VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV VÝROBNÍCH STROJŮ, SYSTÉMŮ A ROBOTIKY
INSTITUTE OF PRODUCTION MACHINES, SYSTEMS AND ROBOTICS

VIRTUÁLNÍ ZPROVOZNĚNÍ VÝROBNÍHO SYSTÉMU
VIRTUAL COMMISSIONING OF PRODUCTION SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jakub Bražina**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jan Vetiška, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav výrobních strojů, systémů a robotiky
Student:	**Bc. Jakub Bražina**
Studijní program:	Strojní inženýrství
Studijní obor:	Výrobní stroje, systémy a roboty
Vedoucí práce:	**Ing. Jan Vetiška, Ph.D.**
Akademický rok:	2018/19

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

Virtuální zprovoznění výrobního systému

Stručná charakteristika problematiky úkolu:

Virtuální zprovoznění výrobního systému se v současnosti stává nedílnou součástí procesu projektování a tvorby těchto systémů. Tato práce je zaměřena na virtuální zprovoznění robotického pracoviště, které se nachází v laboratoři ústavu výrobních strojů, systémů a robotiky. V průběhu práce na tématu si student má osvojit teoretické a praktické dovednosti v procesu virtuálního zprovoznění výrobního systému.

Cíle diplomové práce:

Rešerše dané problematiky.
Popis návrhu v prostředí Process Simulate.
Virtuální zprovoznění VS a ověření řídicího systému.
Návrh programů robotu výrobního systému.

Seznam doporučené literatury:

SICILIANO, Bruno a Oussama. KHATIB. Springer handbook of robotics. Berlin: Springer, c2008. ISBN 978-3-540-23957-4.

KOLÍBAL,Z. a kol.: Roboty a robotizované výrobní technologie. VUTIUM Brno, 2016, ISBN 978-8--214-4828-5.

NOF, S. Y. Springer Handbook of Automation. Springer, 2009. 1812 s. ISBN 978-3-540-78830-0.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2018/19

V Brně, dne

L. S.

doc. Ing. Petr Blecha, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Tato diplomová práce se zabývá virtuálním zprovozněním výrobního systému nacházejícího se v laboratořích Ústavu výrobních strojů, systému a robotiky VUT v Brně. V teoretické části práce je popsána problematika virtuálního zprovoznění, následována popisem jednotlivých zařízení nacházejících se v tomto výrobním systému. V praktické části je popsán návrh 3D modelu, návrh řídicího PLC programu a následně samotné virtuální zprovoznění. Na konci této práce je popsána tvorba programu robotu. Pro realizaci virtuálního zprovoznění bylo použito několika softwarových nástrojů firmy Siemens (TECNOMATIX Process Simulate, TIA Portal a PLCSIM Advanced 2.0).

ABSTRACT

This diploma thesis deals with virtual commissioning of production system which is located in the laboratories of the Institute of Production Machines, Systems and Robotics at the BUT. The issue of virtual commissioning is described in the theoretical part of the thesis, followed by a description of each device located in this production system. The design of the 3D model, the design of the PLC control program and also the virtual commissioning itself are described in the practical part of the thesis. There is the creation of the robot‘s program described at the end of the thesis. Several Siemens software tools were used for virtual commissioning realization (TECNOMATIX Process Simulate, TIA Portal and PLCSIM Advanced 2.0).

KLÍČOVÁ SLOVA

Virtuální zprovoznění, Process Simulate, PLCSIM Advanced, TIA Portal, Tecnomatix, Siemens, simulace robotického pracoviště, ABB Rapid program

KEYWORDS

Virtual commissioning, Process Simulate, PLCSIM Advanced, TIA Portal, Tecnomatix, Siemens, robotic workplace simulation, ABB Rapid program

BIBLIOGRAFICKÁ CITACE

BRAŽINA, Jakub. *Virtuální zprovoznění výrobního systému* [online]. Brno, 2019 [cit. 2019-05-23]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/116785. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav výrobních strojů, systémů a robotiky. Vedoucí práce Jan Vetiška.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce Ing. Janu Vetiškovi, Ph.D. za odbornou pomoc a další cenné rady při zpracování této diplomové práce. Dále děkuji za neskutečně cenné rady Alexandru Dinca z rumunské firmy ADA computers.

Největší dík si ovšem zaslouží mí skvělí přátele a samozřejmě rodina, za bezmeznou podporu a pomoc.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. Jana Vetišky, Ph.D. a s použitím literatury uvedené v seznamu.

V Brně dne 20.5.2019 ……

Jakub Bražina

OBSAH

1 ÚVOD

V dnešní době je vlivem nedostatků zaměstnanců ve výrobě snaha tento deficit eliminovat pomocí automatizovaných robotizovaných výrobních systému, které dosahují mnohdy větší rychlostí a přesnosti při výkonu požadovaných úkonů než člověk. Dalším značným benefitem těchto výrobních systému je možnost práce v takřka nepřetržitém provozu. Snahou je také, aby činnosti, které jsou opakované a algoritmizovatelné vykonával robot, což vede k možnosti využití lidského potenciálu při řešení mnohem komplexnějších úloh, které robot zatím nezvládne. Tato globální automatizace pak nachází uplatnění ve všemožných průmyslových odvětvích od výroby jídla až po těžký průmysl.

Dnes jsou požadavky na výrobce robotizovaných výrobních systému ze strany zákazníku čím dál větší. Hlavními požadavky jsou převážně snížení časových a finančních nákladů, avšak oproti tomu se stavící požadavky na co největší flexibilitu a možnost dodatečně úpravy navrhnutého výrobního systému. Aby bylo možné tyto náročné požadavky uspokojit, je vhodné vytvořit model výrobního systému ve 3D prostředí. Na tomto modelu mohou být pak simulovány a ověřeny veškeré procesní a technologické operace z hlediska časového rozložení, plánování materiálového toku, či kolizních stavů. Taktéž je možno simulovat obsluhu, u které je kladen důraz hlavně na ergonomii práce. Zároveň je poté tento model k dispozici po čas celého životního cyklu výrobního systému, díky tomu je snadná jeho případná úprava či rozšíření a následné nasazení do reálné výroby.

Tato diplomová práce se zaobírá problematikou virtuálního zprovoznění, což je využití zmiňovaného 3D modelu k ověření správnosti PLC programu použitého k řízení navrhovaného výrobního systému. Virtuální zprovoznění bylo provedeno na výrobním systému nacházejícím se v laboratořích Ústavu výrobních strojů, systému a robotiky na půdě fakulty strojního inženýrství Vysokého učení technického v Brně.

Aby bylo možné provést virtuální zprovoznění je nutné využít širokou škálu softwarových nástrojů. Jedná se zejména o prostředí, ve kterém je vytvářen 3D model výrobního systému se všemi potřebnými náležitostmi. Dále je nutné použít vhodné řídicí PLC a prostředí sloužící k vytvoření jeho programu. Posledním krokem je volba rozhraní pro komunikaci PLC programu s 3D modelem.

2 MOTIVACE

Tato práce byla vytvořena z důvodu poukázání na možnost povýšení 3D modelu, zadaného výrobního systému, v prostředí TECNOMATIX Process Simulate na plně dynamickou simulaci řízenou pomocí externího PLC, jehož program měl být posléze nasazen na řízení reálného výrobního systému v laboratořích ÚVSSR VUT v Brně.

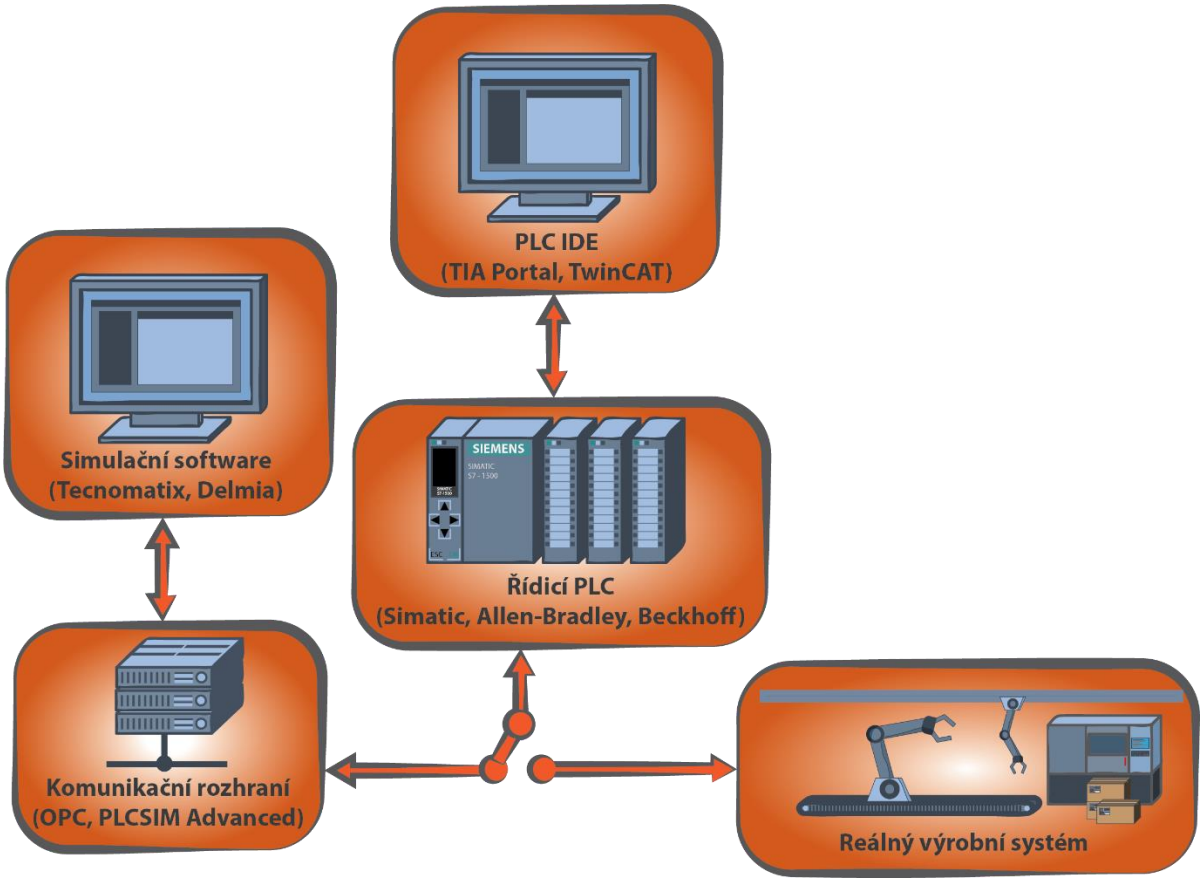
Snahou bylo vytvořit co možná nejvěrnější virtuální reprezentaci zadaného výrobního systému a k tomu navrhnout adekvátní PLC program. Při propojení těchto dvou částí pak mělo být provedeno virtuální zprovoznění poukazující na možnost tvorby a následného ladění výrobních operací a řídícího PLC programu, bez nutnosti použití fyzického hardwaru.

Zmíněné virtuální zprovoznění poté šetří v praxi značné množství času a celkových nákladů, což by do budoucna mohlo přilákat řadu firem majících zájem dále spolupracovat se školou na téhle problematice. Taktéž se toto virtuální zprovoznění může promítnou do výuky, kde by se budoucí studenti mohli s řešením této problematiky seznámit a vyzkoušet si ji na reálném výrobním systému v laboratořích ÚVSSR VUT v Brně.

3 PŘEHLED SOUČASNÉHO STAVU POZNÁNÍ

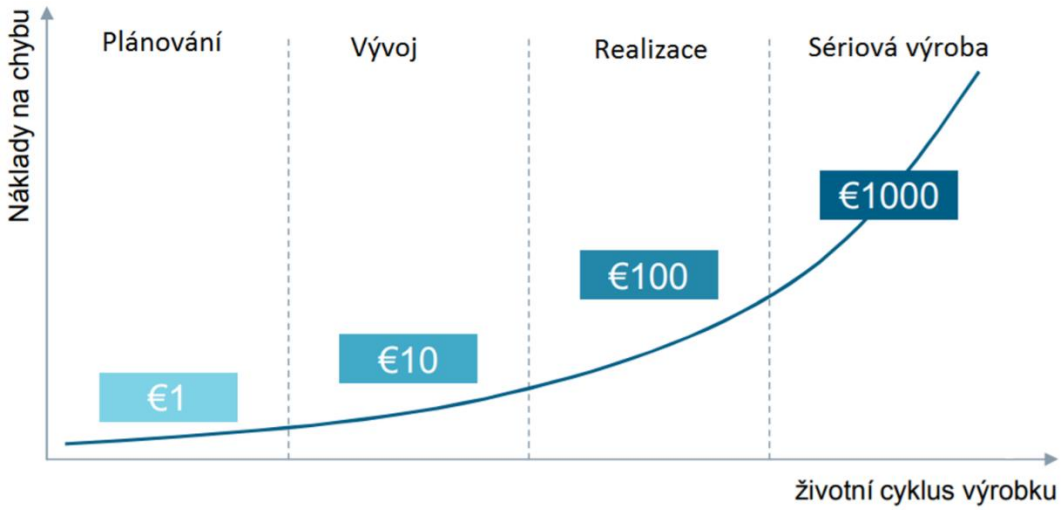
3.1 Virtuální zprovoznění

V návaznosti na čtvrtou průmyslovou revoluci (Průmysl 4.0) se stále více skloňuje pojem virtuální zprovoznění (Obr. 3-1). Při použití vhodných softwarových nástrojů se může stát virtuální zprovoznění výraznou součástí zavádění výrobních systému, či strojů do provozu, což se projeví ve značné úspoře celkových nákladů na jejich výrobu, fyzické zprovoznění a odladění, případně jejich následnou údržbu. Tato metodika je odrazem dnešní doby, kdy výrobci linek a strojů čelí stále větším požadavkům na snižovaní celkové ceny, avšak s protichůdnými požadavky narůstající složitosti, variability a flexibility navrhovaných zařízení. Jakožto řešení těchto požadavků se jeví tzv. virtuální inženýrství, které povyšuje obyčejný CAD model na komplexní virtuální model obsahující kompletní kinematiku pohyblivých komponent, strukturu senzorů a logickou řídicí strukturu. Takový model je pak možno použít pro simulaci chování vyvíjeného zařízení ve 3D prostředí, což vede k možnosti ověření celkového návrhu konstrukce, kontrole kolizního chování, ověření trvání jednotlivých operací a konečně k verifikaci navrženého řídicího PLC programu, ovládacích panelů (HMI) a zkoumání situací co-kdyby. Model vyráběné linky nebo stroje je společný pro všechna oddělení vývoje a ulehčuje tak práci konstruktérům, provozním inženýrům a programátorům robotů při navrhování optimálního řešení zadaného projektu [1].



Obr. 3-1 Schéma virtuálního zprovoznění

Důležitost propracovaného virtuálního modelu výrobního systému a jeho následné simulování, spočívá především v možnosti odhalení chyb v návrhu a jejich odstranění před reálnou výrobou takového systému. Pokud nedojde k dostatečnému odladění výrobního systému a tím i výrobního procesu už v průběhu plánování a vývoje, velikost nákladu roste exponenciálně. V tomto případě platí tzv. „*pravidlo desítek*“ (Obr. 3-2), ze kterého vyplývá, že výše nákladu na odstranění chyby narůstá desetkrát v průběhu postupu etapami životního cyklu výrobku [2].



Obr. 3-2 Pravidlo desítek [2]

Mezi hlavní benefity virtuálního zprovoznění tedy patří:

- Verifikace a odladění programu v řídicích jednotkách (PLC)
- Značná úspora času na inženýring
- Ověření správného návrhu senzoru a jejich napojení na PLC
- Testovaní blokací a bezpečnostních prvků
- Testování různých variant konstrukčních řešení
- Testování různých variant řídicích programů
- Odladění chybových scénářů
- Optimalizace celkového návrhu bez nutnosti testování na reálném zařízení
- Významná úspora času při oživování a ladění reálného zařízení
- Velmi brzké zaškolení obsluhy ještě před samotným reálným zprovozněním
- Opravdový mechatronický přístup díky propojení mechaniky, elektroniky a softwaru [1]

3.2 Programování průmyslových robotů

Jelikož jsou důležitou součástí výrobních systému průmyslové roboty, a protože dílčím cílem této práce je zpracování programu robotu, byla v této kapitole přiblížena problematika programování průmyslových robotů.

V dnešní době je pomocí robotů automatizováno množství výrobních operací, kdy nejčastěji se jedná o operace svařovací (kontinuální nebo bodové), lepení, kalení, dále tzv. pick and place operace, například zakládání do stroje, odkládání obrobených součástí

na palety a spoustu dalších. Průmyslové roboty lze programovat dvěma hlavními způsoby on-line a off-line.

3.2.1 On-line programování

Programování robotů se provádí on-line, to znamená, že programátor je přítomen v blízkosti robotu a program pro reálnou aplikaci je vytvářen zadáváním příkazů přímo do řídicího systému robotu. On-line programování se dále dělí na programování přímé a nepřímé.

Přímé programování:

Jedná se o manuální programování spočívající v tom, že nástroj robotu je veden obsluhou a řídicí systém si ukládá jednotlivé trajektorie a rychlosti, kde posléze v automatickém režimu vykonává tuto naučenou trajektorii [3].

Nepřímé programování:

Jedná se o dálkové ovládané programování spočívající v zadávání jednotlivých bodů trajektorie pomocí řídícího panelu. Trajektorie mezi těmito body je pak dopočítána řídícím systémem robotu [3].

3.2.2 Off-line programování

Druhým přístupem je off-line programování (OLP). Program je vytvořen na základě simulací v počítači, kde je reálna aplikace reprezentovaná 3D modelem a následně je tento program nahrán do robotu. Pro tvorbu simulací slouží široká škála softwaru. Lze nalézt free softwary (RoboDK), dále se na trhu nacházejí softwary jednotlivých výrobců průmyslových robotů (KUKA Sim – KUKA, Roboguide – Fanuc, RobotStudio – ABB). Oficiální softwary od výrobců disponují kopií reálného řídicího systému, díky tomu jsou pak dosažené simulované časy téměř shodné s realitou. Ve většině případu tyto softwary využívají k řízení navrhovaného výrobního systému pouze řídicího systému robotu, bez možnosti řízení pomocí externích PLC. Dalšími softwary jsou softwary zaměřené na tvorbu simulací s možností využití množství robotů od různých výrobců (WorkSpace5, RoboLogix, RobotExpert). Poslední skupinou softwaru umožňujících simulování výrobních systému a tím i off-line programování robotů jsou softwary z balíku tzv. digitálních továren, které jsou používaný v kombinaci se softwary zajišťující správu životních cyklů (PLM) vyráběných produktů. Tyto softwary umožňují vytvářet simulace komplexnějších a větších výrobních systému k jejíž řízení nestačí pouze řídicí systém průmyslového robotu. V těchto balících se nachází softwarové nástroje usnadňující vývoj produktu od návrhu přes konstrukci, plánování výroby, po simulování a ověření výrobních a logistických procesů. Důležitým aspektem těchto simulačních softwaru je rovněž možnost virtuálního zprovoznění. Mezi simulační softwary těchto balíku patří například DELMIA firmy Dassault Systèmes, Process Simulate digitální továrny Tecnomatix firmy Siemens a FASTSUITE Edition 2 německé firmy Cenit. Jednotlivé významné softwary sloužící k off-line programování a virtuálnímu zprovoznění jsou přiblíženy v další kapitole.

3.3 Nástroje potřebné k virtuálnímu zprovoznění

Pro virtuální zprovoznění výrobního systému je potřeba několika softwarových nástrojů. Jako první je nutné využít simulačních softwaru, kde je vytvořen kompletní 3D model navrhovaného systému s veškerými kinematickými strukturami a s prvky logiky, které následně slouží pro řízení a tvorbu simulací jednotlivých operací, jež bude výrobní systém

vykonávat. Dále je nutné zvolit vhodnou platformu PLC, kterou bude výrobní systém řízen a naprogramovat adekvátní řídicí program. Posledním neméně důležitým nástrojem je software zajišťující komunikační rozhraní mezi simulovaným 3D modelem a řídicím PLC.

3.3.1 Simulační softwary

ABB RobotStudio

ABB RobotStudio (dále RS) je softwarovým nástrojem určeným k off-line programování průmyslových robotů společnosti ABB. Hlavní výhodou tohoto simulačního softwaru je to, že simulované roboty jsou řízeny pomocí technologie VirtualRobot™, což je přesná kopie reálného řídicího systému a díky tomu časy simulovaných pracovních cyklů odpovídají hodnotám v reálném provozu. Pro jednoduchou tvorbu 3D modelu simulovaného výrobního systému je do RS možno importovat většinu standardizovaných CAD formátu (IGES, STEP, CATIA atd.). Další velmi zajímavou funkcí je Multimove, která umožňuje řídit několik robotů pomocí jednoho kontroléru [4], [5], [6].

Mezi další funkce RS patří například:

- **AutoPath** umožňuje na základě použitého 3D modelu generovat body dráhy robotu
- **AutoReach** umožňuje analyzovat dosah robotu a tím ověřit vhodnost jeho umístění
- **PathOptimalization** je funkce, která analyzuje ty trajektorie, ve kterých robot prochází blízko singulárním bodům a tím pádem je možné udělat včasnou korekci
- **CollsionDetection** tato funkce sleduje, jestli za chodu programu nedochází ke kolizím vybraných součástí modelu [6]

Velmi užitečnou součásti RS je tzv. VirtualFlexPendant, který odpovídá reálnému řídicímu panelu robotu a umožňuje využívat všechny jeho funkce v rámci RS. To vede k možnosti zaškolení obsluhy bez rizika poškození reálného robotu a prvků v jeho okolí [3], [6].

Novinkou je rovněž možnost ověření simulace v prostředí virtuální reality za použití systému virtuální reality jako je například Oculus Rift, HTC Vive apod. V nejnovějších verzích RS je možno průmyslové roboty ve virtuální realitě přímo programovat, což off-line programování posouvá technologicky stále více dopředu. V budoucny by například mohla existovat síť 3D skenerů, která by přenášela v reálném čase reálné výrobní prostředí do této virtuální reality, kde by pak programátor programoval, případně prováděl korekci programu robotu získaného z RS nebo jiných softwarů, jako by byl připojen k robotu on-line.

Hlavní nevýhodou RS, je zaměření pouze na roboty firmy ABB a tím pádem nemožnost provádět off-line programování robotů jiných výrobců. Virtuální zprovoznění je rovněž omezeno pouze na řídicí systémy ABB, bez možnosti připojení externích PLC.

FASTSUITE Edition 2

Mezistupeň mezi softwary pro off-line programování a virtuální zprovoznění jednotlivých výrobců průmyslových robotů a plně univerzálními balíky digitálních továren tvoří software německé firmy Cenit FASTSUITE Edition 2, který existuje ve dvou verzích. První verze je implementovatelná do softwaru CATIA/DELMIA V5 francouzské firmy Dassault Systèmes. Druhá verze existuje ve formě nezávislé platformy [7].

FASTSUITE Edition 2 je platforma se silnou orientaci na aplikace průmyslových robotů a obráběcích strojů, kde poskytuje rozšiřitelné řešení pro různou komplexnost navrhovaného výrobního systému od jednoho stroje nebo robotu po kompletní výrobní linky. Tento software obsahuje všechny důležité prvky nutné k úspěšnému zprovoznění výrobního systému, jako jsou manipulace s materiálem, automatizace výroby, ověření navržených operací, off-line programování, virtuální zprovoznění atd. Rozhraní je koncipováno tak, aby bylo co nejvíce uživatelský přívětivé a jednoduché i při nárustu komplexnosti řešeného projektu [7].

Hlavní výhodou oproti ABB RS je možnost instalace kontroléru různých výrobců robotů a rovněž připojení externích PLC, čímž je docíleno větší všestrannosti tohoto softwaru [7].

Digitální továrny

Digitální továrny jsou balíky softwarů, které slouží k zajištění správy celého životního cyklu výrobku (PLM). To znamená, že je navrhnut výrobek a následně jeho výroba, kde se tato výroba skládá z nespočtu úkonu, jako je vhodné rozložení celé výrobní haly, následně rozložení jednotlivých výrobních buněk, plánování logistiky, toku materiálu, kapacit skladů atd. Pro všechny výše zmíněné úkony se v balíku digitální továrny nachází konkrétní software řešící danou problematiku, případně se vytváří kombinace softwarů odpovídající zákaznickým požadavkům.

Digitální továrna DELMIA

Jelikož je tato práce zaměřena na virtuální zprovoznění výrobního systému, jsou zde přiblíženy pouze softwary, které konkrétně umožňují virtuální zprovoznění, jako tomu je u balíku digitální továrny DELMIA, kde k tomuto účelu slouží software Delmia V5 Robotics. Značnou výhodou Delmie V5 Robotics je rozsáhla knihovna obsahující přes 700 přesných modelů průmyslových robotů. Delmia V5 Robotics nabízí dostupně, flexibilní a uživatelský přívětivé řešení pro programování robotů, rozvržení a následnou simulaci výrobních buněk. Díky struktuře V5 je garantována jednoduchá návaznost dalších nástrojů, které slouží například k plánování výroby a optimalizaci rozvržení výrobních prostor podniků (Delmia Process Engineer), nebo je možné rozšířit softwarový balík o Delmia V5 Human, kde lze mimo navrhování pracoviště z hlediska mechatronického ověřovat ergonomii, bezpečnost a rozvrhovat operace plněné obsluhou s vidinou dosažení co nejvyšší efektivity práce [8].

Digitální továrna TECNOMATIX

Digitální továrna Tecnomatix firmy Siemens poskytuje širokou škálu softwarových nástrojů propojujících všechny disciplíny týkající se výroby produktu. Jednotlivé softwary digitální továrny Tecnomatix mohou být vzájemně propojeny se zaručenou kompatibilitou, ale tyto softwary jsou rovněž použitelné samostatně. Umožňuje také přímé napojení na ostatní softwary firmy Siemens, jako jsou NX, Teamcenter atd. Jednotlivými nástroji digitální továrny Tecnomatix jsou Plant Simulation, Process Designer, Robot Expert, Robcad, Jack a Process Simulate [9].

Process Designer slouží k hrubému plánování, kde je navržené předběžné rozložení celé výrobní buňky, linky či haly. Na těchto modelech jsou pak prováděny odhady nákladů a celkových času cyklů. Následně je celý model importován do Process Simulatu, kde dochází k jemnému plánování a verifikování návrhu provedených v Process Designeru [8], [9].

Process Simulate, dále jen PS, je modulární software sloužící k jemnému plánování, čímž je myšleno simulování nejrůznějších výrobních nebo procesních operací v návaznosti na modely získané z Process Designeru. PS lze rozdělit do dvou částí: PS Robotics, sloužící k tvorbě dynamických simulací zahrnující roboty a PS Human, zprostředkovávající simulace operací prováděných osobami, kde dochází převážně k ověření ergonomie. Obě části je možno mít zahrnuty v jedné simulaci a verifikovat tak práci robotů i člověka najednou [8], [9].

Plant Simulation má přímou vazbu na předchozí softwary a slouží k dynamickému simulování diskrétních událostí, tvorbě digitálních modelů výrobních a logistických systému umožňující následné zkoumání jejich charakteristik a optimalizaci jejich výkonnosti. Umožňuje simulaci různých situací, které mohou v systému nastat a jejich následnou interpretaci. Díky řadě analytických nástrojů, je zprostředkováno značné množství spolehlivých dat, což vede k možnosti dělat zásadní rozhodnutí a upravovat proces výroby v počátcích jejího plánování. Rovněž lze na základě těchto simulací optimalizovat toky materiálu, využití zdrojů a logistiky na všech úrovních plánování od jednotlivých strojů, až po globální výrobní závody [8], [9].

V této práci bylo použito pouze PS Robotics, kde byly vytvořeny a simulovány veškeré robotické operace. PS umožňuje tvorbu dvou typu simulací, kde první typ je časově řízená simulace a druhý typ je simulace řízena událostmi. Při událostním řízení jsou simulace řízeny pomocí logické signálové struktury, kdy je možno tuto strukturu napojit na externí PLC, čímž je docíleno virtuálního zprovoznění.

