TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

HMI APLIKACE K PŘÍMÉMU OVLÁDÁNÍ FUNKCÍ ELEKTRICKÉHO SERVOPOHONU

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy
Autor práce: **Filip Gaudel**
Vedoucí práce: Ing. David Lindr, Ph.D.

Liberec 2015

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

HMI APPLICATION FOR DIRECT CONTROLLING OF THE ELECTRICAL DRIVE

Bachelor thesis

Study programme: B2612 – Electrical Engineering and Informatics
Study branch: 2612R011 – Electronic Information and Control Systems
Author: **Filip Gaudel**
Supervisor: Ing. David Lindr, Ph.D.

Liberec 2015

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

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Filip Gaudel**
Osobní číslo:	**M11000152**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**HMI aplikace k přímému ovládání funkcí elektrického servopohonu**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s principem programování elektrických servopohonů Sinamics S120 a ovládacích panelů firmy Siemens.
2. Otestujte synchronizační funkce na reálných servopohonech laboratorního standu a vytvořte k nim HMI aplikaci na dotykovém ovládacím panelu.
3. Vypracujte závěrečnou textovou dokumentaci, zhodnoťte výsledky vaší práce a do elektronických příloh uveďte mimo jiné i zdrojové kódy vytvořených programů.

Rozsah grafických prací:	**dle potřeby dokumentace**
Rozsah pracovní zprávy:	**30–40 stran**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **Rydlo P.: Řízení elektrických střídavých pohonů. Skriptum, FM TUL, 2007, ISBN 978-80-7372-223-4.**

[2] **Souček,P.: Servomechanismy ve výrobních strojích, Vydavatelství ČVUT Praha, 2004.**

Vedoucí bakalářské práce:	**Ing. David Lindr, Ph.D.** Ústav mechatroniky a technické informatiky
Konzultant bakalářské práce:	**Ing. Martin Diblík, Ph.D.** Ústav mechatroniky a technické informatiky
Datum zadání bakalářské práce:	**10. října 2014**
Termín odevzdání bakalářské práce:	**15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2014

Prohlášení

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

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

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

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

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

Datum: 15.5.2015

Podpis: 

Poděkování

Touto cestou bych rád poděkoval těm, kteří svou pomocí přispěli ke vzniku této práce, především za vedení práce Ing. Davidu Lindrovi, Ph.D. Také bych chtěl poděkovat celé své rodině za jejich podporu při studiu.

Abstrakt

HMI aplikace k přímému ovládání funkcí elektrického servopohonu

Tato bakalářská práce popisuje jak pracovat s frekvenčním měničem Sinamics S120 a dvěma synchronními motory, které jsou ovládány pomocí zobrazovacího zařízení OP 177B a vizualizačního systému WinCC flexible. Cílem této práce je otestovat jednotlivé funkce servopohonu a vytvořit aplikaci pro dotykový panel, kterým bude možné tyto funkce nastavovat a ovládat v reálném čase. Dále pak otestovat funkce synchronizační knihovny pro realizaci elektronických vazeb mezi více elektrickými pohony Siemens Sinamics. V první části se práce zabývá obecnou teorií a přehledem pohonů značky Simenes. Ve druhé části je popsáno jak správně naparametrovat měnič a naprogramovat vizualizaci.

Klíčová slova

SINAMICS S120, HMI, STARTER, WinCC flexible, OP177B, DCC chart

Abstract

HMI application for direct controlling of the electrical drive

This thesis describes how to work with a frequency converter Sinamics S120 and two synchronous motors, which are controlled by the OP 177B display device and how to work with visualization system WinCC flexible. The aim of this work is to test individual drive functions and create an application for a touch panel which will allow using these functions set and controlled in real time. Then test the synchronization function library for implementing electronic links between multiple electric drives Siemens Sinamics. The first part deals with a general overview of the theory and drives brand Siemens. The second part describes how to properly frequency converter parameterised and how to programmed visualization.

Key words

SINAMICS S120, HMI, STARTER, WinCC flexible, OP177B, DCC chart

Obsah

Seznam obrázků

Předmluva

Tématem bakalářské práce je otestování jednotlivých funkcí servopohonu a vytvoření aplikace pro dotykový panel, kterým bude možné tyto funkce ovládat.

Předmětem teoretické části je seznámení se základními funkcemi měničů a motorů a také přehled nabízených měničů a motorů firmou Siemens a jejich optimální využití v konkrétních aplikacích.

Praktická část se věnuje nastavení pohonu a jeho vizualizaci. Je zde popsán návrh pomocí softwaru a jak se nastavují jednotlivé funkce měniče pro správný chod motoru. Dále je pak vytvořena vizualizační aplikace, kterou je daný motor ovládán.

Motivace

Motivací pro vypracování je seznámení se s principem programování elektrických servopohonů, jelikož jsou součástí většiny automatizovaných aplikací, kde například pohybují součástmi stroje nebo slouží k dopravě materiálu či výrobku. Elektrické pohony s měniči jsou však i v jednodušších aplikacích, jako jsou třeba ventilátory, kde je potřeba regulovat asynchronní motor.

Cíle

Cílem mé práce je navrhnout vizualizační software pro přímé ovládání měniče S120 bez použití PLC, a ukázat některé funkce měniče. Dále pak otestovat funkce synchronizační knihovny a realizovat elektronickou vazbu mezi více elektrickými pohony Siemens.

1 Měniče frekvence

S elektrickými pohony se setkáváme každý den a to v mnoha podobách, například u výtahů, lanovek, jeřábů, výrobních strojů a mnoha dalších. Elektrický pohon je označení pro soubor všech technických prostředků, které zajišťují pohon strojního mechanismu za pomocí elektrické energie, zpravidla za pomocí nějakého elektromotoru, který pak obvykle tvoří základní část elektrického pohonu.

Aplikace většinou vyžadují, aby mohl být pohon regulován. Jenže nejjednodušší asynchronní motory se bez použití frekvenčního měniče obtížně regulují. Dříve se proto v pohonech používaly stejnosměrné stroje s cizím buzením. Moderní výkonové polovodičové součástky ale umožnily, aby konstrukčně náročné stejnosměrné pohony byly nahrazovány pohony s asynchronními motory s frekvenčním měničem, které docílí stejně kvalitní regulace pohonu.

Frekvenční měniče najdeme i v kombinaci se synchronními motory. Synchronní servomotory díky své konstrukci a momentové přetížitelnosti našly uplatnění v dynamicky náročných aplikacích.

1.1 Rozdělení frekvenčních měničů

Frekvenční měniče se dělí na dvě základní skupiny a to na přímé a nepřímé. Přímé měniče ze vstupního sinusového napětí vytvoří napětí s požadovanou frekvencí a napětím. Přímé měniče jsou určeny pro motory velkých výkonů. Nepřímé měniče se dají rozdělit podle stejnosměrného obvodu na měniče s obvodem napěťovým nebo proudovým. Nejpoužívanější z nepřímých měničů je napěťový měnič. Zde se nejprve střídavá elektrická energie usměrní a vytvoří se stejnosměrný „meziobvod“, který je typický velkým elektrolytickým kondenzátorem. Pak střídačem (měničem) vytvoříme střídavé napětí o požadované frekvenci (až cca 200Hz, případně i více). Tento typ měničů je v průmyslových aplikacích nejpoužívanější. Proudový měnič se od měniče napěťového liší tím, že proudový „meziobvod“ obsahuje tlumivku. Obvod je charakterizován konstantním směrem proudu. Proudové měniče se používají pro pohony velkých výkonů (nad 1 MW).

2 Přehled měničů Siemens Sinamics

Produktovou řadu Sinamics lze rozdělit na dvě hlavní skupiny. Sinamics Sxxx pro energeticky nebo dynamicky náročné aplikace a na Sinamics Gxxx pro základní aplikace. V nabídce jsou i vysokonapěťové frekvenční měniče GM150, GL150 a SM150. Spektrum funkcí zahrnuje základní řízení pohonu s jedním motorem, dále víceosé pohony nebo dynamicky náročné aplikace. Univerzálností řady Sinamics je modulárnost a škálovatelnost jednotlivých komponent. Přičemž celá řada si zachovává jednotný styl ovládání. Různé produkty řady Sinamics jsou totiž založeny na společné hardwarové a softwarové platformě. [1]

2.1 TIA (Totally integrated automation)

Plně integrovaná automatizace je koncepce řízení společnosti Siemens, která nabízí řešení na jednotné hardwarové i softwarové základně pro různé automatizační úlohy a ve všech průmyslových odvětvích. TIA znamená i propojení měničů Sinamics jednou výkonnou komunikační sběrnicí. Všechny měniče řady Sinamics (kromě G110, který klade důraz na nízkou cenu) podporují propojení na Profibus a Profinet. TIA znamená i jednotnou správu dat v celém automatizovaném zařízení. Sinamics se v propojeném automatizovaném systému chová jako prostředník mezi motorem a nadřazenou úrovní.

2.2 Použití měničů Sinamics

Společnost Siemens nabízí prostřednictvím skupiny Sinamics kompletní řešení pohonů, které optimálně splňují veškeré požadavky pro použití v oblasti nízkého, středního a stejnosměrného napájení. Pohony Sinamics mohou být nasazeny od základních až po komplexní aplikace, jako jsou nepřetržité běžící nebo vysoce dynamické extrudery, odstředivky nebo výrobní stroje. Prostřednictvím přednastavených funkčních modulů dochází jak ke zkrácení doby uvedení do provozu, tak i doby potřebné k uvedení na trh, což představuje snížení nákladů. Sinamics také nabízí optimální pohony pro všechny typy obráběcích strojů. Ať už jsou to kontinuální či vysoce dynamická vřetena nebo podávací či pomocné osy obráběcích strojů.

2.3 Frekvenční měniče Sinamics S120

Koncepce tohoto měniče je navržena pro řízení náročných aplikací především v oblasti pohonů a výrobních strojů a robotů. Zvládne dynamické polohování a synchronizaci pohybů pro více os. Může pracovat samostatně nebo mu může být nadřazeno PLC z řady Simatic. Tento frekvenční měnič je nabízen ve čtyřech provedeních a to kompakt, modulární, vestavené a skříňové

Frekvenční měnič modulární (Booksize) se skládá ze dvou hlavních částí. Jsou jimi řídící jednotka (Closed-loop control) CU320, která je schopná řídit až čtyři výkonové členy, a dvojitý výkonový člen (Power unit) řídící dva servomotory. Řídící jednotka provádí výpočty pro řízení výkonového členu a současně poskytuje rozhraní pro komunikaci s nadřazeným systémem. Výkonový člen (též nazývaný motorový modul) napájí připojené motory. V této kapitole jsem vycházel [2].

AC/AC jednotky pro jednoosé aplikace		DC/AC jednotky pro víceosé aplikace		
Blocksize	Chassis	Booksize	Chassis	Cabinet
IP20	IP20	IP20	IP00/IP20	IP20 (IP21/IP23/IP54)
0,12 kW až 90 kW	110 kW až 250 kW	1,6 kW až 107 kW	75 kW až 1200 kW	1,6 kW až 4500 kW

Obr. 1: Frekvenční měnič Sinamics S120 [2]

Příklad typických aplikací frekvenčního měniče Sinamics S120 – udávané společností Siemens s.r.o.:

- Balící stroje
- Sklářské stroje
- Dřevoobráběcí stroje
- Stroje na výrobu plastických výrobků
- Textilní stroje
- Lisy, děrovačky
- Tiskařské stroje
- Manipulátory a zdvihací zařízení
- Montážní a testovací linky

Za hlavní přednosti frekvenčního měniče Sinamics S120 se dá podle společnosti Siemens s.r.o. považovat:

- Flexibilita daná modulární výstavbou
- Konfigurovatelný výkon, nabídka funkcí, počet os
- Jednoduché uvádění do provozu, autokonfigurace
- Spolupráce s asynchronními motory

2.3.1 Části frekvenčního měniče S120

2.3.1.1 Řídící jednotka

Řídící jednotka provádí výpočty pro zařízení výkonové části měniče a současně poskytuje rozhraní k nadřízenému systému, s nímž komunikuje isochronním, tedy velmi rychlým a časově přesným protokolem. Na řídící jednotku je možné připojit jeden nebo více motorových modulů.

Řídící jednotka CU320 byla navržena k řízení více pohonů. Konkrétně můžeme ovládat až

- 8 motorů v režimu U/f (vhodné pro skupinové pohony)
- 6 motorů v režimu Servo (pro velmi dynamické aplikace)
- 4 motory v režimu vektorového řízení (pro aplikace vyžadující vysokou přesnost řízení otáček a momentu)

2.3.1.2 Napájecí moduly

Napájecí moduly usměrňují střídavé napětí ze sítě a napájí stejnosměrný obvod. Jsou k dostání ve třech různých verzích, které se liší regulací stejnosměrného obvodu a systémem přenosu energie.

