KATEDRA ELEKTRICKÝCH
POHONŮ A TRAKCE

ČESKÉ VYSOKÉ UČENÍ
TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

NAVÍJEČKA ŘÍZENÁ POMOCÍ SIMATIC S1500

DIPLOMOVÁ PRÁCE

KVĚTEN 2021

ONDŘEJ
KALÁT

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kalát** Jméno: **Ondřej** Osobní číslo: **457239**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra elektrických pohonů a trakce**

Studijní program: **Elektrotechnika, energetika a management**

Specializace: **Elektrické pohony**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Navíječka řízená pomocí SIMATIC S1500

Název diplomové práce anglicky:

Winder Controlled by SIMATIC S1500

Pokyny pro vypracování:

1) Seznamte se s logickým automatem SIMATIC S 1500
2) Navrhněte logické řízení pro navíječku cívek
3) Navržené logické řízení implementujete na kitu s PLC a dotykovým panelem
4) Řízení doplňte o vizulaizaci na HMI panelu

Seznam doporučené literatury:

[1] Weidauer J., Messer R. Electrical Drives, Publics Erlangen, 2014
[2] SCE Training Curriculum. Siemens AG, 2016
[3] Durry B. The Control Techniques Drives and Controls Handbook 2nd ed., IeT, 2009

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jan Bauer, Ph.D., katedra elektrických pohonů a trakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **24.01.2021** Termín odevzdání diplomové práce: **21.05.2021**

Platnost zadání diplomové práce: **30.09.2022**

Ing. Jan Bauer, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu práce Ing. Janu Bauerovi, Ph.D. za odborné vedení a poučné konzultace. Konzultace k praktické části práce mi poskytli odborníci z firmy Siemens, Ing. Martin Kozák a Ing. Daniel Knotek, za což jim také děkuji. Další poděkování patří samotné firmě Siemens s.r.o. za zapůjčení softwarového vybavení pro praktickou část práce. Poděkovat také chci své rodině za podporu během psaní práce.

PROHLÁŠENÍ

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 12. května 2021

.....

ABSTRAKT

Práce v teoretické části zpracovává základní problematiku pohonů používaných v navíjecích aplikacích. Obsahuje také popis standardizované aplikace pro navíječky s vedoucí osou od firmy Siemens. V praktické části práce je detailně popsán postup při zprovoznění této aplikace, je upozorněno na několik jejích nevýhod a omezení spolu s řešením problémů při uvádění do provozu. V závěru teoretické části práce jsou zobrazeny nasimulované průběhy z běhu aplikace a poznatky z testování programu na reálném stroji.

Klíčová slova: frekvenční měnič, PLC, HMI, IRT komunikace, servomotor, navíječka, enkodér, centrální řízení, TIA portál, simulace, vzdálené řízení

ABSTRACT

This thesis describes in the theoretical part specific issues and requirements for drives used in winding applications. It also contains description of a standardized Siemens application for winders with traversing drive. The practical part describes in detail commissioning of this application, there are pointed out several disadvantages and limitations of this application along with problems during commissioning and their solutions. There are Traces of simulation run displayed at the end of the theoretical part as well as the findings from testing the program on a real machine.

Keywords: frequency converter, PLC, HMI, IRT communication, servo motor, winder, encoder, central control, TIA portal, simulation, remote control

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Frekvenční měniče (dále jen FM) poskytují oproti jednodušším způsobům řízení elektrických motorů velmi velkou flexibilitu, co se požadavků na aplikaci týče. Už i samotná řídicí jednotka (dále jen CU) jednoduchého FM je poměrně výkonný průmyslový počítač, který dokáže řídit pohon s poměrně velkou dynamikou, a to velmi přesně. Toho se, i přes vyšší pořizovací náklady FM, velmi často využívá u aplikací, kde je vyžadována vysoká přesnost, synchronizace s jiným pohonem, anebo práce s průběžně se měnícími parametry, kde je FM prakticky nezastupitelný. [2]

V dnešní době digitalizace a automatizace, tedy s nástupem tzv. Průmyslu 4.0, je však kladen důraz i na další aspekty takovýchto úloh. Moderní aplikace jsou široce standardizovány, od používaných programovacích jazyků, přes strukturu proměnných až po komunikační protokoly. Místo ovládacích přepínačů a kontrolek umožňují návrh interaktivních panelů s nápovědami pro obsluhu, jejím managementem např. pro administrátory a servis, a možnost vše řídit vzdáleně a centrálně, třeba i několik desítek takových aplikací. [1] Vrcholem této technologie jsou rozsáhlé simulace, které od jednoduchého simulování digitálních vstupů a výstupů (dále jen DI/DO) dnes přechází na simulaci ovládacího rozhraní panelů (dále jen HMI), simulace programovatelných logických automatů (dále jen PLC) až po tzv. digitální dvojče (digital twin), kdy je celý pohon včetně připojené technologie sestaven, uveden do provozu a otestován pouze v simulacích. [27]

V této práci je cílem dle těchto moderních postupů popsat teorii a uvést do provozu pohon s komponenty od firmy Siemens s.r.o. v aplikaci navíječky s vedoucí osou. Jde tedy o velmi přesnou synchronizaci lineární a otáčivé osy, kde se vyskytují neustále se za běhu měnicí parametry, např. průměr navíjené cívky. Jsou zde použity servomotory (2.1.3) [21] poháněné FM (2.1.2), [20] pro možnost využití více funkcí a flexibility výpočetního výkonu, je zde využito centrální řízení pohonu, a výpočet řízení pohybu tedy probíhá v PLC (2.1.1) [17]. Sestava je doplněna o základní zobrazovací panel (2.1.4) [22] pro jednoduché a intuitivní interaktivní rozhraní.

Projekt je postaven na standardizované aplikaci od firmy Siemens (1.2). [3] Cílem těchto vzorových projektů je poskytnout zákazníkovi základ, na kterém může postavit svou aplikaci a přizpůsobit si ji přesně dle svých představ. [14] Aplikace také umožňuje využít všech výhod FM s centrálním řízením (1.2.1.1) [6] a přidává další funkce (1.2.2) jako např. Spike (1.2.2.3), Stroke (1.2.2.4), nebo automatický výpočet průměru cívky (1.2.2.6), které by klasickým způsobem bylo složité naprogramovat, pokud by to vůbec bylo v rámci CU možné. Struktura parametrů je navíc standardizovaná, což usnadňuje práci uživatelům při uvádění do provozu více těchto aplikací.

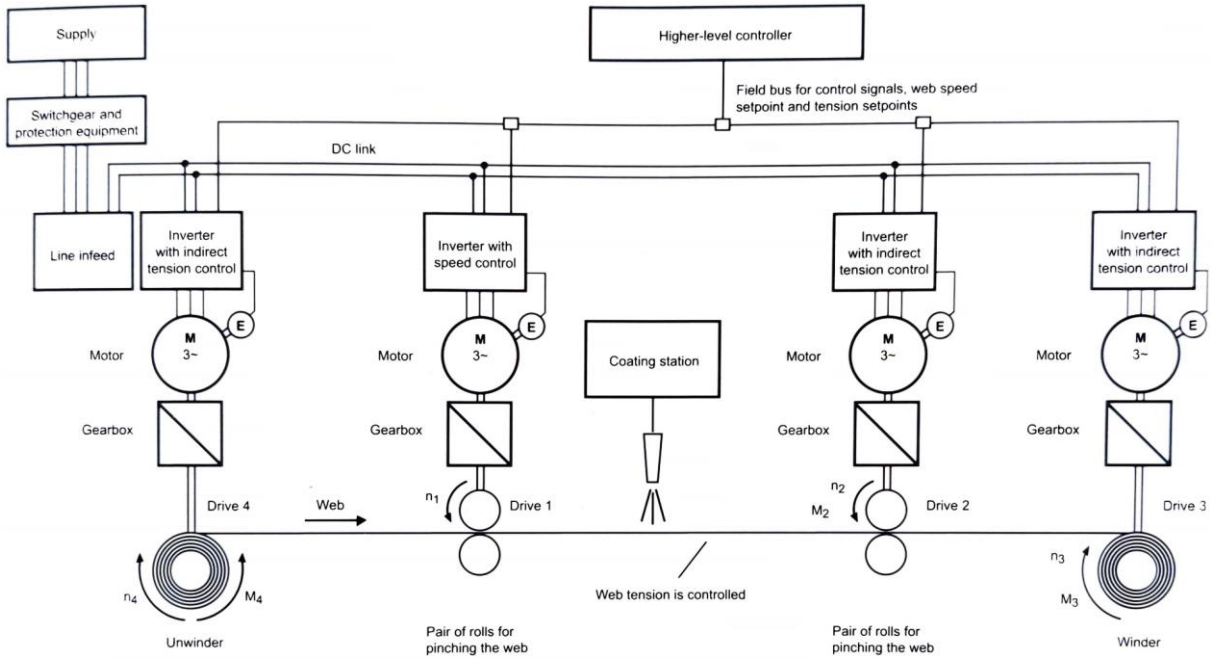
Mimo výhody aplikace je zde upozorněno i na několik překážek a nevýhod (1.2), které z použití této aplikace plynou. Jako taková sice nemá nikdy sloužit jako program připravený na nahrání a okamžité spuštění, vždy je potřeba od zákazníka zásah do kódu. V několika případech ale způsob naprogramování vzorového projektu práci spíše komplikuje, např. nastavením módu navíjení (2.2.2.4). Parametry, které lze jednoduše inicializovat v uživatelském rozhraní programovacího software (dále jen SW), je nutné měnit přímo v základní struktuře kódu (2.2.2.3) a použití vybavenějšího Comfort panelu, jehož zjednodušená verze Basic panel použitá v této práci nemá všechny jeho funkce, [5] může způsobit, že i když program nehlásí žádnou chybu, nelze ho zkompilovat a spustit (2.2.2.6, 2.2.3). Součástí práce je i naprogramování externího enkodéru pro odměřování odvinutého materiálu (2.2.2.5), na což aplikace není nijak připravena. Všechny tyto a další překážky a nevýhody, včetně jejich řešení jsou v textu detailně popsány.

Celý projekt je připraven na testování v simulacích (2.3.1), s reálným hardware (dále jen HW) na místě (2.3.2.1), provozován a uváděn do provozu může být ale také vzdáleně (2.3.2.1). Úzká integrace HW a SW firmy Siemens umožňuje připojení přes router k HW tak, jako by byl připojen klasickým PROFINET připojením (1.3). [28] Je zde tedy cílem, aby se aplikace z hlediska praktického aplikování dále blížila moderním standardům popsaným dříve a zároveň dokázala, že je možné i takto poměrně složitý pohon spouštět bez přítomnosti obsluhy např. z důvodu globální pandemie a z toho vyplývajícími distančními omezeními.

KAPITOLA 1: TEORETICKÁ ČÁST

Typické aplikace pro navíječky jsou takové kde se mění pozice materiálu, tedy např. navíjení (rolování), tah skrze jiný technologický celek, nebo je materiál natahován. V kombinaci s dalšími technologickými celky lze pak měnit např. povrchovou úpravu navíjeného materiálu, tedy proces lakování, impregnace, nebo lepení. Příklad takové aplikace je na Obr. 1-1. Navíjený materiál je obvykle odvíjen ze zásobníkové cívky (Drive 4) a veden skrze systém válců (Drive 1, 2), které zajišťují potřebné pnutí materiálu. Několik, ale ne nutně všechny válce a cívky mohou být poháněné a zvyšovat tak tažný moment. Poté, co navíjený materiál projde všemi technologickými celky (Coating station) je navinut zpět na cívku (Drive 3). Všechny tyto pohony spolu komunikují pomocí průmyslové komunikační směrnice (např. PROFINET, nebo starší PROFIBUS). Pokud je požadováno, aby navíjený materiál byl na cívce uspořádán rovnoměrně, lze ho navádět na přesnou pozici pomocí vedoucí osy (Traverser). Uchycení materiálu na cívce je obvykle realizované třecí silou samotného materiálu pomocí pogumovaných jader cívek. Další možností je mechanické uchycení. Aplikace vyžadující malé síly tahu mohou takto zpracovávat např. plastové pásky, pro zpracování materiálů, jako jsou např. plechy, nebo ocelové dráty jsou potřeba podstatně větší síly a výkony. [1]

Pokud nejde o velmi jednoduchou aplikaci navíjení, kde je poháněna pouze jedna osa např. osa navíjecí cívky, je většinou nutné použít alespoň dvě poháněné osy. Pak lze aplikace dělit podle řízení mechanického napětí navíjeného materiálu. Jednodušší aplikace, jako je např. natahování materiálu, nebo případy, kde je technologie ovládaná přímo obsluhou lze řídit v režimu rychlostní osy (speed-control). Žádaná rychlost se tedy s každou další poháněnou osou zvětšuje, čímž se zvyšuje i mechanické napětí a materiál se natahuje (případně jinak zpracovává). V případě, kdy je naopak nežádoucí změna fyzikálních rozměrů navíjeného materiálu, nebo ho jednoduše natáhnout nelze, je nutné mechanické napětí řídit tak, aby nebylo moc velké (a nedošlo právě k deformaci materiálu), ale ani příliš nízké (pak hrozí nepřesné navíjení, povolování smyček). [1] Zde je více možností, jak pohon řídit, ale nejzásadnější je rozhodnout, jak získáme informaci o velikosti skutečného mechanického napětí. Pokud bude přímo měřeno na technologii, pak jde o přímé řízení tahu. Pokud bude mechanické napětí pouze vypočítáváno, jde o nepřímé řízení tahu. Takové řízení je zobrazeno na Obr. 1-1.

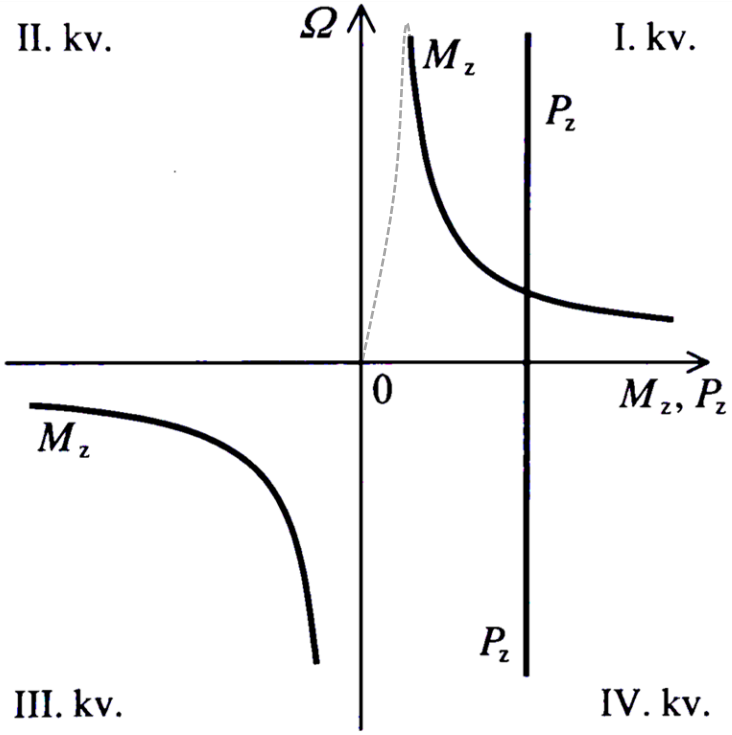


Obr. 1-1 Schéma pohonu s nepřímým řízením tahu [1]

1.1 Požadavky na pohon

Jedním z elementárních parametrů při dimenzování pohonu a jeho přizpůsobování je charakteristika zátěžného momentu dané aplikace. Jde o závislost zátěžného momentu M_z na mechanické úhlové rychlosti stroje Ω. Těchto charakteristik je možné rozlišovat mnoho druhů podle aplikací, typicky jde třeba o velmi častou ventilátorovou charakteristiku. Lze je vyjadřovat analyticky i graficky. [2] Analytické vyjádření je vhodnější pro matematické modely a výpočty, grafické zase srozumitelněji vysvětluje chování M_z v praxi. Na Obr. 1-2 je graficky zobrazena navíječková charakteristika, společně s ní je v grafu vykreslena závislost potřebného výkonu P_z na mechanické úhlové rychlosti Ω.

Charakteristika je vykreslena pro zátěže s konstantním tahem. Navíječka začíná navíjet (při kladné úhlové rychlosti) v bodě nejvyšší Ω a nejmenší M_z. S postupným navíjením materiálu postupně klesá Ω a naopak exponenciálně narůstá M_z až do okamžiku, kdy je cívka navinuta, tedy M_z dosáhne určité požadované hodnoty.



Obr. 1-2 Navíječková charakteristika M_z [2], upraveno

Z charakteristiky je možné vypozorovat, že energii dodanou motorem do mechanismu nelze z mechanismu vzít zpět. To je dané tím, že závislost $M_z = f(\Omega)$ zasahuje do I. a III. kvadrantu. Ve všech případech mají tedy obě veličiny stejné znaménko a výsledný výkon je vždy kladný a vždy se tedy spotřebovává, jak lze vidět ze závislosti $P_z = f(\Omega)$. Jde tedy o tzv. pasivní zátěž, tedy se změnou směru otáčení se změní i smysl zátěžného momentu. V reálných aplikacích má zátěž vždy do určité míry povahu obou typů zátěžného momentu, v tomto případě tedy i aktivní. [2] Samovolné otáčení navíječky by ale více než přímo zátěžným momentem mohlo být způsobeno např. nerovnoměrným navíjením, nebo vyšším tahem jiné části technologie, proto zde není uvažován.

Je důležité, že tato charakteristika je ideální matematický popis, který plně neodpovídá skutečnosti. Kromě již zmíněné aktivní části zátěžného momentu také z Obr. 1-2 vyplývá, že při nulových a velmi nízkých otáčkách roste M_z do nekonečna. Popisuje to skutečnost, že v navíječkových aplikacích musí být vždy přítomno alespoň nějaké mechanické napětí. Prakticky by samozřejmě nebylo možné stroj v takovém pohonu roztočit, tah se při prvním rozběhu uvolňuje a navíjený materiál se utahuje až po dosažení požadovaných otáček stroje. Příklad křivky M_z při rozběhu je do obrázku dokreslen přerušovanou šedou linkou.

1.2 SIMATIC Traverser

Aplikace navíječky včetně bloků LTrav je jedna z knihoven SIMATIC Converting Toolbox [14] poskytované firmou Siemens. Ta je vytvořena tak, aby byla co možná nejvíce univerzální a snadno přizpůsobitelná konkrétní aplikaci zákazníka. Zároveň je standardizovaná napříč dalšími aplikacemi SIMATIC, což zjednodušuje práci programátorům, kteří pracují s podobným systémem PLC bloků. Mimo knihovny Traverser obsahuje např. aplikace pro letmou pilu, frézy, pohony pro tisk a mnohé další. Veškeré používané bloky a funkce jsou open-source, žádné z nich, mimo základní programové organizační bloky (dále jen OB), tedy nejsou know-how firmy. Bloky jsou popsané v manuálu, interně testované a stručně komentované, lze je libovolně upravovat pro optimální funkčnost konkrétní aplikace zákazníka. Další výhodou je, že aplikace standardně pracuje se standardním řídicím rozhraním z knihovny LAxisCtrl [8], a to napříč všemi aplikacemi. Lze tedy skrze jedno rozhraní ovládat stejným způsobem více stejných, nebo i různých aplikací bez nutnosti změny vstupů na každé aplikaci zvlášť. Stejně tak je možné naopak ovládat jednoduše každou aplikaci jednotlivě při uvádění do provozu, např. při testování osy, referencování apod.

Během programování této aplikace se ovšem ukázalo také několik nevýhod použití této standardizované aplikace a rozhodně tak neplatí, že lze projekt vždy jednoduše nasadit na jakoukoli aplikaci. V knihovně LTrav poskytnuté společností Siemens je připraveno několik základních řídicích bloků v jednoduchém grafickém žebříkovém programovacím jazyce LAD. Tyto bloky (převážně funkční bloky, dále jen FB) mají vnitřní strukturu napsanou ve strukturovaném řídicím jazyce SCL, mají ovšem velmi dobře navržené a přehledné rozhraní. Blok má tedy jen několik vstupů a výstupů, veškeré logické a matematické operace se dějí přímo uvnitř bloku. Programátor tak teoreticky ani nemusí znát jejich vnitřní strukturu. Na tyto bloky se odkazuje i přiložený manuál [3], kde je velmi stručně popsáno, jak bloky mezi sebou zapojit. Samotný vzorový projekt ale tyto bloky vůbec nepoužívá a integruje je do celků sestavených z několika FB, kde každý obsahuje několik, nebo jen části bloků poskytnutých v knihovně LTrav. Podrobněji je tato problematika rozebrána v podkapitolách inicializace 2.2.2.3 a hlavní sekvence 2.2.2.4.

Omezením této aplikace také je, že spoléhá na použití sofistikovanějšího SIMATIC Comfort panelu, namísto jednoduššího Basic panelu. Pro účely předvedení všech možností samotné aplikace, i právě těchto panelů je pochopitelné, že byl využit lepší, variabilnější a vybavenější HW. Pokud ale zákazník chce použít Basic panel, aplikace při pouhém překlopení a nahrání do HW nebude fungovat. Tím tento projekt ztrácí významnou výhodu a zákazník ztrácí motivaci tuto aplikaci použít. Tato práce byla vytvářena s Basic panelem KTP400 popsaného v podkapitole 2.1.4 a tyto problémy a jejich řešení byly popsány. Nejzásadnější omezení je takové, že Basic panel nepodporuje přenos všech datových typů, které umožňuje Comfort panel, a který je ve výchozím nastavení přenášen (více v podkapitole 2.2.2.6). Tyto přenášené parametry pak nelze jednoduše upravit. Drobná omezení, která ale také komplikují nasazení Basic panelu jsou např. používání grafických prvků unikátních pro Comfort panel, nebo fakt, že projekt je vytvořen s nejmenší 12" verzí Comfort panelu, přičemž Basic panel se nabízí už od 4" varianty (více v podkapitole 2.2.3).

Další zjištěná nevýhoda souvisí s předchozími body. Komplikované a nejasně propojené programové bloky v programovacím jazyku SCL společně s ne příliš flexibilní možností úpravy přenášených parametrů vedly pravděpodobně k tomu, že některé funkce nastavitelné na HMI vzorového projektu nefungují. Jde pouze o několik parametrů, které nenarušují základní funkčnost programu, např. kontrola integrity zadaných parametrů cívky. Ovšem jeden z parametrů, který ve výchozím programu není možné změnit i když samotný parametr je možné nastavovat je volba Winding mode. Popsaný je v podkapitole 1.2.2, podrobnější popis problému je opět v podkapitole 2.2.2.4.

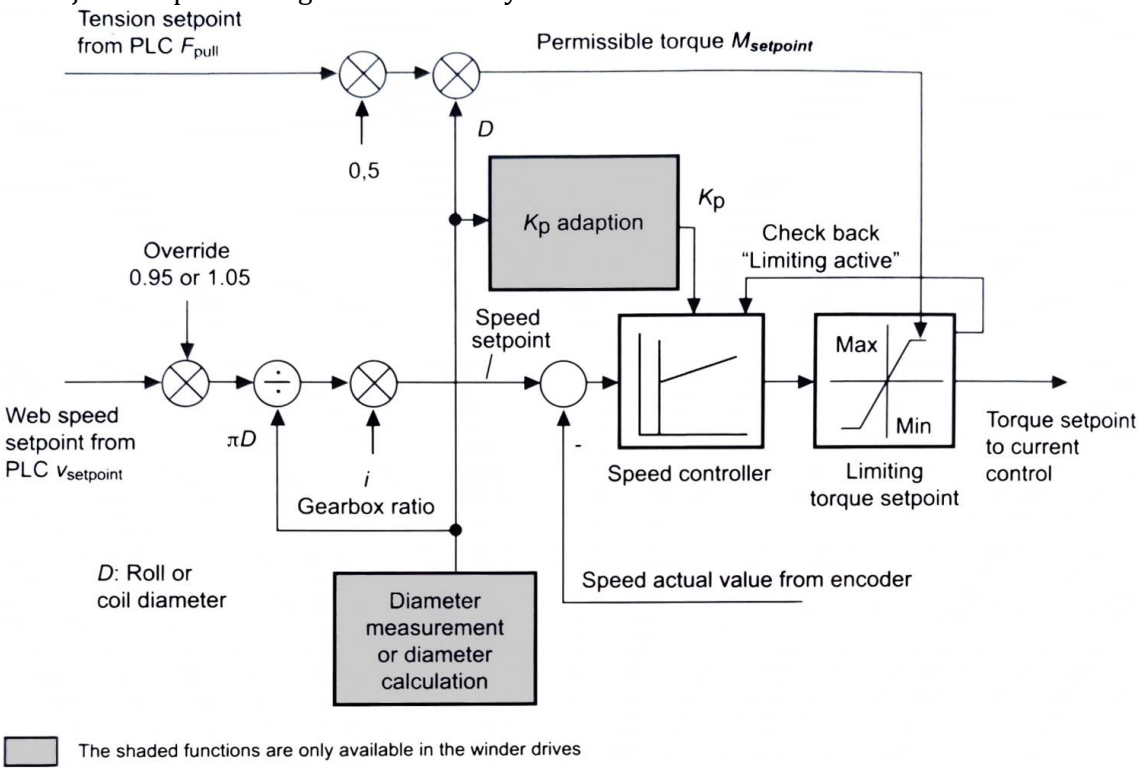
Tyto a několik dalších překážek při uvádění do provozu této aplikace a jejich řešení bude v příslušných podkapitolách popsáno. Aplikace je ke stažení z odkazu [3], nejaktuálnější byla v době psaní této práce verze 1.0.2 ze září roku 2019. Vzorový projekt v TIA portále je vytvořen ve verzi 15, lze ho tedy spustit na této a jakékoliv novější verzi.

1.2.1 Základní funkce

Knihovna SIMATIC traverser je určena pro aplikace, kde je během navíjení vyžadováno přesné umístění navíjeného materiálu na cívku. Typicky tedy dráty, kabely, textil nebo fólie. Byla vytvořena s cílem jednoduchého ovládání tak, že sama vyžaduje pouze nejnutnější informace o mechanice jako jsou data v technologickém objektu (dále jen TO) a tzv. traversing profile, a další parametry se dopočítávají v pozadí. Tyto parametry lze navíc měnit i za běhu a sledovat tak přímo jejich dopad na navíjení. Z hlediska programu jsou osy řízeny tak, aby bylo dosaženo co nejvyšší přesnosti. Reálně však přesnost záleží i na mechanice navíjecí osy, posunovací osy, ale např. i navíjeném materiálu. [3]

Aplikace pracuje s předpokladem nepřímého řízení tahu, reguluje se tedy moment poskytovaný motorem. Regulační schéma používané v aplikaci je na Obr. 1-3. Vstupem regulátoru je požadovaná tažná síla F_{pull}, vynásobená konstantou 0,5. Po vynásobení aktuálním průměrem cívky získá systém maximální možný moment $M_{setpoint}$, který slouží jako horní omezení momentu v bloku omezení „Limiting torque setpoint“ a zároveň jako požadovaný moment. Aktuální průměr cívky se vypočítá v bloku Diameter measurement or diameter calculation, podrobněji je funkce popsaná v podkapitole 1.2.2.6.

Pro samotnou funkčnost udržování konstantního vypočítaného tahu by toto zapojení společně s regulátorem rychlosti stačilo a fungovalo správně. Mělo by však jednu podstatnou nevýhodu ve chvíli, kdy by se navíjený materiál přetrhl. Pak by totiž vypočítaný tah zůstával nulový a stroj by neustále zrychloval se záměrem obnovit tah, přičemž by mohlo dojít k jeho poškození, nebo k poškození technologie. V praxi tedy čistě momentovou regulaci není možné použít a je třeba přidat i regulaci samotné rychlosti.



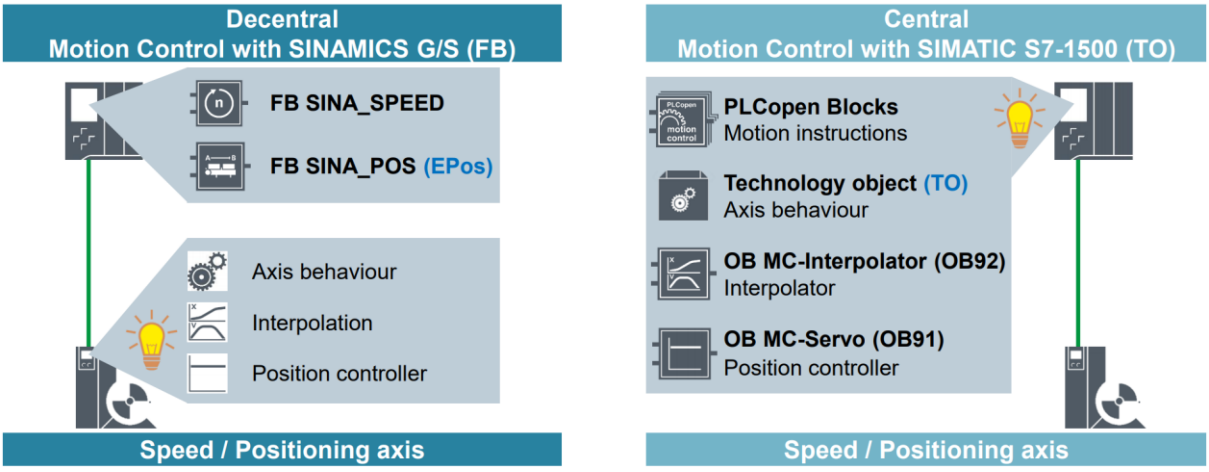
Obr. 1-3 Regulační schéma nepřímého řízení tahu [1]

Do systému je potřeba přivést informaci o žádané rychlosti navíjeného materiálu $v_{setpoint}$. Tato rychlost se následně vynásobí hodnotnou Override, která hodnotu rychlosti omezí podle potřeby. Na Obr. 1-3 je neznačeno rozmezí od 95 do 105 % požadované hodnoty, lze ji ale v aplikaci přizpůsobit. Po přepočtu na úhlovou rychlost, tedy vydělením hodnotou πD kde D je opět průměr cívky vypočítaný funkcí, a započítání případného převodu převodovky už následuje typické schéma rychlostního PID regulátoru. Rozdíl požadované a skutečné hodnoty rychlosti změřené

enkodérem je vstupem do PID regulátoru rychlosti, jehož výstupem je požadovaný moment. Protože je potřeba proporční konstantanu regulátoru K_p v závislosti na průměru cívky měnit, je tato konstanta vypočítána v bloku K_p adaption opět z hodnoty D. Moment je pak omezen blokem „Limiting torque setpoint“. Pohon tak prakticky pracuje v rychlostní regulační smyčce, je ale v ideální případě stále na maximu tohoto omezovače momentu, tedy udržuje požadovaný moment $M_{setpoint}$. [1] Hodnota výsledného momentu pak pokračuje do regulátoru momentotvorné složky proudu.

1.2.1.1 Centrální řízení

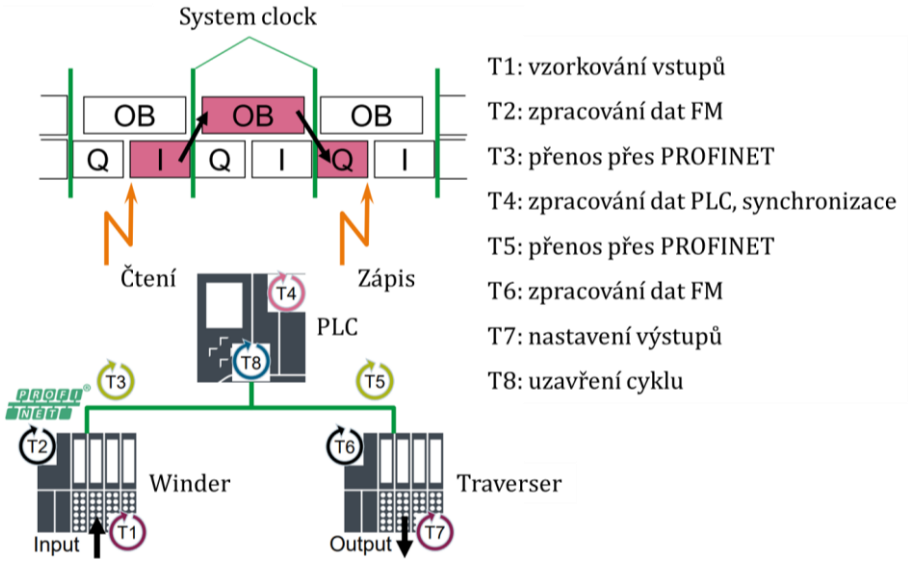
Z hlediska řízení pohonu je tato aplikace koncipována jako centrálně řízená. Řízení pohybu (Motion Control, dále jen MC) tedy zařizuje PLC a FM už pouze vykonává vypočítané povely. Aplikace předpokládá použití PLC S7-1500T s firmware (dále jen FW) V2.5, nebo vyšší. S touto konfigurací je pak možné využívat TO pro polohovou a synchronní osu (a mnohé další) a programování se standardizovanými PLCopen bloky. To zásadně zjednodušuje programování např. není třeba programovat bitovou komunikaci („Bit-set“, „Bit-reset“) a lze využít zabudované diagnostiky, nebo automatického generování alarmů (i do HMI) a jejich resetování. [6] Podrobněji jsou funkce TO společně s jejich konfigurací popsány v podkapitole 2.2.2.1. V případě decentrálního řízení je vypočítávána rychlostní osa v každém FM zvlášť a PLC slouží pouze jako prostředník ke komunikaci. Výhodou takového řízení je, že počet os není určen maximální pamětí a výpočetní výkonem PLC, ale přímo ho určuje počet FM. Z hlediska programování je ovšem potřeba použít komplikovanější funkční bloky. Základní schéma těchto typů řízení je na Obr. 1-4.



Obr. 1-4 Decentrální a centrální řízení [6]

1.2.1.2 IRT komunikace

Komunikace mezi PLC a FM je realizována pomocí cyklické výměny dat. Možnosti jsou dvě – RT a IRT. Zde je použito IRT, tedy izochronní přenos dat s vysokou stabilitou pro časově kritické aplikace, např. právě řízení pohybu (MC). V tomto režimu je zajištěna vysoká rychlost a spolehlivá reakční doba systému, která je založena na skutečnosti, že všechna data jsou poskytována „just-in-time“. [25] V každém programovém cyklu je tak provedena synchronizace přednostně v tzv. IRT paketech a jakýkoliv další zápis, nebo čtení parametrů je upozaděno. Pokud by potřebná doba čtení/zápisu parametrů byla delší než určené maximální okno před další synchronizací, jsou tyto úkony přeskočeny a vykonají se až v dalším programovém cyklu. [15] Jeden programový cyklus je zobrazen na Obr. 1-5. Vstupními parametry FM Winder „Input“ může být myšlen např. pokyn k navíjení, výstupními parametry FM Traverser „Output“ pak může být myšleno např. zahájení pozicování osy traverseru.

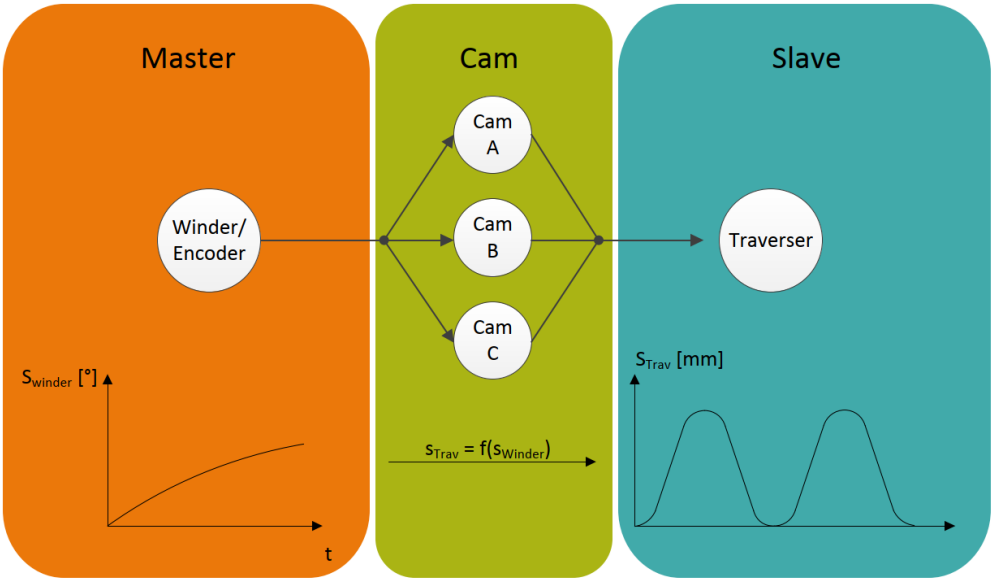


Obr. 1-5 Princip IRT komunikace [15], upraveno

V projektu se dále uplatňuje i acyklické vyčítání/zápis hodnot NRT. Používá se při komunikaci PLC s HMI, nebo k připojení osobního/programovacího počítače (dále jen PC/PG) na síť za účelem diagnostiky. Možnost RT je vhodná pro přenos dat v prioritních ethernetových rámcích (ethernet frames). Každé zařízení má zde jinou dobu aktualizace dat. Tím se může zrychlit odezva pohonu, výrazně se ale zhorší synchronizace dat a pro tuto, a obecně pro všechny aplikace kde je kritická přesnost pohybu se tak nehodí.

1.2.1.3 Struktura programu

Je důležité rozlišovat hierarchii HW a programových celků. Z hlediska HW byla struktura popsána v podkapitole 1.2.1.1 a z principu centrálního řízení je tedy Master PLC, funkce Slave plní FM. Samotný program v PLC má ale v sobě zakomponovanou také tuto strukturu. Tvoří jí samostatné TO, v základním programu jsou potřeba 3 typy. Posunovací osa (Traverser) plní funkci Slave a je synchronizována k řídicí navíjecí ose (Winder). Ta je tedy Master z hlediska programových celků. Vazba mezi nimi je virtuální osa (Line), na Obr. 1-6 označená jako funkce společně s blokem vačka Cam. Ta stojí mimo tuto strukturu a vykonává se nezávisle na ostatních TO. V programu jsou potřeba celkem tři synchronizační body – jeden na začátek cívky, druhý na konec a třetí je pohyblivý dle potřeby. Každý tento bod má vlastí TO. Mimo výchozí program je zde přidán ještě TO externího enkodéru, jeho nastavení, i nastavení ostatních TO je popsáno v podkapitole 2.2.2.1.



Obr. 1-6 Blokové schéma aplikace [3]

Schéma reálného HW použitého v praktické části společně se zobrazením typů připojení je na Obr. 2-1.

1.2.2 Popis funkcí

Pro samotnou synchronizaci dvou os navíjení by nemělo velký smysl připravovat rozsáhlou aplikaci. Naopak, typická reálná aplikace navíječky většinou vyžaduje větší přizpůsobení, než jen synchronizaci dvou polohových os. Jednou z již zmiňovaných výhod knihovny LTrav je, že umožňuje aplikaci přizpůsobit více situacím a požadavkům zákazníka. Mimo základní synchronizaci tak aplikace nabízí další funkce, které upravují základní způsob navíjení. Před jejich popisem je ale nutné definovat základní pojmy používané ve vzorovém programu. Tyto základní parametry jsou v knihovně důsledně používané ve všech navazujících funkcích a nepředpokládá se, že se budou měnit. V dalších podkapitolách jsou popsané funkce, jejichž použití je volitelné (nebo je potřeba vybrat si jednu z více možností) a jejichž vlastnosti lze poměrně jednoduše v programu měnit.

Zde, i ve zbytku práce budou tyto parametry označovány anglickými názvy dle manuálu [3] s velkým počátečním písmenem, např. Winding mode. V případě proměnných z programu, které jsou používané především v praktické části, budou opět uvedeny anglicky standardním programátorským formátem proměnných v jazyce C a pozměněným formátováním, např. `windingMode`. Často se zde objevují jména proměnných s tečkou na začátku. To značí, že nejde o celou cestu k parametru a jde pouze o zkratku, např. `.windingMode` má v programu definovaný název `"LTrav_Data".travInput["HMI".selectedIndex].windingMode`. Tyto názvy nejsou překládány do českého jazyka, komplikovalo by to případné reference k manuálu, nebo k samotnému programu. Zachováním těchto názvů je pak možné v případě nutnosti podrobnějších informací hledat v manuálech/programu rovnou požadovaný parametr a není nutné ho opět překládat zpět z češtiny do angličtiny.

- Traversing cycle

  Osa traverseru je definována úsečkou AB, kde A je počáteční a zároveň koncová poloha osy Traverser. Bodem A je vždy myšlen levý konec osy a vždy má menší hodnotu, než bod B. Jeden cyklus tedy značí přesun osy Traverser z bodu A do bodu B a poté zpět do bodu A. Graficky je cívka znázorněna na Obr. 1-9.