3.3.2 PLC

PLC (Programovatelný logický automat) je průmyslový počítač navržený pro práci v nepříznivém prostředí průmyslových podniků, sloužící k řízení výrobních procesů. PLC je zkonstruováno tak, aby bylo jednoduché jeho programování, ladění, změna programu a aby bylo co nejspolehlivější [10].

Hlavním rysem PLC je, že program je vykonáván cyklicky v nekonečné smyčce, kdy odpověď na všechny části programu musí být realizována ve stanovenou dobu, což je délka jednoho cyklu smyčky. To znamená, že běží v reálném čase. Každý cyklus smyčky je vykonáván ve třech krocích:

1. Načtení vstupů a jejich dočasné uložení do paměti
2. Vyhodnocení veškeré logiky na základě načtených vstupů
3. Nastavení výstupů na základě vyhodnocené logiky [11]

PLC hrají při virtuálním zprovoznění kritickou roli, jelikož hlavním úkolem virtuálního zprovoznění je verifikace funkčnosti PLC programu výrobního systému, který je graficky reprezentován 3D modelem v simulačním softwaru [10].

Na trhu se dnes nachází PLC mnoha výrobců. Z globálního hlediska lze konstatovat, že největšímu rozšíření se pyšní PLC firmy Siemens, následující PLC americké firmy Rockwell Automation a japonské Mitsubishi Electric. Dalšími výrobci jsou pak Schenider Electrics, Omron, B&R atd. Na ÚVSSR se využívají také PLC německé firmy Beckhoff. Důležité je zmínit, že není možné jednoznačně říct, který výrobce je nejlepší, jelikož vždy záleží na konkrétní aplikaci a požadavcích zákazníků [12], [13].

Pro virtuální zprovoznění a následné řízení zadaného výrobního systému bylo použito PLC firmy Siemens SIMATIC S7-1512C-1PN, momentálně jedno z nejvýkonnějších PLC v portfoliu firmy Siemens, což je vhodné zejména pro další výzkumné účely [14].

3.3.3 Softwary zajišťující komunikační rozhraní

Aby bylo možné propojit simulační software a řídicí PLC je nutné použít některý z nabídky softwarových nástrojů tuto komunikaci zajišťující. Standardně používanou je technologie OPC, která je použitelná u všech simulačních softwaru umožňujících virtuální zprovoznění. Další nástroje závisí na výrobci simulačního softwaru. Jelikož byl v této práci používán Tecnomatix Process Simulate firmy Siemens, byly zde popsány jeho možnosti připojení k externímu PLC. To je umožněno technologií OPC, díky softwarovému emulátoru PLCSIM Advanced 2.0, staršímu PLCSIM, nebo případně SIMITu.

Technologie OPC

OPC (Open Platform Communications) je standardem průmyslové komunikace, který byl vytvořen ve spolupráci mnoha světových společností zabývajících se průmyslovou automatizaci a společností Microsoft. Jedná se o rozhraní umožňující snadné připojení a komunikaci mezi různými zařízeními určenými pro monitorování a řízení výrobních procesů, což vede k zamezení závislosti softwaru na výrobci hardwaru. O vývoj technologie OPC se stará mezinárodní organizace OPC Foundation. OPC existuje v několika specifikacích vytvářející OPC standard [15]. Process Simulate je možno připojit skrze OPC DA nebo novější OPC UA.

OPC DA (Data Access Protocol) je nejzákladnějším protokolem OPC standardu a je omezen pouze na platformu Windows [15].

OPC UA (Unified Architecture) bylo vytvořeno společností OPC Foundation se zaměřením primárně na komunikaci jednotlivých zařízení, založené na komunikačních standardech TCP/IP, díky čemu není omezená platforma pouze na Windows. Protokol OPC UA rovněž disponuje robustnějším zabezpečením než jeho předchozí specifikace [16].

PLCSIM Advanced 2.0

PLCSIM Advanced 2.0 je softwarový nástroj, sloužící k simulování PLC, který přišel s verzí TIA Portal V14. Největší výhodou oproti klasickému simulátoru S7-PLCSIM je možnost propojení s OPC UA a schopnost softwarů rodiny Tecnomatix přímo komunikovat se softwarem PLCSIM Advanced, což má za následek zefektivnění využívání digitálních továren. Nevýhodou je možnost simulovat pouze PLC Simatic S7 řady 1500 a kontrolér ET200SP [2].

PLCSIM

PLCSIM je předchůdce softwaru PLCSIM Advanced běžící na lokálním zařízení a sloužící k simulaci PLC programu, přímo v prostředí TIA Portal. PLCSIM rovněž umožňuje simulovat širší škálu PLC automatů z řady Simatic S7 a taktéž umožňuje simulovat PLC program ve starším prostředí STEP7 [17].

SIMIT

SIMIT je software firmy Siemens, sloužící k virtuální simulaci zprovozňovaného systému. Simulovat je možno jednotlivé stroje, linky, ale také celé továrny, elektrárny atd. Hlavní výhodou SIMITu oproti PLCSIM Advanced 2.0 je širší konektivita k různým řadám PLC [18].

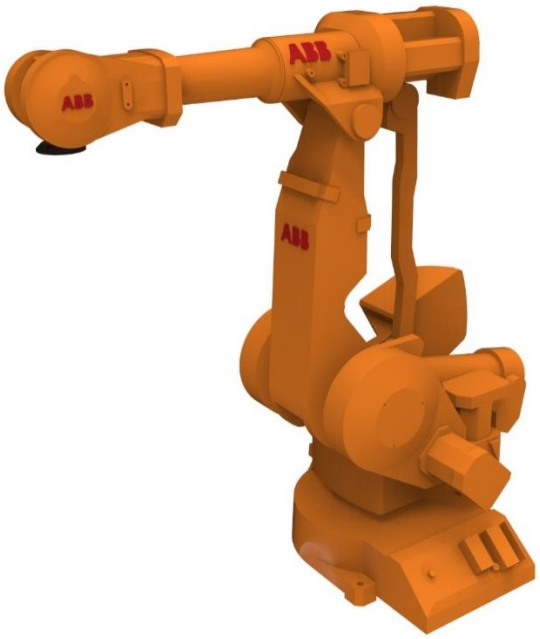
3.4 Analýza zadaného výrobního systému

Virtuální zprovoznění bylo provedeno na výrobní buňce nacházející se v dílně Ústavu výrobních strojů, systému a robotiky (Obr. 3-3). Tato buňka se skládá z množství zařízení, které byly dále popsány.

Obr. 3-3 Zprovozňovaný výrobní systém v laboratoři ÚVSSR

3.4.1 Průmyslový robot ABB IRB 4400/60

ABB IRB 4400/60 (Obr. 3-4) je velmi rychlý a kompaktní šestiosý průmyslový robot umožňující manipulaci se středně těžkými až těžkými součástmi. V případě použití dvojitého chapadla a díky své nosnosti až 60 kg při velmi vysokých rychlostech, dovoluje manipulaci i se dvěma součástmi zároveň. Robot rovněž svou všestrannosti a robustní konstrukcí, umožňuje použití v široké škále aplikaci jako jsou lepení, řezání, broušení, měření atd. Verze s krytím IP 67 jsou také vhodně pro aplikace v lakovnách, slévárnách a obecně nepříznivých prostředích. Pro řízení robotu je použito řídícího systému firmy ABB IRC5 [19].



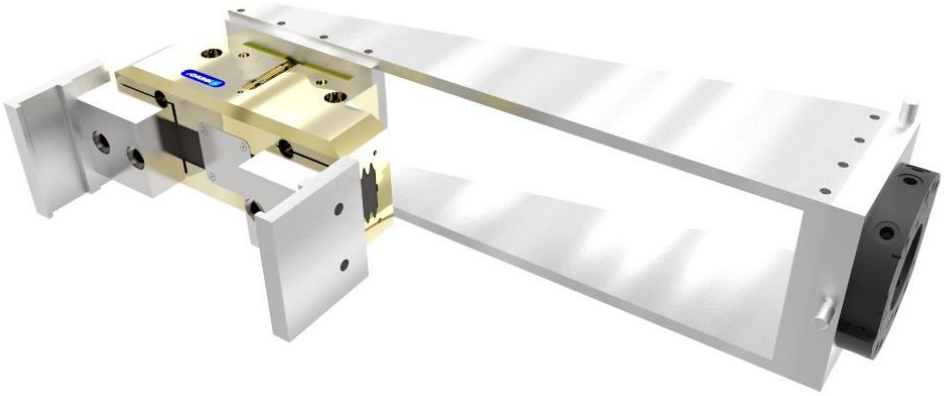
Obr. 3-4 Průmyslový robot ABB IRB 4400/60

3.4.2 Nástroje robotu

Pro manipulaci se součástmi je použito tří nástrojů robotu, které je možno řadit z hlediska použití mezi uchopovače.

Nástroj s chapadlem Schunk

Nástroj pro manipulaci s obrobkem (Obr. 3-5) slouží k obsluze tříosé CNC vertikální frézky MCV 754 QUICK. Chapadlo Schunk PGN plus 200 je pneumaticky poháněno, dosahující zdvihu čelisti 25 mm, uchopovací síly až 2700 N a hmotnosti 5,4 kg [20].



Obr. 3-5 Nástroj pro manipulaci s obrobkem – Schunk

Nástroj s chapadlem SMC

Nástroj s chapadlem SMC (Obr. 3-6) slouží pro manipulaci s kostkami. Chapadlo SMC MHZ2 – 40D je pneumaticky poháněno o hmotnosti 1,27 kg, dosahující zdvihu čelistí 30 mm a uchopovací síly až 318 N [21].

Obr. 3-6 Nástroj pro manipulaci s kostkami – SMC

Nástroj s přísavkovým chapadlem

Nástroj s přísavkovým chapadlem (Obr. 3-7) složí pro manipulaci s kartónem a je řízen elektropneumatickými ventily SMCVCQ2200KN – 5W, operující v tlakovém intervalu 0,1 – 1 MPa [22].

Obr. 3-7 Nástroj s přísavkovým chapadlem

3.4.3 Automatická výměna nástrojů

Aby mohl robot plnit celou škálu operací, pro které jsou nutné různé nástroje, je potřeba jej osadit systémem pro automatickou výměnu nástrojů. V zadaném výrobním systému je zastaralá výměna nástrojů, která bude do budoucna nahrazena modernějším systémem firmy Schunk SWS (Obr. 3-8). Ten se skládá z hlavy pro rychlou výměnu (SWK-060), umístěné na přírubě robotu a adaptéru (SWA-060) umístěném na každém nástroji. Tento systém je určen

pro hmotnosti dosahující až 75 kg, což je při maximální nosnosti robotu 60 kg více než dostatečné [23]. V práci bylo dále použito systému SWS i když ve skutečnosti byla používána stále starší verze.

Obr. 3-8 Systém automatické výměny nástroje Schunk SWS [23]

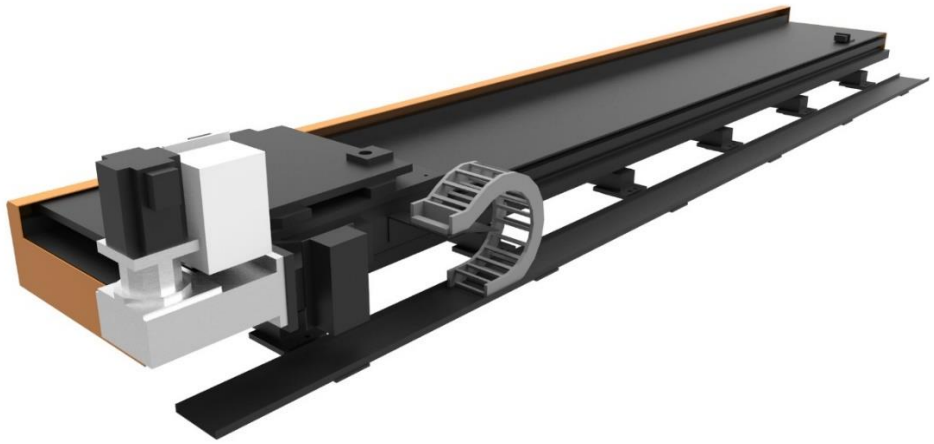
3.4.4 Řídicí systém robotu ABB IRC5

Pro řízení robotu je použit řídicí systém firmy ABB IRC5 (Obr. 3-9). Tento řídicí systém umožňuje robotům firmy ABB vykonávat úkony velmi efektivním způsobem. Díky pokročilému dynamickému modelování je výkon robotu automaticky optimalizován pomocí zkracování dob cyklů jednotlivých úkonů a zprostředkováváním velmi přesné trajektorie. To znamená, že přesnost trajektorie není závislá na rychlosti. Řídicí systém ABB IRC5 disponuje také lepší bezpečností označovanou jako *„SafeMove2“*, která umožňuje tvorbu bezpečnějších, flexibilnějších a rozměrově menších výrobních buněk, zprostředkovávajících kooperaci člověka a robotu. Pro programování úkonu robotu využívá řídicí systém ABB IRC5 jazyku RAPID [24].

Obr. 3-9 Řídicí systém robotu ABB IRC5

3.4.5 Pojezd ABB IRBT 4003

Pojezd ABB IRBT 4003 (Obr. 3-10) přináší robotu IRB 4400 sedmou osu, která rozšiřuje jeho pracovní prostor. Nejdůležitějším aspektem těchto pojezdů je rychlost, která je až o 40% větší v porovnání s předchozími modely. Další předností tohoto pojezdu je možnost plného zakomponování do řídicího systému robotu a tím tak využití všech jeho vlastností, týkající se dynamiky a přesnosti polohování. Pojezd je modulární a lze jej získat v délkách 1.9–19.9 m s maximální rychlostí 2 m/s a zrychlením 2.5 m/s^2. Rovněž jej je možné osadit jedním nebo dvěma roboty, které lze řídit pouze jedním řídicím systémem pomocí funkce MultiMove®, což vede k dalšímu zefektivnění výroby [25]. V buňce ÚVSSR je použito pojezdu s aktivní délkou 4,7 m.



Obr. 3-10 Pojezd ABB IRBT 4003

3.4.6 Kovosvit MCV 754 QUICK a SPM 16

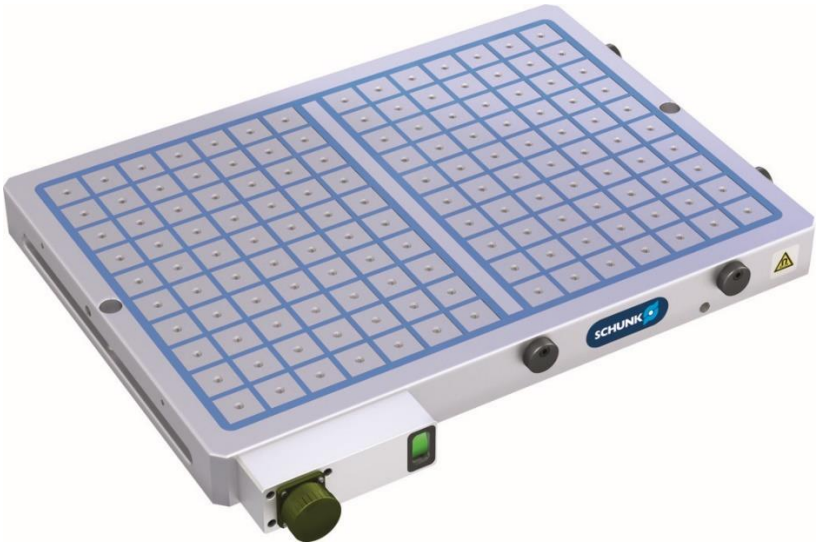
Stroj Kovosvit SPM 16 (Obr. 3-11a) je CNC soustruh. Protože tento soustruh je staršího data výroby, není, bez větších zásahu převážně do jeho řídicího systému, možná jeho implementace do automatického výrobního systému. Je zde však uveden, jelikož je nedílnou součástí zástavby zadaného výrobního systému.

Stroj Kovosvit MCV 754 QUICK (Obr. 3-11b) je tříosá CNC vertikální frézka umožňující širokou škálu obráběcích operací, díky značnému výkonu vřetene 9 kW dosahující až 10 000 ot./min [26].

Obr. 3-11 a) Soustruh SPM 16 b) Tříosá CNC frézka MCV 754 QUICK [26]

3.4.7 Magnetický upínač Schunk MFRS-A1-032

Aby bylo možné obsluhovat stroj MCV 754 QUICK v plně automatickém režimu, je nutné jej vybavit automatickým upínačem. Tomuto vyhovuje magnetický upínač firmy Schunk MFRS-A1-032 (Obr. 3-12), dosahující značných upínacích sil až 54 kN při rozměrech upínací plochy 315 x 315 mm [27].



Obr. 3-12 Magnetický upínač Schunk MFRS-A1-032 [27]

3.4.8 Bezpečnost výrobního systému

Při návrhu automatického výrobního systému je nutné řešit jeho bezpečnost. Bezpečnost by měla být posouzena na základě analýzy rizik, která zprostředkovává všechny informace o vzniku možných rizik a o rizicích zbytkových. Analýza rizik nebyla cílem této diplomové práce, protože při těchto rozměrech by její komplexnost vystačila na samostatnou diplomovou práci. Avšak prvky bezpečnosti jsou součásti zástavby výrobního systému a bylo vhodné je zmínit.

U obráběcích strojů MCV 754 QUICK a SPM 16 se předpokládá jejich vlastní plnění bezpečnosti, blokováním spuštění operací při otevřených dveřích a rovněž jsou vybaveny STOP tlačítkem pro nouzové zastavení [28].

Další bezpečnostní prvky je možno rozdělit na dvě skupiny, a to fyzické bezpečnostní prvky a softwarové bezpečnostní prvky. První zmíněna skupina zamezuje obsluze, respektive člověku, vstoupit do pracovního prostoru robotu. O toto se stará oplocení firmy ABB Quick-guard, které je navržené pro potřeby daného výrobního systému. Toto oplocení je vybaveno světelnými závorami ABB Orion1-4-30-090-E s roztečí paprsků 30 mm, umístěnými při zakládací stanici a světelnými závorami ABB Orion3-4-K2C-120-B, které zabraňují vstupu člověka do pracovního prostoru robotu při robotické obsluze stroje MCV 754 QUICK. Oplocení je také vybaveno třemi dveřními vstupy, jejichž otevíraní je zajištěno pomocí bezpečnostních dveřních zámku ABB Knox, které neumožní otevření dveří, pokud je robot v pohybu. Pro ovládání dveří a nouzové zastavení výrobního systému je oplocení vybaveno čtyřmi ovládacími tlačítkovými krabicemi ABB Smile 41 EWWWP u každých dveří a dále tlačítkem nouzového zastavení na řídicím systému robotu. Tyto prvky však zastavují pouze robot a jeho periferie. Pro zastavení celého výrobního systému včetně obráběcích strojů bude použito tlačítko nouzového zastavení umístěné na případném ovládacím panelu [28].

Za druhou skupinu lze považovat softwarové bezpečnostní prvky, které jsou implementovány v řídicím systému robotu, aby nedošlo k proražení oplocení a tím pádem ohrožení osob a majetku nacházejících se v blízkosti výrobního systému. Toto je zajištěno pomocí rozdělení pracovního prostoru robotu do zón, které vymezují pracovní prostor robotu v automatickém režimu, což je možné díky bezpečnostní funkcí SafeMove 2. Vzdálenost mezi oplocením a jakoukoliv části robotu nesmí být nikdy menší než 200 mm [28].

3.5 Požadované operace a senzory (signály)

V zadaném výrobním systému je požadavek na několik robotických úloh. První úlohou je Obsluha stroje MCV 754 Quick, dále úloha Manipulační operace s kostkami a poslední jsou robotické operace výměny nástrojů. Důležité je zmínit, že každá úloha se skládá z několika robotických operací, které jsou vytvořeny v PS a také v reálném programu robotu, a ty jsou následně volány PLC programem k vykonání.

Jelikož jsou simulace řízeny událostně, což znamená na základě signálů, je nutné tyto signály zprostředkovat pomocí vhodné soustavy senzorů, díky tomu jsou pak v PLC programu vytvořeny podmínky, při jejichž splnění jsou volány požadované robotické operace.

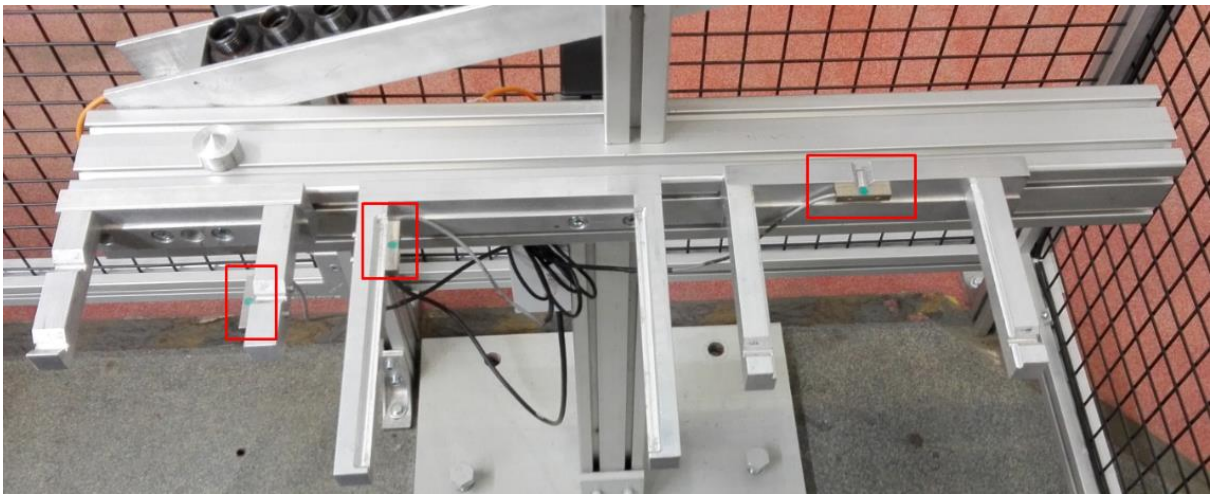
3.5.1 Výměna nástrojů

Aby bylo možné vykonávat požadované manipulační operace obsluhy stroje nebo manipulaci s kostkami, je nutné, aby byl robot osazen potřebným koncovým efektorem. V případě tří možných nástrojů to znamená dvanáct možných operací výměny:

1. Bez nástroje → Nástroj s chapadlem Schunk
2. Bez nástroje →Nástroj s chapadlem SMC
3. Bez nástroje → Nástroj s vakuovým chapadlem
4. Nástroj s chapadlem Schunk → Bez nástroje
5. Nástroj s chapadlem Schunk→ Nástroj s chapadlem SMC
6. Nástroj s chapadlem Schunk → Nástroj s vakuovým chapadlem
7. Nástroj s chapadlem SMC → Bez nástroje
8. Nástroj s chapadlem SMC → Nástroj s chapadlem Schunk
9. Nástroj s chapadlem SMC → Nástroj s vakuovým chapadlem
10. Nástroj s vakuovým chapadlem → Bez nástroje
11. Nástroj s vakuovým chapadlem → Nástroj s chapadlem Schunk
12. Nástroj s vakuovým chapadlem → Nástroj s chapadlem SMC

I když v rámci požadovaných úloh nejsou nutné všechny tyto výměny, je vhodné je vytvořit, jelikož v případě rozšíření o další operace či nástroje bude jejich editace značně jednoduchá.

Při výměně nástrojů jsou důležité signály udávající kompletní přehled o nástrojích a jejich přítomnosti ve stojanu, což je zajištěno třemi indukčními senzory (Obr. 3-13). Na přírubě robotu by měly být umístěny rovněž tři indukční sensory informující o tom, který nástroj je na robotu zrovna upnut.

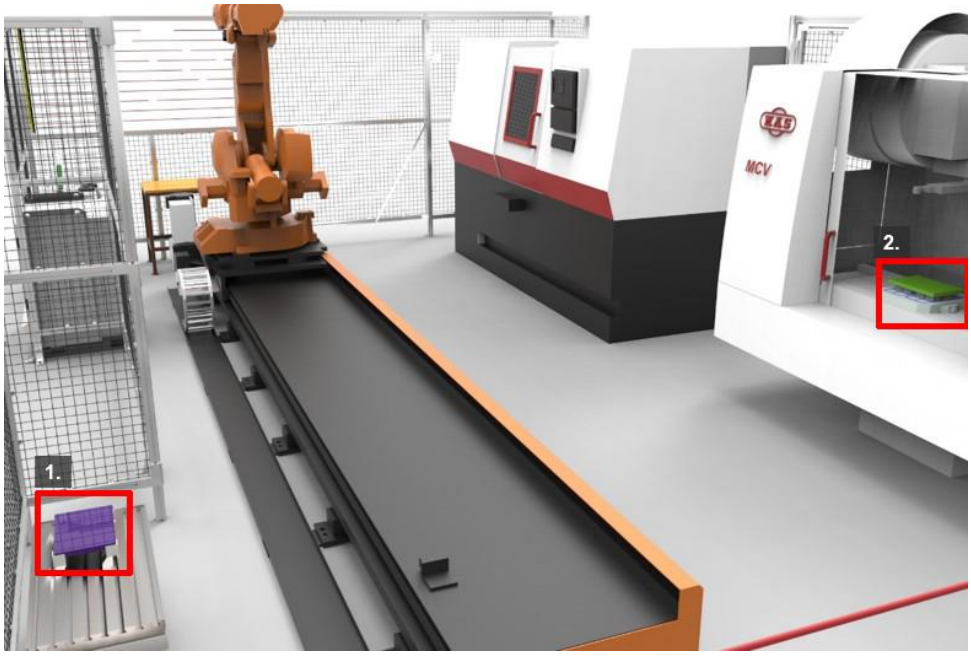


Obr. 3-13 Indukční senzory na stojanu pro nástroje

3.5.2 Obsluha stroje MCV 754 Quick

Úloha Obsluha stroje MCV 754 Quick je zajištěna dvěma robotickými operacemi, kdy první operací je založení polotovaru do stroje (Obr. 3-14 bod 2.) a druhou operací je vyjmutí a odložení obrobku. Polotovar je brán a následně jako obrobek odkládán na stejné místo (Obr. 3-14 bod 1.).

Při obsluze jsou důležité signály zprostředkovávající informace o přítomností polotovaru a o volnosti odkládací plochy. Jelikož je odkládací plocha jednotná pro polotovar i obrobek, je toto zajištěno pomocí jednoho indukčního sensoru. Dále je důležitý signál, který dává povel k zavření a otevření dveří u stroje, a signály dávající stroji povel k započnutí obráběcího cyklu a následně signál ukončení obráběcího cyklu vycházející z řídicího systému stroje, který je podmínkou pro provedení vyjmutí obrobku.

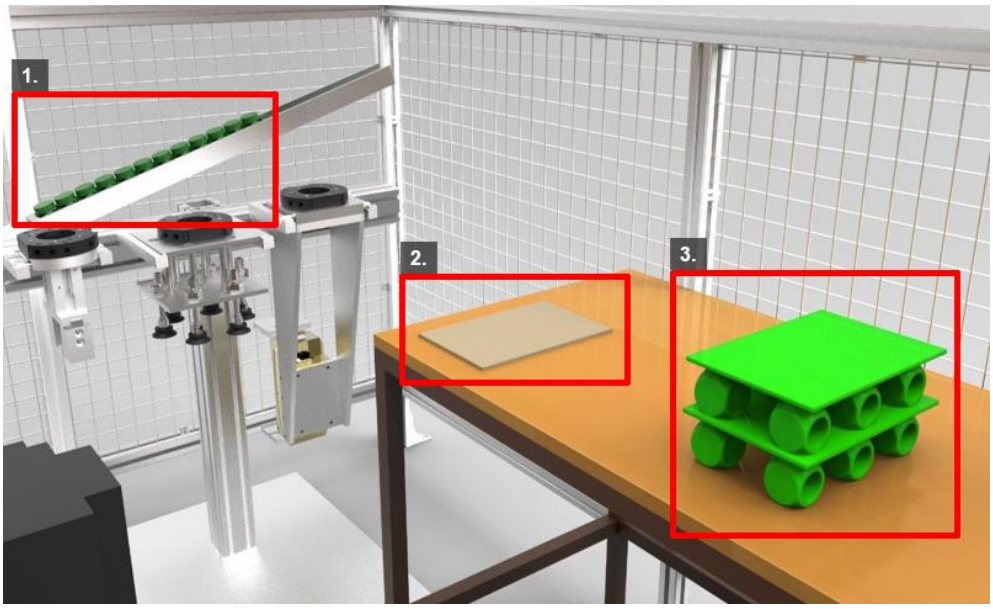


Obr. 3-14 Grafická reprezentace úlohy Obsluha stroje

3.5.3 Manipulační operace s kostkami

Úloha Manipulační operace s kostkami sestává z celkem čtrnácti robotických operací, kde dvanáct operací je manipulace s kostkami a zbylé dvě manipulace s kartónem. V první části je pomocí nástroje s chapadlem SMC postupně uloženo šest kostek ve dvou řadách, následně je pomocí nástroje s vakuovým chapadlem na kostky umístěn kartónový blok. Poté je vše znovu zopakováno s tím, že kostky a kartón jsou umístěny v druhé vrstvě na vrstvu předchozí. Kostky jsou odebírány ze zásobního skluzu (Obr. 3-15bod 1.) a kartóny ze stolu (Obr. 3-15bod 2.). Celou sestavu je možno vidět na obrázku (Obr. 3-15 bod 3.).