Existují v provedení:

- *Basic Line Module* – Nejjednodušší z napájecích modulů usměrňuje střídavé napětí se sítě a napájí neregulovaný stejnosměrný obvod. Tento napájecí modul neumí vracet energii zpět do sítě.
- *Smart Line Module* – Největší rozdílem oproti Basic modulu je, že umí rekuperovat, tedy vracet přebytečnou energii zpět do sítě.

- *Active Line Module* – Active modul umí stejně jako Smart modul napájet stejnosměrný obvod i vracet energii zpět do sítě. Rozdíl oproti Smart modulu je, že je napětí ve stejnosměrném obvodu regulované. To znamená, že pokud dojde k fluktuaci v síti, tak se neprojeví v napětí motoru.

2.3.2 Motorové moduly

Motorové moduly jsou napájeny ze stejnosměrného obvodu a konvertují ho na střídavý proud, který pak napájí asynchronní nebo synchronní motor. Výkonový rozsah těchto modulů se pohybuje od 1,6 kW až do 1200 kW. Pro menší výkony je nabízen i double motor modul, což znamená, že obsahuje dva motorové moduly.

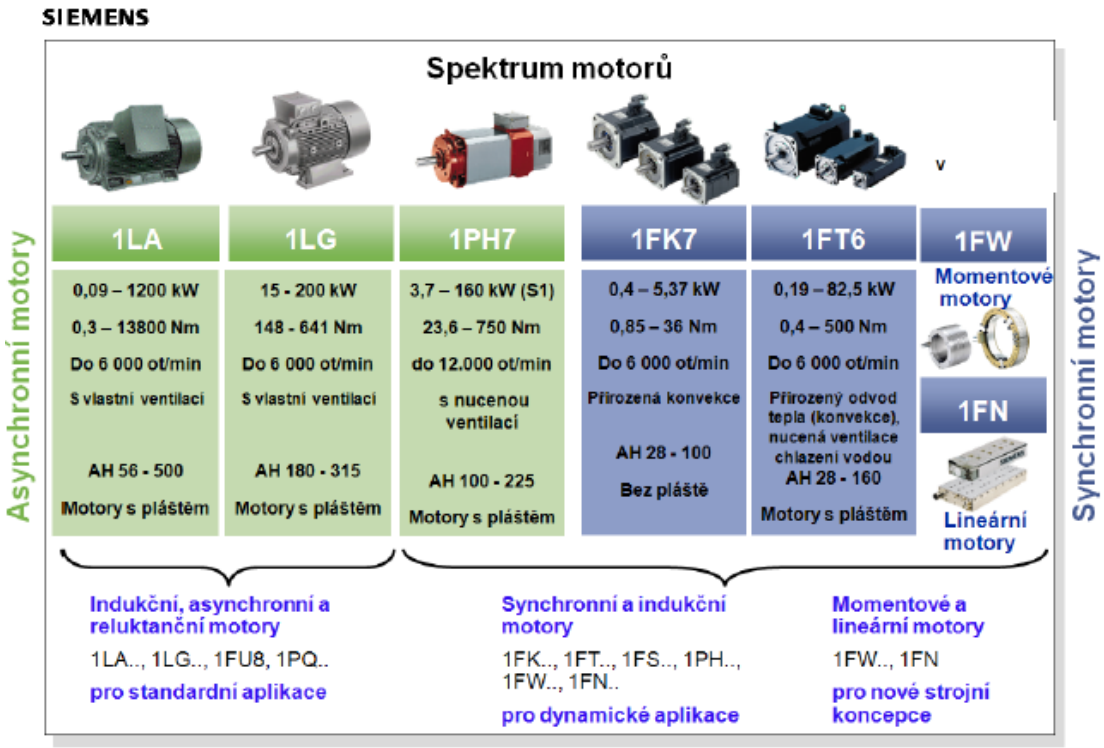
2.3.3 Sběrnice Drive-CliQ

Sběrnice Drive-CliQ je uzavřenou sběrnicí firmy Siemens určenou k propojení jednotlivých výše uvedených modulů měniče. V návaznosti na použití této sběrnice na měniči vyrábí firma Siemens i motory s tímto datovým rozhraním. Výhodou je čtení štítkových údajů přímo z motoru, včetně jedinečného čísla (ID) daného kusu motoru.

3 Motory

Motor je důležitou součástí pohonu, tudíž záleží na jeho správném výběru a nadimenzování pro správnou funkčnost celé aplikace. Pro správný výběr vhodných komponent slouží software Sizer od Siemens. S měniči Sinamics se používají dva základní typy motorů: synchronní a asynchronní. V současné době jsou pohony se stejnosměrnými motory nahrazovány asynchronními motory. To je způsobeno tím, že došlo k pokroku v regulaci asynchronních motorů pomocí frekvenčních měničů. Asynchronní motory jsou již velmi rozšířené v mnoha aplikacích kvůli jejich jednoduché konstrukci, nenáročné údržbě a příznivé ceně. Synchronní motory se nejčastěji používají v aplikacích, kde se požaduje přesné nastavení polohy, trajektorie, zrychlení, zpomalení a rychlosti. Tyto aplikace jsou například v automatizovaných výrobních linkách nebo v robotice. [9]

3.1 Přehled motorů



Obr. 2: Přehled motorů Siemens vhodných pro pohony s Sinamics [9]

4 Ovládací panely

Z hlediska ovládání strojů a zařízení jsou nedílnou součástí řídícího systému ovládací panely, které tvoří rozhraní mezi člověkem a strojem na té nejnižší úrovni. Ovládací panely lze nalézt přímo na výrobních linkách nebo jednotlivých strojích. Jejich hlavní úlohou je umožnit nastavení výrobních parametrů, spuštění nebo zastavení výrobního procesu, sledování stavu jednotlivých komponent a rovněž vyhodnocení příčin případně poruchy. V nabídce společnosti Siemens lze nalézt ovládací panely pro téměř každé použití.

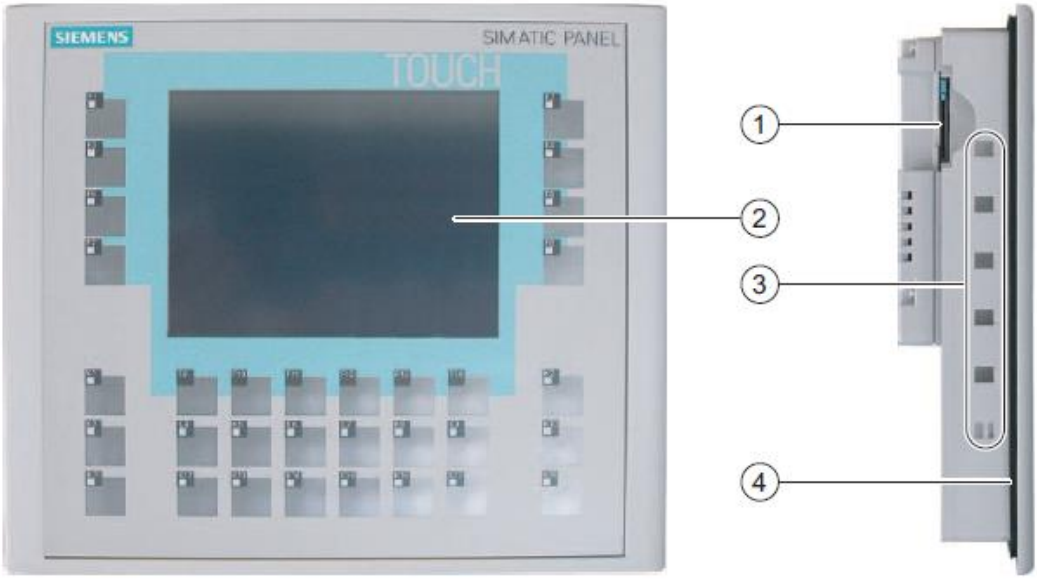
4.1 Panel OP 177B

Model OP177B je prvním panelem u kterého je dotykové ovládání kombinované s funkčními tlačítky.

Vlastnosti:

- Typ displeje LCD
- Velikost displeje 5.7“
- Rozlišení 320x240
- Počet barev 256

4.1.1 Pohled zepředu a z boku



Obr. 3: Design ovládacího panelu OP 177B [6]

Legenda:

1 Slot pro Multimediální kartu

2 Dotyková obrazovka

3 Úchyt pro montážní svorky

4 Montážní těsnění

4.1.2 Pohled zezadu

Obr. 4: Design ovládacího panelu OP 177B [6]

Legenda:

1 Slot pro Multimediální kartu

2 Štítek

3 Přepínač DIP

4 Název rozhraní

5 Použitý software pro nastavení pohonu

5.1 Simotion Scout (Starter)

Starter je nástroj, který umí nastavit pohony Siemens pomocí parametrů. Program Starter může být součástí aplikace Scout (nástroj pro Simotion) nebo integrován ve Step 7. Starter podporuje všechny měniče od firmy Siemens, tudíž nebývá problém s konektivitou. Pro základní nastavení pohonů má Starter průvodce, aby zaručeně došlo k nastavení důležitých parametrů. Nastavení se provádí v grafickém prostředí, které znázorňuje strukturu parametrů měniče. Pro zkušené uživatele tu je takzvaný Expert list, kde jsou zobrazeny veškeré parametry v seznamu.

Program Starter nabízí užitečnou funkci pro diagnostiku pohonu a to funkci Trace, která zobrazí vybrané parametry v čase a vytvoří graf. Například zobrazení aktuální rychlosti motoru.

5.2 WinCC flexible

WinCC flexible je program který slouží k vytvoření vizualizace pro ovládací panely Siemens. Vizualizace slouží k jednoduššímu a pohodlnějšímu ovládání technologie a k zobrazování a archivování veličin.

Projektant vytvářející vizualizační aplikaci, vytváří jednotlivé obrazovky, do kterých přidává objekty z knihoven. Tyto objekty mohou být funkčními prvky nebo jen grafické prvky znázorňující řízenou technologii. K dispozici jsou obsáhlé knihovny, které obsahují velké množství technologických prvků, jako jsou ventily, potrubí, motory, atd. Další funkcí WinCC je zobrazování událostí z řízené technologie. Tyto události mohou být archivovány v databázích. Použití vizualizačního softwaru a možnosti archivace měřených veličin umožňuje jednoduchou analýzu provozu. V neposlední řadě přehlednost a propracovanost aplikace eliminuje chyby obsluhy.

6 Praktická část

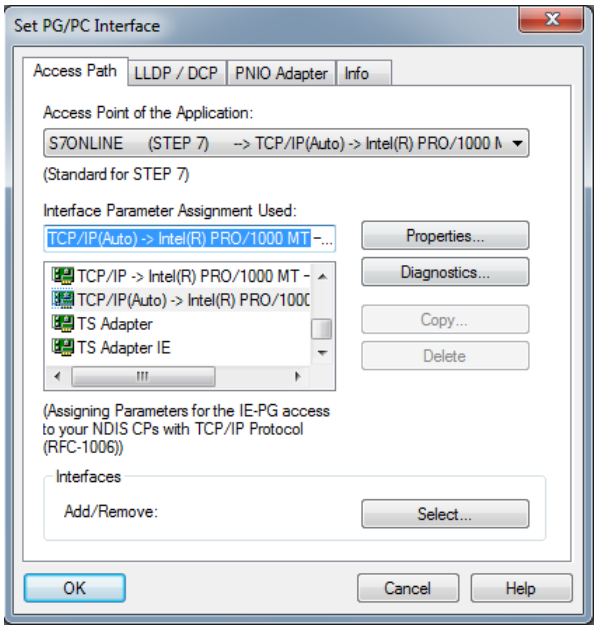
První část praktické části se věnuje parametrování měniče pro chod v otáčkové vazbě (Motor1) a pro chod v polohové vazbě (Motor2). Dále pak programování grafického prostředí pro ovládání měniče z dotykového panelu a jak přistupovat k parametrům, aby bylo možné kontrolovat a řídit měnič.

Měnič Sinamics S120 má všechny svoje vlastnosti a nastavení uložené v parametrech, tudíž parametrováním měniče je myšleno nastavování těchto vlastností a tím i měniče. Jelikož je těchto parametrů celá škála, firma Siemens vyvinula software Starter, který umí procházet tyto parametry v grafickém prostředí a pro některé nastavení má i průvodce. Starter obsahuje i různé měřící funkce a panel pro ovládání měniče. Díky těmto funkcím je Starter důležitým nástrojem pro nastavování měničů.

Druhá část se věnuje synchronizační funkci. Do Starteru přidáme takzvaný Drive Control Chart (DCC), který stáhneme ze stránek firmy Siemens. Tento Drive Control Chart obsahuje synchronizační bloky, které jsou napojeny na parametry měniče a propojovány mezi sebou pomocí grafického editoru, tudíž už jsou plně nastaveny a nám jen zbývá nastavit parametry, aby mohla synchronizace začít. Tyto parametry jsou pak znovu vyvedeny na ovládací panel, kterým vše řídíme.