- Motion profile

  Označuje synchronizovaný pohyb absolutní polohy osy Traverser a relativní polohy (natočení) cívky. Hodnoty pro obě osy se počítají v pozadí, jsou potřeba minimálně čtyři vstupní parametry: Acceleration angle, Waiting angle, Winding step a Displacement angle.

- Acceleration angle [°]

  Úhel, který cívka urazí, zatímco osa Traverser zrychluje nebo zpomaluje. Velký úhel zajišťuje plynulejší rozjezd/brzdění osy Traverser, ale snižuje rychlost navíjení.

- Waiting angle [°]

  Úhel, který cívka urazí, zatímco osa Traverser stojí na koncovém bodě A, nebo B. Alespoň malý úhel je vhodný pro stabilní vytvoření okrajových bodů vinuté cívky.

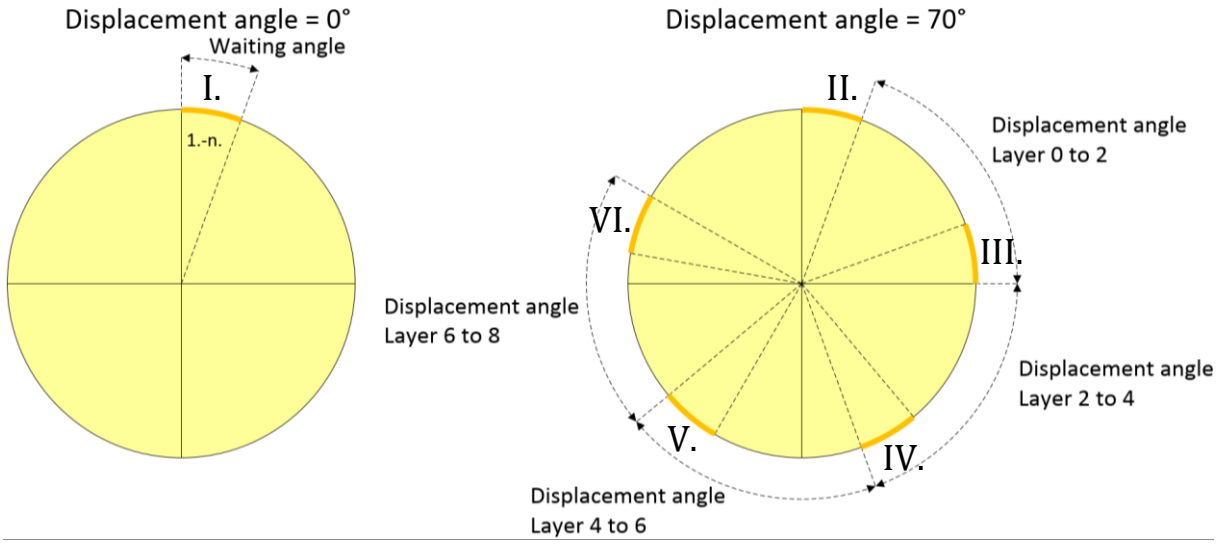
- Winding step [mm/ot]

  Vzdálenost mezi navíjeným materiálem na cívce za jednu otáčku. Pokud je nastavena stejná hodnota, jako je šířka materiálu, pak je materiál navíjen těsně vedle sebe. Lze zadat i menší hodnotu a pak se jednotlivé vrstvy překrývají.

- Displacement angle [°]

  Úhel, který cívka urazí za dobu, za kterou osa Traverser přejede z pozice A do pozice B, tedy polovinu Traversing cycle. Pokud tedy na začátku navíjecího cyklu je cívka v úhlu x° a proběhne celý navíjecí cyklus (včetně waiting angle v bodu A i B), skončí cívka v úhlu y°. Displacement angle je pak absolutní hodnota rozdílu těchto úhlů. Při výpočtu se počítá s přetečením ve 360°, tedy

pokud je např. úhel cívky v bodě A 30° a v bodě B 100°, pak je Displacement angle roven 70°. Jde o velmi užitečný parametr pro zajištění rovnoměrného navíjení na okrajích cívky, dopad na navíjení při jeho různém nastavení je na Obr. 1-7.



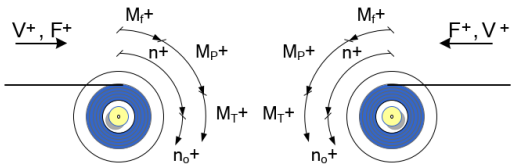
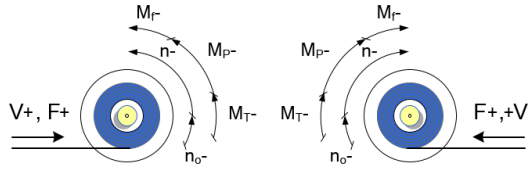
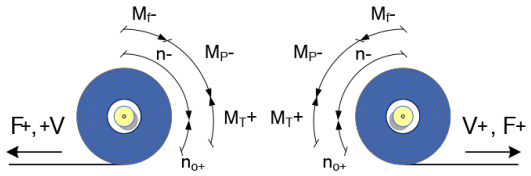
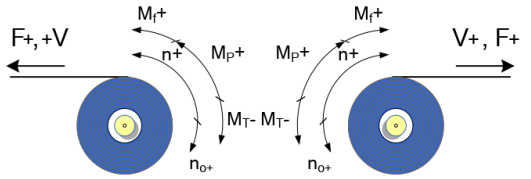
Obr. 1-7 Displacement angle [3], upraveno

Nastavení hodnoty Displacement angle 0° znamená, že osa Traverser dovede navíjený materiál na konec cívky vždy na stejné pozici. Na Obr. 1-7 je žlutým kruhem vyznačena hrana (konec) cívky (např. v bodě B) a zvýrazněná oblast **I.** označuje místo, kam se na této hraně navine materiál při určeném Displacement angle. Délka zde navinutého materiálu bude odpovídat úhlu natočení cívky v tomto krajním bodě, tedy nastavenému Waiting angle. S každou vrstvou se tak na tomto místě navine další materiál a ve všech ostatních bodech bude část bez navinutého materiálu v šířce dosahující až nastaveného Winding step v mm. Při nastavení většího Displacement angle se místo navíjení na hraně cívky s každou navinutou vrstvou posune vzhledem k cívce právě o Displacement angle a rovnoměrně se rozprostře mezi body **II.** až **VI.** dle aktuální navíjené vrstvy.

- Winding mode

V navíjecích aplikacích je relativně velký rozdíl hodnot tahu a potažmo potřebného momentu stroje v případě, že se navíjený materiál na cívku navíjí, nebo odvíjí a zároveň jestli je veden shora, nebo zdola. Rozdíl je především v třecím momentu, pnutí a rychlosti otáčení cívky, graficky jsou módy znázorněny na Obr. 1-8. Aplikaci by mělo být možné připravit na všechny tyto případy pomocí parametru windingMode.

Nastavovaných parametrů je více, zde jsou pouze základní, které potřebují podrobnější popis. Nastavení nejen těchto základních parametrů je popsáno v podkapitole 2.2.2.4.

Mode	Representation	Description
Rewind from above		V: Machine velocity F: Tension in the material web n: Speed of the roll being wound n_O: Velocity override for speed controller override M_f: Frictional torque M_P: Precontrol torque M_T: Tension torque
Rewind from below		
Unwind from below		
Unwind from above		

Obr. 1-8 Navíjecí mód (Winding mode) [3]

1.2.2.1 AUTO/Manual mode

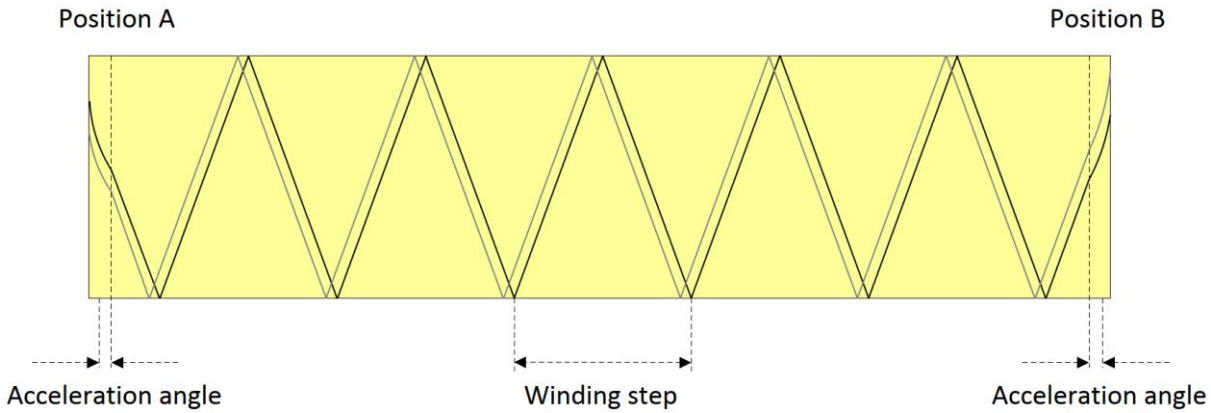
Protože v provozním stavu je při této aplikaci nutná synchronizace dvou pohonů, není možné řídit každý pohon zvlášť. Naopak se řídicí systém musí postarat o to, aby nebylo možné aktivovat a rozběhnout pouze jeden pohon, protože pak systém vypadne ze synchronismu. TO pak budou hlásit chybu o tom, že jedna z os je mimo předpokládanou pozici. Společně s tím bude pak i technologie nesprávně synchronizovaná a nevíjení nebude probíhat správně. Blokace individuálního spouštění os, a naopak spouštění pouze za předpokladu, že jsou k tomu připravené obě osy je zajištěno v režimu AUTO.

V některých situacích, zpravidla při uvádění do provozu, je ale nutné osy řídit odděleně. V případě osy Traverser je potřeba určit krajní body osy, u osy Winder zase natočení pro vhodnou počáteční pozici navíjení, nebo odvíjení. Pro tyto situace je možné se přepnout do režimu Manual. Zde je možné nastavit parametry os zcela nezávisle na režimu AUTO, tedy např. pokud je potřeba pomalu natočit osu cívky, je možné nastavit malou úhlovou rychlost nehledě na to, jaká hodnota byla aktuální v AUTO režimu. Lze zde osy testovat i za účelem ladění TO, zadávaní bodů A, B pro osu Traverser přímo podle fyzické polohy apod. Je ale důležité, že nejde o klasický provozní stav pohonu a využívá se pouze právě při uvedení do provozu, nebo servisu a diagnostice.

Nastavení této funkce je popsáno v podkapitole 2.2.2.4.

1.2.2.2 Traversing mode

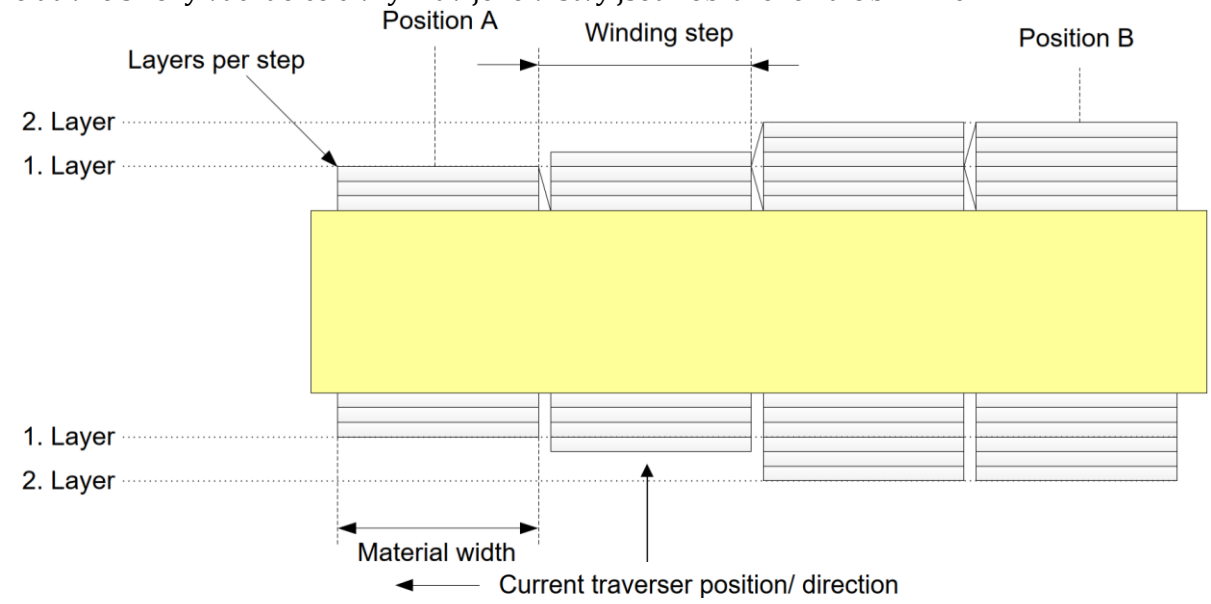
Podle požadavku lze v aplikaci vybrat způsob navíjení ze dvou možností. Každý mód má navíc specifické požadavky na zadání parametrů a zároveň parametry se stejným názvem mají jiný význam. Mód Continuous, tedy spojitý, je na řízení jednodušší režim, kde se navíjený materiál navíjí po jednotlivých vrstvách podél celé délky cívky. Lze ho využít pro většinu aplikací, např. navíjení vinutí pro elektrické motory. Výsledná navinutá vrstva je na Obr. 1-9. Winding step zde může být menší než tloušťka materiálu, ale z obrázku je patrné, že pak se bude navíjený materiál překrývat i v rámci jedné vrstvy, což může způsobovat nerovnoměrné navíjení následující vrstvy.



Obr. 1-9 Spojité navíjení (Continuously) [3]

V tomto módu se tedy osa Traverser urychluje konstantním zrychlením na zadanou rychlost. Během této doby se cívka pootočí (a navine se materiál) o úhel roven parametru Acceleration angle [°]. Materiál se poté navíjí konstantní rychlostí podél celé cívky dle nastavených pozic bodů A, B, mezera mezi navíjeným materiálem je na jednu otáčku rovna hodnotě zadané v parametru Winding step [mm/ot]. Na konci cívky osa Traverser zastaví s konstantním zpomalením, cívka se pootočí o úhel roven opět parametru Acceleration angle [°] a proces se opakuje. Grafy zaznamenané v simulacích s tímto režimem jsou na Obr. 2-24, Obr. 2-25 a Obr. 2-26.

V některých případech je vhodnější podél cívky vytvořit více „disků" z navinutého materiálu. Pro tyto potřeby je možné využít mód Step wind. Lze ho využít např. při navíjení VF cívek, kde je celková indukčnost tvořena více sériově řazenými menšími celky vinutí pro potlačení skinefektu. Častější použití je ale např. v textilním průmyslu, nebo obecně aplikacích, kde je navíjený materiál relativně široký vůči délce cívky. Navíjené vrstvy jsou zobrazené na Obr. 1-10.



Obr. 1-10 Krokové navíjení (Step wind) [3]

V tomto módu navíjení probíhá tak, že se osa Traverser přesune na pozici prvního disku, kde zůstává v klidu a nechá navíjet materiál jen na část cívky definované parametrem Material width. Po navinutí vrstev definovaných parametrem Layers per step se osa Traverser přesune na další disk a pokračuje v navíjení. V tomto případě je nutné správně napočítat šířku jednotlivých disků vůči počáteční pozici A. Osa Traverser pak povede navíjený materiál tak, že všechny disky budou mít stejnou šířku Material width. Počet disků aplikace dopočítá automaticky na celá čísla se zaokrouhlením dolů. Pro aplikaci s tenkým drátem nedává použití tohoto módu příliš velký smysl,

pro účely demonstrace funkčnosti tohoto módu jsou ale také zaznamenané grafy v simulacích na Obr. 2-27.

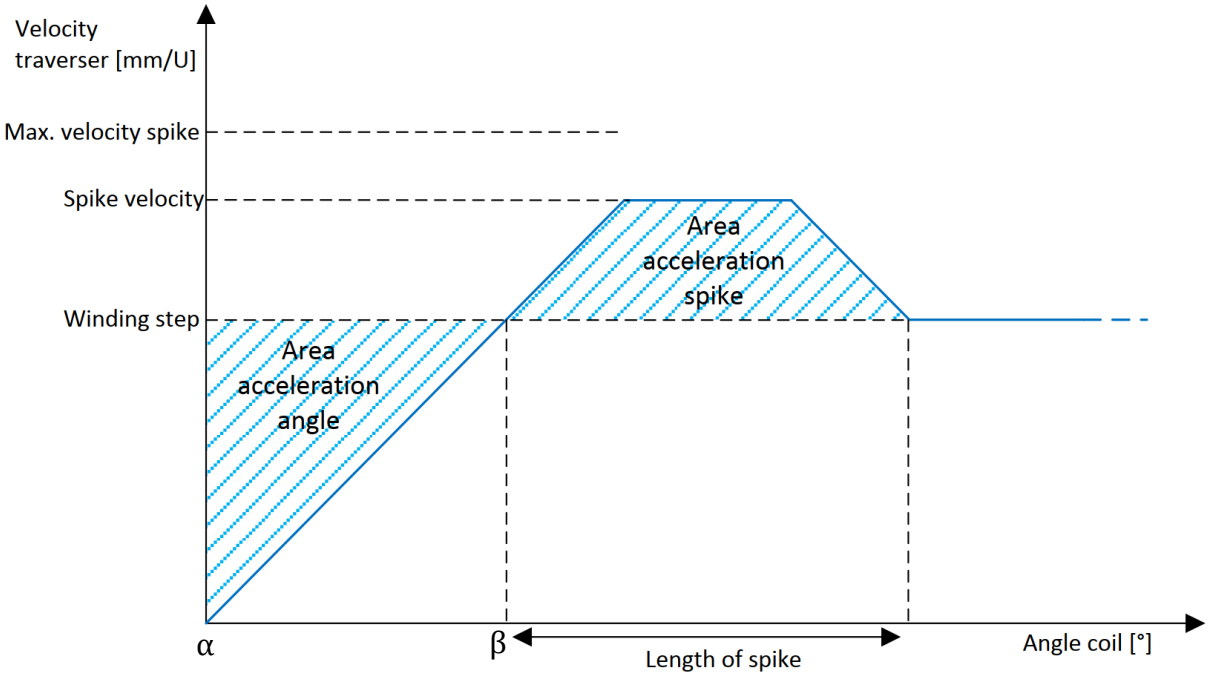
1.2.2.3 Spike

Spikes, neboli „špičky“ jsou dalším způsobem, jak předejít hromadění materiálu na koncích cívky. Kompenzuje místa při Acceleration angle, tedy ve chvíli, kdy osa Traverser zrychluje, nebo zpomaluje. V praktické části jsou použity 50W servomotory řízené 100W FM (popsané v podkapitole 2.1), navíjející tenký lehký drát se zanedbatelným protimomentem. Osu Traverser je tedy možné urychlit prakticky okamžitě a hromadění navíjeného materiálu není potřeba ošetřovat. V praxi ale existují situace, kde zrychlení osy Traverseru není možné provést takto rychle, např. aplikace s velkými výkony (kde by byl omezující max. proud motoru osy Traverser), nebo aplikace s velkým protimomentem zásobovací cívky (kde by byl omezující max. moment motoru osy Traverser). Tato kompenzace dává smysl pouze při navíjecím módu Continuous a proto je v módu Step wind ignorována.

Princip kompenzace je zobrazen na Obr. 1-11. Aplikace počítá integrál rozdílu požadované hodnoty rychlosti traverseru vůči skutečné hodnotě, popsané jako Area acceleration angle.

$$Area\ acceleration\ angle\ = windingStep \cdot \beta - \int_{\alpha}^{\beta} \left((Velocity\ traverser)\ d(Angle\ coil) \right) \qquad (1\text{-}1)$$

Když osa Traverser dosáhne maximální rychlosti, aplikace spočítá tuto hodnotu a dle nastavených parametrů zrychlí nad standardní navíjecí rychlost Winding step tak, aby během definovaného maximálního úhlu Lenght of spike dosáhla kompenzační plocha stejné hodnoty Area acceleration spike. Do aplikace se tedy nezadávají přímo hodnoty rychlosti a úhlu Spike, ale pouze maxima, která osa Traverser nepřekročí. Kromě toho lze také zvolit, jak moc má osa ztrátu při rozjezdu/brzdění kompenzovat v rozsahu 0 – 200 %, kde výchozí hodnota je 100 %. Při volbě např. 200 % tedy aplikace vypočítá plochu Area acceleration angle, ale kompenzační plochu vytvoří dvakrát větší. V případě, že jsou požadovaná regulační okna příliš úzká, aplikace spočítá, že není možné požadovaný Spike vytvořit v plném rozsahu. Vytvoří tedy maximální, který je v nastaveném úhlu s nastavenou rychlostí možný, a vyšle varování, které je zpracováno na HMI.

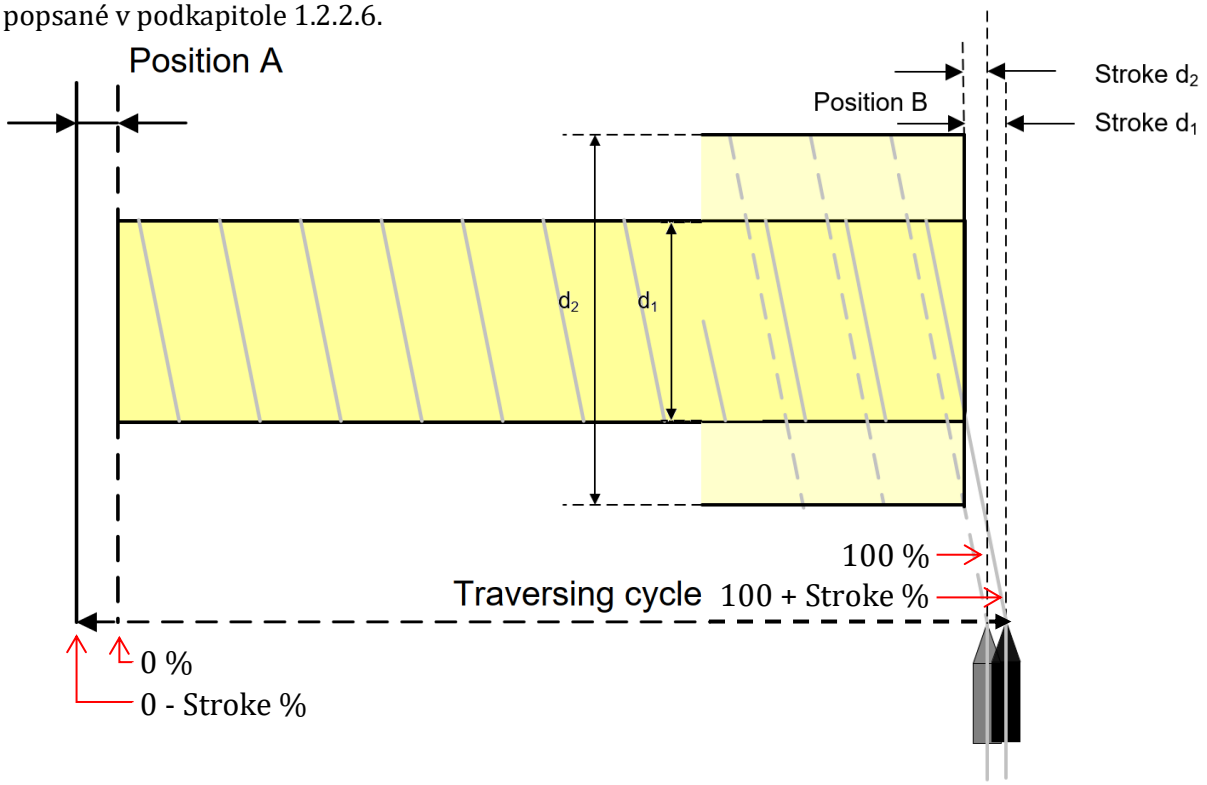


Obr. 1-11 Funkce Spike [3], upraveno

Grafy ze simulací s použitím této funkce jsou na Obr. 2-25.

1.2.2.4 Stroke

Pokud jsou kladeny vysoké požadavky na souvislost navíjené vrstvy na cívce, lze použít funkci Stroke. Tato funkce opět řeší problém nerovnoměrného navinutí materiálu na konci cívky, způsobeného zpomalováním/zrychlováním a změnou směru osy Traverser. Ideálního navinutí nelze dosáhnout nikdy, protože by musel být materiál navíjen kolmo na cívku, což není možné. Funkce Stroke tedy umožní ose Traverser přejet za definovanou hodnotu konce cívky o hodnotu Stroke určenou v mm. Princip je zobrazený na Obr. 1-12. Stroke se počítá jako záporný vzhledem k startovací pozici A, a kladný k pozici B. Pokud je tedy Stroke nastaven na 15 % celkové vzdálenosti, pak bude rozmezí pohybu osy Traverser -15 až 115 % celkové vzdálenosti AB. Na obrázku je dále naznačena funkce Stroke adapting. Tuto možnost lze volitelně zapnout a aplikace bude automaticky dopočítávat novou hodnotu Stroke s každou navinutou vrstvou. Ta se totiž s přibývajícím navinutým materiálem, a tedy zvětšujícím se průměrem cívky postupně zmenšuje. K těmto výpočtům je potřeba počítat přesný průměr cívky pomocí funkce Diameter calculation popsané v podkapitole 1.2.2.6.

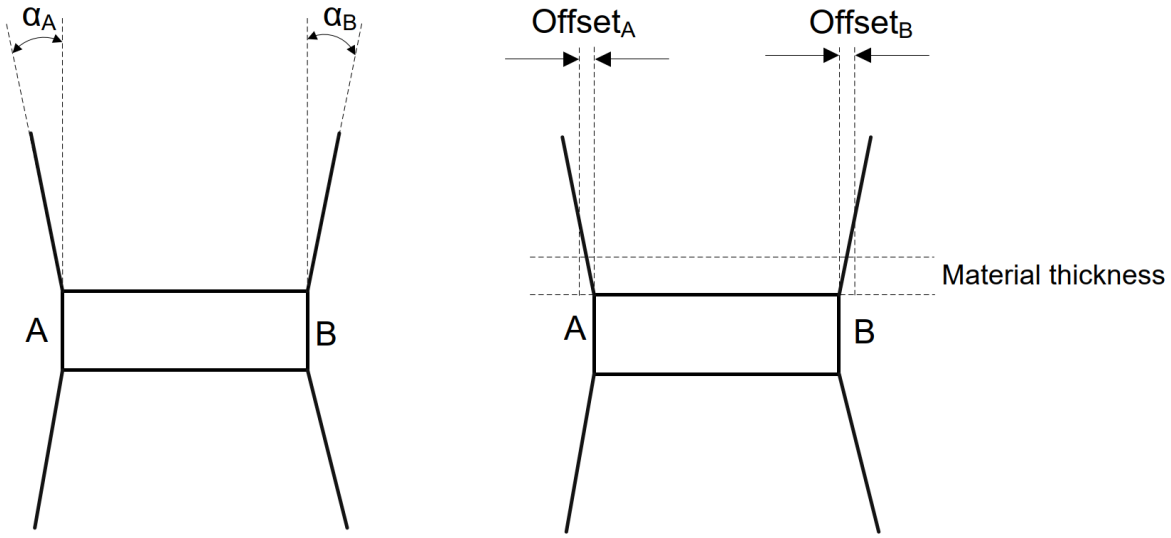


Obr. 1-12 Funkce Stroke [3], upraveno

Grafy ze simulací s použitím této funkce jsou na Obr. 2-25.

1.2.2.5 Conical coil profiles

Pokud je navíjená cívka kónického tvaru, Traversing cycle se s každou navitou vrstvou prodlužuje, nebo zkracuje podle toho, zda se cívka rozšiřuje, nebo zužuje. Aby nebylo nutné dopočítávat ke každé vrstvě nový Traversing profile, lze použít funkci Conical coil profile. Nejdříve je nutné zadat, jak bude tvar cívky popsán. Na Obr. 1-13 je jako první možnost zadání úhlu α. Pokud je úhel roven nule, šířka cívky zůstává stále stejná. V případě kladných čísel se šířka s přibývajícími vrstvami zvětšuje, v případě záporných pak zmenšuje. Zároveň je nutné znát přesný průměr cívky pomocí funkce Diameter calculation popsané v podkapitole 1.2.2.6. Druhou možností je změření a zadání offsetu cívky na tloušťku jedné vrstvy navíjeného materiálu v µm. Jako v předchozím módu znamenají kladné hodnoty zvětšující se průměr, záporné pak snižující. Tato metoda používá k dalším výpočtům aktuální počet navinutých vrstev a není tedy potřeba funkci Diameter calculation používat.



Obr. 1-13 Funkce Conical coil profile [3]

Grafy ze simulací s použitím této funkce jsou na Obr. 2-26.

1.2.2.6 Diameter calculation

Tato funkce je potřeba pro správné fungování několika předchozích funkcí ve specifických případech. Je ale nutné tuto veličinu znát i pro správné řízení navíječky. Možnosti výpočtu jsou dvě. Interní výpočet používá zadané parametry cívky a její aktuální stav. Výpočet pak vypadá dle (1-2) a(1-3), použité proměnné odpovídají proměnným používaných v DB LTrav_Data, který je součástí aplikace.

$$effThic = matThic\frac{matWidth}{windingStep} \qquad (1\text{-}2)$$

$$diameter = coreDiameter + layerCount \cdot 2 \cdot effThic \qquad (1\text{-}3)$$

Tyto vzorečky jsou popsány i v manuálu k aplikaci [3], je zde ovšem zavedeno jiné pojmenování proměnných, než je následně používáno v programu. Zápis přímo pomocí symbolických názvů proměnných z programu je pro rychlou orientaci, a hlavně při vyhledávání v textu vhodnější, a proto zde bude dodržován.

Druhou možností je externí výpočet průměru. Ten je v aplikaci kontrolován, a přepočítáván dle efektivní tloušťky navíjeného materiálu, zde však není využit.

1.3 Externí enkodér a vzdálený přístup

Mimo standardní aplikaci SIMATIC Traverser byl do projektu přidán externí enkodér pro odměřování rychlosti, délky a zrychlení navíjeného materiálu. Jde o jednoduchý jednostopý optoelektrický inkrementální enkodér s rozlišením 4 pulsy na otáčku pro základní představu o těchto veličinách. Výchozí aplikace s jeho použitím nijak nepočítá a bylo tedy nutné pro něj připravit od začátku novou programovou infrastrukturu. Při jejím vytváření byl dodržován stejný formát struktury programu, jako má standardní aplikace. Pro enkodér byly tedy připraveny vlastní PLC typy, funkční a datové bloky a následně i obrazovka v HMI.

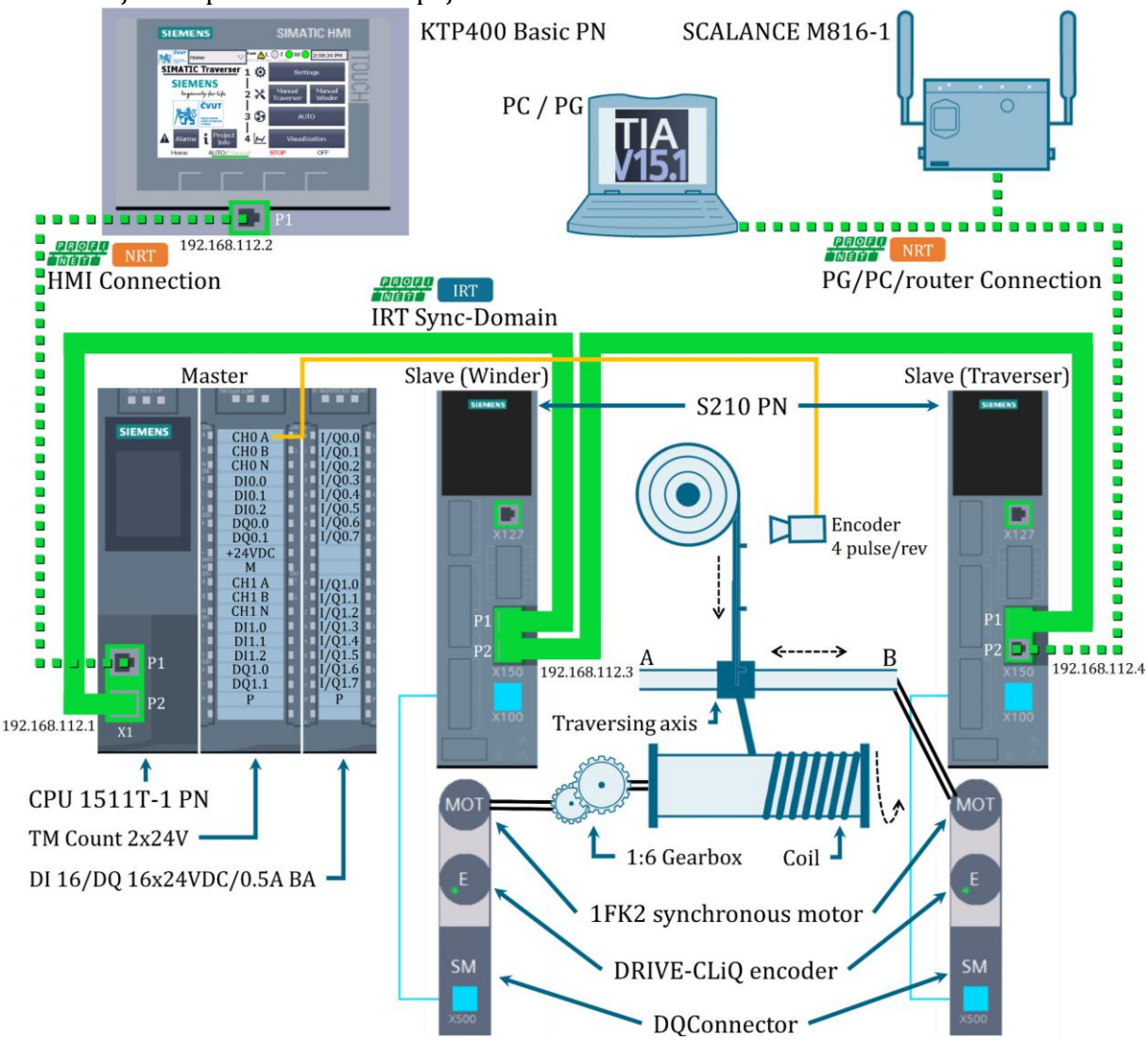
Během psaní této práce jsou již přes rok uplatňována protiepidemická opatření v souvislosti s šířením koronaviru covid19. Je tedy vhodné a někdy přímo nutné znát rozsah možností SW, co se vzdáleného přístupu týče. Ekosystém HW a SW firmy Siemens má v tomto obrovskou výhodu. TIA portál je totiž jednotný SW pro všechny nastavované (a mnoho dalších) komponent přímo firmy Siemens, ale i konkurenčních řešení (popsán je v podkapitole 2.2).

SW SINEMA Remote Connect (zkráceně RC) je jednoduchá platforma pro správu vzdálených sítí a je přímo navržena pro vzdálené ovládání zařízení Siemens. Servisní technik a stroj, který má být nastavován, navazují samostatná připojení k SINEMA RC serveru, kde identita účastníků je určena výměnou certifikátů. Teprve po úspěšné výměně je povolen vzdálený přístup ke stroji. [28]

KAPITOLA 2: PRAKTICKÁ ČÁST

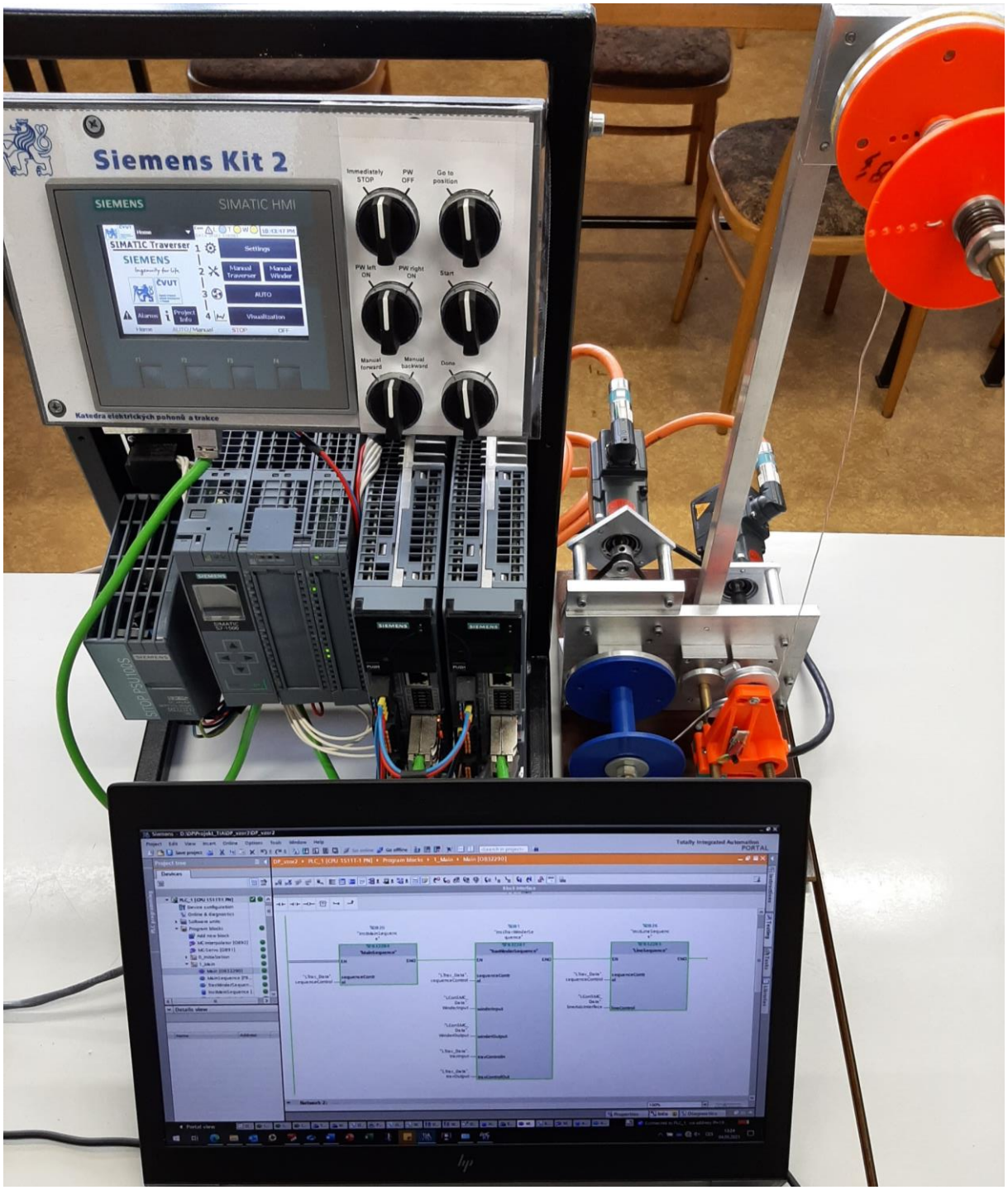
Celková HW konfigurace je zobrazena na Obr. 2-1. Z hlediska HW je Master PLC S7-1511T, jak je popsáno v podkapitole 1.2.1.1. K PLC (popis v 2.1.1) je pomocí PROFINET IRT komunikace IRT Sync-Domain připojeny dva FM S210 PN (popis v 2.1.2) pracující jako Slave. FM pracují zároveň jako switch, PLC tedy může při zapojení jako na obrázku komunikovat i s FM Traverser – signál putuje z PLC portu P2 do FM Winder portu P1 a dále z portu P2 do portu P1 FM Traverser. Každý FM je připojen pomocí systému připojení jedním kabelem (dále jen OCC) ke konektoru DQConnector. Ten zprostředkovává připojení k DRIVE-CLiQ enkodéru pro motory 1FK2 (popis v 2.1.3). Motor řízený FM Winder pohaní osu cívky (Winder), což je Master z hlediska programových bloků. Připojený je přes převodovku zapojenou do pomala s převodem 20:120. Motor řízený FM Traverser pohání posunovací osu (Traverser), ta je z hlediska programových celků Slave, rozdělení struktury programových celků je popsáno v podkapitole 1.2.1.3. Na odvíjecí ose, která není poháněná je umístěn jednoduchý enkodér poskytující rozlišení 4 pulsy na otáčku, pro výpočet množství odvinutého materiálu.

K PLC je poté ještě pomocí PROFINET HMI Connection připojen panel HMI KTP400 Basic (popis v 2.1.4) a pro uvádění do provozu a diagnostiku je použito PROFINET PG/PC/router Connection, v obou případech jde o asynchronní NRT komunikaci. IP adresy uváděné na obrázku jsou přímo nastavené v HW konfiguraci a používané v projektu. Zapojení PROFINET sítě i pulsů enkodéru je také podle reálného zapojení na demo kitu.



