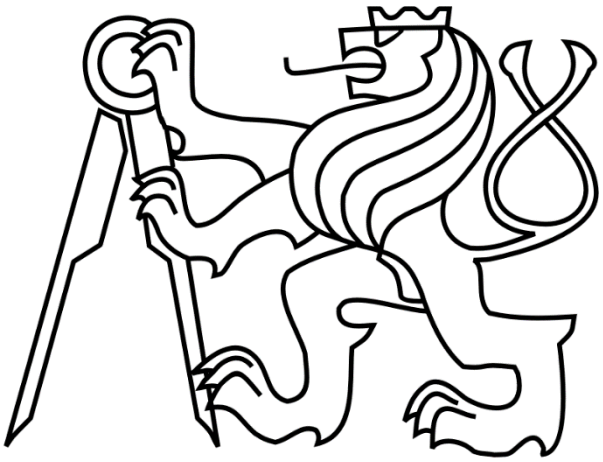
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STROJNÍ

Ústav výrobních strojů a zařízení



Bakalářská práce

Návrh řešení pro SIL simulaci výukové sestavy stroje

2017 **Jan Ferkl**

ZADÁNÍ BP

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a že jsem uvedl v přiloženém seznamu veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací, vydaným ČVUT v Praze 1. 7. 2009.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 30.6.2017 …..…..…….

podpis

Poděkování

Tímto bych rád poděkoval svému vedoucímu Ing. Lukáši Novotnému, Ph.D. za spolupráci a poskytnuté zázemí.

Anotační list

Autor:	Jan Ferkl
Název BP:	Návrh řešení pro SIL simulaci výukové sestavy stroje
Rozsah práce:	45 str., 21 obr., 2 tab.
Školní rok vyhotovení:	2017
Škola:	České vysoké učení technické v Praze – Fakulta strojní
Ústav:	Ú12135 – Ústav výrobních strojů a zařízení
Vedoucí bakalářské práce:	Ing. Lukáš Novotný Ph.D.
Zadavatel:	České vysoké učení technické v Praze – Fakulta strojní
Využití:	Simulace zpětné vazby výukové sestavy stroje pomocí SIL simulace
Klíčová slova:	PLC, řídící automat, automatizace, simulace;
Anotace:	Tato bakalářská práce se zabývá řešením Software-in-the-loop (SIL) simulace výukové sestavy stroje, která slouží k testování řídícího programu výukové sestavy stroje. Součástí této bakalářské práce je také samotné řešení řídícího programu.

Annotation

Author:	Jan Ferkl
Title of bachelor dissertation:	Solution for SIL simulation of educational machine assembly
Extent:	45 p., 21 fig., 2 tab.
Academic year:	2017
University:	CTU in Prague – Faculty of Mechanical Engineering
Department:	Ú12135 – Department of Production Machines and Equipment
Supervisor:	Ing. Lukáš Novotný Ph.D.
Submitter of the Theme:	CTU- Faculty of Mechanical Engineering
Application:	Simulation of feedback from educational machine assembly with SIL simulation.
Keywords:	PLC, automation, simulation;
Annotation:	This bacheor's thesis describes the solution to SIL simulation of the educational machine assembly, which is meant to be used for testing a controlling program of the machine assebly. The bachelor's thesis also contains the solution of a controlling program.

Obsah

Seznam použitých zkratek

Zkratka	Význam
CAD	computer aided design
ČSN	česká technická norma
di	digital input (digitální vstup)
do	digital output (digitální výstup)
EN	evropská norma
FB	funkční blok
FBD	function block diagram
F-TRIG	falling trigger
FUN	funkce
HMI	human-machine interface
IEC	international electrotechnical commision
IL	instruction list
LD	ladder diagram
NC	numeric control
obr.	obrázek
PLC	programmable logic controller
POU	program organisation unit
PROG	program
R-TRIG	rising trigger
SCADA	supervisory control and data acquisition
SCL	structured control language
SIL	software in the loop
ST	structured text
tab.	tabulka
TIA	totally integrated automation
TON	timer on
TP	time pulse

1. Úvod

Tato bakalářská práce je věnována řešení software in the loop simulace výukové sestavy stroje, která slouží jako pomůcka při výuce programování PLC. Motivací k vytvoření simulace je především omezený přístup do laboratoří, kde se zařízení nachází. Úkolem virtuálního modelu sestavy je generovat zpětnou vazbu, která bez připojení k fyzickému modelu chybí, a tedy umožnit testování řídícího programu PLC i mimo výuku.

Úvod práce je věnován stručné charakteristice zařízení. Dále je osvětlen princip simulace SIL a jeho využití v rámci testování řídícího programu zařízení. Následuje představení dostupných způsobů řešení, ale také rozbor výhod a nevýhod konkrétních přístupů. Poslední kapitola úvodu této závěrečné práce je věnována výběru jedné z uvedených variant řešení a zdůvodnění volby.

1.1. Stručná charakteristika PLC

Programovatelný automat je řídicí systém určený pro řízení technologických procesů. Své využití nachází v průmyslu i mimo něj. Převážně se ale využívá pro logické úlohy. Mimo českého názvu programovatelný automat se zařízení obecně označuje jako PLC (Programmable Logic Controller).

Pro pochopení postavení programovatelných automatů je vhodná krátká exkurze do historie. Až do šedesátých let dvacátého století veškerou průmyslovou automatizaci obstarávaly řídicí systémy založené na relé logice. Snaha o častou obměnu produkovaných automobilů však v roce 1968 motivovala americkou automobilku General Motors k hledání nového způsobu řízení automatizace. Při použití relé obvodů bylo se změnou výroby nutné obměnit i elektrické zapojení systému. Tento nešvar elegantně vyřešilo zavedení PLC řídicího systému. Při změně činnosti výroby stačí „pouze“ obměna řídícího programu, elektrické zařízení zůstává stejné, případně lehce upravené (modulární PLC). [1]

Řízení osobním počítačem nebylo vhodné z důvodu hrubého prostředí průmyslové výroby a nároků kladených na psaní řídícího programu. Tehdejším technologům blízká relé logika posloužila jako vzor pro vznik grafického programovacího jazyka LD (Ladder Diagram), kterým se PLC programuje do dnes. Evropský protějšek jazyka LD, který vznikl v USA je textový jazyk IL (Instruction List). Jedná se o seznam instrukcí, jež svou strukturou připomíná assembler. Procesnímu průmyslu blízký je další grafický jazyk

a to FBD (Function Block Diagram). Grafické bloky zde reprezentují funkce, funkční bloky a programy, které se starají o zpracovávání signálů. Je tedy možné si je představit jako hradla v elektrických obvodech. Poslední z běžně užívaných programovacích jazyků je textový ST (Structured Text). Obsahuje všechny prvky moderního programovacího jazyka, je velmi sofistikovaný a blízký jazykům Pascal nebo C. [1], [2]

Automobilce se nově zavedená technologie více než osvědčila a za nedlouho nový řídicí systém nabízela celá řada výrobců. K logickým operacím s proměnnými typu BOOL se brzy přidaly i výpočty. Nároky na PLC neustále rostly a s nimi i vyspělost nabízených systémů. PLC se tak naučily pracovat s texty a převádět mezi různými formáty hodnot zadávaných a zobrazovaných. O vizualizaci a komunikaci s obsluhou se postaraly systému SCADA a HMI. Výrobci se předháněly v nabízených funkcích, upravovaly programovací jazyky a v neposlední řadě zlepšovaly hardwarové vybavení. Vývoj pokračoval mílovými kroky a za nedlouho nastala situace, kdy znalost PLC jednoho výrobce ani z daleka nestačila k práci s řídicím systémem výrobce druhého. Tato skutečnost kladla nemalé nároky na programátory a vedla ke vzniku mezinárodně uznávané normy IEC 61 131-3 na přelomu osmdesátých a devadesátých let dvacátého století. [1]

O vytvoření normy se postarala organizace PLC open. První vydání vyšlo v roce 1993 a je věnováno pouze sjednocení požadavků kladených na PLC. Postupně přibývaly další části zaměřené na základní informace o PLC, požadavky na hardwarové provedení a jeho testování ale také část věnovaná kodifikaci způsobů programování a syntaxi unifikované sady programovacích jazyků (LD, IL, FBD, ST). Z počátku pro jednotlivé výrobce neměla valný přínos, dnes už ale patří k základním předpokladům úspěšného působení na trhu s PLC řídícími systémy. Norma byla přijata jako evropská (EN) a i jako česká (ČSN). [1]

Programování PLC je samostatná disciplína, která se od psaní programu pro počítač liší především přítomností mechanického zařízení, které PLC ovládá, a to pomocí digitálních ale i analogových vstupů a výstupů. Použité PLC a jeho program se označují souhrnným názvem projekt řídicího systému. Samotný řídící program se pak skládá z jednotlivých POU (Program Organization Unit). Jedná se o nejmenší nezávislou část uživatelského programu a je buď psaná uživatelem nebo dodaná výrobcem. POU se dále dělí na tři základní typy: funkce, funkční bloky a program. [1], [2]

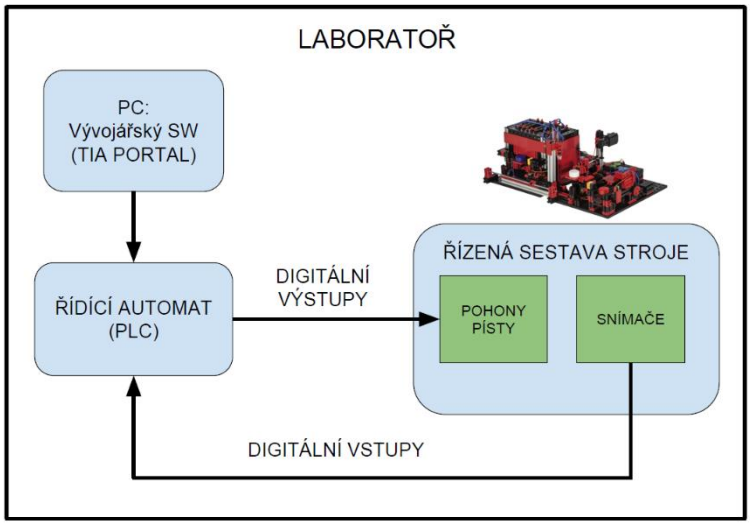
Funkce (FUN) se považuje za nejjednodušší formu POU. Zpracovává vstupní proměnné a vrací pouze jeden parametr výstupní. Na rozdíl od funkčního bloku nemá vlastní paměť a výstupní parametr je tedy závislý pouze na vstupních hodnotách. [2]

Funkční blok (FB) je vyšší forma POU než funkce. Je schopen pracovat s vyšším počtem vstupních parametrů a vracet více výstupních parametrů. Charakteristická vlastnost funkčního bloku je také možnost vlastnit paměť. Výstupní parametry se tedy při volání se shodnými vstupními hodnotami mohou lišit, a to právě vlivem vnitřních (lokálních) proměnných. Základní forma FB je jeho deklarace. Jedná se o předpis algoritmů a struktury dat, podle kterých pak FB vykonává svou činnost. Přítomnost vnitřní paměti ale vyžaduje při konkrétním použití FB vytvořit vlastní kopii FB, která má vlastní název a přesně stanovené vstupní a výstupní proměnné. Popsanému jevu se říká instance funkčního bloku. [2]

Program (PROG) představuje nejvyšší formu POU. Norma jej definuje jako „logický souhrn programovacích jazyků a konstrukcí nutných pro zamýšlené zpracování signálů, které je vyžadováno pro zamýšlené řízení stroje nebo procesu systémem programovatelného automatu“. Program může volat funkce a funkční bloky. Existuje celá řada typu programů (například „startup“, který proběhne pouze jednou po zapnutí zařízení nebo „main“, který běží cyklicky v každém taktu PLC). [1], [3]

1.2. Software-in-the-loop simulace řízené sestavy stroje

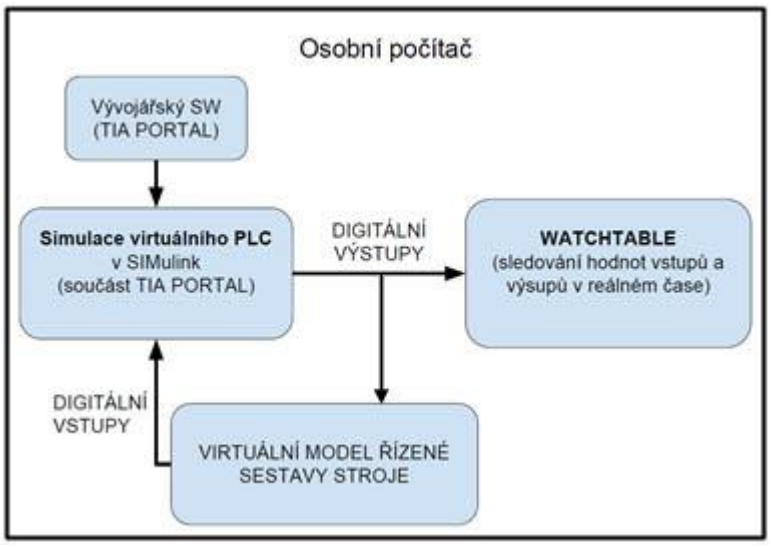
Pro pochopení významu samotné simulace, konkrétně pak simulace SIL, je vhodné nejprve popsat ideální situaci (Obr. 1.: Blokové schéma ideální situace), kdy programátor, píšící řídící program PLC, má k dispozici veškeré zařízení (PLC včetně HMI panelu a řízený stroj).



Obr. 1.: Blokové schéma ideální situace

Programátor pracuje na osobním počítači a ve vývojářském prostředí (v případě PLC od firmy Siemens se jedná o program TIA Portal) píše řídící program, který následně pomocí síťového kabelu nahraje do PLC. K tomu je kromě osobního počítače také připojeno řízené zařízení, které je osazeno aktory a snímači. Pomocí digitálních výstupů PLC jsou řízeny aktory a naopak zařízení o sobě poskytuje informace ve formě digitálních vstupů generovaných snímači. Pro chod, a tedy i testování řídícího programu je potřeba: PLC, digitální vstupy PLC a možnost ovládat manuálně řízené proměnné (pomocí HMI panelu nebo osobního počítače připojeného k PLC).

Obr. 2.: Blokové schéma situace bez přístupu k řízené sestavě znázorňuje případ, kdy se plnohodnotný chod řídícího programu odehrává pouze v rámci osobního počítače. Vedle TIA Portal firma Siemens také nabízí nástroj určený k softwarové simulaci řídícího automatu zvaný PLCSIM. Do virtuálního PLC je analogickým způsobem jako do PLC skutečného (hardwarového) nahrán řídící program a pomocí tak zvaných sledovacích tabulek (v anglické verzi TIA Portal „watch tabels“) jsou sledovány hodnoty výstupů řídícího programu. Bez připojené sestavy stroje ale chybí hodnoty digitálních vstupů PLC, které jsou pro chod programu nepostradatelné. Je tedy nutné vytvořit virtuální model řízeného stroje, který bude schopen generovat potřebnou zpětnou vazbu. Celá simulace se odehrává pouze softwarově. Jedná se o simulaci nazývanou Software-In-the-Loop (software ve smyčce, software vrací zpětnou vazbu softwaru). Řešení popsané problematiky a tvorbě virtuálního modelu řízené sestavy stroje je věnována tato závěrečná práce.



Obr. 2.: Blokové schéma situace bez přístupu k řízené sestavě

1.3. Motivace

Přínosů simulace je celá řada. Tato krátká kapitola je věnována pouze stručnému nastínění významu simulace při automatizaci průmyslových zařízení.