6.1 Nastavení pohonu pomoci Simotion Scout

Nejprve je potřeba propojit měnič s počítačem. K tomu slouží sběrnice Profinet. Profinet pracuje na bázi Ethernetu a jeho výhodou je práce v reálném čase. Potom, co hardwarově propojíme měnič a PC, můžeme spustit program Scout a založit nový projekt. Klikneme na Insert single drive unit a přidáme jednotku Sinamics S120 CU320. Dále je potřeba nastavit komunikaci mezi měničem a PC. V menu Options vybereme Set PG/PC Interface a jako rozhraní nastavíme síťovou kartu (TCP/IP(Auto) -> Intel(R) PRO/1000 MT).



Obr. 5: Nastavení komunikace

V tuhle chvíli máme vše nastavené a můžeme přepnout do online modu a začít nastavovat měnič. Do online modu se přepneme pomocí tlačítka *Connect to Target system*. To zda jsme v online nebo offline modu zjistíme tak, že se podíváme do pravého dolního rohu v programu Scout.

Obr. 6: Tlačítko Connect to Target systém

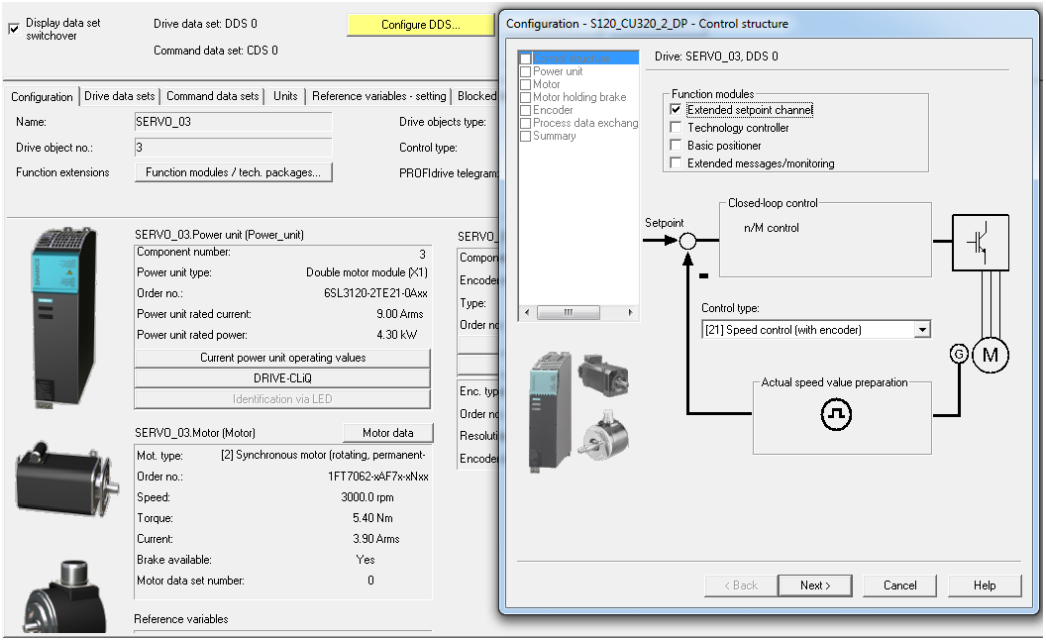
Dále je potřeba nakonfigurovat zařízení. Jsou dvě možnosti. Kliknutím na *automatic configuration* spustíme automatickou konfiguraci a vše se nastaví samo nebo můžeme nakonfigurovat pohon ručně v offline modu a pak konfiguraci nahrát do měniče. Automatická konfigurace využívá systém Drive-CliQ, kde je celá topologie pohonu včetně štítkových údajů zjištěna automaticky. Můžeme kliknout na *Topology* a podívat se, zda došlo ke správné detekci HW.

6.1.1 Nastavení os (motorů)

6.1.1.1 Osa v otáčkové vazbě

Pro práci v otáčkové vazbě bude nastaven motor1. Ve stromové struktuře rozevřeme *motor1* a klikneme na *configuration*. Otevře se nám okno, kde jsou

zobrazeny všechny části osy se svými štítkovými údaji. Klikneme na *Configure DDS* (Drive Data Set). Zobrazí se nám průvodce, ve kterém nastavíme osu na *Extended setpoint channel* – zadávání rychlosti. Jako kontrolní metodu zvolíme *Speed control (with encoder),* otáčková vazba s encoderem. Další položky stačí pouze zkontrolovat, ale žádné změny pro náš projekt není třeba provádět, jelikož všechny údaje byly načteny při automatické konfiguraci.



Obr. 7: Nastavení motoru v otáčkové vazbě

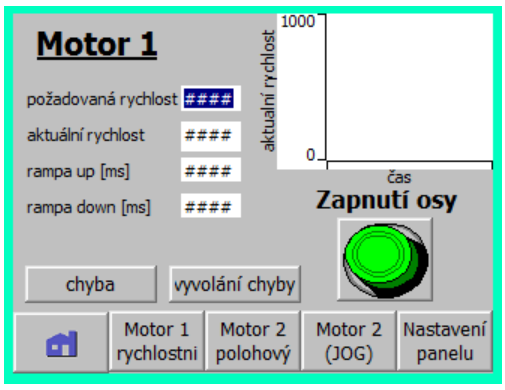
Následně je potřeba nechat měnič, aby si osu proměřil a doplnil si údaje o motoru. K tomu vybereme ve stromové struktuře *commisioning* a *stationary/turning measurments*. Otevře se nám okno, kde vybereme measurment type: *complete calculation of the motor/control parameters* a necháme menič měřit. Dojde k sérii měření. Až bude měnič nastaven, můžeme si vyzkoušet, zda se motor v pořádku točí. K tomu slouží nástroj Control panel, který najdeme ve stromu pod *Commisioning*. Control panel je ovládací pult, který ovládá měnič, můžeme z něho zapínat a vypínat osu a zadávat rychlost otáčení. Control panel má i mnoho jiných funkcí které se týkají polohování, to si ukážeme při nastavování druhé osy.

Pro ovládání měniče nástrojem control panelem je třeba „převzít kontrolu“, to znamená, že znemožníme ovládat měnič jiným způsobem (např. Digitálními vstupy). Motor je proměřený a točí se.

Při nastavování osy vznikala současně vizualizace s parametrizací měniče. Nyní však předpokládejme, že vizualizaci máme již hotovou z důvodu přehlednosti, samotnou vizualizací se budeme zabývat v následující kapitole. Tedy máme přepínač pro zapnuto/vypnuto, pole pro zadání rychlosti, pole pro dobu náběhu a dobu doběhu, tlačítko pro vyvolání poruchy a pro potvrzení poruchy. Dále předpokládejme, že WinCC od měniče očekává údaje o rychlosti a stavu měniče.

Obr. 8: Zapnutí měniče

Zapnutí a vypnutí měniče se parametruje v položce *Control logic* parametrem p840. Tento parametr se nedá z WinCC měnit přímo, proto si musíme pomoci parametrem p2099, který mění hodnotu parametru r2094. Parametrer p2099 je rozdělen do 16 bitů, nám stačí pouze jeden bit pro zapnutí a vypnutí, takže do parametru p840 zapíšeme hodnotu r2094 bit 0, který budeme měnit z WinCC.



Obr. 9: Ovládání motoru 1

Pro nastavení rychlosti využijeme takzvanou fixní frekvenci. Fixní frekvence je pevně zadaná hodnota, kterou zapínáme a vypínáme pomocí digitálního vstupu.

Fixních frekvencí je možné do měniče uložit více a pak pomocí digitálních vstupů mezi nimi přepínat, my však zapneme trvale pouze jednu fixní frekvenci a budeme měnit její uloženou hodnotu. Ve stromové struktuře vybereme *Setpoint chanel* a *Speed Setpoint*. Hlavní hodnotu rychlosti (main setpoint p1070) nastavíme na fixní frekvenci p1001. Pak ve stromové struktuře vybereme *Fixed setpoints* a nastavíme parametr p1020 na hodnotu 1, tím trvale sepneme fixní frekvenci p1001. Tímto jsme řekli měniči, aby jeho *main setpoint* byla brána z parametru p1001, kterou budeme měnit pomocí vizualizace.

Dále nastavíme rychlostní rampy, což jsou časy, které udávají, za jakou dobu se motor rozeběhne na požadovanou rychlost a naopak, za jakou dobu se zastaví. Nastavení se provádí v záložce *Ramp-function generator* pomocí parametrů p1120 a p1121. Tyto parametry budeme měnit přímo ve vizualizaci.

Dalším tlačítkem chceme simulovat externí chybu měniče a následně ji potvrdit. Externí chybou může například být, když se člověk dostane do prostoru, kde by mohlo dojít k úrazu, a je nutné zastavit proces. Externí chyba má parametr p2106 a najdeme ho v záložce *control logic*. Proces nelze opětovně spustit, dokud není chyba odstraněna a její odstranění není potvrzeno na měniči. Externí chyba se potvrzuje parametrem p2103, který najdeme také v záložce *control logic*. Oba tyto parametry budeme měnit přímo z vizualizace.

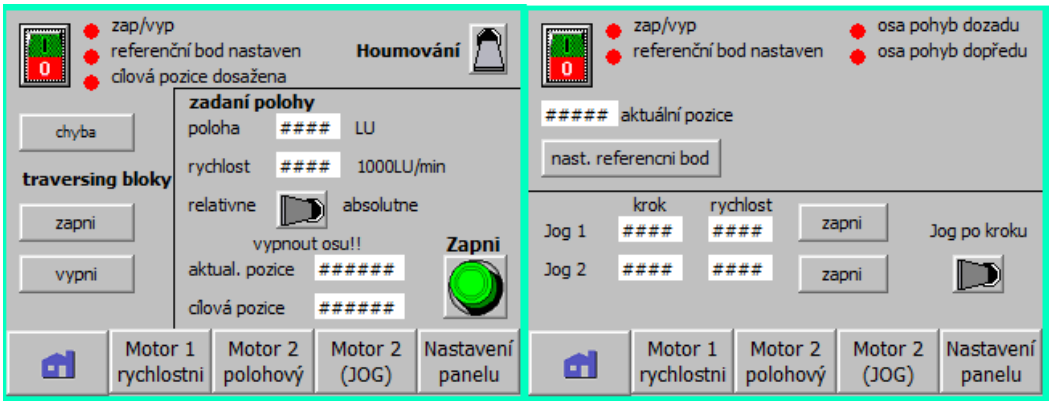
Tímto máme nastavené parametry, které vstupují do měniče, ještě však budeme chtít sledovat, aktuální stav motoru. Konkrétně nás zajímá aktuální rychlost, která má parametr r63. Dále nás pak zajímá, jestli je měnič v chybě, k tomu slouží takzvaná stavová slova. Stavové slovo je skupina bitů, kde každý reprezentuje nějaký stav měniče. Takové slovo najdeme v parametru r2139, kde třetí bit signalizuje přítomnost externí chyby. Tímto máme nastavené vše pro osu s otáčkovou vazbou.

6.1.1.2 Osa v polohové vazbě

Základní nastavení provedeme stejně jako s motorem 1. Ve stromové struktuře rozevřeme „motor 2“ a klikneme na *configuration* a zde na *Configure DDS*, ale tentokrát zaškrtneme *Basic positioner* a *closed-loop control*. Pokračujeme dále v průvodci a kontrolujeme nastavené hodnoty. Stejně jako u motoru 1 je potřeba proměřit motor, tedy znovu vybereme ve stromové struktuře *commisioning* a *stationary/turning measurments* a vybereme *complete calculation of the motor/control parameters* a sputíme automatickou proceduru měření pohonu

měničem. I zde je vhodné vyzkoušet pomocí control panelu, zda se motor chová tak, podle očekávání. Control panel má možnost polohování, tedy je možné zadat přesnou polohu a maximální povolenou rychlost a menič zajistí, že bude této polohy dosaženo. Navíc umí takzvaný *Jog*, což je krokování. Nebo umožnuje vyzkoušet si *Homing (referencování)*, což je vrácení se do referenční polohy.

Na obrázku vidíme námi vytvořený panel pro polohovou osu ve WinCC, postupně si projdeme všechny ovládací prvky a vysvětlíme si jejich funkce.