Pro tuto úlohu jsou nutné pouze signály informující o přítomnosti kostky v zásobním skluzu a o přítomnosti kartonového bloku na stole. Předpokládá se, že místo, kde mají být kostky složeny, bude vždy uvolněné, což zajistí případná obsluha. V případě, kdy by mělo být využito senzorů, které by měly zprostředkovat informace o stavu plochy, kde mají být kostky složeny, by bylo nutno celkem šest senzorů na místech, umístění první vrstvy kostek, případně by muselo být užito nějakého skeneru nebo strojového vidění, což by bylo pro tuto ukázkovou aplikaci podstatně ekonomicky nevhodné.



Obr. 3-15 Grafická reprezentace úlohy Manipulační operace s kostkami

4 PRAKTICKÁ ČÁST

V této části práce je popsán postup vedoucí k virtuálnímu zprovoznění. V první řadě bylo nutné vytvořit 3D model zprovozňovaného výrobního systému se všemi kinematikami, operacemi, logickými bloky a materiálovými toky. Následně bylo nutné navrhnout a naprogramovat PLC řídicí tento 3D model. Nad rámec práce byla rovněž vytvořena vizualizace ovládacího panelu (HMI) sloužící k ovládání naprogramovaného PLC. PLC, HMI a 3D model byly pak spojeny skrze softwarový emulátor PLCSIM Advanced 2.0, díky čemu bylo následně provedeno virtuální zprovoznění.

4.1 Postup tvorby simulací v prostředí Process Simulate

V této kapitole je popsaná metodika tvorby simulace v softwaru Process Simulate. Zde nebylo možné popsat veškeré funkce a možnosti PS v rámci této práce, jelikož použitý software je velmi komplexní, obsahující širokou škálu možností tvorby simulací, operací atd. Proto v této kapitole bylo přistoupeno k zjednodušenému popisu metodiky tvorby simulace zadaného výrobního systému nutné k virtuálnímu zprovoznění.

Pro tvorbu požadované simulace byla k dispozici dříve zpracována simulace v PS. Tato simulace ale ne zcela odpovídala, převážně zahrnutými operacemi, reálnému výrobnímu systému, na kterém mělo být později provedeno virtuální zprovoznění. Jelikož po podrobnějším prozkoumání bylo zjištěno, že změny v simulaci by byly natolik markantní, že by mohlo docházet později ke konfliktům různých souborů, případně k chybám v definici komponent, či nemožnosti upravovat operace, bylo rozhodnuto o znovu vytvoření simulace se všemi prvky potřebnými pro následné virtuální zprovoznění s tím, že byl kladen důraz na pozdější možnost rozšíření a editaci této nové simulace.

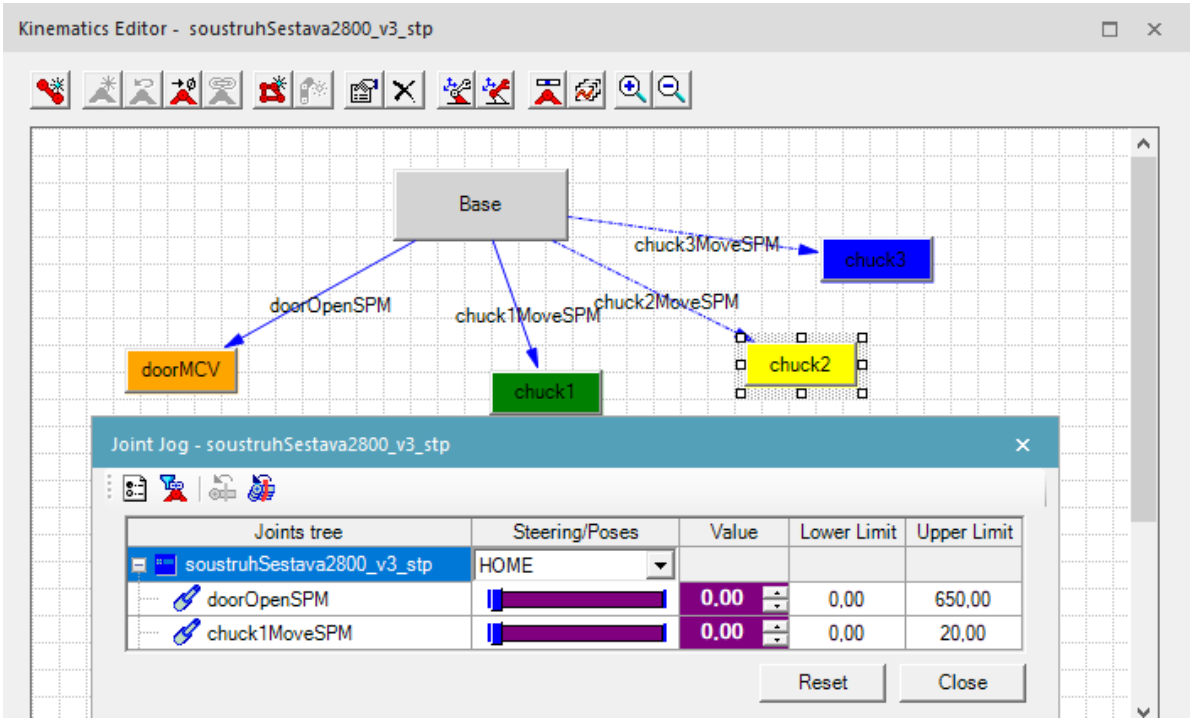
Prvotní simulace byla ovšem přínosná, jelikož její rozložení některých zařízení korespondovalo s realitou. Jedná se o soustruh SPM 16, frézku MCV 754 Quick, robot ABB IRB 4400/60 s pojezdem IRBT 4003 a ochranné ploty s bezpečnostními prvky. Zdrojové soubory součásti v dodané simulaci, které jsou použity v nové simulaci byly překopírovány do nového pracovního adresáře a s těmi bylo následně pracováno.

4.1.1 Modely použitých zařízení

V dodané simulaci byly použity nepřesné nebo zjednodušené modely některých zařízení. Aby bylo možné vytvořit věrnou reprezentaci rozložení výrobního systému, bylo nutné tyto modely vytvořit nebo upravit stávající. Jednalo se především o model soustruhu SPM 16, magnetický upínač MFRS-A1-032 a nástroje robotu.

Model soustruhu Kovosvit SPM 16

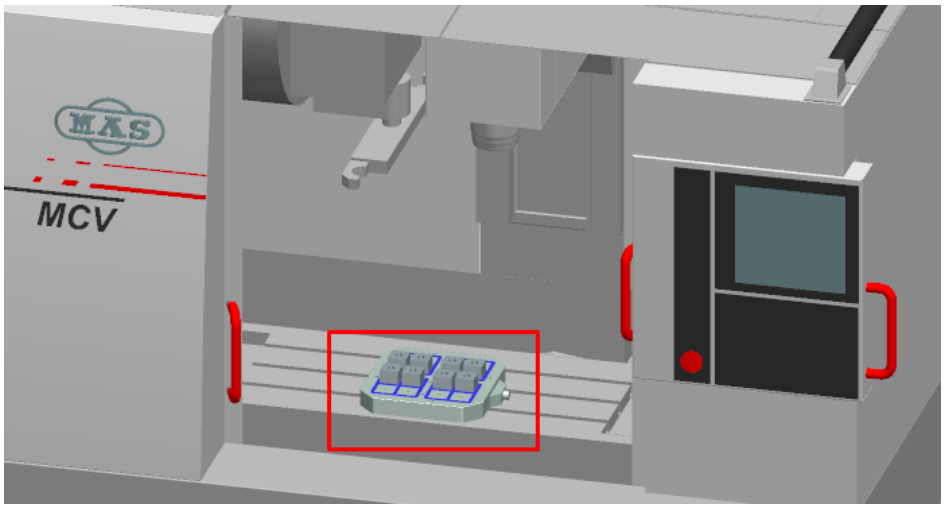
V dodané simulaci byl použit zjednodušený model soustruhu SPM 16, ten bylo nutné vytvořit jako věrnější model reálného soustruhu s možností otevíraní dveří a pohybu čelistí sklíčidla simulující upnutí obrobku. I když k soustruhu nebyly v této diplomové práci vztažené žádné operace, v případě požadavku na rozšíření a tvorbu následných simulací zahrnujících tento soustruh, je vše připraveno. Soustruh byl vytvořen v softwaru Inventor Professional 2018, následně byl nahrán do PS a byla mu vytvořena základní kinematická struktura reprezentující zavírání dveří a pohyb kamenů sklíčidla (Obr. 4-1).



Obr. 4-1 Kinematická struktura soustruhu SPM v PS

Model magnetického upínače Schunk MFRS-A1-032

Další prvek, který byl v dodané simulaci reprezentován značně zjednodušeným modelem, byl magnetický upínač Schunk MFRS-A1-032. Ten bylo nutné vymodelovat ve věrnější podobě reálného upínače. Model byl vytvořen v prostředí Siemens NX 12.0 a následně byl nahrán do PS a umístěn na stůl stroje MCV (Obr. 4-2).

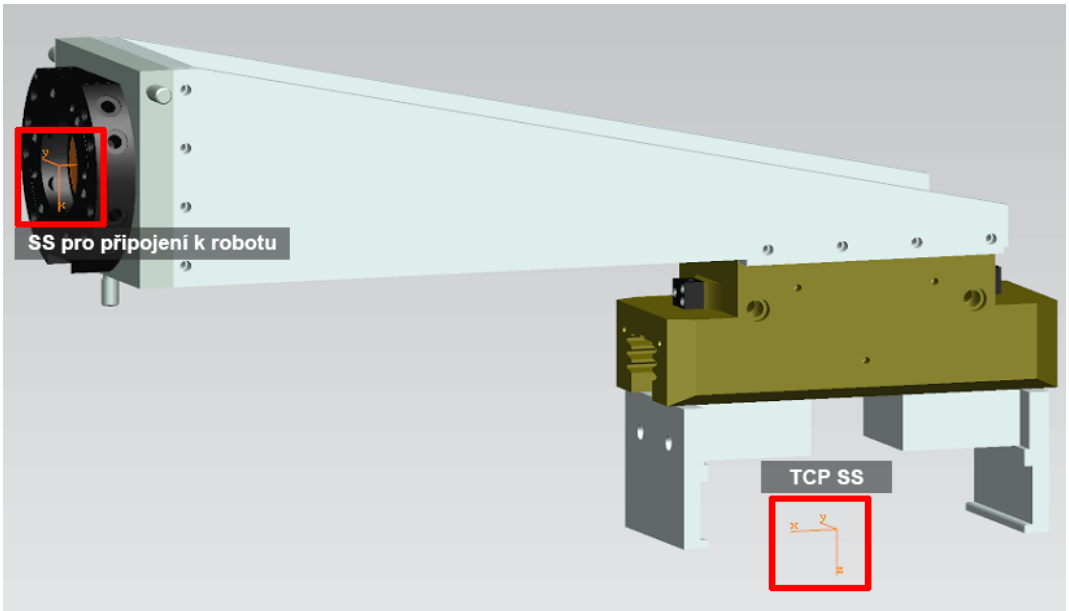


Obr. 4-2 Magnetický upínač umístěn ve stroji MCV

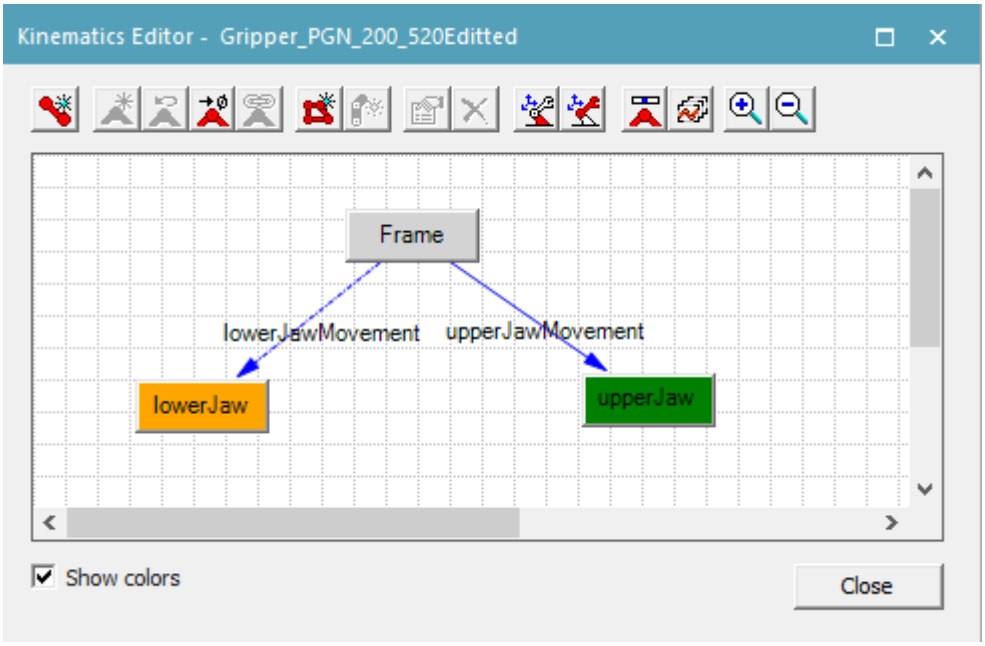
Modely nástroje s chapadlem Schunk

Jelikož v době tvorby poskytnuté simulace nebyl jasně definován nástroj, který by měl provádět obsluhu stroje MCV 754 Quick, bylo v ní použito velmi nekvalitního modelu. Po konzultaci byl dodán nový model nástroje, který bylo nutné osadit adaptérem pro výměnu nástrojů. Takto vytvořený model byl pak nahrán do PS (Obr. 4-3), byla mu vytvořena

adekvátní kinematika (Obr. 4-4), referenční souřadný systém TCP a souřadný systém pro připojení nástroje k robotu.



Obr. 4-3 Model nástroje s chapadlem Schunk v PS



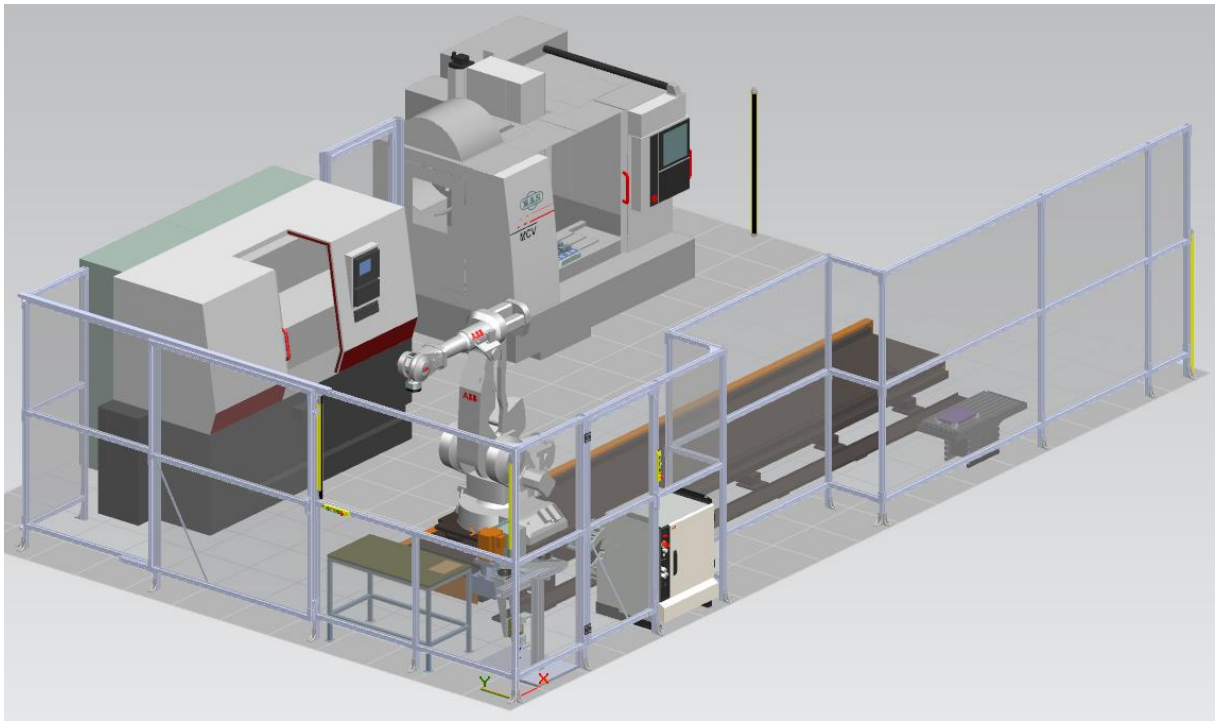
Obr. 4-4 Kinematická struktura nástroje s chapadlem Schunk v PS

Další modely

V rámci výrobního systému bylo nutné vytvořit dále model stolu, kde je prováděna úloha Manipulační operace s kostkami a model starého stolu frézky, který je použit jako plocha pro odkládání obrobku při robotické obsluze stroje MCV 754 Quick. Taktéž byl upraven model oplocení, kde byly odebrány všechny vyplňující pletiva, která byla nahrazená průhlednými bloky. Díky tomu jsou simulace přehlednější a výpočetně méně náročné.

4.1.2 Rozložení výrobního systému

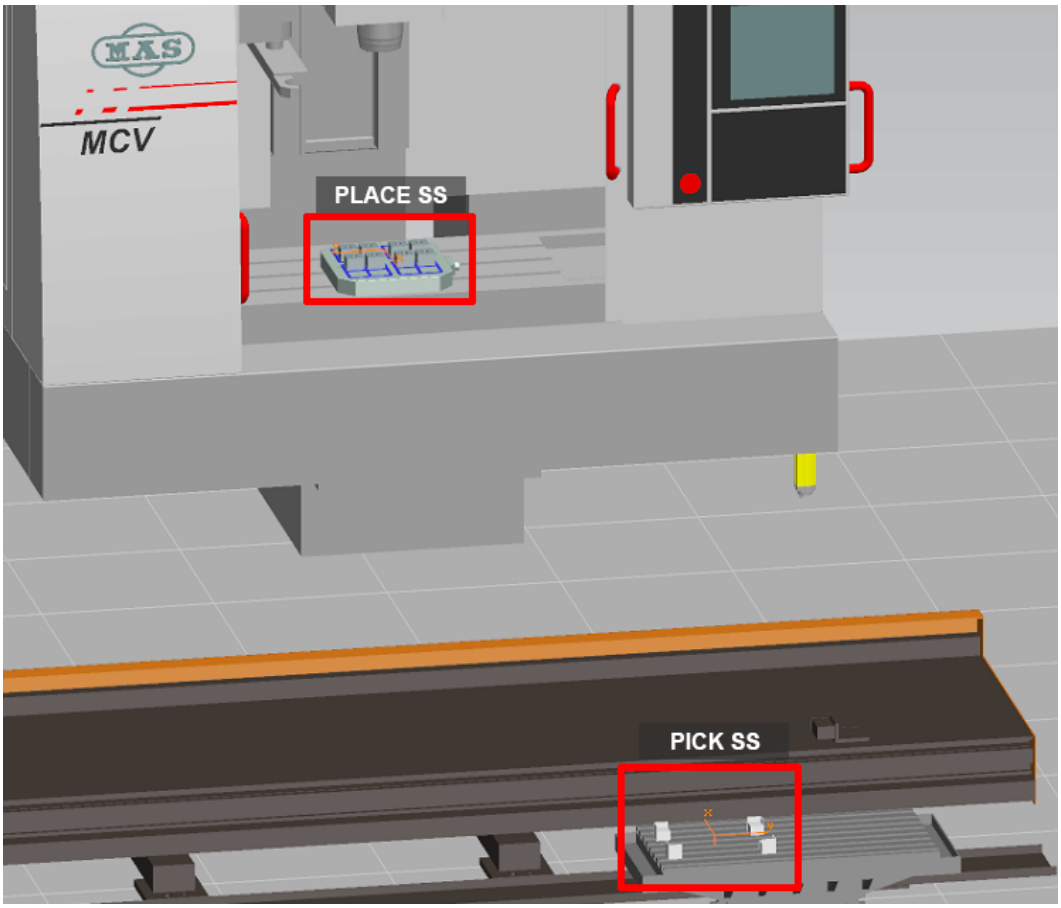
Po nakopírování všech dílčích modelů do pracovního adresáře a vytvoření potřebných kinematik, byly modely importovány do PS. Po importování bylo nutné vytvořit soustavu charakteristických souřadných systémů, do kterých pak byly přesunuty jednotlivé modely, čímž bylo docíleno rozložení korespondující s rozložením reálného výrobního systému (Obr. 4-5).



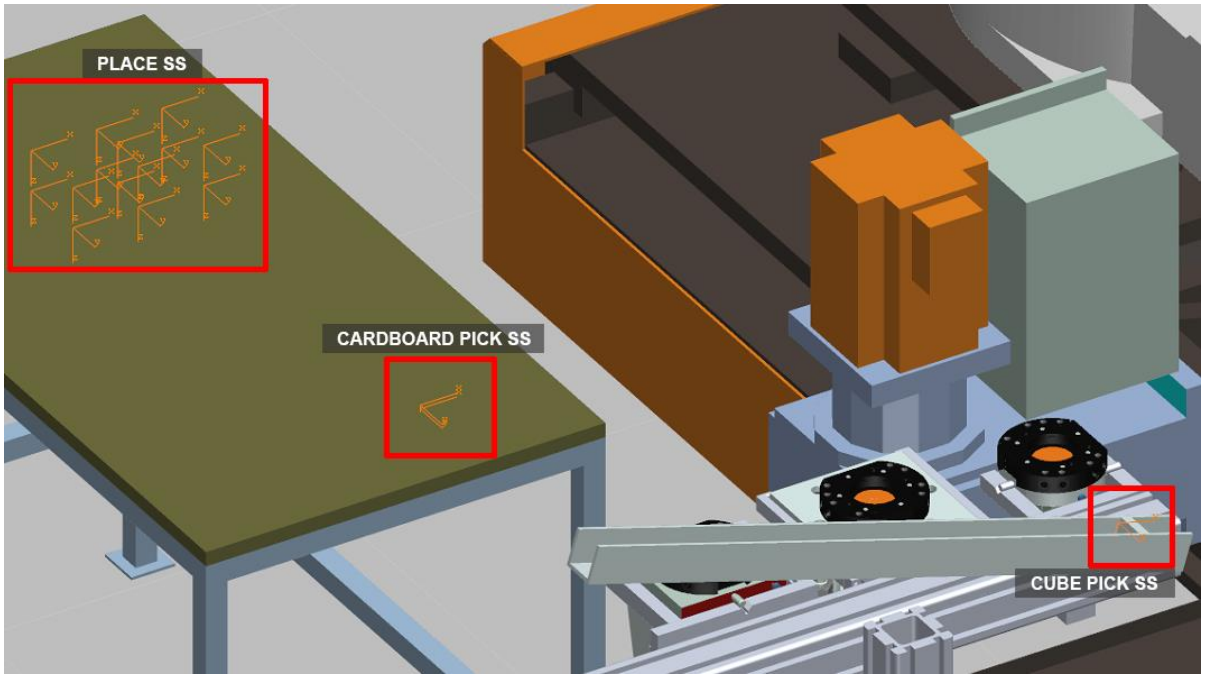
Obr. 4-5 Model kompletního výrobního systému v PS

4.1.3 Tvorba požadovaných operací

Aby bylo možné vytvořit požadované operace, bylo nutné vytvořit několik souřadných systému odpovídající místům, kde mají být manipulované součásti umístěny. V případě úlohy Obsluhy stroje MCV 754 Quick se jednalo o souřadný systém umístěný na stolu staré frézky a na magnetickém upínači v útrobách samotného stroje (Obr. 4-6). U úlohy Manipulační operace s kostkami bylo nutno vytvořit celkem šestnáct souřadných systému. První byl vytvořen jeden souřadný systém ve skluzu reprezentující místo odebrání kostky a jeden odpovídající místu, kde se bude ve výchozí pozici nacházet kartón. Dále bylo vytvořeno celkem dvanáct souřadných systému pro dvě vrstvy kostek a dva souřadné systémy pro uložení kartónů (Obr. 4-7).



Obr. 4-6Souřadné systémy nutné pro Obsluhu stroje



Obr. 4-7 Souřadné systémy nutné pro Manipulaci s kostkami

Po vytvoření všech potřebných souřadných systémů byly vytvořeny manipulační (Pick and Place) operace tyto souřadné systémy spojující. Pro úlohu Obsluha stroje MCV 754 Quick bylo nutné sestavit dvě robotické operace (viz. Kapitola **3.5.2**). Pro úlohu Manipulace s kostkami bylo vytvořeno celkem čtrnáct manipulačních (Pick and Place) robotických operací (viz. Kapitola **3.5.3**). Posledním krokem bylo vytvoření operací výměny nástrojů (viz. Kapitola **3.5.1**).

Důležitým krokem bylo vložení operace do programu robotu a přiřadit jim požadovaná čísla cesty „*#Path*“, jelikož na základě těchto hodnot jsou následně tyto operace volány z řídícího programu nadřazeného PLC. Operace (Obr. 4-8) jsou uspořádaný tak, aby při spuštění funkce „*Auto Teach*“ proběhly v návaznosti jedna na druhou, a tím se tak zapsaly do programu robotu. Tento program lze po úpravě nahrát do reálného řídícího systému robotu.

Paths & Locations	Path #
Main	
ToolChangeFlangeToSchunk	1
PartInsert	101
PartTakeOut	102
ToolChange_SchunkToVac	13
ToolChange_VacToSchunk	31
ToolChange_SchunkToSMC	12
ToolChange_SMCToSchunk	21
ToolChange_SchunkToFlange	10
ToolChangeFlangeToSMC	2
Cubes_1_1_1	201
Cubes_2_1_1	202
Cubes_3_1_1	203
Cubes_1_2_1	204
Cubes_2_2_1	205
Cubes_3_2_1	206
Cubes_1_1_2	207
Cubes_2_1_2	208
Cubes_3_1_2	209
Cubes_1_2_2	210
Cubes_2_2_2	211
Cubes_3_2_2	212
ToolChange_SMCToVac	23
Cubes_Kart_1	250
Cubes_Kart_2	251
ToolChange_VacToSMC	32
ToolChange_SMCToFlange	20
ToolChangeFlangeToVac	3
ToolChange_VacToFlange	30

Obr. 4-8 Operace a jejich čísla (#Path) zahrnuty v programu robotu

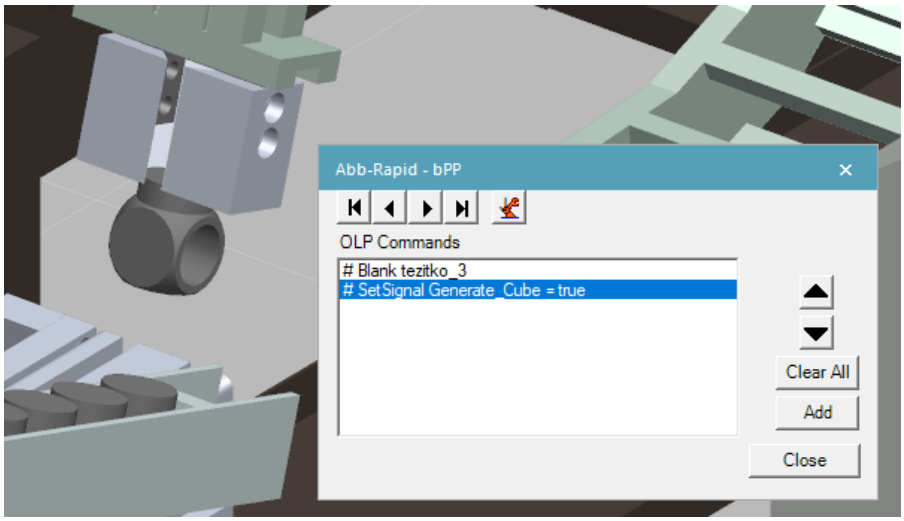
4.1.4 Materiálový tok

Díky tomu, že je simulace událostně řízená, bylo nutné definovat materiálový tok jednotlivých manipulovaných součástí, jelikož v průběhu simulace se tyto součásti objevují pouze za splnění určitých podmínek.