V dosavadní praxi byla tvorba řídícího programu striktně fixována na vznik prototypu případně výsledného zařízení. Možnosti vývojáře řídícího programu byly velmi omezené a vázané na konstrukci. Tato skutečnost mnohdy vedla k snižování časových prostředků, které byly pro automatizaci vyhrazeny. Virtuální model vyráběného stroje popsanou problematiku plnohodnotně řeší. Umožňuje pracovat na tvorbě řídícího programu paralelně s vývojem konstrukce.

Další možnosti a výhody simulace průmyslového zařízení jsou detailně popsány v kapitole 1.4.2.

Při výuce tvorby řídících programů, konkrétně pak ve školství, zmíněná pozitiva přestávají být zásadní. Výukové modely průmyslových zařízení jsou k dispozici ve finální podobě. Problém je však jejich omezená dostupnost, a to pouze v prostorách školních laboratoří. Okamžitá zpětná vazba pro nezkušeného programátora je však velmi přínosná a mimo výuku nedostupná. Simulace SIL umožňuje studentům otestovat vlastní zdrojový kód, a to i v případě, že zařízení není fyzicky přítomné. Ke spuštění napsaného programu pak stačí osobní počítač dostatečně softwarově vybavený pro simulaci PLC a výukové sestavy stroje.

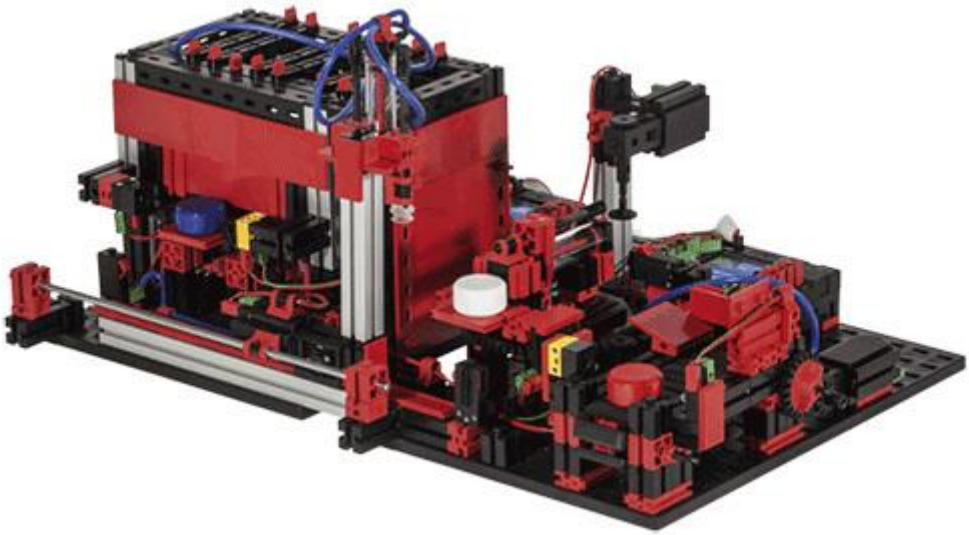
1.3.1. Projekt III.

Tato bakalářská práce vznikla ve spolupráci s Ústavem výrobních strojů a zařízení Fakulty strojní ČVUT, kde se automatizace vyučuje v rámci předmětu Projekt III. Školní laboratoře jsou vybaveny PLC Simatic S7-1500 od firmy Siemens opatřenými HMI panely KTP700 Basic PN a sadou modelů průmyslových zařízení od firmy Fischertechnik.

Studenti na přiděleném stroji mají za úkol vymyslet scénář provozu včetně požadavků na obsluhu a ošetření chybových stavů. Zbytek semestru je věnovaný zprovoznění a odladění řídících programů. Tento způsob výuky je velmi blízký praxi a poskytuje absolventům cenné zkušenosti z oblasti automatického řízení. Přínos praktické části této bakalářské práce spočívá ve vytvoření virtuálního modelu řízené sestavy stroje, který by studenti mohli využít pro samostatnou práci mimo hodiny předmětu Projekt III.

1.3.2. Výuková sestava stroje

Tato bakalářská práce je věnována tvorbě virtuálního modelu výukové sestavy stroje, konkrétně pak stylizovaného modelu pracoviště, které se skládá z žíhací pece, podtlakového dopravníku, otočného stolu, frézky a pásového dopravníku. (Obr. 3.: Výuková sestava stroje [4]) Tato sestava nese katalogové označení „536632 - Multi Processing Station with oven 24V“, výrobcem je německá společnost Fischertechnik.



Obr. 3.: Výuková sestava stroje [4]

Toto zařízení je vybaveno celou řadou aktorů, tedy akčních prvků ovládaných digitálními výstupy řídícího automatu. Jedná se o elektromotory, kompresor, písty

a světlo uvnitř pece. Všechny digitální výstupy (do) včetně bitových adres v paměti PLC jsou uvedeny v Tab. 1.: Výpis digitálních výstupů PLC.

Tab. 1.: Výpis digitálních výstupů PLC

digitální výstupy	popis	název proměnné
do_0.0	motor otočného stolu, směr posuvný pás	do_pohyb_ot_stul_pas
do_0.1	motor otočného stolu, směr úchop	do_pohyb_ot_stul_uchop
do_0.2	motor pásového dopravníku	do_pas
do_0.3	motor frézky	do_frezka
do_0.4	motor stolu pece, směr dovnitř	do_pohyb_stul_p_dovnitr
do_0.5	motor stolu pece, směr ven	do_pohyb_stul_p_ven
do_0.6	motor úchopu, směr pec	do_pohyb_uchop_pec
do_0.7	motor úchopu, směr otočný stůl	do_pohyb_uchop_ot_stul
do_1.0	světlo pece	do_svetlo
do_1.1	kompresor	do_kompresor
do_1.2	podtlak úchopu	do_podtlak
do_1.3	uchopení obrobku	do_chnap
do_1.4	vrata pece, pozice otevřeno	do_vrata
do_1.5	píst otočného stolu	do_pist

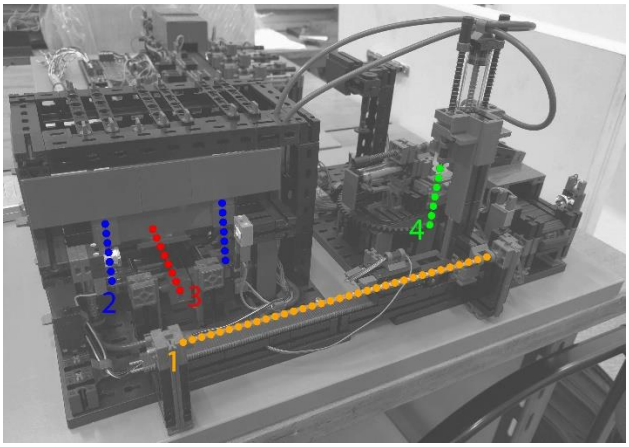
Krom aktorů je sestava stroje také osazena snímači, které detekují stav zařízení. Koncová čidla indikují krajní polohu polohovatelných komponent (stolu pece, podtlakového úchopu a otočného stolu). Světelné závory dávají informace o přítomnosti obrobku a zvláštním případem je digitální vstup CENTRAL STOP, tedy prvek sloužící k nouzovému zastavení provozu. Kompletní výpis vstupů PLC včetně bitových adres je uveden v Tab. 2.: Výpis digitálních vstupů PLC.

Tab. 2.: Výpis digitálních vstupů PLC

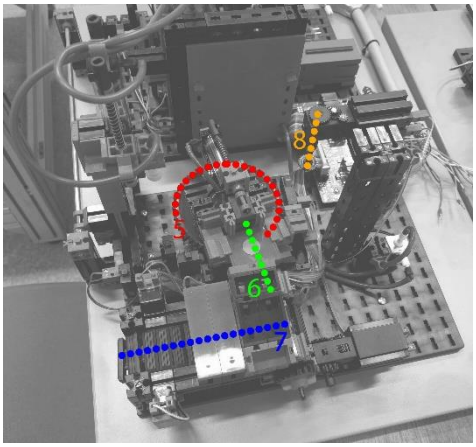
digitální vstupy	popis	název proměnné
di_0.0	koncové čidlo otočného stolu, pozice úchop	di_koncak_ot_stul_uchop
di_0.1	koncové čidlo otočného stolu, pozice pás	di_koncak_ot_stul_pas
di_0.2	světelná závora pásu	di_zavora_pas
di_0.3	čidlo otočného stolu, pozice frézka	di_ot_stul_frezka
di_0.4	koncové čidlo úchopu, pozice otočný stůl	di_koncak_uchop_ot_stul
di_0.5	koncové čidlo stolu pece, pozice uvnitř	di_koncak_stul_p_dovnitr
di_0.6	koncové čidlo stolu pece, pozice vně	di_koncak_stul_p_ven
di_0.7	koncové čidlo úchopu, pozice pec	di_koncak_uchop_pec
di_1.0	světelná závora pece	di_zavora_pec
di_1.1	central stop	di_central_stop

Obr. 4.: Kinematické schéma č. 1 znázorňuje osy pohybů zařízení pece a podtlakového manipulátoru. Číslem 1 je označena pohybová osa podtlakového úchopu, po které se toto zařízení polohuje. 2 je naznačen rozsah pohybu vrat pece. Možnost

polohování stolu pece určuje osa číslo 3. Vertikální pohyb ramene podtlakového dopravníku symbolizuje číslo 4.



Obr. 4.: Kinematické schéma č. 1



Obr. 5.: Kinematické schéma č. 2

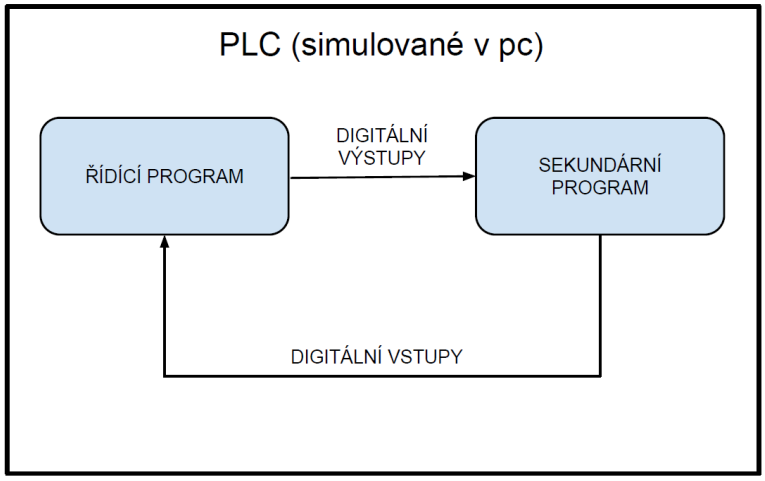
Obr. 5.: Kinematické schéma č. 2 zobrazuje veškeré pohyby, které koná zařízení otočný stůl, frézka a pásový dopravník. Číslem 5 je znázorněna rotace otočného stolu, 6 pak vyznačuje pohyb pístu, kterým se obrobek přesune z otočného stolu na pásový dopravník. Dráhu, kterou obrobek urazí při pohybu pásu, symbolizuje číslo 7 a dále 8 naznačuje osu rotace nástroje frézky.

1.4. Možnosti řešení

Pro tvorbu virtuálního modelu sestavy stroje, která by sloužila k testování řídících PLC programů jsou dostupné dva způsoby řešení. Jejich detailnímu představení, srovnání a následnému výběru jsou věnovány následující kapitoly.

1.4.1. Sekundární program

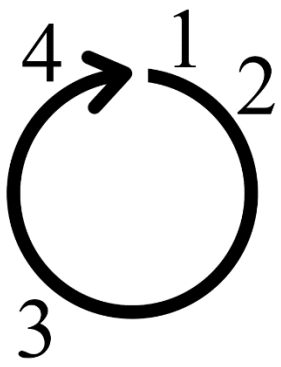
První, jednodušší řešení spočívá ve vytvoření sekundárního programu, který by pracoval na pozadí s programem řídícím na stejném virtuálním PLC. Tento program by sledoval výsledky práce programu řídícího a na jejich základě generoval potřebnou zpětnou vazbu. Jinak řečeno: vstupy sekundárního programu jsou výstupy programu řídícího, a naopak výstupní hodnoty sekundárního programu slouží jako hodnoty vstupní pro program řídící. Popsaný princip je schematicky znázorněn na Obr. 6.: Komunikace řídícího a sekundárního programu.



Obr. 6.: Komunikace řídícího a sekundárního programu

Jak už bylo řečeno v kapitole 1.1, řídicí automaty pracují v taktech. Jeden takt řádově trvá desítky až stovky milisekund (zásadní vliv má náročnost vykonávaného programu na výpočetní výkon). Během každého taktu proběhne cyklus znázorněný na Obr. 7.: Takt PLC.

Pozice 1 (na Obr. 7.: Takt PLC) symbolizuje zahájení cyklu, kdy dochází ke čtení vstupních proměnných PLC. V tuto chvíli je vhodné poznamenat, že vykonávané programy sice běží z pohledu uživatele paralelně, ale je možné určit, který z nich v rámci jednotlivých taktů řídícího automatu proběhne přednostně. Po načtení vstupních proměnných následuje vykonávání samotných programů. Je velmi důležité, aby sekundární program (pozice 2) proběhl před zahájením chodu programu řídícího (pozice 3) a stihl tak přepsat digitální vstupy na hodnoty nově vytvořené (simulované). Celý cyklus uzavírá zápis výstupních proměnných PLC (výstup práce řídícího programu; pozice 4).



Obr. 7.: Takt PLC

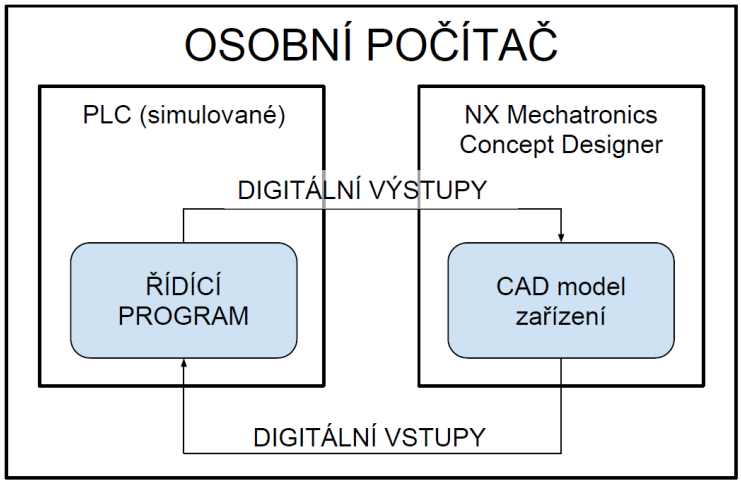
Z principu práce PLC v taktech vyplývá, že hodnoty vstupů se nepřenášejí mezi jednotlivými cykly. Tedy každá hodnota vytvořená sekundárním program pro konkrétní digitální vstup je ukončením taktu ztracena. Z toho důvodu je potřeba pro každý digitální vstup vytvořit pomocnou proměnnou, která disponuje vlastním bitem v paměti PLC, její hodnotu určuje sekundární program a v rámci každého taktu je zapsána do příslušné vstupní proměnné.