Obr. 10: Ovládání motoru 2

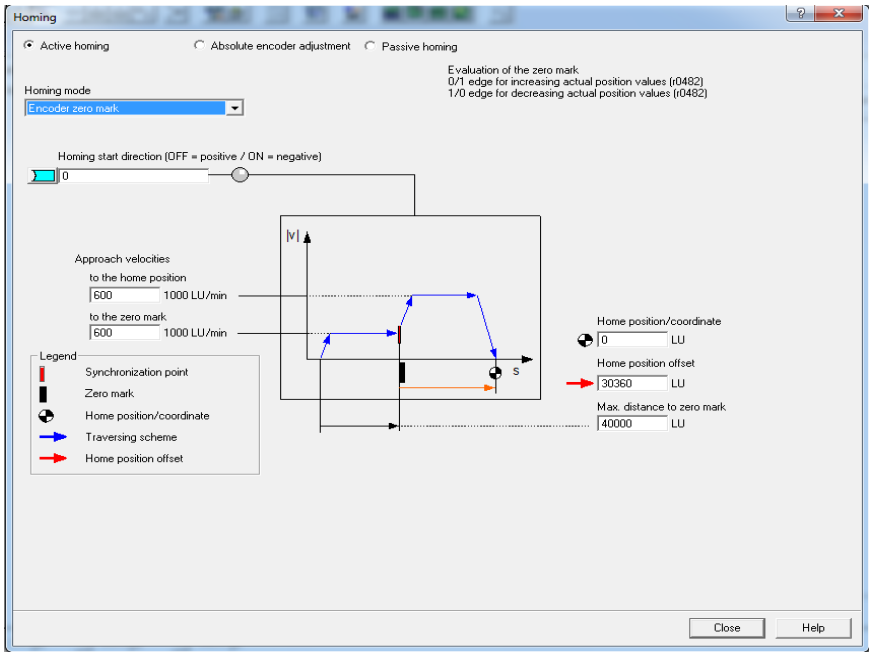
Zapínání a vypínání osy a potvrzování chyby je stejné jako u osy v otáčkové vazbě.

Homing

Abychom mohli využívat polohovací funkce, potřebuje měnič znát nulovou polohu. Toho se docílí operací, která se nazývá *referencing* a máme tyto možnosti:

- Ručně zadat referenční polohu – když je motor v poloze, kterou chceme mít referenční, tak změníme hodnotu parametru p2596.
- Inkrementální enkodér – menič sám najde referenční značku enkodéru nebo najede na referenční vačku.
- Absolutní enkodér – Nulová poloha se nastavuje pouze jednou a to při prvním uvedení do provozu.

V našem pohonu máme inkrementální čidlo, tedy po zapnutí měniče je potřeba „ukázat“ na referenční bod, v našem případě tedy zvolíme *referencing* na nulovou značku encodéru. Nastavení bude vypadat jako na obrázku 11.

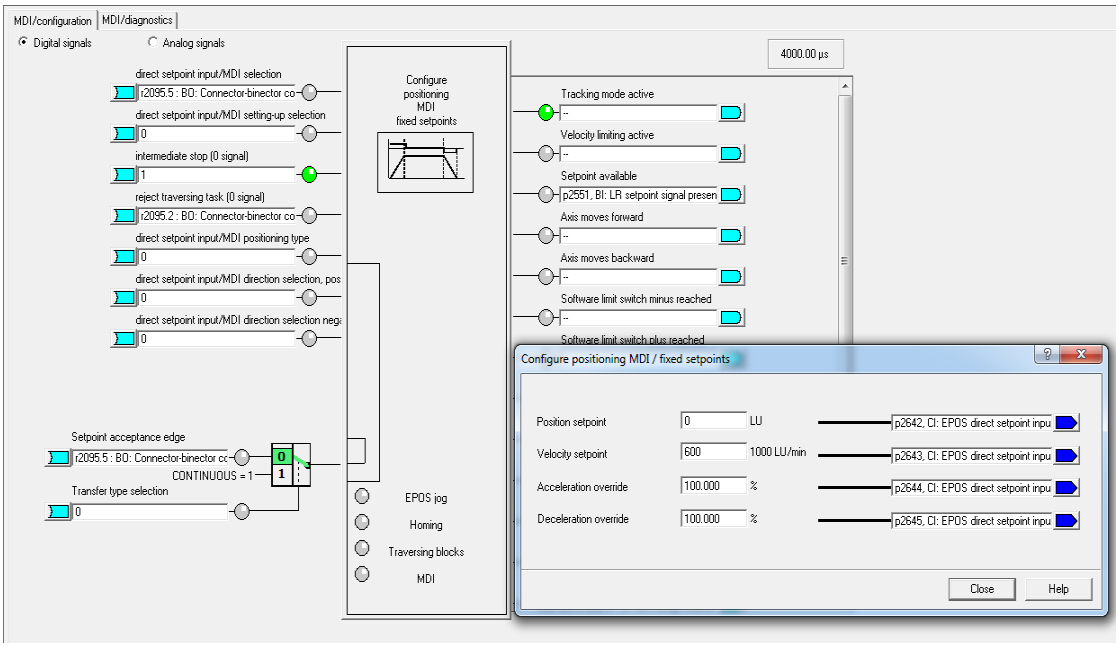


Obr. 11: Homing (referencing)

Dále je potřeba nastavit *referencing start*, do pole reprezentující parametr p2595 dáme hodnotu r2095 bit 1, tím naparametrujeme spuštění referencování.

Přímé zádávání polohy

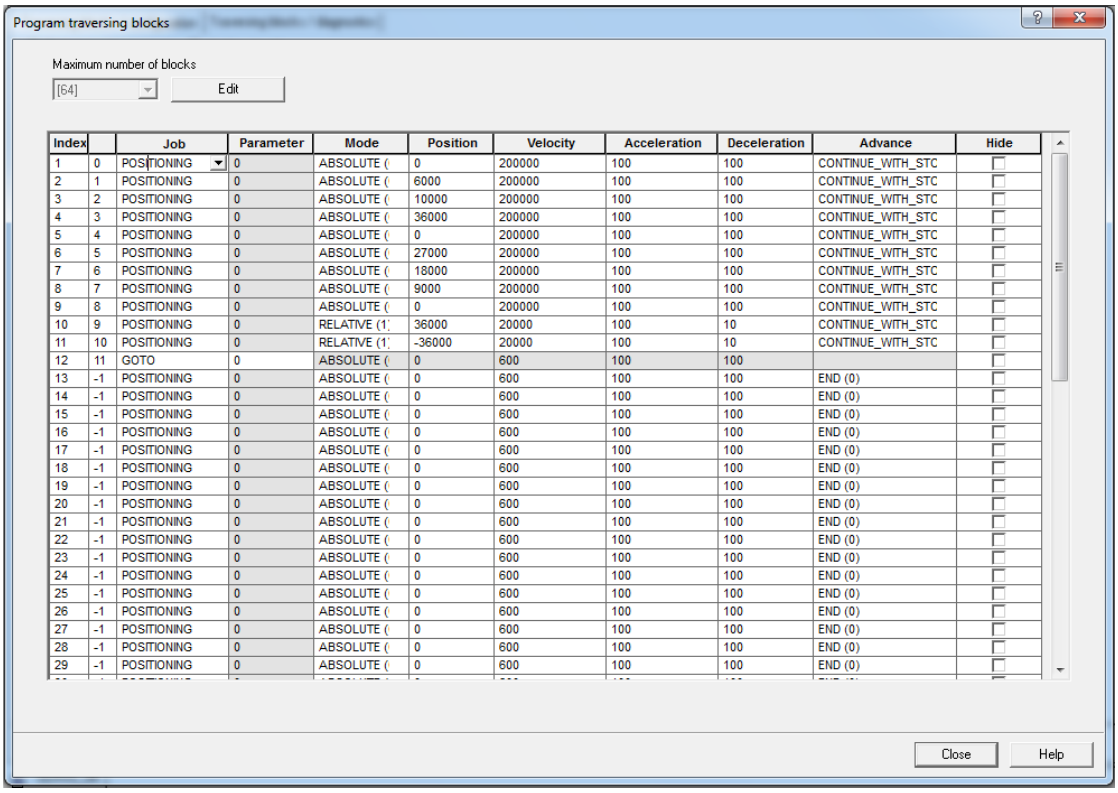
Přímou polohu můžeme zadávat dvěma způsoby, buďto relativně nebo absolutně. Relativní zadání znamená, že se pozice bude počítat od aktuální polohy motoru. Absolutní zadávání znamená, že se pozice bude počítat od nulové polohy (reference point). K přepínání mezí absolutním a relativním zadáváním slouží parametr p2648. Pokud je parametr v jedničce, polohování bude probíhat absolutně, pokud je v nule, tak relativně. Pro spuštění polohování slouží parametr p2647, do kterého dáme hodnotu r2095 bit 5. K nastavení pozice a rychlosti slouží parametry p2642 a p2643. Kromě nastavení rychlosti a pozice je možné taky nastavit zrychlení a zpomalení. Funkci přímého zadávání polohy najdeme ve Scoutu v záložce *Technology, Basic positioner* a *Direct setpoint specification/MDI*.



Obr. 12: Přímé zadávání polohy

Traversing blocks

Pokud chceme, aby pohon vykonával různé úlohy za sebou, například otočit o 60 stupňů doprava, počkat 5 vteřin a otočit o 90 stupňů doleva, pak použijeme *Traversing blocks*. Můžeme zadat až 64 různých úloh (tzv. tasks), které se mají vykonat. Tyto „úlohy“ zadáváme v záložce *Technology*, *Basic positioner, Traversing blocks* a *Program traversing blocks*.

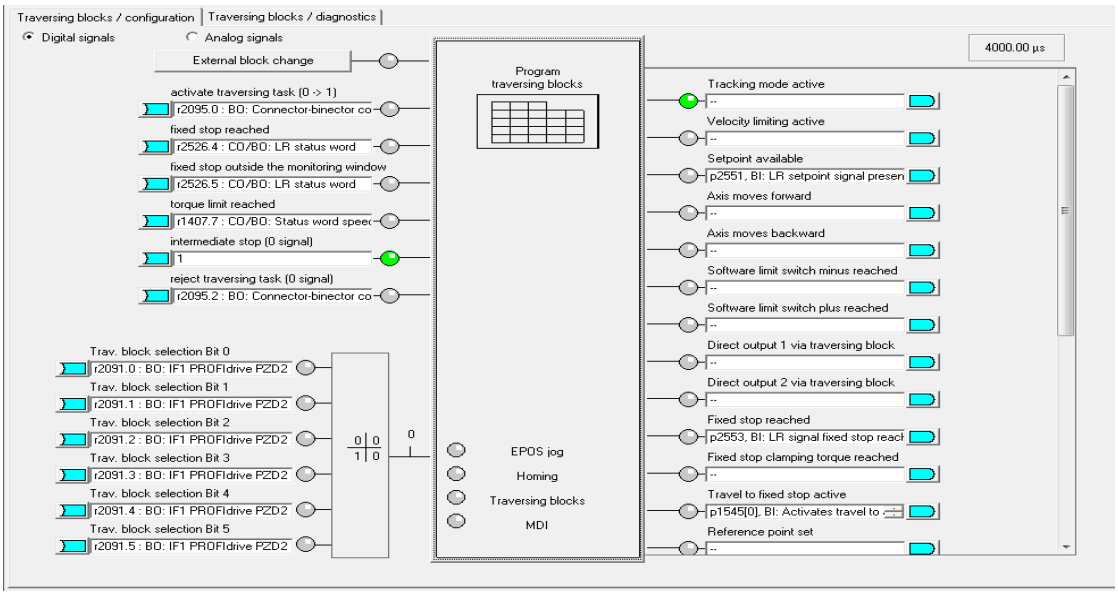


Index		Job	Parameter	Mode	Position	Velocity	Acceleration	Deceleration	Advance	Hide
1	0	POSITIONING	0	ABSOLUTE (	0	200000	100	100	CONTINUE_WITH_STC	
2	1	POSITIONING	0	ABSOLUTE (	6000	200000	100	100	CONTINUE_WITH_STC	
3	2	POSITIONING	0	ABSOLUTE (	10000	200000	100	100	CONTINUE_WITH_STC	
4	3	POSITIONING	0	ABSOLUTE (	36000	200000	100	100	CONTINUE_WITH_STC	
5	4	POSITIONING	0	ABSOLUTE (	0	200000	100	100	CONTINUE_WITH_STC	
6	5	POSITIONING	0	ABSOLUTE (	27000	200000	100	100	CONTINUE_WITH_STC	
7	6	POSITIONING	0	ABSOLUTE (	18000	200000	100	100	CONTINUE_WITH_STC	
8	7	POSITIONING	0	ABSOLUTE (	9000	200000	100	100	CONTINUE_WITH_STC	
9	8	POSITIONING	0	ABSOLUTE (	0	200000	100	100	CONTINUE_WITH_STC	
10	9	POSITIONING	0	RELATIVE (1	36000	20000	100	10	CONTINUE_WITH_STC	
11	10	POSITIONING	0	RELATIVE (1	-36000	20000	100	10	CONTINUE_WITH_STC	
12	11	GOTO	0	ABSOLUTE (	0	600	100	100		
13	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
14	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
15	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
16	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
17	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
18	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
19	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
20	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
21	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
22	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
23	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
24	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
25	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
26	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
27	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
28	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	
29	-1	POSITIONING	0	ABSOLUTE (	0	600	100	100	END (0)	

Obr. 13: Traversing tasks

Podíváme-li se do naší tabulky traversing tasků, tak uvidíme, že task číslo 0 absolutně zapolohuje do pozice 0 (reference point). Task 1 absolutně zapolohuje na pozici 6000. Jelikož máme nastaveno, že jedna otáčka odpovídá 36000 LU, tak pozice 6000 LU je od pozice 0 vzdálena 60 stupňů. Další zajímavý task je číslo 11 (GOTO), který zajišťuje návrat cyklu na začátek.