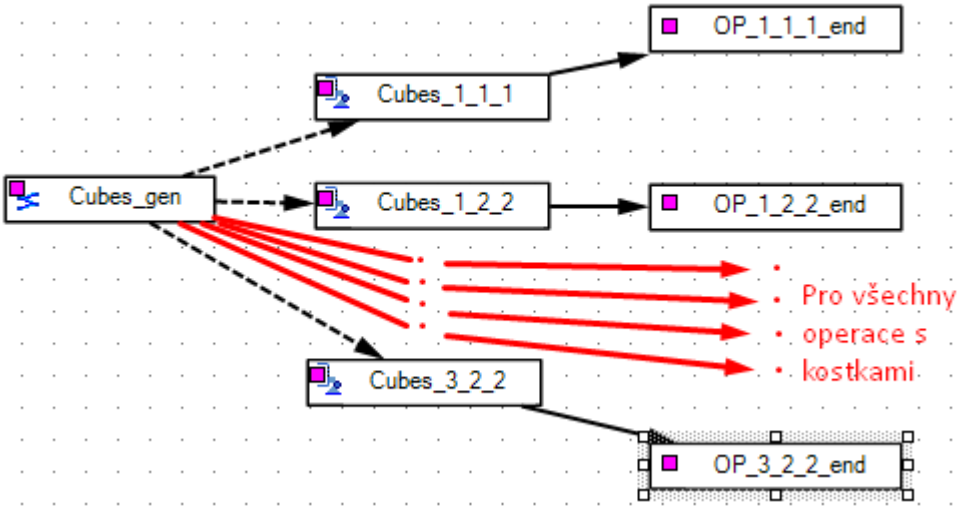
Při obsluze stroje MCV 754 Quick je polotovar odebírán ze stejného místa, na které je následně odkládán obrobek. To znamená, že v průběhu této operace nemůže dojít k „objevení“ dalšího polotovaru, proto zde není nutné vytvářet materiálový tok, protože v celé simulaci bude vždy existovat pouze jeden tento polotovar/obrobek.

Avšak při operaci manipulace s kostkami bylo využití materiálového toku nadmíru vhodné, jelikož je možné tyto kostky postupně zobrazovat v simulaci a na jejím konci je

dvanáct jejich grafických reprezentací (instancí). Kostky umístěné ve skluzu se postupně skrývají, ale k manipulaci dochází pouze s jednou objevující se kostkou na základě materiálového toku. Kostka je vygenerována pouze v případě zavolání úlohy Manipulace s kostkami, nebo když je signál „*Generate_Cube*“ roven „*TRUE*“. Tento signál je uveden do stavu „*TRUE*“, díky OLP příkazu „*SetSignal*“, který je vybaven, jakmile robot projde bodem trajektorie po odebrání kostky ze skluzu (Obr. 4-9). Výsledný materiálový tok ve zkrácené podobě je možno vidět na obrázku (Obr. 4-10).



Obr. 4-9 Nastavení signálu pro generování další kostky

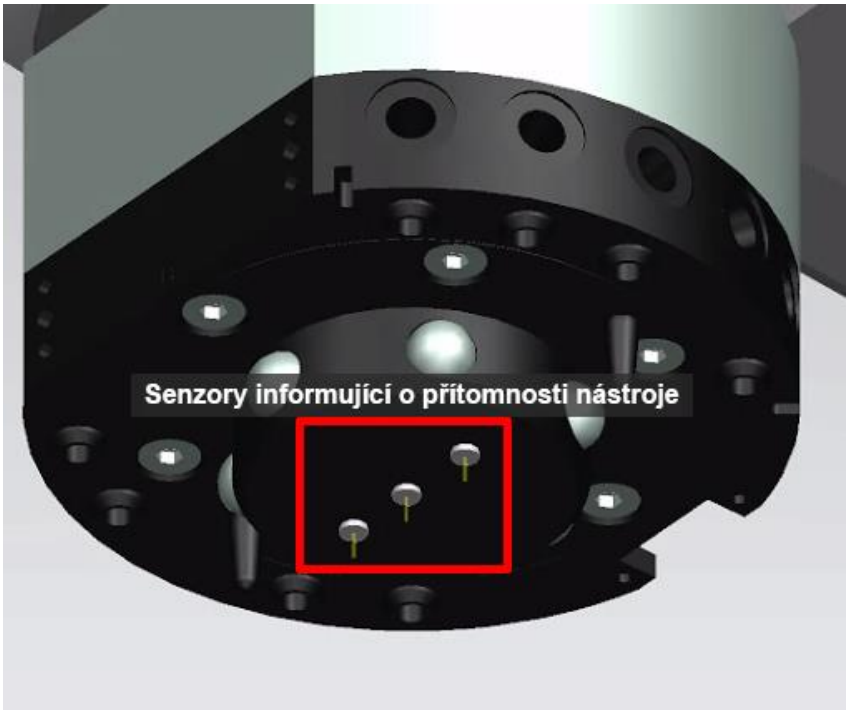


Obr. 4-10 Výsledný materiálový tok (zkrácená verze)

4.1.5 Senzory a signály

Aby bylo možné řídit simulaci v PS pomocí externě připojeného PLC, bylo nutné vytvořit signálovou strukturu poskytující potřebné vstupní a výstupní signály řídicímu PLC. Jedná se o signály robotu (Obr. 4-12), signály senzorů (Obr. 4-13), případně signály logických bloků. Pro robot jsou vytvořeny základní signály. Dále bylo vytvořeno několik *„Photoelectric Sensor“*, které snímají nástroje na přírubě robotu (Obr. 4-11). Obdobně byly vytvořeny další senzory snímající přítomnost nástrojů ve stojanu a přítomnost součásti nutných k vykonání

manipulačních operací, jako je polotovar při obsluze stroje, nebo kostky a kartóny při manipulaci s kostkami. Tyto signály byly namapované na vstupy/výstupy řídicího PLC.



Obr. 4-11 Zobrazení virtuálních sensorů na přírubě robotu

Robot Signals - " irb4400_60_rev02 "

PLC Signal Name	Robot Signal Name	I/O	Signal Function	HW T...	Address
irb4400_60_rev02_startProgram	startProgram	Q	Starting Program	BOOL	No Address
irb4400_60_rev02_programNumber	programNumber	Q	Program Number	BYTE	No Address
irb4400_60_rev02_emergencyStop	emergencyStop	Q	Program Emergency Stop	BOOL	No Address
irb4400_60_rev02_programPause	programPause	Q	Program Pause	BOOL	No Address
irb4400_60_rev02_programEnded	programEnded	I	Ending Program	BOOL	No Address
irb4400_60_rev02_mirrorProgramNumber	mirrorProgramNumber	I	Mirror Program Number	BYTE	No Address
irb4400_60_rev02_errorProgramNumber	errorProgramNumber	I	Error Program Number	BOOL	No Address
irb4400_60_rev02_robotReady	robotReady	I	Robot Ready	BOOL	No Address
irb4400_60_rev02_at_HOME	HOME	I	Pose Signal	BOOL	No Address

OK Cancel Apply

Obr. 4-12 Základní signály robotu

Signal Viewer

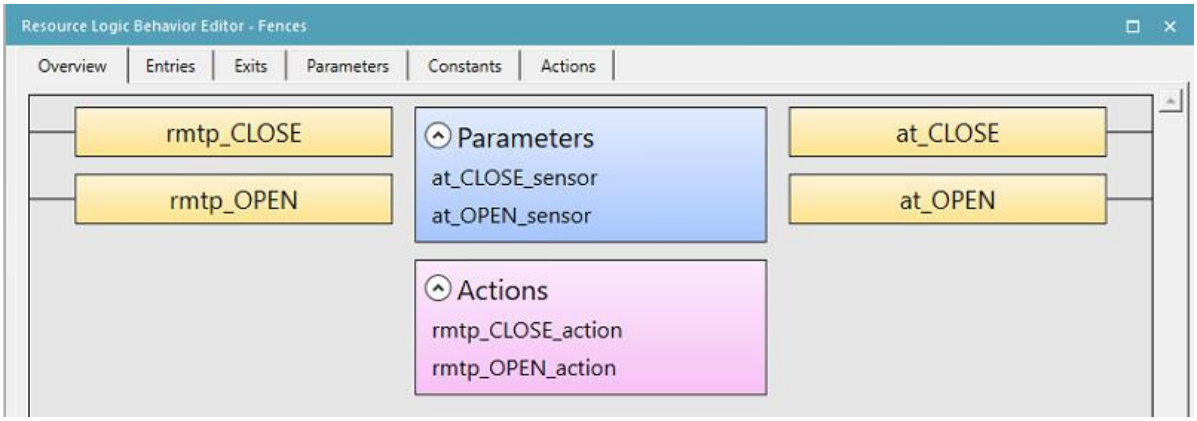
Signal Name	Memory	Type	Address	IEC Format	PLC Connection	Resource
Type here to filter			*Type here to filter*	*Type here to fil*		
CART_SENS	☐	BOOL	0.0	I0.0	☑	_cartonSensor
CUB_SENS	☐	BOOL	0.1	I0.1	☑	_cubesSensor
PNP_SENS	☐	BOOL	0.2	I0.2	☑	HandlingPartSensor
SCHUNK_SENS_F	☐	BOOL	1.0	I1.0	☑	SchunkSensorFlange
SMC_SENS_F	☐	BOOL	1.2	I1.2	☑	SMCSensorFlange
VAC_SENS_F	☐	BOOL	1.3	I1.3	☑	VacSensorFlange
SCHUNK_SENS_S	☐	BOOL	2.0	I2.0	☑	_schunkInStand
SMC_SENS_S	☐	BOOL	2.1	I2.1	☑	_SMCInStand
VAC_SENS_S	☐	BOOL	2.2	I2.2	☑	_vacInStand

Obr. 4-13 Signály senzorů

4.1.6 Logické bloky

Logické bloky (dále LB) v PS slouží k vytvoření logické struktury, která je nezávislá na řídicím PLC. Vnitřní logika LB je sestavena na základě logických podmínek, kdy na základě vstupních signálu „*Entries*“ jsou ovlivněny signály výstupní „*Exits*“. Od PS verze 14 je možné v LB používat i funkce jako Set-Reset, časovače, čítače apod. LB je taktéž možno vytvořit pro jednotlivá zařízení.

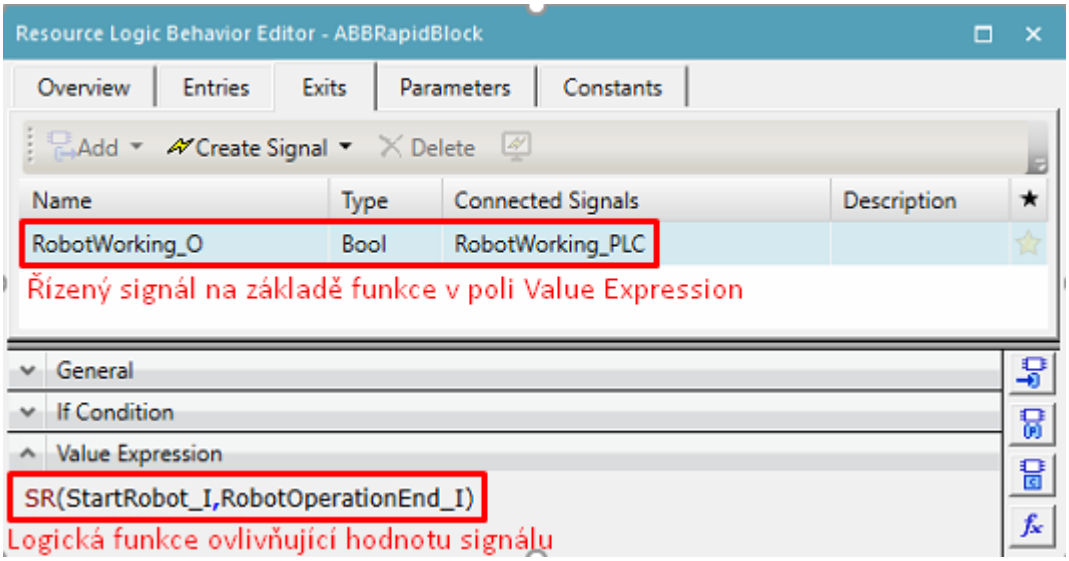
Pro zařízení se LB vytvářejí automaticky příkazem „*Create LB Pose Action And Sensors*“. Příklad takto vytvořeného LB zařízení (plotů) je možné vidět na obrázku (Obr. 4-14), kde jako vstupní signály jsou použity „*rmtp_CLOSE*“ a „*rmtp_OPEN*“, které jsou svázaný s akcemi „*rmtp_CLOSE_action*“ a „*rmtp_OPEN_action*“. To znamená, že v případě, kdy jsou vstupní signály v hodnotě „*TRUE*“, je volána příslušná akce zavřít nebo otevřít dveře. Toho by bylo možné využít v případě, kdy by dveře byly automatizované a signál pro zavření, případně otevření, by posílalo řídicí PLC. Dále jsou v bloku parametry (senzory) „*at_CLOSE_sensor*“ a „*at_OPEN_sensor*“, zprostředkovávající informaci o tom, v jaké se zrovna nacházejí dveře poloze. Na základě těchto parametru jsou pak vystaveny výstupní proměnné „*at_CLOSE*“ a „*at_OPEN*“. Proměnná, respektive signál „*at_CLOSE*“ poskytuje informaci, zda jsou veškeré dveře uzavřeny, což je velmi důležitá podmínka pro splnění bezpečnostního hlediska řídicího PLC před voláním operace.



Obr. 4-14 LB pro ploty

Dále byl vytvořen LB „*ABBRapidBlock*“, který zprostředkovává řídicímu PLC informaci o tom, zda robot zrovna pracuje. Jak je možno vidět na obrázku (Obr. 4-15), v části „*ValueExpression*“, je použita Set-Reset funkce, která je aktivována v případě vybavení

vstupního signálu „*StartRobot_I*“. Funkce je resetovaná v případě vybavení vstupního signálu „*RobotOperationEnd_I*“.



Obr. 4-15 LB reprezentující práci robotu

4.2 Program PLC

Jako řídící PLC pro zadaný výrobní systém bylo použito PLC firmy Simens Simatic S7 1512 - PN/C. Výsledný program byl sestaven v prostředí TIA Portal V15. Hlavní cyklická rutina programu je psána v jazyce LAD (LadderLogic) a skládá se z několika funkcí (dále FC), které jsou psány v jazycích LAD a SCL (Structured Control Text). Ty syntakticky vychází z jazyku Pascal. Důležité části programu jsou popsány pomocí vývojových digramů, které graficky zobrazuji tok programu, a jsou tak mnohem přehlednější a názornější než samotné kódy.

4.2.1 Vizualizace ovládacího panelu

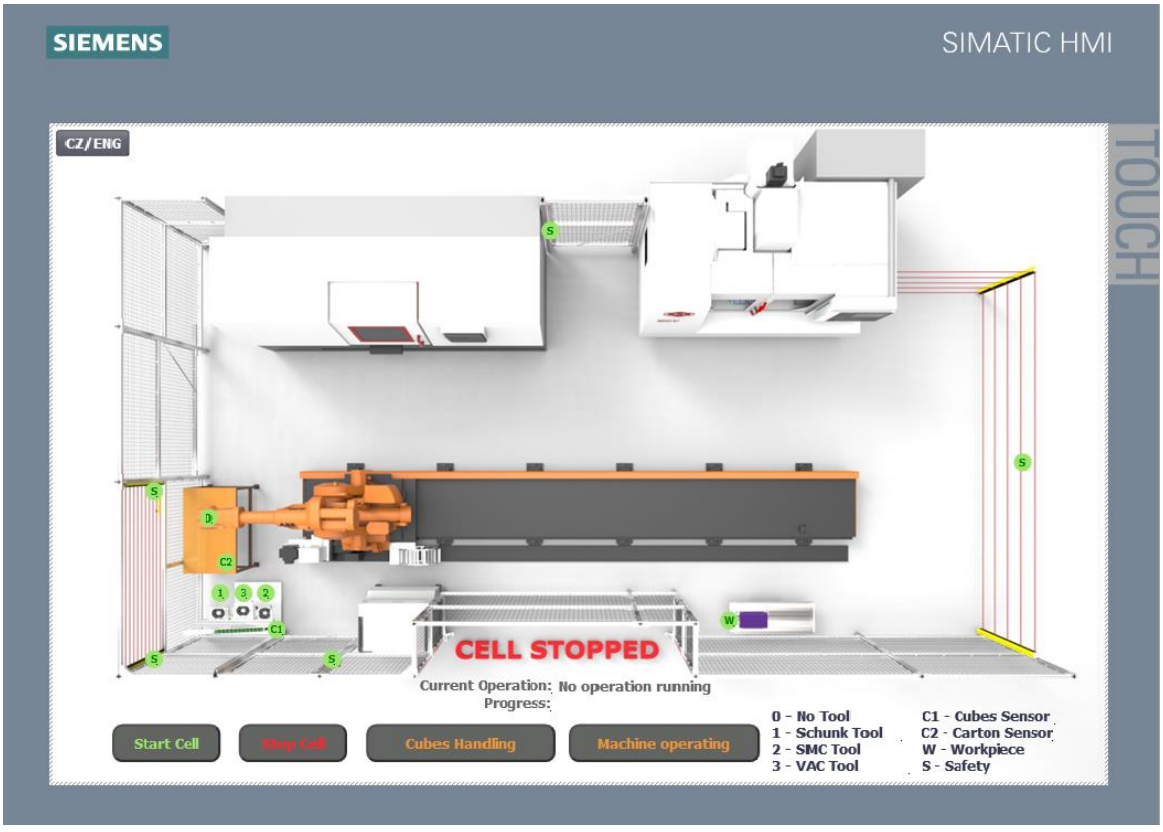
Nad rámec práce byla vytvořena zjednodušená vizualizace ovládacího panelu sloužícího k ovládání řídicího PLC (Obr. 4-16), tzv. HMI, jelikož je pravděpodobné budoucí použití takového panelu pro ovládání celého výrobního systému. Nebylo možné předurčit, o jaký panel se bude jednat, proto byla vizualizace navržena pro dvanáctipalcový panel střední třídy KTP1200 řady Basic ze série SIMATIC HMI firmy Siemens. Této vizualizace bylo rovněž použito k ovládání PLC, a tím tak řízení a ověření virtuálního modelu výrobního systému.

Ve vizualizaci je prozatím použitá jedna plocha, kde je zobrazen pohled shora na výrobní systém, ve kterém jsou zasazeny zelené/červené ukazatele (kolečka) reprezentující použité senzory. Pokud je ukazatel zobrazen v červené barvě, je signál senzoru roven „*FALSE*“, v barvě zelené je signál roven „*TRUE*“. Jednotlivé písmena a čísla ukazatelů jsou vysvětlené v pravé dolní části panelu. Na přírubě robotu se nachází ukazatel čísla přítomného nástroje:

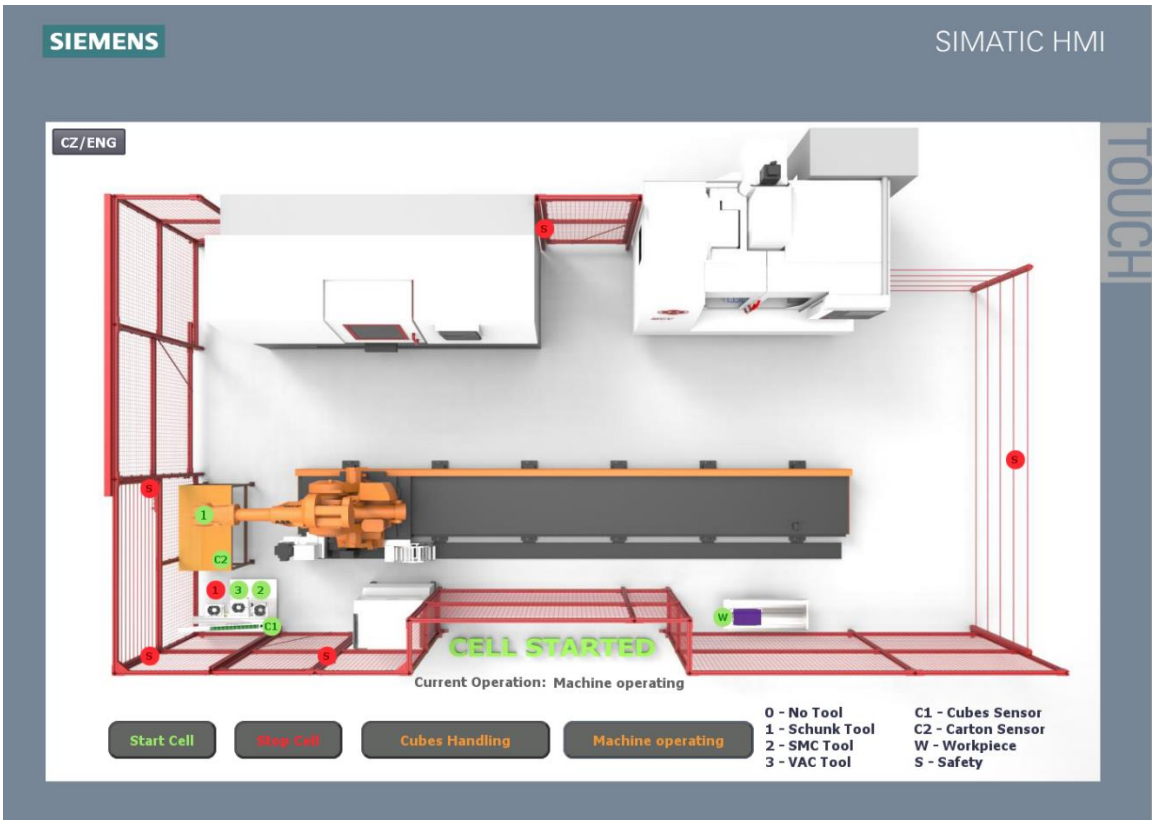
- 0 – Žádný nástroj
- 1 – Nástroj s chapadlem Schunk
- 2 – Nástroj s chapadlem SMC
- 3 – Nástroj s vakuovým chapadlem

Dále jsou v oblasti stojanu ukazatele zprostředkovávající informace o přítomnosti jednotlivých nástrojů. Taktéž se zde nachází ukazatel C1 reprezentující přítomnost kostek v zásobním skluzu. Na odkládacím stolu se nachází ukazatel C2, který udává informaci o přítomnosti manipulovaných kartónu, a v pravé části je ukazatel W, který informuje o přítomnosti polotvaru/obrobku určenému k zakládání/odebírání ze stroje. Poslední ukazatele jsou značeny písmenem S reprezentující bezpečnost. Jelikož se o bezpečnost stará bezpečnostní PLC, které momentálně vrací, zda je bezpečnost splněna a buňka je tak v bezpečném stavu nebo ne, proto nebylo možno ve vizualizaci rozlišit, kde není bezpečnost splněna, zda na dveřních zámcích, či světelných závorách.

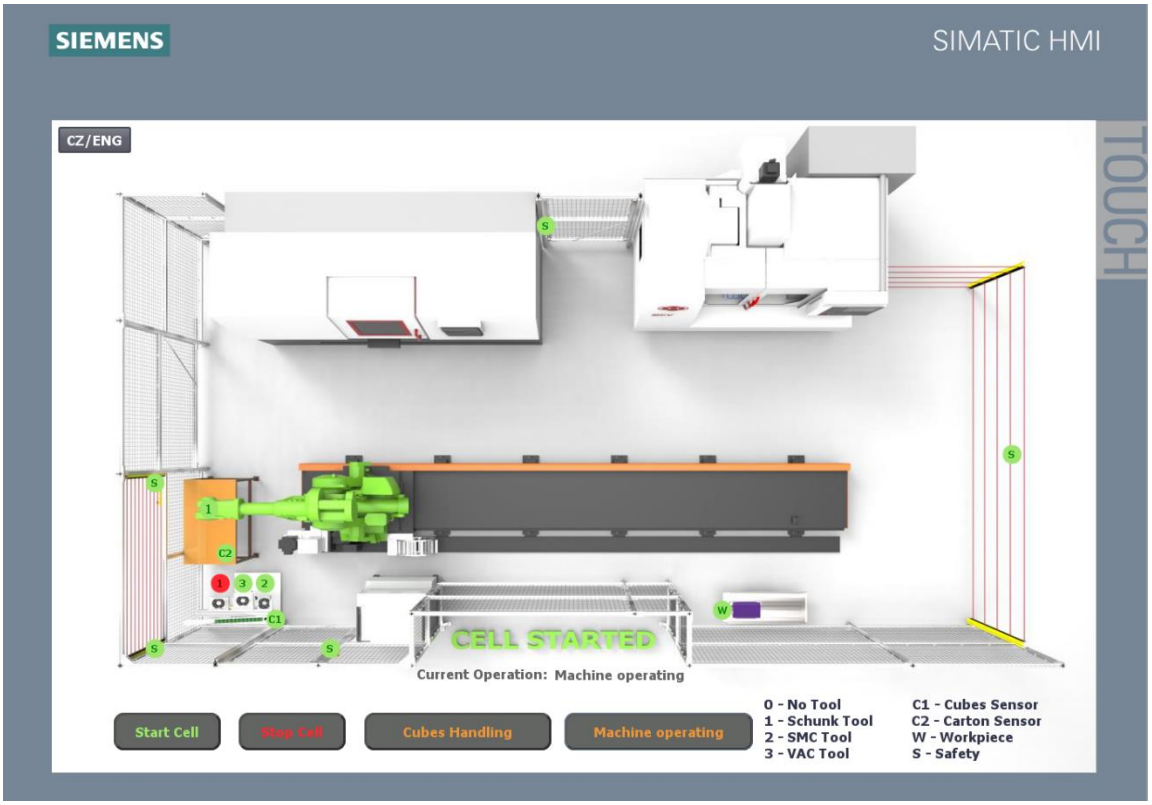
Navržená vizualizace rovněž zobrazuje informaci o tom, zda je buňka v sepnutém nebo vypnutém stavu, dále jaká úloha je zrovna vykonávána a její aktuální postup, tj. kolik dílčích robotických operací je splněno. V případě nesplněné bezpečnosti a požadavku na start robotu se ploty a světelné závory rozblikají červenou barvou (Obr. 4-17). Pokud je bezpečnost splněna a robot pracuje, bliká zeleně (Obr. 4-18). Nakonec byla vytvořená jazyková sada pro možnost přepínání vizualizace mezi českým a anglickým jazykem (Obr. 4-19).