Přístup popsaný v této kapitole je účelný. Plnohodnotně splňuje jediný požadavek kladený ze strany programátora automatizace, a sice simulace zpětné vazby zařízení (digitálních vstupů PLC). Mezi výhody zcela jistě patří také nulové zvýšení požadavků na softwarové vybavení osobního počítače uživatele a možnost naprogramování chybových stavů zařízení (vhodné k testování bezpečnosti řídícího programu). Tento virtuální model slouží pouze jako pomůcka při psaní řídícího programu. Tato skutečnost má za důsledek relativně nízké časové nároky na realizaci.

Nevýhodou tohoto řešení je velmi specifické využití. Pro práci se sekundárním program je nutné prvotní seznámení se skutečným zařízením. Tato negativa se při využití ve výuce předmětu Projekt III. neprojeví.

1.4.2. Virtuální zprovoznění

Druhý možný přístup nese název „virtuální zprovoznění“. Spočívá ve vytvoření CAD modelu zařízení (tak zvané virtuální dvojče) ve virtuálním prostředí. Tento model je identický se zařízením skutečným a je tedy vybaven shodnými aktory a snímači. Realizaci popsaného virtuálního modelu umožňuje software NX Mechatronics Concept Designer (řešení od firmy Siemens). Uvedený program je schopný zohlednit významné fyzikální vlivy, které při provozu působí. Výsledné simulované zařízení je schopné vykonávat shodné úkony jako reálný stroj. Nadefinované snímače pak simulovanému PLC vracejí odpovídající zpětnou vazbu (hodnoty digitálních vstupů PLC). Princip komunikace virtuálního modelu a simulovaného PLC znázorňuje Obr. 8.: Komunikace v rámci „virtuálního zprovoznění“.



Obr. 8.: Komunikace v rámci „virtuálního zprovoznění“

„Virtuální zprovoznění“ patří mezi nové, progresivní metody v průmyslové praxi, kde, jak již bylo popsáno v kapitole 1.3, výrazně zlepšuje časové možnosti programátora. Jeho další přínos také spočívá ve zlepšení pozice výrobce zařízení při jednání se zákazníkem. Konkrétní detaily a specifikace zakázky mohou být upřesněny na základě virtuálního zprovoznění, a to bez nutnosti vzniku fyzického prototypu stroje.

Širší využití tohoto řešení má za následek zvýšení nároků, a to jak na vybavenost osobního počítače, tak tvůrce virtuálního dvojčete. Ten se neobejde bez znalosti (vedle TIA Portal) dalšího programu.

1.5. Volba řešení, zdůvodnění

Metoda virtuálního zprovoznění své uplatnění nachází především v průmyslové praxi, kdy výrobce zařízení je schopen na požadavky zákazníka reagovat konkrétními specifikacemi (na příklad strojním časem a NC kódem pro výrobu požadovaného obrobku). Uvedené přínosy ale nenacházejí uplatnění v rámci školní výuky.

Očekávané výsledky naopak velmi dobře splňuje relativně nenáročné řešení situace pomocí sekundárního programu.

Na základě pozitiv a negativ uvedených v kapitolách věnovaných konkrétním způsobům řešení volím přístup popsaný v kapitole 1.4.1, tedy sekundární program. Pro školní účely považuji řešení „virtuální zprovoznění“ za příliš komplikované a tedy nevhodné. Není žádoucí zvyšovat nároky kladené na zprovoznění simulace, pokud jejím účelem je zpřístupnit testování řídícího programu.

2. Řešení řídícího programu

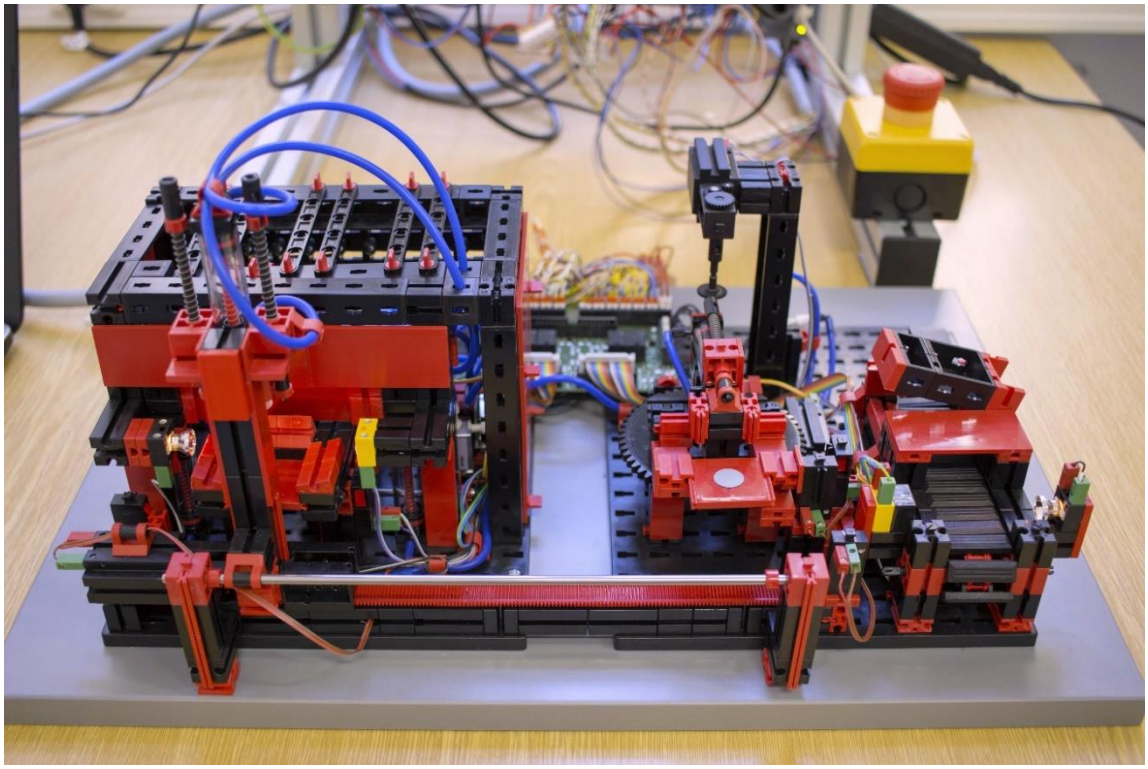
Jak již bylo zmíněno, výuková sestava stroje je řízena PLC od firmy Siemens, a to modelem Simatic S7-1500. Samotný řídící program je pak psán ve vývojářském prostředí TIA Portal v jazyce strukturovaného textu (v prostředí TIA Portal je strukturovaný text označován zkratkou SCL- Structured Control Language). Výše uvedený software je velmi rozsáhlý a univerzální nástroj. Tato práce se zabývá zejména psaním zdrojového kódu řídícího a sekundárního programu. Problematika tvorby projektu a zajištění správné komunikace mezi jednotlivými komponenty (PLC a HMI) v této práci popsána není.

Vytyčené cíle pro tvorbu řídícího programu korespondují s přítomnými digitálními vstupy (snímači) a výstupy (aktory) PLC, které určují možnosti automatizace. Samozřejmostí je využití veškerých přítomných zařízení na pracovišti (celé trati) a zajištění co pokud možná nejplynulejšího provozu. Požadavkem také je možnost vložení druhého obrobku po uvolnění začátku trati (stolu pece). V neposlední řadě by měl být řídící program bezpečný, této problematice je věnována kapitola 2.4.6.

2.1. Popis trati pracoviště

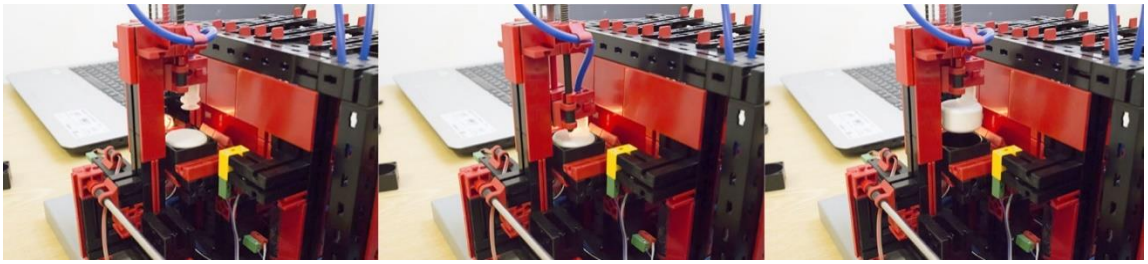
Výuková sestava stroje je stylizovaný model pracoviště, které sestává z žíhací pece, podtlakového dopravníku, otočného stolu vybaveného frézkou a pásového dopravníku. Tato kapitole je věnována stručnému popisu scénáře práce celého zařízení.

Trať začíná stolem pece, který se v základní pozici nachází vně pece, a na který je obsluhou umístěn obrobek. Následuje otevření vrat pece a polohování stolu pece na vnitřní koncový snímač. Provoz pokračuje zavřením vrat a rozsvícení světla uvnitř komory pece (symbolizuje vytápění). Po dokončení tepelného zpracování obrobku se vrata pece opět otevřou a stůl se s obrobkem vrací na pozici vně pece, kde je obrobek připraven na uchopení podtlakovým dopravníkem.



Obr. 9.: Výuková sestava stroje

Podtlakový úchop obstarává přenos vyžíhaných obrobků na otočný stůl. Podtlak vytváří dvojice pístů a kompresor. Rameno zařízení svislým pohybem dolů zajistí kontakt mezi obrobkem a těsnící gumovou hlavicí. Následuje spuštění podtlaku a návrat ramene do horní polohy, ve které setrvá, než se celý úchop přesune na pozici otočný stůl, kde obrobek odloží (obdobným postupem, ale v opačném sledu operací).

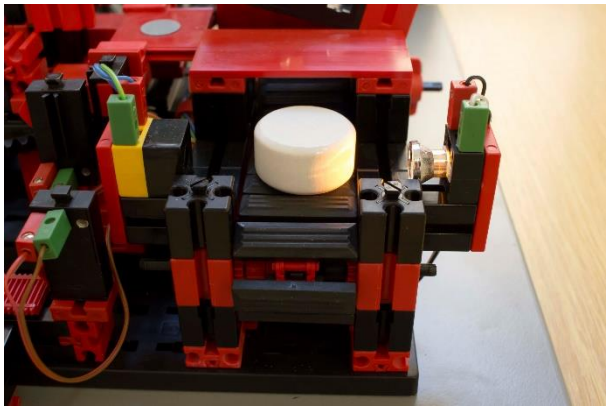


Obr. 10.: Uchopení obrobku podtlakovým dopravníkem

Otočný stůl se poté co obdrží obrobek přesouvá na pozici frézka, na které (symbolicky) probíhá třískové obrábění obrobku. Po jeho dokončení otočný stůl pokračuje v pohybu až na koncový snímač pozice pás. Zde pak píst otočného stolu přesouvá obrobek na pásový dopravník. Pohyb pásu obrobek dopraví na samotný konec trati, který je snímán světelnou závorou. Po přerušení závory obrobkem přestává pás konat pohyb a očekává se odebrání obrobku obsluhou zařízení. (Obr. 11.: Konec trati)

Jak je uvedeno v předešlých odstavcích, pracoviště je složeno z několika zařízení.

Tato skutečnost umožňuje nezávislý provoz jednotlivých úkonů, a tedy zpracovávání více než jednoho obrobku současně. Tímto způsobem je možné výrazně zvýšit produktivitu. Detailní popis řešení je uveden v následujících kapitolách.



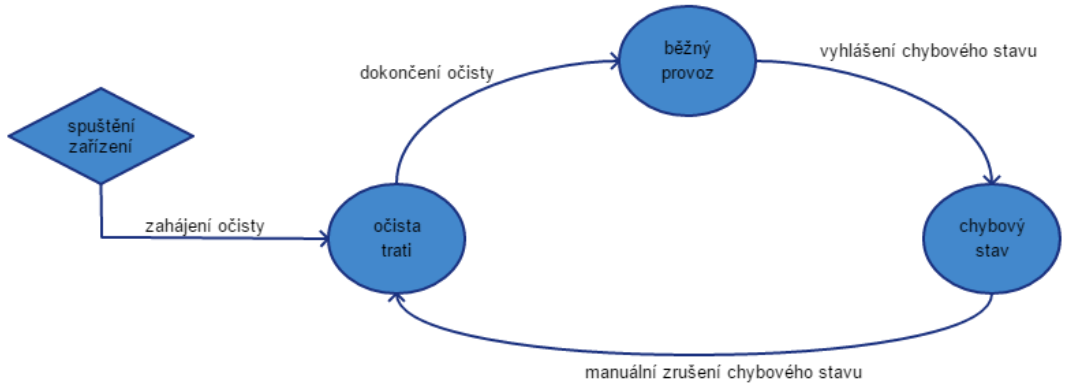
Obr. 11.: Konec trati

2.2. Požadavky na řídící program

Tato kapitola se věnuje popisu konkrétních požadavků na výstup práce řídícího programu. Patří mezi ně ovládání digitálních výstupů (aktorů zařízení) a to s přihlédnutím na produktivitu a bezpečnost provozu, ale také komunikace s obsluhou zařízení za pomoci dotykového HMI panelu.

Scénář běžného chodu zařízení, který byl popsán v předešlé kapitole jako sled operací, kterými obrobek během projetí tratě projde, je přímočarý a popisuje práce na jednom vloženém obrobku. Z důvodu zvýšení propustnosti zařízení je umožněno na trati pracovat s jedním i více dílci. Při paralelním provozu více než jednoho stroje sestavy je nutno zajistit, aby na sebe příslušné úkony smysluplně navazovaly. Na příklad je nevhodné, aby byl obrobek ze stolu pece uchopen ještě dříve, než otočný stůl opustí předešlý obrobek a je dokončeno polohování na pozici úchop (než je zajištěn prostor pro odložení uchopeného obrobku). Tato problematika je řešena pomocí následujících podmínek: otočný stůl se na pozici úchop (kde se na něj odkládá obrobek) polohuje až po odebrání předcházejícího obrobku z pásového dopravníku; odvoz obrobku ze stolu pece na otočný stůl je zahájen až po sepnutí koncového snímače otočného stolu pozice úchop (tedy ve chvíli, kdy je kam odkládat) a práce na nově vloženém obrobku obsluha může spustit až poté, co podtlakový dopravník uvolní stůl pece (do tohoto okamžiku obsluha ani nemá nový obrobek kam vložit). Realizace výše uvedených principů ve zdrojovém kódu je detailně popsána v kapitole 2.4.1.

Z důvodu bezpečnosti je v řídícím programu ošetřen i možný chybový stav zařízení. Chybový stav je vyhlášen pokud: doba polohování některé řízené osy je delší než přesně stanovená bezpečná hodnota (ošetření proti nefungujícímu koncovému snímači); obrobek nedorazil na konec trati; obsluha zmáčkla nouzové tlačítko CENTRAL STOP. Při vyhlášení chyby je nutné, aby se zařízení přestalo pohybovat (jediný pohyb je otevření vrat pece). Také je nutné zajistit, aby kompresor držel podtlak, pokud je spuštěn (a tedy nedošlo k upuštění obrobku převáženého podtlakovým dopravníkem) a bylo zastaveno vytápění pece. Poté co obsluha manuálně zruší chybový stav je obrobek odložen na otočný stůl (pokud je uchopen podtlakovým dopravníkem), stůl pece se polohuje na koncové čidlo vně pece a následně je zahájena očista celé trati, která zaručí, že je pracoviště opět připraveno k běžnému provozu.



