDA, Miloš; VÍTEČKOVÁ, Miluše. Automatické řízení 1. vydání. Akademické nakladatelství CERM, 2007. 324 s. ISBN 978-80-214-3491-2.

[2] *Programovatelný automat S7-200, Systémový manuál*. červen 2004. 534 s.

[3] *Seznam parametrů, SINAMICS G110*. [s.l.] : [s.n.], duben 2003. 84 s.

[4] *Návod k obsluze - stručný : SINAMICS G 110*. listopad 2004. 28 s.

[5] *SIMATIC HMI device OP 73micro, TP 177micro (WinCC flexible)*. leden 2005. 186 s.

[6] *Using USS4 Protocol Instructions to Communicate with Drives*. červenec 2001. 25 s.

[7] FALC, Petr. *Www.odbornecasopisy.cz* [online]. březen 2005 [cit. 2010-05-27]. Rotační inkrementální snímače polohy. Dostupné z WWW: <http://www.odbornecasopisy.cz/index.php?id_document=30393>.

[8] *Www.siemens.cz* [online]. 2010 [cit. 2010-05-27]. Industry automation & Drive Technologies. Dostupné z WWW: <http://www.siemens.cz/siemjet/cz/home/ad/Main/index.jet>

[9] TIŠNOVSKÝ, Pavel. *Http://www.root.cz* [online]. 18.12.2008 [cit. 2010-05-27]. Sběrnice RS-422, RS423 a RS 485. Dostupné z WWW: <http://www.root.cz/clanky/sbernice-rs-422-rs-423-a-rs-485/>.

[10] MÖHWALD, Jiří. *UKÁZKA ŘÍZENÍ OTÁČEK MOTORU FREKVENČNÍM MĚNIČEM*. Brno, 2009. 92 s. Diplomová práce. VUT v Brně,ÚAI FSI.