Obr. 2-1 Schéma HW konfigurace

Zapojení HW z předchozího obrázku je na Obr. 2-2 zobrazen ve skutečném provedení. Zde bude podrobněji popsána naviječka, její samostatný obrázek je na Obr. B-9. Naviječka má hliníkovou konstrukci a cívky jsou vytisknuté na 3D tiskárně. Zásobníková cívka (oranžová nahoře) není poháněna. Navíjecí cívka (modrá, dole) je poháněná FM Winder a navíjí drát ze zásobníkové cívky shora. Drát je na navíjecí cívku veden skrze dvě kladky, přičemž spodní kladka je snímána externím enkodérem pro zjišťování polohy a rychlosti navíjeného materiálu. Tyto kladky se pohybují podél osy navíjecí cívky a vedou tak drát na přesně určené místo pomocí FM Traverser. Na obrázku je dále PC pro uvedení do provozu a diagnostiku, a již popisovaný HW.



Obr. 2-2 Reálná HW konfigurace

2.1 Použitý HW

Zde budou krátce představeny nejdůležitější komponenty, ze kterých je Siemens Kit složen. Jejich seznam je v Tab. 2-1. Je zde také uvedená použitá verze FW a objednací číslo (dále jen MLFB), což je jednoznačný identifikátor produktů firmy Siemens. Detaily o použitém HW lze nalézt v odkazech uvedených u použité literatury, nebo po zadání MLFB do stránek Industry Mall firmy Siemens na následující stránce:

https://mall.industry.siemens.com/

Tab. 2-1 Použitý HW

Jméno	Verze FW	MLFB
SIMATIC S7-1500T, CPU 1511T-1 PN	V2.6	6ES7511-1TK01-0AB0
SIMATIC S7-1500 digital input/output module	V1.0	6ES7523-1BL00-0AA0
SIMATIC S7-1500, TM count 2x24V	V1.3	6ES7550-1AA00-0AB0
SIMATIC S7, Paměťová karta		6ES7954-8LE03-0AA0
SIMATIC S7-1500, DIN lišta 160 mm		6ES7590-1AB60-0AA0
SINAMICS S210, 0,1 kW	V5.2	6SL3210-5HB10-1UF0
2x SIMOTICS S-1FK2 HD Servo motor		1FK2102-0AG00-1MA0
2x Kabel SPEED-CONNECT M12, 2m		6FX5002-8QN04-1AC0
SIMATIC HMI, KTP400 Basic	V15.1.0.0	6AV2123-2DB03-0AX0
SCALANCE M816-1 ADSL router	V6.1.2	6GK5816-1BA00-2AA2
SITOP PSU100S 24 V/5 A		6EP1333-2BA20

2.1.1 PLC SIMATIC S7-1500

Řada S7-1500 patří v portfoliu firmy Siemens k nejvyšší řadě PLC. Ta přináší kromě vyššího výpočetního výkonu hlavně větší možnosti přizpůsobení aplikaci pomocí dalších modulů a také jednoduchosti obsluhy. Všechny modely mají barevný displej s uhlopříčkou 1,35" pro základní diagnostiku. PLC umožňuje také sledovat veškeré veličiny uložené v PLC tagu a vykreslovat je do grafů v reálném čase (Trace), což zjednodušuje ladění.

Konkrétní PLC použité v této práci má procesor (CPU) 1511T-1 PN. Z hlediska řady S7-1500T jde o základní verzi CPU a je popisováno jako vhodné pro aplikace se středními požadavky na rozsah programu a rychlost zpracování. Písmeno T v označení pak znamená Technology. Jsou zde tedy integrovány bloky funkce Motion control (PLCopen), není tedy nutné k jejich použití přidávat další moduly. PLC má 225KB operační paměť pro program, 1MB paměť pro data prostřednictvím SD paměťové karty a disponuje dvouportovým PROFINET switchem. [11]

Protože PLC samotné nedisponuje žádnými DI/DO, je na kitu k dispozici ještě modul vstupů a výstupů SIMATIC S7-1500 digital input/output module. [19] Modul se připojuje k PLC přímo pomocí konektoru z boku a umožňuje tak vyhodnocovat 16 DI a 16 DO. Ty mohou být použity k zapojení přepínačů na kitu. Pro zpracování signálů externího enkodéru je dále PLC doplněno o modul TM count, k PLC se připojuje stejným způsobem. [18]

PLC je nutné dovybavit zmiňovanou paměťovou kartou. Ta zde tvoří zaváděcí paměť (Load memory) a nahrávají se na ní data a program z TIA portálu. Při spuštění PLC jsou pak tato data nahrána do vnitřní paměti PLC a dále dělena na program a data, případně se vybrané datové celky přepisují do zálohy (retain memory). Důvodů proč se používá SD karta, a ne rovnou paměť PLC je více. Nejdůležitější je, že paměť SD karty je stálá (nevolatilní). Paměť uvnitř PLC je rychlejší, ovšem je nestálá (volatilní), a po odpojení od zdroje napětí se celá smaže. Jde také o systém, který byl používán už ve starším vývojovém prostředí S5 a je zde tedy jistá kompatibilita. Dále je možné takto program nosit s sebou mezi PLC, nebo omezit jeho používání v rámci jiných PLC, pokud je

potřeba chránit know-how programu. Nakonec je tu možnost snadné výměny v případě poškození karty, tedy není potřeba servisní zásah při rozbití paměti, ale stačí koupit novou.

Obr. 2-3 PLC SIMATIC S7-1500 s moduly [17][18][19]

2.1.2 FM SINAMICS S210

Jde o nástupce FM S110, který ještě rozšiřuje jeho možnosti. Aplikace pro tento měnič jsou takové, kde se nevyžaduje dlouhodobě plynulý pohyb, středních a vysokých požadavků na řízení momentu, rychlosti, pozice a koordinace více os. Typicky tedy lisy, pohon válcovacích stolic, nebo právě navíječky. U těchto měničů se předpokládá použití s PLC řady S7-1500, nebo S7-1500T. Nemají tedy integrované žádné polohovací funkce. Měnič disponuje kompletní sadou Safety funkcí dle normy DIN EN 61800-5-2, v Čechách přijata jako ČSN EN 61800-5-2 (351720).

Konkrétní FM použitý v této práci má vstupní napájecí napětí 200-240V 1AC, maximální proud 1.4A, výstupní frekvence 0-550Hz, výkonový modul (dále jen PM) je 0.1kW, krytí IP20. CU disponuje pěti DI, z čehož 2 mohou být použity pro Safety funkce. Dále 1 F-DI lze požít pouze pro Safety funkce STO a SS1, detailněji jsou popsány ke konci podkapitoly 2.2.1.2. Protože je potřeba ovládat dva motory a tento měnič samotný neumožňuje řídit je současně, jsou použity dva tyto měniče. [12]

Obr. 2-4 FM SINAMICS S210 [20]

2.1.3 Servomotor SIMOTICS S-1FK2

Při použití FM S210 je vhodné použít motor řady 1FK2. Tyto komponenty byly vyvíjeny od začátku pro vzájemnou spolupráci a z jejich použití plyne např. výhoda v připojení pomocí OCC, tedy připojení silové části, ale i pulsů enkodéru. Tyto servomotory jsou tvořeny synchronními motory s permanentními magnety (dále jen PMSM) s enkodérem. Konkrétní model použitý v této práci má jmenovitý výkon 0,05 kW, jmenovité otáčky 3000 ot/min a jmenovitý moment 0,14 Nm. Používá enkodér AM22DQC, absolutní enkodér 22bit + 12bit multiturn. Výstupem je tedy

absolutní úhel natočení rotoru v určeném rozlišení (až 4194304 pulsů na otáčku). Lze ale také pomocí něj počítat až 4096 otáček, k čemuž slouží přídavných 12 bitů. [12]

Obr. 2-5 Servomotor SIMOTICS S-1FK2 [21]

2.1.4 Panel SIMATIC HMI KTP400 Basic

Panel Siemens SIMATIC HMI KTP je základní panel z řady Basic. Přesto jde o velmi komplexní a všestranně použitelné zařízení. Nabízí se ve velikostech 4" až 12", displej je vždy typu TFT s 64000 barvami a disponuje USB vstupem pro připojení myši, klávesnice, nebo např. čtečky čárových kódů. Lze použít maximálně 800 HMI tagů v jednom komunikačním rozhraní. Pomocí USB lze také ukládat vybraná vyčítaná data. Všechny velikosti je možné provozovat také v režimu na výšku. [5][11]

V tomto případě má panel barevného displeje uhlopříčku 4 palce s poměrem stran 16:9, tedy přibližně 28 na 16 cm. Označení KTP znamená Key Touch Panel. Je tedy dotykový, k ovládání je ale možné také použít čtyři hardwarová tlačítka umístěná pod displejem. Dotyková vrstva funguje na kapacitním principu, displej tedy není možné ovládat v rukavicích, pokud k tomu nejsou přímo určeny. [24]

Obr. 2-6 Panel SIMATIC HMI KTP400 Basic [22]

2.2 TIA Portál

Totally Integrated Automation Portal, zkráceně TIA portál je software firmy Siemens, s jehož pomocí je možné konfigurovat a programovat téměř celé portfolio zařízení. Zastřešuje tři hlavní aplikace – WinCC pro návrh a programování HMI panelů, StartDrive pro konfiguraci měničů a Step7 pro programování PLC. Mimo to, lze ale také nastavovat softstartéry, síťové prvky, snímače a monitorující zařízení, zdroje napětí a další. Lze zde také provádět rozsáhlé simulace téměř všech těchto zařízení, nastavení Safety funkcí, diagnostiku v reálném čase i bezpečnost přenosu dat. [10]

Podrobný popis konfigurace projektu, všech možností nastavení, popis funkčních bloků a mnohých dalších funkcí TIA portálu není předmětem této práce. Zde budou popsány pouze nejdůležitější části při vytváření programu a zároveň nastavení, která nejsou úplně zřejmá z jiných návodů nebo nápověd. Podrobný popis vytváření projektu, přidávání správného hardwaru do projektu a další základní nastavení je k nalezení v manuálech [3], [4], [8] a [14] ze seznamu použité literatury, nebo přímo v nápovědě TIA portálu. [24]

Ačkoli je v době psaní této práce již k dispozici TIA portál verze 16, zde byla použita předchozí verze 15.1 s nejnovějšími aktualizacemi k 8.2.2021. Rozdíly mezi těmito verzemi jsou ovšem minimální. Hraniční použitelná verze SW je verze 14, u starších verzích může už ovšem docházet k problémům s programováním. Např. ve verzi 14 a 15 není FM S210 přímo integrován do HW katalogu. Je tedy nutné ho do projektu importovat pomocí tzv. GSDML souboru, což nastavení mírně komplikuje. V manuálu [4] je popsáno nastavení i při použití této verze. Kompletní seznam softwaru nainstalovaného na PC, kde konfigurace probíhala, je v Tab. 2-2.

Tab. 2-2 Použitý SW

Jméno	Verze	Vydání
Automation License Manager	V6.0 + SP8	06.00.08.00_01.02.00.01
S7-PLCSIM	V5.4 + SP8	V05.04.08.01_01.24.00.01
SIMATIC ProSave	V15.1	V15.01.00.00_28.01.00.01
SIMATIC S7-PLCSIM	V15.1 Upd1	V15.01.00.01_02.00.54.01
SIMATIC STEP 7 Professional - WinCC Professional	V15.1 Upd5	V15.01.00.05_03.01.00.01
SIMATIC WinCC Runtime Advanced Simulation	V15.1 Upd5	V15.01.00.05_03.01.00.01
SIMATIC WinCC Runtime Professional Simulation	V15.1 Upd5	V15.01.00.05_03.01.00.01
SINAMICS G110M, G120, G120C, G120D, G120P	V15.1 Upd3	V15.01.00.03_37.04.00.03
SINAMICS G130, G150, S120, S150, SINAMICS MV, S210	V15.1 Upd3	V15.01.00.03_37.04.00.03
SINEMA RC Client	V2.1	V2.1.0.0-01.01.00.12
STARTER	V5.3.0.1	V05.03.00.01_05.30.13.05
STEP 7 OEM	V5.6 + HF1 OEM2	K5.6.0.1_5.2.0.1
TIA Administrator	V1.0	V01.00.02.00_01.10.00.01
TIA Openness	V15.1	V15.01.00.00_37.00.00.01
TIA Portal Multiuser Server	V15.1 Upd5	V15.01.00.05_03.01.00.01
User Management Component	V1.9 SP1	V01.20.00.00_01.01.00.01

Jedna z hlavních výhod TIA portálu je řečena přímo v názvu tohoto SW – na rozdíl od konkurenčních programů např. Mitsubishi Electric, nebo OMRON je zde programovaní PLC, návrh HMI, uvedení do provozu FM a mnoho dalšího integrováno do jednoho programu. Nelze se ale tak vyhnout tomu, že je TIA portál velmi komplexní SW s velkým množstvím kontextových nabídek a příkazových oken už v základním stavu po nainstalování. Na první pohled tedy může být prostředí nepřehledné a matoucí. Zde bude prostředí krátce popsáno. Podrobnější popis lze nalézt například v manuálu [8].

Po spuštění TIA portálu je potřeba založit nový projekt. Zde byl pojmenován jednoduše Kalat_Ondrej_DP a tento název se tedy bude objevovat i v přiložených obrázcích. Na tento vytvářený projekt bude dále odkazováno jako na „projekt“, na výchozí projekt z manuálu [3] bude odkazováno jako na „vzorový projekt,“ nebo „standardní projekt“. Po pojmenování projektu a

určení cesty uložení tohoto souboru se projekt otevře v tzv. Portal view. Toto rozhraní je přehledné a jednoduché, ovšem také velmi stručné a spíše informativní. Pro samotné nastavení pohonu je tedy potřeba se přepnout na Project view tlačítkem vlevo dole, kde prostředí obsahuje mnohem více nastavení.

Prostředí má čtyři hlavní pracovní okna, jako příklad obrazovky pro jejich popis zde bude použit Obr. 2-7 z podkapitoly 2.2.1. Každé toto okno je možné zvětšit nebo zmenšit, maximalizovat, a s výjimkou okna 1 ho odepnout z hlavního okna Windows a přesunout ji např. na druhý monitor.

Okno 1 je navigační podokno (Project tree). Zde je možné se přepínat mezi HW částmi projektu (PLC, HMI, FM, atd.), ale například v rámci PLC je možné se přepínat mezi programovými celky (TO, PB, PLC/HMI tags atd.).

Okno 2 je hlavní pracovní plocha. Zde se zobrazí obsah vybraný v části 1. U některých obrazovek (např. zde u HW konfigurace) může být rozdělena na dvě podskupiny, kde v pravé části je zpravidla detailní popis prvků zobrazených v levé části. Navíc lze přidat ještě jednu celou tuto obrazovku a rozdělit je horizontálně, nebo vertikálně. Pak je možné pracovat velmi pohodlně např. na dvou různých PLC, je ale velmi vhodné v tomto případě mít k dispozici alespoň dva monitory. Mezi těmito okny je možné se přepínat pomocí záložek ve spodní části podobně jako v internetovém prohlížeči.

Okno 3 (nazývané také chytrá karta) je pomocné a primárně slouží k výběru prvků používaných v okně 2. Obsah se zde mění podle toho, co je zvolené v okně 1. Na Obr. 2-7 je to např. HW, který je možné použít v projektu, lze zde ale také vybrat funkce používané v programu PLC, nebo importovat knihovny. Okno má v pravé části několik záložek, mezi kterými lze přepínat. Nejdůležitější je ale záložka Libraries, kde lze spravovat a importovat knihovny PLC Types, PLC Blocks a mnoho dalších. Protože zde často bývá potřeba orientovat se ve velkých knihovnách, lze tuto konkrétní záložku maximalizovat přes celou pracovní plochu.

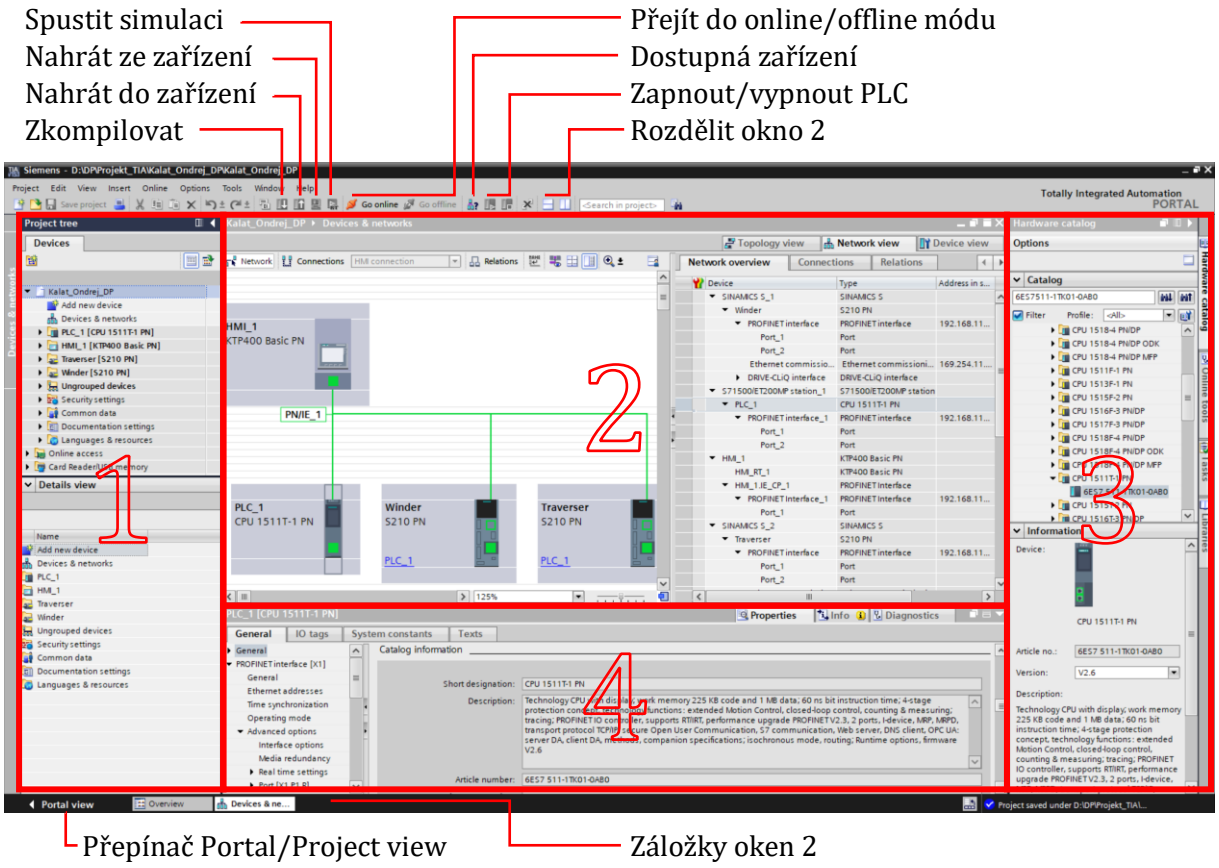
Okno 4 má opět několik záložek. V záložce „Properties“ se zobrazují podrobnosti o položce vybrané v okně 2 a lze zde také provést její detailní nastavení. Typicky tedy při nastavování HW například nastavování IP adres, často se také používá při návrhu HMI. Záložka „Info“ zobrazuje informace o kompilaci, nebo simulaci programu. Záložka „Diagnostics“ pak ukazuje okamžité hodnoty parametrů a stavy připojeného HW, pokud je připojený a v on-line stavu.

V dalším textu bude při popisu umístění ovládacích prvků dodržováno toto značení oken pro jednodušší orientaci.

2.2.1 Konfigurace HW

Jako první po vytvoření projektu je nutné do něj přidat HW, se kterým se bude pracovat. V okně 1 se tedy zvolí Devices & networks. V okně 2 jsou nyní 3 možnosti zobrazení. Topology view ukazuje HW zapojený tak jak je ve skutečnosti, je tedy dobré začít HW přidávat zde. V okně 3, záložce Hardware catalog lze pak HW vyhledávat podle MLFB žádané komponenty. V případě PLC a některých panelů a měničů je možné přidat tzv. nespecifikovaný produkt. Program tak přidá pouze kostru komponenty, např. PLC řady S7-1500. Při prvním připojení pak program sám určí, o jaké PLC přesně jde. Nastaví i podle toho další parametry podle jeho možností (podpora TO, počet DI/DO, atd.) a není nutné tak mít k dispozici štítek, pokud je špatně přístupný, nebo nečitelný.

V tomto případě tedy bylo přidáno jedno PLC, dva měniče a jedno HMI. Tahem mezi zeleně vyznačenými PROFINET porty se zařízení propojí PROFINET linkou. Finální topologii je možné vidět na Obr. 2-7. Je důležité, aby souhlasily porty PROFINET rozhraní s reálným zapojením. Nakonec je ještě vhodné si zařízení pojmenovat. Zde bylo ponecháno výchozí pojmenování PLC a HMI, byla ovšem změněna jména FM na Traverser a Winder. Každé toto zařízení si nyní vytvořilo vlastní složku v okně 1.



Obr. 2-7 TIA portal: HW konfigurace

Pro kontrolu je možné se přepnout do Network view. Zde je pouze potřeba zkontrolovat, zda jsou všechna zařízení propojena PROFINET komunikací a jestli jde o ten samý kanál – např. PN/IE_1 – pro všechna zařízení.

2.2.1.1 Konfigurace PLC

Dále je potřeba provést konfiguraci přidaných komponent. V okně 2 je potřeba přepnout do Device view a z rolovacího menu vybrat PLC. V okně 3, záložce Hardware catalog se nyní objeví široká škála možných rozšiřovacích karet a modulů pro toto PLC. Jak je napsáno v Tab. 2-1, k PLC je připojen TP Count karta a DI/DO modul. Je tedy potřeba tento HW přidat, k vyhledání je možné opět použít MLFB. Je důležité dodržet jejich správné pořadí, odpovídající reálnému zapojení. Další nastavení už souvisí s topologií aplikace a provádí se v okně 4, záložce Properties. Obsah tohoto okna se mění podle toho, jaká komponenta je označena v okně 2. Zde tedy musí být označené samotné PLC, ne např. vstupně-výstupní (dále jen IO) modul.

Je potřeba najít menu s cestou PROFINET interface [X1]/Advanced options/Real time settings. Zde se nastaví parametry komunikace. V menu IO communication je možné nastavit Send clock. Parametr ukazuje čas mezi dvěma po sobě následujícími komunikačními cykly. Jedná se o nejkratší možný interval přenosu pro výměnu dat. [25] Protože zde nebude potřeba komunikovat s velkým množstvím zařízení, je možné ho nastavit na maximální hodnotu, tedy 4 ms. V menu Real time options se pak ihned dopočítá a zobrazí čas zpracování jednoho taktu komunikace a zároveň ukáže procentuální vytížení linky. Při tomto nastavení je vytížení odhadnuto na 0,554 %, potenciál pro rozšíření aplikace je tedy z hlediska komunikační datové propustnosti PLC velký. Poslední nastavovanou částí je menu Synchronization. Zde už bude správně nastavená synchronizační doména Sync-Domain_1 (komunikace PROFINET s taktem 4 ms) a je potřeba zvolit roli PLC v synchronizaci. Z již probrané topologie je zřejmé, že PLC bude Master a je potřeba tedy nastavit roli Sync master.

Pro pozdější účely programování je vhodné zapnout systémové a čítačové paměťové bity (AlwaysTRUE, Clock_10Hz atd.). To se provede v menu PROFINET interface [X1]/Systém and clock memory zaškrtnutím systémových i čítačových bitů. Jejich adresu není nutné specificky nastavovat, pouze nesmí být u obou typů bitů stejná. Zde byly nastaveny čítačové bity na adresu 0 a systémové na adresu 1.

V tomto menu je možné nastavit také všechny připojené moduly. Protože DI/DO nebyly zde nijak využity, je nutné nastavit pouze čítačovou kartu TM Count. Enkodér se bude řídit (zapínat a referencovat) pomocí MC objektů, což se nastaví po kliknutí na čítačovou kartu v okně 4, záložce Properties, General, v menu TM Count 2x24V/Channel 0/Operating mode zaškrtnutím Position input for technology object „Motion Control“. Na stejné úrovni menu, v kolonce Module parameters se zadají informace o enkodéru, tedy Signal type: Pulse (A) (jednopulsní enkodér), Increments per revolution: 4 (napočítaná hodnota za jednu otáčku) a Reference speed: 3000 ot/min (referenční rychlost).

2.2.1.2 Konfigurace FM

Podobným způsobem je potřeba nastavit oba FM. Po jejich zvolení v Device view je potřeba v okně 3, záložce Hardware catalog vybrat motor. Je zde jediná varianta, a to motor 1FK2. V okně 4 je ale potřeba nastavit správný model motoru, opět dle MLFB to zde bude 1FK2102-0AG0x-xMxx.

V okně 4, záložce Properties je pak nutné také správně nastavit komunikaci. Cesta je stejná jako v případě PLC, nastavuje se zde ale pouze jeden parametr. V menu Synchronization se zde nastavuje RT class. Zde je potřeba zvolit komunikaci IRT. O úroveň výše, tedy cestou PROFINET interface [X1]/Advanced options/Isochronous mode je potřeba zaškrtnout že měnič bude provozován v izochronním režimu.

Nakonec je možné se v menu s cestou PROFINET interface [X1]/Telegram configuration podívat na konfiguraci telegramu. Na obou stranách, tedy odesílaný i přijímaný (transmit, recieve) je možnost volby Organization block a Process image. Tyto není potřeba nastavovat, protože se správně samy nastaví po inicializaci TO. Organization block se nastaví na MC-Servo-OB [OB91] a Process image na PIP OB Servo. Oba tyto parametry budou vysvětleny v podkapitole 2.2.2.1. Pak je ještě potřeba zkontrolovat, že je použit standardní Siemens telegram 105 opět na obou stranách. Odesílaná data jsou vypásána v Tab. 2-3.

Tab. 2-3 Siemens telegram 105 [13]

PZD01	PZD02	PZD03	PZD04	PZD05	PZD06	PZD07	PZD08	PZD09	PZD10
STW1	NSET_B		STW2	MOMRED	G1_STW	XERR		KPC	
ZSW1	NACT_B		ZSW2	MSGW	G1_ZSW	e.g.: G1_XIST1		e.g.: G1_XIST2	

Přijímaný (recieve) Odesílaný (transmit)

Přenos dat mezi měničem a PLC pomocí komunikace PROFINET probíhá pomocí cyklicky odesílaných zpráv, tzv. telegramů. Jde o uspořádaná data rozdělená do několika vždy stejně po sobě jdoucích bloků. Z hlediska samotné skladby telegramů je důležitá především jeho část PZD, která přenáší řídící a stavová slova a další informace. Zde je důležité, že stavové/řídicí slovo, resp. odesílaný/přijímaný telegram se u centrálního řízení vždy myslí z hlediska FM. Např. požadovaná rychlost NSET_B je z hlediska PLC odesílaná veličina, ale z hlediska FM je přijímaná, a proto je tak označována v telegramech. Telegram 105 je rozšířením telegramu 3 a lze ho použít pouze pro IRT komunikace. Přenáší následující data:

- Řídicí slovo 1 a 2 (STW1, STW2)

  Základní řídicí informace, základní řízení pozicování. STW1 je v Tab. 2-4, STW2 je v Tab. B-1.
- Stavové slovo 1 a 2 (ZSW1, ZSW2)

  Základní stavové informace, stav polohy. ZSW1 je v Tab. 2-4, ZSW2 je v Tab. B-1.

- Požadovaná a skutečná rychlost (NSET_B, NACT_B)

  Využívají celou 16bit proměnnou a jde o povinný parametr u všech telegramů.
- Řídicí a stavové slovo enkodéru 1 (G1_STW, G1_ZSW)

  Požadovaný režim pozicování, aktivace funkcí, indikace aktivování funkcí, homingu atd. Zobrazeny jsou v Tab. B-2.
- Skutečná pozice 1 a 2 enkodéru 1 (G1_XIST1, G1_XIST2)
- Hodnota pro redukci momentu (MOMRED)
- Komunikační slovo (MELDW)

  V Tab. 2-3 je označeno jako MSGW, pravděpodobně jde ovšem o chybu, protože jinak je vždy dodržováno značení MELDW. Z FM se zde odesílá stav pohonu a měničem informace o momentu a otáčkách vzhledem k nastaveným mezím a také informace o přehřívání měniče/motoru. Detailně je popsán v Tab. B-3.
- Rozdíl skutečné a požadované polohy (XERR)
- Proporční hodnota regulátoru polohy (KPC)

  Odesílá vypočítanou hodnotu z funkce K_p adaption popsaná v podkapitole 1.2.1.

Každá část PZD1, PZD2, … je 16bitová proměnná typu Word. Znamená to, že je primárně zobrazována v hexadecimálním formátu a maximální hodnota je 4000 Hex, což je 16384 Dec, tedy polovina maximální hodnoty 16 bitů. [24] Pro ukázku je na následujícím obrázku základní řídicí a stavové slovo 1.

Tab. 2-4 Řídicí a stavové slovo 1 [13]

Control word 1 (STW1)		Status word 1 (ZSW1)	
Bit	**Meaning**	**Bit**	**Meaning**
00	ON / OFF1	00	Ready for switching on
01	OFF2	01	Ready for operation
02	OFF3	02	Operation enabled
03	Enable operation	03	Fault active
04	Enable ramp-function generator	04	No coasting down active
05	Continue ramp-function generator	05	No fast stop active
06	Enable speed setpoint	06	Switching on inhibited active
07	Acknowledge fault	07	Alarm active
08	Reserved	08	Speed setp - act val deviation in tolerance t_off
09	Reserved	09	Control request
10	Control by PLC	10	Comparison value reached/exceeded
11	Reserved	11	Alarm class bit 0
12	Open holding brake	12	Alarm class bit 1
13	Reserved	13	Reserved
14	Torque / speed control	14	Closed-loop torque control active
15	Reserved	15	Reserved

Velmi užitečné a zákazníky často vyžadované funkce jsou Safety funkce. Toto označení zahrnuje mnoho aspektů FM, např. chybovost a rychlost DI, předdefinované funkce pro zastavení, nebo řízení rychlosti stroje. Využívají se v aplikacích, kde hrozí riziko zranění obsluhy, především v takových případech, kdy je nějaká úroveň tohoto rizika zároveň i standardním provozním stavem, např. aplikace obsluhou ovládané letmé pily. Účelem těchto funkcí je minimalizovat riziko poranění obsluhy, zajistit s co největší pravděpodobností zastavení stroje v případě poruchy, nebo nehody a chránit tak před újmami na majetku, ale hlavně na zdraví obsluhy. [13] Zde budou krátce představeny možnosti Safety funkcí měniče použitelné na tuto aplikaci.

Jak již bylo popsáno, FM S210 disponuje jedním fail-safe DI (F-DI) a dvěma rychlými Safety DI. K dispozici je 10 Safety funkcí, kde 3 jsou součástí Basic Functions a jde o funkce pro bezpečné zastavení STO, SS1 a bezpečné brzdění SBC. Další funkce jsou součástí Extended Functions a k jejich využití je potřeba Safety licence. Všechny tyto funkce také vyžadují FW měniče V5.1 SP1 a vyšší. Jde o pokročilejší funkce zastavení SS2, SOS, funkce limitace rychlosti SOS, SLS, SSM, funkce

bezpečného rozběhu ve správném směru SDI, limitace zrychlení SLA a pokročilé funkce bezpečného brzdění SBT. [12]

V tomto programu nebyly použity žádné z Safety funkcí. Důvodem je jednak to, že výkony motorů a hodnoty mechanických napětí jsou velmi nízké a v i praxi by přidávání Safety do takto bezpečných aplikací nedávalo smysl. Hlavním důvodem je ale to, že při centrálním řízení, jak je popsáno v podkapitole 1.2.1.1, řídí FM centrálně PLC a pokud má tedy FM využívat Safety, musí tuto funkcí mít i PLC. Označení pro Safety funkce pro PLC (a obecně napříč portfoliem firmy Siemens) je F na konci označení, což zde použité není a centrálně Safety funkce tak nelze provozovat.

Při podstatném zvýšení mechanických a fyzických parametrů aplikace by ovšem bylo naopak velmi vhodné tyto funkce využít. Při škálování výkonu nahoru, dle popisu navíječkové charakteristiky na Obr. 1-2 v podkapitole 1.1, navíjecí stroje obvykle pracují s velkým momentem, potažmo velkým mechanickým napětím. Zde se automaticky nabízí funkce STO a SS1. První funkce po zastavení stroje např. při údržbě, nebo dokončení navíjení zajistí jeho nulový moment a zabrzdí ho, pokud tomu tak není. Druhá funkce má podobný úkol, pouze nemonitoruje moment ale rychlost. Zde by tak nemohlo dojít k tomu, že např. odvíjená cívka „přetáhne“ motor a navíjená cívka se začne samovolně odvíjet. Z pokročilých funkcí se opět přímo nabízí funkce SDI. S touto aktivovanou funkcí měnič monitoruje směr otáčení motoru. Pokud se motor otáčí nepřípustným směrem, pohon zastaví motor co nejrychleji. Funkce by mohla být použita ze stejného důvodu, jako předchozí, ale při rozběhu pohonu po dokončení navíjení, nebo po údržbě. Lze však určitě najít využití pro další funkce, protože každá z nich je velmi flexibilní, co se přizpůsobení týče a vše pak záleží na konkrétní realizaci aplikace. Obecně, aby tato Safety opatření měla smysl, by měl být výkon strojů řádově alespoň v desítkách kW, není to však pravidlem.

2.2.2 Programování PLC

TIA portál umožňuje programovat PLC v pěti programovacích jazycích z nichž každý má své výhody, nevýhody a vhodné případy použití. STL je jazyk nejnižší programovací úrovně podobající se jazyku symbolických adres (assembleru). SCL je vyšší jazyk vycházející s Pascalu, který již umožňuje strukturované programování. Oproti Pascalu nabízí více funkcí specifických pro PLC, např. přímé kódování jejich IO, volání bloků, nebo čítače. Zároveň umožňuje pracovat s proměnnými, které jsou k němu napojené přes LAD, nebo FBD a může je i plně nahradit (na úkor přehlednosti). [7] FBD je grafický programovací jazyk který popisuje vstupy a výstupy funkčních bloků a zapojuje je do větších celků. LAD, tedy klasický žebříkový diagram je opět grafický programovací jazyk vycházející z kontaktních (relé) schémat. Poslední a nejmodernější jazyk je S7-GRAPH. Jde opět o grafický programovací jazyk vhodný pro popis sekvenčních cyklů, které jsou rozděleny na jednotlivé kroky pro větší přehlednost. V těchto krocích se vykonávají potřebné operace a lze je doplňovat o textový komentář, ale i obrázky.

Tab. 2-5 Programovací jazyky v SIMATIC S7 podle IEC 61131-3 [26]

Jazyky podle IEC 61131-3	Odpovídající jazyky v SIMATIC S7
IL – Instruction List	STL (pokrývá pouze část)
ST – Structured Text	S7-SCL (odpovídá IEC 61131-3 a PLCopen Base Level)
FBD – Function Block Diagram	FBD (pokrývá pouze část)
LD – Ladder Diagram	LAD (pokrývá pouze část)
SFC – Sequential Function Chart	S7-GRAPH (odpovídá IEC 61131-3 a PLCopen Base Level)

Všechny tyto jazyky splňují normu IEC 61131-3 (přijata i v ČR jako ČSN EN 61131-3). Tato norma definuje jazyky pro programování řídicích jednotek a sjednocuje syntaxi i sémantiku za účelem sjednocení jazyků používaných v průmyslu. Dále norma definuje požadované společné prvky, např. deklarace proměnných a datových typů, funkcí a funkčních bloků atd., nezakazuje ovšem přidání dalších funkcí specifických pro požadovanou aplikaci. [16] V Tab. 2-5 jsou vypsány

programovací jazyky dle IEC 61131-3 a jejich odpovídající jazyky používané v TIA portále s PLC SIMATIC S7.

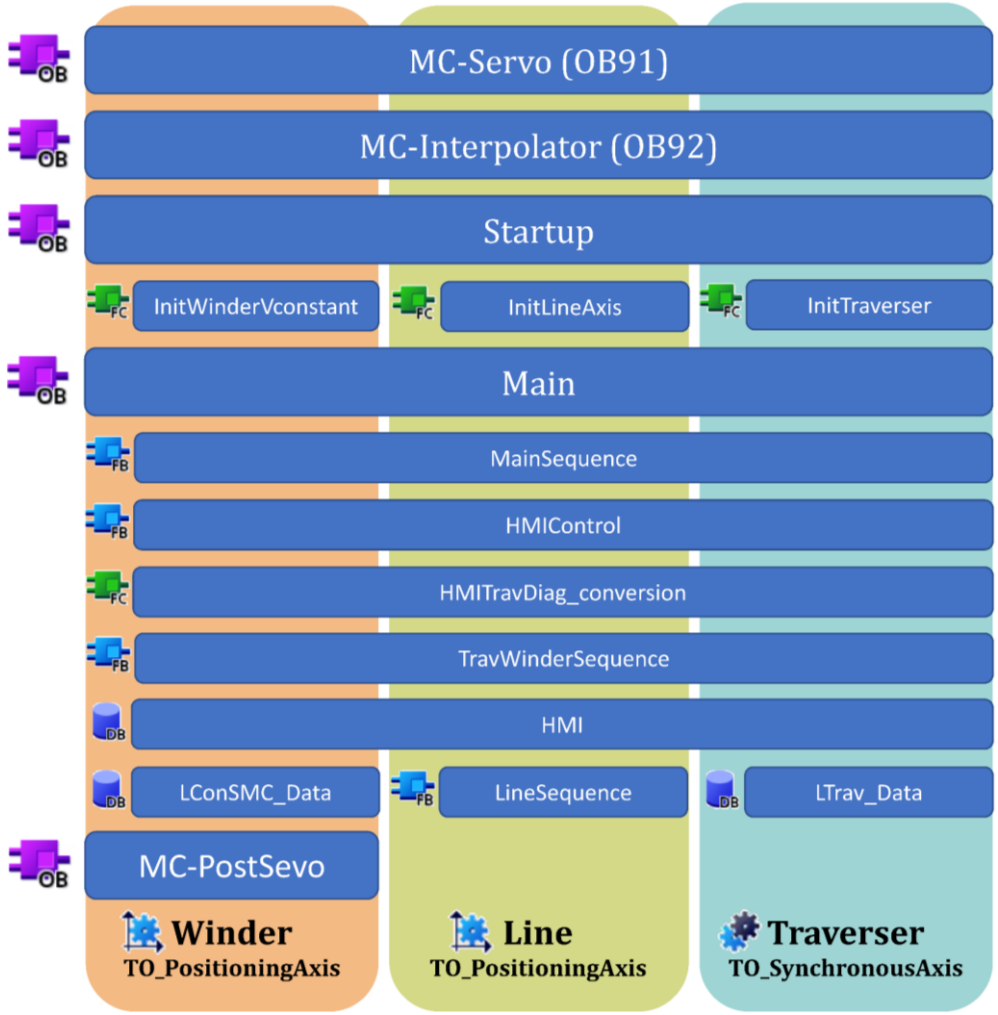
Program popisovaný v této práci byl naprogramován s použitím především jazyku SCL a bloky vytvořené tímto jazykem byly propojeny v jazyku LAD. Program používá na pozadí v nutných OB i jazyk STL, do těchto částí programu ale nebylo zasahováno (většina z nich je know-how firmy Siemens). Zde tedy bude velmi krátce na příkladu popsána syntaxe a funkce SCL, protože některé názvy proměnných budou používány i v textu. Výpis z kódu níže je část FB MainSequence, tedy hlavní sekvence běhu. Jde skutečně pouze o část, na které bude demonstrována syntaxe a sama o sobě by takto nefungovala správně.