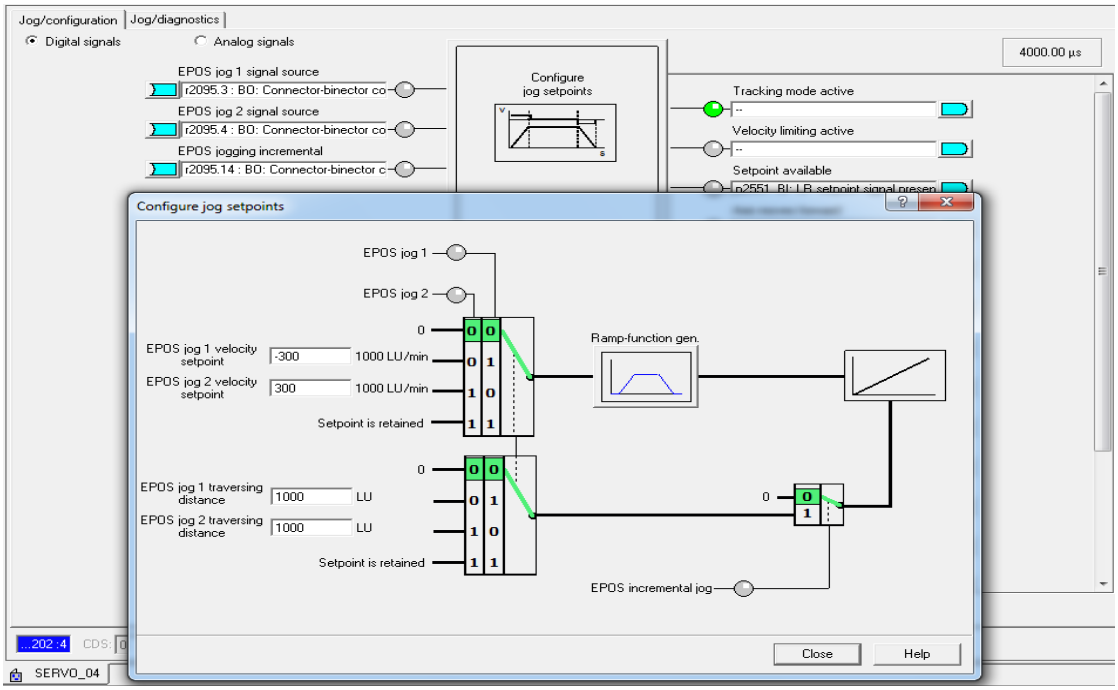
Ještě je potřeba nastavit parametr p2631 na R2095 bit 0, tím nastavíme spouštění traversing bloků pomocí r2095. Podobně nastavíme i vypínání traversing bloků, p2641 nastavíme na r2095 bit 2. Na parametry r2095 budeme přistupovat z WinCC pomocí paramatru p2099.



Obr. 14: Traversing blocks

Jog

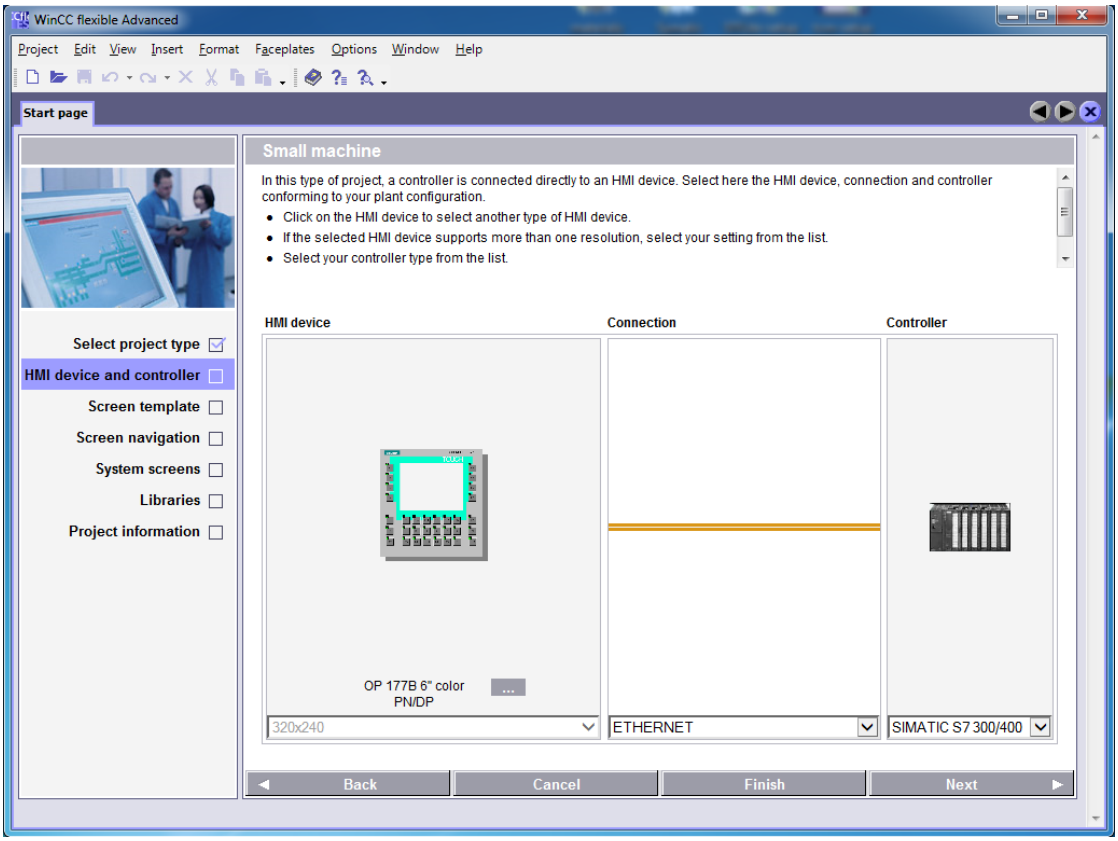
Funkce *Jog*, neboli krokování, funguje tak, že nastavíme hodnotu, která udává o kolik a jak rychle se bude osa motoru pohybovat. Funkci *Jog* najdeme v položce *Basic positioner* a *Jog*. Zde nastavíme digitální vstupy, kde budeme krokování ovládat. Vstupy jsou dva, obvykle se jedním spouští krok doprava a druhým krok doleva, ale není to vždy podmínkou.



Obr. 15: Nastaveni Jog

6.2 Programování vizualizace pomocí WinCC

Vytváření vizualizace zahájíme spuštěním průvodce, který nám plno věcí usnadní. Vybereme *Small machine*, jelikož naše sestava je z pohledu WinCC jen jedno zařízení s jedním HMI a klikneme na *next*. Na další obrazovce je potřeba nastavit typ zobrazovacího zařízení, typ kontrolovaného zařízení a spojení mezi nimi. Naše zobrazovací zařízení je ovládací panel, takže nastavíme HMI device jako OP177B. Měnič Simatic S120 se chová vůči WinCC jako řídící systém Simatic S7-300, takže jako kontrolované zařízení nastavíme Simatic S7 300/400. Komunikace bude probíhat po Ethernetu.



Obr. 16: Nastavení WinCC

V dalším okně se nastavuje vzhled vizualizace, my však nic měnit nepotřebujeme a tak přejdeme na další okno, kde si nastavíme, kolik chceme navigačních oken. Kromě jedné *Start Screen* si přidáme tři *Section Screen* a klikneme na *next*. Zde pak zaškrtneme *System Screen*, tím si přidáme další okno, ve kterém bude nastavení panelu. V dalším kroku pak vybereme všechny nabízené knihovny a v posledním okně pak můžeme vyplnit svoje iniciály a popis projektu. Pak klikneme na tlačítko *finish* a tím dokončíme průvodce.

Poté se nám otevře pracovní plocha. V horní části obrazovky najdeme menu a ikony často používaných funkcí. Vlevo najdeme stromovou strukturu našeho projektu. Vpravo najdeme knihovnu s objekty a dole okno s vlastnostmi právě označeného objektu. Nakonec budeme upravovat vzhled vizualizační aplikace na pracovní ploše umístěné uprostřed.

6.2.1 Tagy

Tagy jsou proměnné, se kterými budeme v projektu pracovat. Jsou dva druhy a to vnitřní a vnější. Vnitřní tagy jsou využívány jen vizualizační aplikací, kdežto vnější tagy jsou propojeny s kontrolovaným zařízením (měničem). Každá proměnná má svůj datový typ, adresu a dobu obnovení. Pokud chceme měnič ovládat, musíme jeho parametry propojit s WinCC a to právě pomocí tagů. Každý tag má svojí adresu, ve které se ukrývá určení objektu měniče (motor1, motor2), číslo parametru a číslo indexu. Pro sestavení adresy platí následující vzorec:

[1] DB[Par].DBD([Obj]*1024 + [Ind])

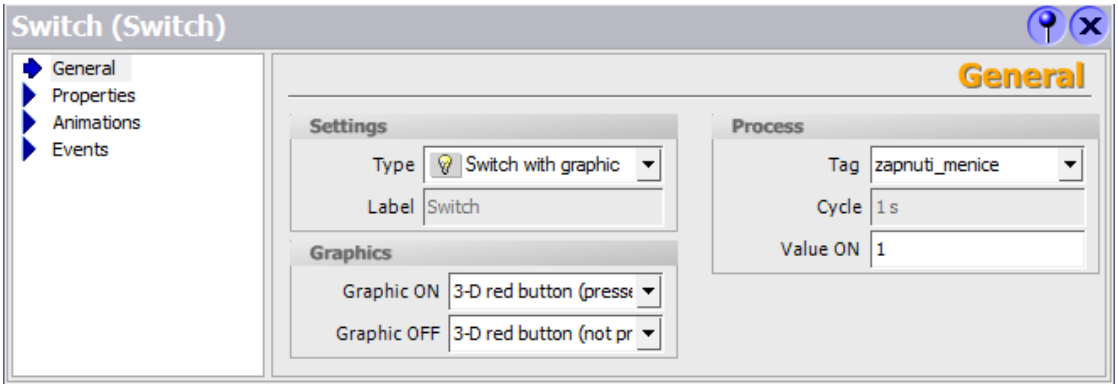
- DBD platí pro datové typy Real a DInt
- DBW platí pro datové typy Word a Int
- Par je číslo parametru
- Ind je index parametru
- Obj je číslo objektu v měniči

Číslo daného objektu najdeme ve Scoutu v záložce *Telegram configuration*.

6.2.2 První obrazovka (Start screen)

První obrazovka bude obsahovat zapnutí měniče. Pro zapínání a vypínání použijeme z knihovny *Simple Objects* objekt *Switch*, což je dvoupolohový přepínač. Ve vlastnostech změníme typ přepínače na *Switch with Graphic* a přiřadíme obrázek pro oba stavy. Poté si vytvoříme tag s názvem *zapnuti_menice* a adresou DB2098 DBW2048 s datovým typem Word. Adresa odpovídá vzorci[1], chceme měnit parametr p2098 s indexem 0 v objektu 2. Následně ve vlastnostech přepínače

v *process* vybereme náš tag a nastavíme *Value ON* na hodnotu 1. Vlastnosti objektu by měli vypadat následovně:



Obr. 17: Vlastnosti přepínače

Obr. 18: Start screen

6.2.3 Druhá obrazovka (osa v otáčkové vazbě)

Pro osu v otáčkové vazbě (motor1) budeme chtít zadávat rychlost otáčení, náběžnou rampu, doběhovou rampu, vyvolávat poruchu a potvrzovat odstranění poruchy a také zapínat a vypínat osu. Mimo tyto funkce, které jsou vstupem do měniče, budeme také chtít sledovat aktuální rychlost otáčení.

Zapínání a vypínání osy

Zapnutí a vypnutí osy bude stejné jako zapnutí a vypnutí měniče s tím rozdílem, že bude jiný pouze tag. Tedy vytvoříme si tag, který bude mít název *zapvyp1* a adresu DB2098 DBW3072 s datovým typem Word. Adresa opět odpovídá vzorci[1], chceme měnit parametr p2098 s indexem 0 v objektu 3.

Zadání a měření rychlosti

Zadávat rychlost budeme tak, že změníme velikost rychlosti uloženou ve fixní frekvenci, k tomu slouží parametr p1001. Vytvoříme si tag s názvem *fixedfrek1*, který bude mít adresu DB1001 DBD3072 a bude datového typu Real. V *Simple Objects* vybereme *IO Field* a do *Process* vybereme náš tag *fixedfrek1*. Ještě nastavíme *Mode* na *Input/Output,* aby bylo možné pomocí tohoto pole zadávat číselnou hodnotu.