Obr. 12.: Diagram základních stavů PLC

Komunikace s obsluhou probíhá pomocí dotykové obrazovky HMI panelu. Při běžném provozu je potřeba aby obsluha stroje ovládala pomocné proměnné řídícího programu, kterými oznamuje určité stavy (na příklad vložení obrobku) nebo příkazy (zahájení očisty trati). Manuálně ovládané úkony stroje jsou uvedeny v následující kapitole.

2.3. Požadavky na obsluhu zařízení

Po startu zařízení je celé pracoviště v nespecifikovaném stavu, a proto je nutné provést očistu celého zařízení, po které je jednak trať zaručeně prázdná, ale také se celé zařízení uvede do základní polohy vhodné k provozu. Úkolem obsluhy je pouze manuálně zahájit očistu zařízení, a to pomocí příslušného tlačítka na panelu „zahajit ocistu“, které je viditelné jen tehdy, pokud ještě očista neproběhla.

Po vykonání očisty je zařízení připraveno k provozu a čeká na vložení obrobku. Poté co obsluha vloží paletu s obrobkem na stůl pece, je potřeba zahájit práci zařízení. K tomu slouží tlačítko „vlozen obrobek“. Řídící program disponuje pomocnou proměnnou „g_02_man_vlozen_obrobek“, do které je po zmáčknutí tlačítka zapsána hodnota 1 a tím se provoz spustí. Naopak odvoz obrobku podtlakovým dopravníkem ze stolu pece do této proměnné zapíše hodnotu 0 a tím znovu umožní vložení dalšího dílce a zmáčknutí tlačítka. Nutno poznamenat, že podtlakový manipulátor přepravuje samotný obrobek. Prázdná paleta zůstane na stolu pece a je potřeba jí před vložením nového obrobku odebrat.

Poté co obrobek projde celou tratí se zastaví na konci pásového dopravníku, kde se nachází světelné závora. Přerušení světelné závory zastaví pohyb pásu a je očekáván odběr obrobku. Jakmile obsluha uvolní pásový dopravník dojde k obnovení světelné závory. Odebrání obrobku ze zařízení tedy není nutné manuálně potvrdit.

Poslední manuálně ovládaný prvek se týká chybového stavu zařízení. Pokud nastane chyba, je provoz zastaven, jak bylo uvedeno v předešlé kapitole. Pro ukončení chybového stavu je na obrazovce umístěno tlačítko „zrusit chybu“, které je viditelné jen v okamžiku, kdy chybový stav nastal.

2.4. Struktura

Několik následujících kapitol se věnuje strukturování zdrojového kódu řídícího programu a použitým principům. V kapitole 2.4.1 je popsáno a odůvodněno rozčlenění programu do jednotlivých kapitol. V rámci kapitol jsou často použity funkční bloky, a to předdefinované nebo uživatelem vytvořené. Princip práce s funkčními bloky (FB) je uveden v kapitole 2.4.2. Problematice zápisu do digitálních výstupů PLC je věnována kapitola 2.4.3.

Uvedení výše vyjmenovaných principů do praxe jsou pak určeny další tři kapitoly, věnované očistě trati, běžnému provozu a ošetření chybového stavu.

2.4.1. Rozdělení řídícího programu

Jak již bylo uvedeno v předešlých kapitolách, PLC pracuje v taktech, které řádově trvají zlomky sekund. Během každého taktu je vykonán celý program. Tato skutečnost vede k potřebě zdrojový kód rozčlenit do jednotlivých kapitol, pomocí kterých je přesně

určováno, která část programu je v příslušném okamžiku aktivní a naopak. Slouží k tomu příkaz „IF“, kterým je vykonávání dané části zdrojového kódu podmíněno.

Zdrojový kód, který výraz „IF“ obsahuje, je vykonán jen v okamžiku, kdy uvedená podmínka je pravdivá (booleovský výraz je roven hodnotě 1). Každá kapitola řídícího programu disponuje svou vlastní pomocnou proměnnou typu bit (tyto proměnné jsou označeny s1, s2 až s8), která funguje jako podmínka provedení zdrojového kódu příslušné kapitoly.

Tímto způsobem je možné určovat, jaké oddíly řídícího programu se zrovna uvedou do provozu. V rámci jedné kapitoly je pak vykonáván určitý úsek provozu zařízení. Po jeho dokončení se do příslušného bitu zapíše nula (čímž skončí jeho vykonávání) a naopak do pomocného bitu následující kapitoly se zapíše hodnota 1 (a tedy se zahájí vykonávání další kapitoly). Tímto způsobem se kapitoly navzájem aktivují a deaktivují.

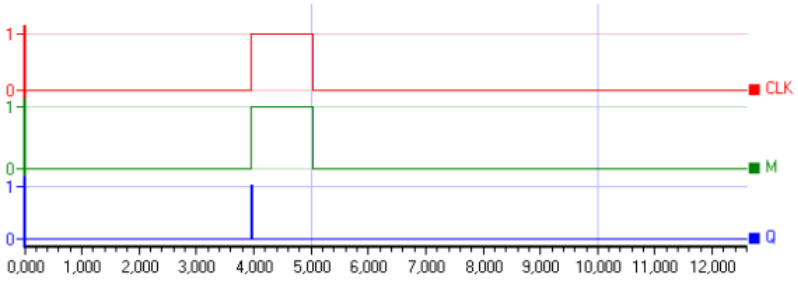
Uvedený způsob podmiňování také umožňuje paralelní chod dvou a více kapitol. Jinak řečeno nezávislý provoz jednotlivých zařízení, ze kterých se pracoviště skládá.

2.4.2. Funkční bloky

Funkční blok lze charakterizovat, jako programovou organizační jednotku, která disponuje vlastní pamětí. Výsledný výstup FB tedy není pouze funkcí vstupních proměnných, ale i proměnných vnitřních. V rámci řídícího programu lze každý předem definovaný FB opakovaně volat. Každé volání tedy musí mít přiřazenou vlastní paměť, aby FB mohly pracovat na sobě nezávisle (instance funkčního bloku).

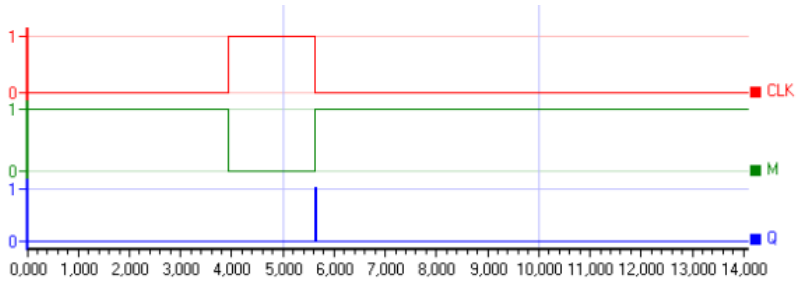
Při psaní řídícího programu byly využity následující předdefinované FB (tedy FB, kterými výrobce software sám vybavil).

R-TRIG, neboli detekce náběžné hrany. Tento FB sleduje hodnotu booleovské vstupní proměnné a v případě, že se hodnota vstupu změní z 0 na 1, výstupní booleovská proměnná nabyde hodnoty 1 právě na dobu jednoho taktu PLC. Obr. 13.: Graf hodnot FB R-TRIG znázorňuje hodnoty proměnných funkčního bloku v čase. „CLK“ označuje vstupní proměnnou a „Q“ výstupní.



Obr. 13.: Graf hodnot FB R-TRIG [5]

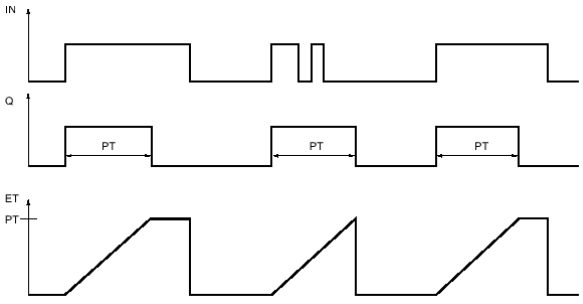
F-TRIG, neboli detekce sestupné hrany. Jedná se o obdobu FB R-TRIG jen s tím rozdílem, že reaguje na změnu vstupní proměnné z hodnoty 1 na hodnotu 0. Princip je znázorněn na Obr. 14.: Graf hodnot FB F-TRIG.



Obr. 14.: Graf hodnot FB F-TRIG [5]

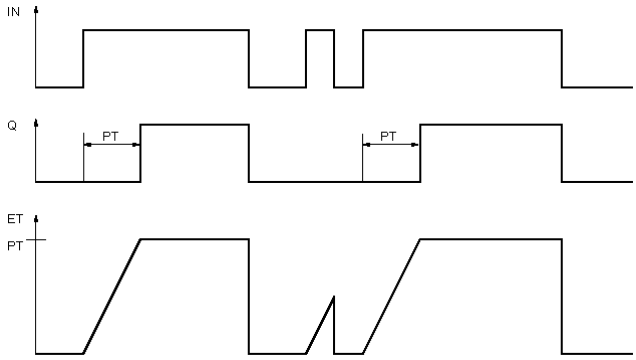
Funkční bloky R-TRIG a F-TRIG jsou vhodné k řízení návaznosti jednotlivých úkonů. FB R-TRIG byl na příklad použit při čekání na odebrání obrobku na konci trati (při přerušení světelné závory). Ve chvíli kdy obsluha odebere dílec dojde k obnovení světelného toku a digitální vstup di_0.2 (světelná závora pásu) tak znovu nabyde hodnoty 1. Pokud tento digitální vstup použijeme jako vstupní proměnnou FB R-TRIG, můžeme jeho výstupní proměnnou použít jako podmínku zahájení následující operace.

Funkční blok **TP** patří mezi časovače a pro jeho použití je nutné definovat časový interval „PT“. Jako reakci na náběžnou hranu vstupní booleovské proměnné FB TP vrací hodnotu 1 výstupní proměnné, a to právě po definovaný čas „PT“. Průběh hodnot v čase je znázorněn na Obr. 15.: Graf hodnot FB TP. Proměnná „IN“ je vstupní a „Q“ výstupní.



Obr. 15.: Graf hodnot FB TP [6]

Poslední předdefinovaný FB, který byl při psaní řídícího programu použit je také časovač a sice **TON** (Obr. 16.: Graf hodnot FB TON). Ke svému chodu také potřebuje přesně definovaný časový úsek „PT“, ten má však oproti TP jiný význam. Jako reakci na náběžnou hranu vstupní booleovské proměnné se spustí časovač a pokud i po uplynutí „PT“ je stále hodnota vstupu 1, TON začne na výstupu vracet hodnotu 1. Tento stav trvá, dokud je na vstupu hodnota 1.



Obr. 16.: Graf hodnot FB TON [6]

Jeden z mnoha případů využití časovačů je naprogramování uchopení obrobku podtlakovým dopravníkem. Při zahájení operace se spustí časovač TP, který 2 sekundy generuje hodnotu 1 na výstupu. Hodnota výstupu funkčního bloku je přiřazena digitálnímu výstupu PLC, který slouží k svislému pohybu ramene úchopu (ten zajišťuje kontakt ústí podtlakové trubice a obrobku). Výstup FB TP, je ale také použit jako vstupní proměnná FB TON, který půl sekundy po vykonání svislého pohybu zařízení spustí podtlak (podtlak trvá až do jeho zrušení při odkládání obrobku z dopravníku).

Tvorba vlastních FB určených pro specifické použití je velmi výhodná. Na příklad při opakovaném používání jednoho zařízení pro tentýž účel. Situaci lze ilustrovat na provozu pásového dopravníku, který se nachází na konci trati. Jeho jediným úkolem je po obdržení obrobku spustit pohon a po přerušení světelné závory na konci dopravníku jej vypnout. Tento úkon vykonává v rámci očisty trati i během běžného provozu. V takový okamžik je vhodné deklarovat FB, který by sloužil těmto účelům, a jehož instance by bylo možné volat ve více kapitolách řídícího programu. Detailní rozbor jednoho z FB je uveden v kapitole 2.5.

Deklarace nového funkčního bloku začíná definicí proměnných. Vedle vstupních a výstupních proměnných jsou přítomny i statické proměnné. Ty jsou přístupné pouze

pro daný funkční blok. Funkční blok je programová organizační jednotka, která disponuje vlastní pamětí. Z toho plyne, že dochází k přenosu hodnot proměnných mezi jednotlivými voláními. Vedle definice proměnných každý funkční blok obsahuje také výkonnou část, tedy zdrojový kód, který je vykonán po zavolání funkčního bloku. Je možné, aby jeden funkční blok volal druhý.

2.4.3. Zápis do digitálních výstupů PLC

Nutno podotknout, že zápis do výstupů PLC z pohledu zdrojového kódu neprobíhá přímo ve funkčních blocích, ani jejich volání. Není tomu tak ani uvnitř jednotlivých kapitol, do kterých je řídící program rozdělen (popsáno v kapitole 2.4.1). Důvodem je vykonávání programu od prvního do posledního řádku zdrojového kódu. Pokud by tedy v rámci jedné kapitoly byla do digitálního výstupu zapisována určitá hodnota a současně by tomu tak bylo i v některé z následujících, došlo by k přepisování výsledků práce jednotlivých oddílů programu.

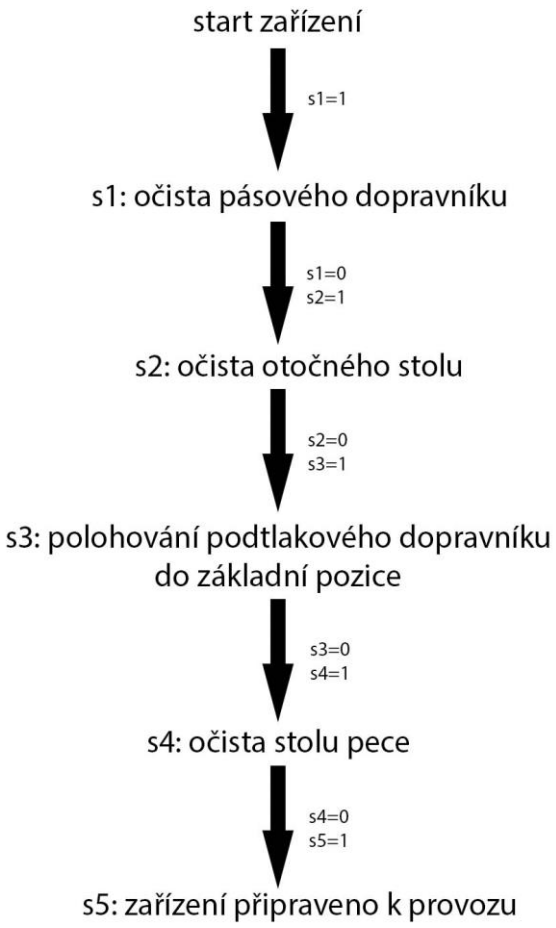
Řešení spočívá v použití výstupů funkčních bloků jako pomocných proměnných, které se na závěr programu přiřadí příslušným výstupům PLC, a to za pomoci booleovského součtu „OR“. Do digitálních výstupů PLC je tedy zapisováno pouze jednou. Jedná se o funkční a přehledné řešení popsané problematiky.

2.4.4. Očista trati

Pro zahájení běžného provozu je nutné po zapnutí zařízení provést tak zvanou očistu trati. Ta slouží k uvedení zařízení do základní polohy a v případě, že předcházející vypnutí neproběhlo standardně, i k očištění trati od obrobků, které neprošly celým pracovištěm.