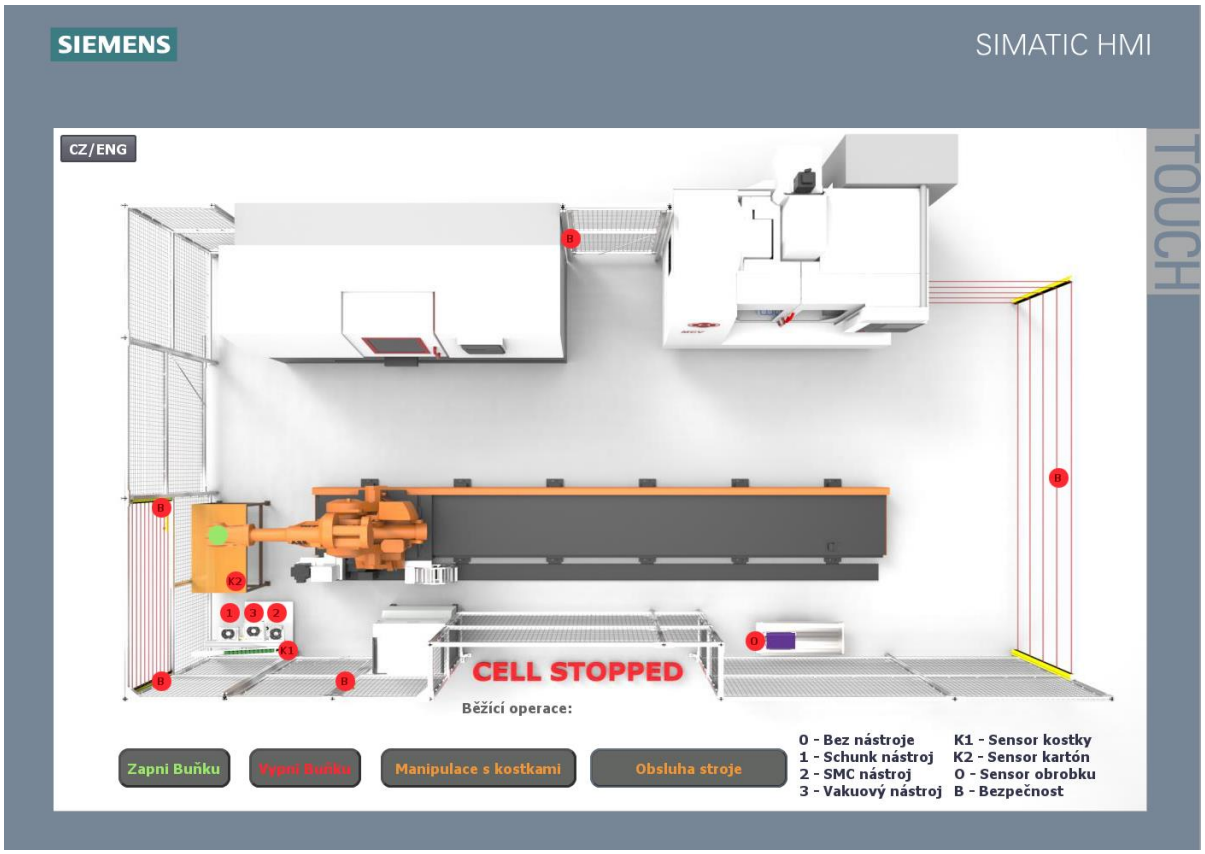
Obr. 4-16 Vizualizace řídicího PLC (HMI)



Obr. 4-17 Vizualizace při nesplněné bezpečnosti



Obr. 4-18 Vizualizace při běžícím robotu

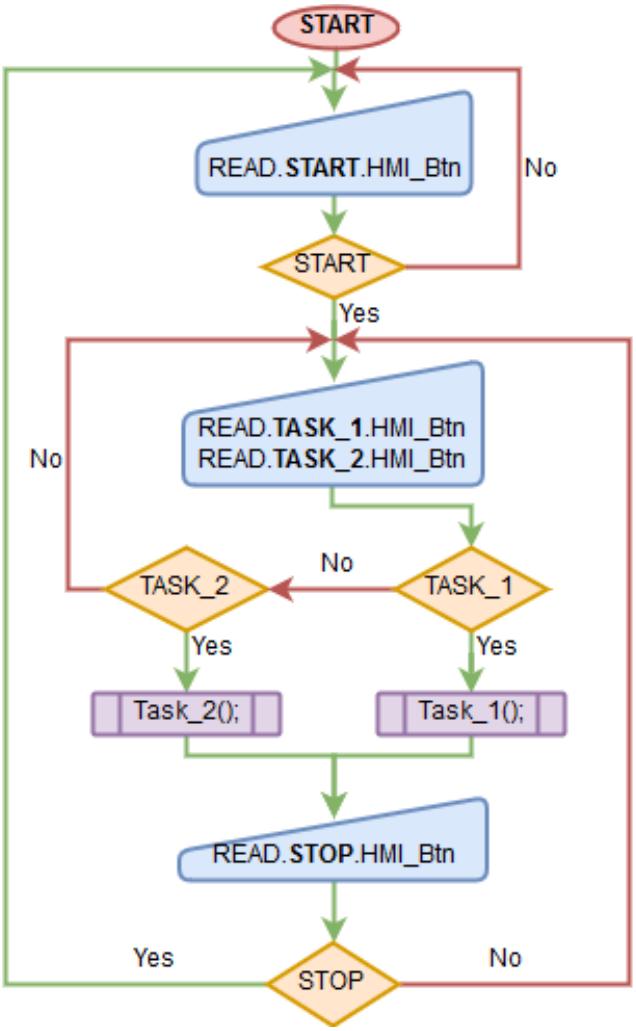


Obr. 4-19 Vizualizace v českém jazyce

4.2.2 Hlavní rutina – Main

Rutina „*Main*“ (Obr. 4-20) je část programu, která se volá po spuštění PLC.

V první části se čeká na sepnutí celé buňky. Toho je docíleno při stištění tlačítka „*Start Cell*“ na ovládacím panelu, což se promítne do proměnné „*START*“. Po spuštění buňky, čímž dojde k její inicializaci načtením všech senzorů a stavu robotu, následuje výběr úlohy. Toho je docíleno načtením proměnných „*TASK_1*“ a „*TASK_2*“, které jsou „*TRUE*“, v případě stištění příslušných tlačítek na ovládacím panelu. Kdy „*TASK_1*“ odpovídá tlačítku „*Machine operating*“ a „*TASK_2*“ odpovídá tlačítku „*Cubes Handling*“. Po načtení těchto proměnných a jejich následném vyhodnocení je volána funkce požadované úlohy. Po ukončení FC volané úlohy je načtena proměnná „*STOP*“, která je určená tlačítkem „*Stop Cell*“. V případě, že tlačítko není stištěno, je program vrácen před volbu provedené úlohy, kdy je možno volat další požadovanou úlohu. Pokud je toto tlačítko stištěno, je program vrácen zpět na počátek, kdy musí být buňka před voláním další úlohy opět spuštěna a inicializována.



Obr. 4-20 Vývojový diagram hlavní rutiny – Main

4.2.3 FC Volání úlohy – *Task_n*

Funkce Volání úlohy (Obr. 4-22) se stará o nastavení správného čísla volané robotické operace a ověření, zda jsou splněny veškeré podmínky nutné pro výkon operace, kde je následně tato operace zavolána. V rámci této funkce je použito několik dalších podfunkcí. V názvu „*Task_**n***“ reprezentuje písmeno „**n**“ číslo úlohy, která má být provedena, jak je naznačeno v hlavní rutině.

Na počátku je načteno číslo nástroje, kterým je zrovna osazen robot, pomocí funkce „*ToolRead*“. Dále je nastaveno číslo robotické operace do paměti v proměnné „*OP_NUMBER_M*“, které je získáno na základě předem definované tabulky „*OperationTable_**n***“ v datovém bloku „*Program_Data_Block*“ (Obr. 4-21), ve kterém jsou čísla požadovaných robotických operací. Poté přichází rozhodnutí, zda je číslo operace rovno nule, pokud ano, je program přesunut nakonec této funkce s tím, že proměnná reprezentující ukončení požadované úlohy „*TASK_END*“ je nastavená do hodnoty „*TRUE*“ a číslo požadované robotické operace „*OP_NUM*“ je vynulováno. To znamená, že kdykoliv program narazí při procházení pole na nulu, je zrovna prováděná úloha ukončena, což zajišťuje možnost jednoduchého rozšíření stávajících úloh o nové dílčí operace.

V případě, kdy je operace nenulová, dochází k ověření, zda osazený nástroj odpovídá požadovanému nástroji dané operace díky funkci „*ToolCheck*“. Pokud ne, je volána funkce

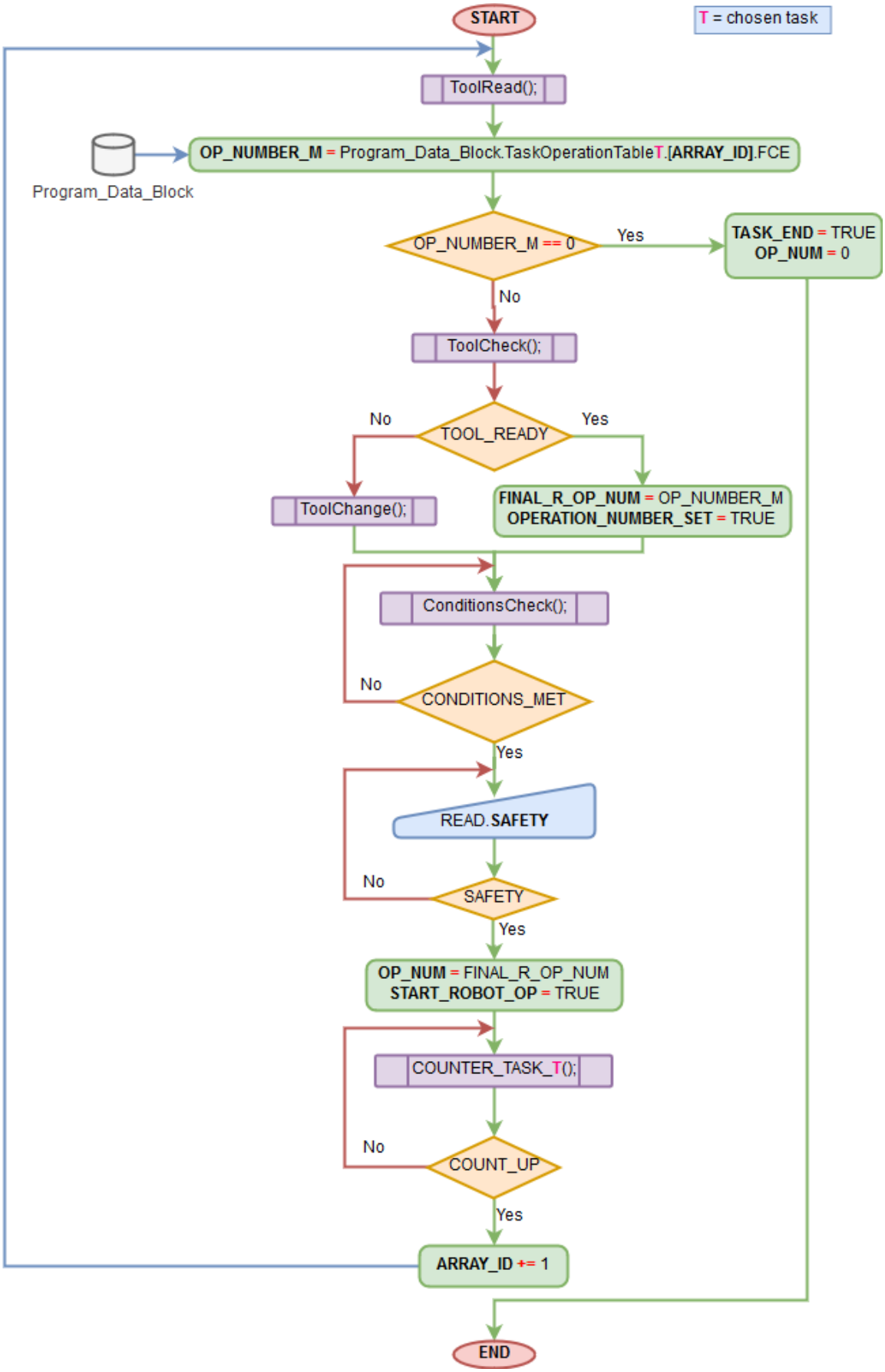
„*ToolChange*“ vracející do proměnné „*FINAL_R_OP_NUM*“ číslo požadované robotické operace výměny nástroje. Jestliže výměna nutná není a nástroj osazený odpovídá nástroji požadovanému, je do proměnné „*FINAL_R_OP_NUM*“ zapsáno číslo požadované robotické operace uložené v proměnné „*OP_NUMBER_M“*. Následně je ověřeno, zda jsou splněné všechny potřebné podmínky pro výkon dané operace pomocí funkce „*ConditionsCheck*“, pokud splněny jsou a je rovněž splněná bezpečnost, je do výstupní proměnné „*OP_NUM*“ zapsáno číslo volané robotické operace nacházející se v proměnné „*FINAL_R_OP_NUM*“ a rovněž je nastavená výstupní proměnná „*START_ROBOT_OP*“ na hodnotu „*TRUE*“, čímž je zavolána požadovaná robotická operace.

Po dokončení operace, což reprezentuje vstupní proměnná „*R_OP_DONE*“, je pomocí funkce „*COUNTER_TASK_**T***“ rozhodnuto, zda má být čítač inkrementován. Takto navržena inkrementace je vhodná tehdy, když dochází mezi požadovanými operacemi k výměně nástrojů, jelikož výměna nástroje není dílčí operací žádné z úloh, a tudíž není nutný posun v deklarovaném poli operací čísel. V názvu funkce pak „**T**“ reprezentuje číslo požadované úlohy, kde 1 je obsluha stroje a 2 manipulace s kostkami.

Jakmile je čítač inkrementován, dochází k posunutí v definovaném poli operací, čímž je vykonána následující operace za předpokladu splnění všech podmínek, jako u operace předchozí.

3	OperationTable1	Array[0..2] of "OperationDescription"	
4	OperationTable1[0]	"OperationDescription"	
5	ID	Int	0
6	TargetStation	Int	1
7	ID_NEXT_OPERATION_ROW	Int	1
8	OP_NUM	Int	101
9	OperationTable1[1]	"OperationDescription"	
10	ID	Int	1
11	TargetStation	Int	1
12	ID_NEXT_OPERATION_ROW	Int	2
13	OP_NUM	Int	102
14	OperationTable1[2]	"OperationDescription"	
15	ID	Int	3
16	TargetStation	Int	1
17	ID_NEXT_OPERATION_ROW	Int	0
18	OP_NUM	Int	0

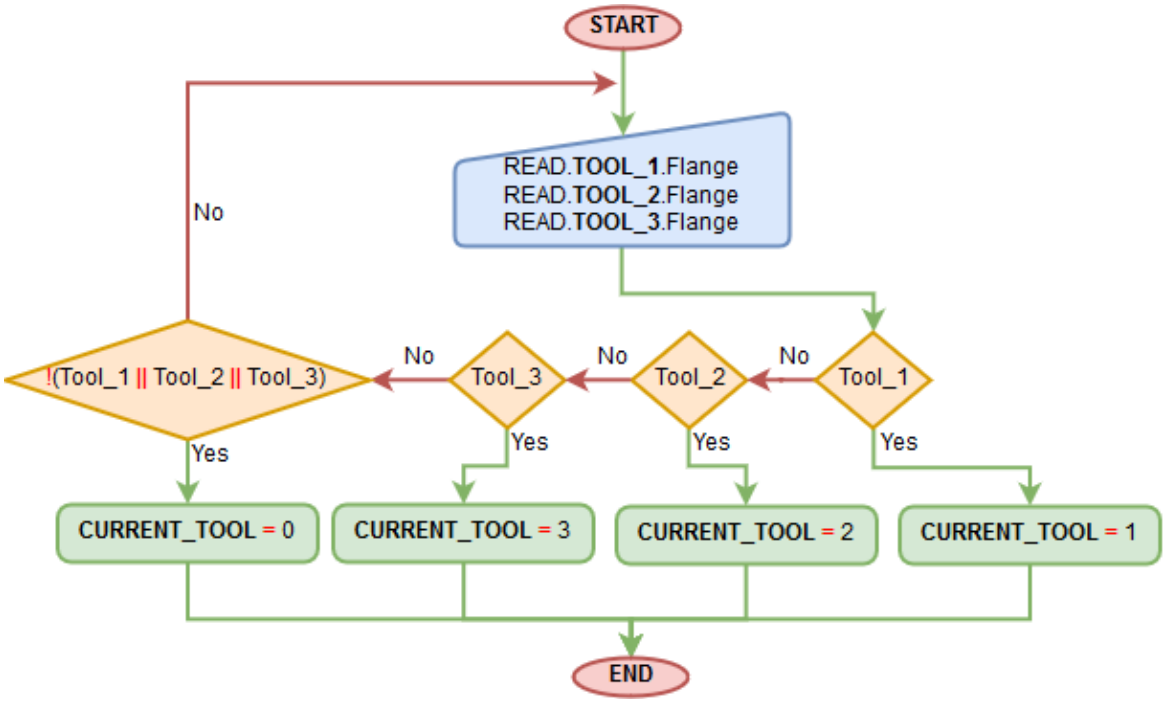
Obr. 4-21 Pole operací Program_Data_Block pro úlohu Obsluha stroje



Obr. 4-22 Vývojový diagram FC Volání úlohy – Task_n

FC Načtení nástroje – ToolRead

Funkce Načtení nástroje (Obr. 4-23) slouží k načtení nástroje, kterým je robot osazen a nastavení jeho číselné hodnoty do proměnné „*CURRENT_TOOL*“. Toho je docíleno pomocí několika IF-ELSEIF podmínek, kdy podle senzoru, který zrovna vrací hodnotu „*TRUE*“ je nastaveno číslo použitého nástroje. V případě, že robot není osazen žádným nástrojem, jsou všechny senzory v logické nule a číslo nástroje je tedy nula.

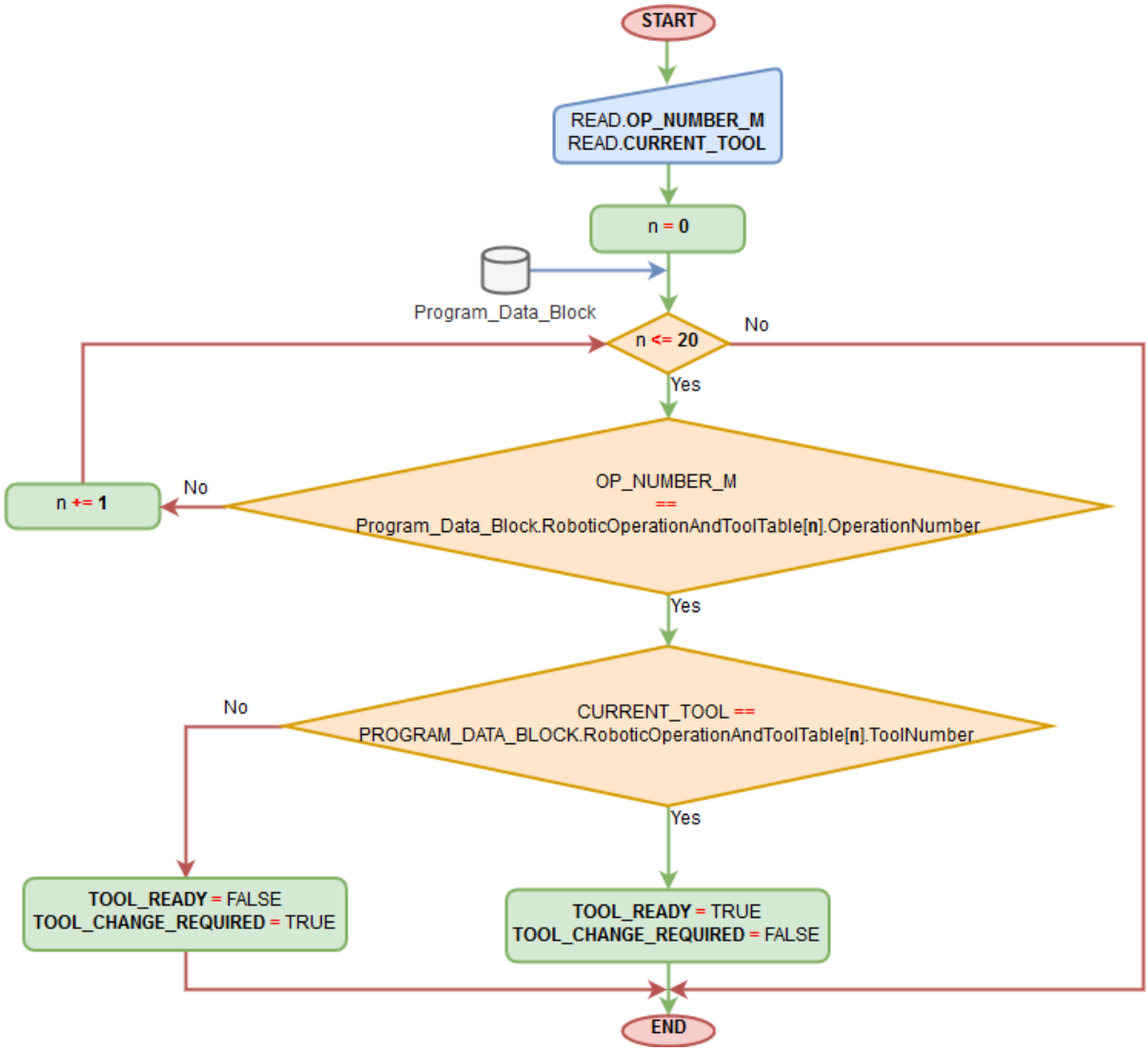


Obr. 4-23 Vývojový diagram FC Načtení nástroje – ToolRead

FC Kontrola nástroje – ToolCheck

Funkce Kontrola nástroje (Obr. 4-24) slouží ke kontrole nástroje vzhledem k požadované robotické operaci. Je logické, že robot nemůže vykonávat operaci při osazení nástrojem, který pro tuto operaci není určen.

Prvně musí být načteny hodnoty proměnných číslo nástroje „*CURRENT_TOOL*“ a číslo požadované robotické operace „*OP_NUMBER_M*“, dále je nastavena hodnota proměnné „*n*“ na nulu. Poté, za použití cyklu FOR jdoucího podle proměnné „*n*“, je projita tabulka operací a nástrojů „*RoboticOperationAndTools*“ (Obr. 4-25). V případě, že se načtené číslo operace rovná číslu operace na řádku „*n*“ je poté ze stejného řádku načtena hodnota nástroje. Tato hodnota je následně porovnána s hodnotou proměnné číslo nástroje, načtenou na počátku tohoto funkčního bloku. Pokud se číslo načteného nástroje shoduje s číslem požadovaného nástroje jsou nastaveny proměnné „*TOOL_READY*“ na „*TRUE*“ a „*TOOL_CHANGE_REQUIRED*“ na „*FALSE*“. V případě, kdy k rovnosti nedochází, je proměnná „*TOOL_READY*“ nastavena na „*FALSE*“ a „*TOOL_CHANGE_REQUIRED*“ na „*TRUE*“.



Obr. 4-24 Vývojový diagram FC Kontrola nástroje – ToolCheck

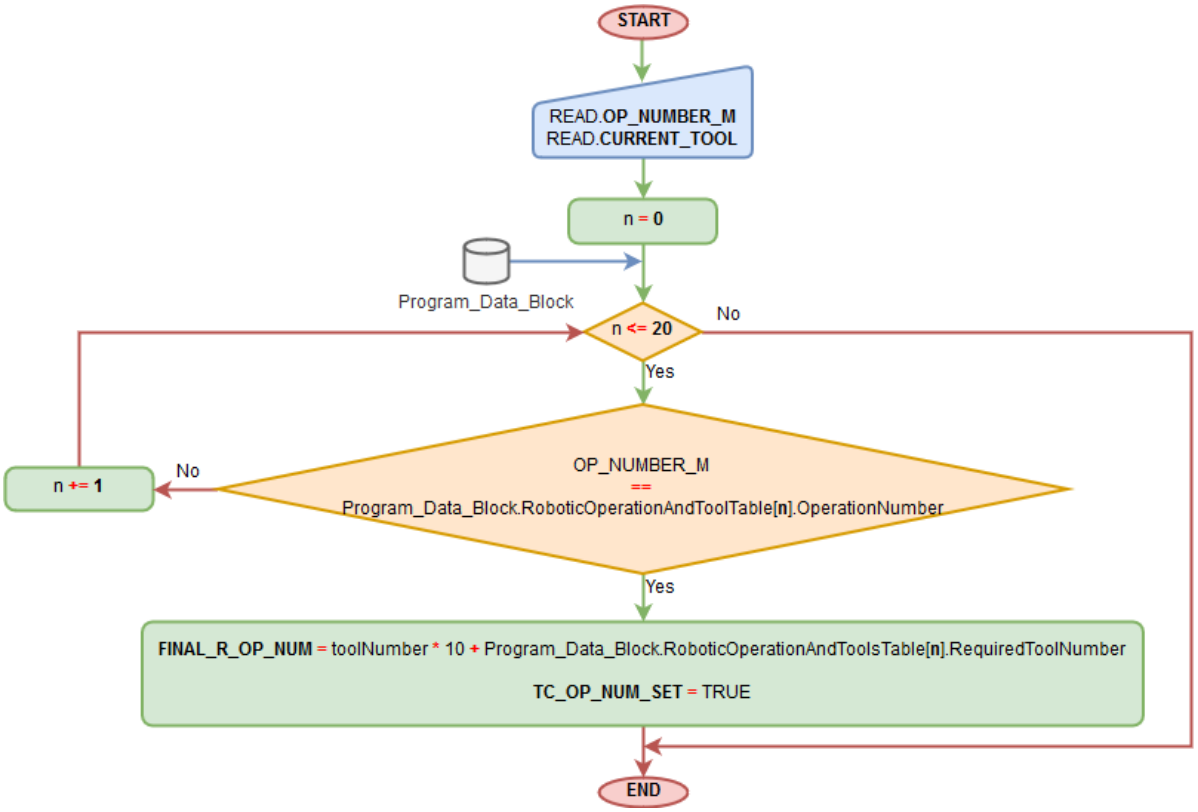
20		RoboticOperationsAndTools	Array[0..50] of "...	
21		RoboticOperationsAndTools[0]	"OperationsAndTools"	
22		CalledOperationNumber	Int	101
23		RequiredToolNumber	Int	1
24		RoboticOperationsAndTools[1]	"OperationsAndTools"	
25		RoboticOperationsAndTools[2]	"OperationsAndTools"	
26		RoboticOperationsAndTools[3]	"OperationsAndTools"	
27		CalledOperationNumber	Int	202
28		RequiredToolNumber	Int	2
29		RoboticOperationsAndTools[4]	"OperationsAndTools"	
30		RoboticOperationsAndTools[5]	"OperationsAndTools"	

Obr. 4-25 Tabulka operací a nástrojů

FC Výměna nástroje – ToolChange

Funkce Výměna nástroje (Obr. 4-26) je volána v případě, že je nutné provést výměnu nástroje. V této funkci jsou načteny dvě vstupní proměnné číslo nástroje „*CURRENT_TOOL*“ a číslo operace „*OP_NUMBER_M*“.

Jako v přechozí funkci Kontrola nástroje, je zde prvně deklarována dočasná proměnná „*n*“ na základě, které prochází FOR cyklus tabulku robotických operací a nástrojů „*RoboticOperationAndTools*“, ve které jsou zapsány nástroje příslušící každé operaci. V případě, že se načtené číslo operace rovná číslu operace na řádku „*n*“, je poté nastavená proměnná „*FINAL_R_OP_NUM*“ jako číslo nástroje, který je přítomný na přírubě robotu násobené desíti, plus číselná hodnota požadovaného nástroje z tabulky robotických operací a nástrojů, což je poté číslem robotické operace výměny v programu robotu.