Na řádku 1 je definován region. Jde o nepovinné rozčlenění kódu pro větší přehlednost. V TIA portálu se při jeho vytvoření (a správném ukončení) vytvoří záložka a lze se tak na celý region rychle přemístit, skrýt ho apod. Region je ukončen na řádku 28. Stejně jako v Pascalu, i zde nejsou středníky součástí příkazu, ale příkazy jen oddělují. Proto za řádkem 1 být nemusí, ovšem za řádkem 28 ano. Na řádku 3 je vidět standardní zápis komentáře. Na řádku 4 je zápis funkce IF pomocí klíčového slova a podmínky. Není nutné zde hlídat složené závorky jako např. v programovacím jazyce C. Ty jednoduché jsou potřeba pouze při nutnosti správné posloupnosti vyhodnocování jednotlivých podmínek. Podmínka na řádku 4 ukazuje přístup k externí proměnné vstupující, nebo vystupující do/z tohoto bloku pomocí znamínka #, tedy `#sequenceControl.travWinderStat`. Hodnota `#ERROR` je pak symbolický znak pro číselnou hodnotu definovanou v hlavičce funkce, zde konkrétně #ERROR = 900. Na řádku 13 začíná příkaz CASE … OF, který podle hodnoty `#sequenceControl.mainState` umožní provedení jednotlivých větví programu. Zde je ukázána pouze jedna větev pro případ `#IDLE`, tedy hodnotu 0. Nakonec, na řádku 17 je vidět zápis přístupu k proměnné v jiném DB, tedy takové, která není přímo vstupem do funkce. Jméno DB se píše do uvozovek `"HMI"` a následuje cesta k požadované proměnné oddělovaná tečkami. Při psaní cesty TIA portál napovídá možné proměnné, takže je jejich hledání jednoduché. Operátor := přiřazuje hodnotu proměnné (nebo konstanty) vpravo, proměnné vlevo.

```
 1 REGION state machine
 2
 3   // Error handling
 4   IF (#sequenceControl.travWinderState = #ERROR
 5     OR #sequenceControl.lineState = #ERROR)
 6     AND #sequenceControl.mainState <> #ERROR
 7   THEN
 8     #statErrorFirstCycle := TRUE;
 9     #sequenceControl.mainErrorState := #sequenceControl.mainState;
10     #sequenceControl.mainState := #ERROR;
11   END_IF;
12
13   CASE #sequenceControl.mainState OF
14     #IDLE:
15       IF #statEnableModule THEN
16         // Reset manual mode inputs from HMI
17         "HMI".HMITravCtrl.ctrTravAxis.enableAxis      := FALSE;
18         "HMI".HMITravCtrl.ctrTravAxis.executePos      := FALSE;
19         "HMI".HMIWinderCtrl.winderAxisCtrl.enableAxis := FALSE;
20         "HMI".HMIWinderCtrl.winderAxisCtrl.executePos := FALSE;
21         #sequenceControl.mainState := #ENABLE_TRAV_WINDER_MODULE;
22       ELSE
23         #sequenceControl.travWinderState := #IDLE;
24       END_IF;
25
26   END_CASE;
27
28 END_REGION ;
```

Samotné bloky funkcí jsou čtyř základních typů – organizační bloky (OB), funkce (FC), funkční bloky (FB) a datové bloky (DB) z nichž každý má své unikátní číslo, které mu TIA portál udělí automaticky a lze ho upravit. OB tvoří rozhraní mezi PLC a uživatelským programem a PLC je volá podle jejich typu při systémové události, např. přerušení (Cyclic interrupt, Time delay interrupt), chybě (Programming error, IO access error), nebo typicky hlavní programová smyčka Main. [25] Speciální OB je také např. MC-Servo nutný pro řízení servopohonů. FC jsou bloky kódu nebo podprogramy bez vyhrazené paměti. Proměnné, které jsou zde definované, nebo je jim přiřazena hodnota a nejsou uloženy např. do DB jsou při dalším průchodu hlavním cyklem smazány. FB jsou bloky kódu podobně jako FC, které ale trvale ukládají hodnoty do jim přidružených speciálních datových bloků, takže zůstávají k dispozici i po vyhodnocení bloku. [25] Ke každému FB je tedy automaticky vytvořen DB kde lze k těmto hodnotám kdykoliv přistupovat. Samotné DB pak ukládají data. Lze je nastavit jako jednoduché, nebo tzv. Multi-instance bloky paměti, kde do příslušného DB může zapisovat hodnoty více FB.

Na Obr. 2-8 je souhrn programových bloků používaných v aplikaci. Základní bloky budou stručně popsány v dalších podkapitolách. Některé z těchto bloků v sobě volají další pomocné bloky, ty zde popsané nebudou. Jde o standardizované bloky aplikace se všemi jejich výhodami (a nevýhodami) popsanými v podkapitole 1.2. Komunikační struktura vstupů a výstupů bloků pro Traverser tak odpovídá blokům Winder, což zjednodušuje orientaci v programu. Použité bloky lze rozdělit do tří základních skupin podle toho, k jakému TO se vztahují, tedy polohovací osa Winder, polohovací osa Line a synchronizační osa Traverser, např. FC InitWinderVconstant v oranžovém poli je velmi jednoduchá funkce inicializace proměnných pro TO Winder a provádí se v OB Startup. Kompletní projektový strom tak jak je zobrazen v TIA portálu je na Obr. B-1 v Příloze B.

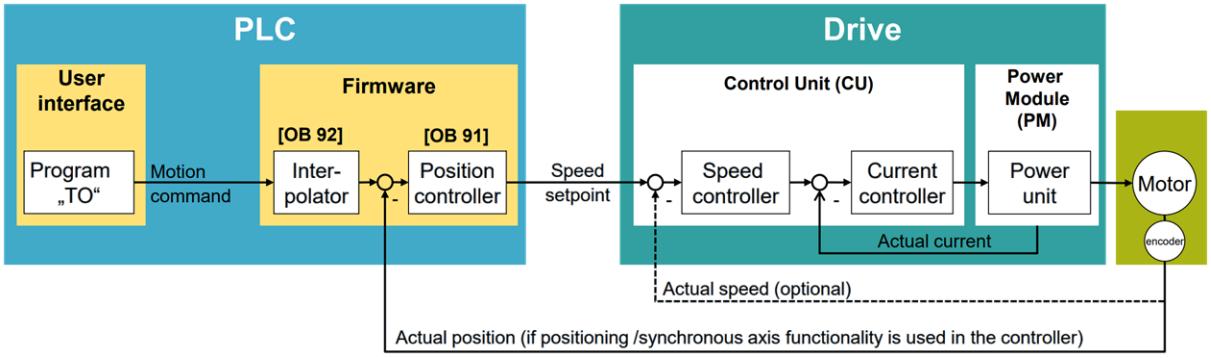


Obr. 2-8 Bloková struktura programu

2.2.2.1 Konfigurace TO

Technologické objekty (TO) jsou SW objekty v PLC, které mají více funkcí. Reprezentují reálné mechanické komponenty připojené k FM a umožňují použití technologických funkcí (bloky PLCopen, MC atd.) včetně jejich následné diagnostiky a sledování jejich stavu. TO je více druhů např. PID bloky, bloky čítání a měření. [6] Zde budou popsány pouze Motion Control (MC) TO, pomocí nichž je možné jednoduše řídit osy pohonu.

Mezi základní osy patří rychlostní osa (TO_SpeedAxis), polohovací osa (TO_PositioningAxis) a synchronizační osa (TO_SynchronousAxis). Tyto osy tvoří komunikační rozhraní mezi PLC a pohonem (FM). Automaticky pak pomocí nadefinovaného telegramu data odesílá. Obsahuje také všechna konfigurační data, žádané a skutečné hodnoty, a stavové informace. Na Obr. 2-9 je blokové schéma řízení pohybu a polohy v PLC. TO slouží jako první člen, který převezme požadavek na změnu rychlost, polohy atd. určenou programem v hlavní smyčce Main a předá ji FW, tedy základním OB91 a OB92. Zde se mimo jiné vyhodnocuje a reguluje aktuální a požadovaná poloha. Výsledná požadovaná rychlost je pomocí např. PROFINET komunikace pomocí telegramu přenesena do FM. Zde následuje regulátor rychlosti, jehož výstupem je požadovaný proud a následně regulátor proudu, jehož výstupem je požadovaný moment. [4] Tato struktura je vygenerována automaticky, není třeba do ní zasahovat. V rámci nastavení TO je možné ladit regulační konstanty, případně aktivovat funkci dynamického řízení servopohonu DSC, která bude popsána později.

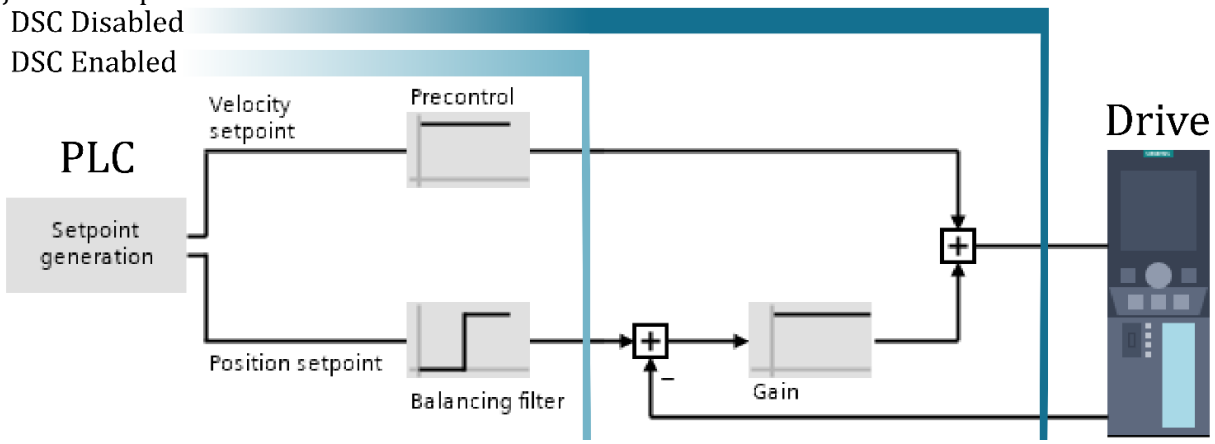


Obr. 2-9 Řídicí smyčka pohonu [6], upraveno

V tomto programu je použito celkem 7 TO, zde bude stručně popsáno jejich nastavení. Osa Winder je typu TO_PositioningAxis, v programu je pojmenována Winder_Lead aby bylo zřejmé, že jde o vedoucí osu. Tato osa vypočítává žádané hodnoty polohy, generuje zrychlovací a brzdné rampy. Bere v úvahu nastavení enkodéru, a vydává podle toho odpovídající žádané hodnoty otáček do pohonu. [6] Protože je jako pohon použit standardní motor 1FK2, TIA portál po jeho přiřazení z HW konfigurace sám pozná typ enkodéru a nastaví základní parametry, jako je např. max. rychlost motoru. Ta byla ale dále omezena z 8000 °/s na 360 °/s. Náběhové a sestupné rampy byly nastaveny na 1 s, tedy zrychlení i zpomalení na 360 °/s^2. Aby pohon nepřecházel do běhu, nebo nezastavoval s velkými momentovými rázy, bylo nastaveno vyhlazování ramp na 0,1 °/s^2, což odpovídá 600 °/s^2. Jako referencování (homing) se používá nulová značka v PROFIdrive telegramu, nouzová brzda je nastavena na 0,1 s.

Osa Traverser je typu TO_SynchronousAxis a je synchronizována k ose Winder. Tento typ osy má oproti předchozí jedno nastavení navíc, a to typ synchronizace. Zde je tedy potřeba zvolit právě TO Winder_Lead. Jako spojení je zadán Setpoint, systém se tedy synchronizuje na ideální požadovanou hodnotu a zanedbává tak případné odchylky skutečného průběhu. Druhou možností by bylo zvolit skutečnou hodnotu, kde jsou případné odchylky respektovány. Pro testování v simulacích na tomto nastavení prakticky nezáleží a při malých rychlostech je možné toto nastavení ponechat i v reálném provozu. Ostatní nastavení bylo nastaveno pro účely simulace stejně jako u Winder s rozdílem, že jde o lineární osu a ne rotační. Hodnoty jsou tak číselně stejné, jen mají jednotky mm/s a mm/s^2.

U obou těchto os je velmi důležité povolit jejich simulaci. Jedině tak je možné s nimi pracovat v PLCSIM a ovládat je bez připojení reálného HW. U obou těchto TO byla také aktivována funkce DSC. Tato funkce umožňuje přenést část polohové regulace do pohonu. [25] Rozdíl výpočtu polohy s a bez DSC je na Obr. 2-10. Při jeho aktivaci tak samotné vyhodnocování skutečné a reálné polohy a následnou úpravu rychlosti obstarává CU. Tímto způsobem se zvyšuje přesnost polohování, protože měnič nemusí čekat celý komunikační cyklus na informace o požadované rychlosti. V čase, kdy nemá k dispozici aktuální požadovanou rychlost z PLC ji interpoluje interně a rovnou sám zpracovává, čímž vyplňuje jinak ztracený čas. Pracuje pouze s telegramy 3, 5, 105 a 106 a pouze s podporovanými FM. Tato funkce se využívá především u aplikací s velkými požadavky na dynamiku pohonu a v případech, kdy se pracuje s velkou rychlostí a zároveň vysokou přesností. Zde tedy není nezbytně nutná a s omezenou rychlostí v TO její výhody pravděpodobně nebudou jednoduše pozorovatelné.



Obr. 2-10 Funkce DSC [25], upraveno

Dalším použitým objektem je TO_Cam, tedy vačka. Jejich použití není nutné, ale pokud jsou použity, jsou potřeba celkem tři, tedy Cam_A, Cam_B a Cam_C. Tyto TO definují funkci f(x) (zde polohu) pomocí interpolačních bodů. Pomocí nich se pak vypočítává navíjecí profil skrze FB LTrav_CamCalc a další. Mít tyto body definované jako TO je vhodné, protože mnoho funkcí může navíjecí profil výrazně měnit, např. při použití kónické cívky, funkce je popsána v podkapitole 1.2.2.5. Tyto TO není potřeba nijak nastavovat, jejich parametry obstarává program a stačí je tak správně přidat a pojmenovat.

Osa synchronizace Winder a Traverser se jmenuje Line a jde opět o TO_PositioningAxis. Jako jediná je čistě virtuální osa, nepropojuje se tedy s žádným měničem z HW konfigurace. Výchozí nastavení osy je vyhovující a není potřeba ho měnit. Je pouze nutné opět povolit simulaci této osy a ujistit se, že je nastavena jako lineární osa.

Poslední TO_ExternalEncoder zpracovává data z jednoduchého enkodéru. V konfiguraci TO je nutné především nastavit, že jde o lineární osu a že je využíván enkodér z PLC karty TM Count na kanále 0. Výchozí telegram 83 pro přenos dat je vyhovující. Ostatní nastavení je již definováno v HW konfiguraci, popsáno je v podkapitole 2.2.1.1, samotný vytvořený program bude popsán v podkapitole 2.2.2.5.

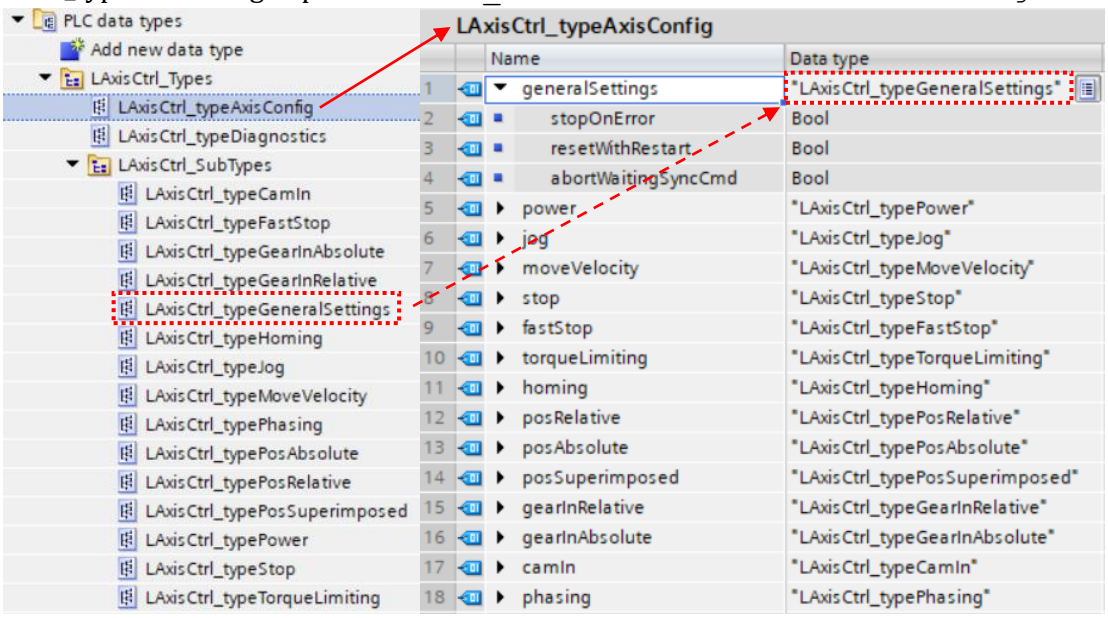
V programu nefiguruje žádná rychlostně řízená osa TO_SpeedAxis. Všechny použité TO jsou verze V4.0. V žádném TO také nesmí být povoleny žádné limity osy, tedy ani pozice, ani momentu. Nejsou přípustné SW ani HW limity, protože aplikace s nimi nepočítá a limity si sama dopočítává ve funkcích. Ke každému TO je pak pro potřeby programování možné přistupovat jako ke klasickému DB, který si TO vždy vytvoří a vyčítat z něj jak nastavené, tak diagnostické parametry.

2.2.2.2 Použité knihovny a PLC typy

Pro správnou funkci programu jsou potřeba knihovny LTrav [3] pro Traverser, LConSMC [8] pro WInder a LAxisCtrl [9] je obecná knihovna pro řízení pohybu os. Knihovny je potřeba po stažení importovat do TIA portálu. To se provede v okně 3, záložce Libraries, kliknutím na Retrieve

library. Ty se pak extrahují do určené složky a automaticky otevřou. Knihovny jsou tvořeny několika složkami, které obsahují nezbytné části programu. Každá knihovna LTrav, LConSMC a LAxisCtrl tedy obsahuje složky Blocks, Tags, Types a Example. Všechny tyto složky je nutné zkopírovat z knihovny v okně 3 do projektu v okně 1, např. přetažením. Pro jednodušší orientaci v programu je možné vytvářet téměř ve všech strukturách okna 1 podsložky (Group) a adekvátně je přejmenovat, např. LTrav_Types.

Složka PLC types obsahuje datové typy aplikace. Tyto typy jsou proměnné obsahující strukturu více proměnných různého typu, např. UInt, nebo DWord, ale lze v nich použít opět další složený typ. Důležité je, že samotný PLC typ slouží pouze jako deklarace samotných proměnných a žádna jejich hodnota pro program se zde nenastavuje. Jako příklad zde bude uveden PLC typ z knihovny LAxisCtrl. Složka LAxisCtrl_Types obsahuje dva PLC typy. LAxisCtrl_typeAxisConfig pro řízení osy a LAxisCtrl_typeDiagnostics pro indikaci stavu osy společně s podsložku LAxisCtrl_SubTypes s patnácti pomocnými typy. Vezměme jako příklad první zmíněný typ. Ten obsahuje 15 proměnných, každá proměnná je datového typu určeného v podsložce LAxisCtrl_SubTypes, tedy např. první proměnná generalSettings je datového typu LAxisCtrl_typeGeneralSettings. A tento datový typ se skládá ze tří proměnných stopOnError, resetWithRestart a abortWaitingSyncCmd, kde každá z těchto proměnných je již standardního typu Bool. [9] Struktura je zobrazena na Obr. 2-11. PLC typy lze přejmenovávat a upravovat, rozhodně však není vhodné měnit již předdefinované typy. V programu se pak použijí tím, že se po pojmenování nastaví jejich příslušný data type a nastavená struktura dalších proměnných se již vytvoří sama. V programu je jednou z proměnných, která má datový typ LAxisCtrl_typeAxisConfig např. `"LConSMC_Data".lineAxisInterface.config`.



Obr. 2-11 Struktura LAxisCtrl_typeAxisConfig

Složka PLC Tags obsahuje nutné PLC Tagy a konstanty pro běh programu. PLC tag je pojmenovaná proměnná se svojí určenou adresou v PLC. Tyto proměnné se vypisují v PLC tag table. Typicky jde o IO PLC, např. po zadání tagu „Start“ a jeho propojení k adrese I0.0 lze pak k tomuto DI vždy přistupovat pod jménem „Start“. Před jejich zkopírováním je dobré zkontrolovat, jaké PLC tagy jsou již v projektu zavedeny. V současném nastavení jsou ve výchozí tag table zavedeny tagy pro telegram 105 (Traverser_Actor_Interface_AddressIn atd.) a tagy pro externí enkodér z karty TM count (Line_Sensor_Interface_AddressIn atd.). Pro účely programování je vhodné si v HW konfiguraci povolit další systémové a čítačové paměťové bity (AlwaysTRUE, Clock_10Hz atd.) jak bylo popsáno v podkapitole 2.2.1.1. Tyto tagy mohou mít strukturu podobnou PLC typům. Součástí PLC tagů jsou také konstanty. Dělí se na uživatelské a systémové,

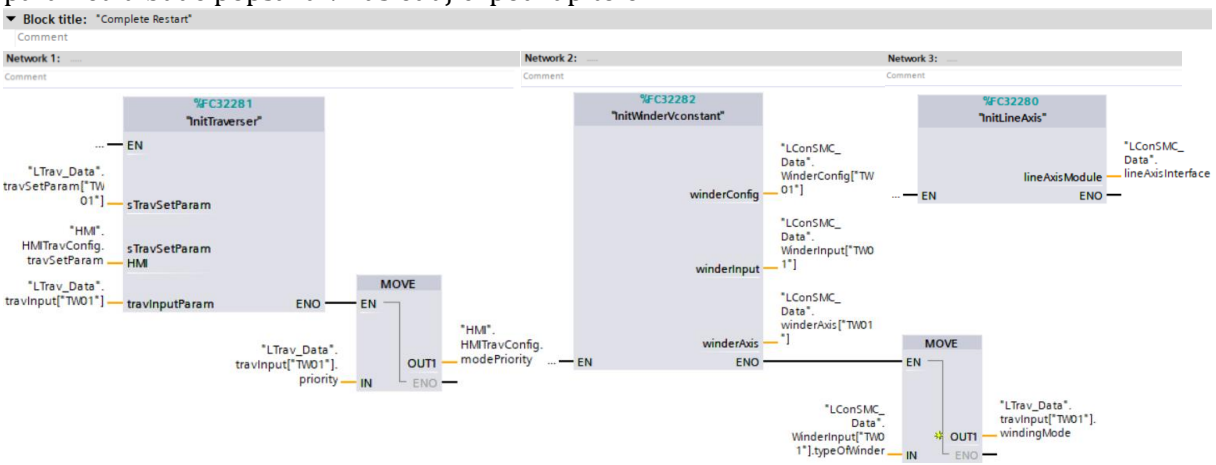
z nichž zde budou přidány pouze uživatelské. Jde o globální konstanty v paměti PLC bez přiřazené adresy, např. se zde ukládá čas komunikačního cyklu.

Složka PLC Blocks obsahuje programové bloky (dále jen PB), tedy OB, FB, FC a DB nutné pro správný běh aplikace. Výchozí aplikace obsahuje více než 30 těchto bloků a většina z nich bude popsána v následujících podkapitolách.

2.2.2.3 Bloky inicializace

Inicializace proměnných po prvním spuštění programu se provádí standardně v OB Startup. Neexistuje žádný jiný způsob, jak vyvolat spuštění tohoto OB znova, než přepnutím PLC do režimu STOP a následně RUN. Po prvním průchodu programu tímto OB se automaticky začne vykonávat OB Main. Počet OB Startup není omezen, pokud je jich více, volají se postupně dle jejich čísel. Nelze zde použít bloky přerušení (Startup nelze přerušit) a tento OB nemá žádný definovaný čas, po kterém se ukončí.

Standardní aplikace obsahuje tři inicializační bloky. Je zde blok pro Traverser InitLineAxis, pro Winder InitWinderVconstant a Line InitLineAxis. Ve všech případech jde o FC, jak lze vidět na Obr. 2-8. V OB Startup se volají jednoduše, blokům se pouze přiřadí adekvátní vstupní a výstupní proměnné jak lze vidět na Obr. 2-12. Ve vzorovém projektu jsou zde ještě dva bloky Move, které kopírují v případě osy Traverser počáteční nastavenou prioritu výpočtu, v případě osy Winder pak parametr `.typeOfWinder`. Tato zvláštní inicializace speciálně druhého jmenovaného parametru bude popsána v následující podkapitole.



Obr. 2-12 Struktura OB Startup

Výchozí hodnoty nastavené ve vzorovém projektu je potřeba upravit. To se provede poklepáním na příslušnou funkci, nebo jejím vyhledáním v okně 1, čímž se otevře textový editor kódu. Tyto bloky jsou zvlášť podrobně komentované a názvy proměnných jsou většinou srozumitelné a výstižné. Jejich vnitřní struktura a funkce jsou navíc velmi podrobně popsány v manuálu. Je ovšem škoda, že nelze tyto inicializační hodnoty nastavit v DB LTrav_Data. TIA portál má k dispozici u každého DB sloupeček Start value s nastavením výchozí hodnoty. Zadání hodnoty přímo v prostředí DB by bylo pro zákazníka určitě jednodušší a přehlednější než upravovat přímo SCL kód. Navíc takto může dojít snadno k omylu, protože v manuálu není nikde specifikováno, že se tato kolonka použít nedá. Jak ale bylo vysvětleno výše, hned po spuštění PLC se provedou všechny bloky v OB Startup a i když po nastavení výchozích hodnot do sloupečku Start value proměnná bude mít skutečně tuto hodnotu, ta bude vzápětí přepsána hodnotou definovanou v FC uvnitř OB Startup. Řešením by tedy bylo buď využít tuto funkci přímého zadání výchozí hodnoty proměnné, nebo jasně varovat před jejím využitím.

Další nesrovnalost je zřejmá po otevření těchto bloků. Ačkoliv je v manuálu [3] několikrát dlouze vysvětlována důležitost hodnot Waiting angle a Displacement angle (popsané jsou v podkapitole 1.2.2) a důvody proč je vhodné, aby nebyly nulové, ve výchozím nastavení obě nulové jsou. Alespoň malý úhel pro obě veličiny (cca 10°) by v některých případech dramaticky

zlepšil kvalitu navíjení bez nutnosti toho, aby zákazník přesně rozuměl jejich významu. Zde by bylo velmi vhodné alespoň nějaký úhel ve výchozím nastavení zachovat, nebo opět důrazně varovat na jejich výchozí nulové nastavení.

Protože pozměněných parametrů v rámci inicializačních FC bylo více, jsou přehledně sepsány v Tab. 2-6. Tabulka obsahuje i dvě inicializační veličiny pro TO a jejich zahrnutím tak obsahuje všechny parametry související s mechanikou aplikace, a další které bylo nutné fyzicky změřit na pohonu (s některými dalšími souvisejícími).

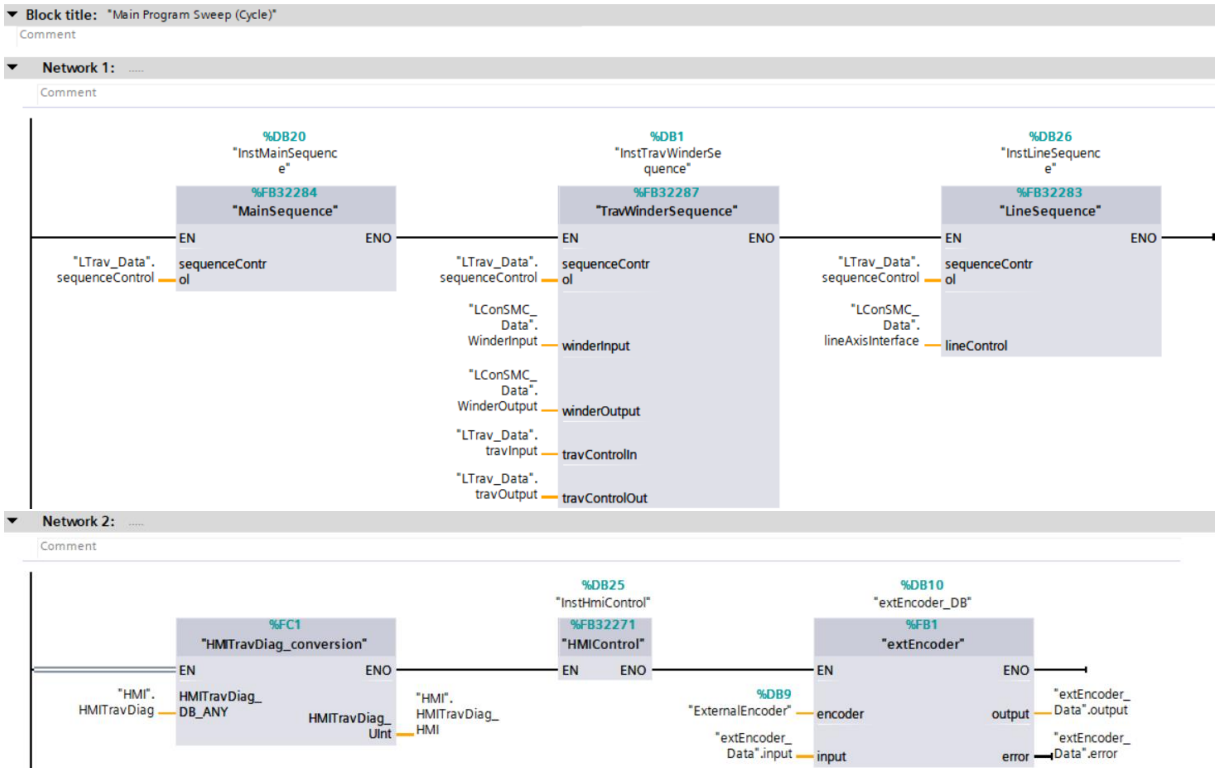
Tab. 2-6 Změněné inicializační hodnoty

Parametr	Význam	Hodnota	Jednotka
	TO Traverser		
.LeadScrew	Zdvih osy traverseru	1,25	[mm/ot]
	TO Winder_Lead		
.LoadGear.Numerator	Počet otáček motoru na jednu otáčku zátěže	6	[-]
	FC InitTraverser		
.coreDiameter	Průměr jádra cívky	17	[mm]
.startPosA	Pozice A vzhledem k počátku (středu cívky)	-29	[mm]
.startPosB	Pozice B vzhledem k počátku (středu cívky)	29	[mm]
.matThick	Tloušťka navíjeného materiálu	0,82	[mm]
.matWidth	Šířka navíjeného materiálu	0,82	[mm]
.waitingAngle.set	Nastavený Waiting angle	20	[°]
.waitingAngle.max	Maximální Waiting angle	180	[°]
.windingStep.min	Minimální Winding step	0,7	[mm/ot]
.windingStep.max	Maximální Winding step	10	[mm/ot]
.displacementAngle	Hodnota Displacement angle	20	[°]
	FC InitWinderVconstant		
.webThickness	Tloušťka navíjeného materiálu [m]	0,00082	[m]
.MaterialWidth	Šířka navíjeného materiálu [m]	0,00082	[m]
.minDia	Minimální dovolený průměr jádra cívky [m]	0,017	[m]

2.2.2.4 Bloky hlavní sekvence

Hlavní programová smyčka Main se ve výchozím programu skládá ze čtyř FB. Hlavní blok MainSequence je poměrně jednoduchá funkce, jejíž vstupem je proměnná `.sequenceControl` obsahující informace stavu všech tří os, včetně jejich chyb. Velmi malá část kódu této funkce byla použita jako příklad kódu pro vysvětlení programovacího jazyka SCL v podkapitole 2.2.2 na str. 26 a zde tedy bude odkazováno na čísla řádku tohoto výňatku. Po úvodním ošetření reakce systému na chyby na řádcích 4 až 11 se zde program větví klíčovým slovem CASE … OF podle parametru `sequenceControl.mainState` na řádku 13, a pro každý ze čtrnácti možných stavů provede nutné operace. Na příkladu je zobrazena pouze reakce na hodnotu parametru 0, tedy IDLE na řádku 14, který je aktivní v základním stavu AUTO režimu. Následuje jednoduchý vnořený příkaz IF. Pokud jsou v tomto stavu zapnuty moduly hlavní sekvence, znamená to, že byl předtím aktivní Manual režim, jehož pokyny zde musí být ignorovány. Program je tedy rovnou pro příští spuštění Manual režimu restuje na řádcích 17 – 20. Na řádku 21 pak změní stav tohoto parametru na hodnotu 10, tedy ENABLE_TRAVWINDER_MODULE a program se posouvá dále. Pokud moduly spuštěny nejsou, hlavní sekvence zůstává v režimu IDLE, jak lze vidět na řádku 23. Procházení všech dalších stavů je podobné a jejich význam je vysvětlen v manuálu [3].

Velmi podobným způsobem jsou naprogramovány další bloky z hlavní programové smyčky. Blok TravWinderSequence obsahuje příkazy pro 29 různých stavů os Traverser a Winder pomocí parametru `.travWinderState`, blok LineSequence zase řeší 18 různých stavů os Line dle `.lineState`. Struktura těchto bloků se všemi připojenými vstupy a výstupy je zobrazena na Obr. 2-13.



Obr. 2-13 Struktura OB Main

Posledním blokem zavedeným ve vzorové aplikaci je blok HMIControl. Jde o nejkomplikovanější FB vzorového projektu a zpracovává řídicí informace z HMI a následné diagnostické informace odesílané do HMI. Samotný blok je stručně popsán v podkapitole 2.2.2.6, zde budou popsány pouze některé konkrétní úkony prováděné v tomto bloku.

Jedna ze základních funkcí pro běh HMI je přepínání mezi AUTO a Manual režimem, jejichž popis je v podkapitole 1.2.2.1. V manuálu k aplikaci je možné zjistit, že toto přepnutí se provede povolením samostatného FB LTrav_TravCtrl. Jak bude ale popsáno níže, takový blok se ve vzorovém projektu nevyskytuje a výňatek z kódu níže je tedy přímo z FB HMIControl. Jde o velmi jednoduchou část kódu, která v případě aktivovaného AUTO módu resetuje hodnoty pro pohyb os Traverser a Winder na řádcích 4 až 17. Poté pouze přenáší hodnotu z parametru `.manualModeSelected` do `.enable` proměnných aktivace uživatelského rozhraní. Pokud je tedy nastaven bit `.manualModeSelected`, jsou nastaveny i bity pro povolení uživatelského rozhraní aplikace. FB HMIControl řídí i mnoho dalších věcí, např. přepínač „setů“ pohonů, který umožňuje ovládat až 255 pohonů touto jednou strukturou (ne však současně), nebo např. ověření konzistence zadaných hodnot.

```
 1 REGION Manual mode
 2
 3 // Reset manual commands
 4   IF NOT "HMI".manualModeSelected THEN
 5     // Reset traverser
 6     "HMI".HMITravCtrl.ctrTravAxis.enableAxis  := FALSE;
 7     "HMI".HMITravCtrl.ctrTravAxis.executePos  := FALSE;
 8     "HMI".HMITravCtrl.ctrTravAxis.executeStop := FALSE;
 9     "HMI".HMITravCtrl.ctrTravAxis.homeAxis    := FALSE;
10     // Reset winder
11     "HMI".HMIWinderCtrl.winderAxisCtrl.enableAxis   := FALSE;
12     "HMI".HMIWinderCtrl.winderAxisCtrl.executePos   := FALSE;
13     "HMI".HMIWinderCtrl.winderAxisCtrl.executeStop  := FALSE;
14     "HMI".HMIWinderCtrl.winderAxisCtrl.homeAxis     := FALSE;
15     "HMI".HMIWinderCtrl.winderAxisCtrl.jogNegative  := FALSE;
16     "HMI".HMIWinderCtrl.winderAxisCtrl.jogPositive  := FALSE;
17     END_IF;
18
19 // enable travUserInterface
20 "LTrav_Data".travUserInterface["HMI".selectedIndex].AxisCommand.enable
     := "HMI".manualModeSelected;
21
22 // enable winderUserAxisInterface
23 "LConSMC_Data".winderUserAxisInterface["HMI".selectedIndex].
     AxisCommand.enable := "HMI".manualModeSelected;
24
25 END_REGION
```

Výchozí program je navrhnut tak, aby pomocí bloků z knihovny LTrav bylo možné ovládat celý pohon, tedy i bloky Winder z knihovny LConSMC. To vede k podstatnému zjednodušení inicializace, protože informace o např. průměru cívky, způsobu navíjení atd. zadané do bloku LTrav se následně v programu přepíšou i do bloků LCon a není nutné je vyhledávat znova. Bohužel to ale neplatí pro všechny parametry. Jak je vidět z Tab. 2-6, při inicializaci je nutné nastavit tloušťku a šířku materiálu v FC InitTraverser i FC InitWinderVconstant. Důvod je prostý, v případě inicializace osy Traverser se tyto hodnoty zadávají v mm, zatímco v případě osy Winder v m. Je zvláštní, že tyto veličiny nejsou sjednocené. Manuál několikrát v úvodním textu vyzdvihuje výhody standardizace napříč těmito aplikacemi, přesto zde nejsou u některých parametrů jednotná ani jména, ale dokonce ani jednotky.

Jak již bylo zmíněno, tato odlišnost ve jménu parametrů je zde i v případě stylu navíjení zobrazeného na Obr. 1-8 a popsaného v podkapitole 1.2.2. Tento parametr přímo souvisí s osou Winder a pro osu Traverser jde pouze o doplňkovou informaci. V knihovně LConSMC se tento parametr jmenuje Type of Winder, v knihovně LTrav je přejmenován na Winding mode. Při nastavování tohoto režimu později v HMI bylo zjištěno, že parametr, na který se při změně tohoto módu odkazuje vzorový projekt, běh aplikace nijak nemění. Pomocí přechodu do Online režimu a zkoumáním změny parametrů (postup takové diagnostiky je detailně popsán v podkapitole 2.3) se ukázalo, že připravený parametr pro změnu tohoto režimu mění hodnotu z knihovny LTrav, tedy Winding mode. Ta by pak měla předat tuto informaci knihovně LConSMC, protože s tou tento parametr přímo souvisí, a právě parametr Type of Winder mění styl navíjení. [8] Struktura programu je zde bohužel velmi nepřehledná, testováním bylo ale zjištěno, že pravděpodobně dochází připraveným parametrem ke změně Winding mode a tato změna se nijak nepropisuje na hlavní parametr typeOfWinder. Ten tak zůstává nastaven tak, jak je inicializován a případné propojení na zbytek programu je nutné dodělat. Program byl takto upraven, funkce otestována, ale protože nebylo možné kvůli rozsahu programu ověřit, zda s touto změnou počítá i zbytek programu a testovací simulace nepřinesly jednoznačnou odpověď, nebylo s touto funkcí dále pracováno.

Nastavení těchto základních bloků, resp. jejich popis v manuálu je tedy jednou z největších nevýhod standardizované aplikace. V něm je prvních 27 stran věnováno rozsáhlému popisu a definicím používaných proměnných (např. Waiting angle, Displacement angle), výhod a popisu různých funkcí (např. Spike, Stroke) apod. Následuje popis vzorového projektu, který by měl být základem pro naprogramování a přizpůsobení vlastního projektu uživatelem. V něm jsou použity dva základní FB LTrav_AxisCtrl a LTrav_TravCtrl a je velmi stručně na necelých dvou stránkách popsáno, jak je mezi sebou propojit pomocí interní komunikace a kam připojit globální DB. Následuje ukázka vzorového programu v praxi s výstřižky z HMI a dříve popsanými funkcemi.

Problémem je jednak to, že LTrav_AxisCtrl potřebuje mimo vypsané parametry ještě vstup pro uživatelské rozhraní, což manuál nezmiňuje, ale pro jeho funkci je parametr nutný. Nikde se také nepíše o tom, co přesně tyto bloky dělají a lze to zjistit až nahlédnutím do jejich vnitřní struktury, která je celá psaná v SCL. Manuál se omezuje na vysvětlení „ovládá pohyb osy Traverser“ a tedy až zkoumáním kódu je možné zjistit že bloky dohromady zajišťují základní pohyb a synchronizace s osou Winder. Tím ale obsahují pouze malou část finálních funkcí aplikace, naprogramováním aplikace podle tohoto návodu by nefungovaly prakticky všechny funkce popsané v podkapitole 1.2.2.

Mnohem větší problém ale je, že se tyto bloky ve vzorovém programu nikde nevyskytují, aplikace evidentně není vytvořená, aby byla takto programovaná. Kód obsažený v těchto FB je součástí vzorového projektu, je ovšem uvnitř několika jiných FB, jejichž hledání je poměrně náročné. Je tedy zarážející, že manuál komplikovaně definuje proměnné, představuje funkce a vysvětluje bloky a k těmto účelům poskytuje i krátké video (obsahující prakticky totéž jako v text manuálu), následně představuje vzorový program s vysvětlením a ukázkami těchto funkcí, ale pokud zákazník podle tohoto manuálu postupuje, bude muset vyřešit několik chyb znemožňující kompilaci programu jen aby zjistil, že program navíjí základním způsobem bez jakékoli možnosti avizované funkce využít.

Řešením by jednak mohlo být podstatné zjednodušení vzorového projektu tak, aby odpovídal popisovanému nastavení manuálu. Mnohem lepším řešením by však bylo doplnění programovací části manuálu (spíše kompletní přepracování) o popis vzorového projektu tak jak je poskytnut. Součástí by měl být popis (nebo alespoň odkazy na jiné manuály) přibližně odpovídající podkapitole 2.2.2 této práce. Ideálně ovšem mnohem detailnější a doplněný obrázky nastavení, což zde není kvůli rozsahu práce a případně i nedostatku informací možné. Celé toto nastavení je ovšem v aktuálním manuálu shrnuto na necelé dvě stránky, a to pro aplikaci s takto složitou programovou infrastrukturou zdaleka nestačí.