Jak již několikrát zaznělo, celý program je rozdělen do několika (8) kapitol. Očista se skládá z kapitoly s1 až s4 a po jejím dokončení je možný běžný provoz pracoviště. Při něm se program pohybuje mezi pozicemi s5 až s8.

Poté co obsluha pomocí HMI panelu manuálně zahájí očistu, spustí se kapitola **s1**. Ta slouží k ujištění se, že se na pásovém dopravníku nenachází obrobek. Uvnitř kapitoly je volán „FB_pas“, který je určen k provozu pásového dopravníku. Spustí pohon a v případě přítomnosti obrobku pás zastaví přerušení světelné závory. Funkční blok se prohlásí za hotový až po odebrání dílce, a tedy obnovení světelného spojení. Pokud se na páse nic nenachází, 3 sekundy koná pohyb na prázdno, následně zastaví a prohlásí FB za hotový. R-TRIG výstupní proměnné FB, která indikuje hotový děj, slouží jako přechodová podmínka „IF“, jež ukončí kapitolu s1 a naopak spustí s2.



Obr. 17.: Sled kapitol věnovaných očistě trati

Druhá kapitola (**s2**) má úlohu očistit otočný stůl. K tomuto účelu byl napsán funkční blok „FB_ocista_ot_stul“. Ten polohuje otočný stůl na pozici pás a následně aktivuje píst sloužící k přemístění obrobku z otočného stolu na pás. „FB_ocista_ot_stul“ dále volá „FB_pas“. Po jeho skončení se otočný stůl polohuje na základní pozici (pozice úchop, která umožňuje odkládání obrobku na otočný stůl). Náběžná hrana koncového snímače otočeného stolu pozice úchop slouží jako uzavření kapitoly s2 a spoušť kapitoly s3.

Kapitola **s3** má za úkol připravit podtlakový dopravník na provoz. Vzhledem k tomu, že očista probíhá buď po startu zařízení nebo po zrušení chyby a případném odložení obrobku, není možné, aby podtlakový dopravník držel obrobek při chodu očisty. Případ kdy během dopravy byla vyhlášena chyba, a tedy přerušen provoz je ošetřen odložením obrobku po skončení chyby a následným zahájením očisty od kapitoly s1. Během kapitoly s3 se tedy pouze polohuje dopravník na pozici pec a po sepnutí koncového snímače se pokračuje na kapitolu s4.

Smysl kapitoly **s4** je příprava pece na vložení nového obrobku, tedy na její provoz. „FB_vyjeti_stolu_pece“ otevře vrata, polohuje stůl pece na koncovou pozici vně pece a následně vrata zavře. Po vyjetí stolu z pece se pomocí světelné závory určí, zda je potřeba odvoz obrobku. Slouží k tomu „FB_odvoz“ a po odložení obrobku na otočný stůl je opět volán „FB_ocista_ot_stul“. V případě že světelná závora nebyla přerušena nebo byl přítomný obrobek odvezen a trať očištěna se ukončí kapitola s4 a zahájí s5. Tím je očista trati dokončena a zařízení je schopno produktivní práce.

Start zařízení a proběhnutí očisty trati je možné shlédnout na přiloženém videu. (Příloha C)

2.4.5. Běžný provoz

Na rozdíl od očisty trati, volání kapitol s5 až s8, tedy kapitol věnujících se produktivní činnosti, se zásadně liší. Program je přizpůsoben vložení dalšího obrobku ihned po uvolnění stolu pece obrobkem předešlým. Ve zdrojovém kódu se takový scénář projeví větším počtem aktivních kapitol současně.

Po ukončení očisty se aktivuje kapitola **s5**. Ta slouží k čekání na vložení obrobku na stůl pece. Podmínka „IF“ se v každém taktu ptá, zda obsluha potvrdila vložení obrobku (zapsala do pomocné proměnné „g_02_man_vlozen_obrobek“ hodnotu 1) a zároveň byla přerušena světelná závora. Pokud je tato podmínka splněna, kapitola s5 se sama deaktivuje, a naopak aktivuje s6.

S6 je určena automatizaci práce žíhací pece. Uvnitř kapitoly jsou volány dva FB. První, „FB_zajeti_pece“ otevře vrata, polohuje stůl na pozici uvnitř pece a následně vrata zavře. Po dobu 5 sekund probíhá symbolické vytápění komory pece (rozsvítí se světlo). Tím je funkční blok dokončen a R-TRIG jeho výstupní proměnné indikující dokončení děje slouží jako spoušť druhého FB, tedy „FB_vyjeti_stolu_pece“. Tento FB byl představen již

v předešlé kapitole věnující se očistě a jeho použití i v běžném provozu usnadňuje práci programátora. V případě většího počtu obrobků na trati je zásadní, aby v případě potřeby program čekal s pokračováním na další kapitoly. Tento problém zajišťuje pomocná proměnná „s6_1“, která se aktivuje po dokončení práce pece, úchopu i otočného stolu. Než se pokračuje na s7 (než se volá odvoz na otočný stůl), je nutné, aby bylo kam odkládat, tedy byla dokončena práce otočného stolu na předchozím obrobku. Podmínkou „IF“ je tento vztah zajištěn.

V případě, že skončily operace pece a otočného stolu, jedinou operací, která zbývá dokončit (před spuštěním dalšího odvozu) je návrat úchopu na pozici pec. V takové situaci je s6_1 rovna hodnotě 0. Hodnota 1 je do s6_1 zapsána ve chvíli, kdy je dokončen i pohyb podtlakového dopravníku. Bohužel ke splnění podmínky "IF "FB_vyjeti_stolu_pece_s6".hotovo AND NOT "s7" AND NOT "s8" AND "FB_zajeti_pece_s6".hotovo THEN" (a tedy znovu zavolání s7) dojde ve stejném taktu jako ukončení kapitoly s7 (dokončení pohybu podtlakového úchopu). Tato skutečnost znamená, že se nespustí odvoz obrobku, který čeká na stolu pece. Situace je ošetřena časovačem TP, který na náběžnou hranu proměnné s6_1 reaguje 0,2 sekundy trvajícím impulsem. Po skončení tohoto impulsu (reakce na sestupnou hranu) proběhne zápis do proměnné s7, a to už zaručené v jiném taktu, než při původní situaci.

Poté co obrobek opustí pec je potřeba zajistit odvoz podtlakovým dopravníkem na otočný stůl. Za tímto účelem vznikla kapitola **s7**. Jak již bylo řečeno v předchozím odstavci, tento úkon odstartuje až ve chvíli, kdy otočný stůl je připraven obrobek přijmout. „FB_odvoz“ pomocí časovače TP obstará dvě sekundy trvající dolní polohu ramene úchopu, naopak pomocí časovače TON půl sekundy po kontaktu těsnění úchopu a obrobku se spustí podtlak. Následuje polohování celého dopravníku do pozice „otočný stůl“ a odložené obrobku, které principiálně probíhá stejně jako uchopení. Na závěr se zařízení polohuje do základní pozice „pec“. K zvýšení produktivity je funkční blok opatřen výstupními proměnnými, které ohlašují uvolnění stolu pece, odložení na otočný stůl a ukončení práce dojetím na pozici „pec“. Řídící program tedy nečeká na dokončení kompletní kapitol s7, ale volá příslušné kapitoly již v průběhu odvozu.

Poslední kapitola, **s8** slouží k řízení provozu otočného stolu. „FB_provoz_ot_stul“ nejprve otočí stůl s obrobkem do pozice frézka, kde dochází k (symbolickému) třískovému opracování obrobku. Následuje pohyb na pozici „pás“, kde se pístem obrobek přemístí

na pásový dopravník a následně se uvnitř FB volá „FB_pas“. Po odebrání obrobku se „FB_pas“ prohlásí za hotový a otočný stůl se vrací do pozice úchop, čímž indikuje možnost přijmout další obrobek. Po skončení funkčního bloku už nedochází k volání další kapitoly, pouze se zruší aktivita kapitola s8.

Běžný provoz zařízení je možné shlédnout na přiloženém videu. (Příloha C)

2.4.6. Ošetření chybového stavu

Pokud nastane chybový stav zařízení, je nutné, aby zařízení přestalo konat pohyb (výjimkou je otevření vrat pece), drželo podtlak v případě potřeby (pokud je aktivní v okamžiku vyhlášení chyby) a zastavilo vytápění pece (pokud je spuštěno). Tento postup je realizován podmínkou „IF“ při zápisu hodnot digitálních výstupů.

Mezi uvažované chybové stavy patří: nouzové zastavení provozu pomocí tlačítka CENTRAL STOP, nedojetí obrobku na konec trati a selhání koncového snímače. CENTRAL STOP je ošetřen sledováním příslušného digitálního vstupu. O ztrátě obrobku informuje funkční blok spravující pásový dopravník, který tento stav identifikuje nepřerušením světelné závory. Selhání koncového snímače řídící program rozpozná, pokud některý z pohybů trvá déle než předem definovaný časový úsek, odpovídající nejdelšímu úkonu. O druhu chyby je obsluha informována na obrazovce panelu HMI.

Poté co obsluha na panelu HMI chybový stav manuálně zruší, program uvede zařízení do bezpečného stavu, ze kterého následně zahájí očistu. Je nutné zajistit odložení obrobku z podtlakového dopravníku a polohování stolu pece na některou z koncových poloh, aby při vypnutí nedošlo ke kolizi s vraty pece. (Vrata pece ovládá píst, který je poháněn kompresorem. Při vypnutí zařízení se tedy vrata zavřou.) Poté co jsou dokončeny uvedené úkony je zahájena očista, a to od kapitoly s1. Po jejím skončení je pracoviště připraveno k dalšímu použití.

Řešení chybového stavu zařízení je možné shlédnout na přiloženém videu. (Příloha C)

2.5. Detailní rozbor funkčního bloku

Zdrojový kód celého programu je velmi obsáhlý. Z toho důvodu byl v předchozích kapitolách popsán pouze principiálně. Tato kapitola je věnována podrobnému představení jednoho z uživatelských funkčních bloků a to „FB_pas“, který zodpovídá za provozování pásového dopravníku. Pásový dopravník slouží jednomu prostému úkonu, a to dopravení

obrobku na konec trati. Z toho důvodu je „FB_pas“ použit jak v očistě trati, tak v běžném provozu.

V první řadě je nutné jmenovat proměnné, se kterými zdrojový kód funkčního bloku pracuje. Kompletní výpis deklarační části „FB_pas“ je vidět na Obr. 18.: Deklarační část „FB_pas“.

fb_pas

		Name	Data type	Default value	Retain
1		Input			
2		IN	Bool	false	Non-ret...
3		di_zavora_pas	Bool	false	Non-retain
4		F_TRIG_di_zavora_pas	Bool	false	Non-retain
5		R_TRIG_g_chyba	Bool	false	Non-retain
6		Output			
7		do_pas	Bool	false	Non-retain
8		neprojeti_obrobku	Bool	false	Non-retain
9		hotovo	Bool	false	Non-retain
10		InOut			
11		<Add new>			
12		Static			
13		nastala_chyba	Bool	false	Non-retain
14		potreba_odber	Bool	false	Non-retain
15		pom_pas	Bool	false	Non-retain
16		s1	Bool	false	Non-retain
17		s2	Bool	false	Non-retain
18		TON_pas	TON_TIME		Non-retain
19		R_TRIG_TON	R_TRIG		

Obr. 18.: Deklarační část „FB_pas“

Vstupní proměnná **„IN“** slouží při volání funkčního bloku k zahájení práce. Při volání v řídícím programu je nutné do proměnné „IN“ dosadit výstup z funkčního bloku typu „TRIG“, tedy proměnnou, která nabývá kladné hodnoty pouze po dobu jednoho taktu. V opačném případě by k zahájení práce „FB_pas“ docházelo opakovaně, a to není žádoucí. Další vstupní proměnnou je „di_zavora_pas“, do které je při volání dosazen příslušný digitální vstup. (Místo této proměnné by bylo možné pracovat přímo s globální proměnnou digitálního výstupu, pro přehlednost je však veškerá komunikace funkčního bloku řešena v deklarační části.) Další vstupní proměnnou je „F_TRIG_di_zavora_pas“, do které je při volání dosazen výstup funkčního bloku, který sleduje sestupnou hranu světelné závory na konci pásu. (Funkční blok typu TRIG je vhodné situovat přímo ve zdrojovém kódu řídícího programu a nepodmiňovat jeho volání.) Poslední vstupní proměnnou je „R_TRIG_g_chyba“, která slouží ke zrušení veškerého průběhu práce funkčního bloku v případě, že nastala chyba.

Mezi výstupy „FB_pas“ patří následující proměnné. „do_pas“ je výstupní hodnota, která je určena k dosazení do proměnné digitálního výstupu. Jedná se o pohyb pásu. „neprojeti_obrobku“ indikuje chybový stav zařízení, kdy obrobek nedorazí na konec trati. Výstupní proměnná „hotovo“ funguje jako oznámení dokončení všech úkonů.

Statické proměnné jsou takové, které jsou přístupné pouze uvnitř funkčního bloku. Patří sem i hodnoty funkčních bloků, které jsou volány uvnitř „FB_pas“ a to způsobem „Multi instance“. To znamená, že datový blok funkčního bloku volaného je uložen v rámci statických proměnných funkčního bloku, ve kterém je volán. To platí pro „TON_pas“ a „R_TRIG_TON“. Pomocná proměnná „pom_pas“ slouží k zápisu do výstupu funkčního bloku, a to až na závěr zdrojového kódu. Proměnná „nastala_chyba“ indikuje stav, kdy v zařízení byl vyhlášen chybový stav. Proměnná „potreba_odber“ jak už název napovídá slouží k rozpoznání stavu, kdy je obrobek na konci trati a připraven k odběru. Proměnné „s1“ a „s2“ slouží k strukturování zdrojového kódu totožným způsobem, jako byl popsán v kapitole 2.4.1.

Následuje rozbor výkonné části funkčního bloku.

```
//zahajeni provozu
IF #IN THEN
   #s1 := 1;
   #s2 := 0;
   #nastala_chyba := 0;
   #hotovo := 0;
   RESET_TIMER(#TON_pas);
END_IF;
```

V případě, že došlo k zavolání „FB_pas“ a vstupní proměnná „IN“ má hodnotu jedna, je splněna podmínka „IF“. V tu chvíli je vykonán zdrojový kód uvnitř příkazu „IF“ a tedy: spustí se kapitola s1, následně je do s2 a „nastala_chyba“ zapsána 0 a dále i do výstupní proměnné „hotovo“ je zapsána hodnota 0. V případě, že funkční blok v minulosti již proběhl je v proměnné „hotovo“ hodnota 1 až do příštího volání. V řídícím programu je tedy nutné s touto výstupní proměnnou pracovat pomocí funkčního bloku R-TRIG, který reaguje na náběžnou hranu. Při splnění podmínky „IF“ je také restartován časovač TON.

```
//pohyb pasu
#TON_pas(IN:=#do_pas,
      PT:=t#3s);

#R_TRIG_TON(CLK:=#TON_pas.Q);

IF #s1 THEN
   #pom_pas := 1;

   IF #R_TRIG_TON.Q OR #F_TRIG_di_zavora_pas THEN
      #pom_pas := 0;
      #s1 := 0;
      #s2 := 1;
   END_IF;

   IF #R_TRIG_TON.Q THEN
```

```
    #neprojeti_obrobku := 1;
  END_IF;

END_IF;
```

V kapitole s1 je spuštěn pohyb pásu a současně spuštěn odpočet časovače TON, který po 3 sekundách trvajícím pohybu začne generovat hodnotu jedna. Pohyb je zastaven dvěma způsoby. Při korektním chodu je obrobkem přerušena světelná závora na konci pásu. F-TRIG příslušného digitálního vstupu jako reakci na sestupnou hranu generuje jeden takt trvající impuls. Ten je použit v podmínce „IF“ pro vypnutí pohonu a přechod na kapitolu s2. V případě nepřerušení světelné závory k zastavení slouží časovač. Pomocí funkčního bloku R-TRIG je sledována výstupní proměnná časovače TON. Pokud výstup R-TRIG nabyde hodnotu jedna (pokud během definovaného časového úseku nebyl zastaven pohyb přerušením světelné závory), dojde k samostatnému zastavení a zapsání hodnoty jedna do proměnné „neprojeti_obrobku“. I v takovém případě dojde k přechodu na kapitolu s2.

```
//zadani odberu obrobku a nasledne ohlaseni dokonceni chodu FB
IF #s2 THEN
  IF NOT #di_zavora_pas THEN
    #potreba_odber := 1;
  ELSE
    #s1 := 0;
    #s2 := 0;
    #hotovo := 1;
  END_IF;

END_IF;
```

V kapitole s2 se čeká na obnovení světelné závory. Při jejím přerušení má digitální vstup hodnotu nula. Program tedy při nulové hodnotě digitálního vstupu v každém taktu zapíše do proměnné „potreba_odber“ hodnotu jedna. V opačném případě („ELSE“) dojde k zápisu hodnoty jedna do proměnné „hotovo“ a ukončení kapitoly s2. Pokud nedošlo k přerušení závory je funkční blok ukončen bez čekání. Je pak na řídícím programu, jak naloží s informací, že neprojel obrobek. Během očisty trati je tato informace ignorována (neví se, jestli se obrobek očekává). Naopak za běžného provozu tento stav znamená vyhlášení chyby.