Stejně vytvoříme pole pro aktuální rychlost, s tím rozdílem, že *Mode* bude nastaven na *Output*. Tag pro aktuální rychlost má název *actspeed1* a adresu DB63 DBD3072 s datovým typem Real. Rychlost se odečítá z parametru r63.

Náběžná a doběhová rampa

Vytvoříme si dva tagy, *rampup1* a *rampdown1*. Tag pro náběžnou rampu (rumpup1) má adresu DB1120 DBD3072 s datovým typem Real. Tag pro doběhovou rampu (rampdown1) má adresu DB1121 DBD3072 také s datovým typem Real. Potom vytvoříme další dvě pole (IO Field) s *Mode* nastaveným na *Input/Output* a přiřadíme jim nově vytvořené tagy.

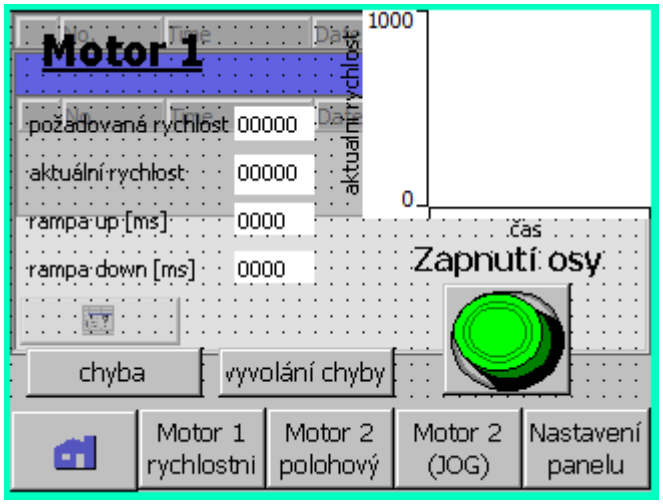
Vyvolání a potvrzení poruchy

Pro vyvolání poruchy použijeme tlačítko (Button), které má tag s názvem *fault1* a adresu DB2106 DBD3072 s datovým typem DWord. Tlačítko bude mít obrácenou logiku, tedy měnič nebude v poruše, pokud bude parametr p2106 v hodnotě 1. Takže stiskem tlačítka budeme měnit parametr z 1 na 0 a při uvolnění tlačítka zpět na 1. Rozklikneme záložku *Events*, kde najdeme události, na které může tlačítko reagovat. Vybereme *Press* (na stlačení) a nastavíme reakci na událost *SetValue,* vybereme tag *fault1* a *Value* nastavíme na 0. Podobně nastavíme reakci na událost *Release* (uvolnění), kde přidáme stejný tag, jenom hodnotu nastavíme na 65536, tato hodnota je měničem brána jako 1.

Vytvoříme si tag *faultackn1* s adresou DB2103 DBD3072 a typem DWord. Tlačítko pro potvrzení odstranění poruchy bude podobné jako tlačítko pro vyvolání poruchy. Reakcí na událost *Press* bude *SetValue* v tagu faultackn1 na hodnotu 65536 a na událost *Release* bude *SetValue* na hodnotu 0. Tlačítko pro potvrzení odstranění poruchy bude navíc fungovat jako signalizace, zda je menič v poruše, takže si

vytvoříme tag, který bude z měniče číst stavové slovo. Stavové slovo se skládá ze stavových bitů, které mohou nabývat hodnot 1 nebo 0 a každý nese význam určitého stavu měniče. Stavové slovo, které signalizuje přítomnost poruchy je ukryto v parametru r2139 pod bitem 3. Takže si vytvoříme tag, který bude mít adresu DB2139 DBB3072 a bude datového typu Byte a nazveme ho *statusfault1*.Ve vlastnostech tlačítka pro potvrzení odstranění poruchy v záložce *Animations* a *Appearence* zaškrtneme *Enabled*. Zvolíme připravený tag a *type* změníme na Bit a nastavíme hodnotu 3. Tím říkáme, že se vzhled tlačítka bude měnit podle bitu 3 v tagu *statusfault1*. Do tabulky přidáme jeden řádek, kde ve sloupečku *Value* bude 1, *Foreground Color* černá, *Background Color* červená, *Flashing Yes*. Tím jsme nastavili, že tlačítko bude při chybě blikat.

Tímto máme vizualizaci pro osu v otáčkové vazbě hotovou. Návrh vizualizace může vypadat například takto:



Obr. 19: Osa v otáčkové vazbě

6.2.4 Třetí obrazovka (osa v polohové vazbě)

Po ose v polohové vazbě (motor2) budeme požadovat funkci *homing*, což je funkce která otočí motorem do polohy home. Dále Přímé zadávání polohy a to jak absolutně, tak relativně s možností měnit rychlost otáčení. Pak možnost spustit a zastavit *Traversing tasks*, což jsou sekvence požadovaných pohybů motoru. Nebude na škodu přidat i aktuální a požadovanou polohu motoru. A v poslední řadě si přidáme stavové slovo pro signalizaci zapnutí a vypnutí motoru, pak je-li nastaven nulový bod (reference point) a jestli byla dosažena žádaná poloha.

Zřejmě nemá smysl popisovat podrobně propojení každého prvku s parametrem, takže přidám pouze soupis tagů a stručný popis parametru, na který je tag adresován.

Zapnutí/vypnutí osy

Zapnutí a vypnutí osy je podobné jako u předchozí osy, liší se akorát tag, který má tentokrát adresu DB2098 DBW4096 a nese jméno *zapvyp2*. Adresa se liší, protože odkazuje na motor2 který má jiné číslo objektu než motor1. Vše opět vyplývá ze vzorce [1].

Signalizace zapnutí/vypnutí osy

Vytvoříme si signalizační „diodu“ a to tak, že v knihovně *Simple Objects* vybereme *Circle* a vytvoříme kruh. Ve vlastnostech pak upravíme požadovaný vzhled a přidáme popisek. Pak si vytvoříme tag *diagmot2*, který má adresu DB898 DBB4096 s datovým typem Byte. Tento tag přivedeme do vlastností kruhu a to tím způsobem, že v záložce *Appearance* zvolíme, aby byl při hodnotě bitu 0 *Background* červený a při hodnotě 1 bude *Background* zelený.

Signalizace nastavení nulového bodu (Reference point)

Postup je stejný jako v případě signalizace zapnutí/vypnutí osy. Jenom tag má název *status2*, adresu DB2684 DBW4096 a bit který signalizuje nastavený nulový bod je bit číslo 11.

Signalizace dosažení cílového bodu

Postup je stejný jako u signalizace nulového bodu, tag má i stejný název a adresu jen bit, který signalizuje dosažení cílového bodu je bit číslo 10.

Tlačítko pro potvrzení poruchy

Tlačítko pro potvrzení poruchy funguje stejně jako u předchozí osy. Tag pro potvrzení poruchy *faultackn2* má adresu DB2103 DBD4096 a reaguje na událost *Press* a *Release*. Tag pro signalizaci poruchy má adresu DB2139 DBB4096 a reaguje na stav v bitu 3.

Homing

Na pracovní plochu si přidáme z knihovny *Simple Objects Switch*, kterému ve vlastnostech nastavíme *Switch with graphic* a vybereme obrázky, které budou znázorňovat polohou zapnuto a vypnuto. Vytvoříme si interní tag, který nastavíme ve vlastnostech přepínače do *Process*. Tento tag není propojen s měničem, je zde jen za účelem toho, aby přepínač fungoval správně, protože potřebujeme měnit bit 1 v parametru p2098. Vytvoříme další tag s názvem *homing2* a adresou DB2098 DBW4097.Ve vlastnostech přepínače v záložce *Events* nastavíme funkci událostem *Switch On* a *Switch Off*. Funkce pro *Switch On* bude *SetBitInTag* pro bit 1. A *Switch Off* bude tento bit resetovat do nuly, tedy *ResetBitInTag*.

Traversing bloky

Traversing bloky se programují ve Scoutu, viz kapitola[6.1.1.2]. Samozřejmě by bylo možné je naprogramovat i ve WinCC, ale znamenalo by to udělat vstupní pole pro 64 bloků, přičemž každý z nich má navíc 9 vlastností. Traversing bloky budeme ovládat pomocí tlačítek zapni a vypni. Tlačítko zapni má dvě události, na které reaguje, jedna je *Press* a druhá *Release*. Ke spuštění traversing bloků je potřeba odblokovat traversing bloky a zároveň je spustit, takže v události *Press* bude dvakrát funkce *SetBitInTag* pro tag *traverstask2* a to v jednom případě pro bit 0 a ve druhém pro bit 2. Událost *Release* bude mít v sobě funkci *ResetBitInTag* pro bit 0 v tagu *traverstask2*. Tlačítko vypni bude na *Press* resetovat bit 2 v tagu *traverstask2* a na *Release* zase vrátí tento bit do jedničky, aby nebyly Traversing bloky zablokovány.

Přímé zadání polohy

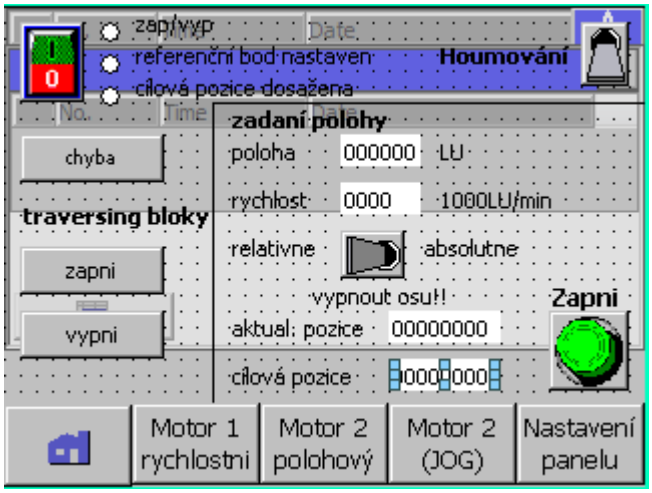
Pro zadání polohy a rychlosti vytvoříme dvě pole (IO field). První pole pro zadání polohy bude mít tag s názvem *MDIpoloha2* a adresu DB2690 DBD4096. Druhé pole pro zadání rychlosti bude mít tag s názvem *MDIrychlost2* a adresu DB2691 DBW4096. Obě pole budou nastaveny na *Mode Input/Output*.

Přepínač pro změnu absolutní nebo relativní polohy pracuje pomocí objektu *Switch*. Zase je potřeba nastavit obrázky pro stavy zapnuto a vypnuto. Do kolonky *Process* znovu vytvoříme interní tag, aby přepínač fungoval tak, jak má. Vytvoříme tag *MDIrelabs2*, který bude mít adresu DB2648 DBD4096 s datovým typem DWord.

Pro událost *Switch On* přidáme funkci *SetValue* pro tag *MDIrelabs2* a to na hodnotu 65536 a pro *Switch off* na hodnotu 0.

Pro spuštění přímého zadávání polohy, vytvoříme přepínač, kde bude potřeba zase použít interní tag. Vytvoříme tag *MDIzapvyp2* který bude mít adresu DB2098 DBW4097 s datovým typem Word. V událostech budeme tentokrát měnit bit 5 toho tagu.

Pro zobrazení aktuální polohy použijeme pole (IO field) a vložíme do něj tag *actposition2*, který má adresu DB2521 DBD4096 s datovým typem DInt.



Obr. 20: Osa v polohové vazbě

6.2.5 Čtvrtá obrazovka (JOG)

Tato obrazovka bude obsahovat podobné prvky jako předchozí a to: zapnutí a vypnutí osy, signalizace zapnutí a vypnutí osy, signalizace nastavení nulového bodu a zobrazení aktuální pozice.

Tlačítko nast. Referencni bod

Tlačítko pro nastavení referenčního bodu, nastaví aktuální pozici jako referenční. Je použit tag *setrefpoint2*, který má adresu DB2596 DBD4096 s datovým typem DWord. Při stisknutí změní tlačítko hodnotu tohoto tagu na 65536 a při uvolnění zpátky na 0.

Signalizace pohybu osy

Signalizace je vytvořena jako ostatní signalizační „diody“. V tomto případě je použit tag DB2683 DBW4096. V tomto stavovém slově nás zajímají bity 4 (pohyb v pravo) a 5 (pohyb vlevo).

Nastavení kroku

Krokem je myšleno o kolik se motor pootočí, stiskneme-li tlačítko pro zapnutí Jogu. Krok může být kladný i záporný, zaleží, na jakou stranu chceme motorem pootáčet. Pole pro zadávání kroku jsou dvě, protože máme dva typy Jogu. Pole jsou typu IO field a jsou použity tagy *jogrych2.1* a *jogrych2.2*. Oba tagy jsou typu Int, jeden má adresu DB2587 DBW4096 a druhý DB2588 DBW4096.