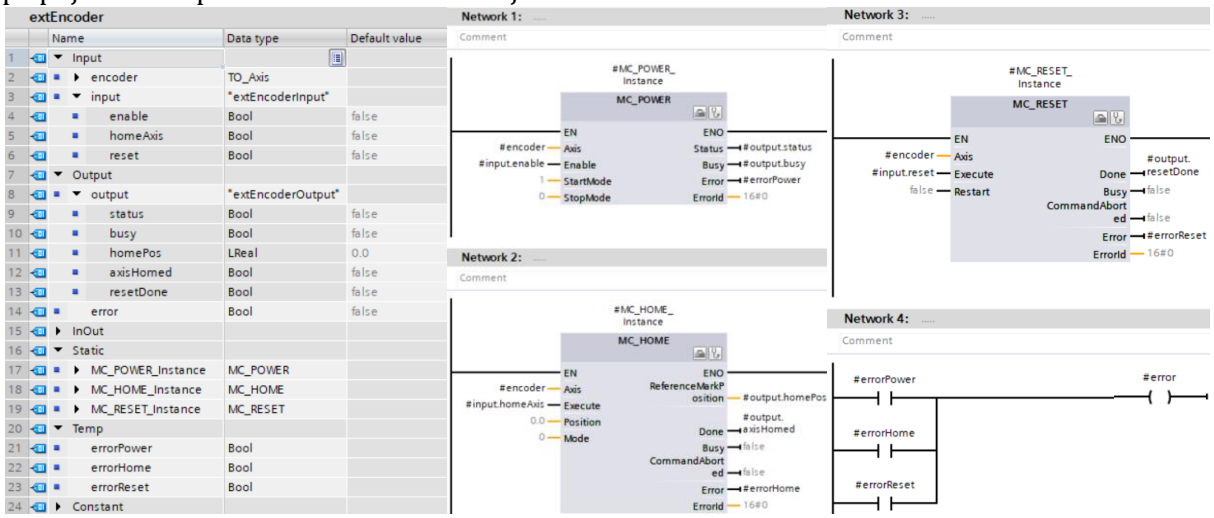
2.2.2.5 Bloky externího enkodéru

S připojením externího enkodéru standardní aplikace nijak nepočítá a kompletní programovou infrastrukturu je tak nutné naprogramovat. Zde byl dodržován formát zavedený standardní aplikací, aby tato funkce přirozeně zapadla do ekosystému těchto vzorových projektů. [14] Vytváření komplikovanějších struktur pro jednodušší zadání, jako je toto, není rozhodně nutné. Pokud je však programová infrastruktura vytvořena univerzálně, lze pak snadno připojit více enkodérů za použití stejného způsobu napojení. Navíc jde o mnohem přehlednější a ohleduplnější způsob programování vzhledem k ostatním programátorům, nebo zákazníkovi.

Po vzoru standardní aplikace je tedy nutné začít vytvářením PLC typů. Pro větší přehlednost byla ve složce PLC types v okně 1 TIA portálu vytvořena další složka (Group) s názvem extEncoder_Types. Označení Types je důležité nejen pro přehlednost, ale také proto, že není možné, aby dva prvky v celém Project tree (okně 1) měly stejný název, ať už jde třeba pouze o složky. Datové typy byly ponechány co nejjednodušší a jsou tedy pouze dva. Typ extEncoderInput obsahuje pouze tři proměnné typu Bool a to `enable` pro spuštění TO, `homeAxis` pro zahájení referencování a `reset` pro vymazaní uložených veličin v TO. Datový typ pro sledování parametrů tohoto TO se nazývá extEncoderOutput a obsahuje čtyři proměnné typu

Bool. Jde o zpětnou reakci na předchozí ovládací prvky: status pro spuštění, `axisHomed` pro referencování a `resetDone` pro reset. Přidán je ještě bit `busy` indikující, že TO pracuje. Poslední proměnnou v tomto typu je `homePos` datového typu LReal, který ukazuje nastavenou referenční pozici.

S takto připravenými proměnnými je možné začít programovat. Za účelem co největší univerzálnosti byl vytvořen obecný FB pro externí enkodér, který následně byl použit. FB má název extEncoder a má dva vstupy, dva výstupy, tři statické proměnné a tři dočasné proměnné. Tímto deklarováním vstupů a výstupů se po přidání této funkce umožní propojit odpovídající datové typy s globálním DB, jak lze vidět na Obr. 2-13. Pak je možné přidat libovolné množství těchto FB a přidáním jiných parametrů (jiných TO, jiných struktur extEncoderInput/Output) řídit stejnou funkcí, stejným způsobem několik různých enkodérů. Vstupem FB je jedna proměnná `encoder` datového typu TO_Axis, která slouží k připojení TO_ExternalEncoder a `input` dříve vytvořeného datového typu extEncoderInput. FB je naprogramován stejně, jako se programují standardní osy pomocí MC bloků, je celý v programovacím jazyku LAD a skládá ze čtyř sítí. Každý MC blok má jako vstup mimo jiné i proměnnou Axis kterou se blok dotazuje, jaké TO bude ovládat. Zde je to vždy nadefinovaná proměnná `encoder` a dále už tedy nebude zmiňována. První obsahuje blok MC_Power pro zapnutí TO. Jeho vstupem je nadefinovaná proměnná `extEncoderInput.enable` a výstupem je proměnná indikující stav TO Status přiřazená `extEncoderOutput.status` a proměnná indikující že TO pracuje Busy přiřazená `extEncoderOutput.busy`. Druhá síť obsahuje blok MC_Home pro zadání referenční pozice. Vstupem je pokyn pro uložení aktuální pozice jako referenční Execute přiřazený `extEncoderInput.homeAxis`, výstupem je indikace dokončení referencování Done přiřazená `extEncoderOutput.axisHomed` a číselná hodnota referenční pozice ReferenceMarkPosition přiřazená `extEncoderOutput.homePos`. Poslední MC blok MC_Reset má jako vstup pokyn k provedení restartu Execute přiřazený `extEncoderInput.reset` a výstup indikaci provedení resetu Done přiřazený `extEncoderOutput.resetDone`. Všechny MC bloky mají také jako výstup hlášení o chybě. Tento výstup je pro každý blok zvlášť převzat jednou dočasnou proměnnou, tedy `errorPower`, `errorHome` a `errorReset`. Tyto všechny proměnné jsou logickým součtem sjednoceny do jedné výstupní proměnné `error`, aby nebylo nutné na výstup připojovat více parametrů. Struktura FB je na Obr. 2-14.



Obr. 2-14 FB extEncoder

Takto vytvořenou funkci je možné vložit do hlavní programové smyčky Main. Protože jde o FB, TIA portál se hned po vložení zeptá jaký DB má tomuto FB přiřadit a jak se bude jmenovat. DB byl pojmenován extEncoder_DB, ale především byl zvolen jako Multi-instance DB. Této funkce zde opět nebude využito, ale pro zajištění univerzálnosti je to velice důležité. Multi-instance DB

umožňuje zapisovat do jednoho DB informace z více FB. Ve vnitřní struktuře DB se vytvoří systém více instancí (listů), které se vzájemně nepřepisují, ale zároveň nevytváří zbytečně další duplicitní DB a šetří tak programovou paměť. Pokud by tak byly např. připojeny dva enkodéry, tak v rámci jednoho DB by jeden mohl být vypnutý, druhý zapnutý a parametr extEncoderInput.enable by se uchoval pro každý enkodér zvlášť. Zavedený FB v OB Main je na Obr. 2-13.

Zbývá jen vytvořit globální DB pro tyto veličiny, ke kterému bude přistupovat zbytek programu. Ten byl pojmenován po vzoru ostatních globálních DB extEncoder_Data, a sem/odsud je možné zapisovat/vyčítat data. Po vytvoření napojení příslušných parametrů je už potřeba pouze vytvořit ovládací rozhraní v HMI, což je popsáno v podkapitole 2.2.3.3.

2.2.2.6 Bloky HMI

Skupina (Group) 4_HMI v programu obsahuje především 3 zásadní PB. Jde o FB HMIControl a jemu přidružený DB InstHmiControl a globální DB HMI. Úkolem funkčního bloku je především zprostředkovat výměnu dat mezi globálními DB os LConSMC_Data, LTrav_Data a globálním DB HMI. Blok bere v úvahu nastavený index pohonu a řídí pouze ten, který je zrovna aktuální. Pokud jakákoli osa, nebo proces hlavní sekvence hlásí chybu, tento blok ihned resetuje povel ke spuštění osy Line a zastaví tak automatické navíjení. Následné regiony obsahují přesně v tomto pořadí bloky výměny dat pro řízení os, Manual mód, kontrolu konzistence dat, změnu parametrů na HMI a nastavení/limitaci čítače vrstev. Přidružený DB je typu Multi-instance, kde každá instance je pro každý nakonfigurovaný pohon. FC se volá v hlavní programové smyčce Main v druhé síti, jak je vidět na Obr. 2-13.

Globální DB HMI obsahuje 12 proměnných, z nichž jsou pouze 3 jednoduché proměnné základního datového typu. Zbylých 9 proměnných jsou složené datové typy, dohromady obsahují přes 120 proměnných. Jde o dvojice řídicích a diagnostických parametrů pro každou osu, tedy pro Traverser HMITravCtrl a HMITravDiag; pro Winder HMIWinderCtrl a HMIWinderDiag a Line HMILineCtrl a HMILineDiag. Zbývající proměnné doplňují data o společných parametrech obou reálných os HMITravConfig a opět dvojice proměnných pro řízení/stav hlavní sekvence HMIMainSequenceCtrl a HMIMainSequenceDiag.

Použití tohoto DB je velmi užitečné, protože sdružuje všechna přenášená data do HMI a při následném vytváření rozhraní je není nutné hledat ve velkém množství dalších DB. V ideální případě by tak nebylo nutné tento DB dále popisovat, ovšem zde se objevilo několik zásadních nevýhod standardizované aplikace.

Není pravděpodobné, že by se to stalo často, ale může nastat situace, kdy uživatel bude chtít do/z HMI přenášet další data, která zde zavedená nejsou. Samotný DB je bez problémů možné rozšířit, ovšem pro propsání parametru přímo do dalších skutečných řídicích/diagnostických DB je nutné předání dat naprogramovat v FB HMIControl v jazyce SCL. Struktura tohoto bloku, byť relativně dobře komentovaná, je bohužel velmi složitá a bez jejího podrobného prostudování je velmi složité najít správný programový cyklus, kam přenos zapsat. Z toho vyplývá i to, že je prakticky nemožné přidat do tohoto bloku externí veličiny, se kterými program nepočítá, jako např. právě funkce externího enkodéru popsaná v předchozí podkapitole. Při pouhém přidání do DB HMI by bylo nutné nejen předání hodnot naprogramovat v FB HMIControl, ale zároveň i ošetřit logiku přepínaní mezi pohony, reakci systému na chybu enkodéru a mnoho dalšího. Hodnoty z DB extEncoder_Data by samozřejmě bylo možné pouze zkopírovat přímo do DB HMI, pak by ale tento blok přišel o svojí funkci pouhého prostředníka při předávání hodnot a v každém z těchto případů by aplikace ztratila kompatibilitu se všemi ostatními podobnými aplikacemi. [14]

Nejzásadnější problém je ale to, že DB HMI je nutné upravit (a porušit tak kompatibilitu) pokaždé, kdy je použit Basic panel. Po přiřazení HMI tagů (jak bude popsáno v následující podkapitole) ve výchozím nastavení s HW konfigurací nastavenou dle podkapitoly 2.2.1 skončí kompilace, třeba i prázdného HMI projektu chybou hlásící, že dvě proměnné jsou datového typu, který nepodporuje komunikační ovladač. Protože komunikace mezi PLC a HMI je standardní NRT komunikace HMI Connection, nelze tuto komunikaci nijak změnit a je tedy nutné upravit

parametry. Z tohoto vyplývá další problém při nutnosti editace výchozích naprogramovaných celků. Dotyčná veličina `cam` je datového typu DB_ANY a vyskytuje se v programu dvakrát pod adresami `"HMI".#HMITravDiag.travActParam.selectedProfileParam.cam` a `HMI".#HMITravDiag.travActParam.activeProfileParam.cam`. Tyto veličiny jsou v programu nezbytné pro správný dopočet bodů pro otočení osy Traverser na koci cívky atd. Datový typ DB_ANY lze změnit na datový typ UInt, který zachová všechny informace a lze ho odesílat/přijímat i přes HMI Connection bez problémů. Ovšem změnit datový typ proměnné cam znamená změnit ho v datovém typu "LTrav_typeProfileParam" a "LTrav_typeProfileParam", protože se v nich vyskytují. A tento změněný datový typ je následně nutné upravit v datovém typu "LTrav_typeActParam" a ten nakonec upravit v datovém typu "LTravHMI_typeTravDiag", který je pak možné vložit do DB. Samozřejmě je nutné následné přetypování ve všech funkcích, které počítají u této proměnné s DB_ANY, protože ačkoliv UInt udrží veškeré hodnoty DB_ANY, má jinou vnitřní programovou strukturu.

Relativně jednoduché řešení, které zde bylo použito, je přidání dalšího datového typu "LTravHMI_typeTravDiag_HMI" a všech ostatních k tomuto typu pomocných datových typů, které jsou přesnou kopií se změněnou proměnou `cam` na UInt. Ta je následně zavedena jako další proměnná v DB HMI, celkový počet proměnných je tedy 13. Protože jde o diagnostický datový typ, všechny tato data jsou výstupem PLC a vstupem pro HMI. Pak je nutné přiřadit všechny veličiny z původního HMITravDiag (datový typ "LTravHMI_typeTravDiag" s proměnnou `cam` DB_ANY) do nového HMITravDiag_HMI (datový typ "LTravHMI_typeTravDiag_HMI" s proměnnou `cam` UInt). K tomuto byla vytvořena speciální FC HMITravDiag_conversion. Funkce obsahuje pouze jeden vstup datového typu "LTravHMI_typeTravDiag" (ve funkci pro přehlednost pojmenován HMITravDiag_DB_ANY) a jeden výstup datového typu "LTravHMI_typeTravDiag_HMI" (interně pojmenován HMITravDiag_UInt). Samotný parametr HMITravDiag obsahuje mimo proměnnou travActParam (jejíž součástí jsou dále právě proměnné cam) ještě další složenou proměnnou a 33 dalších proměnných základních datových typů. V první síti jsou tedy v programovacím jazyku LAD pomocí funkce CONV převedeny obě proměnné z DB_ANY na UInt a následně rovnou uloženy pomocí funkce MOVE do nové proměnné.

Ačkoliv se liší v těchto proměnných pouze jedna hodnota, je bohužel nutné všechna ostatní čísla z proměnných selectedProfileParam a activeProfileParam přepsat postupně, protože mají jiný datový typ a není možné přepsat hromadně pouze část datového typu. Druhá síť funkce pak jednotně přepíše proměnnou travActCheckParam. Postupné přepisování ale platí i pro všech 33 samostatných proměnných, které je nutné také vypsat postupně. Zde se alespoň naskytla příležitost některé tyto hodnoty upravit. Jak bude popsáno v podkapitole 2.2.3.3 u popisu podobrazovky Coil, hodnota relativního umístění osy Traverser `.travPosPercent` je pravděpodobně z důvodu vyšší přesnosti výpočtu 10x vyšší. Podobně hodnoty `.posAAuto` a `.posBAuto` indikující nastavenou pozici A a B pro osu Traverser jsou ze stejného důvodu vynásobeny 100. Vydělením těchto proměnných odpovídajícím číslem pomocí funkce DIV tak je možné získat reálné parametry těchto veličin bez nutnosti jejich úpravy v HMI, tato operace je náplní sítě 5. Protože jde pouze o indikační bloky přenášené do HMI pouze pro zobrazení, nebude s nimi operováno ve výpočtech a tuto změnu tak není nutné zohledňovat v programu. Struktura celé této funkce je na Obr. B-3.

Všechny tyto funkce budou fungovat ovšem pouze v případě, že při odkazování v HMI budou používány výhradně parametry z HMITravDiag_HMI. Pouze jeden parametr, třeba i jiný než parametr `cam`, volaný z HMI z původní proměnné HMITravDiag vytvoří HMI tag celého původního datového typu, jehož součástí je vždy proměnná cam datového typu DB_ANY a HMI přestane fungovat. Použití lepšího a vybavenějšího Comfort panelu ve vzorové aplikaci je z hlediska firmy Siemens pochopitelné a správné. Ovšem přehlédnutí toho, že se základním panelem ve výchozím stavu aplikace nefunguje vůbec by pro zákazníka mohlo být velmi odrazující. Řešení zde popsané,

ačkoliv jde o jednodušší variantu obejití tohoto problému je stále poměrně komplikované, navíc s nevýhodou již zmíněného porušení kompatibility standardní aplikace, což je jedna z jejích hlavních výhod. Zarážející je také to, že manuál aplikace k tomuto problému vůbec nic neříká. I když je Comfort panel psán jako doporučený, přizpůsobení i Basic panelu by bylo (pokud by na něj bylo myšleno od začátku) jednoduché a podstatným způsobem by rozšířilo uživatelskou základnu. Zvláštní je také to, že ani nápověda TIA portálu, ani web technické podpory (SIOS) nemá k dispozici vysvětlení chyby, která při pokusu přenosu proměnné DB_ANY na Basic panel vzniká a nikde není napsáno (manuály, katalogy, nápověda, SIOS), že by tento přenos Basic panel neumožňoval. TIA portál navíc mate uživatele tím, že i při nakonfigurovaném Comfort panelu v HW konfiguraci svítí parametr datového typu DB_ANY červeně jako chybový, nicméně kompilace proběhne bez problémů. Kvůli nedostatku informací bylo nutné tento problém řešit přímo s experty z firmy Siemens.

2.2.3 Návrh HMI

Zde bude popsáno vytváření grafického rozhraní, s jehož pomocí je možné ovládat a sledovat dříve nakonfigurované parametry a veličiny. Rozhraní obsahuje i jednoduchou vizualizaci navíjení. Stejně jako u programu PLC je vhodné pravidelně rozhraní nahrávat do panelu. Program se zkompiluje a varuje před případnými chybami. Při ladění byl použit především simulační program SIMATIC WinCC Runtime Advanced Simulation. Při jeho použití není potřeba mít panel fyzicky zapojený, simulátor vytvoří jeho virtuální kopii v prostředí Windows. Při zapojení všech ostatních reálných komponent je možné z tohoto softwarového panelu ovládat reálný FM/PLC, nebo jejich simulace stejným způsobem, jako na reálném panelu. [5]

Původní rozhraní ve vzorovém projektu bylo vytvořeno pomocí 12" Comfort panelu. Ten nemá žádná fyzická tlačítka, místo nich má 12 tzv. Softkeys (SW tlačítka) F a dalších 12 Softkeys Shift+F. S pouhými čtyřmi fyzickými tlačítky na zde použitém 4" Basic panelu a dalšími omezeními tak bohužel nebylo možné se o připravenou ovládací a vizualizační strukturu opřít a bylo nutné rozhraní kompletně přepracovat. TIA portál umožňuje ovládací prvky přizpůsobit jiné velikosti, nebo panelu. Přesnost automatického škálování je však velmi omezená a většinu funguje lépe při přechodu z menšího na větší panel (ne naopak, jako je tomu v tomto případě). Zde sice TIA portál dokázal zmenšit některé ovládací prvky, ale jejich pozice byla většinou nesprávná, prvky velmi malé a komplice selhávala kvůli použitým funkcí Comfort panelu, které Basic panel nemá. Protože byl použit nejmenší 4" panel, bylo často obtížné vměstnat na ovládací obrazovku všechny potřebné ovládací prvky. Bylo tak nutné upravit některé výchozí funkce panelu, např. funkci zrcadlení fyzických tlačítek (popsáno v podkapitole 2.2.3.1), nebo často pracovat se systémem vrstev obrazovek popsaným níže (popis nastavení vrstev je v podkapitole 2.2.3.3). Při případném nasazení této aplikace do reálného provozu jde tak pro zákazníka vyžadujícího použití Basic panelu o další velký zásah do výchozího programu. To je další nevýhoda standardizované aplikace, která velkou mírou spoléhá na použití Comfort panelu.

Při prvním spuštění panelu je možné, že bude potřeba aktualizovat jeho FW. Pak se na něm objeví jeho základní FW menu. Zde je možné spustit nahrané grafické rozhraní, je zde i kolonka nastavení. V továrním nastavení je panel nastaven tak, že při každém dotyku vydá potvrzující tón. To nemusí být vhodné do každého prostředí, zde byla tedy tato funkce zakázána.

TIA portál nabízí hned po přidání jakéhokoli panelu spuštění jednoduchého průvodce vytvářením grafického rozhraní. Je zde možné rozvrhnout strukturu jednotlivých obrazovek, přidat běžné ovládací prvky, jako např. roletu pro přechod mezi obrazovkami, lze přidat funkce jednotlivým tlačítkům ať už softwarovým, nebo i hardwarovým a spoustu dalších funkcí. Zde nebyl tento průvodce použit, protože v době prvního testování rozhraní ještě nebylo plně navrženo. Pro většinu případů však průvodce poslouží pro dobré základní rozvržení rozhraní a výrazně zjednodušší přechody mezi obrazovkami, které je jinak nutné dělat ručně.

Před popisem a vytvářením obrazovek je potřeba v HMI definovat měněné parametry pomocí HMI tagů. Může jít o interní tagy, které HMI vyhodnocuje samo (a vůbec se přes komunikační

rozhraní nepřenáší) a samotné PLC tagy, na které HMI naopak pouze odkazuje. Tyto tagy se definují v okně 1 ve složce HMI, podsložce HMI tags. Podobně jako v adresářích PLC, i zde je možné vytvořit podložku pomocí tlačítka Add new tag table a zachovat tak program přehledný. Zde byly vytvořeny 2 tag table. Jedna pro interní tagy Traverser_intern a jedna pro PLC tagy SIMATIC_Traverser. Druhý jmenovaný lze rovnou nastavit, protože je zde potřeba pouze přidat všechny položky z DB HMI, tedy celkem 12 tagů. Zobrazeny jsou na Obr. B-2 v Příloze B. To je jedna z velkých výhod standardizované aplikace, kde není potřeba řídicí tagy dohledávat ve složitém systému DB a PLC typů, ale předají se jednotně celému DB HMI s předpřipravenými datovými typy. Protože je ale použit Basic panel, NESMÍ však být přidána proměnná HMITravDiag, ale místo ní musí být použita proměnná HMITravDiag_HMI, jak bylo popsáno v předchozí podkapitole. Po jejich přidání (např. přetažením) je dobré zkontrolovat, zda program skutečně přiřadil tagy jako externí z PLC v kolonce Connection. Zde by mělo být pouze jedno připojení HMI_Connection a musí odpovídat správnému PLC, pokud jich je v projektu více. V případě problémů lze HMI_Connection upravit ve Složce HMI, podsložce Connection. Posledním důležitým parametrem u HMI tagů je Acquisition cycle, který určuje s jakou periodou bude HMI žádat/aktualizovat tyto tagy. Výchozí nastavení 1 s je vyhovující pro diagnostické tagy, ale zcela nevyhovující pro řídicí tagy, protože by stroj reagoval na povely se znatelným zpožděním. Protože HMI nebylo zatěžováno větším množství tagů a přenosová rychlost byla dostatečná, byly všechny tagy nastaveny na minimální hodnotu, tedy 100 ms. Interní PLC tagy byly postupně přidávány dle potřeby během vytváření rozhraní a budou tedy popsány v následujících kapitolách.

Kliknutím pravým tlačítkem na složku HMI v okně 1 a zvolením Properties se následně dostaneme do informačního okna HMI. Zde, v záložce General, menu Information lze mimo jiné najít informace o množství využitých power tagů. To, že je připojeno přes PLC pouhých 12 složených tagů neznamená, že se do limitu 800 tagů počítají pouze ty. Jako power tag se myslí každý tag na proměnnou základního datového typu. S takto připravenou aplikací je celkový počet tagů 128, což je bez problémů pod hranicí maximálního počtu.

Součástí vzorového projektu je také několik položek v Text and graphic lists, které je vhodné do projektu přidat. Tyto listy přiřazují text IO poli, nebo obrázky grafickému poli podle hodnoty přiřazeného tagu. Zde nebyly použity všechny, některé byly upravovány a další byly přidávány.

Před vytvářením rozhraní je nutné znát systém vrstev používaný v TIA portálu. Jsou zde celkem čtyři typy obrazovek, každá s jinou prioritou zobrazení. Zde jsou seřazeny podle priority zobrazení sestupně, na Obr. 2-15 jsou vyobrazeny graficky. [24]

- System screen

  Obrazovka s nejvyšší prioritou. Tuto obrazovku není možné žádným způsobem nastavovat. Slouží např. k zobrazování zadávacích polí, klávesnic, kurzorů a pro menu FW.

- Global screen

  Tuto obrazovku lze modifikovat jen velmi málo. Lze zde umístit indikaci oznámení o aktivních chybách. Pozici a velikost těchto objektů lze měnit, jejich funkčnost ne. Zde se dají také konfigurovat hardwarová tlačítka panelu, pokud jimi disponuje. V rámci této a všech ostatních typů obrazovek lze využívat až 31 dalších vrstev, čehož bylo často využíváno. Její nastavení je popsáno v podkapitole 2.2.3.1.

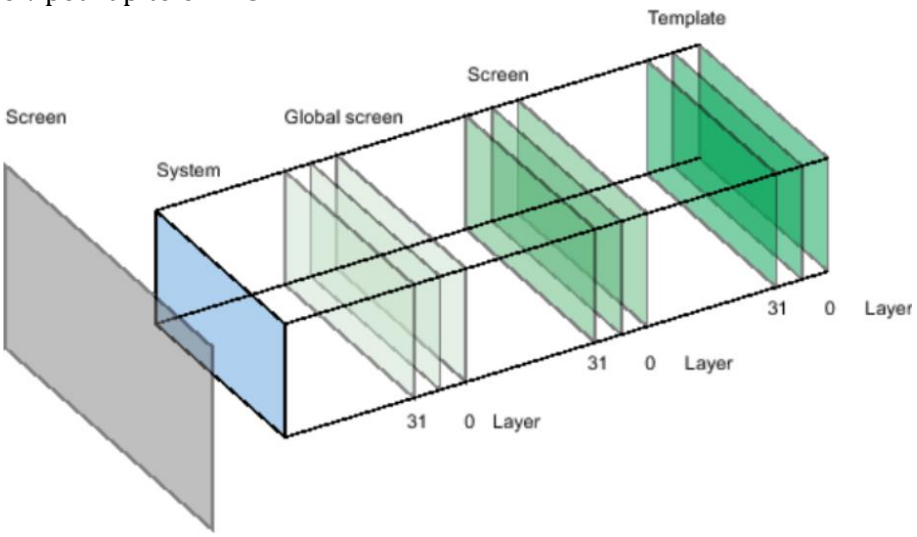
- Screen

  Jde o hlavní náplň grafického rozhraní. Lze sem zobrazovat mnoho různých objektů, jako např. zadávací a textová pole, vizualizační objekty, grafy apod. Jejich nastavení je popsáno v podkapitole 2.2.3.3.

- Template

  Tato obrazovka zůstává vždy v pozadí. Je velmi užitečná např. pro tlačítka, která se objevují na více obrazovkách. Pomocí obrazovky Template (šablona) není nutné tlačítka kopírovat do

každé další obrazovky, ale pouze jí přiřadit jednou vytvořenou šablonu. Její nastavení je popsáno v podkapitole 2.2.3.2.



Obr. 2-15 Viditelnost obrazovek HMI [25]

Jako poslední před návrhem rozhraní je dobré zmínit, že při přidávání objektů je nezbytně nutné, aby text, grafika, nebo jiný objekt nepřesahoval hranice pracovní plochy. Některé objekty to totiž umožní a zvlášť při optimalizaci využití veškerého volného prostoru na malém displeji může k přesahu dojít. Objekt se ovšem pak nezobrazí vůbec, HMI pracuje jako kdyby nebyl přidán. V nápovědě TIA portálu o tomto není zmínka a je tak možné, že jde o chybu tohoto vydání TIA portálu, resp. WinCC. Při programování indikátorů, které se nezobrazují vždy a zároveň jsou velmi blízko okrajům displeje může tato skutečnost programátora poměrně jednoduše zmást.

2.2.3.1 Global screen

Ke Global screen se v TIA portálu přistupuje v okně 1, složce s HMI, podsložce Screen management. Ve výchozím nastavení obsahuje indikátor alarmů společně s třemi okny alarmů. Jsou zde okna pro aktivní alarmy, čekající alarmy a alarmy, které ještě nebyly kvitovány. Všechna okna zde byla ponechána a bylo upraveno pouze jejich rozložení. To je velmi výhodné, protože v případě alarmu je takto zajištěno, že se alarm zobrazí nehledě na to, co je právě zobrazeno na displeji. Global screen se vždy dostane do popředí.

Zde je také nutné nastavit HW tlačítka panelu. 4" model, který byl zde použit má celkem čtyři. Na tato tlačítka je vhodné namapovat funkce, které se používají velmi často a lze je tedy pohodlně stisknout opakovaně s jistotou mechanické odezvy. Zároveň je dobré použít takové funkce, které je možné použít v rozsahu celého programu. Není žádoucí, aby mechanické tlačítko, které je samozřejmě vidět neustále nehledě na vykreslené obrazovce, měnilo parametr ukrytý hluboko v systému obrazovek. Pak by totiž bylo možné jeho náhodné stisknutí obsluhou bez jejího zaznamenání této skutečnosti. Pokud už je nutné takovouto funkci mechanickému tlačítku přiřadit, je vhodné jeho stav a případnou změnu ovládaného bitu indikovat buďto na panelu za každých okolností, nebo ještě lépe pomocí DI na PLC (kontrolkou přímo na PLC, nebo externí signální LED). Poslední vhodnou funkcí pro tato tlačítka je funkce, kterou by mohlo být obtížné udělat pomocí dotykového panelu, např. kalibrace displeje.

Vybrané funkce, které byly přiřazeny fyzickým tlačítkům jsou zobrazeny na Obr. 2-16. Zleva jde o tlačítka:

- F1

Slouží k rychlému přechodu na obrazovku 0_Home. Tato jednoduchá funkce se mu přiřadí po kliknutí na něj v okně Global screen a následně se v záložce Properties, Events nastaví funkce ActivateScreen s parametrem 0_Home při události Press key.

- F2

Přepíná mezi AUTO a Manual módem aplikace. Oba tyto režimy jsou popsány v podkapitole 1.2.2.1. Stisknutím tlačítka (opět událost Press key) se invertuje bit `"HMI_VAR.manualModeSelected"` pomocí funkce InvertBit. Přestože program sám při přechodu z AUTO módu resetuje jeho povely (jak je vysvětleno v podkapitole 2.2.2.4), laděním bylo zjištěno že tato změna může přijít příliš pozdě a program pak zastaví hlavní sekvenci chybou synchronizace 3106_8001. [3] Proto je toto tlačítko samo ještě doplněno o zastavení synchronizační osy (ResetBit `"HMI_VAR.HMILineCtrl".startLine`) a hlavní sekvence (ResetBit `"HMI_VAR.HMIMainSequenceCtrl".startMainSequence`). Nakonec je zde ještě deaktivováno pomocné řídicí okno z obrazovky 4_Visualization pomocí ResetBitInTag `panelVisualization`.

- F3

Slouží k zastavení obou pohonů. Podobně jako F2 zastavuje synchronizační osu i hlavní sekvenci, zároveň ale také vysílá povel executeStop na obě osy a zároveň resetuje bit executePos u obou os. Takto je zajištěno, že se pohon zastaví nehledě na to, v jakém módu je zrovna provozován. Vše opět po události Press key. V reálné aplikaci by samozřejmě bylo vhodné mít další fyzické bezpečnostní STOP tlačítko a zde není snahou ho tímto tlačítkem na panelu nahradit. Jde tak spíše o jednoduché zastavení bez nutnosti se proklikávat dalším nastavením k účelům ladění aplikace.

- F4

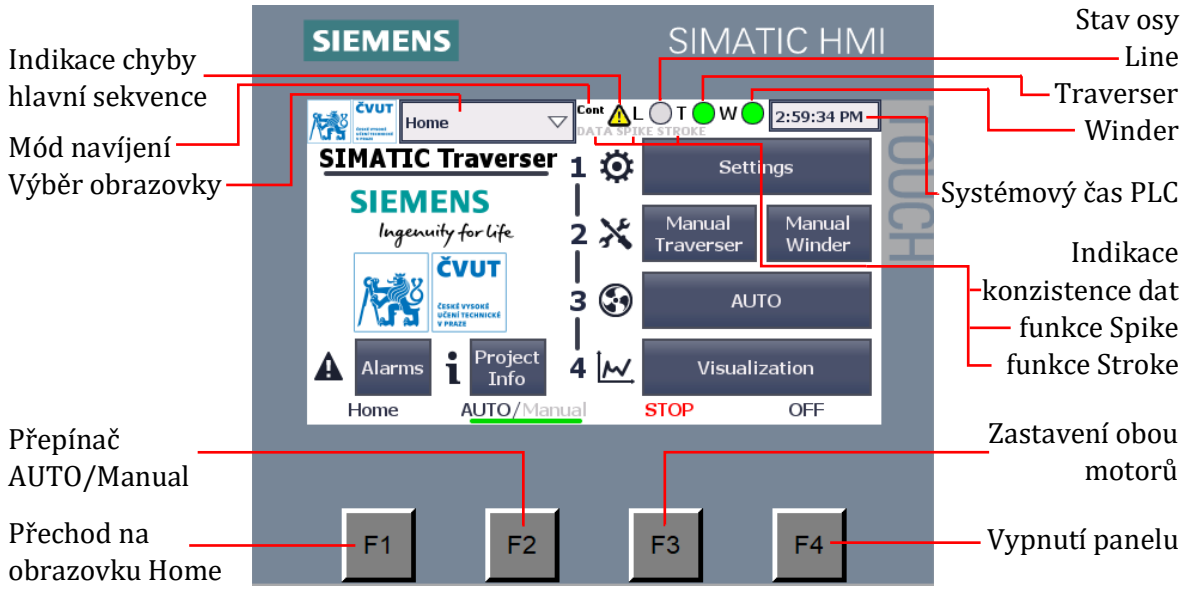
Poslední fyzické tlačítko F4 pak slouží k vypnutí panelu a přechodu do jeho FW menu. Povel StopRuntime se opět provede na událost Press key.

Kvůli již zmíněné prostorové limitaci, co se velikosti obrazovky týče, bylo nutné upravit výchozí nastavení fyzických tlačítek. Ta jsou totiž v továrním nastavení zrcadlena na dolní část dotykové obrazovky. Oblast zrcadlení se v editoru zobrazí po kliknutí na příslušné tlačítko, stisknutí fyzického tlačítka tak odpovídá zároveň stisknutí této oblasti na dotykovém panelu. Protože by však toto zrcadlení tlačítek na panel plýtvalo cenným místem na obrazovce, je vhodné toto nastavení změnit. Samotné zrcadlení nelze úplně zakázat, lze ale zmenšit zrcadlící plochu panelu na minimum. To se provede v okně 1, Složce HMI, Runtime settings, záložka Screens, Function keys zaškrtnutím User-defined pictogram size. Zde je pak možné velikost zrcadlení změnit, nejmenší hodnota je ovšem 16 px. Zde tedy byla ponechána hodnota x rovna 64 a hodnota y nastavena právě na minimální hodnotu.

2.2.3.2 Template

Template, neboli šablony jsou přístupné ze stejné složky okna 1 jako Global screen. Jde o velmi užitečný nástroj pro správu obrazovek. Grafika, text a ovládací prvky na obrazovce Template jsou zobrazeny v úplném pozadí obrazovky. Lze zde vytvořit základní tlačítka, ikony, indikátory a další grafiku, která bude jednotná pro více dalších ovládacích obrazovek. [5] Tato šablona se pak obrazovce nastaví pravým kliknutím, Properties, záložka Properties, General v poli Template.

Velikost zde použitého panelu nedovoluje efektivně využít výrazně odlišné šablony. Při nastavování parametrů i při běhu je naopak vhodné vidět všechny zásadní informace najednou. I proto byla v tomto projektu použita pouze jedna šablona pro všechny obrazovky. Skládá se pouze z horního a spodního informačního pásu, oba jsou tedy vždy na každé obrazovce zobrazovány. Z toho vyplývají i informace které jsou v nich zobrazené. Vytvořená šablona je na Obr. 2-16 vyobrazena na ovládací obrazovce 0_Home. Indikátory použité v šabloně a jejich umístění jsou zde také popsány. Součástí použité šablony je při popisu zleva nahoře následující:



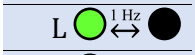
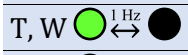
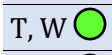
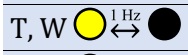
Obr. 2-16 Obrazovka Home s fyzickými tlačítky

- Logo ČVUT.
- IO pole pro zadání a zpětné zobrazení aktuální zobrazené ovládací obrazovky nastavené jako vstupně/výstupní. Vypisované stránky odpovídají text listu TextList_ScreenNames a odpovídají tagu Tag_ScreenNumber. Podrobněji je struktura obrazovek popsána v následující podkapitole.
- Indikátor zvoleného navíjecího módu popsaného v podkapitole 1.2.2.2.
- Indikátor chyby hlavní sekvence. Přestože měněný parametr mainControlState má mnoho dalších nadefinovaných stavů, zde se zobrazuje pouze chybový stav. Je totiž možné, že nastane chyba v hlavní sekvenci bez toho, aby byla chyba v jakékoliv jiné ose, typicky např. při nesprávném referencování. Obsluha by tak nemohla spustit stroj, ale žádná osa přitom nehlásí chybu. Ve vzorovém projektu je chyba hlavní sekvence indikována pouze na jedné obrazovce slovem ERROR v černé barvě a standardní velikostí písma, což může být snadno přehlédnuto, proto byla zde zvolena jasnější indikace vedoucí k rychlejšímu zjištění příčiny.
- Stav os Line (L), Traverser (T) a Winder (W). Indikátor má několik barev pro několik různých stavů, popsány jsou v Tab. 2-7. Indikátory pro osu T a W jsou vytvořeny dle stejné logiky, ale s parametry pro odpovídající osu, např. parametr axisEnabled je barevně značen pro obě osy stejně, ale pro T je odpovídající parametr axisEnabled osy T a podobně pro osu W.
- Indikace konzistence dat (DATA). Některé parametry aplikace nelze změnit za běhu. Pokud jsou za běhu změněny, tak je jejich nové nastavení ignorováno, ale je uloženo do doby, než jsou data re-inicializována. Na tuto nekonzistenci používaných a nastavených dat upozorňuje tento indikátor, který se při nekonzistenci zbarví do červena.
- Indikace funkcí Spike a Stroke. Tyto indikátory se zbarví do černé barvy při zadání nenulových hodnot těchto funkcí, a tedy jejich povolením.
- Systémový čas PLC. Slouží pro rychlou orientaci např. při zjišťování okamžiku výskytu chyby, kterou PLC zaznamená právě dle tohoto času.
- Popisky fyzických tlačítek na dolní liště. Přepínač AUTO/Manual indikuje aktuálně používaný režim.

Podrobnější popis významu indikátorů je v Tab. 2-7. Ke každému indikátoru je zde vypsán parametr, jehož změnu reflektuje. Není zde popsána celá cesta k parametru (byla by pro tabulku příliš dlouhá), ale lze ho podle označení snadno dohledat. Ve třetím sloupečku je pak hodnota tohoto parametru při tomto zobrazení a její význam. Poslední čtyři řádky tabulky, tedy ikonky

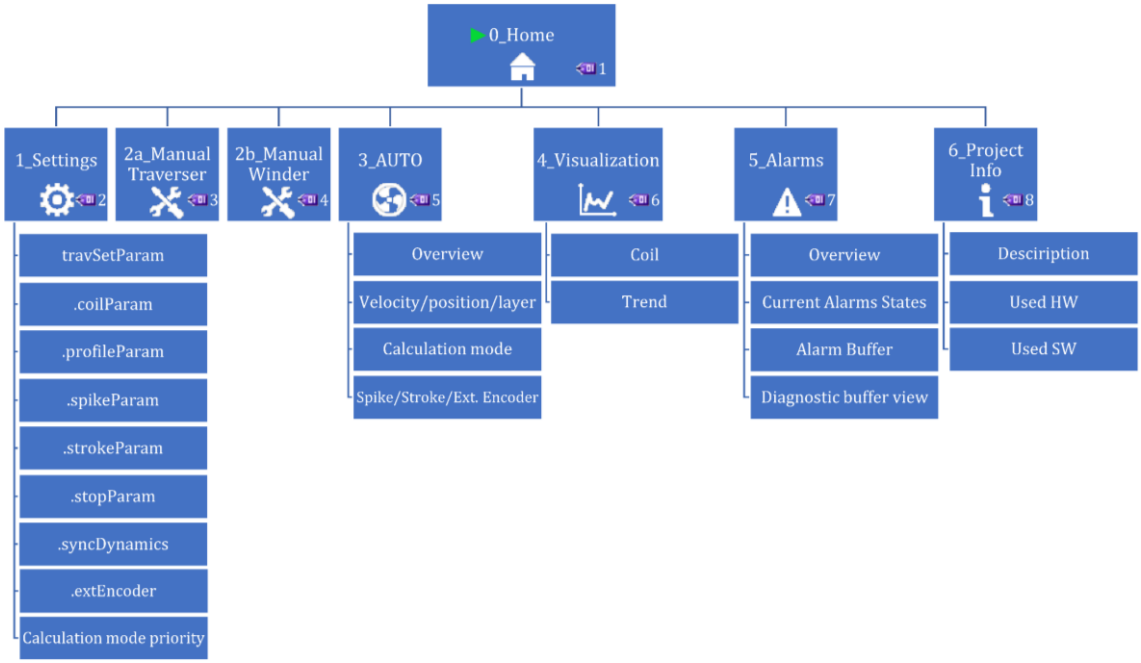
motorů nejsou součástí šablony, ale využívají se pouze v obrazovce 4_Visualization, podobrazovce Coil. Zde jsou pouze z důvodu ucelenosti vypisovaných indikátorů.