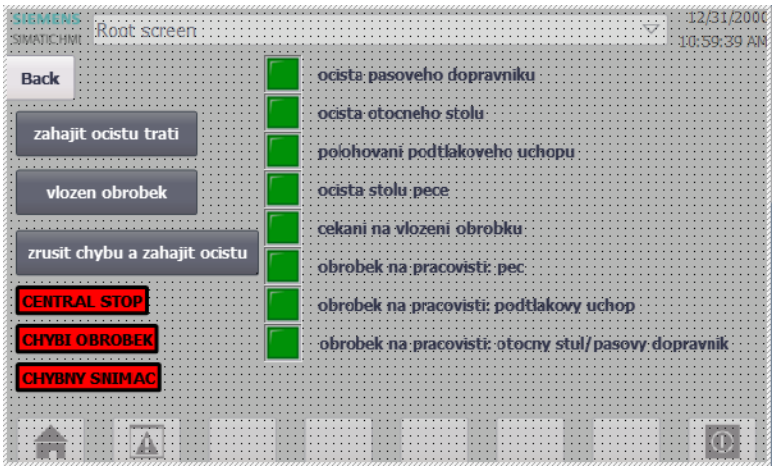
```
//vyhlaseni chyby
IF #R_TRIG_g_chyba THEN
  #nastala_chyba := 1;
END_IF;

//zapis do vystupu FB
IF NOT #nastala_chyba THEN
```

```
   #do_pas := #pom_pas;
ELSE
   #s1 := 0;
   #s2 := 0;
   #do_pas := 0;
   #pom_pas := 0;
END_IF;
```

Na závěr výkonné části se nachází dvě podmínky „IF“. První slouží k zapsání hodnoty 1 do proměnné „nastala_chyba“ jako reakci na náběžnou hranu globální proměnné „g_02_man_chyba“. Druhá podmínka „IF“ slouží k zápisu do výstupních proměnných. V Případě, že nenastala chyba se přiřadí příslušná pomocná proměnná do výstupu funkčního bloku. V opačném případě se do kapitol s1 a s2 zapíše 0 a dále se zapíše 0 i do výstupní a pomocné proměnné.

2.6. HMI panel

HMI (human-machine interface) je rozhraní určené pro komunikaci mezi PLC a obsluhou pracoviště. V případě školní úlohy se jedná o dotykovou obrazovku od firmy Siemens, konkrétně pak model KTP700 Basic PN. Jejím úkolem je umožnit zapisovat do manuálně ovládaných proměnných, zobrazovat stav zařízení a varovat o chybovém stavu sestavy stroje.



Obr. 19.: Ovládací obrazovka

K zápisu hodnot do proměnných řídícího programu, které jsou určeny k manuálnímu ovládání, postačují tři tlačítka v levém horním rohu obrazovky. Jejich viditelnost je podmíněna určitým stavem zařízení. Po startu je viditelné pouze tlačítko **„zahajit ocistu trati“**. Po skončení očisty řídící program setrvává v kapitole s5, určené k čekání na vložení obrobku. Tímto stavem je podmíněna viditelnost tlačítka **„vlozen obrobek“**, kterým obsluha stvrzuje přítomnost obrobku na začátku trati. Posledním tlačítkem je **„zrusit chybu**

a zahajit ocistu“, kterým se při chybovém stavu zařízení prohlásí chyba za odstraněnou (zařízení následně odloží obrobek (v případě držení obrobku podtlakovým dopravníkem), polohuje stůl pece na pozici vně pece a zahájí očistu trati.

V pravé části obrazovky se nachází indikace stavu zařízení. Jednotlivé stavy odpovídají kapitolám s1 až s8 a o jejich aktivitě informuje zelená světelná signalizace.

Velmi kontrastní prvkem ovládací obrazovky jsou červená hlášení o chybovém stavu. V levé dolní části obrazovky jsou umístěny tři oznámení, které odpovídají jednotlivým druhům poruch. Viditelnost těchto hlášení je podmíněna příslušným stavem zařízení.

3. Řešení sekundárního programu

Sekundární program je řešen stejnými prostředky jako program řídící. V aplikaci TIA Portal je vytvořen v rámci shodného projektu a je psán stejným programovacím jazykem, tedy strukturovaným textem.

3.1.1. Požadavky na sekundární program

Jak již bylo popsáno v úvodních kapitolách této práce, od sekundárního programu se očekává poskytnutí zpětné vazby řídícímu programu. Jedná se o digitální vstupy PLC, které při standardním zprovoznění generují snímače sestavy stroje. Poskytují informace o poloze jednotlivých zařízení (koncová čidla), ale také o přítomnosti obrobku v daném úseku trati (světelné závory). Nutno zdůraznit skutečnost, že veškeré vstupy PLC jsou proměnné booleovského typu, tedy nabývají pouze hodnot jedna a nula.

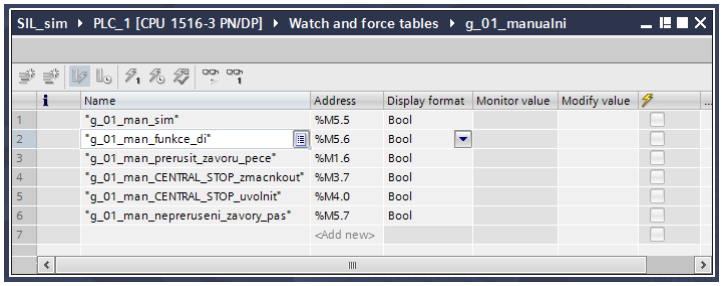
Je nutné, aby se simulovaný model svým chováním co nejvíc přiblížil skutečnému. Sekundární program se proto snaží sledovat vztahy mezi jednotlivými aktory a snímači, a to jak logické, tak časové.

Při tvorbě simulačního programu byl také kladen důraz na jeho univerzálnost. Programování je činnost velmi tvořivá a simulace musí být schopna reagovat odpovídajícím způsobem na rozdílné řídící programy vytvořené studenty.

3.1.2. Požadavky na obsluhu sekundárního programu

Od uživatele simulace se očekává ovládání prostředí TIA Portal na základní úrovni. Sledování práce programu řídícího i sekundárního probíhá v tak zvaných „watch tables“,

kde jsou monitorovány hodnoty proměnných. Během simulace je také nutné ovládat proměnné řídícího i sekundárního programu, které jsou určeny k manuální obsluze.



Obr. 20.: Manuálně ovládané proměnné

Obr. 20.: Manuálně ovládané proměnné znázorňuje „watch table“, obsahující veškeré proměnné sekundárního programu, které jsou ovládány uživatelem. Ten pomocí příkazu „modify“ mění jejich hodnoty. Detailní návod obsluhy sekundárního programu je uveden v přiloženém uživatelském návodu (Příloha A).

Proměnná **„g_01_man_sim“** slouží k zahájení chodu programu „prg_01_simulace“ a je tedy nutné jí při každém zahájení simulace manuálně modifikovat na hodnotu 1. Po spuštění PLC má hodnotu 0, čímž umožňuje celý projekt (včetně simulace) nahrát i do reálného PLC. („prg_01_simulace“ po nahrání projektu do reálného PLC nesmí být aktivní. Jeho práce by byla v konfliktu se snímači zařízení. Proto je spuštění simulace řízeno manuálně.)

Proměnná **„g_01_man_funkce_di“** slouží k zapnutí a vypnutí funkčnosti veškerých digitálních vstupů PLC. Po modifikace „g_01_man_sim“ na hodnotu jedna se upraví i hodnota „g_01_man_funkce_di“. Do jednotlivých proměnných se při hodnotě nula přestane zapisovat a trvale tak nabydu nulovou hodnotu. Tím je simulován chybový stav zařízení, kdy přestane fungovat zpětná vazba systému. Je tedy možné testovat bezpečnost řídícího programu. (Žádoucí je stav, kdy řídící program sám rozpozná situaci a zastaví pohyb.)

Proměnná **„g_01_man_prerusit_zavoru_pece“** slouží k pomyslnému vložení obrobku na stůl pece (začátek trati). Závora se sama automaticky obnoví poté, co podtlakový dopravník uchopí obrobek (simulace reaguje na sestupnou hranu „do_chnap„ a zároveň držený podtlak).

Proměnné **„g_01_CENTRAL_STOP_zmacknout“** a **„g_01_CENTRAL_STOP_uvolnit“** slouží k obsluze červeného tlačítka CENTRAL STOP, kterým je každé PLC v laboratoři vybaveno.

Proměnná **„g_01_nepreruseni_zavory_pas“** slouží k simulaci chybového stavu, kdy obrobek nedorazí na konec trati a tím je možné testovat bezpečnost řídícího programu. (Žádoucí je stav, kdy řídící program sám rozpozná situaci a zastaví pohyb.)

S fyzickým zařízením je velmi vhodné být předem seznámen. To následně dovolí určitý nadhled nad digitálními vstupy a jejich účelem.

3.1.3. Struktura

V každém okamžiku je potřeba zajistit, aby simulované hodnoty všech proměnných korespondovaly s chování skutečného zařízení. Sekundární program proto na rozdíl od řídícího už není rozdělen do jednotlivých kapitol, ale je aktivní celý zdrojový kód. Jedinou podmínkou je pak hodnota jedna proměnné „go_01_man_sim“. Tato podmínka dovoluje vypnout simulaci při zprovoznění projektu na skutečném PLC s připojenou sestavou stroje.

Zdrojový kód je psaný v odstavcích, kde každý odstavec odpovídá konkrétnímu digitálnímu vstupu. Úzce svázány pak jsou snímače shodného zařízení (na příklad pár koncových snímačů stolu pece).

Na závěr programu probíhá zápis do digitálních vstupů PLC popsaný v následující kapitole.

3.1.3.1. Pomocné proměnné pro zápis do digitálních vstupů PLC

Jak již bylo uvedeno v kapitole 1.4.1, PLC pracuje v taktech. Na začátku každého taktu probíhá načítání vstupů PLC. Je tedy zřejmé, že hodnoty vstupů nejsou určeny k přenosu mezi jednotlivými takty. Tento fakt je při simulaci nutný ošetřit, a to pomocnými proměnnými.

Každému vstupu PLC odpovídá jedna pomocná proměnná, která disponuje pamětí PLC. Výstupy sekundárního programu jsou pak ukládány do této pomocné proměnné a na závěr sekundárního programu jsou hodnoty pomocných proměnných dosazeny do proměnných vstupů.

3.1.3.2. Koncové snímače

Po spuštění simulace je uvažován neurčitý stav zařízení. Žádný snímač určující polohu nemá hodnotu jedna. Princip simulace pak spočívá v reakci na daný pohyb zařízení.

Pro názornost v následujícím odstavci budou popsány vztahy mezi pomyslným pohybem stolu pece a reakcí sekundárního programu.

Jako reakce na náběžnou hranu proměnné pohybu stolu pece jedním směrem je do proměnné odpovídající koncovému snímači opačného směru zapsána hodnota nula. (Při obecné poloze zařízení tento úkon nemá význam.) Zároveň je spuštěn časovač TP, který po dobu tří sekund generuje na výstupu hodnotu jedna. Hodnota výstupu časovače slouží jako vstup pro funkční blok „F-TRIG“, který reaguje na sestupnou hranu. Pokud „F-TRIG“ nabyde hodnotu jedna (pokud nastane sestupná hrana výstupu časovače), zapíše se do koncového snímače nacházejícího se ve směru pohybu hodnota jedna. Tyto akce jsou podmíněny trvajícím pohybem zařízení.

Jinak řečeno: Pokud nastane pohyb, koncové čidlo opačného směru pohybu nabyde hodnotu nula. Pokud pohyb trvá alespoň tři sekundy, koncové čidlo ve směru pohybu nabude hodnotu jedna. Tato hodnota je uložena v pomocné proměnné daného digitálního vstupu a zruší jí až pohyb opačným směrem.

V případě otočného stolu se situace komplikuje přítomností třetího snímače polohy, který je umístěn přibližně v polovině rozsahu otáčení. Jedná se o snímač indikující polohu otočného stolu na pracovišti frézka. Ke správnému chodu simulace jsou potřeba tři pomocné proměnné. První „g_01_pom_o2_1“ slouží k určení stavu, kdy otočný stůl dosáhl po třech sekundách trvajícím pohybu polohy frézka. (Tato pomocná proměnná je společná pro oba směry pohybu.) V případě zastavení otočného stolu na pozici frézka zůstává simulována tato poloha. Při následujícím zahájení pohybu (nebo při nezastavení na poloze frézka) dochází ke zrušení této polohy, a po dalších třech sekundách se do příslušného koncového snímače zapíše hodnota jedna. To umožňují proměnné „g_o2_pom_smer_pas_2“ a „g_o2_pom_smer_uchop_2“, jejichž hodnota jedna znamená stav, kdy už poloha frézka proběhla a otočný stůl tak směřuje na koncový snímač.

3.1.3.3. Světelné závory

Pracoviště je vybaveno dvěma světelnými závorami. První se nachází na samém začátku trati a snímá přítomnost obrobku na stolu pece, pokud je na poloze vně pece. Tuto závoru tedy v reálném provozu přeruší obsluha vložením palety s obrobkem. Při simulaci její přerušení obstarává manuálně ovládaná proměnná

„g_01_man_prerusit_zavoru_pece“. Obnovení světelného toku už je automatické a nastane při pohybu hlavice podtlakového úchopu směrem nahoru a současném držení podtlaku.