Nastavení rychlosti

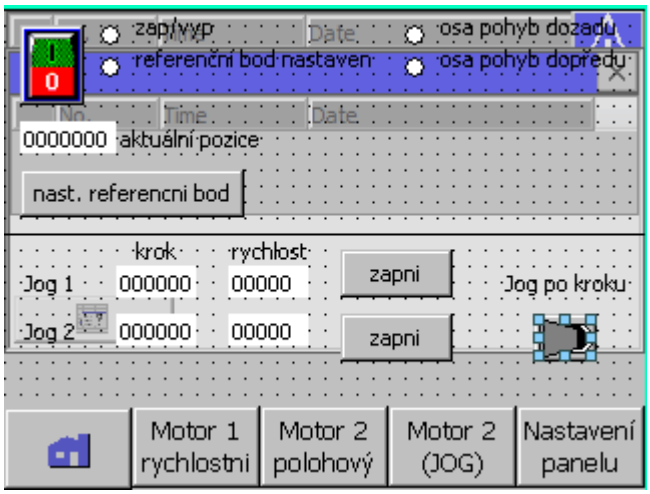
Pro nastavení rychlosti máme dva tagy typu Int. Jeden má adresu DB2585 DBW4096 a druhý DB2586 DBW4096. Pole pro zadávání rychlosti jsou vytvořeny stejně jako pole pro zadávání kroku.

Tlačítka pro zapnutí Jogu

Tlačítka využívají bit 3 a 4 v tagu *joging2*, který má adresu DB2098 DBW4097.

Přepínač Jog po kroku

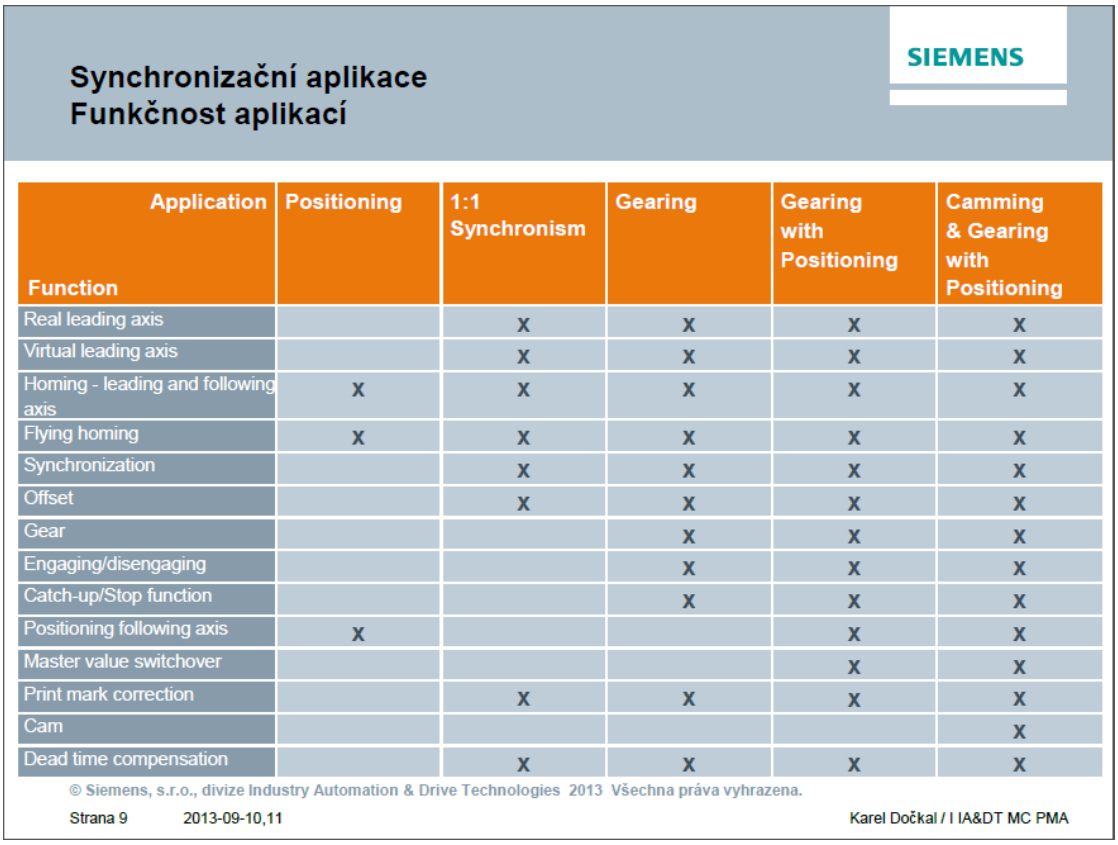
Přepínač slouží k změně režimu jogování. Buďto stisknutím tlačítka zapni, se osa pootočí o jeden krok o dané velikosti a rychlosti, nebo se pohybuje danou rychlostí po dobu, co je tlačítko stisknuté. Aby tlačítko fungovalo správně, je potřeba znovu vytvořit interní tag. Do událostí *Switch On* a *Switch Off* vložíme tag *joging2* a budeme měnit hodnotu bitu 14.



Obr. 21: Osa v polohové vazbě a funkce JOG

7 Synchronní aplikace S120 s DCC

V měničích SINAMICS S120 lze vytvářet vlastní uživatelské aplikace pomocí programu DCC (Drive Control Chart). Lze vkládat logické, aritmetické, regulační i speciální technologické bloky. Součástí DCC je mimo jiné i sada funkcí pro řešení naviječky přímo v měničích. DCC je vhodné jak k naprogramování rychlých regulačních algoritmů přímo v měničích u rozsáhlých technologických celků, tak i k realizaci jednoduché logiky u malých strojů, kde potom není nutno použít žádný nadřazený řídící systém. SINAMICS DCC je první etapa pokročilých technologických funkcí a slouží jako doplněk k provozu STARTER. Cílem je nalezení levného automatizačního řešení pro jednoduché stroje. Zákazníci Siemens si mohou bezplatně stáhnout synchronizační a polohovací bloky z *Industry Online Portalu*, nebo mohou přes regionální zastoupení požádat o vývoj specifických zákaznických bloků. [7]



Synchronizační aplikace
Funkčnost aplikací

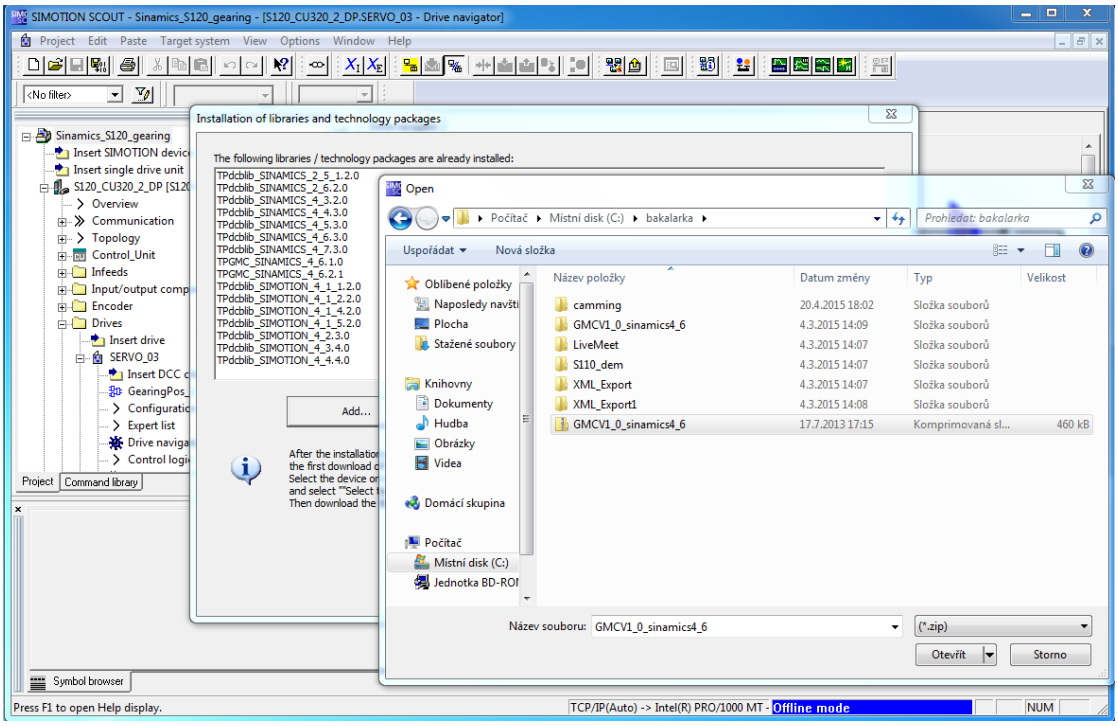
SIEMENS

Application / Function	Positioning	1:1 Synchronism	Gearing	Gearing with Positioning	Camming & Gearing with Positioning
Real leading axis		x	x	x	x
Virtual leading axis		x	x	x	x
Homing - leading and following axis	x	x	x	x	x
Flying homing	x	x	x	x	x
Synchronization		x	x	x	x
Offset		x	x	x	x
Gear			x	x	x
Engaging/disengaging			x	x	x
Catch-up/Stop function			x	x	x
Positioning following axis	x			x	x
Master value switchover				x	x
Print mark correction		x	x	x	x
Cam					x
Dead time compensation		x	x	x	x

Obr. 22: Přehled synchronizačních aplikací a jejich funkčnost [8]

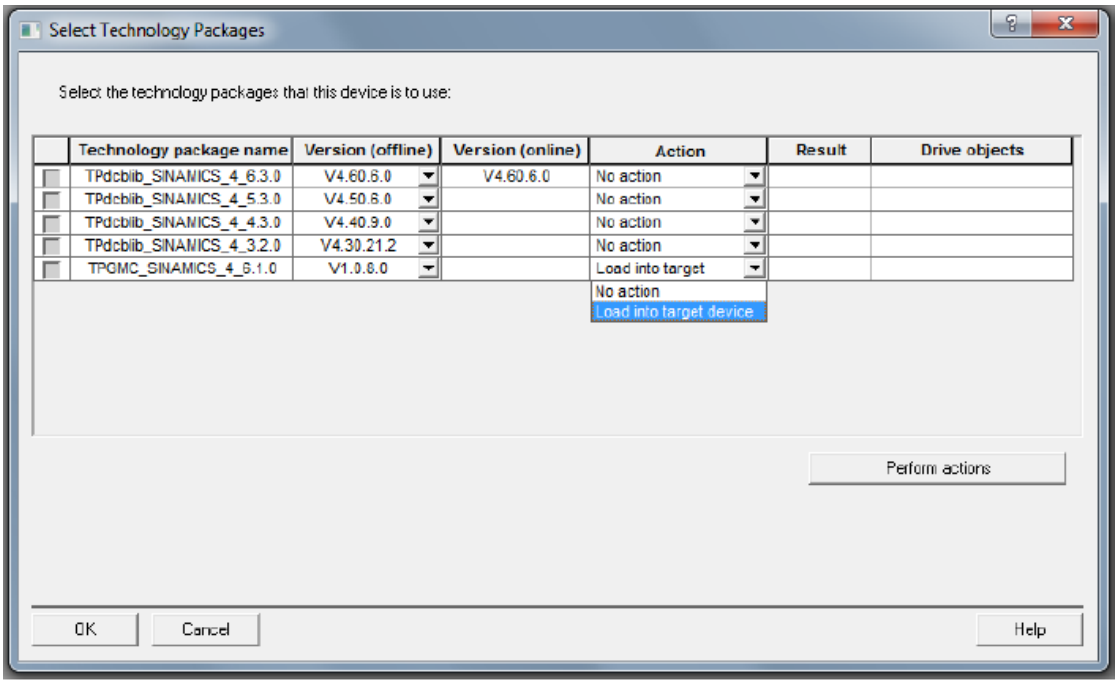
7.1 Implementace knihovny do STARTER

Abychom mohli použít Drive Control Chart, je potřeba do aplikace STARTER implementovat knihovnu s bloky. Nejdříve je zapotřebí ukončit všechny projekty, poté kliknout na *Options* a vybrat *Installation of libraries and technology packages*. Pak pomocí tlačítka *Add..* najít a přidat knihovnu, která je ke stažení na stránkách Siemens.



Obr. 23: Přidání knihovny

Jakmile přidáme knihovnu tak se ihned spustí instalace, která může trvat i několik minut. Po doběhnutí instalalce je potřeba jít do režimu online, tedy připojit se k řídící jednotce. Až budeme připojeni tak klikneme pravým tlačítkem na řídící jednotku a zvolíme *Select technology packages*…. Pro nově nainstalovanou knihovnu vybereme akci *Load into target device* a stiskneme *Perform actions*.