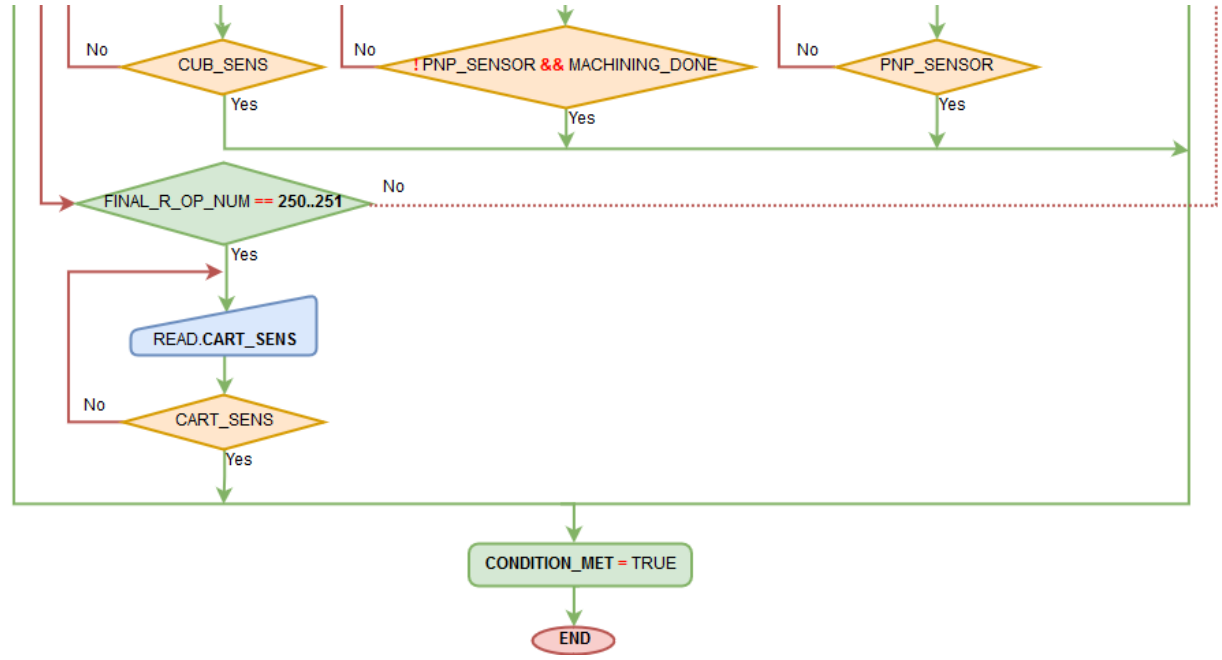
Obr. 4-26 Vývojový diagram FC Výměna nástroje – ToolChange

FC Ověření podmínek – ConditionsCheck

Funkce Ověření podmínek (Obr. 4-27) slouží k verifikaci, zda jsou všechny logické podmínky nutné k vykonání požadované robotické operace splněny.

Prvně je načteno číslo požadované robotické operace „*FINAL_R_OP_NUM*“ a následně je na základě tohoto čísla pomocí SWITCH-CASE podmínky, provedeno rozhodnutí, zda je nebo není podmínka splněna. Jestli je podmínka splněna, je rozhodnuto díky signálům získaným ze senzorů. V případě výměn nástrojů se jedná zejména o přítomnost nástrojů ve stojanu, což jsou operace 1 – 3, 10 – 13, 20 – 23 a 30 – 32. Další jsou operace úlohy Obsluhy stroje 101 a 102, kde při operaci 101, což je zakládání do stroje, je důležitá přítomnost zakládaného polotovaru a při operaci 102, což je odebírání obrobku ze stroje, je důležité, aby bylo dokončeno obrábění a místo pro odložení obrobku bylo prázdné.

Posledními jsou operace při úloze Manipulace s kostkami, kde pro dílčí manipulační operace s kostkami 201 – 212 je nutné, aby vůbec nějaká kostka byla ve skluzu přítomná, a při manipulaci s kartóny 250 – 251 je rovněž nutná jejich přítomnost. Pokud jsou veškeré podmínky pro danou operaci splněny, je nastavená výstupní proměnná „*CONDITION_MET*“ na hodnotu „*TRUE*“. V opačném případě se čeká na splnění podmínek.

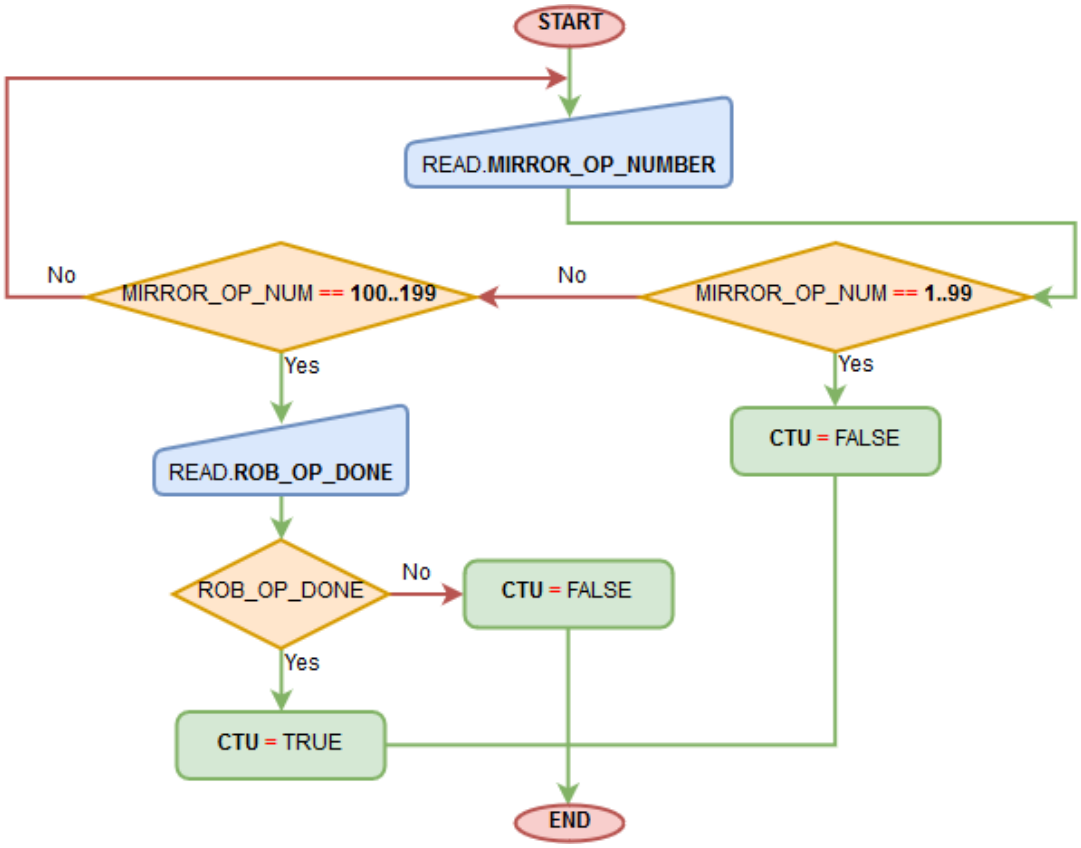


Obr. 4-27 Vývojový diagram FC Ověření podmínek – ConditionsCheck

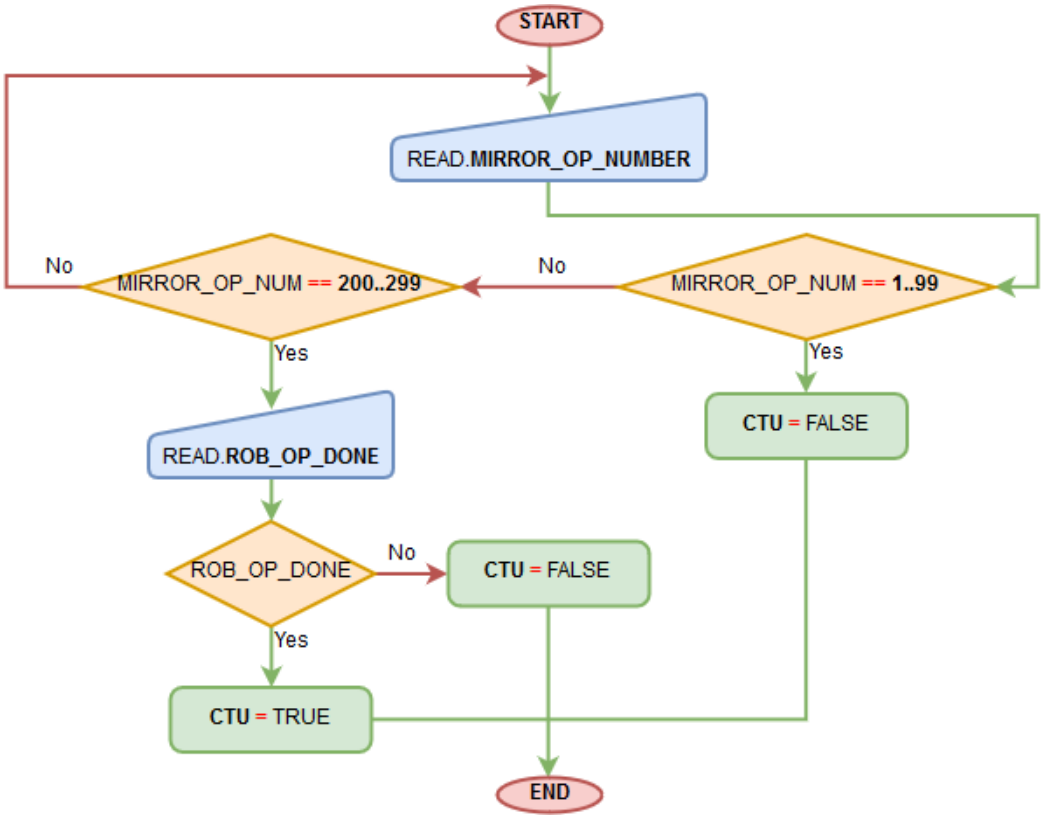
FC Inkrementace čítače – COUNTER_TASK_T

Funkce Inkrementace čítače slouží k vystavení proměnné „*CTU*“ do hodnoty „*TRUE*“ tehdy, když je prováděná operace definovaná v tabulce operací „*OperationTable_**T***“, kde „**T**“ reprezentuje prováděnou úlohu (Obr. 4-28 a Obr. 4-29). To zamezuje, aby došlo k inkrementaci, tj. k volání další operace i když byla prováděná pouze mezioperační výměnná nástrojů, které nemá na celkový průběh zvolené úlohy vliv. Operace, které nejsou započítávány, se nacházejí v rozsahu 1 – 99 a momentálně se jedná o výměny nástrojů.

Funkce začíná načtením proměnné „*MIRROR_OP_NUM*“, což je číslo operace vykonávané robotem, které vrací řídicí systém robotu. Následně je ve SWITCH-CASE struktuře rozhodnuto, o jakou operaci se jedná. Pokud je operace z rozsahu 1 – 99 je výstupní proměnná „*CTU*“ nastavena jako „*FALSE*“. V případě, že je operace „*MIRROR_OP_NUM*“ v rozsahu 100 – 199 a je dokončená operace reprezentována načtením proměnné „*ROB_OP_DONE*“, která se rovná „*TRUE*“, je výstupní proměnná „*CTU*“ nastavena jako „*TRUE*“. Jestliže rozsah odpovídá, ale operace není dokončená, pak je proměnná „*CTU*“ nastavena jako „*FALSE*“. Toto platí pro úlohu Obsluha stroje. Pro Manipulaci s kostkami je rozsah operací „*MIRROR_OP_NUM*“ 200 – 299.



Obr. 4-28 Vývojový diagram FC Inkrementace čítače pro úlohu Obsluha stroje



Obr. 4-29 Vývojový diagram FC Inkrementace čítače pro úlohu Manipulace s kostkami

4.2.4 PLC signály (Tagy)

Jakmile byl program vytvořen, bylo nutné namapovat jeho signály na vstupní/výstupní proměnné tzv. PLC tagy (Obr. 4-30). Tyto signály korespondují se signály v simulaci v PS (Obr. 4-31). Signály jsou v PS i PLC brány z pohledu PLC. Adresy vstupních signálu začínají písmenem I a výstupních signálu písmenem Q.

ProcessSimulate

	Name	Data type	Address	Comment
1	CART_SENS	Bool	%I0.0	Carton sensor
2	CUB_SENS	Bool	%I0.1	Cubes sensor
3	PNP_SENS	Bool	%I0.2	Handling part sensor
4	SCHUNK_SENS_F	Bool	%I1.0	Schunk sensor flange
5	SMC_SENS_F	Bool	%I1.2	SMC sensor flange
6	VAC_SENS_F	Bool	%I1.3	VAC sensor flange
7	SCHUNK_SENS_S	Bool	%I2.0	Schunk sensor stand
8	SMC_SENS_S	Bool	%I2.1	SMC sensor stand
9	VAC_SENS_S	Bool	%I2.2	VAC sensor stand
10	RPE	Bool	%I10.0	Robot operation end
11	RW	Bool	%I10.1	Robot working
12	PRP	Bool	%I10.2	Pause robotic program
13	RPMN	Int	%IW16	Robotic program mirror number
14	SAFETY_OK	Bool	%I100.0	Safety check
15	SRO	Bool	%Q12.0	Start robotic operation
16	RPN	Int	%QW14	Robotic program number
17	EMS	Bool	%Q100.1	Emergency stop
18	SM	Bool	%Q100.2	Servis Mode

Obr. 4-30 Namapované signály v PLC

Signal Viewer

Signal Name	Memory	Type	Address	IEC Format	PLC Connection
Type here to filter			*Type here to filter*	*Type here to fil*	✓
CART_SENS	☐	BOOL	0.0	I0.0	☑
CUB_SENS	☐	BOOL	0.1	I0.1	☑
PNP_SENS	☐	BOOL	0.2	I0.2	☑
SCHUNK_SENS_F	☐	BOOL	1.0	I1.0	☑
SMC_SENS_F	☐	BOOL	1.2	I1.2	☑
VAC_SENS_F	☐	BOOL	1.3	I1.3	☑
SCHUNK_SENS_S	☐	BOOL	2.0	I2.0	☑
SMC_SENS_S	☐	BOOL	2.1	I2.1	☑
VAC_SENS_S	☐	BOOL	2.2	I2.2	☑
irb4400_60_rev02_programEnded	☐	BOOL	10.0	I10.0	☑
RobotWorking_PLC	☐	BOOL	10.1	I10.1	☑
irb4400_60_rev02_programPause	☐	BOOL	10.2	Q10.2	☑
irb4400_60_rev02_startProgram	☐	BOOL	12.0	Q12.0	☑
irb4400_60_rev02_programNumber	☐	INT	14	Q14	☑
irb4400_60_rev02_mirrorProgramNumber	☐	INT	16	I16	☑
SAFETY_FENCES	☐	BOOL	100.0	I100.0	☑
irb4400_60_rev02_emergencyStop	☐	BOOL	100.1	Q100.1	☑

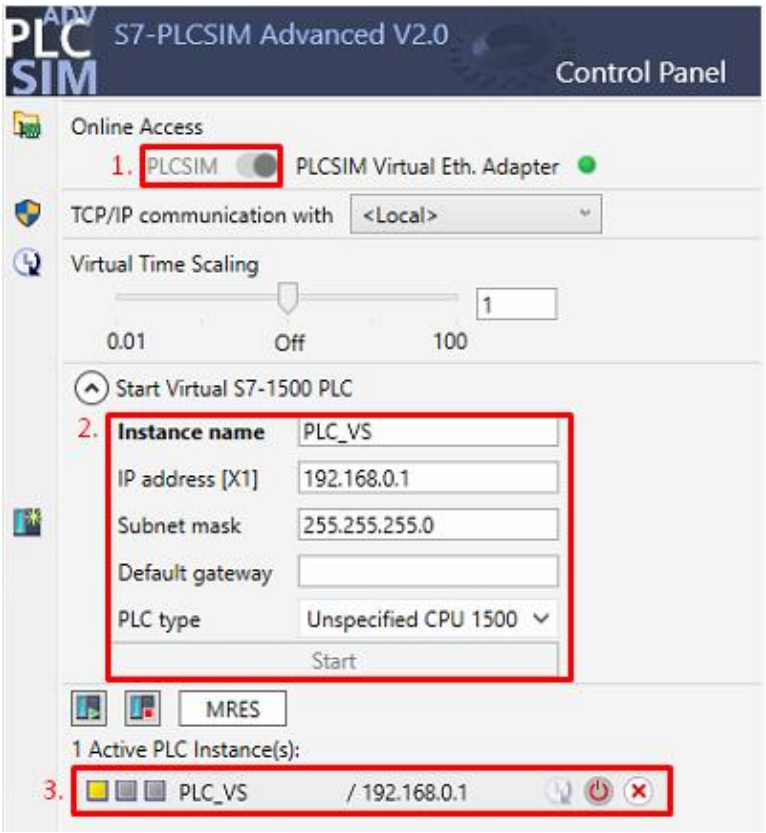
Obr. 4-31 Namapované signály v PS

4.3 Virtuální zprovoznění

Pro virtuální zprovoznění v této práci bylo použito softwaru PLCSIM Advanced 2.0 sloužící k vytvoření virtuální PLC instance do které byl nahrán vytvořený řídicí program. Následně bylo nutné správně nastavit PS pro připojení k externímu kontroléru. Dalším krokem bylo nastavení PG/PC rozhraní, úprava projektu a nahrání programu do PLC instance prostřednictvím TIA portálu. Nakonec bylo nutné spustit simulace a ovládat skrze HMI panel virtuální model, čímž bylo provedeno virtuální zprovoznění.

4.3.1 Tvorba PLC instance v SW PLCSIM Advanced 2.0

Pro vytvoření PLC instance je nutné po spuštění softwaru PLCSIM Advanced 2.0 zpřístupnit instanci on-line (Obr. 4-32 bod 1). Toto je důležité, jelikož by při snaze nahrát program z TIA portálu do vytvořené PLC instance nebyla tato instance vidět. Dále je nutné zvolit název, IP adresu a typ PLC. V případě této práce je zvolen typ PLC jako „*Unspecified CPU 1500*“. Po zadání všech parametrů je instance vytvořena stiskem tlačítka „*Start*“ (Obr. 4-32 bod 2). Jakmile je instance vytvořena, je ji možno vidět v seznamu (Obr. 4-32 bod 3).

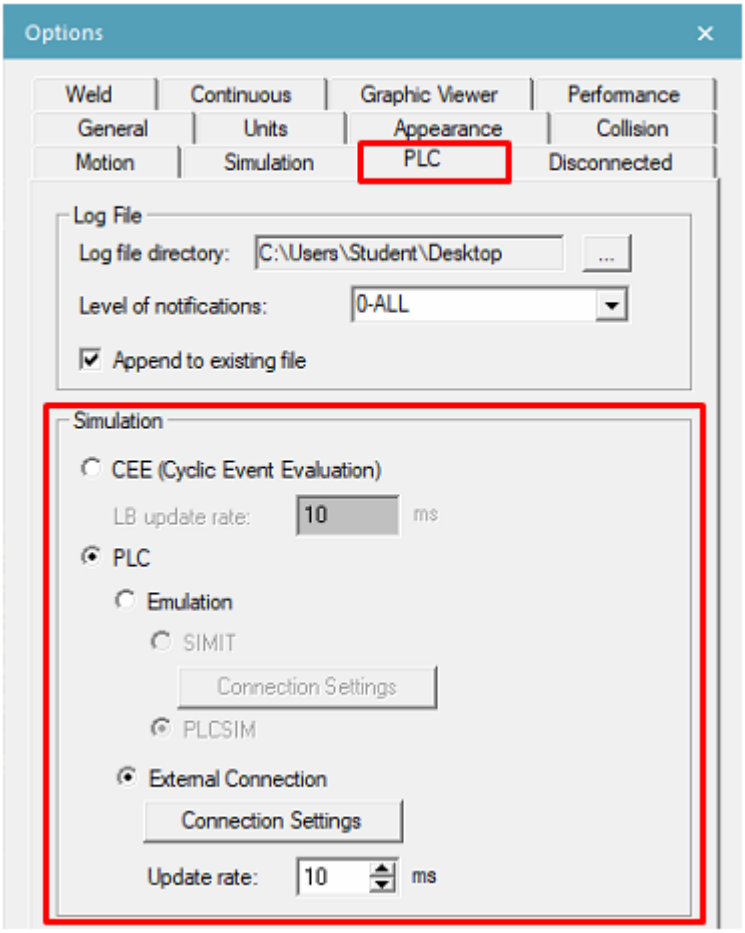


Obr. 4-32 Tvorba PLC instance v PLCSIM Advanced 2.0

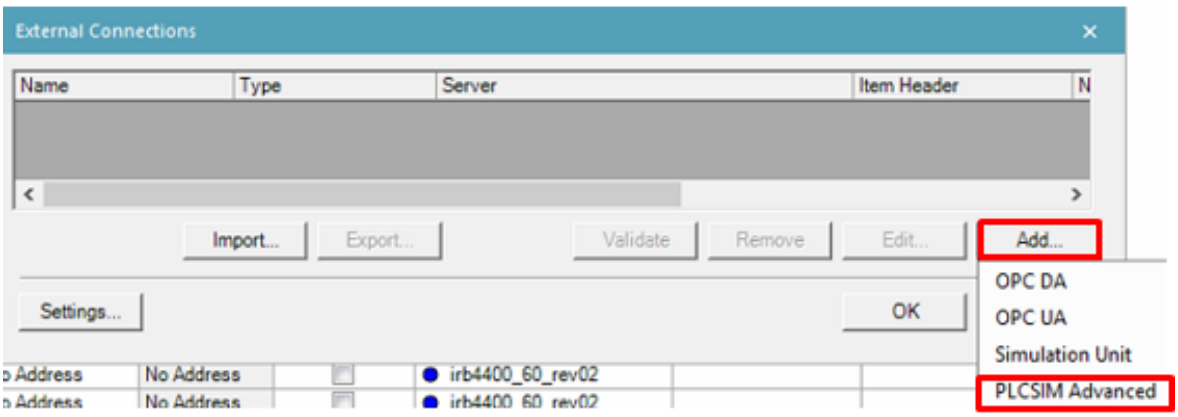
4.3.2 Nastavení PS pro externí řízení

Aby bylo možné připojit PS k externímu řízení bylo nutné jej správně nastavit. V okně „*Options*“ (File→Options) pod záložkou „*PLC*“ v sekci „*Simulation*“ je nutno zaškrtnou políčko „*PLC*“ a následně „*ExternalConnection*“ (Obr. 4-33). Dále po rozkliknutí „*ConnectionSettings*“ je v okně „*ExternalConnections*“ pomocí tlačítka „*Add…*“ přidáno

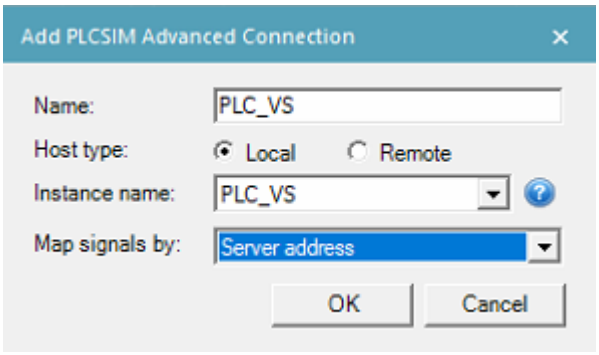
připojení k PLCSIM Advanced (Obr. 4-34). Poté je v okně „*Add PLCSIM AdvancedConnection*“ definován název připojení a vybraná vytvořená instance v PLCSIM Advanced 2.0 (Obr. 4-35). Po vytvoření externího připojení je vhodné toto připojení ověřit kliknutím na tlačítko „*Validate*“.



Obr. 4-33 Nastavení PS pro externí řízení



Obr. 4-34 Přidání externího připojení

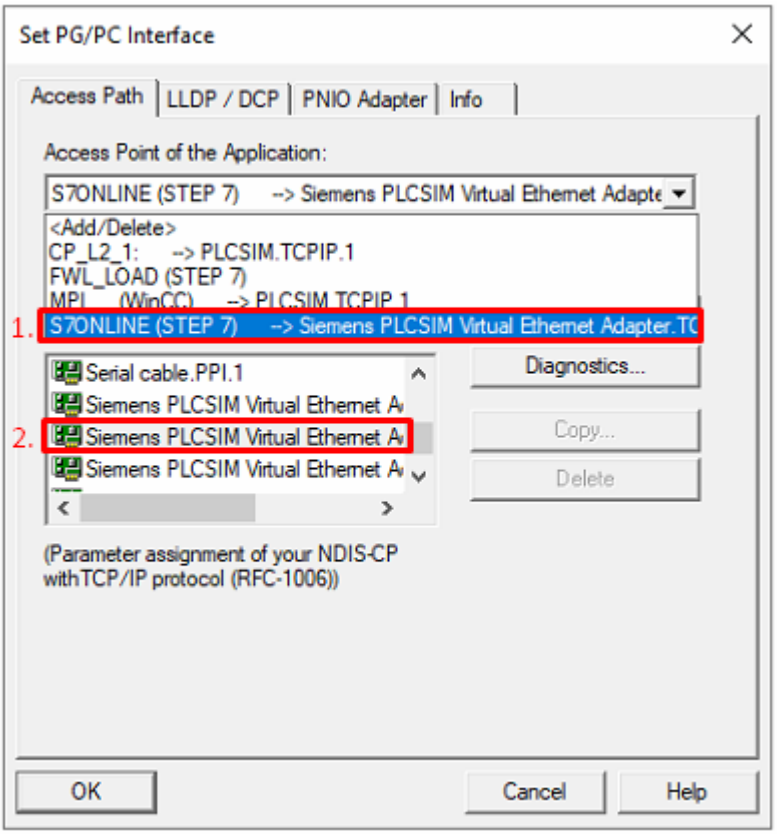


Obr. 4-35 Deklarace externího připojení

4.3.3 Nastavení PG/PC rozhraní

Aby bylo možné ovládat PLC program skrze virtuální HMI panel je nutné upravit PG/PC rozhraní, které vytváří spojení mezi virtuálním ethernetovým adaptérem, ke kterému je připojena vytvořená PLC instance a virtuální simulací ovládacího HMI panelu.

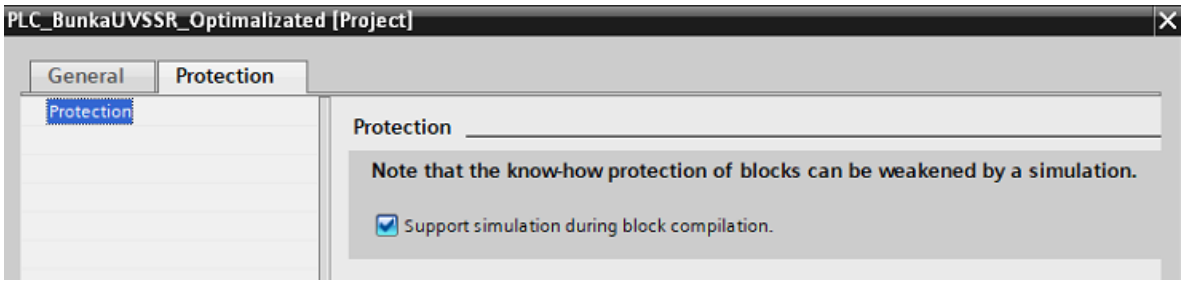
Toto nastavení lze nalézt v ovládacích panelech počítače. Po otevření je třeba rozkliknout záložku a vybrat přístupový bod „*S7ONLINE (STEP 7)* → *Siemens PLCSIM Virtual Ethernet Adapter.TCPIP.1*“ (Obr. 4-36 bod 1), čímž se přidal do spodního okna, kde je jej třeba vybrat (Obr. 4-36 bod 2) a poté potvrdit stiskem tlačítka „*OK*“.