Druhá světelná závora se nachází na konci trati, a tedy na pásovém dopravníku. Přerušení světelné závory v reálném provozu je zapříčiněno přítomností obrobku na pohybujícím se páse. Při simulaci není možnost rozlišit mezi přítomností a nepřítomností obrobku na páse. Samotný pohyb pásu ve skutečnosti ještě neznamená přerušení závory, při běžném stavu simulace je ale tento vztah předpokládán. Uživatel simulace ale má k dispozici manuálně ovládanou proměnnou „g_01_man_nepreruseni_zavory_pas“, kterou přerušení světelného toku zamezí. To slouží k simulaci chybového stavu zařízení, kdy obrobek nedorazí na konec trati. Tomuto tématu je věnována následující kapitola.

3.1.3.4. Chybové stavy zařízení

První ze tří možných chybových stavů je zmáčknutí nouzového tlačítka CENTRAL STOP. Jeho pomyslné zmáčknutí a uvolnění ovládá uživatel simulace pomocí proměnných „g_01_CENTRAL_STOP_zmacknout“ a „g_01_CENTRAL_STOP_uvolnit“.

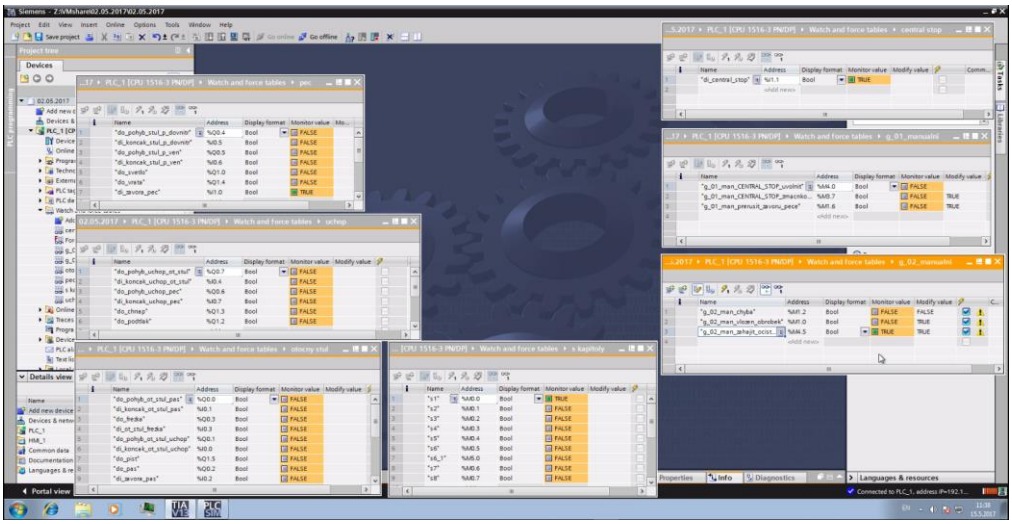
Druhý chybový stav, který je možné při simulaci vyvolat je neprojetí obrobku světelnou závorou na konci trati, jehož princip byl vysvětlen v předešlé kapitole. Tento případ v běžném provozu znamená ztrátu nebo uvíznutí obrobku během práce některého ze zařízení pracoviště.

Poslední chybový stav je selhání zpětné vazby pracoviště. Je spuštěn modifikací proměnné „g_01_man_funkce_di“ na hodnotu nula. Tento stav brání dosazení hodnot příslušných pomocných proměnných do digitálních vstupů a ty tak nabydou hodnotu nula.

3.1.4. Zkušenosti autora práce s vlastním simulačním programem

Sekundární (simulační) program byl poprvé použit už při testování řídícího programu popsaného v této závěrečné práci. Chod řídícího programu při zprovoznění za pomoci simulace odpovídal provozu na skutečném zařízení. Drobná komplikace však představuje velké množství proměnných, které je v každém okamžiku potřeba sledovat. (Obr. 21.: Watch tables) Tento problém byl vyřešen pomocí aplikace OBS Studio, která slouží k nahrávání obrazovky počítače a následný export záznamu v některém z běžných

video formátů. Tímto způsobem bylo možné zpětně analyzovat proběhlé simulace. Velkým přínosem je možnost pozastavení záznamu. Alternativou OBS Studia je funkce „traces“, kterou je TIA Portal vybaven. Slouží k logování průběhů signálů přímo na PLC.



Obr. 21.: Watch tables

4. Závěr

Cílem práce je navrhnout a zrealizovat vhodné řešení virtuálního SIL modelu výukové sestavy stroje. Tento model by měl umožňovat ladění řídícího PLC programu, a to bez fyzického připojení k výukové sestavě stroje. Virtuální model, kterým se zabývá tato závěrečná práce, je určen jako pomůcka při výuce programování PLC. Stěžejním přínosem je možnost testovat řídící program i mimo prostory laboratoře školy.

Rešeršní část je věnována popisu dostupných řešení, ze kterých bylo následně jedno vybráno a zpracováno, a to s ohledem na danou aplikaci. Konkrétně pak řešení problematiky pomocí simulačního PLC programu, který běží na pozadí programu řídícího. Byla vybrána jedna ze čtyř výukových sestav, kterými je laboratoř vybavena. Na té následně proběhla realizace zadaných cílů práce. Byl realizován virtuální model, který umožňuje působení aktorů a poskytování zpětné vazby ze snímačů do řídícího PLC programu. Simulační program tímto způsobem umožňuje ladění řídícího programu, a to bez připojení k výukové sestavě stroje. Pro chod takového modelu je nezbytné vytvořit i samotný řídící program, kterým se tato závěrečná práce také zabývá. Vypracovaný program byl opatřen komentáři a uživatelským návodem, který je přílohou této závěrečné práce.

Předložená závěrečná práce splňuje zadání a byly dosaženy veškeré stanovené cíle.

5. Seznamy

5.1. Citovaná literatura

[1.] Šmejkal, Ladislav. Časopis Automa Esperanto programátorů PLC: programování podle normy IEC/EN 61131-3 (část 1). *automa.cz.* [Online] [Citace: 2. 5 2017.] http://automa.cz/cz/casopis-clanky/esperanto-programatoru-plc-programovani-podle-normy-iec/en-61131-3-cast-1-2011_08_44606_5601/.

[2.] Mosaic - Programování dle IEC-61131-3. *www.tecomat.com.* [Online] 11 2007. [Citace: 11. 5 2017.] http://www.tecomat.com/wpimages/other/DOCS/cze/TXV00321_01_Mosaic_ProgIEC_cz.pdf.

[3.] Berger, Hans. *Automating with SIMATIC S7-1500: Configuring, Programming and Testing with STEP 7 Professional.* s.l. : Publicis, 2014. 978-3895784040.

[4.] 536632 - Multi Processing Station with oven 24V - HELAGO-CZ, s.r.o. *helago-cz.cz.* [Online] [Citace: 8. 5 2017.] https://www.helago-cz.cz/eshop-536632-multi-processing-station-with-oven-24v.html.

[5.] *Nápověda v programu TECO Mosaic v.2017.2.* 2017.

[6.] *Nápověda v programu SIEMENS TIA Portal V13.* 2017.

5.2. Seznam obrázků

5.3. Seznam tabulek

5.4. Seznam příloh

textové

Příloha A	Uživatelský návod simulačního programu

elektronické

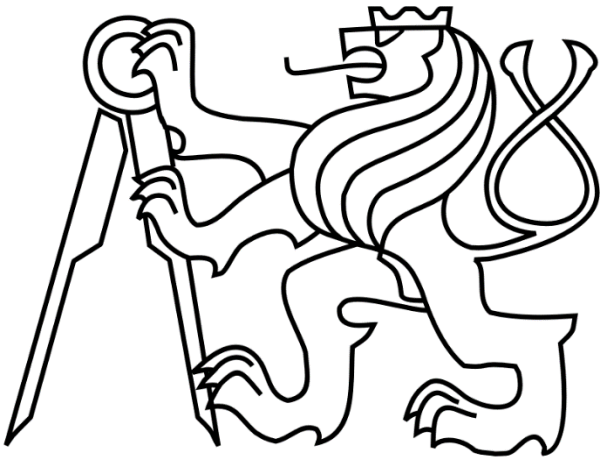
Příloha B	Kompletní projekt programu TIA Portal V13
Příloha C	Videa provozu řídícího programu na výukové sestavě stroje
Příloha D	Kompletní bakalářská práce v PDF

Příloha B a Příloha C jsou z důvodu značné datové velikosti nahrány pouze na přiloženém DVD.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STROJNÍ

Ústav výrobních strojů a zařízení



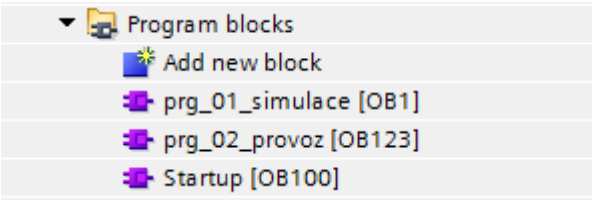
Přílohy

Návrh řešení pro SIL simulaci výukové sestavy stroje

2017 **Jan Ferkl**

Úvod

Simulace výukové sestavy stroje je realizována jako samotný PLC program, který běží paralelně s řídícím programem (psaným uživatelem simulace). Uživatel simulace tedy svůj program píše v projektu, ve kterém už je simulace připravena. Situace je znázorněna na Obrázek 1.: Výpis programů. Program „prg_01_simulace“ obstarává simulaci a uživatel by do něj neměl zasahovat. Program „prg_02_provoz“ je určen uživateli k realizaci vlastního řídícího programu.



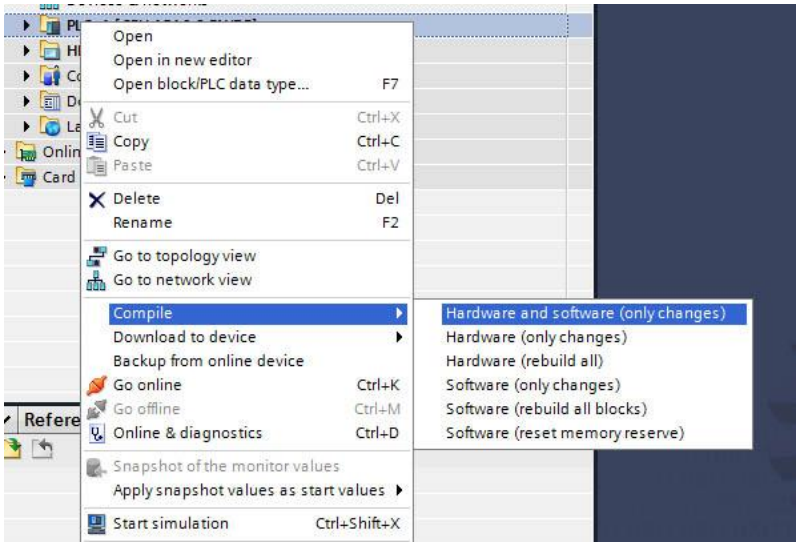
Obrázek 1.: Výpis programů

Spuštění simulace

K spuštění simulace (a testování řídícího programu) je nutné zdrojový kód zkompilovat, spustit simulaci samotného PLC v programu PLCSIM, nahrání zkompilovaných programů do virtuálního PLC a připravit tak zvané WATCH TABLES pro sledování a modifikaci hodnot proměnných v reálném čase. Seznámení s obsluhou „prg_02_simulace“ je také věnována samostatné kapitola. V této příručce je také popsán doporučený postup při analyzování průběhu simulace pomocí nahrávání obrazovky počítače pomocí aplikace OBS Studio.

Kompilace zdrojového kódu

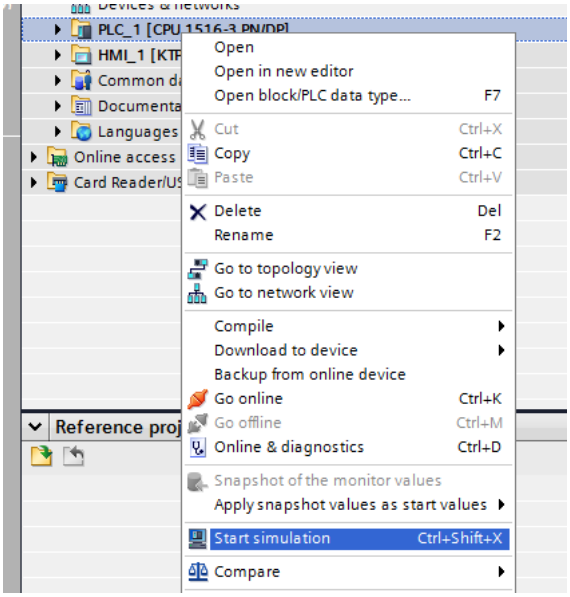
Zkompilovat zdrojový kód je nutné před jeho nahráním do PLC (ať už simulovaného nebo skutečného). Tento úkon se provede pomocí příkazu „compile-Hardware and software (only changes)“.



Obrázek 2.: Kompilace zdrojového kódu

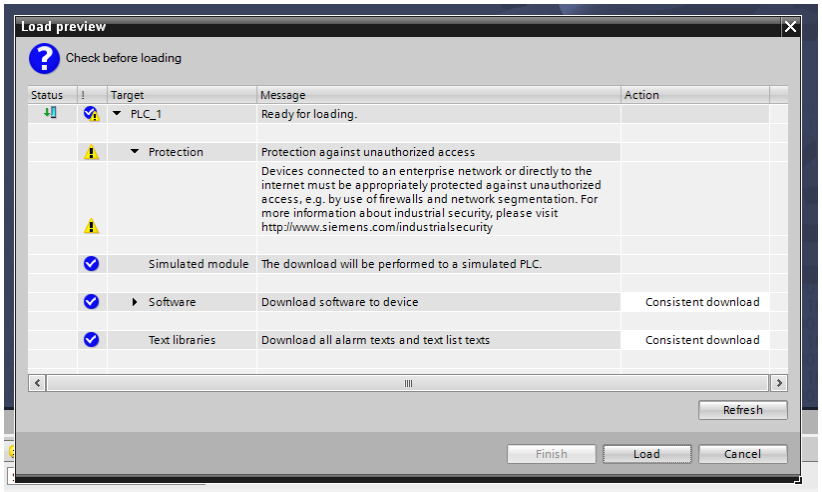
Simulace PLC, nahrání zkompilovaného projektu

Ke spuštění simulace slouží tlačítko „Start simulation“.

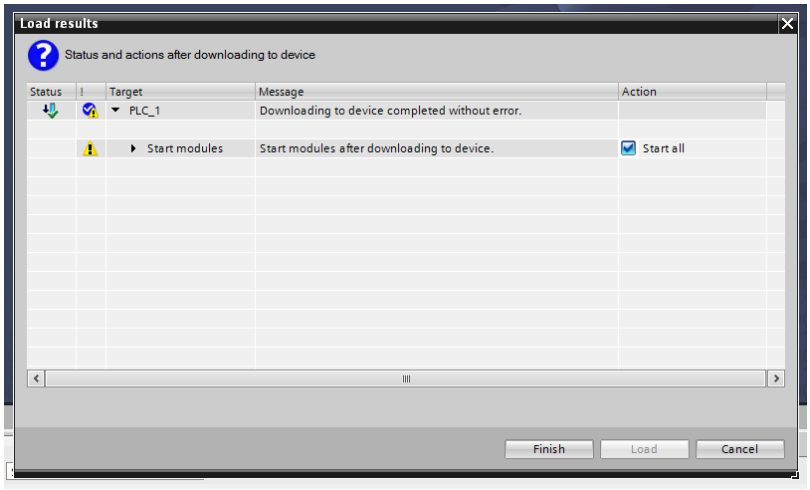


Obrázek 3.: Spuštění simulace PLC

Po zmáčknutí tlačítka se spustí PLCSIM a protokol nahrání projektu do virtuálního PLC. Nahrávání je potřeba potvrdit zmáčknutím tlačítka load a na další obrazovce zaškrtnout „start all“ (zahájí provoz PLC) a následně operaci dokončit pomocí „Finish“.