Tab. 2-7 Ikony v HMI a jejich význam

Ikona	Parametr	Hodnota a význam
Cont	traversingMode	0 → Continuously (spojité navíjení)
Step		1 → Step wind (krokové navíjení)
	mainControlState	# → mainControlState
		900 → ERROR (chyba)
L 	lineControlState	0 → IDLE, manualModeSelected (nečinný, Manual mód)
L 		60 → READY_TO_START (připraven)
L 		80 → LINE_IN_MOTION (běh)
L 		100 → LINE_STOPPING (brzdění)
L 		900 → ERROR (chyba)
T, W 	axisEnabled	0 → disabled (zakázána)
T, W 		1 → enabled (povolena)
T, W 	startLine	1 → startLine (spuštěna)
T, W 	manualModeSelected	1 → manualModeSelected (Manual mód)
T, W 	axisError	1 → error (chyba)
DATA	setParamEqual	0 → FALSE (data nejsou konzistentní)
DATA		1 → TRUE (data jsou konzistentní)
SPIKE	spikeParam.lengthAcc	0 → spikeNotUsed (Spike nulový, nebo zakázán)
SPIKE		# → spike (Spike nenulový, používán)
STROKE	strokeParam.stroke	0 → strokeNotUsed (Stroke nulový, nebo zakázán)
STROKE		# → stroke (Stroke nenulový, používán)
AUTO/Manual	manualModeSelected	0 → manualModeSelected (AUTO mód)
AUTO/Manual		1 → manualModeSelected (Manual mód)
	manualModeSelected	1 → manualModeSelected (Manual mód)
	TWControlState	0 → IDLE (nečinný)
		200 → MODULES_IN_OPERATION (běh)
		900 → ERROR (chyba)

2.2.3.3 Ovládací obrazovky

S připravenými fyzickými tlačítky z Global screen a šablonou je možné začít tvořit hlavní náplň HMI. Při vytváření uživatelského rozhraní bylo postupováno dvěma různými přístupy. Nastavování parametrů na obrazovce 1_Settings a manuální pohyb s osami na obrazovkách 2a_Manual Traverser/2b_Manual Winder je vytvořeno přímočařeji s předpokladem, že zadávající osoba programu rozumí, nebo má k ruce manuál [3], [8], [9], popř i tuto práci. Vytvoření nápovědy by bylo možné, s přihlédnutím k velikosti displeje by však musela být také velmi stručná a obsluha by se pravděpodobně stejně nevyhnula referenci s manuálem. Ostatní obrazovky se snaží být maximálně uživatelsky přívětivé a přehledné, ovládací prvky jsou co možná největší pro snadné dotykové zadávání. Program obsahuje 7 základních obrazovek, některé obrazovky mají více podoken. Přehledně je struktura obrazovek zobrazena diagramem na Obr. 2-17. Pro zachování přehlednosti byla podokna základních obrazovek vytvořena v systému vrstev, a ne jako samotné obrazovky. Jejich vytváření je popsáno v popisu obrazovky 1_Settings.



Obr. 2-17 Struktura ovládacích obrazovek

Protože, jak již bylo zmíněno, systém obrazovek nebyl vytvořen pomocí průvodce při úvodním nastavení HMI, bylo nutné obrazovky propojit. Ve složce tagů Traverser_intern byl tedy vytvořen interní tag Tag_ScreenNumber datového typu DInt, který určuje číslo aktuální aktivní obrazovky. Každá obrazovka pak musí při aktivaci tuto hodnotu změnit na jemu odpovídající. Zvolené číslování je zobrazeno na Obr. 2-17 u malé ikonky tagu. Změna se provede v okně 4 na požadované obrazovce ve chvíli, kdy není označen žádný jiný parametr. V záložce Properties, Events při události Loaded lze vybrat funkci SetTag, vybrat tag Tag_ScreenNumber a hodnotu adekvátně nastavit.

Zde bude stručně popsána funkce a vlastnosti všech obrazovek. Jejich číselné pojmenování odpovídá systému zavedenému ve vzorovém projektu.

- 0_Home

Domovská obrazovka 0_Home je hlavním rozcestníkem pro všechny další obrazovky. Mimo loga ČVUT a Siemens tedy obsahuje pouze velká tlačítka s nastaveným přechodem na jinou obrazovku. To se provede v záložce Properties, Events, při události Click funkcí ActivateScreen s příslušnou stránkou. Naznačené číslování odpovídá číslům z Obr. 2-17 a navádí tak obsluhu ke správné posloupnosti nastavování dle Obr. 2-23. Samotná obrazovka je na Obr. 2-16 v předchozí podkapitole, nebo na Obr. B-4 v Příloze B.

- 1_Settings

Z hlediska množství nastavitelných parametrů jde o nejsložitější obrazovku. Vzorový projekt obsahuje nastavení přibližně 44 parametrů na jedné hlavní a jedné pomocné obrazovce. Protože zde bylo snahou ukázat co největší množství funkcí standardní aplikace, byly zde na jedné obrazovce zahrnuty téměř všechny tyto parametry.

Na 4" obrazovku by samozřejmě nebylo možné zobrazit všechny parametry najednou, bylo zde tedy využito celkem 8 vrstev (Layers) společně s jednou pro externí enkodér. Tento způsob vytváření rozhraní má tu výhodu, že není nutné vytvářet velké množství samostatných obrazovek ve složce Screen okna 1 což zpřehledňuje program. Zároveň to zjednodušuje management obrazovek, protože není nutné hlídat větší množství čísel obrazovek pro tag Tag_ScreenNumber.

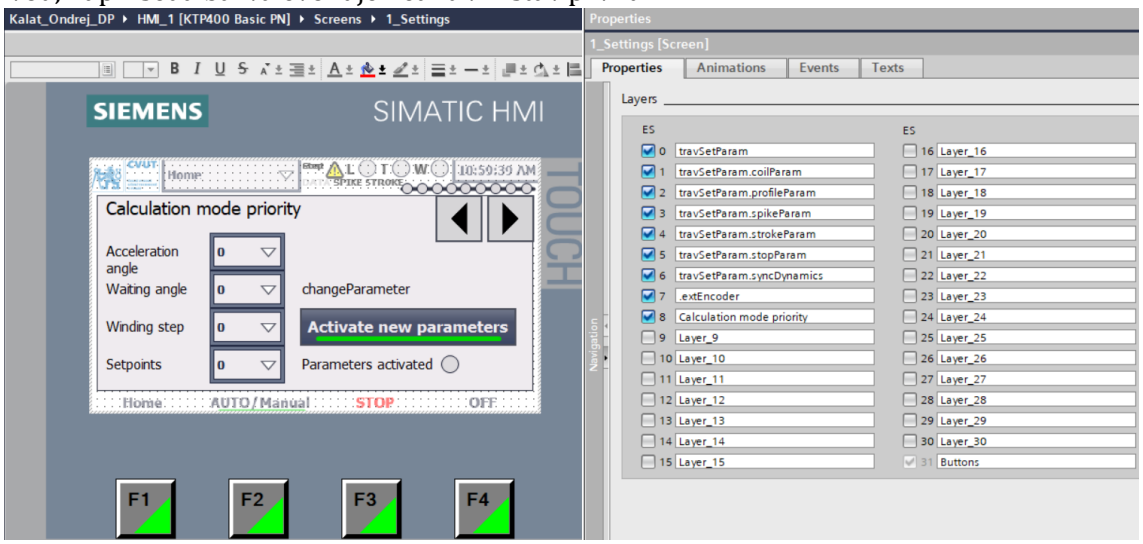
Každá z těchto vrstev tak má jako základ obdélník, na kterém jsou příslušné ovládací prvky. Po jejich vložení je jim možné mimo dalších nezbytných parametrů nastavit v okně 4, záložce Properties, Miscellaneous, v boxu Layer vrstvu, ve které se prvky budou zobrazovat. Po zvolení

samotné obrazovky, tedy ve chvíli, kdy není vybrán žádný ovládací prvek, lze přepnout momentálně zobrazovanou vrstvu. Na Obr. 2-18 je toto menu zobrazeno v pravé části, v levé části je zobrazeno editační okno s aktivovanými vrstvami 0 až 7 a 31. Vrstvy lze pro každou obrazovku individuálně přejmenovávat, což program dále zpřehledňuje a usnadňuje editaci.

Při programování rozhraní bylo zjištěno, že je velmi výhodné nejdůležitější ovládací prvky přesunout rovnou na vrstvu 31 a mít ji neustále aktivovanou. Jak bylo ukázáno na Obr. 2-15, maximální počet vrstev je 31, s tím že právě 31. vrstva je nejvíce v popředí. Tak je tedy zajištěno, že ovládací prvky, jako např. tlačítka pro přechod mezi vrstvami nebudou nikdy překryta a programátor s nimi vždy počítá, nehledě na to, jaká vrstva je momentálně upravována. Na Obr. 2-18 jsou tedy v levé části v editoru viditelná tlačítka na přepínání vrstev a lišta s indikací zobrazené vrstvy, protože jsou obě součástí vrstvy 31. Dále je v editoru vidět obdélník menu Calculation mode priority, který je součástí vrstvy 7. Všechny další použité vrstvy 0 až 6 jsou sice aktivované, ale překrývá je vrstva 7 která má nejvyšší prioritu. Ovládací prvky z podkapitoly 2.2.3.2 jsou součástí šablony a nejsou odsud editovatelné, proto jsou částečně průhledné.

Velmi důležité je také to, že systém vrstev je pouze nástrojem zjednodušení editace. Po spuštění panelu, ať už v simulaci, nebo na reálném HW, se tak vždy budou zobrazovat prvky ve vrstvě, která je nejvíce v popředí. Mezi vrstvami nelze přepínat pomocí žádného tagu, z hlediska Runtime jde o interní funkci, která z něj není přístupná.

Na těchto obrazovkách je přepínaní mezi vrstvami řešené tak, že všechny prvky jednotlivé vrstvy jsou viditelné pouze při určité hodnotě nastaveného interního tagu. To se nastaví u každého prvku v okně 4, záložce Properties, Animations, Display, přidáním funkce Visibility a nastavením příslušného řídicího tagu. Zde je to tag menuSettings, jehož hodnota 0 až 8 určuje zobrazenou vrstvu. Je nutné správně nastavit, jestli při zadané hodnotě tagu (nebo rozmezí hodnot) bude objekt viditelný, nebo právě při této hodnotě zmizí. Tuto funkci lze přidávat hromadně více objektům najednou. Na Obr. 2-18 tedy např. všechny prvky boxu Calculation mode priority mají nastaveno, že jsou viditelné pouze pokud je tag menuSettings roven 7. Ovládání hodnoty tagu menuSettings je realizováno přes dvě tlačítka s šipkami, které mají jako Event při události Click nastavenu funkci IncreaseTag, resp. DecreaseTag. Jednoduše lze omezit rozsah změněných hodnot tagu přidáním podmínky viditelnosti pro tato tlačítka. Tlačítko pro zmenšení tagu se tedy stane neviditelným, pokud je tag menuSettings roven 0 (nelze ho tedy už zmáčknout a tag dále snížit) a podobně pro tlačítko zvyšování hodnoty tagu a hodnotu 8. Omezení, se kterým se musí v tomto případě počítat spočívá v tom, že nelze již používat animace Visibility pro jednotlivé prvky, protože ke každému objetu je možné přiřadit pouze jeden animační Visibility tag. Toto lze obejít např. tím, že požadovaný text, grafika, nebo jiný objekt nezmizí, ale bude zbarven jinou barvou, např. šedá barva evokuje neaktivní stav prvku.



Obr. 2-18 Přepínání vrstev (Layer)

Zde budou krátce popsány všechny vrstvy, tedy podobrazovky 1_Settings. Jak již bylo zmíněno, zde se předpokládá základní znalost programu, nebo možnost odkázat se na manuál. Měněné parametry jsou zde tedy označeny zpravidla tak, jak jsou naprogramovány. Nejsou podrobněji vysvětlovány, což šetří místem, ve většině případů to neubírá na přehlednosti (symbolické názvy dobře reflektují význam měněného parametru) a zjednodušuje to případnou referenci právě k manuálu, kde jsou také používány tyto symbolické názvy proměnných. Všechny podobrazovky jsou zobrazené na Obr. B-4 v Příloze B.

- travSetParam

  Názvem odkazuje na parametr `"HMI".#HMITravConfig.travSetParam` v DB HMI, který samotný obsahuje pouze dva parametry pro změnu navíjecího módu `traversingMode` a změnu inicializace dat po startu `enableBehavior`. Doplněna je ještě parametrem rozhodujícím o počátečním směru navíjení `travStartDir` a způsobu změny parametrů za běhu `parameterChangeMode`. Nakonec je zde ještě možnost re-inicializace parametrů přes parametr `changeParameter` a zpětná kontrola provedení tohoto úkonu `setParamEqual`. Parametr `.travSetParam` ovšem dále obsahuje 6 dalších proměnných složeného datového typu, které jsou náplní dalších šesti podobrazovek.

- .coilParam

  Obsahuje 12 parametrů vztahujících se k základním parametrům cívky, jako je např. pozice bodu A a B, jak je definován konec cívky, nebo informace o navíjeném materiálu. Podle nastavení parametru `.coilMode` se také správně zobrazují jednotky u parametrů `.coilAngA` a `.coilAngB`. Byl zde vynechán parametr `externDiameter`, protože externí výpočet průměru cívky se u této aplikace nepředpokládá.

- .profileParam

  Obsahuje 9 parametrů, které společně definují Motion profile definovaný v podkapitole 1.2.2. Tři z těchto parametrů, konkrétně `accelerationAngle`, `waitingAngle` a `windingStep` jsou datového typu LTrav_typeSetBorderValue. Ten vždy obsahuje další tři parametry `min`, `set` a `max`, a je použit pro zjednodušení, protože v aplikaci je potřeba více parametrům zadávat minimální, maximální a nastavenou hodnotu. Pro zjednodušení je upravovaná veličina v této aplikaci vždy ta nastavovaná, tedy `set` a ostatní jsou nastaveny při inicializaci programu bez možnosti změny přímo v HMI. Pokud je nastaven `.traversingMode` na spojité navíjení, jsou parametry vztahující se ke krokovému navíjení zašedlé a naopak.

- .spikeParam

  Obsahuje 5 parametrů nutných pro správné nastavení funkce Spike popsané v podkapitole 1.2.2.3. Dvě z těchto proměnných jsou opět datového typu LTrav_typeSetBorderValue a opět se v programu nastavuje pouze hodnota `set`.

- .strokeParam

  Obsahuje pouze dva parametry pro nastavení funkce Stroke popsané v podkapitole 1.2.2.4.

- .stopParam

  Obsahuje tři proměnné, které umožňují upravit brzdný profil osy Traverser. Ve výchozím nastavení jsou ve všech polích hodnoty -1, což značí že se používají výchozí hodnoty zadané v příslušném TO. U zadávacích polí je ve výchozím stavu ikona TO def., která značí tento stav a v případě jiné hodnoty u zadávacího pole, než -1 zašedne.

- .syncDynamics

  Obsahuje čtyři proměnné, které umožňují upravit dynamiku synchronizace os. Podobně jako u předchozí podobrazovky jsou zde parametry s indikátorem TO def. fungující na stejném principu.

- .extEncoder

  Obsahuje tři parametry vztahující se k TO_ExternalEncoder, konkrétně povolení bloku (MC_Power), referencování (MC_Home) a reset (MC_Reset), vše s indikátorem.

- Calculation mode priority

  Obsahuje čtyři parametry, které jsou součástí parametru `.modePriority` a nastavují tak prioritu výpočtu požadované hodnoty. Obsahuje také druhé tlačítko pro aktivování a re-inicializaci nových parametrů.

- 2a_Manual Traverser

Po prvním spuštění této obrazovky je vyplněna hláškou o nutnosti přepnutí na Manual mód, protože program je ve výchozím nastavení připraven v režimu AUTO. Informační box šipkou naznačuje, kterým tlačítkem se mód přepíná. Toto přepínání může zdánlivě být zvlášť při ladění zbytečným zdržováním, protože lze tyto módy přepínat automaticky. Podobně jako při nastavování tagu s číslem obrazovky by bylo možné na událost obrazovky Loaded nastavit změnu `.manualModeSelected` a takto je HMI naprogramováno např. u vzorového programu. Tato změna ale byla ponechána na uživateli jednak proto, že přepnutí se provede pomocí tlačítka F2 velmi jednoduše bez nutnosti změny aktuální obrazovky, ale především aby obsluha věděla a měla vždy kontrolu nad tím, kdy se v jakém módu aplikace nachází. Nechtěné přepnutí z Manual do AUTO módu a následné automatické spuštění stroje ve chvíli, kdy je např. prováděna výměna cívky by mohla vést k poškození stroje, technologie, navíjeného materiálu a u větších strojů i ke zranění obsluhy.

Jde o jednoduchou obrazovku pro manuální ovládání osy Traverser. Slouží k snadnému pohybu pro referencování osy (homing) a jejímu vyzkoušení. Zároveň zde lze nastavit pozici A a B podle aktuální fyzické pozice osy Traverser pomocí `.saveAsPosAAuto` a `.saveAsPosBAuto`. Není tak nutné odměřování a nastavování definice začátku cívky. Obrazovka prakticky sdružuje všechny prvky standardních MC bloků pro řízení osy v TO. Obsahuje tedy povolení osy `.enableAxis` s indikací `.travAxisEnabled`, což odpovídá bloku MC_Power. Bez povolení osy není možné zadávat pokyny k pozicování a referencování, na což program opět upozorňuje informačním boxem.

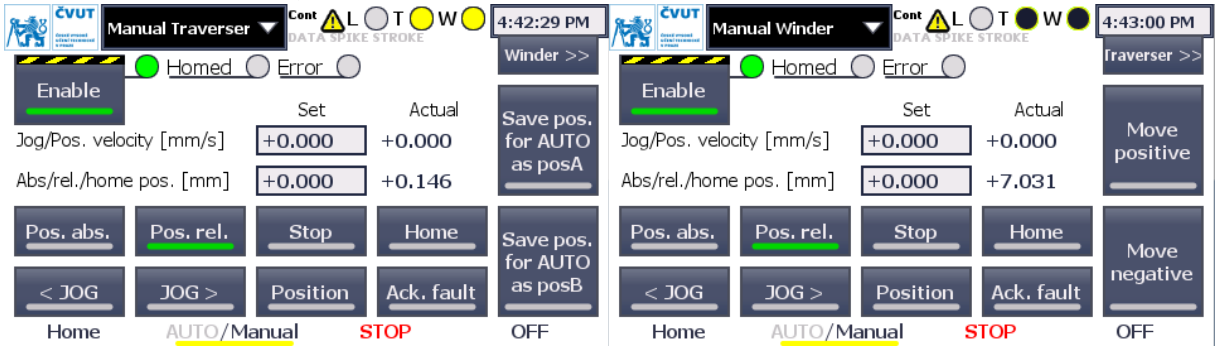
Dále je zde možnost relativního a absolutního pozicování, včetně zastavení, referencování a JOG pohybu, které odpovídají blokům MC_MoveRelative, MC_MoveAbsolute, MC_Halt, MC_Home a MC_MoveJog. Standardně se každému tomuto bloku přiřazuje vlastní rychlost a vlastní pokyn k vykonání. V rámci snahy ušetřit místo a ovládací prvky jich bylo zde několik ušetřeno. Obrazovka obsahuje dvě zadávací pole. Jedno pro rychlost `.posJogVelocity` a jedno pro pozici `.absRelHomePosition`. Už z názvů parametrů je zřejmé, že zde zadaná rychlost je jednotná pro JOG i oba typy pozicování, pozice je zase jednotná pro oba typy pozicování i referencování. Mód pozicování se nastaví pomocí přepínačů `.posAbsolute`, kde na tlačítku Pos. abs. je přiřazena funkce SetBit k tomuto parametru a na tlačítku Pos. rel. ResetBit. Jako výchozí se tedy nastaví relativní pozicování. Tlačítko Position pak vykoná aktuálně nastavený pohyb pomocí parametru `.executePos`, tlačítko Home zase osu referencuje pomocí `.homeAxis` a tato skutečnost je indikována pomocí `.travAxisHomed`.

Tlačítko Stop zastaví pohyb tím, že resetuje parametr pozicování a naopak nastaví parametr `.executeStop`. To odpovídá bloku MC_Halt. Odsud je také možné nastavit parametr pro reset chyby pomocí `.ackFault`, který resetuje chyby ve všech předchozích blocích najednou. Stejně tak je zde indikace chyby `.travAxisError`, která indikuje jakoukoliv chybu osy. Nakonec je zde

ještě tlačítko pro rychlý přechod na obrazovku 2b_Manual Winder. Obrazovka je zobrazená níže na Obr. 2-19, nebo na Obr. B-5 v Příloze B.

- 2b_Manual Winder

  Obrazovka kopíruje nastavení minulé obrazovky, všechny hodnoty jsou zde ale vztaženy k ose Winder. Rozdíly jsou pouze ve třech tlačítkách. Zde nedává smysl ukládaní pozice A a B, navíječ to nevyžaduje. Tato tlačítka jsou tedy nahrazena tlačítky Move positive a Move negative, která fungují podobně jako JOG a jsou realizovaná i stejným parametrem `.jogPositive` a `.jogNegative`. Rozdílem zde je, že v klasickém JOG módu se osa pohybuje, dokud je stlačeno tlačítko a docíleno je toho tak, že pokyn SetBit je přiřazen události Press a ResetBit hned události Relase. Zde jsou ovšem obě funkce přiřazeny události Click a po prvním kliknutí tedy bit zůstane nastaven a resetuje se až po dalším kliknutí. Poslední rozdíl spočívá v tom, že tlačítko s rychlým přechodem na obrazovku nyní zobrazí 2a_Manual Traverser. Společně s ní je také tato obrazovka zobrazena na Obr. 2-19, nebo na Obr. B-5 v Příloze B.



Obr. 2-19 2a_Manual Traverser, 2b_Manual Winder

- 3_AUTO

  Podobně jako u předchozích obrazovek, i tato je v Manual módu překryta informačním boxem, který upozorňuje na nutnost přepnutí do módu AUTO. Po jeho aktivaci se kromě první podobrazovky, která bude popsána níže, objeví čtyři základní ovládací tlačítka, která jsou vidět napříč všemi podobrazovkami. Tlačítko Enable modules na událost Click invertuje bit `.startMainSequence` pomocí funkce InvertBit. Aby se předešlo okamžitému rozběhu pohonu, je zde zároveň vždy resetován bit `.startLine`. Tento parametr je nutné aktivovat až právě druhým tlačítkem Line START/STOP. Obě tato tlačítka mají výstražné zvýraznění, protože jejich aktivací je možné rozběhnout pohon a je tedy nutné dodržet bezpečnost provozu. Tlačítko Enable modules se zároveň zbarví do žluta a text do červena, pokud je zjištěna nekonzistence dat popsaná v předchozí podkapitole. To upozorňuje obsluhu na to, že pohon běží s touto nekonzistencí, ale zároveň na možnost aktualizace dat pomocí vypnutí a zapnutí modulů, protože pak dojde k re-inicializaci a není nutné přecházet na obrazovku 1_Settings. Přepínání mezi podobrazovkami je provedeno podobně jako na obrazovce 1_Settings, tag pro změnu vrstev se jmenuje `menuAUTO`. Všechny podobrazovky jsou zobrazené na Obr. B-6 v Příloze B, dvě vybrané níže na Obr. 2-20.

  - Overview

    Obsahuje základní diagnostická data ohledně hlavní sekvence a všech os. Zároveň se zde nastavuje rychlost osy Line.

  - Velocity/position/layer

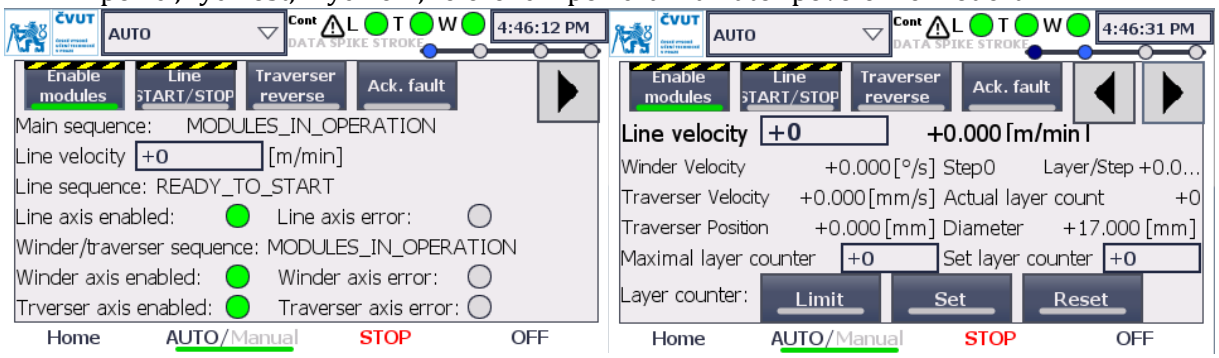
    Zobrazuje rychlost a pozici os. Lze zde také vyčítat a nastavovat limit navíjených vrstev, zadat ručně jejich počet anebo jejich počet resetovat. Společně s předchozí podobrazovkou je na Obr. 2-20.

  - Calculation mode

    Zobrazuje informace o Motion profile, včetně pozic A a B a informacích o navíjeném materiálu.

- Spike/Stroke/Ext. Encoder

  Pokud jsou funkce Spike a Stroke použity, zde se zobrazí jejich nastavená i okamžitá velikost. Zároveň obsahuje informace naměřené externím enkodérem, tedy aktuální pozici, rychlost, zrychlení, referenční pozici a indikátor povolení enkodéru.



Obr. 2-20 3_AUTO: Overview, Velocity/position/layer

- 4_Visualization

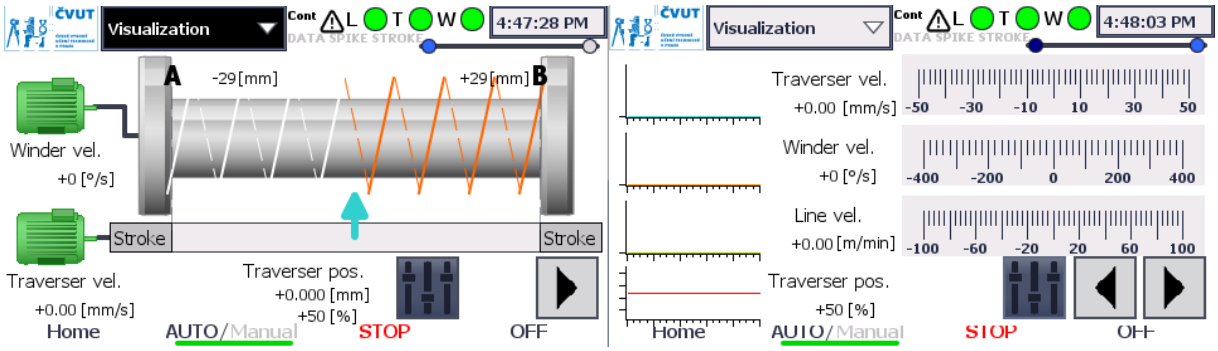
Tato obrazovka má pouze dvě podobrazovky. Zobrazují se zde vybrané informace z předchozí obrazovky v grafické podobě. Sledovat případné změny po nastavení parametrů, nebo zvýšení rychlosti by ovšem bylo poměrně složité, kdyby to bylo potřeba udělat na jiné obrazovce. Proto je zde na obou podobrazovkách možné vyvolat pomocný panel. Ten obsahuje všechna čtyři základní tlačítka z obrazovky 3_AUTO a pole pro zadání rychlosti osy Line. Tak je možné alespoň základní obsluhu zařídit přímo z této obrazovky. Panel nelze používat v Manual módu a při přechodu z AUTO do Manual módu se vždy sám zavře. Podobrazovka Trend s pomocným panelem je zobrazena na Obr. B-7 v Příloze B, podobrazovka Coil se vyskytuje ve videu v Příloze C, obě pak jsou níže na Obr. 2-21.

- Coil

  Zobrazení cívky s pozicí a navíjením vychází z grafiky vytvořené pro výchozí program. [3] Pro potřeby tohoto programu by ale musela být přepracována. Obrázek je samozřejmě menší, počet grafických závitů je pouze 15. Ve výchozím programu je zbarvení jednotlivých závitů z bílé (průhledné) na oranžovou (viditelnou) podmíněno parametrem `.travPosPercent`, kde 100 % se rovná hodnotě 1000. Tento přepočet byl použit pravděpodobně pro větší přesnost při vykreslování a vyhodnocování, jak již bylo popsáno v podkapitole 2.2.2.6 u FC HMITravDiag_conversion. Zde byl ale tento parametr vydělen 10 a hodnota přímo odpovídá procentům. Tento parametr zároveň řídí animaci jezdce Traverseru pomocí animace Move horizontaly. Maximální animovaný Stroke je 15 %. Systémem překrývání vrstev je pak signalizováno navíjení dopředu a zpět. O viditelnosti směru navíjení rozhoduje parametr `.travLayerFWD`, v dopředném směru tento bit signalizuje že je závit viditelný, ve zpětném směru naopak. Obrazovku doplňují informace o nastavených hodnotách pozice A a B přímo u cívky, i zde je hodnota zobrazována v jiných hodnotách a pro získání skutečné hodnoty bylo potřeba vydělit původní parametry 100. Dále pozice zobrazena číselně i v procentech, rychlost obou os a dvě ikonky motorů, které indikují jejich stav dle Tab. 2-7.

- Trend

  Obsahuje opět informace o rychlosti všech tří os a pozici osy Traverser. Pro všechny osy vykresluje HMI samostatný graf, pro první tři zároveň graficky zobrazuje velikost na plnícím se obdélníku. Velikost je přizpůsobená maximálním výchozím hodnotám, v případě jejich změny by muselo být toto nastavení upraveno. Obě obrazovky jsou na Obr. 2-21.



Obr. 2-21 4_Visualization: Coil, Trend

- 5_Alarms

  Obrazovka sdružuje alarmová a diagnostická okna. Zobrazena je na Obr. B-7 v Příloze B.

  - Overview

    Obrazovka zobrazuje stav hlavní sekvence `.mainControlState`, indikuje chyby všech tří os, tedy i osy Line .lineAxisError a pro osy Traverser a Winder zobrazuje případně i kód aktivní chyby `"HMI_VAR.HMITravDiag_HMI".outErrorIdFBTC`, resp. `"HMI_VAR.HMIWinderDiag".error`. Pokud není chyba v žádné ose, ale v některém z FB, příslušný kód se také ukáže zde přes `"HMI_VAR.HMITravDiag_HMI".outErrorIdFBSub`. Odsud lze také všechny chyby kvitovat. Podobrazovka je zobrazena na Obr. 2-22.

  - Current Alarms States

    Pomocí okna Alarm view jsou zde zobrazeny všechny aktivní alarmy, tedy chyby a varování PLC a FM, systémové chyby, kvitované i nekvitovné. I odsud lze vybrané chyby kvitovat a k základním chybám lze zobrazit nápovědu.

  - Alarm Buffer

    Pomocí stejného okna Alarm view ukládá všechny alarmy, jako v předchozí podobrazovce, ovšem zde se ukládají i již neaktivní alarmy. HMI přiřadí každé chybě časovou značku, což může pomoct obsluze při případné diagnostice pohonu.

  - Diagnostic buffer view

    Toto okno využívá stejně nazvaný objekt, který mimo alarmy zobrazuje i kdy byly kvitovány, v jakém stavu se HMI, PLC a pohon obecně nacházel, jsou zde ukládány systémové hlášky (např. připojení/odpojení HMI k/od PLC) apod.

- 6_Project Info

  Obrazovka doplňuje HMI o tlačítka na kalibraci displeje a zvýšení a snížení jeho jasu. Všechny podobrazovky jsou zobrazené na Obr. B-8 v Příloze B, dvě vybrané pak níže na Obr. 2-22.

  - Desciription

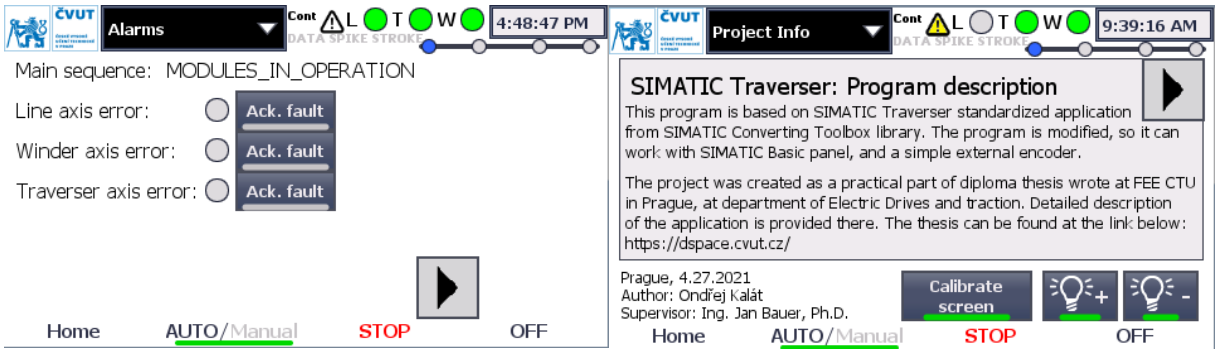
    Zde je uveden stručný popis aplikace. Podobrazovka je zobrazena na Obr. 2-22.

  - Used HW

    Zde je uveden použitý HW, odpovídá Tab. 2-1.

  - Used SW

    Zde je uveden použitý SW, odpovídá Tab. 2-2.



Obr. 2-22 5_Alarms: Overview, 6_Project Info: Description

2.3 Používání aplikace

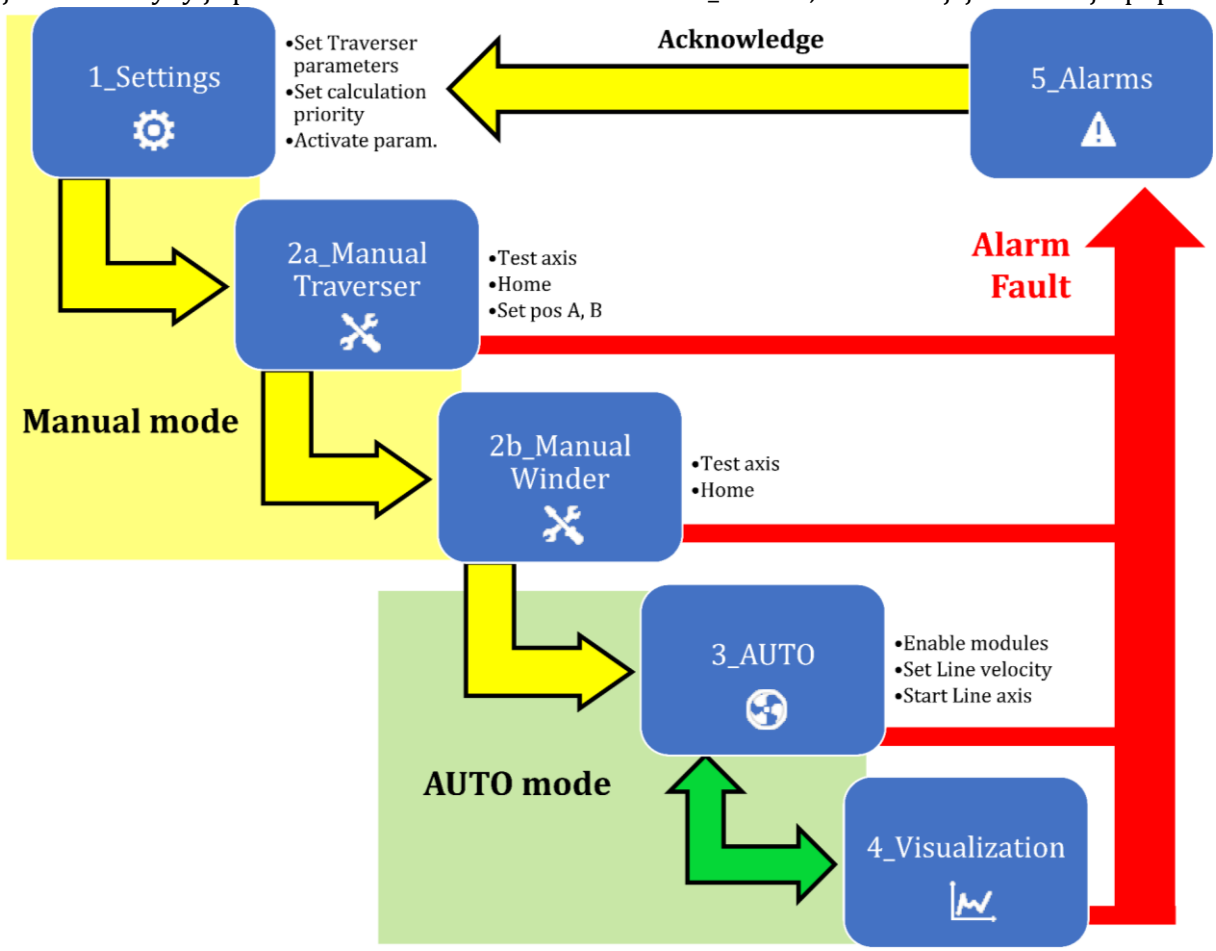
Vytvořený program byl kompletně vyzkoušen v simulacích a následně otestován na reálném HW pomocí vzdáleného přístupu i standardně přímo u stroje. Výsledkům testování se věnují následující kapitoly. Z hlediska prvního nahrání programu se postup při simulaci, i na reálném HW zásadně neliší, bude tedy popsán zde. K nahrání projektu je potřeba mít připravený TIA portál verze 15.1, nebo vyšší, na notebooku se standardním ethernet RJ-45 konektorem v případě nahrávání do reálného HW. V případě simulací je pak nutné pouze zahájit simulace PLC a HMI tlačítkem zobrazeným na Obr. 2-7 na str. 22, což spustí v samostatném Windows okně PLCSIM v případě PLC a WinCC Runtime v případě HMI. U reálného HW je potřeba se připojit ethernet kabelem do posledního volného slotu P2 na FM Traverser (jak je zobrazeno na Obr. 2-1 na str. 15), resp. připojit se k VPN v případě vzdáleného přístupu. Pak lze program ve všech případech nahrát do PLC a HMI pomocí tlačítka zobrazeného opět na Obr. 2-7 a aplikace je takto připravena k použití.

Vzhledem k velkému množství parametrů v programu nebylo možné, ani vhodné všechny zobrazit na HMI. Je tedy možné, že některé z nich bude potřeba sledovat přes TIA portál. Ten obsahuje velmi široké možnosti diagnostiky parametrů a zde budou krátce popsány. U téměř každé obrazovky okna 1 ze složky PLC lze přejít do Online módu, přechod se provede pomocí tlačítka Go Online zobrazeném na Obr. 2-7. Vytvoří se tím acyklická NRT komunikace PC/PG Connection mezi PC a skutečným/simulovaným PLC, umožňující PC sledovat o zobrazovat skutečné parametry vypočítané PLC.