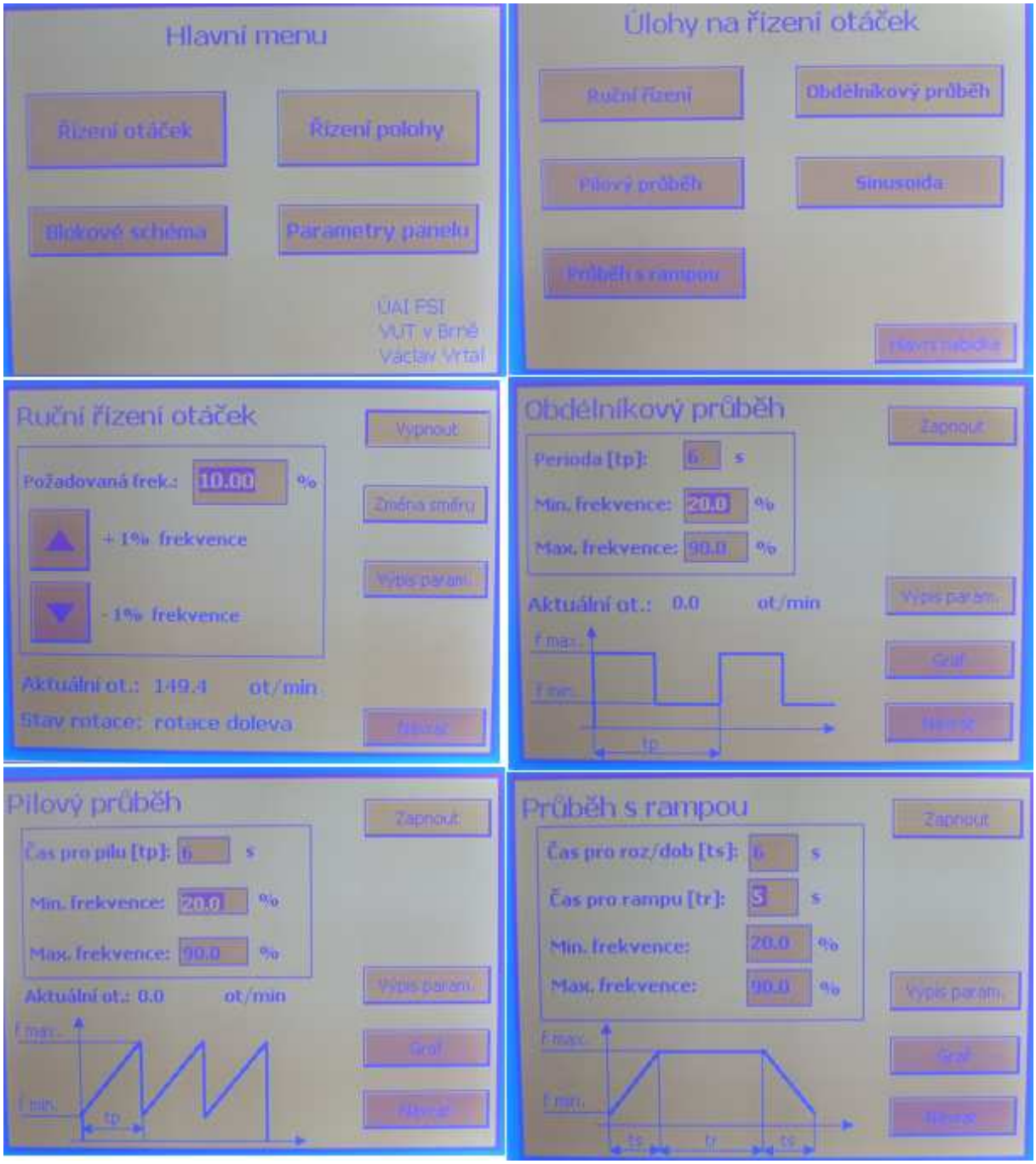
SEZNAM PŘÍLOH

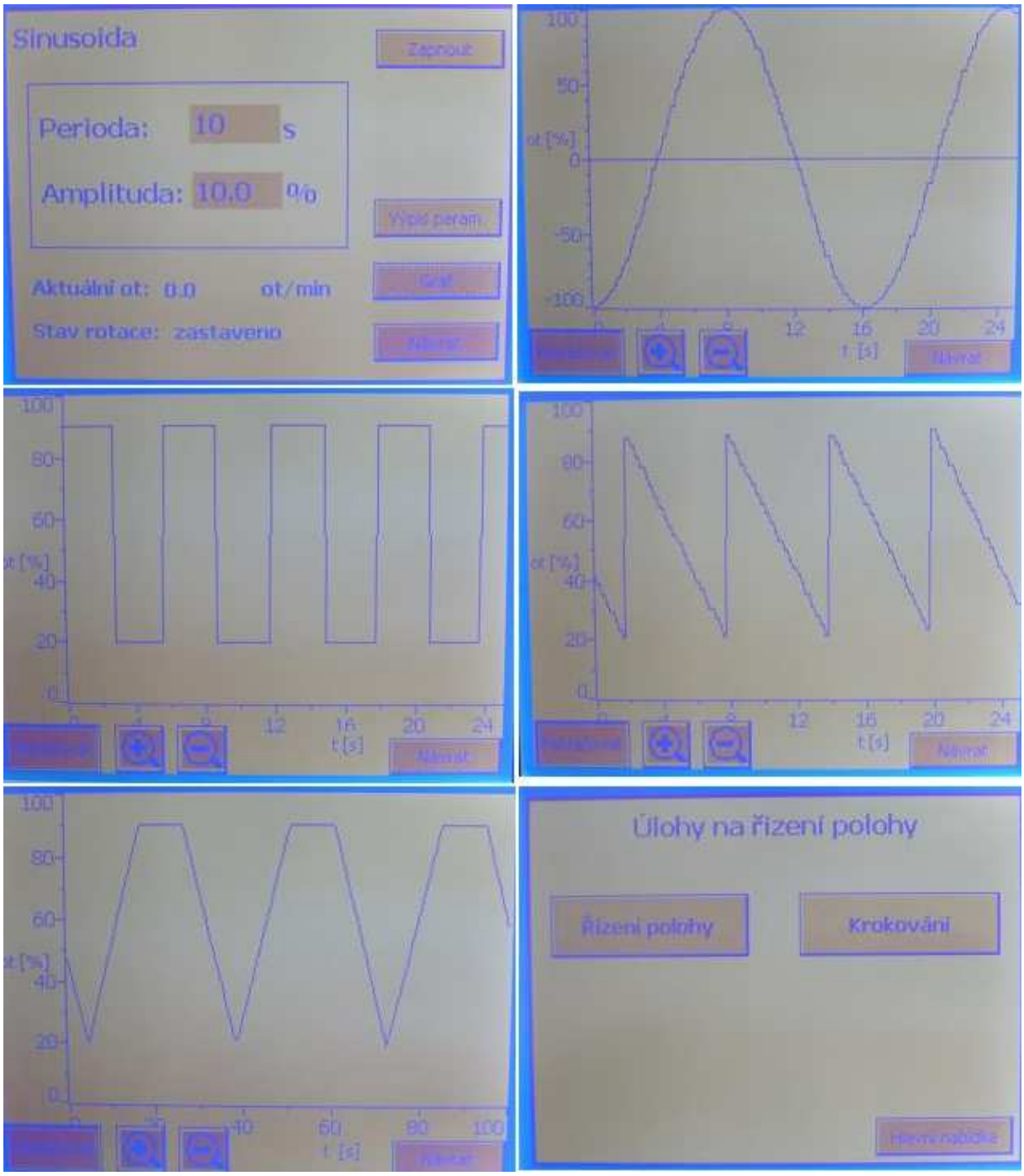
Příloha č. 1 – Obrazovky grafického prostředí operátorského panelu