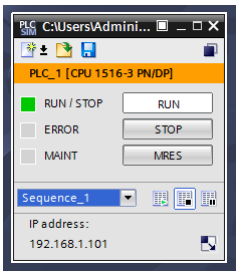


Obrázek 4.: Okno „Load preview“



Obrázek 5.: Okno „Load result“

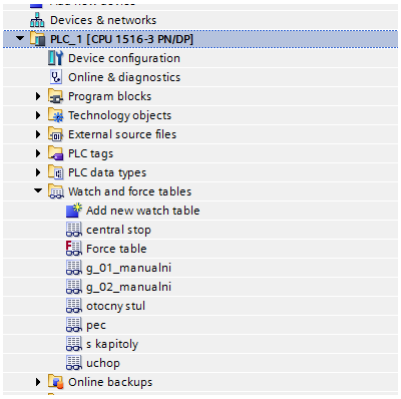
Tím se spustilo virtuální PLC, které je možné ovládat v samostatném okně (aplikace PLCSIM). Příkaz RUN je v případě předchozího potvrzení „start all“ aktivní. Příkazem „STOP“ a následně obnovením „RUN“ je možné PLC restartovat a tím i vymazat paměť.



Obrázek 6.: PLCSIM

Watch tables

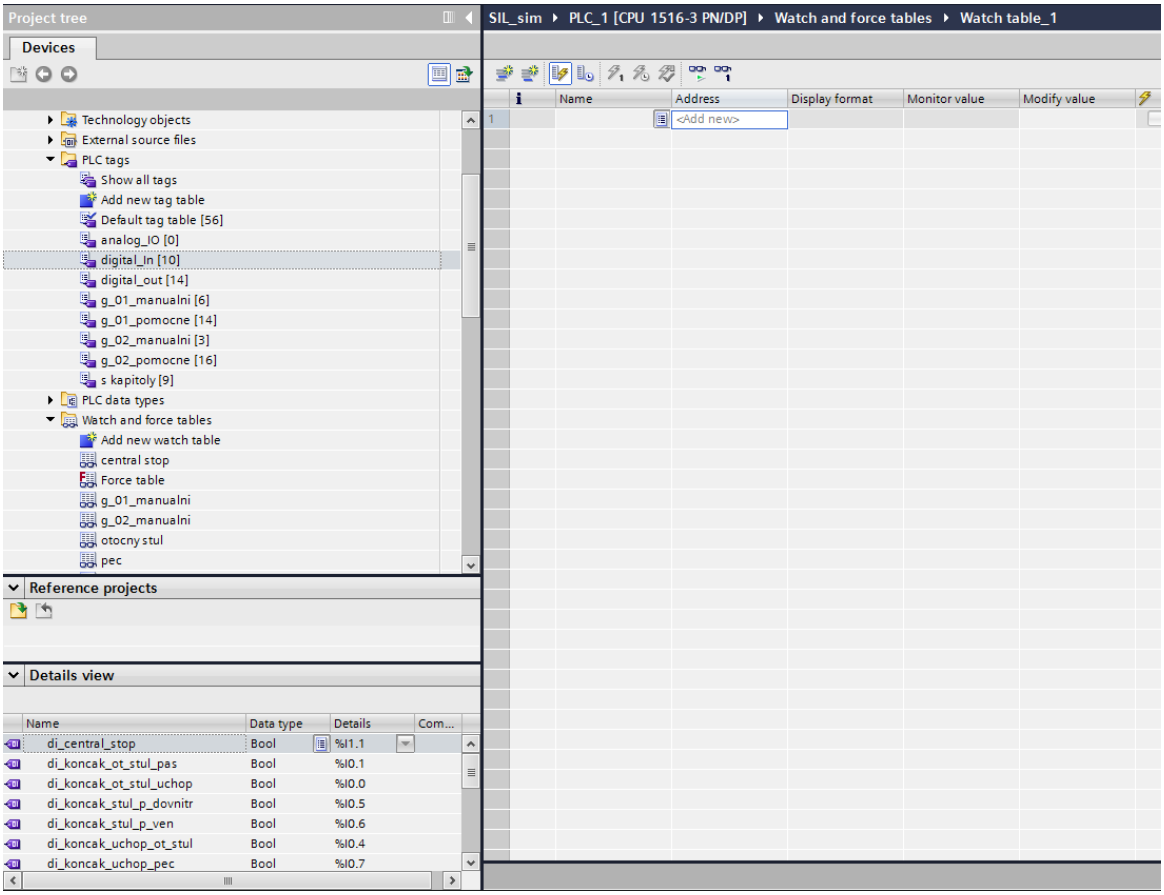
Pro sledování hodnot proměnných v reálném čase slouží funkce „Watchtables“.



Obrázek 7.: Watchtables

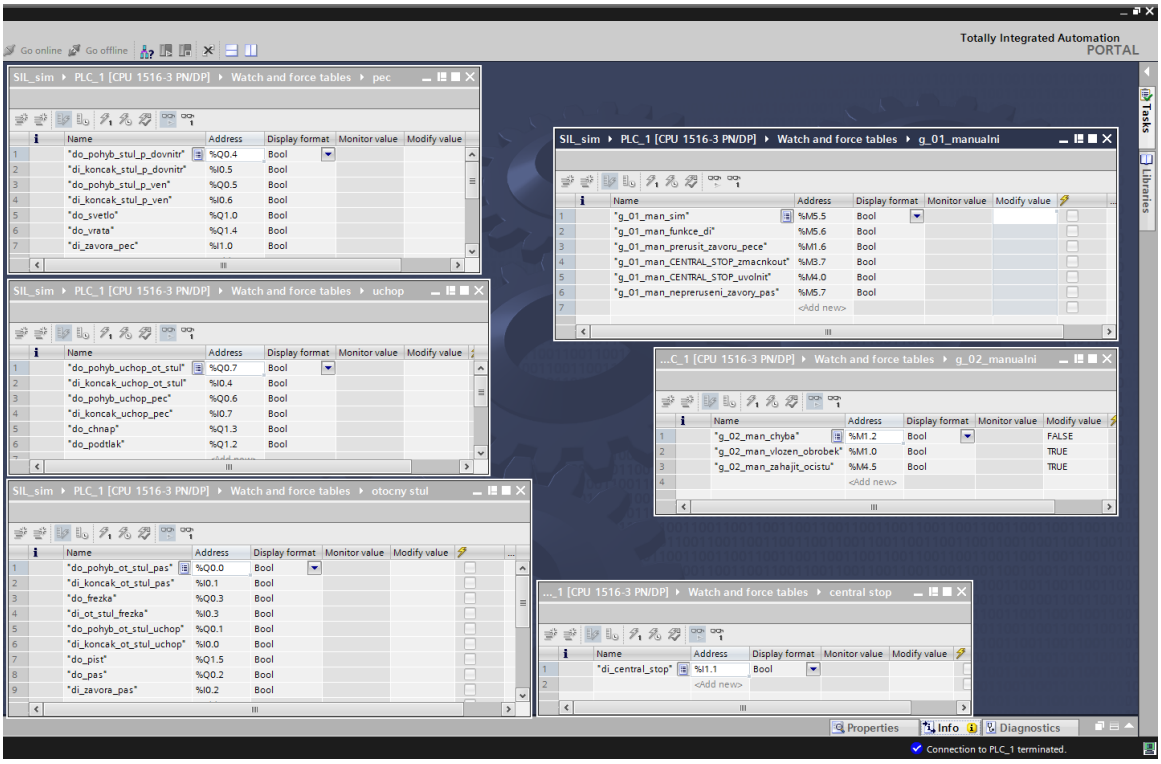
Příkazem „Add new watch table“ je možné přidat novou tabulku, do které lze následně přidávat proměnné (tagy), které uživatel chce sledovat. Přidání proměnných do „watch table“ je nejlépe provést pomocí otevření okna příslušné „watch table“, označením příslušné složky proměnných v „Project tree“ a následnému přetáhnutí

konkrétních proměnných z levého dolního rohu („detail view“) do aktivního okna „watch table“.



Obrázek 8.: Aktivní okno „watch table“ a označená složka PLC tagů

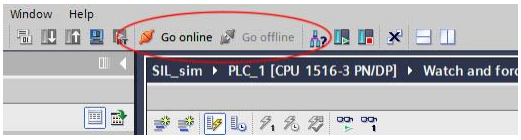
Pro přehlednost je vhodné rozdělit sledované proměnné do několika „watch tables“ a ty pak následně sledovat v režimu „float“.



Obrázek 9.: Rozložení oken v režimu „float“

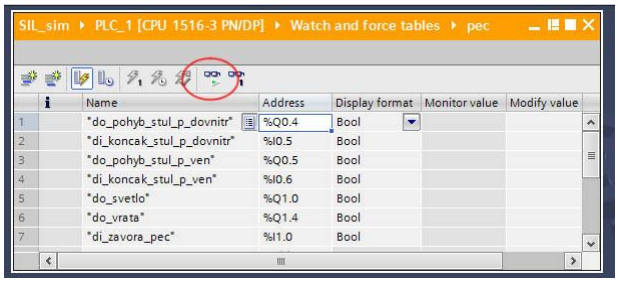
Jak je vidět na Obrázek 9.: Rozložení oken v režimu „float“, je vhodné proměnné rozdělit podle příslušných pracovišť. Manuálně ovládané proměnné obou programů jsou rozděleny do samostatných oken. Význam jednotlivých proměnných programu „prg_02_simulace“ bude vysvětlen v samostatné kapitole.

Protože už PLC, a tedy i nahrané programy běží (režim „RUN“ v okně aplikace PLCSIM), je možné přejít do režimu „online“ v rámci aplikace TIA Portal. Tím aplikace TIA PORTAL začne komunikovat s aplikací PLCSIM.



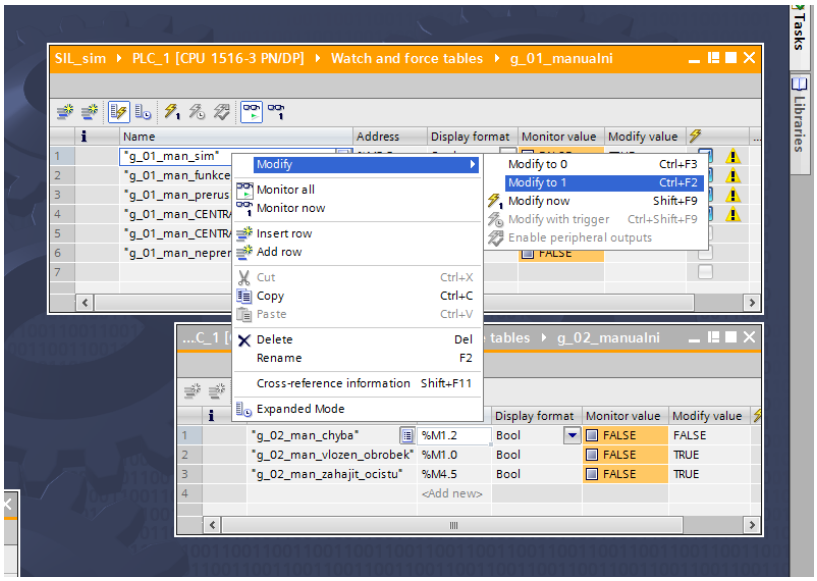
Obrázek 10.: Tlačítko „Go online“

Následně je potřeba v jednotlivých oknech „watch tables“ aktivovat příkaz „Monitor all“, který zajistí zobrazování aktuálních hodnot proměnných.



Obrázek 11.: Příkaz „Monitor all“

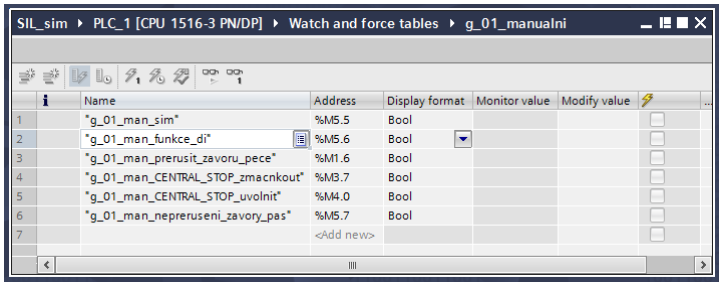
V rámci „watch tables“ je možné modifikovat hodnoty jednotlivých proměnných a tím ovládat proměnné určené k manuální obsluze (ve skutečném provozu se o tento úkon stará dotyková obrazovka HMI). K tomuto slouží příkaz „Modify“.



Obrázek 12.: Příkaz „Modify“

Tímto jsou popsány veškeré základní principy potřebné k provozování simulace. Následující kapitola je věnována významu manuálně ovládaných proměnných programu „prg_01_simulace“.

Manuálně ovládané proměnné



Obrázek 13.: Manuálně ovládané proměnné

Proměnná **„g_01_man_sim“** slouží k zahájení chodu programu „prg_01_simulace“ a je tedy nutné jí při každém zahájení simulace manuálně modifikovat na hodnotu 1. Po spuštění PLC má hodnotu 0, čímž umožňuje celý projekt (včetně simulace) nahrát i do reálného PLC. („prg_01_simulace“ po nahrání projektu do reálného PLC nesmí být aktivní. Jeho práce by byla v konfliktu se snímači zařízení. Proto je spuštění simulace řízeno manuálně.)

Proměnná **„g_01_man_funkce_di“** slouží k zapnutí a vypnutí funkčnosti veškerých digitálních vstupů PLC. Po modifikace „g_01_man_sim“ na hodnotu jedna se upraví i hodnota „g_01_man_funkce_di“. Do jednotlivých proměnných se při hodnotě nula přestane zapisovat a trvale tak nabydu nulovou hodnotu. Tím je simulován chybový stav zařízení, kdy přestane fungovat zpětná vazba systému. Je tedy možné testovat

bezpečnost řídícího programu. (Žádoucí je stav, kdy řídící program sám rozpozná situaci a zastaví pohyb.)

Proměnná **„g_01_man_prerusit_zavoru_pece“** slouží k pomyslnému vložení obrobku na stůl pece (začátek trati). Závora se sama automaticky obnoví poté, co podtlakový dopravník uchopí obrobek (simulace reaguje na sestupnou hranu do_1.3 a zároveň držený podtlak).

Proměnné **„g_01_CENTRAL_STOP_zmacknout“** a **„g_01_CENTRAL_STOP_uvolnit“** slouží k obsluze červeného tlačítka CENTRAL STOP, kterým je každé PLC v laboratoři vybaveno.

Proměnná **„g_01_nepreruseni_zavory_pas“** slouží k simulaci chybového stavu, kdy obrobek nedorazí na konec trati a tím je možné testovat bezpečnost řídícího programu. (Žádoucí je stav, kdy řídící program sám rozpozná situaci a zastaví pohyb.)

Nahrávání obrazovky pro analyzování výsledků simulace

Proměnných, které je potřeba během chodu programu sledovat je velké množství a je tedy žádoucí zjednodušit vyhodnocování výsledků chodu řídícího programu. Jednou z možností je po dobu simulace zaznamenávat obrazovku počítače a následně tento záznam analyzovat (zásadní je možnost obraz pozastavit). Autor simulace k tomu doporučuje freeware aplikaci OBS Studio. Druhá možnost je využít funkci „traces“, která slouží k logování průběhu signálu a grafickému znázornění v grafech.