Obr. 4-36 Nastavení PG/PC rozhraní

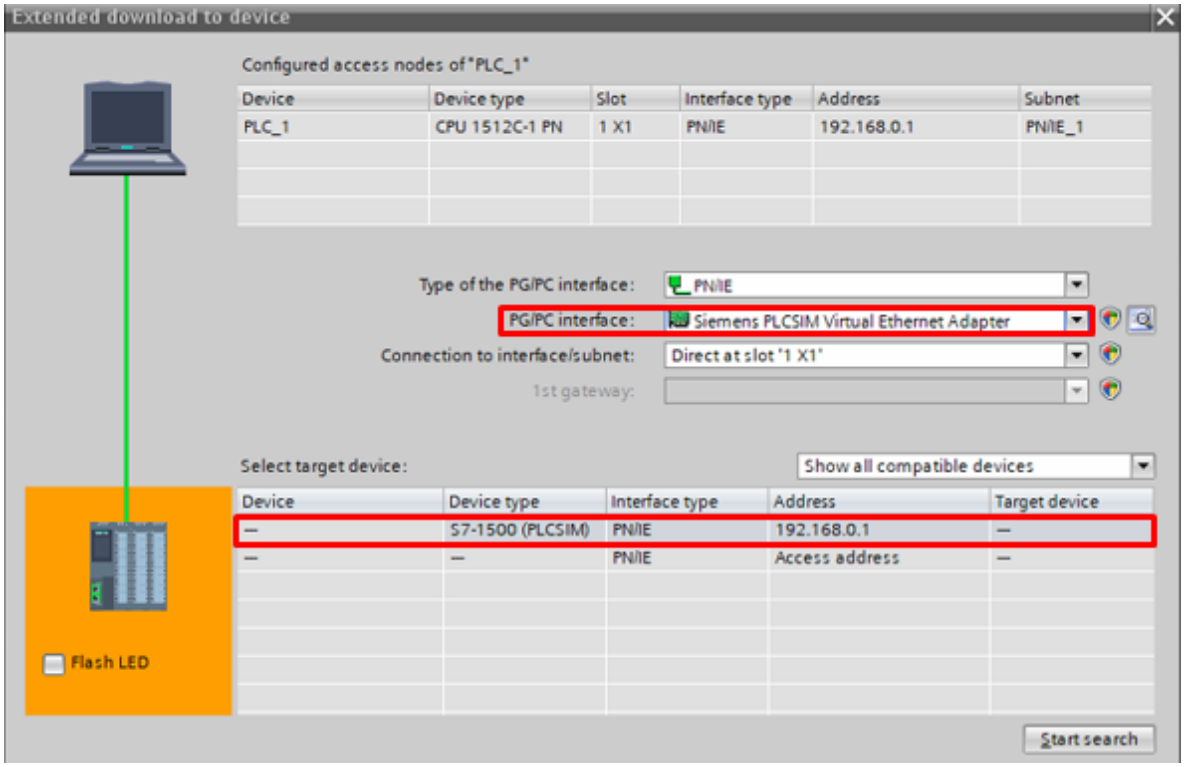
4.3.4 Nastavení projektu a nahrání PLC programu

Aby bylo možné nahrát PLC program do vytvořené instance a virtuálně jej ovládat, je nutné v nastavení projektu v záložce „*Protection*“ zaškrtnout „*Support Simulation During Block Compilation*“ (Obr. 4-37).



Obr. 4-37 Změna v nastavení projektu

Poté je možné PLC program nahrát do vytvořené instance, kde je nutné v PG/PC interface vybrat „*Siemens PLCSIM Virtual Ethernet Adapter*“, a následně po vyhledání stištěním tlačítka „*Start search*“ by měla být nalezena vytvořená instance se zadanou IP adresou (Obr. 4-38).

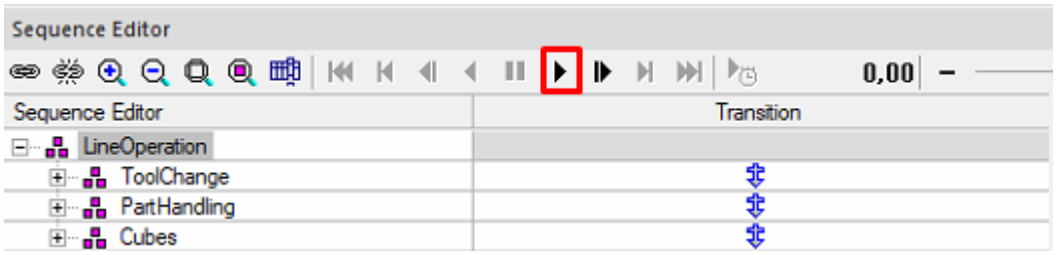


Obr. 4-38 Nahrávání PLC programu do vytvořené instance

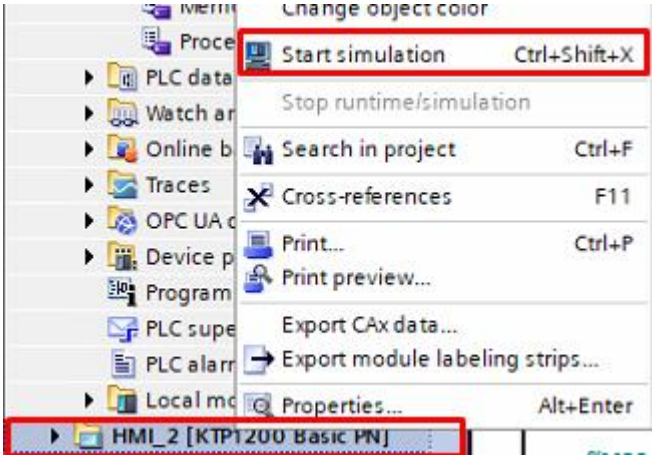
4.3.5 Spuštění virtuálního zprovoznění

Simulaci je nutno spustit v PS kliknutím na tlačítko „*Play Simulation*“ v „*Sequence Editoru*“ (Obr. 4-39). Poté je nutno spustit simulaci HMI panelu v TIA portálu kliknutím pravým tlačítkem myši na HMI a poté stištěním příkazu „*Start Simulation*“ (Obr. 4-40). Po spuštění

těchto dvou simulací je možné ovládat model výrobního systému v PS skrze virtuální HMI panel, čímž bylo provedeno virtuální zprovoznění.



Obr. 4-39 Spuštění simulace v PS

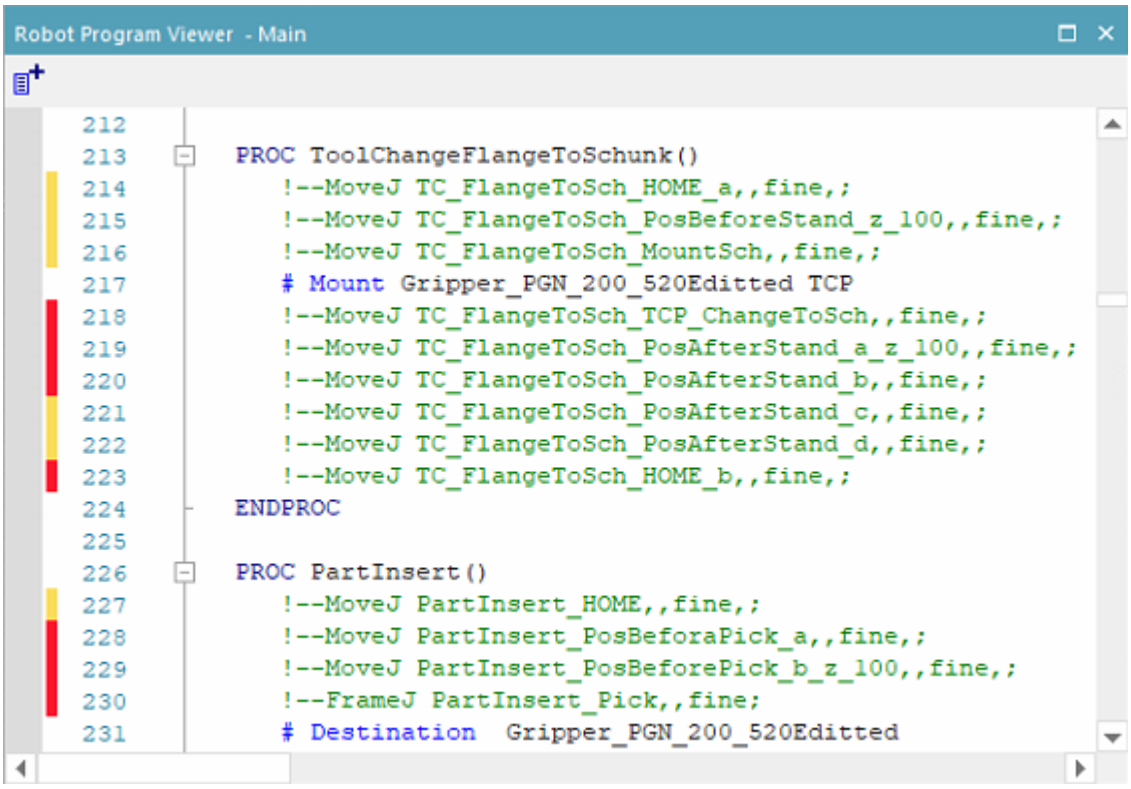


Obr. 4-40 Spuštění simulace HMI panelu v TIA portálu

4.4 Program robotu

Program robotu byl v hrubé formě získán z PS (Obr. 4-41). Vlivem nemožnosti využít při návrhu operací RCS (Robot Controller Simulation) modul, který slouží k získání přesných dat pro plánování pohybu a dopočítávání inverzní kinematiky, byl program vyexportován z PS bez definic rychlostí, nástrojů a referenčních souřadných systému, což znamenalo, že každá pohybová instrukce byla neúplná a bylo ji nutno těmito atributy doplnit. Každá operace v PS je v programu reprezentována jednou procedurou. Pro všechny procedury byly v deklarační části programu vygenerovány body trajektorie všech operací, což ve výsledku čítá 254 bodů.

Vygenerovaný program byl posléze importován do prostředí Robot Studio, kde byly veškeré pohybové instrukce doplněny o chybějící parametry. Dále do jednotlivých procedur byly přidány reálné příkazy určené k výměně nástrojů, či uzavření a otevření jednotlivých nástrojů, čímž byly nahrazeny OLP příkazy použité pro tyto úlohy pouze v prostředí PS. Příklad procedury doplněné o chybějící parametry je možno vidět na obrázku (Obr. 4-42). Každá procedura musela být následně simulována, kde byly definovány konfigurace os robotu pro jednotlivé body požadované trajektorie (Obr. 4-43), čímž se zajistilo, že se robot vždy dostane do těchto bodů a nedojde k nechtěnému přetočení os.



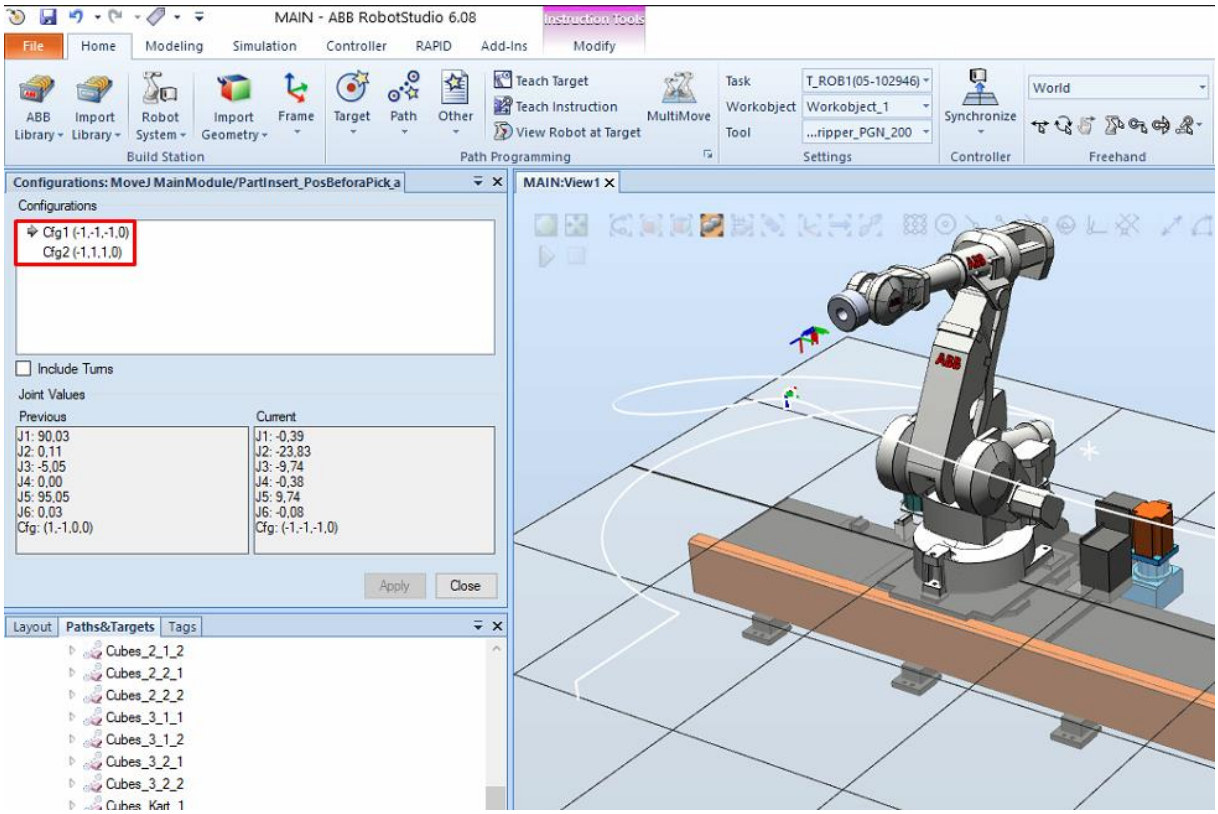
Obr. 4-41 Program robotu v hrubé formě

```
!--Procedure where robot mounts Schunk Tool
PROC ToolChangeFlangeToSchunk()
    SetDO OPERATION_DONE,low; !--Setting output signal OPERATION_DONE as false
    MoveJ TC_FlangeToSch_HOME_a,v1000,fine,toolChanger\WObj:=Workobject_1;
    MoveJ TC_FlangeToSch_PosBeforeStand,v1000,z1,toolChanger\WObj:=Workobject_1;
    MoveL TC_FlangeToSch_MountSch,v100,fine,toolChanger\WObj:=Workobject_1;
    !--Several move instructions
    !--Point to point or linear movement, velocity, zone, tool with reference frame

    PulseDO Local_IO_0_DO1;
    WaitTime 0.5;
    !--Pulse signal to attach tool

    MoveL TC_FlangeToSch_PosAfterStand_a,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_FlangeToSch_PosAfterStand_b,v300,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_FlangeToSch_PosAfterStand_c,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_FlangeToSch_PosAfterStand_d,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_FlangeToSch_HOME_b,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    !--Next movement instructions with the same attributes as on the beginning
    SetDO OPERATION_DONE,high; !--Setting output signal OPERATION_DONE as false
ENDPROC
```

Obr. 4-42 Příklad doplněné procedury



Obr. 4-43 Ukázka nastavení konfigurace os robotu pro zvolený bod trajektorie

Nakonec byla vytvořena procedura „*CellProgram*“ (Obr. 4-44), ve které dochází k volání jednotlivých procedur operací na základě funkce „*Call by variable*“, což je ekvivalent známe SWITCH-CASE podmínky. Proměnná „*OPERATION_NUMBER*“ je numerická a reprezentuje, která z procedur má být volána. Volání procedury je podmíněno proměnnou „*START_ROBOT*“ Při každém volání procedury je nejprve vystavená proměnná „*ROBOT_WORKING*“ do hodnoty „*TRUE* a proměnná „*MIRROR_NUMBER*“ na hodnotu „*OPERATION_NUMBER*“. Následně je provedena procedura a po jejím ukončení je proměnná „*ROBOT_WORKING*“ nastavena na „*FALSE*“. V rámci každé podprocedury je na jejím začátku nastavena proměnná „*OPERATION_DONE*“ na hodnotu „FALSE“ a následně, po vykonání všech instrukcí, je nastavena na hodnotu „*TRUE*“. Tyto proměnné odpovídají proměnným v PLC programu, které byly použity pro řízení virtuálního modelu, díky čemu není nutné měnit již odladěný PLC program.

```
274     PROC CellProgram()
275         TEST AInput(OPERATION_NUMBER)
276         CASE 1:
277             IF DInput(START_ROBOT)=high THEN
278                 SetDO ROBOT_WORKING,high;
279                 SetAO MIRROR_NUMBER,OPERATION_NUMBER;
280                 ToolChangeFlangeToSchunk;
281                 SetDO ROBOT_WORKING,low;
282             ENDIF
283         CASE 2:
284             IF DInput(START_ROBOT)=high THEN
285                 SetDO ROBOT_WORKING,high;
286                 SetAO MIRROR_NUMBER,OPERATION_NUMBER;
287                 ToolChangeFlangeToSMC;
288                 SetDO ROBOT_WORKING,low;
289             ENDIF
290         CASE 3:
291             IF DInput(START_ROBOT)=high THEN
292                 SetDO ROBOT_WORKING,high;
293                 SetAO MIRROR_NUMBER,OPERATION_NUMBER;
294                 ToolChange_FlangeToVac;
295                 SetDO ROBOT_WORKING,low;
296             ENDIF
297         CASE 10:
298             IF DInput(START_ROBOT)=high THEN
299                 SetDO ROBOT_WORKING,high;
300                 SetAO MIRROR_NUMBER,OPERATION_NUMBER;
301                 ToolChange_SchunkToFlange;
302                 SetDO ROBOT_WORKING,low;
303             ENDIF
```

Obr. 4-44 Ukázka z cyklicky volané procedury CellProgram

5 ZHODNOCENÍ A DISKUZE

Virtuální zprovoznění lze považovat za celkem komplexní problematiku. V této práci byly zpracovány všechny zadané dílčí cíle vedoucí k virtuálnímu zprovoznění. Zpracovaní těchto cílů je adekvátní vzhledem k požadavkům kladeným na diplomovou práci, avšak pro užití v praxi by bylo vhodné některé části doladit.

V případě, kdy by mělo docházet ke zprovozňování více výrobních systému s roboty společností ABB či společností jiných, bylo by vhodné vytvořit sjednocenou strukturu. Touto strukturou se myslí, že pro PLC by se tvářil stejně robot virtuální jako robot reálný. To znamená, že v rámci Process Simulatu by byl vytvořen logický blok se signálovou strukturou odpovídající reálnému robotu, a v PLC by byla vytvořená knihovna s funkčními bloky se stejným údělem. Po virtuálním zprovoznění a odladění řídicího PLC programu by pak byl v HW konfiguraci pouze přidán robot, jako další zařízení a spojen s PLC skrze sběrnici PROFINET. Následně by byla nutná pouze změněna adres řídicích signálu.

Dále by bylo vhodné upravit, respektive přidat, řídicí signály v robotu, které by byly svázaný s vnitřními signály řídicího systému robotu. Tím by se zaručilo, že například signál zprostředkovávající informaci, zda robot pracuje, by byl opravdu vystaven pouze, pokud je robot v pohybu, jelikož v nynějším programu robotu je signál vystaven, pokud je volána procedura, nikoliv pokud je robot opravdu v pohybu. To samé platí pro signály konce operace, čísla prováděné operace atd.

Taktéž by bylo vhodné, aby v HW konfiguraci řídicího PLC bylo přidáno bezpečnostní PLC, díky čemu by bylo možné rozlišovat, kde došlo k chybě a proč bezpečnost není splněna, což by poté mohlo být vyobrazeno na HMI panelu.

Posledním krokem, by mělo být rozšíření vizualizace o servisní mód, různé alarmy, případně další diagnostické prvky.

6 ZÁVĚR

Hlavním cílem této diplomové práce bylo virtuální zprovoznění výrobního systému nacházejícího se v laboratořích Ústavu výrobních strojů, systému a robotiky FSI VUT v Brně. To obnášelo tvorbu příslušného 3D modelu se všemi požadovanými operacemi v prostředí TECNOMATIX Process Simulate. Následně bylo nutné vytvořit řídicí PLC program, který byl použit pro rozpohybování a řízení 3D modelu, čímž bylo docíleno virtuálního zprovoznění. Posledním krokem byla tvorba programu robotu, který byl připraven pro nahrání do fyzického kontroléru a odladění v reálném výrobním systému.

Práce je koncipována do několika části, kde v první části je přiblížená teoretická problematika virtuálního zprovoznění. Tato teoretická část je rovněž doplněná o analýzu zadaného výrobního systému. Druhá část je zaměřená na praktické řešení této problematiky, tvorbu 3D modelu a řídicího PLC programu.

Jak bylo zmíněno, teoretická část je seznámením s problematikou virtuálního zprovoznění, kde jsou přiblíženy hlavní benefity virtuálního zprovoznění v praxi. Dále jsou zde rozebrány softwarové nástroje používané pro řešení této problematiky, což jsou softwary pro tvorbu 3D modelu a jeho následných simulací, následovány popisem nejčastěji používaných PLC ve světě průmyslové automatizace a konečně softwary umožňující propojení 3D modelu s řídicím PLC.

Následuje analýza zadaného výrobního systému, kde je popsán hardware zahrnut v jeho zástavbě. Zde se jedná převážně o technická specifika robotu ABB IRB 4400/60 a všech jeho součásti jako jsou pojezd IRBT 4003, kontrolér IRC5 a používané nástroje. Tato část je také doplněna o popis operací, které by měl výrobní systém vykonávat.

V praktické části je kladen důraz na výstižný popis tvorby 3D modelu zadaného výrobního systému, kde jsou popsány jednotlivé kinematické struktury, operace a senzory. Další, neméně důležitou částí, byla tvorba řídicího PLC programu, která je podrobně popsána soustavou okomentovaných vývojových diagramů. Nad rámec práce byla rovněž vytvořena vizualizace HMI panelu, kterou je možno použít jako ovládací rozhraní reálného výrobního systému.

Taktéž byl v této části zahrnut popis tvorby virtuálního PLC skrze software PLCSIM Advanced 2.0, ke kterému byl následně připojen 3D model výrobního systému v prostředí TECNOMATIX Process Simulate, a do kterého byl nahrán vytvořený PLC program z prostředí TIA Portal.

Poté následovalo virtuální zprovoznění. Je možno konstatovat, že PLC program plní svou funkci, nikde nedochází ke kolizím a operace jsou volány postupně, jak bylo plánováno. V případě narušení bezpečnosti je program robotu pozastaven a po opětovném splnění bezpečnosti program pokračuje tam, kde skončil. Na základě výše zmíněného, lze virtuální zprovoznění považovat za úspěšné.

Posledním cílem této diplomové práce byla tvorba programu robotu, který byl v hrubé formě získán na základě operací a programu robotu z Process Simulatu. Tento program byl importován do prostředí Robot Studio, avšak byl ochuzen o množství atributů specifikujících jednotlivé pohybové instrukce. Tyto atributy bylo nutné v deklaraci každé pohybové instrukce doplnit, čímž byly vytvořeny funkční procedury,

které byly ověřeny na modelu získaném ze zálohy reálného řídicího systému robotu. Nakonec byla v programu vytvořena procedura CellProgram starající se o volání procedur požadovaných operací a vrácení signálu, potřebných pro správný chod PLC programu.

Závěrem lze konstatovat, že v práci byly splněny všechny dílčí cíle. Práce rovněž pokládá základ pro další rozšiřování, případně pro aplikaci ve výuce, kde může být studentům přiblížena problematika virtuálního zprovoznění na množství nových aplikací, které mohou být implementovány jak do simulací a 3D modelu v prostředí TECNOMATIX Process Simulate, tak do řídícího PLC programu v prostředí TIA Portal. Dále je možné 3D model obohatit o simulace zahrnující lidskou obsluhu, což může být podkladem řešení ergonomické vhodnosti pracoviště z hlediska kvality, bezpečnosti a spolehlivosti.

7 BIBLIOGRAFIE

[1] *Automa: časopis pro automatizační techniku* [online]. Děčín: Automa - časopis pro automatizační techniku, s.r.o., 2016, **22**(5) [cit. 2019-03-10]. ISSN 1210-9592. Dostupné z: http://automa.cz/page-flip/casopis/automa/2016/05/index.html#page/1

[2] HAMPL, Lukáš. *Návrh a realizace robotických operací v systému Tecnomatix Process Simulate*. Brno, 2018. Diplomová práce. Vysoké učení technické v Brně. Vedoucí práce Václav Kaczmarczyk.

[3] SYNEK, Martin. *Optimalizace části svářecí linky karoserií auta YETI pomocí digitální simulace v 3D systému Process Simulate*. Praha, 2012. Diplomová práce. České vysoké učení technické v Praze. Vedoucí práce Cyril Tichý.

[4] ŠABLATURA, Jiří a Jan LIPINA. *ABB Robot studio - návody: laboratorní cvičení v oboru II*. První. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, Fakulta strojní, 2011. ISBN ISBN978-80-248-2734-6.

[5] LIPINA, Jan a Jiří MAREK. *Ovládání a programování robotů ABB: laboratorní cvičení oboru II*. První. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, Fakulta strojní, 2012. ISBN 978-80-248-2753-7.

[6] RobotStudio. *ABB* [online]. Švýcarsko: ABB, 2019 [cit. 2019-03-10]. Dostupné z: https://new.abb.com/products/robotics/cs/robotstudio

[7] *FastSUITE: A CENIT PRODUCT* [online]. Německo: CENIT AG., 2017 [cit. 2019-03-11]. Dostupné z: https://www.fastsuite.com/

[8] *DIGITAL FACTORY* [online]. Plzeň: Západočeská univerzita v Plzni, Fakulta strojní, Katedra průmyslového inženýrství a managementu, 2011 [cit. 2019-03-11]. Dostupné z: https://www.digipod.zcu.cz/

[9] *AXIOM TECH* [online]. Zlín: AXIOM TECH s.r.o., b.r. [cit. 2019-03-11]. Dostupné z: https://www.axiomtech.cz/

[10] HEIDARI, Ali a Oliver SALAMON. *Virtual Commissioning of an Existing Manufacturing Cell at Volvo Car Corporation Using DELMIA V6*. Gothenburg, Sweden, 2012. Master Thesis. Chalmers University of Technology.

[11] JOHANSSON, Oscar. *Testing and Evaluation of Virtual Commissioning: Case study of an existing robot cell at Scania modelled with 3D Experience*. Gothenburg, Sweden, 2018. Master Thesis. Chalmers University of Technology. Vedoucí práce Andreas Rosén.

[12] TWINKLE, Rai. Which PLC (Programmable Logic Controller) is better: SIEMENS or Allen-Bradley?. *Quora: A place to share knowledge and better understand the world* [online]. Kalifornie, USA: Quora Inc., 2019 [cit. 2019-03-13]. Dostupné z: https://www.quora.com/Which-PLC-Programmable-Logic-Controller-is-better-SIEMENS-or-Allen-Bradley

[13] Global PLC market share as of 2017, by manufacturer. *Statista: The Statistics Portal*

[online]. Hamburg, Německo: statista, 2019 [cit. 2019-03-13]. Dostupné z: https://www.statista.com/statistics/897201/global-plc-market-share-by-manufacturer/

[14] URBÁNEK, Tomáš. *Aplikace řídícího systému Simatic pro řízení balícího stroje*. Brno, 2014. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce Jan Knobloch.

[15] HROMEK, Jiří. *Komunikace OPC serverů se systémem MES (COMES)*. Brno, 2013. Diplomová práce. Vysoke učení technické v Brně, Fakulta elktrotechniky a komunikačních technologií. Vedoucí práce Jan Pásek.

[16] PATEL, Darshan. OPC-UA vs DA. *TheAutomatization.com* [online]. Theautomatization, b.r. [cit. 2019-03-13]. Dostupné z: https://theautomization.com/opc-ua-vs-da/

[17] SIMATIC S7-PLCSIM V5.4. *Siemens: Ingenuity for life* [online]. Mnichov: Siemens AG, c1996-2019 [cit. 2019-05-13]. Dostupné z: https://w3.siemens.com/mcms/simatic-controller-software/en/step7/simatic-s7-plcsim/pages/default.aspx

[18] Virtual commissioning and operator training with SIMIT. *Siemens: Ingenuity for life* [online]. Mnichov: Siemens AG, c1996-2019 [cit. 2019-05-13]. Dostupné z: https://new.siemens.com/global/en/products/automation/industry-software/simit.html

[19] IRB 4400. *ABB* [online]. Švýcarsko: ABB, 2019 [cit. 2019-03-04]. Dostupné z: https://search-ext.abb.com/library/Download.aspx?DocumentID=PR10035EN_R8&LanguageCode=en&DocumentPartId=&Action=Launch

[20] PGN-plus 200-2. *SCHUNK: Superior Clamping and Gripping* [online]. Německo: Schunk, 2019 [cit. 2019-05-18]. Dostupné z: https://schunk.com/cz_en/gripping-systems/seriesaccessory/pgn-plus/product/668-0371155-pgn-plus-200-2/

[21] SMC MHZ2-40D gripper, MHZ2 GRIPPER, PARALLEL. *SMC pneumatics* [online]. USA: SMC pneumatics, b.r. [cit. 2019-05-18]. Dostupné z: https://www.smcpneumatics.com/MHZ2-40D.html

[22] SMC VQC2300-51 valve, dbl sol, plug-in, VQC2000 SOL VALVE 5-PORT. *SMC pneumatics* [online]. USA: SMC pneumatics, b.r. [cit. 2019-05-18]. Dostupné z: https://www.smcpneumatics.com/VQC2300-51.html

[23] SWS. *SCHUNK: Superior Clamping and Gripping* [online]. Německo: Schunk, 2019 [cit. 2019-03-10]. Dostupné z: https://schunk.com/cz_cs/uchopovaci-systemy/series/sws/

[24] IRC5: Industrial Robot Controller. *ABB* [online]. Švýcarsko: ABB, 2019 [cit. 2019-03-04]. Dostupné z: https://search-ext.abb.com/library/Download.aspx?DocumentID=ROB0295EN&LanguageCode=en&DocumentPartId=&Action=Launch

[25] IRBT 4004/6004/7004: Track Motions for robots. *ABB* [online]. Švýcarsko: ABB, 2019 [cit. 2019-03-04]. Dostupné z: https://search-ext.abb.com/library/Download.aspx?DocumentID=IRBT%20PR10335EN_R2&LanguageCode=en&DocumentPartId=&Action=Launch

[26] MCV 754 QUICK. *KOVOSVIT MAS* [online]. Sezimovo Ústí: Kovosvit, 2016 [cit. 2019-

03-04]. Dostupné z: https://www.kovosvit.cz/mcv-754-quick-p3.html#main

[27] MFRS-A1-032 315 x 315. *SCHUNK: Superior Clamping and Gripping* [online]. Německo: Schunk, 2019 [cit. 2019-03-10]. Dostupné z: https://schunk.com/cz_cs/upinaci-technika/product/52137-0423118-mfrs-a1-032-315-x-315/

[28] CHROMČÍK, Adam. *Návrh virtuálního modelu robotického pracoviště* [online]. Brno, 2018 [cit. 2019-03-10]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=174311. Diplomová práce. Vysoké učení technivké v Brně. Vedoucí práce Jan Vetiška.