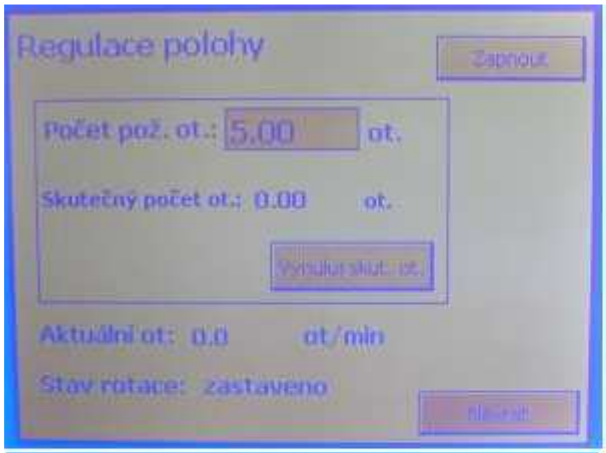
Příloha č. 2 – CD, které obsahuje:

- Elektronickou podobu diplomové práce
- Přílohy:
  - Program v prostředí STEP –Micro/WIN V4.0 pro PLC S7-224XP
  - Program v prostředí WinCC flexible 2008 Micro pro operátorský panel TP 177 micro.

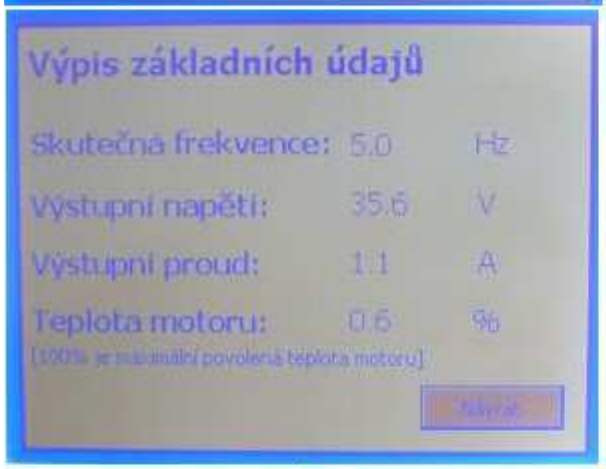
Příloha č. 1 – Obrazovky grafického prostředí pro OP

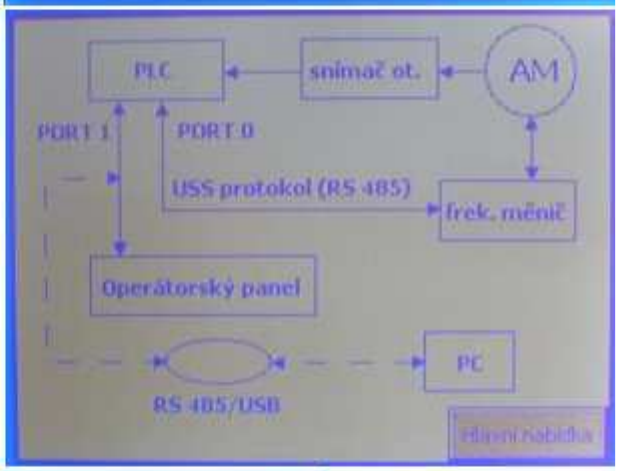



Sinusoida
Zapnout
Perioda: 10 s
Amplituda: 10.0 %
Výpis param.
Aktuální ot: 0.0 ot/min
Stav rotace: zastaveno
Návrat
ot [%]
t [s]
Úlohy na řízení polohy
Řízení polohy
Krokování


Regulace polohy
Zapnout
Počet pož. ot.: 5.00 ot.
Skutečný počet ot.: 0.00 ot.
Vynulovat skut. ot.
Aktuální ot: 0.0 ot/min
Stav rotace: zastaveno
Hlavní nabídka


Krokování motoru
Velikost kroku:
0.25 ot.
0.5 ot.
0.75 ot.
ON 1.0 ot.
+ krok
- krok
Počet požad. ot: 5.000
Skutečný počet ot: 0.000
Zapnout
Hlavní nabídka


Výpis základních údajů
Skutečná frekvence: 5.0 Hz
Výstupní napětí: 35.6 V
Výstupní proud: 1.1 A
Teplota motoru: 0.6 %
(100% je maximální povolená teplota motoru)
Hlavní nabídka


PLC
snímač ot.
AM
PORT 1
PORT 0
USS protokol (RS 485)
frek. měnič
Operátorský panel
PC
RS 485/USB
Hlavní nabídka


Nastavení panelu
Kontrast:
Hlavní nabídka