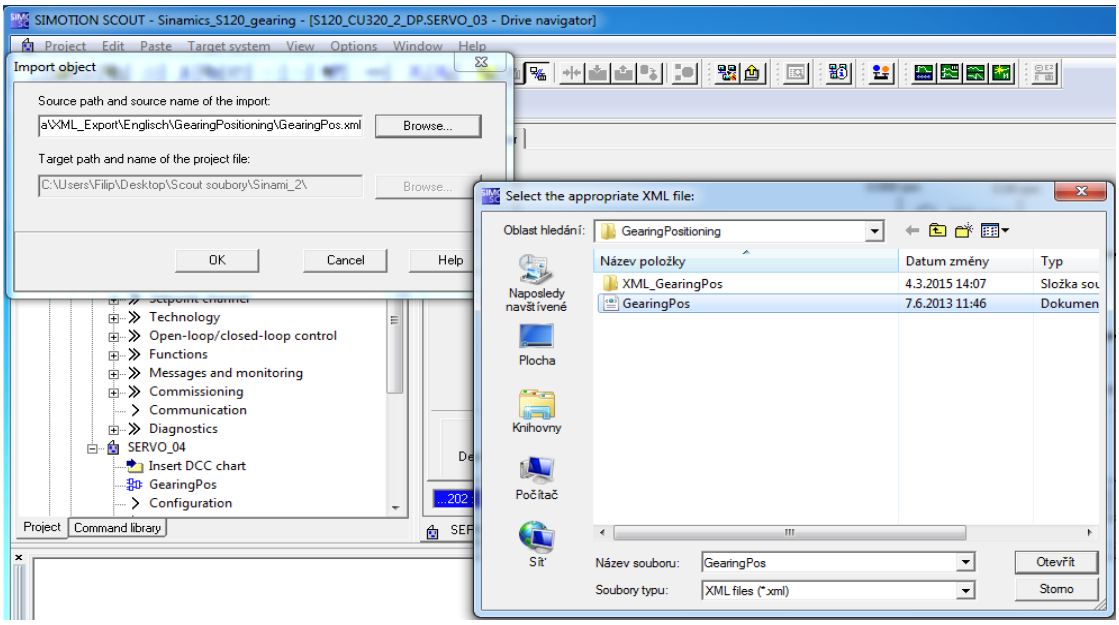


Obr. 24: Nahrání knihovny do cílového zařízení

Tímto máme nainstalovanou knihovnu v aplikaci STARTER a i v cílovém zařízení.

7.2 Konfigurace motorů a implementace DCC

V offline režimu vybereme motor který chceme konfigurovat a v záložce *configuration* klikneme na *Function modules/technology packages..*, kde zaškrtneme pozicování (Cl-loop pos ctrl). Pak klikneme pravým tlačítkem na motor a přejedem na *Expert* a vybereme *Import object*. Poté je potřeba vybrat a implementovat DCC chart, které jsou k dostání na stránkách Siemens. Až implementace proběhne, tak najdeme náš DCC chart v našem stromovém adresáři pod záložkou daného motoru, do kterého byla implementace provedena. Pak už stačí jen uložit a zkompilovat projekt a propojení se základním systémem bude hotovo.



Obr. 25: Implementace DCC chart

7.3 Použití aplikace DCC chart

Všechny funkce aplikace lze uvést do provozu pomocí *Expert Listu*, který obsahuje všechny parametry, kterými lze pohon ovládat. My se budeme zabývat tzv. „Higher-level control“, neboli řízením vyšší úrovně. Zde vyvedeme parametry do ovládacího panelu, abychom mohli řídit synchronizační aplikaci. Postup je identický jako v kapitole[6.2].

7.4 Programování vizualizace pro řízení synchronizace

Založíme si nový projekt, který nastavíme podobným způsobem jako v kapitole[6.2], rozdílem bude, že zde si vystačíme se třemi obrazovkami. Projekt tedy bude obsahovat jednu *Start Screen* a přidáme dvě *Section Screen*.

7.4.1 První obrazovka (Start Screen)

První obrazovka bude obsahovat zapnutí měniče, zapnutí obou motorů a potvrzení chyby pro oba motory. Tyto přepínače a tlačítka jsme již vytvářeli, viz kapitola[6.2.2].

Obr. 26: Start screen

7.4.2 Druhá obrazovka (spuštění virtuální osy)

Pro virtuální osu budeme chtít zadávat rychlost otáčení, náběžnou rampu a doběhovou rampu. Sledovat budeme chtít aktuální rychlost a aktuální polohu.

Zapnutí a vypnutí virtuální osy

Pro zapnutí virtuální osy použijeme z knihovny přepínač (switch), kterému přiřadíme obrázky pro zapnuto a vypnuto. Vytvoříme tag, který bude měnit hodnotu parametru p21801, což je tag který byl vytvořen v rámci implementace DCC. Tag ponese název *zapvyp_virtual* a má adresu DB21801 DBD4096 s datovým typem DWord. Pro události přepínače *Switch On* nastavíme *SetValue* na hodnotu 65536 a pro *Switch Off* na hodnotu 0.

Zadání a měření virtuální rychlosti

Z knihovny použijeme IO Field, v jednom případě nastavíme pole na Input/Output abychom mohli zadávat rychlost, ve druhém případě můžeme nastavení nechat. Pro zadání rychlosti vytvoříme tag, který se bude jmenovat *fixedfrek_virtual* a má adresu DB21806 DBD4096 s datovým typem Real. Pro meření rychlosti vytvoříme tag *actspeed_virtual* s adresou DB21861 DBD4096 a datovým typem Real.

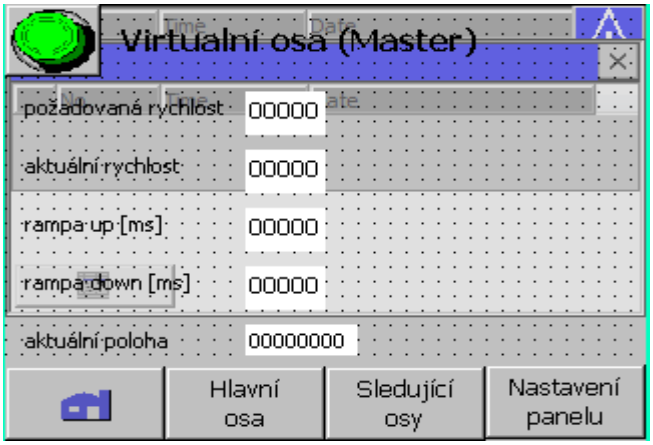
Náběžná a doběhová rampa virtuální osy

Pole budou stejná jako u zadávání rychlosti, změníme jim jen tagy. První tag pro náběžnou rampu se bude jmenovat *rampup_virtual* a má adresu DB21810

DBD4096 s datovým typem Real. Druhý tag bude *rampdown_virtual* pro doběhovou rampu a má adresu DB21811 DBD4096 s datovým typem Real.

Zobrazení aktuální polohy virtuální osy

Vytvoříme stejné pole jako pro měření rychlosti. Tag pro toto pole bude mít název *actposition_virtual* s adresou DB21860 DBD4096 s datovým typem Real.



Obr. 27: Virtuální osa

7.4.3 Třetí obrazovka (nastavení sledujících os)

Pro nastavení sledujících os bude použit převod, který umožnuje v jakém poměru se budou sledující osy točit vůči virtuální ose. Pro přehlednost zobrazíme rychlost a polohu sledujících os. Dále přidáme kontrolky, zda jsou motory zapnuty, jelikož není potřeba mít spuštěné obě osy, aby mohla synchronizace proběhnout.

Signalizace zapnutí/vypnutí os

Signalizační kontrolky vytvoříme stejně jako v kapitole[6.2.4]. Vytvořený tag pro první motor bude mít název *diagmot1* s adresou DB898 DBB3072. Tag pro druhý motor bude *diagmot2* a bude mít adresu DB898 DBB4096. Oba tagy budou typu Byte.

Sledování rychlosti a polohy

Máme dva motory, tedy je potřeba mít čtyři pole, které budou typu IO Field. Pole slouží akorát ke čtení, tedy není potřeba jim nastavovat *mode Input/Output*. Pro čtení rychlosti vytvoříme tag *actspeed1* pro první osu a *actspeed2* pro druhou osu. Tag *actspeed1* má adresu DB63 DBD3072 a tag *actspeed2* DB63 DBD4096. Pro čtení

polohy vytvoříme *actposition1* pro první osu a *actposition2* pro druhou osu, kde *actposition1* bude mít adresu DB2521 DBD3072 a *actposition2* DB2521 DBD4096.

Nastavení převodu

Pro nastavení převodu je potřeba vytvořit dvě pole pro každou osu, kde první pole bude zastávat funkci čitatele a druhé jmenovatele. Čitatel a jmenovatel pro nastavení převodu první osy budou mít tag s názvem gear1 a gear1_0.Kde gear1 zastává funkci čitatele a má adresu DB22301 DBD3072, a gear1_0 DB22302 DBD3072. Pro druhou osu budou tagy s názvem gear2 a gear2_0 s adresami DB22301 DBD4096 a DB22302 DBD4096.

Obr. 28: Sledující osy

Obr. 29: synchronizační funkce se změnou převodu v programu trace

Závěr

Cílem práce bylo seznámit se s problematikou řízení motoru frekvenčním měničem. Představili jsme si základní komponenty řídícího systému pohonu (frekvenční měnič a příslušný software, vizualizační panel) používané v automatizačním průmyslu. Zaměřila se na produkty firmy Siemens, která je jedním z předních výrobců automatizační techniky.

V praktické části bylo na reálné soustavě demonstrováno nastavení měniče pomocí parametrů, návrh vizualizace na panelu HMI a propojení vizualizačního panelu s frekvenčním měničem. Díky tomu je v praxi možná jednoduchá a přehledná obsluha stroje, robota či jiného zařízení.

V textové části práce je rozebráno základní nastavení a uvedení do provozu. To proběhlo pomocí softwaru Starter/Scout. Byla popsána filosofie programu a také konkrétně předvedeno, jak nastavit jednotlivé funkce pohonu pomocí parametrů a také bylo vysvětleno, že cokoliv se v měniči stane, tak se projeví v jednom nebo více parametrech. Jak jsme si ukázali, změnou příslušné sady parametrů lze ovládat a měnit stav měniče, čehož jsme využili v programování vizualizace. Bylo ukázáno jak přistupovat k jednotlivým parametrům přes WinCC a tím ovládat měnič nebo naopak z něj získávat informace. Po přečtení této části by měl být čtenář schopen vytvořit ve WinCC jakoukoliv funkci, kterou měnič umí.

Ve druhé části jsme se zabývali synchronizační knihovnou. Zde jsme si vyzkoušeli jak tuto knihovnu implementovat do programu Starter/Scout a následně jak vytvořit vizualizaci, aby bylo možné řídit synchronizaci z HMI panelu.

Seznam použité literatury

[1] SIEMENS. 2011. *Projekční manuál Sinamics* [online]. [cit. 2015-05-05]. Dostupné z: http://www1.siemens.cz/ad/current/index.php?ctxnh=507f00e632&ctxp=doc_manualy

[2] SIEMENS. 2009. *Katalog měničů řady SINAMICS S120 a SINAMICS S150* [online]. [cit. 2015-05-05]. Dostupné z: http://www1.siemens.cz/ad/current/index.php?ctxnh=507f00e632&ctxp=doc_katalogy

[3] SIEMENS. 2009. *SINAMICS S120 Commissioning Manual* [online]. [cit. 2015-05-06]. Dostupné z: https://support.industry.siemens.com/cs/document/26547069?dti=0&lc=en-WW

[4] SIEMENS. 2013. *Prospekt SINAMICS a SIMOTICS* [online]. [cit. 2015-05-06]. Dostupné z: http://www1.siemens.cz/ad/current/index.php?vw=0&ctxnh=507f00e632&ctxp=doc_prospekty

[5] *Dokumentace k programu WinCC Flexible* [online]. 2011. [cit. 2015-05-06]. Dostupné z: http://www.relko.cz/docs?siemens%2FAD3/

[6] *Ovládací panely TP 177A, TP 177B, OP 177B* [online]. 2008. [cit. 2015-05-06]. Dostupné z: http://www.elintosprekyba.lt/library/files/hmi_TP177a_tp177b_op177b_operating_instructions_en_US_en-US.pdf

[7] SIEMENS. 2010. *SINAMICS Drive Control Chart (SINAMICS DCC)* [online]. [cit. 2015-05-07]. Dostupné z: https://mall.industry.siemens.com/mall/cs/cz/Catalog/Products/10256155?tree=CatalogTree#Overview

[8] SIEMENS. 2010. *Positioning and synchronism with SINAMICS* [online]. [cit. 2015-05-07]. Dostupné z: https://support.industry.siemens.com/cs/document/103471886?dti=0&lc=en-CN

[9] SIEMENS. 2015. *Průmyslové elektrické pohony* [online]. [cit. 2015-05-07]. Dostupné z: https://www.cee.siemens.com/web/cz/cz/corporate/portal/home/produkty_a_sluzby/IADT/tia_na_dosah/Pages/TIAnadosah.aspx#

[10] Rydlo P.: Řízení elektrických střídavých pohonů. Skriptum, FM TUL, 2007, ISBN 978-80-7372-223-4.

[11] Souček P.: Servomechanismy ve výrobních strojích, Vydavatelství ČVUT Praha, 2004.