Přechod do tohoto módu také přímo podmiňuje spuštění určitých oken. Jde o okna Commissioning (uvedení do provozu) která jsou k nalezení v okně 1, jednak u složek s pohony (vytvořenými při HW konfiguraci, pracuje pouze s reálným HW) a také u všech TO obsahujících pohyblivou osu (lze i simulovat). Tyto obrazovky slouží k základnímu vyzkoušení os mimo program (tedy i mimo Manual mód) a obsahují podobné rozvržení. Vždy je zde hlavní tlačítko Enable, které u TO v podstatě odpovídá bloku MC Power a u složek měničů přepínači režimu ovládání HAND. Oba tyto režimy se především vyznačují tím, že po jejich aktivovaní dostanou přednost před jakýmkoliv signálem z hlavního programu. Nefungují tedy žádné funkce, např. nastavené nouzové zastavení, reverzace, standardní brzda. Vše se ovládá pouze z tohoto panelu, dokud se tento režim neukončí a řízení se předá zpět hlavnímu programu. Jde o velmi jednoduchý a rychlý způsob, jak rychle ověřit funkci os, zároveň ale z této logiky může vyplynout několik nebezpečných situací, na které TIA portál upozorňuje. Jednak jde o to, že pokud by byl např. zadán pokyn v tomto módu pro běh vpřed a následně vypadlo spojení PC a PLC, nebo FM, nebylo by možné FM zastavit žádným způsobem jiným, než např. odpojením napájení měniče. To je stručně obsahem výstražné hlášky, která se objeví před povolením HAND módu/TO. V tom samém dialogovém okně je možné nastavit si Monitoring time [ms], který tomuto předchází. Pokud je ztracena komunikace, tak měnič vyčkává tento čas (označovaný sign of life) a pokud do té doby není obnovena, tak zastaví měnič chybou F01030 a předá řízení zpět PLC. Nutné je, aby obsluha byla obezřetná i při opouštění tohoto módu. Předání řízení PLC totiž proběhne okamžitě a pokud

jsou splněny podmínky PLC k běhu, např. PLC bylo spuštěno před zahájením tohoto režimu, je toto nastavení ihned obnoveno a stroj se rozběhne. Na možný okamžitý start stroje s výzvou zachování bezpečnosti upozorňuje při zákazu tohoto módu další hláška. Ovládací rozhraní pak prakticky kopíruje nastavení obrazovek 2a_Manual Traverser a 2b_Manual Winder popsaných v podkapitole 2.2.3.3.

Po vyzkoušení os je možné spustit připravený program. Na Obr. 2-23 je schematicky nakreslená hlavní programová smyčka z hlediska uživatele tak, jak bude procházet jednotlivými obrazovkami. Samotné jejich pořadí tomu napovídá. Na obrazovce 1_Settings se nastaví požadovaná aplikace (body A, B, funkce Spike, Stroke atd.) a nové parametry se potvrdí. Na obrazovkách 2a a 2b se může osa ještě jednou vyzkoušet, určitě je však nutné zde provést referencování (homing). Tyto úkony se provádějí v Manual módu, po jejich nastavení je už možné přejít do AUTO módu. Zde se na obrazovce 3_AUTO nastaví rychlost osy Line a aplikace je připravena k běhu. Na obrazovce 4 je pak možné sledovat detailněji průběh navíjení, v případě jakékoliv chyby je pak možné odkázat se na obrazovku 5_Alarms, kde bude její detailnější popis.



Obr. 2-23 Standardní programová smyčka

V případě, že by bylo nutné sledovat další parametry, které nejsou na HMI zobrazeny, např. při ladění aplikace, je možné v TIA portálu v režimu Online sledovat přímo hodnotu zadaných parametrů v jednotlivých DB, lze sledovat i průběžné parametry a aktivované logické sítě ve všech OB, FB a FC. Velmi užitečná jsou také předdefinovaná diagnostická okna FM a TO, kde je zobrazen výpis základních hodnot, např. přímo parametry z FM, enable z MC_Power apod. V základním stavu je kvůli šetření výkonu a datové propustnosti PC a PLC i v online režimu tento mód vypnutý, zapíná se vždy stejným tlačítkem s ikonkou brýlí a zelené šipky v horní části Okna 2.

Poslední velmi užitečnou diagnostickou funkcí je mimo Trace, který bude podrobně představen v následující podkapitole také Watch tabulka. Jde o tabulky, ve které je možné si

přehledně zobrazit vybrané vyčítané parametry z jakéhokoliv PB, TO, nebo PLC tagu. Tabulku lze doplňovat komentáři a v případě simulací zde lze i měnit jejich hodnoty.

2.3.1 Simulace

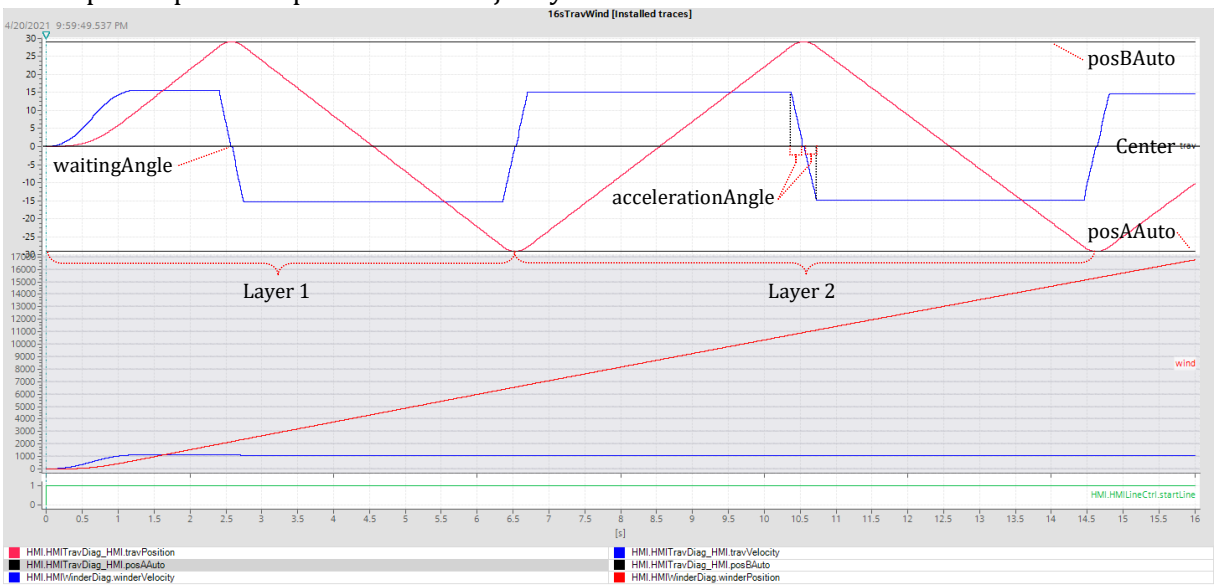
TIA portál umožňuje pořizovat záznamy jakéhokoliv tagu, parametru FM, nebo proměnné vytvořené v programu. Této funkci se říká Trace a jde o velmi mocný nástroj pro testování simulací, nebo diagnostiku reálného stroje. Přistupuje se k němu v okně 1, složce PLC, podsložce Traces. Po přidání nového Trace je možné si ho pojmenovat a zachovat tak přehlednost programu. Po jeho otevření se v okně 2 zobrazí okno se dvěma záložkami. V záložce Configuration je potřeba Trace nejdříve nastavit. Přidají se zde samozřejmě sledované parametry. Následně se určí vzorkovací perioda, typicky jde o nějaký násobek průchodu hlavní programovou smyčkou Main. Lze určit i dobu nahrávání (počet vzorků). Trace je možné spustit manuálně, nebo je možné v menu Trigger nastavit automatické spuštění dle určeného bitu. Lze navíc nastavit událost bitu, na kterou Trace zareaguje (1, 0, náběžná hrana, sestupná hrana, jakákoliv změna) a dokonce i nastavit Pre-trigger, kdy Trace zaznamená určený počet vzorků i předtím, než je spouštěcí podmínka splněna. Poslední možností je ukládat nahraný průběh rovnou na paměťovou kartu.

Nahrání Trace se provede tlačítkem v horní části okna. V tu chvíli přejde prostředí do Online režimu a je tedy nutné mít už připojen zaznamenávaný HW (v tomto případě jeho simulace). Ve stejné části okna je pak možné zahájit nahrávání, v případě nastaveného Triggeru tím Trace zahájí vyčkávání na požadovanou událost triggeru. S Trace je možné dále pracovat po jeho nahrání, jako např, upravení rozsahu, popisků a barvy os, přizpůsobení časové osy, a hlavně přidělování škálovacích skupin, kde proměnné ve stejné skupině budou vykresleny v jednom grafu. Vykreslené grafy lze pak ukládat jako obrázky ve formátu .bmp, uložit lze i konfiguraci Trace pro další měření a také uložit samotný Trace do složky Measurements ve stejné podložce okna 1 pro pozdější úpravy.

Pro ukázku funkce programu v simulacích byl vytvořen Trace s názvem 16sTravWind a z něho vytvořena čtyři měření pro ukázku nejdůležitějších funkcí. Sledováno bylo sedm veličin: rychlost a pozice osy Traverser a Winder, aktuální vypočítaná pozice A a B a bit spuštění osy Line, který zde zároveň slouží jako trigger s nastaveným pre-triggerem 5 vzorků. Obrázky jsou vždy stejně rozvržené. Rychlost [mm/s] (modrá) a pozice [mm] (červená) osy Traverser, společně s pozicí A a B [mm] (obě černé) jsou součástí jedné škálovací skupiny trav zobrazené v grafu nahoře. Rychlost [°/s] (modrá) a pozice [°] (červená) osy Winder má samostatný graf uprostřed. Vykreslen je pro informaci i trigger [-] (zelená) dole. Zaznamenán je vždy maximální možný počet 15886 vzorků při záznamu každého průchodu Main, tedy přibližně 16 s. Systém byl vždy před začátkem záznamu referencován (začíná se vždy od nuly) a nastaven na výchozí parametry vypsané v Tab. 2-6. Ve všech případech je rychlost osy Line nastavena na 10 m/s, Winding step je nastaven na 5 mm/ot. To znamená celkem velké mezery mezi jednotlivými závity, nastaveno je to ovšem proto, aby se za 16 s stačila navinout alespoň jedna vrstva a bylo možné pozorovat chování pohonu při změně směru vedení naviječného materiálu. Některé další parametry byly změněny a na ty bude upozorněno u každého Trace zvlášť.

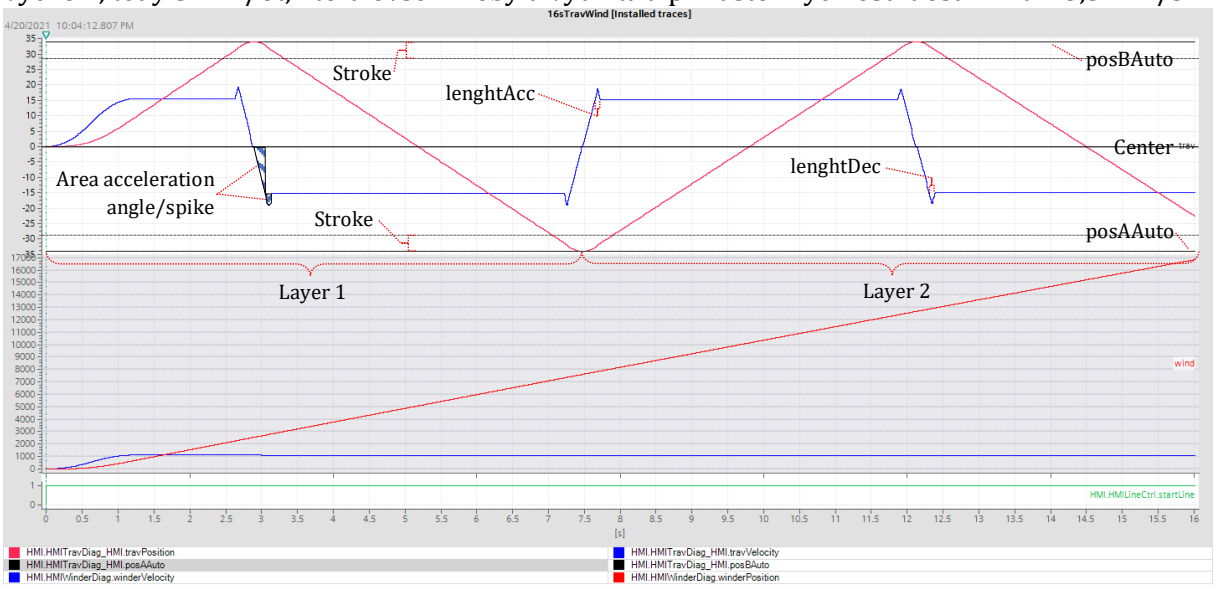
Na Obr. 2-24 je zobrazen průběh sledovaných veličin prakticky ve výchozím stavu. Navíjecí mód je nastaven jako spojitý, nejsou použity žádné přidané funkce, a hodnoty zrychlení a rychlosti jsou také ponechány výchozí. Jde tak o nejjednodušší způsob navíjení. Hodnota pozice po referencování je ve výchozím stavu nastavena na střed cívky. Červený průběh pozice osy Traverser ukazuje, že systém skutečně začal naviječ z prostředka cívky a pokračuje směrem k bodu B, protože jak je popsáno v podkapitole 1.2.2 u Traversing cycle, bod A má vždy nižší hodnotu, než bod B. Jelikož je nastavena délka reálné cívky 58 mm, pozice bodů A a B ±29 mm tomu odpovídá. Pozice obou bodů zobrazené černě zůstávají konstantní a rámují tak červený průběh pozice osy Traverser. U rychlosti je zřejmý podstatně pomalejší první rozběh, než jsou následné změny rychlosti na koncích cívky. Ten je způsoben nastavením TO pro opatrný rozběh a případnou reakci obsluhy na nesprávně nastavené mezní body. Na krajích cívky se pak rychlost

mění dle nastaveného Acceleration angle, na úplném kraji cívky osa Traverser chvíli čeká dle nastaveného Waiting angle. Z tohoto hlediska nejsou průběhy osy Winder podstatné, zde se pouze načítá pozice při velmi pomalu se snižující rychlosti.



Obr. 2-24 Trace 1, 16sTravWind

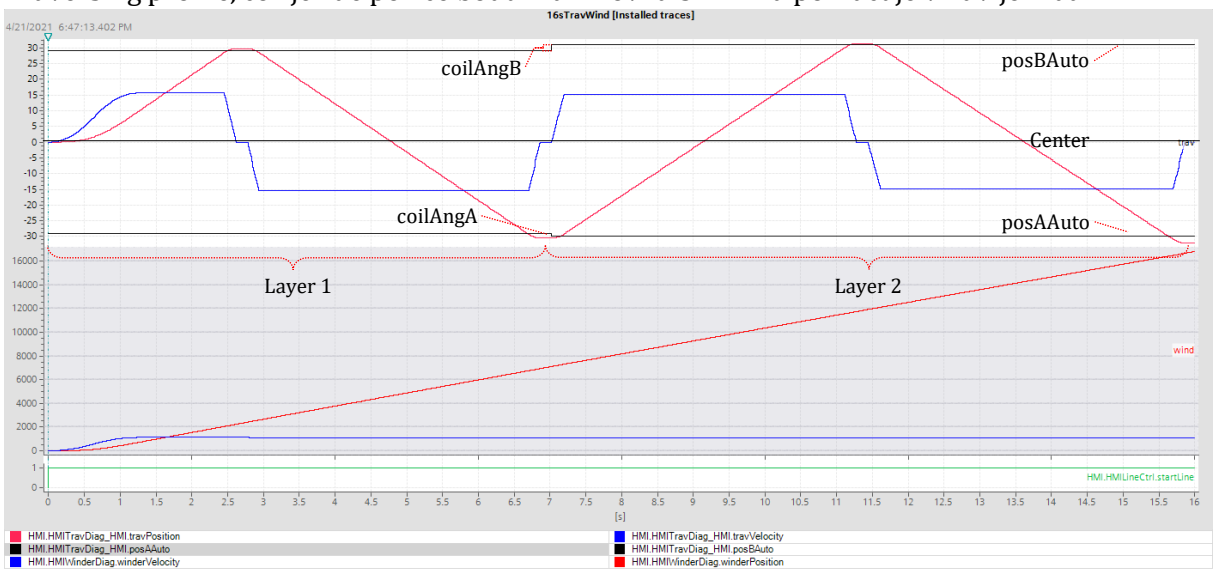
Druhý Trace na Obr. 2-25 kopíruje nastavení předchozího měření, jsou zde ale aktivované funkce Spike a Stroke popsané v podkapitolách 1.2.2.3 a 1.2.2.4. To je možný reálný případ použití, kdy „přetažení“ délky osy Traverser o hodnotu Stroke a následný ztracený čas a rychlost je možné kompenzovat funkcí Spike. Hodnota Stroke byla nastavena na 10 mm, tedy posun každého bodu o 5 mm, což lze vidět na černě zobrazené hodnotě obou bodů A, B, které mají nyní hodnotu ±34 mm. Spike byl nastaven na délku 80° s adaptací 20 %. Vzhledem k tomu, že základní rychlost (před Spike) je zde zaznamenaná 16 mm/s a nejvyšší hodnota (při Spike) přibližně 19,3 mm/s, kompenzace 20 % proběhla správně. Maximální kompenzační rychlost Spike byla ponechána výchozí, tedy 5 mm/ot, která ovšem nebyla využita a přírůstek rychlosti dosáhl max. 3,3 mm/s.



Obr. 2-25 Trace 2, 16sTravWind_Spike_Stroke

Třetí Trace na Obr. 2-26 se reálnému využití poněkud vzdaluje. Nastavení opět kopíruje výchozí stav Trace 1, tedy bez funkcí Spike a Stroke, ale je zde ukázáno, jak funkce změní Traversing profile při použití kónických cívek popsaná v podkapitole 1.2.2.5. Aby ale funkce byla na Trace dobře vidět, byl úhel rozšiřování cívky nastaven na 80°, což v reálném provozu není moc

častý tvar navíjených cívek. Cílem je zde jasně ukázat jak funkce změní rozsah bodů A a B, ovšem protože aplikace mění tento rozsah každou jednu vrstvu podle výšky navíjeného materiálu a tím je tenký drátek vysoký pouze 0,82 mm, bylo by při zadání malého úhlu rozšiřování navíjené cívky z grafu složité vypozorovat změnu. Zde je tedy pozorovatelná na průběhu černě vyznačených nastavených pozic bodů A a B, kde na začátku jsou hranice výchozí stejně jako u Trace 1, 29 mm. Po navití první vrstvy se cívka v další vrstvě rozšíří, aplikace dle nastaveného úhlu vypočítá nový Traversing profile, což je zde pozice bodů A a B rovna 31 mm a pokračuje v navíjení dál.

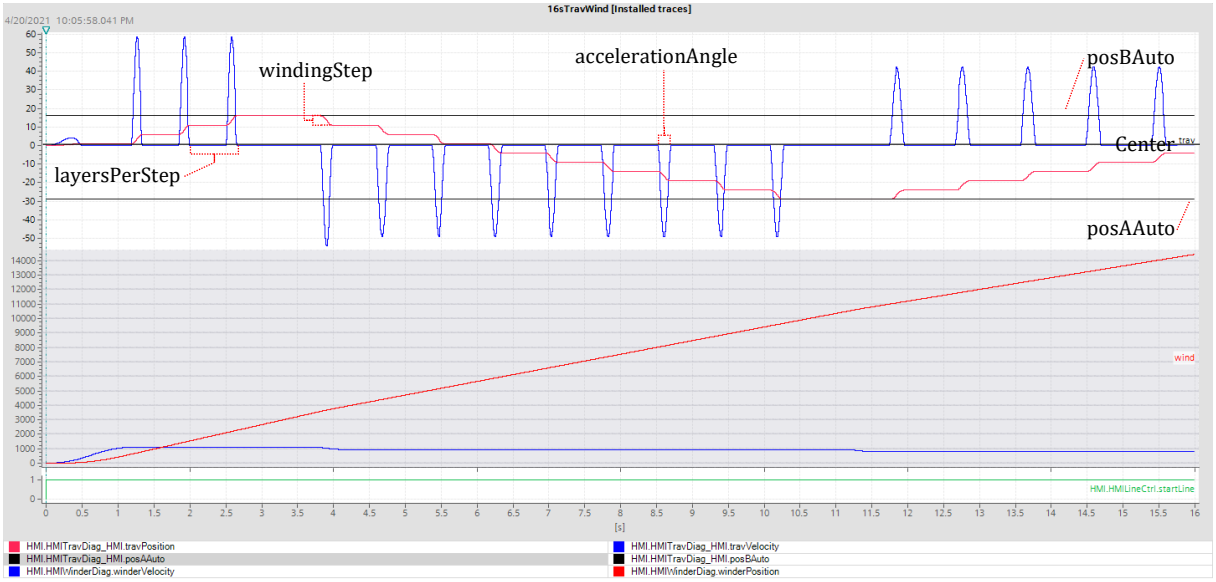


Obr. 2-26 Trace 3, 16sTravWind_Conical

Poslední Trace na Obr. 2-27 je výrazně odlišný od ostatních. Na rozdíl od nich je zde použit krokový navíjecí mód, popsaný v podkapitole 1.2.2.2. Nastaveno bylo 10 kroků vinutí se dvěma vrstvami na každý krok. To je opět poměrně nelogické množství vzhledem k reálnému provozu, zde pro ukázku je však množství ideální, protože se stihne vykonat jeden celý Traversing cycle. Osa Traverser se na začátku nastaví do pozice vypočítané tak, aby se na zadanou délku cívky 58 mm s nastaveným Winding step 5 mm/ot vešlo 10 kroků. Tento výpočet se provádí od bodu A směrem k bodu B. Bod A má tedy hodnotu -29 mm jako v předchozích grafech a následuje 10 kroků po 5 mm, kde končí poslední krok. Finální hodnota pozice bodu B je tedy 16 mm, což odpovídá výpočtu (2-1). Počet kroků je nutné snížit o jedna, protože první krok je nultý (na nulové pozici vzhledem k bodu A).

$$usedCoilLenght = |posAAuto - posBAuto| = windingStep \cdot (numberOfSteps - 1) \qquad (2\text{-}1)$$

Mezi těmito kroky pak osa Traverser přeskakuje a podle rychlosti Winder zde zůstává pouze po dobu navití dvou vrstev. Z grafu lze pomocí kurzorů přibližně určit, že Traverser zůstává na jedné vrstvě přibližně 5,1 s, přičemž osa Winder se otáčí rychlostí přibližně 1200 °/s, což značí že za tu dobu se osa Winder otočila přibližně o 612°, tedy skoro dvě celé otáčky.



Obr. 2-27 Trace 4, 16sTravWind_Step

Všechna tato měření byla také uložena jako měření Measurements se jmény odpovídající titulkům obrázků a jsou součástí programu.

2.3.2 Reálný HW

Program byl kvůli distančním omezením testován především v simulacích. Před použitím na reálném HW ho tedy bylo potřeba upravit, aby odpovídal realitě. Veškeré úpravy zde prováděné jsou ovšem dané specifickým zapojení HW a mohou se u různých aplikací lišit. Samotný program zůstává stejný, protože je naprogramován univerzálně, není potřeba v něm nic měnit.

Po nahrání programu PLC hlásilo chybu připojených periferií. Zde je nutné dodržet nejen správné IP adresy PLC, HMI a FM, ale také jejich PN jména. Protože byl program zkoušen na školním demo kitu, nebyla změněna jména měničů na ta nastavená v programu, ale naopak jména v programu byla přizpůsobena již nastaveným jménům FM, tedy "levy" a "pravy". Další odchylkou v reálné HW konfiguraci bylo použití staršího FW 1.0 v DI/DO modulu, namísto nejnovějšího 1.1, jak je napsáno v Tab. 2-1. Při přepínání do diagnostického pohledu na PC také docházelo k výpadku PLC kvůli chybě nedostatečné paměťové propustnosti. Ve výchozím nastavení je totiž 50 % paměťové šířky rezervováno právě pro NRT komunikace PG/PC Connection a HMI Connection. Upravit to lze v HW konfiguraci PLC, v okně 4, kartě Properties, General v menu PROFINET interface [X1]/Communication load snížením hodnoty např na 30 %.

Při prvním rozběhu je možné, že bude potřeba invertovat osu Winder, např. kvůli tomu že motor bude uchycen k ose z druhé strany, než bylo očekáváno. Pak je nutné v příslušném TO tuto osu invertovat. Je ovšem velmi důležité v tomto případě také invertovat směr enkodéru. Protože enkodér lze umístit i na stranu zátěže, je možné aby měly tyto komponenty opačný směr. V žádném případě to ale nesmí nastat, pokud je, jako zde, enkodér přímo na motoru. Pokud je invertována pouze jedna komponenta, pak enkodér čte rozbíhaní motoru inverzně, tedy zrychlování nesprávným směrem se snaží regulovat zrychlováním, aby byla snížena regulační odchylka. Tím se ale pouze zvýší rychlost inverzním směrem a motor se tak roztáčí neustále na vyšší otáčky. Pro reálný běh byla také v TO omezena rychlost osy Traverser na 10 mm/s kvůli mechanickým limitacím této osy.

Při reálném provozu je nutné počítat s jistou nepřesností pohybu motorů a vyčítání enkodérů. Je také možné, že zátěž může s osami drobně otáčet. Při spuštění aplikace a referencování osy Traverser se tak snadno může stát, že aplikace hlásí chybu 0401_8008, tedy osa Traverser je ve své aktuální pozici mimo svůj Traversing profile a nelze tedy spustit navíjení. Aplikace umožňuje jednoduchou úpravu pomocí parametru tolStartPos [mm], kde lze zadat

hodnotu o kterou může být pozice A menší, resp. pozice B vyšší, aby přesto došlo k zahájení navíjení. V simulacích toto není potřeba, ovšem zde je to nutné. Pro účely zkoušení byla nastavena poměrně velká tolerance 5 mm.

Poslední drobné úpravy byly provedeny na blocích externího enkodéru. Po spuštění enkodér hlásil chybu Adaptation. Ve výchozím nastavení TO_ ExternalEncoder je totiž vždy zaškrtnuto nastavení Automatic data exchange for encoder values (online). Toto nastavení automaticky identifikuje a nastaví parametry enkodéru, pokud jsou v něm tato data k dispozici. Protože byl ale použit velmi jednoduchý enkodér, je nutné toto nastavení v Configuration, záložce Hardware interface, menu Data exchange zrušit. Zároveň je nutné nastavit, kolik mm odvinutého drátu odpovídá jedné otáčce enkodéru, tedy 94,2 mm/ot jak bylo změřeno a zapsáno do parametru Leadscrew pitch v záložce Extended parameters, menu Mechanics. Na videu v příloze je vidět, že externí enkodér správně odečítá odvinutý materiál, ale rychlost i zrychlení zůstavují nulové. Jde o důsledek příliš nízkého rozlišení tohoto enkodéru, které zamezuje správnému čtení těchto hodnot. Ty tedy zůstávají nulové.

2.3.2.1 Kontaktně

Při uvádění do provozu přímo u reálného HW byl dodržován postup popisovaný v předchozích podkapitolách. Nejdříve byly otestovány samotné osy Traverser a Winder pomocí menu Commissioning přímo ve složkách FM. Zde byla otestována základní funkčnost os a rozhodnuto o přibližných maximálních bezpečných hodnotách rychlosti navíjení (360 °/s pro Winder, 10 mm/s pro Traverser). Ve stejném menu, ovšem v rámci TO již byla testována i správnost směru a funkčnost indikací HMI. Zde došlo k invertování osy Winder. Nakonec byly osy testovány ještě jednou už přímo v navrhnutém rozhraní v rámci obrazovek 2a_Manual Traverser a 2b_Manual Winder. Zde již byla testována především přímo funkčnost rozhraní a zjišťováno, zda může být toto ovládání náhradou za předchozí dva způsoby testování.

Protože program fungoval dle předpokladů a simulací, je aplikace uživatelský přívětivá i z toho pohledu, že k testování os není potřeba PC s TIA portálem, ale lze vše otestovat přímo na těchto obrazovkách. Funkčnost aplikace demonstrují Obr. 2-28, Obr. B-9 a video které je součástí Přílohy C. Na videu je nejdříve předvedena změna parametru waitingAngle, jehož popis je v podkapitole 1.2.2, na hodnotu 500 °. Hned po její změně začne HMI indikovat nekonzistenci dat zčervenáním indikátoru DATA, jak je popsáno v podkapitole 2.2.3.2. Dále je aktivován a referencován externí enkodér popisovaný v podkapitole 2.2.2.5. Následuje přechod na obrazovku 2a_Manual Traverser, kde je jeho pozice upravena přibližně na bod A, tedy pozici posA pomocí pohybu JOG, jak je popsáno v podkapitole 2.2.3.3. Po přechodu na obrazovku 3_AUTO a nastavení rychlosti osy Line na 8 m/min je spuštěno navíjení. V počátečním průchodu nastavení si lze všimnout, že byl nastaven parametr windingStep na 2 mm. Protože šířka navíjeného materiálu je 0,82 mm, jsou mezi jednotlivými závity mezery 1,18 mm. Při navíjení tohoto konkrétního materiálu (tenký drátek) nejsou většinou mezery mezi závity na cívce žádoucí, zde jde ovšem o demonstraci funkčnosti tohoto parametru. Navinutá cívka s tímto nastavením je na Obr. B-9 v Příloze B, část navinuté cívky s nastaveným windingStep rovným tloušťce materiálu je níže na Obr. 2-28.

Video je vytvořeno kombinací nahrávaného záznamu z kamery, která snímá technologii a záznam obrazovky PC, které bylo k síti připojeno. Zde je důležité, že použitím Basic panelu není možné jednoduše zrcadlit obrazovku HMI na okno simulace Windows. Spuštěním simulace se vytvoří rozhraní, které není připojené k PLC a na jeho pokyny nereaguje, resp. neukazuje vyčítané veličiny. Zde je nutné změnit PG/PC interface počítače. Nastavení je v ovládacích panelech systému Windows, ikon Set PC/PG interface a zde je nutné mít nastavenou správnou Ethernet síť. Pak je vytvořením simulace HMI okno propojeno komunikací NRT s PLC a lze touto simulací ovládat aplikaci a zároveň nahrávat toto okno jako zde pro demonstrační účely. Obrazovka se ale nikdy nebude zrcadlit na skutečný HW HMI panel, to je výsada právě pouze pokročilejších Comfort panelů.

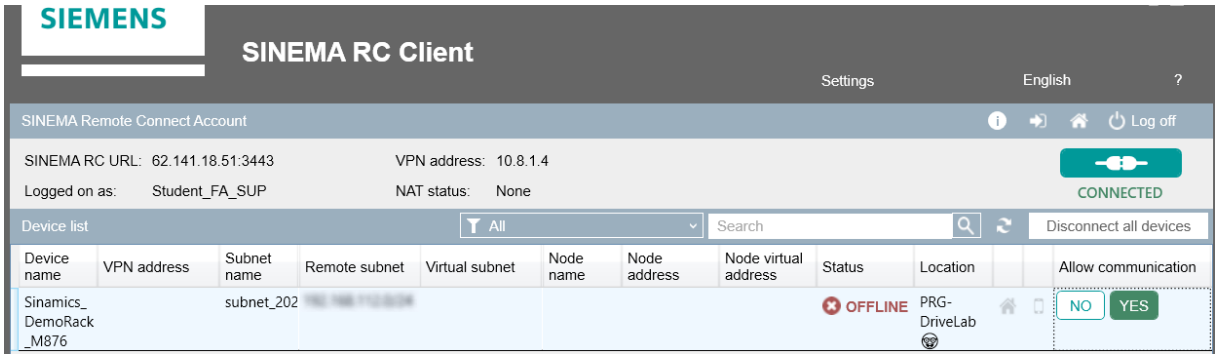
Obr. 2-28 Navíječka na konci navíjení (windingStep 0,82 mm)

2.3.2.2 Vzdálený přístup

Aplikace byla krátce laděna i testována vzdáleně pomocí aplikace Siemens SINEMA RC Client. Na pracovišti byl k dispozici ADSL router SCALANCE M816-1 a webkamera. Router byl připojen PN kabelem k volnému portu P2 FM Traverser místo PG/PC. Zařízení obsahuje čtyřportový switch, s připojeným PLC a kamerou, lze tedy připojit další dvě zařízení.

Obr. 2-29 Router SCALANCE M816-1 [23]

Prostředí aplikace SINEMA RC Client je přehledné a jednoduché. Po zadání správných přístupových údajů k serveru se objeví okno zobrazené na Obr. 2-30. Zde je nutné připojit se vzdáleně k routeru pomocí tlačítka Connect v pravém horním rohu. Protože je v aplikaci možné být v rámci jednoho serveru připojen k více routerům, je nutné dále specifikovat s jakým z nich je požadováno navázat spojení. Zde je pouze jedna možnost, tedy stojan Sinamics_DemoRack_M876 a u něj je tedy nutné povolit komunikaci.



Obr. 2-30 Prostředí SINEMA RC Client

Nahrání programu do PLC pak probíhá stejně jako při připojení přes PN kabel na místě, jak bylo popsáno v podkapitole 2.3.2. Jediný rozdíl je, že při nahrávání nesmí TIA portál vyhledávat spojení na Ethernet síti, ale je nutné toto nastavení změnit na WiFi síť zařízení. Pak se lze k PLC připojit, sledovat v režimu Online měněné veličiny apod. K webkameře se dá po připojení vzdáleného přístupu připojit jednoduše zadáním její IP adresy do internetového vyhledávače a následně zadat jméno a heslo.

Ačkoliv je při prvním uvádění do provozu vždy doporučováno být přímo u stroje, vzdálený přístup představuje vynikající alternativu k následnému ladění. Pokud není možné se k nastavovaném stroji jednoduše dostat, nebo jsou uvedena v platnost distanční omezení, která toto podstatně komplikují, je vzdálený přístup skvělým způsobem nejen pro ladění aplikace, ale také pro případnou první diagnostiku v případě nestandardního, nebo neočekávaného chování aplikace.

ZÁVĚR

Navíječkové aplikace mohou být mimo jednoduchých převíjecích strojů i poměrně komplikovaná soustrojí skládající se z několika poháněných os. [1] V Kapitola 1: této práce byly tyto aplikace teoreticky popsány, s důrazem na aplikace s nepřímým řízením tahu. Popsány byly také specifické požadavky na pohony používané v těchto aplikacích vycházející z navíječkové charakteristiky zátěžného momentu (1.1) [2], nebo regulace hnacího momentu při nepřímém řízení tahu (1.2.1). [1]

Práce v teoretické části dále popisovala aplikaci, která byla uváděna do provozu v části praktické. Byly zde popsány její specifikace, výhody [3] i nevýhody zjištěné při uvádění do provozu (1.2). Byl zde také představen princip Centrálního řízení pohonů přes PLC a jeho výhody a nevýhody oproti decentrálnímu řízení (1.2.1.1). Popisuje specifika izochronní IRT komunikace mezi PLC a FM včetně vysvětlení proč je tato komunikace v těchto případech výhodná (1.2.1.2). Následuje obsáhlý popis programu z praktické části (1.2.1.3) včetně všech implementovaných funkcí (1.2.2) [3], zahrnutí externího enkodéru a popisu principu připojení a uvádění do provozu pomocí vzdáleného přístupu (1.3). [28]

Kapitola 2:, tedy praktická část obsahuje podrobný popis uvádění do provozu této aplikace. Je zde nejdříve představena HW konfigurace (Obr. 2-1) včetně samotné navíječky (Obr. 2-2). Jednotlivé HW komponenty jsou dále detailně popsány (2.1). V práci následuje popis prostředí TIA portálu (2.2) a konfigurace HW pomocí tohoto SW (2.2.1).

Práce velmi detailně popisuje způsoby programování v TIA portálu, konfiguraci TO a infrastrukturu proměnných použitou v aplikaci. Dále popisuje nastavení PB nutných k běhu aplikace, a to pro inicializaci, hlavní programovou smyčku, infrastrukturu pro výměnu dat s HMI i přidaný externí enkodér (2.2.2). Dále je opět detailně popsán způsob vytváření rozhraní pro HMI, kde bylo využito všech typů obrazovek poskytovaných TIA portálem (2.2.3). Kapitoly jsou doplněné obrázky, kompletní navrhnuté rozhraní je součástí Přílohy B.

Na konci práce je popsán postup a různé způsoby uvádění do provozu aplikace. Jsou zde popsány možnosti TIA portálu z hlediska simulací, včetně nasimulovaných průběhů (2.3.1). Jsou zde také poznatky z uvádění do provozu na reálném stroji (2.3.2) a to jak při spouštění přímo u stroje (2.3.2.1), tak při vzdáleném přístupu (2.3.2.2).

Práce si dávala za cíl především zprovoznit standardní Siemens aplikaci [3] s využitím co nejvíce možností TIA portálu. Aplikace byla prakticky celá naprogramována a vyzkoušena v simulátorech. Přesto bylo při prvním spouštění na reálném HW potřeba upravit pouze několik málo nastavení souvisejících především s mechanikou pohonu, samotná aplikace pak nemusela být upravena vůbec (2.3.2). Tento způsob programování tak potvrdil, že lze i takto poměrně složitou aplikaci nastavit do takřka funkčního stavu, aniž by byla nahrána do reálného HW.

Problémy ovšem nastaly se samotnou standardní aplikací. Ačkoliv má zákazníkovi poskytnout základní program který pak bude jen drobně upraven a připraven k použití, nemusí tomu tak vždy být. Největším problémem je nejasná a někdy zbytečně komplikovaná struktura výchozího programu, např. bloky inicializace (2.2.2.3), nebo nefunkční možnost nastavení navíjecího módu (2.2.2.4). Velkým problémem je také přizpůsobení aplikace pokročilejšímu Comfort panelu ve standardní aplikaci, což při použití jednoduššího Basic panelu (2.1.4) zamezí kompilaci kvůli nekompatibilním datový typům (2.2.2.6), nepodporovaným funkcím a prakticky tak vždy znamená nutnost kompletního přepracování rozhraní (2.2.3). Aplikace svou uzavřenou datovou infrastrukturou také není příliš uživatelsky přívětivá k přidání dalších externích PB, jako např, externího enkodéru (2.2.2.5).

Všechny tyto problémy a překážky při programování jsou v práci podrobně popsány, je k nim doplněno i jejich řešení a návrh na úpravu standardní aplikace. Program byl úspěšně otestován na reálném Siemens demo kitu, což demonstrují Obr. 2-28, Obr. B-9, a především video které je společně s programem nahráno na přiloženém CD v Příloze C.

LITERATURA

[1] WEIDAUER, Jens a Richard MESSER. *Electrical Drives: Principles, Planning, Applications, Solutions*. Erlangen: Publicis Publishing, 2014. ISBN 978-3-89578-434-7.

[2] KOBRLE, Pavel a Jiří PAVELKA. *Elektrické pohony a jejich řízení*. 3. přepracované vydání. V Praze: České vysoké učení technické, 2016. ISBN 978-80-01-06007-0.

[3] *SIMATIC Traversing Drive* [online]. SIEMENS AG, 2019 [cit. 2021-02-06]. Dostupné z: https://support.industry.siemens.com/cs/document/109758582/simatic-traversing-drive?dti=0&lc=en-WW

[4] *Configuration of technology objects with SIMATIC S7-1500 and SINAMICS S210 in the TIA Portal* [online]. SIEMENS AG, 2019 [cit. 2021-02-06]. Dostupné z: https://support.industry.siemens.com/cs/document/109749795/configuring-technology-objects-with-simatic-s7-1500-and-sinamics-s210-in-tia-portal?dti=0&pnid=13204&lc=en-WW

[5] PLACHÝ, Ladislav. *SIMATIC HMI: Nástroje pro všechny vizualizační úlohy* [online]. SIEMENS AG, 2018 [cit. 2021-03-16]. Dostupné z: https://moodle.fel.cvut.cz/mod/resource/view.php?id=174275

[6] NOVOTNÝ, Radek. *SIMATIC S7-1500 T-CPU: Technologické objekty* [online]. Praha: SIEMENS AG, 2019 [cit. 2021-02-13]. Dostupné z: https://www.tianadosah.cz/upload/01-technologicke-objekty-v-simatic-s7-1500-t-cpu.pdf

[7] *Learn-/Training Document: High-Level Language Programming with SCL and SIMATIC S7-1200* [online]. Nuremberg: SIEMENS AG, 2018 [cit. 2021-04-02]. Dostupné z: https://www.automation.siemens.com/sce-static/learning-training-documents/tia-portal/advanced-programming-s7-1200/sce-051-201-scl-s7-1200-r1709-en.pdf

[8] *SIMATIC Winder and Tension Control* [online]. SIEMENS AG, 03/2021 [cit. 2021-03-24]. Dostupné z: https://support.industry.siemens.com/cs/document/58565043/simatic-winder-and-tension-control?dti=0&lc=en-WW

[9] *"LAxisBasics" Easy Control of the "Axis" TO in the SIMATIC S7-1500* [online]. SIEMENS AG, 05/2020 [cit. 2021-03-14]. Dostupné z: https://support.industry.siemens.com/cs/document/109749348/simatic-s7-1500-s7-1500t%3A-standard-application-axis-control?dti=0&lc=en-DE

[10] *SCE Training Curriculum: Automation System SIMATIC S7-1500* [online]. Nuremberg, Německo: SIEMENS AG Digital Factory Division, 04/2016 [cit. 2021-02-12]. Dostupné z: https://www.automation.siemens.com/sce-static/learning-training-documents/tia-portal/summary-sce-training-curriculum-s7-1500-en.pdf

[11] *Products for Totally Integrated Automation: Catalog ST 70* [online]. SIEMENS AG, 2019 [cit. 2021-02-06]. Dostupné z: https://support.industry.siemens.com/cs/document/109744167/catalog-st-70%3A-products-for-totally-integrated-automation-simatic?dti=0&pnid=13204&lc=en-WW

[12] *SINAMICS S210 Servo Drive Systém: Catalog D 32* [online]. SIEMENS AG, 2020 [cit. 2021-02-06]. Dostupné z: https://support.industry.siemens.com/cs/document/109754381/catalog-d-32%3A-sinamics-s210-servo-drive-system?dti=0&pnid=13204&lc=en-WW

[13] *SINAMICS S210 / SIMOTICS S-1FK2: Operating Instructions* [online]. SIEMENS AG, 2020 [cit. 2021-02-06]. Dostupné z: https://support.industry.siemens.com/cs/document/109781874/sinamics-s210-simotics-s-1fk2?dti=0&pnid=13204&lc=en-WW

[14] *SIMOTION/SIMATIC/SINAMICS - Converting Toolbox* [online]. SIEMENS AG, 2020 [cit. 2021-02-06]. Dostupné z: https://support.industry.siemens.com/cs/document/109744606/simotion-simatic-sinamics-converting-toolbox?dti=0&pnid=13204&lc=en-WW

[15] WESTLAKE, Robert. *PROFINET – RT vs IRT* [online]. SIEMENS AG, 2020 [cit. 2021-02-13]. Dostupné z: https://assets.new.siemens.com/siemens/assets/api/uuid:30e0ba94-b1af-4488-98dd-8fe716382cc0/siemens-profinet-rt-vs-irt-webinar-13oct2020.pdf

[16] URBAN, Luboš. Programování PLC podle normy IEC EN 61131-3: víc než jednotné jazyky. *Automa – časopis pro automatizační techniku* [online]. 2005, **2005**(2) [cit. 2021-04-07]. Dostupné z: https://automa.cz/cz/casopis-clanky/programovani-plc-podle-normy-iec-en-61131-3-vic-nez-jednotne-jazyky-2005_02_30310_1237/

[17] *Industry Mall: 6ES7511-1TK01-0AB0* [online]. SIEMENS AG, 2020 [cit. 2021-02-06]. Dostupné z: https://mall.industry.siemens.com/mall/cs/cz/Catalog/Product/6ES7511-1TK01-0AB0

[18] *Industry Mall: 6ES7550-1AA00-0AB0* [online]. SIEMENS AG, 2020 [cit. 2021-03-21]. Dostupné z: https://mall.industry.siemens.com/mall/cs/cz/Catalog/Product/6ES7550-1AA00-0AB0

[19] *Industry Mall: 6ES7523-1BL00-0AA0* [online]. SIEMENS AG, 2020 [cit. 2021-03-21]. Dostupné z: https://mall.industry.siemens.com/mall/cs/cz/Catalog/Product/6ES7523-1BL00-0AA0

[20] *Industry Mall: 6SL3210-5HB10-1UF0* [online]. SIEMENS AG, 2020 [cit. 2021-02-06]. Dostupné z: https://mall.industry.siemens.com/mall/cs/cz/Catalog/Product/6SL3210-5HB10-1UF0

[21] *Industry Mall: 1FK2102-0AG00-1MA0* [online]. SIEMENS AG, 2020 [cit. 2021-02-06]. Dostupné z: https://mall.industry.siemens.com/mall/cs/cz/Catalog/Product/1FK2102-0AG00-1MA0

[22] *Industry Mall: 6AV2123-2DB03-0AX0* [online]. SIEMENS AG, 2020 [cit. 2021-02-06]. Dostupné z: https://mall.industry.siemens.com/mall/cs/cz/Catalog/Product/6AV2123-2DB03-0AX0

[23] *Industry Mall: 6GK5816-1BA00-2AA2* [online]. SIEMENS AG, 2020 [cit. 2021-5-5]. Dostupné z: https://mall.industry.siemens.com/mall/cs/cz/Catalog/Product/6GK5816-1BA00-2AA2

[24] KALÁT, Ondřej. *Konfigurace řídicí části pohonu s asynchronním motorem* [online]. [cit. 2021-02-06]. Dostupné z: https://dspace.cvut.cz/handle/10467/82803

[25] *Totally Integrated Automation Portal. V15.1*. Software. SIEMENS AG, Nápověda v programu.

[26] *IEC 61131-3 and SIMATIC S7* [online]. SIEMENS AG, 06/2003n. l. [cit. 2021-04-07]. Dostupné z: https://support.industry.siemens.com/cs/document/8790932/iec-61131-3-and-simatic-s7?dti=0&lc=en-WW

[27] *Digital Twins: Simulation at Siemens* [online]. SIEMENS AG, 2021 [cit. 2021-04-19]. Dostupné z: https://new.siemens.com/global/en/company/stories/research-technologies/digitaltwin/digital-twin.html

[28] *SINEMA Remote Connect: Platforma pro správu vzdálených sítí* [online]. SIEMENS AG, 2021 [cit. 2021-04-19]. Dostupné z: https://new.siemens.com/cz/cs/products/automation/industrial-communication/industrial-remote-communication/remote-networks/sinema-remote-connect-access-service.html

OBSAH PŘÍLOH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

PŘÍLOHA A: SEZNAM SYMBOLŮ A ZKRATEK

A.1 Seznam symbolů

M_z	[Nm]	Zátěžný moment
Ω	[rad^{-1}]	Úhlová rychlost
P_z	[W]	Úhlová rychlost
F_{pull}	[N]	Požadovaná tažná síla
$M_{setpoint}$	[Nm]	Požadovaný tažný moment
$v_{setpoint}$	[mm · s^{-1}]	Požadovaná úhlová rychlost
D	[mm]	Aktuální průměr cívky
K_p	[-]	Proporční konstanta regulátoru

A.2 Seznam zkratek

PLC	Programmable Logic Controller (programovatelný logický automat)
FM	Frekvenční měnič
PM	Power module (výkonový modul)
PC/PG	Personal Computer/Programmiergeräten (osobní/programovací počítač)
PID	Proportional–integral–derivative (typ regulátoru)
GSDML	General Station Description Markup Language
MLFB	Maschinen lesbare Fabrikate bezeichnung (Strojově čitelné označení produktu)
CPU	Central Processing Unit (procesor)
OCC	One Cable Connection (připojení pomocí jednoho kabelu)
PMSM	Permanent-Magnet Synchronous Motor (synchronní motor s perm. magnety)
HMI	Human machine interface (rozhraní člověk-stroj)
KTP	Key Touch Panel (dotykový panel s tlačítky)
TFT	Thin-Film Transistors (varianta LCD displeje)
LCD	Liquid Crystal Display (displej z kapalných krystalů)
USB	Universal Serial Bus (univerzální sériová sběrnice)
PN	PROFINET
TIA	Totally Integrated Automation
DI/DO	Digital input/digital output (digitální vstup/digitální výstup)
IO	Input Output (vstup, výstup)
HW	Hardware
SW	Software
FW	Firmware
RT	Real Time
IRT	Isochronous Real Time
NRT	None Real Time
TO	Technology object (technologické objekty)
MC	Motion Control (řízení pohybu)
PB	Program Blocks (bloky programovacích celků)
OB	Organization block (organizační blok)
FB	Function block (blok funkce)
FC	Function (funkce)
DB	Data block (datový blok)
SCL	Structured Control Language (strukturovaný řídicí jazyk)
LAD	Ladder (žebříkový programovací jazyk)
FBD	Function Block Diagram (diagramy funkčních bloků)
STL	Statement List (základní jazyk pro PLC)
DSC	Dynamic Servo Control (řízení vysoce aplikací dynamických servopohonů)

PŘÍLOHA B: OBRÁZKOVÁ PŘÍLOHA

Tab. B-1 Řídicí a stavové slovo 2 [13]

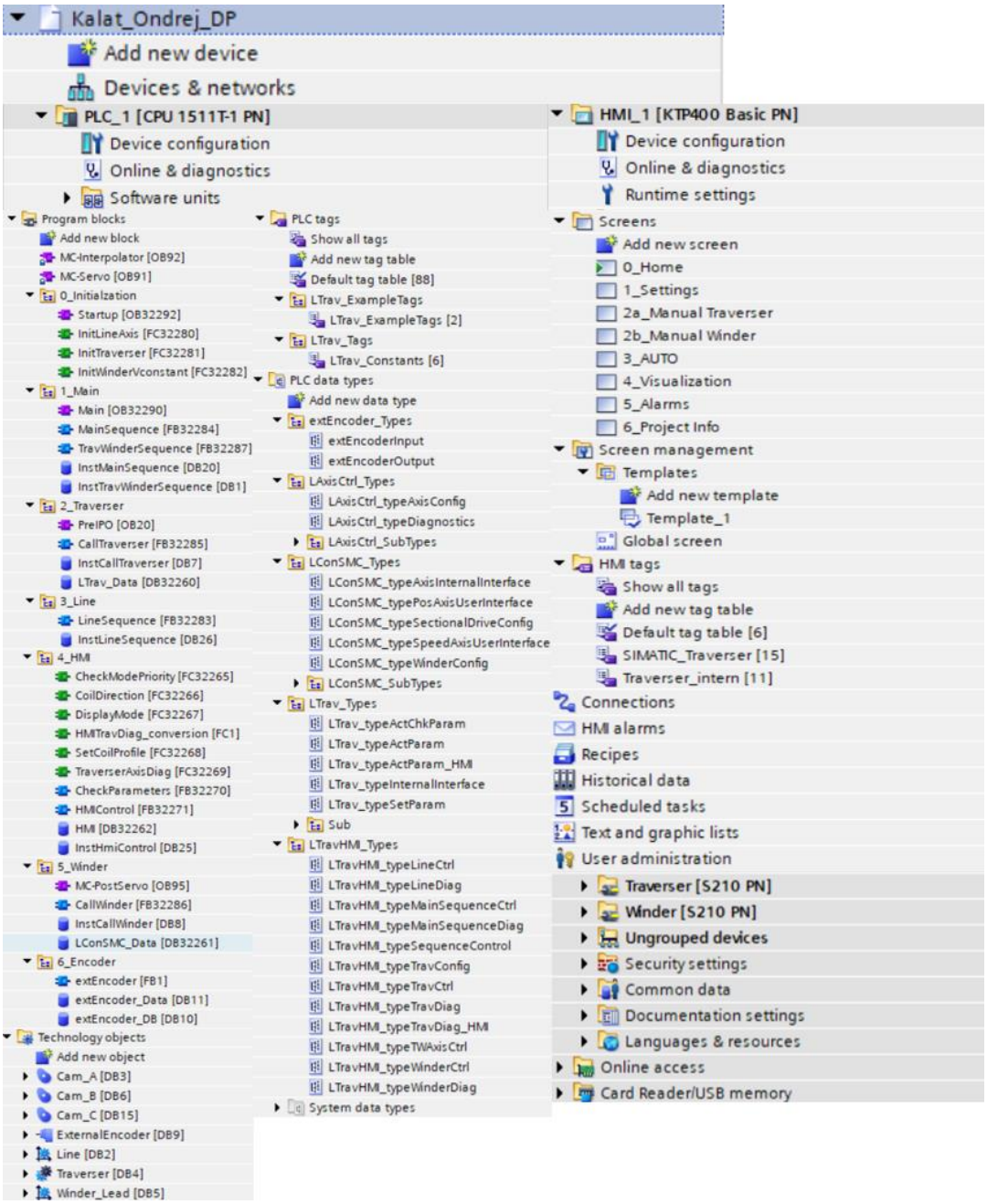
Control word 2 (STW2)		Status word 2 (ZSW2)	
Bit	**Meaning**	**Bit**	**Meaning**
00	Reserved	00	Reserved
01	Reserved	01	Reserved
02	Reserved	02	Reserved
03	Reserved	03	Reserved
04	Reserved	04	Reserved
05	Reserved	05	Open holding brake
06	Integrator disable, speed controller	06	Integrator disable, speed controller
07	Parking axis selection	07	Parking axis active
08	Travel to fixed stop	08	Travel to fixed stop
09	Reserved	09	Reserved
10	Reserved	10	Reserved
11	Reserved	11	Reserved
12	Controller sign-of-life bit 0	12	Device sign-of-life bit 0
13	Controller sign-of-life bit 1	13	Device sign-of-life bit 1
14	Controller sign-of-life bit 2	14	Device sign-of-life bit 2
15	Controller sign-of-life bit 3	15	Device sign-of-life bit 3

Tab. B-2 Řídicí a stavové slovo 1 enkodéru [13]

Control word 1 (G1_STW)		Status word 1 (G1_ZSW)	
Bit	**Meaning**	**Bit**	**Meaning**
00	Request function 1	00	Function 1 active
01	Request function 2	01	Function 2 active
02	Request function 3	02	Function 3 active
03	Request function 4	03	Function 4 active
04	Request command bit 0	04	Value 1
05	Request command bit 1	05	Value 2
06	Request command bit 2	06	Value 3
07	Mode	07	Value 4
08	Reserved	08	Measuring input 1 deflected
09	Reserved	09	Measuring input 2 deflected
10	Reserved	10	Reserved
11	Reserved	11	Acknowledge encoder fault active
12	Reserved	12	Reserved
13	Request absolute value cyclically	13	Cyclic absolute value
14	Request parking encoder	14	Parking encoder active
15	Acknowledge encoder fault	15	Encoder fault

Tab. B-3 Komunikační slovo [13]

Message word 2B (MELDW)	
Bit	**Meaning**
00	Reserved
01	Torque utilization < threshold 2
02	|n_actual| < speed threshold 3
03	|n_actual| < speed threshold 2
04	Reserved
05	Reserved
06	No warning motor overtemperature
07	No warning converter overtemperature
08	n-target/actual deviation within tolerance
09	Reserved
10	Reserved
11	Servo enable
12	Drive ready
13	"Pulses enabled"
14	Reserved
15	Reserved



Obr. B-1 Projektový strom v TIA portálu

SIMATIC_Traverser

Name	Data type	Connection	PLC name	PLC tag	Acquisition cycle
extEncoder.error	Bool	HMI_Connectio...	PLC_1	extEncoder_Data.error	100 ms
extEncoder.input	extEncoderInput	HMI_Connectio...	PLC_1	extEncoder_Data.input	100 ms
extEncoder.output	extEncoderOutput	HMI_Connectio...	PLC_1	extEncoder_Data.output	100 ms
HMI_VAR.acknowledge	Bool	HMI_Connectio...	PLC_1	HMI.acknowledge	100 ms
HMI_VAR.HMILineCtrl	LTravHMI_typeLineCtrl	HMI_Connectio...	PLC_1	HMI.HMILineCtrl	100 ms
HMI_VAR.HMILineDiag	LTravHMI_typeLineDiag	HMI_Connectio...	PLC_1	HMI.HMILineDiag	100 ms
HMI_VAR.HMIMainSequenceCtrl	LTravHMI_typeMainSequenceCtrl	HMI_Connectio...	PLC_1	HMI.HMIMainSequenceCtrl	100 ms
HMI_VAR.HMIMainSequenceDiag	LTravHMI_typeMainSequenceDiag	HMI_Connectio...	PLC_1	HMI.HMIMainSequenceDiag	100 ms
HMI_VAR.HMITravConfig	LTravHMI_typeTravConfig	HMI_Connectio...	PLC_1	HMI.HMITravConfig	100 ms
HMI_VAR.HMITravCtrl	LTravHMI_typeTravCtrl	HMI_Connectio...	PLC_1	HMI.HMITravCtrl	100 ms
HMI_VAR.HMITravDiag_HMI	LTravHMI_typeTravDiag_HMI	HMI_Connectio...	PLC_1	HMI.HMITravDiag_HMI	100 ms
HMI_VAR.HMIWinderCtrl	LTravHMI_typeWinderCtrl	HMI_Connectio...	PLC_1	HMI.HMIWinderCtrl	100 ms
HMI_VAR.HMIWinderDiag	LTravHMI_typeWinderDiag	HMI_Connectio...	PLC_1	HMI.HMIWinderDiag	100 ms
HMI_VAR.manualModeSelected	Bool	HMI_Connectio...	PLC_1	HMI.manualModeSelected	100 ms
HMI_VAR.selectedIndex	USInt	HMI_Connectio...	PLC_1	HMI.selectedIndex	100 ms

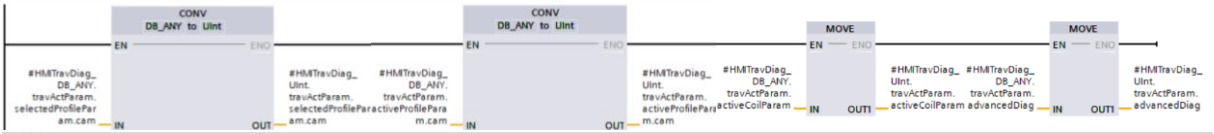
Obr. B-2 Přenášené HMI tagy

Block title:

This FC converts HMITravDiag (data type "LTravHMI_typeTravDiag") parameter to HMITravDiag_HMI (data type "LTravHMI_typeTravDiag_HMI"), where no sub-parameter of DB_ANY data type is used

Network 1:

Conversion and move .cam parameters



Network 2:

Move the rest of travActParam parameter

```
#HMITravDiag_UInt.travActParam.selectedProfileParam.startPosA := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.startPosA;
#HMITravDiag_UInt.travActParam.selectedProfileParam.endPosA := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.endPosA;
#HMITravDiag_UInt.travActParam.selectedProfileParam.posB := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.posB;
#HMITravDiag_UInt.travActParam.selectedProfileParam.windingStep := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.windingStep;
#HMITravDiag_UInt.travActParam.selectedProfileParam.accAng := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.accAng;
#HMITravDiag_UInt.travActParam.selectedProfileParam.waitAng := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.waitAng;
#HMITravDiag_UInt.travActParam.selectedProfileParam.displAng := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.displAng;
#HMITravDiag_UInt.travActParam.selectedProfileParam.stroke := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.stroke;
#HMITravDiag_UInt.travActParam.selectedProfileParam.revPoint := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.revPoint;
#HMITravDiag_UInt.travActParam.selectedProfileParam.centerPoint := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.centerPoint;
#HMITravDiag_UInt.travActParam.selectedProfileParam.lastRevPoint := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.lastRevPoint;
#HMITravDiag_UInt.travActParam.selectedProfileParam.layerGap := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.layerGap;
#HMITravDiag_UInt.travActParam.selectedProfileParam.stepWindPosBMasterPosition :=
#HMITravDiag_DB_ANY.travActParam.selectedProfileParam.stepWindPosBMasterPosition;
#HMITravDiag_UInt.travActParam.selectedProfileParam.addSpeedAcc := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.addSpeedAcc;
#HMITravDiag_UInt.travActParam.selectedProfileParam.accSpike := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.accSpike;
#HMITravDiag_UInt.travActParam.selectedProfileParam.addSpeedDec := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.addSpeedDec;
#HMITravDiag_UInt.travActParam.selectedProfileParam.decSpike := #HMITravDiag_DB_ANY.travActParam.selectedProfileParam.decSpike;

#HMITravDiag_UInt.travActParam.activeProfileParam.startPosA := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.startPosA;
#HMITravDiag_UInt.travActParam.activeProfileParam.endPosA := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.endPosA;
#HMITravDiag_UInt.travActParam.activeProfileParam.posB := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.posB;
#HMITravDiag_UInt.travActParam.activeProfileParam.windingStep := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.windingStep;
#HMITravDiag_UInt.travActParam.activeProfileParam.accAng := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.accAng;
#HMITravDiag_UInt.travActParam.activeProfileParam.waitAng := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.waitAng;
#HMITravDiag_UInt.travActParam.activeProfileParam.displAng := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.displAng;
#HMITravDiag_UInt.travActParam.activeProfileParam.stroke := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.stroke;
#HMITravDiag_UInt.travActParam.activeProfileParam.revPoint := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.revPoint;
#HMITravDiag_UInt.travActParam.activeProfileParam.centerPoint := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.centerPoint;
#HMITravDiag_UInt.travActParam.activeProfileParam.lastRevPoint := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.lastRevPoint;
#HMITravDiag_UInt.travActParam.activeProfileParam.layerGap := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.layerGap;
#HMITravDiag_UInt.travActParam.activeProfileParam.stepWindPosBMasterPosition :=
#HMITravDiag_DB_ANY.travActParam.activeProfileParam.stepWindPosBMasterPosition;
#HMITravDiag_UInt.travActParam.activeProfileParam.addSpeedAcc := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.addSpeedAcc;
#HMITravDiag_UInt.travActParam.activeProfileParam.accSpike := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.accSpike;
#HMITravDiag_UInt.travActParam.activeProfileParam.addSpeedDec := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.addSpeedDec;
#HMITravDiag_UInt.travActParam.activeProfileParam.decSpike := #HMITravDiag_DB_ANY.travActParam.activeProfileParam.decSpike;
```

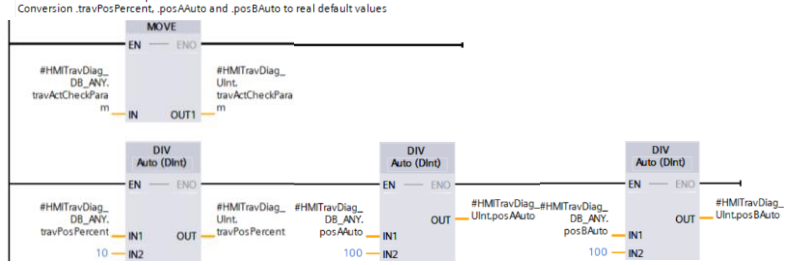
Network 3:

Move base HMITravDiag parameters excluding .travPosPercent, .posAAuto and .posBAuto

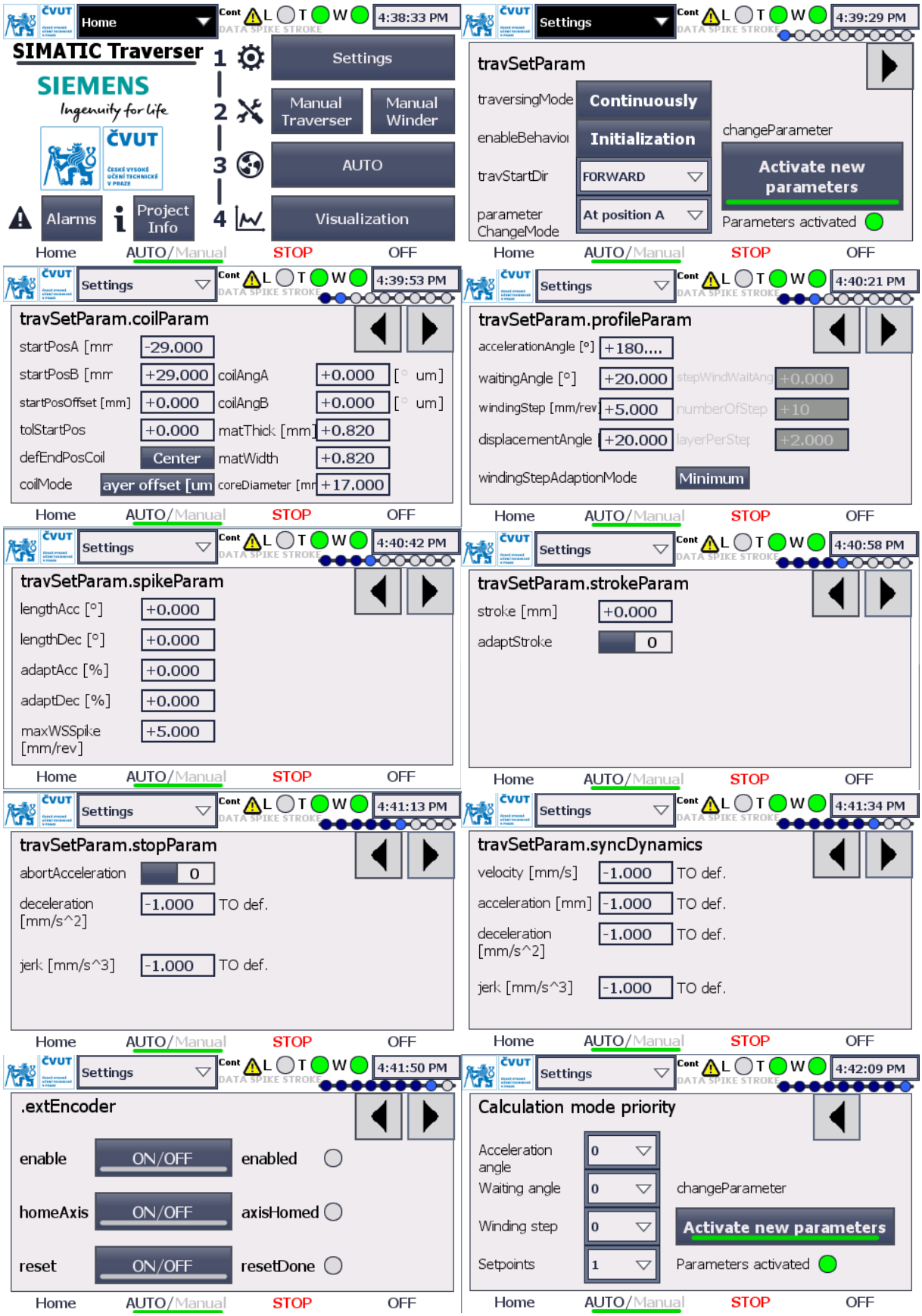
```
#HMITravDiag_UInt.multiCheckDone := #HMITravDiag_DB_ANY.multiCheckDone;
#HMITravDiag_UInt.multiCheckBusy := #HMITravDiag_DB_ANY.multiCheckBusy;
#HMITravDiag_UInt.multiCheckError := #HMITravDiag_DB_ANY.multiCheckError;
#HMITravDiag_UInt.checkInvalid := #HMITravDiag_DB_ANY.checkInvalid;
#HMITravDiag_UInt.checkWarning := #HMITravDiag_DB_ANY.checkWarning;
#HMITravDiag_UInt.outErrorFBTC := #HMITravDiag_DB_ANY.outErrorFBTC;
#HMITravDiag_UInt.outErrorFBTCSub := #HMITravDiag_DB_ANY.outErrorFBTCSub;
#HMITravDiag_UInt.travAxisEnabled := #HMITravDiag_DB_ANY.travAxisEnabled;
#HMITravDiag_UInt.travAxisHomed := #HMITravDiag_DB_ANY.travAxisHomed;
#HMITravDiag_UInt.travAxisError := #HMITravDiag_DB_ANY.travAxisError;
#HMITravDiag_UInt.travCtrlEnabled := #HMITravDiag_DB_ANY.travCtrlEnabled;
#HMITravDiag_UInt.travLayerFWD := #HMITravDiag_DB_ANY.travLayerFWD;
#HMITravDiag_UInt.setParamEqual := #HMITravDiag_DB_ANY.setParamEqual;
#HMITravDiag_UInt.multiCheckErrorID := #HMITravDiag_DB_ANY.multiCheckErrorID;
#HMITravDiag_UInt.outErrorIdFBTC := #HMITravDiag_DB_ANY.outErrorIdFBTC;
#HMITravDiag_UInt.outErrorIDFBTCToDINT := #HMITravDiag_DB_ANY.outErrorIDFBTCToDINT;
#HMITravDiag_UInt.outErrorIdFBSub := #HMITravDiag_DB_ANY.outErrorIdFBSub;
#HMITravDiag_UInt.displayProfileA := #HMITravDiag_DB_ANY.displayProfileA;
#HMITravDiag_UInt.displayProfileB := #HMITravDiag_DB_ANY.displayProfileB;
#HMITravDiag_UInt.travPositionDINT := #HMITravDiag_DB_ANY.travPositionDINT;
//#HMITravDiag_UInt.travPosPercent := #HMITravDiag_DB_ANY.travPosPercent;
#HMITravDiag_UInt.stepWindLayersPercent := #HMITravDiag_DB_ANY.stepWindLayersPercent;
//#HMITravDiag_UInt.posAAuto := #HMITravDiag_DB_ANY.posAAuto;
#HMITravDiag_UInt.posACheck := #HMITravDiag_DB_ANY.posACheck;
//#HMITravDiag_UInt.posBAuto := #HMITravDiag_DB_ANY.posBAuto;
#HMITravDiag_UInt.posBCheck := #HMITravDiag_DB_ANY.posBCheck;
#HMITravDiag_UInt.generalDisplayMode := #HMITravDiag_DB_ANY.generalDisplayMode;
#HMITravDiag_UInt.generalDisplayModeCheck := #HMITravDiag_DB_ANY.generalDisplayModeCheck;
#HMITravDiag_UInt.calcModeSettings := #HMITravDiag_DB_ANY.calcModeSettings;
#HMITravDiag_UInt.outActCalcMode := #HMITravDiag_DB_ANY.outActCalcMode;
#HMITravDiag_UInt.TWControlState := #HMITravDiag_DB_ANY.TWControlState;
#HMITravDiag_UInt.travVelocity := #HMITravDiag_DB_ANY.travVelocity;
#HMITravDiag_UInt.travPosition := #HMITravDiag_DB_ANY.travPosition;
```

Network 4:

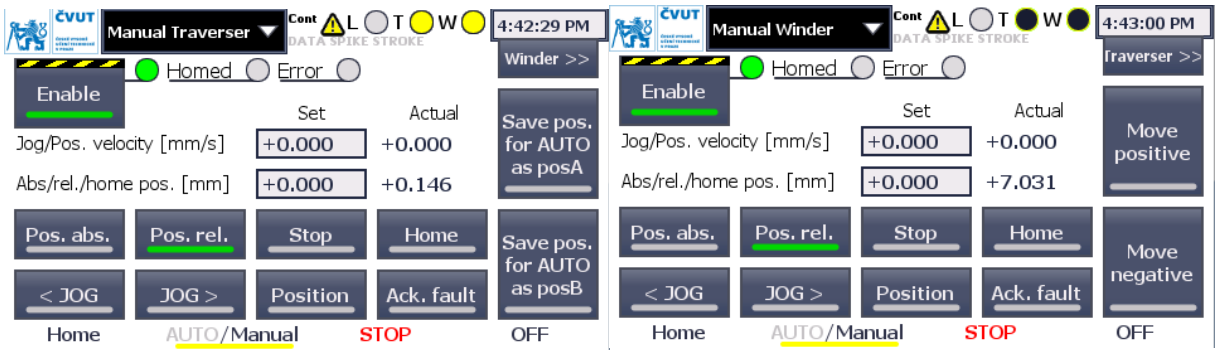
Move travActCheckParam parameter
Conversion .travPosPercent, .posAAuto and .posBAuto to real default values



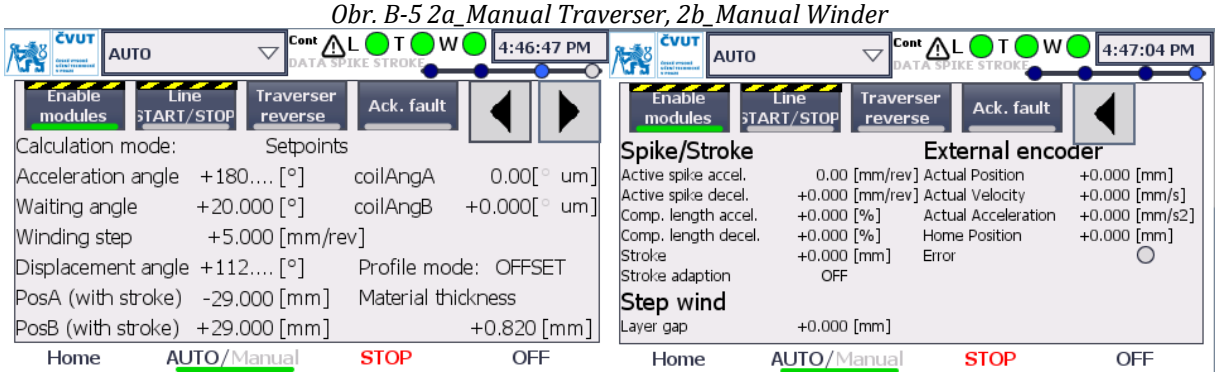
Obr. B-3 FC HMITravDiag_conversion



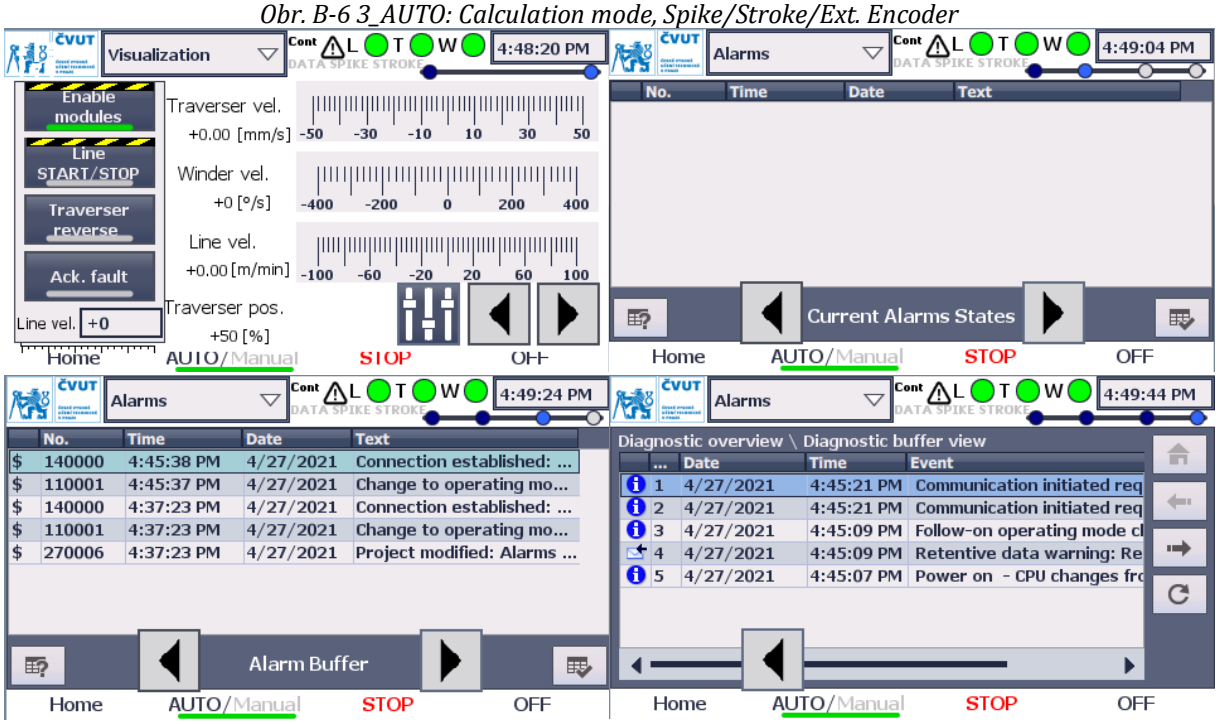
Obr. B-4 0_Home, 1_Settings



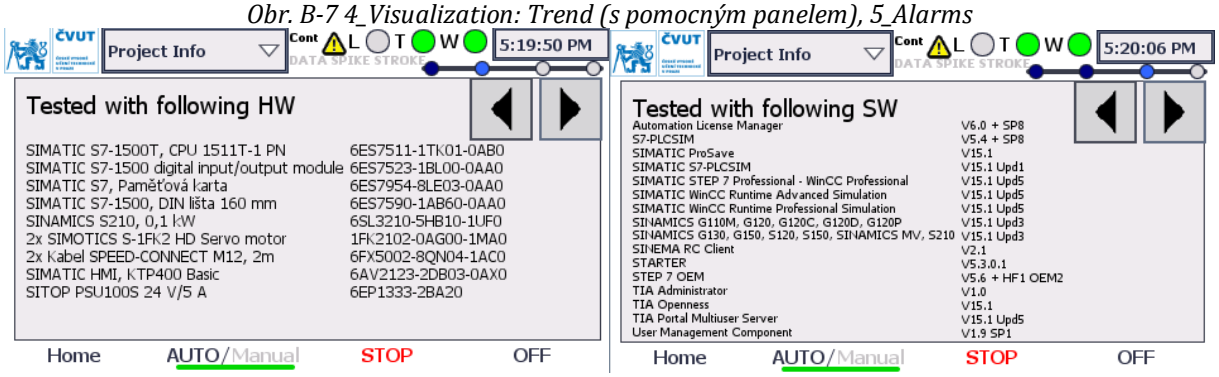
Obr. B-5 2a_Manual Traverser, 2b_Manual Winder



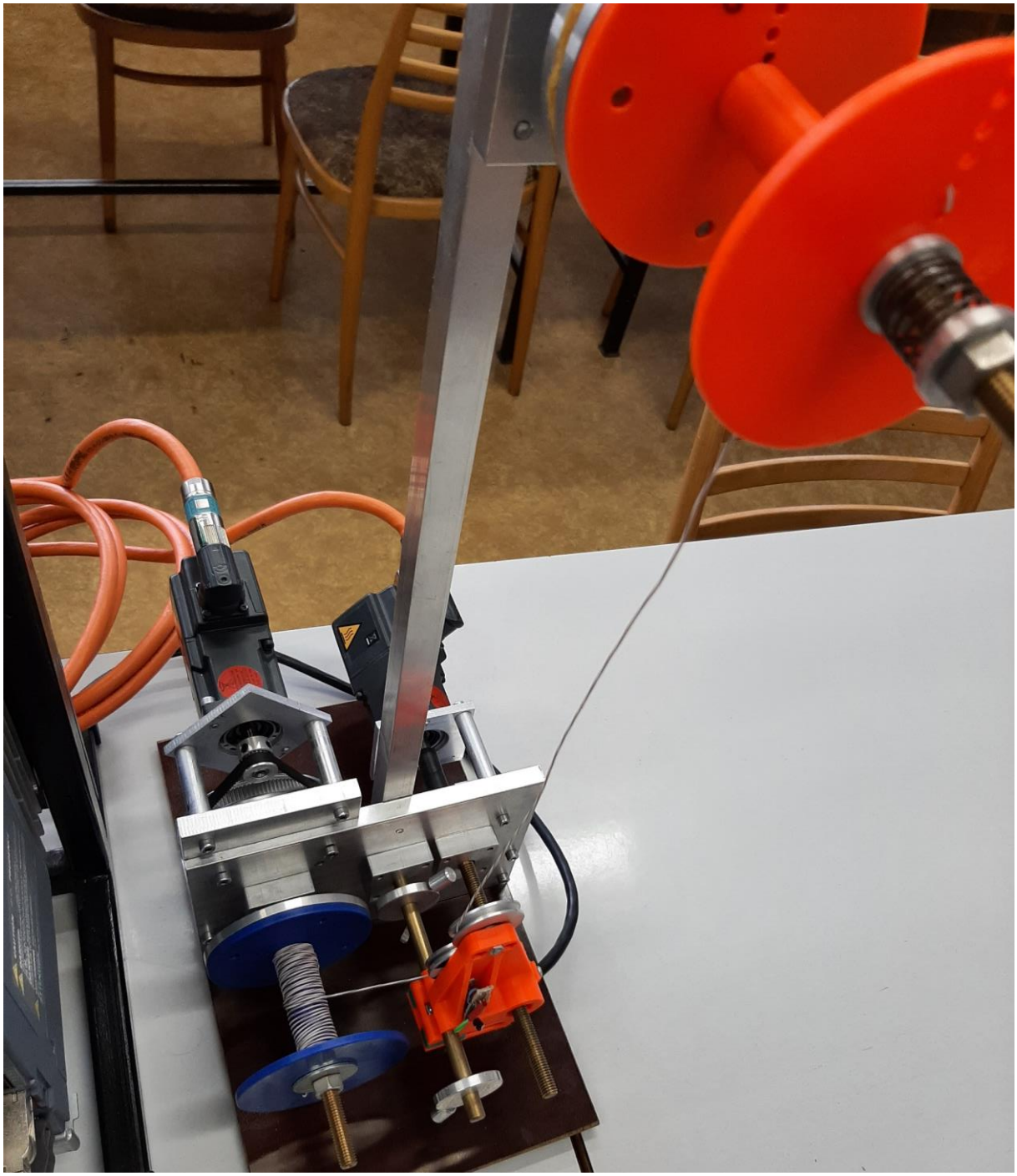
Obr. B-6 3_AUTO: Calculation mode, Spike/Stroke/Ext. Encoder



Obr. B-7 4_Visualization: Trend (s pomocným panelem), 5_Alarms



Obr. B-8 6_Project Info



Obr. B-9 Navíječka na konci navíjení (windingStep 2 mm)

PŘÍLOHA C: OBSAH PŘILOŽENÉHO CD

- Složka "Kalat_Ondrej_DP"
  Obsahuje soubory potřebné pro spuštění programu PLC v TIA portálu V15.1.
- Soubor "Kalat_Ondrej_DP_vid4_HQ.mp4"
  Obsahuje krátké video demonstrující funkčnost programu a grafického rozhraní.