8 SEZNAM ZKRATEK, SYMBOLŮ, OBRÁZKŮ A TABULEK

8.1 Seznam zkratek

CAD	Computer-Aided Design
CNC	Počítačové číslicové řízení (Computer Numerical Control)
FC	Funkce
HMI	Human Machine Interface
LAD	Ladder Logic
LB	Logický blok
OLP	Off-line programming
OPC	Open Platform Communications
OPC DA	Open Platform Communications Data Acces
OPC UA	Open Platform Communications Unified Architecture
PLC	Programable Logic Controller
PLM	Product Lifecycle Management
PS	Process Simulate
RCS	Robot Controller Simulation
RS	Robot Studio
SCL	Structured Control Text
SS	Souřadný systém
TCP	Tool Center Point
TIA	Totally Integrated Automation
ÚVSSR	Ústav výrobních strojů, systému a robotiky
VUT	Vysoké učení technické v Brně

8.2 Seznam obrázku

PŘÍLOHA

Program robotu

```
MODULE MainModule
    !--ToolData
    TASK PERS tooldata toolGripper:=[TRUE,[[3.05726,-16.9492,308.991],
[0.999973,0.00616026,0.00412827,0.000025556]],[1,[0,0,100],[1,0,0,0],0,0,0]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration for tool change Flange to Schunk
    !-----------------------------------------------------------------
    LOCAL CONST robtarget TC_FlangeToSch_HOME_a:=[[0,-0.02,0],[1,0,0,0.000001],[0,0,-1,0],[0,9E
+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToSch_PosBeforeStand:=[[-393.98,1252.18,1027.1],
[0.704314,0,-0.000001,0.709889],[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToSch_MountSch:=[[-393.98,1252.18,1127.1],
[0.704314,0,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToSch_PosAfterStand_a:=[[-395.88,1492.12,1573.2],
[0.498026,-0.501967,0.498024,0.501967],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToSch_PosAfterStand_b:=[[-394.3,1292.13,1573.2],
[0.498026,-0.501967,0.498024,0.501967],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToSch_PosAfterStand_c:=[[-150.78,959.99,1573.21],
[0.707014,-0.011471,0.707014,0.011472],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToSch_PosAfterStand_d:=[[129.08,969.07,793.21],
[0.707014,-0.011471,0.707014,0.011472],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToSch_HOME_b:=[[239.95,-0.02,466.11],
[0.707107,-0.000001,0.707107,0],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration where robot operating machines MCV
    !-----------------------------------------------------------------
    LOCAL CONST robtarget PartInsert_HOME:=[[239.95,-0.02,466.1],[0.707107,0,0.707107,0.000002],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartInsert_PosBeforaPick_a:=[[4784.39,-1316.25,33.16],
[0.704308,0.002776,0.002797,0.709883],[-1,-1,-1,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartInsert_PosBeforePick_b_z_100:=[[5206.52,1460.76,1459.71],
[0.709889,0.000001,0.000001,-0.704314],[-2,0,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartInsert_Pick:=[[5206.52,1460.76,1559.71],
[0.709889,0.000001,0.000001,-0.704314],[-2,0,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartInsert_PosAfterPick_z_100:=[[5206.52,1460.76,1459.71],
[0.709889,0.000001,0.000001,-0.704314],[-2,0,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartInsert_PosBeforeMCV:=[[5085.27,-1856.98,1012.7],
[0.704314,0.000001,-0.000001,0.709889],[-1,-1,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartInsert_PosBeforePlace_z_100:=[[5090.79,-2556.95,1012.7],
[0.704314,0.000001,-0.000001,0.709889],[-1,-1,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartInsert_Place:=[[5090.79,-2556.95,1112.7],
[0.704314,0.000001,-0.000001,0.709889],[-1,-1,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartInsert_PosAfterPlace_z_100:=[[5090.79,-2556.95,1012.7],
[0.704314,0.000001,-0.000001,0.709889],[-1,-1,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartInsert_WaitingForMachining:=[[5085.27,-1856.98,1012.7],
[0.704314,0.000001,-0.000001,0.709889],[-1,-1,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget PartTakeOut_WaitingForMachining:=[[5085.27,-1856.98,1012.7],
[0.704314,0.000001,-0.000001,0.709889],[-1,-1,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartTakeOut_PosBeforePick_z_100:=[[5090.79,-2556.95,1012.7],
[0.704314,0.000001,-0.000001,0.709889],[-1,-1,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartTakeOut_Pick:=[[5090.79,-2556.95,1127.7],
[0.704314,0.000001,-0.000001,0.709889],[-1,-1,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartTakeOut_PosAfterPick_z_100:=[[5090.79,-2556.95,1012.7],
[0.704314,0.000001,-0.000001,0.709889],[-1,-1,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartTakeOut_PosAfterMCV:=[[5085.27,-1856.98,1012.7],
```

```
[0.704314,0.000001,-0.000001,0.709889],[-1,-1,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartTakeOut_PosBeforePlace_z_100:=[[5206.52,1460.76,1459.71],
[0.709889,0.000001,0.000001,-0.704314],[-2,0,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartTakeOut_Place:=[[5206.52,1460.76,1574.71],
[0.709889,0.000001,0.000001,-0.704314],[-2,0,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartTakeOut_PosAfterPlace_a:=[[5206.52,1460.76,1459.71],
[0.709889,0.000001,0.000001,-0.704314],[-2,0,0,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartTakeOut_PosAfterPlace_b:=[[4784.39,-1316.25,33.16],
[0.704308,0.002776,0.002797,0.709883],[-1,-1,-1,0],[3693,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget PartTakeOut_HOME:=[[239.95,-0.02,466.1],[0.707107,0,0.707107,0.000002],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration for tool change Schunk to Vac
    !-----------------------------------------------------------------
    LOCAL CONST robtarget TC_SchToVac_HOME_a:=[[240.45,-0.02,466.1],[0.707107,0,0.707107,0.000001],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToVac_MoveToStand_a:=[[129.08,969.07,793.21],
[0.707014,-0.011471,0.707014,0.011472],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToVac_MoveToStand_b:=[[-150.78,959.99,1573.21],
[0.707014,-0.011471,0.707014,0.011472],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToVac_PosBeforeStand_a:=[[-394.3,1292.13,1573.21],
[0.498025,-0.501967,0.498025,0.501968],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToVac_PosBeforeStand_b:=[[-395.87,1492.12,1480.21],
[0.498025,-0.501967,0.498025,0.501968],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToVac_ReleaseSchunk:=[[-395.87,1492.12,1593.21],
[0.498025,-0.501967,0.498025,0.501968],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToVac_PosBetweenStand_a:=[[-393.98,1265,957.11],
[0.704314,0,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToVac_PosBetweenStand_b:=[[-144.27,1226.64,1041.6],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToVac_MountVac:=[[-144.27,1226.64,1141.6],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToVac_PosAfterStand_a:=[[-144.27,1226.65,1320.7],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToVac_PosAfterStand_b:=[[-142.3,976.65,1320.7],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToVac_HOME_b:=[[0,-0.02,199.1],[1,0.000001,0,0.000001],[1,-1,1,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration for tool change Vac to Schunk
    !-----------------------------------------------------------------
    LOCAL CONST robtarget TC_VacToSch_HOME_a:=[[0,-0.02,199.11],[1,0.000001,0,0.000001],[1,0,-1,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSch_PosBeforeStand_a:=[[-142.3,976.65,1320.7],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSch_PosBeforeStand_b:=[[-144.27,1226.64,1250],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSch_ReleaseVac:=[[-144.27,1226.64,1340.7],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSch_PosBetweenStand_a:=[[-144.27,1226.64,957.6],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSch_PosBetweenStand_b:=[[-393.98,1252.18,1027.11],
[0.704314,0,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSch_MountSch:=[[-393.98,1252.18,1127.11],
[0.704314,0,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSch_PosAfterStand_a:=[[-395.88,1492.12,1573.21],
```

```
[0.498026,-0.501967,0.498024,0.501967],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSch_PosAfterStand_b:=[[-394.3,1292.13,1573.21],
[0.498026,-0.501967,0.498024,0.501967],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSch_MoveToHome_a:=[[-150.78,959.99,1573.21],
[0.707014,-0.011471,0.707014,0.011472],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSch_MoveToHome_b:=[[129.08,969.07,793.21],
[0.707014,-0.011471,0.707014,0.011472],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSch_HOME_b:=[[240.45,-0.02,466.1],[0.707107,0,0.707107,0.000001],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration for tool change Schunk to SMC
    !-----------------------------------------------------------------
    LOCAL CONST robtarget TC_SchToSMC_HOME_a:=[[240.45,-0.02,466.1],[0.707107,0,0.707107,0.000001],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToSMC_MoveToStand_a:=[[129.08,969.07,793.21],
[0.707014,-0.011471,0.707014,0.011472],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToSMC_MoveToStand_b:=[[-150.78,959.99,1573.21],
[0.707014,-0.011471,0.707014,0.011472],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToSMC_PosBeforeStand_a:=[[-394.3,1292.13,1573.21],
[0.498025,-0.501967,0.498025,0.501968],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToSMC_PosBeforeStand_b:=[[-395.87,1492.12,1573.21],
[0.498025,-0.501967,0.498025,0.501968],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToSMC_ReleaseSchunk:=[[-395.87,1492.12,1593.21],
[0.498025,-0.501967,0.498025,0.501968],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToSMC_PosBetweenStand_a:=[[-393.98,1265,957.11],
[0.704314,0,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToSMC_PosBetweenStand_b:=[[79.95,1262.41,1037.33],
[0.003944,0,0.000001,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToSMC_MountSMC:=[[79.95,1262.41,1137.33],
[0.003944,0,0.000001,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToSMC_PosAfterStand_a:=[[79.95,1262.41,1323.43],
[0.003944,0,0.000001,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToSMC_PosAfterStand_b:=[[81.13,1112.42,1323.43],
[0.003944,0,0.000001,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToSMC_HOME_b:=[[0,-0.02,206.1],[1,0.000001,0,0.000001],[1,-1,2,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration for tool change SMC to Schunk
    !-----------------------------------------------------------------
    LOCAL CONST robtarget TC_SMCToSch_HOME_a:=[[0,-0.02,206.1],[1,0.000001,0,0.000001],[0,0,3,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToSch_PosBeforeStand_a:=[[81.13,1112.42,1283.43],
[0.003944,0,0.000001,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToSch_PosBeforeStand_b:=[[79.95,1262.41,1283.43],
[0.003944,0,0.000001,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToSch_ReleaseSMC:=[[79.95,1262.41,1343.43],
[0.003944,0,0.000001,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToSch_PosBetweenStand_a:=[[79.95,1255.41,957.33],
[0.003944,0,0.000002,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToSch_PosBetweenStand_b:=[[-393.98,1252.18,1027.11],
[0.704314,0,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToSch_MountSch:=[[-393.98,1252.18,1127.11],
[0.704314,0,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToSch_PosAfterStand_a:=[[-395.88,1492.12,1573.21],
[0.498026,-0.501967,0.498024,0.501967],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToSch_PosAfterStand_b:=[[-392.96,1122.13,1573.21],
```

```
[0.498026,-0.501967,0.498024,0.501967],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCTOSch_MoveToHome_a:=[[-150.78,959.99,1573.21],
[0.707014,-0.011471,0.707014,0.011472],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCTOSch_MoveToHome_b:=[[129.08,969.07,793.21],
[0.707014,-0.011471,0.707014,0.011472],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToSch_HOME_b:=[[240.45,-0.02,466.1],[0.707107,0,0.707107,0.000001],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration for tool change Schunk to Flange
    !-----------------------------------------------------------------
    LOCAL CONST robtarget TC_SchToFlange_HOME_a:=[[239.95,-0.02,466.11],
[0.707107,0,0.707107,0.000002],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToFlange_MoveToStand_a:=[[129.08,969.07,793.21],
[0.707014,-0.011471,0.707014,0.011472],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToFlange_MoveToStand_b:=[[-150.78,959.99,1573.21],
[0.707014,-0.011471,0.707014,0.011472],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToFlange_PosBeforeStand_a:=[[-394.3,1292.13,1573.21],
[0.498025,-0.501967,0.498025,0.501968],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToFlange_PosBeforeStand_b:=[[-395.87,1492.12,1573.21],
[0.498025,-0.501967,0.498025,0.501968],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToFlange_ReleaseSchunk:=[[-395.87,1492.12,1593.21],
[0.498025,-0.501967,0.498025,0.501968],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToFlange_PosAfterStand:=[[-393.99,1252.18,1027.11],
[0.704314,0,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SchToFlange_HOME_b:=[[0,-0.02,0],[1,0,0,0.000001],[1,-1,1,0],[0,9E
+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration for tool change Flange to SMC
    !-----------------------------------------------------------------
    LOCAL CONST robtarget TC_FlangeToSMC_HOME_a:=[[0,-0.02,0],[1,0,0,0.000001],[0,0,-1,0],[0,9E
+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToSMC_PosBeforeStand:=[[79.95,1262.41,1037.33],
[0.003944,0,0.000002,-0.999992],[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToSMC_MountSMC:=[[79.95,1262.41,1137.33],
[0.003944,0,0.000002,-0.999992],[1,-1,-2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToSMC_PosAfterStand_a:=[[79.95,1262.41,1323.43],
[0.003944,0,0.000002,-0.999992],[1,-1,-2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToSMC_PosAfterStand_b:=[[81.13,1112.42,1323.43],
[0.003944,0,0.000002,-0.999992],[1,-1,-2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToSMC_HOME_b:=[[0,-0.02,206.1],[1,0.000001,0,0.000001],
[1,-1,-2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration where robot manipulation with cubes
    !-----------------------------------------------------------------
    LOCAL CONST robtarget C_111_HOME:=[[0,-0.02,206.1],[1,0.000001,0,0.000002],[0,0,-1,0],[0,9E
+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_111_bPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_111_pick:=[[52.04,1455.2,1128.71],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_111_aPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_111_bPP:=[[85.39,935.45,1033.09],[0.988991,0.000583,-0.147921,0.003907],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_111_bPlace:=[[-418.94,165.65,1098.7],[0.999992,0.000001,0,0.003942],
```

```
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_111_place:=[[-418.94,165.65,1198.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_111_aPlace:=[[-418.94,165.65,1098.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_111_bPP_2:=[[85.39,935.45,1033.09],
[0.988991,0.000583,-0.147921,0.003907],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_111_bPick_2:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget C_211_bPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_211_pick:=[[52.04,1455.2,1128.71],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_211_aPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_211_bPP:=[[85.39,935.45,1033.09],[0.988991,0.000583,-0.147921,0.003907],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_211_bPlace:=[[-508.94,164.94,1098.7],[0.999992,0.000001,0,0.003942],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_211_place:=[[-508.94,164.94,1198.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_211_aPlace:=[[-508.94,164.94,1098.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_211_bPP_2:=[[85.39,935.45,1033.09],
[0.988991,0.000583,-0.147921,0.003907],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_211_bPick_2:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget C_311_bPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_311_pick:=[[52.04,1455.2,1128.71],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_311_aPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_311_bPP:=[[85.39,935.45,1033.09],[0.988991,0.000583,-0.147921,0.003907],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_311_bPlace:=[[-598.94,164.23,1098.7],[0.999992,0.000001,0,0.003942],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_311_place:=[[-598.94,164.23,1198.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_311_aPlace:=[[-598.94,164.23,1098.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_311_bPP_2:=[[85.39,935.45,1033.09],
[0.988991,0.000583,-0.147921,0.003907],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_311_bPick_2:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget C_121_bPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_121_pick:=[[52.04,1455.2,1128.71],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_121_aPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_121_bPP:=[[85.39,935.45,1033.09],[0.988991,0.000583,-0.147921,0.003907],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_121_bPlace:=[[-418.19,70.65,1098.7],[0.999992,0.000001,0,0.003942],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    LOCAL CONST robtarget C_121_place:=[[-418.19,70.65,1198.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_121_aPlace:=[[-418.19,70.65,1098.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_121_bPP_2:=[[85.39,935.45,1033.09],
[0.988991,0.000583,-0.147921,0.003907],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_121_bPick_2:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget C_221_bPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_221_pick:=[[52.04,1455.2,1128.71],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_221_aPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_221_bPP:=[[85.39,935.45,1033.09],[0.988991,0.000583,-0.147921,0.003907],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_221_bPlace:=[[-508.19,69.94,1098.7],[0.999992,0.000001,0,0.003942],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_221_place:=[[-508.19,69.94,1198.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_221_aPlace:=[[-508.19,69.94,1098.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_221_bPP_2:=[[85.39,935.45,1033.09],
[0.988991,0.000583,-0.147921,0.003907],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_221_bPick_2:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget C_321_bPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_321_pick:=[[52.04,1455.2,1128.71],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_321_aPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_321_bPP:=[[85.39,935.45,1033.09],[0.988991,0.000583,-0.147921,0.003907],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_321_bPlace:=[[-598.19,69.23,1098.7],[0.999992,0.000001,0,0.003942],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_321_place:=[[-598.19,69.23,1198.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_321_aPlace:=[[-598.19,69.23,1098.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_321_bPP_2:=[[85.39,935.45,1033.09],
[0.988991,0.000583,-0.147921,0.003907],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_321_bPick_2:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget C_112_bPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_112_pick:=[[52.04,1455.2,1128.71],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_112_aPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_112_bPP:=[[85.39,935.45,1033.09],[0.988991,0.000583,-0.147921,0.003907],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_112_bPlace:=[[-418.94,165.65,1042.7],[0.999992,0.000001,0,0.003942],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_112_place:=[[-418.94,165.65,1142.7],[0.999992,0.000001,0,0.003942],
```

```
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_112_aPlace:=[[-418.94,165.65,1042.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_112_bPP_2:=[[85.39,935.45,1033.09],
[0.988991,0.000583,-0.147921,0.003907],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_112_bPick_2:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget C_212_bPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_212_pick:=[[52.04,1455.2,1128.71],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_212_aPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_212_bPP:=[[85.39,935.45,1033.09],[0.988991,0.000583,-0.147921,0.003907],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_212_bPlace:=[[-508.94,164.94,1042.7],[0.999992,0.000001,0,0.003942],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_212_place:=[[-508.94,164.94,1142.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_212_aPlace:=[[-508.94,164.94,1042.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_212_bPP_2:=[[85.39,935.45,1033.09],
[0.988991,0.000583,-0.147921,0.003907],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_212_bPick_2:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget C_312_bPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_312_pick:=[[52.04,1455.2,1128.71],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_312_aPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_312_bPP:=[[85.39,935.45,1033.09],[0.988991,0.000583,-0.147921,0.003907],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_312_bPlace:=[[-598.94,164.23,1042.7],[0.999992,0.000001,0,0.003942],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_312_place:=[[-598.94,164.23,1142.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_312_aPlace:=[[-598.94,164.23,1042.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_312_bPP_2:=[[85.39,935.45,1033.09],
[0.988991,0.000583,-0.147921,0.003907],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_312_bPick_2:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget C_122_bPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_122_pick:=[[52.04,1455.2,1128.71],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_122_aPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_122_bPP:=[[85.39,935.45,1033.09],[0.988991,0.000583,-0.147921,0.003907],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_122_bPlace:=[[-418.19,70.65,1042.7],[0.999992,0.000001,0,0.003942],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_122_place:=[[-418.19,70.65,1142.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    LOCAL CONST robtarget C_122_aPlace:=[[-418.19,70.65,1042.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_122_bPP_2:=[[85.39,935.45,1033.09],
[0.988991,0.000583,-0.147921,0.003907],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_122_bPick_2:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget C_222_bPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_222_pick:=[[52.04,1455.2,1128.71],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_222_aPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_222_bPP:=[[85.39,935.45,1033.09],[0.988991,0.000583,-0.147921,0.003907],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_222_bPlace:=[[-508.19,69.94,1042.7],[0.999992,0.000001,0,0.003942],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_222_place:=[[-508.19,69.94,1142.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_222_aPlace:=[[-508.19,69.94,1042.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_222_bPP_2:=[[85.39,935.45,1033.09],
[0.988991,0.000583,-0.147921,0.003907],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_222_bPick_2:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget C_322_bPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_322_pick:=[[52.04,1455.2,1128.71],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_322_aPick:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_322_bPP:=[[85.39,935.45,1033.09],[0.988991,0.000583,-0.147921,0.003907],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_322_bPlace:=[[-598.19,69.23,1042.7],[0.999992,0.000001,0,0.003942],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_322_place:=[[-598.19,69.23,1142.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_322_aPlace:=[[-598.19,69.23,1042.7],[0.999992,0.000001,0,0.003942],
[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_322_bPP_2:=[[85.39,935.45,1033.09],
[0.988991,0.000583,-0.147921,0.003907],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget C_322_bPick_2:=[[81.3,1455.43,1033.09],
[0.988991,0.000584,-0.147922,0.003899],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration for tool change SMC to VAC
    !-----------------------------------------------------------------
    LOCAL CONST robtarget TC_SMCToVac_HOME_a:=[[0,-0.02,206.1],[1,0.000001,0,0.000001],[0,0,-1,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToVac_PosBeforeStand_a:=[[81.13,1112.42,1223.43],
[0.003944,0,0.000001,-0.999992],[1,0,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToVac_PosBeforeStand_b:=[[79.95,1262.41,1223.43],
[0.003944,0,0.000001,-0.999992],[1,0,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToVac_ReleaseSMC:=[[79.95,1262.41,1343.43],
[0.003944,0,0.000001,-0.999992],[1,0,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToVac_PosBetweenStand_a:=[[79.95,1240.41,940.33],
[0.003944,0,0.000002,-0.999992],[1,0,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    LOCAL CONST robtarget TC_SMCToVac_PosBetweenStand_b:=[[-144.27,1226.64,1041.6],
[0.704314,0.000001,-0.000001,0.709889],[1,0,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToVac_MountVac:=[[-144.27,1226.64,1141.6],
[0.704314,0.000001,-0.000001,0.709889],[1,0,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToVac_PosAfterStand_a:=[[-144.27,1226.64,1320.7],
[0.704314,0.000001,-0.000001,0.709889],[1,0,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToVac_PosAfterStand_b:=[[-142.3,976.65,1320.7],
[0.704314,0.000001,-0.000001,0.709889],[1,0,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToVac_HOME_b:=[[0,-0.02,199.11],[1,0.000001,0,0.000002],[0,0,-1,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration where robot manipulation with cardboards
    !-----------------------------------------------------------------
    LOCAL CONST robtarget Cubes_Kart_1Home:=[[0,-0.02,199.1],[1,0.000001,0,0.000002],[0,0,-1,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_1bPick:=[[-422.82,658.13,1141.71],
[0.999992,0.000001,0,0.003942],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_1pick:=[[-422.82,658.13,1237.71],
[0.999992,0.000001,0,0.003942],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_1aPick:=[[-422.82,658.13,1141.71],
[0.999992,0.000001,0,0.003942],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_1bPlace:=[[-508.56,117.44,1141.7],
[0.999992,0.000001,0,0.003942],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_1place:=[[-508.56,117.44,1191.7],
[0.999992,0.000001,0,0.003942],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_1aPlace:=[[-508.57,117.44,1141.7],
[0.999992,0.000001,0,0.003942],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_1Home2:=[[0,-0.02,199.1],[1,0.000001,0,0.000002],[1,-1,0,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    LOCAL CONST robtarget Cubes_Kart_2Home:=[[0,-0.02,199.1],[1,0.000001,0,0.000002],[0,0,-1,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_2bPick:=[[-422.82,658.13,1141.71],
[0.999992,0.000001,0,0.003942],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_2pick:=[[-422.82,658.13,1242.71],
[0.999992,0.000001,0,0.003942],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_2aPick:=[[-422.82,658.13,1035.71],
[0.999992,0.000001,0,0.003942],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_2bPlace:=[[-508.56,117.44,1035.7],
[0.999992,0.000001,0,0.003942],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_2place:=[[-508.56,117.44,1135.7],
[0.999992,0.000001,0,0.003942],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_2aPlace:=[[-508.56,117.44,1035.7],
[0.999992,0.000001,0,0.003942],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget Cubes_Kart_2Home2:=[[0,-0.02,199.1],[1,0.000001,0,0.000002],[1,-1,0,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration for tool change VAC to SMC
    !-----------------------------------------------------------------
    LOCAL CONST robtarget TC_VacToSMC_HOME_a:=[[0,-0.02,199.11],[1,0.000001,0,0.000001],[1,0,-1,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSMC_PosBeforeStand_a:=[[-142.3,976.65,1220.7],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSMC_PosBeforeStand_b:=[[-144.27,1226.64,1220.7],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSMC_ReleaseVac:=[[-144.27,1226.64,1340.7],
```

```
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSMC_PosBetweenStand_a:=[[-144.27,1226.64,950.6],
[0.704314,0,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSMC_PosBetweenStand_b:=[[79.95,1262.41,1037.33],
[0.003944,0,0.000002,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSMC_MountSMC:=[[79.95,1262.41,1137.33],
[0.003944,0,0.000002,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSMC_PosAfterStand_a:=[[79.95,1262.41,1323.43],
[0.003944,0,0.000002,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSMC_PosAfterStand_b:=[[81.13,1112.42,1323.43],
[0.003944,0,0.000002,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToSMC_HOME_b:=[[0,-0.02,206.1],[1,0.000001,0,0.000001],[1,-1,2,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration for tool change SMC to Flange
    !-----------------------------------------------------------------
    LOCAL CONST robtarget TC_SMCToFlange_HOME_a:=[[0,-0.02,206.1],[1,0.000001,0,0.000001],[0,0,3,0],
[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToFlange_PosBeforeStand_a:=[[81.13,1112.42,1223.43],
[0.003944,0,0.000001,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToFlange_PosBeforeStand_b:=[[79.95,1262.41,1223.43],
[0.003944,0,0.000001,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToFlange_ReleaseSMC:=[[79.95,1262.41,1343.43],
[0.003944,0,0.000001,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToFlange_PosAfterStand:=[[79.95,1262.41,950.33],
[0.003944,0,0.000002,-0.999992],[1,-1,2,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_SMCToFlange_HOME_b:=[[0,-0.02,0],[1,0,0,0.000001],[1,-1,2,0],[0,9E
+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration for tool change Flange to VAC
    !-----------------------------------------------------------------
    LOCAL CONST robtarget TC_FlangeToVac_HOME_a:=[[0,-0.02,0],[1,0,0,0.000001],[0,0,-1,0],[0,9E
+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToVac_PosBeforeStand:=[[-144.27,1226.64,1041.6],
[0.704314,0.000001,-0.000001,0.709889],[1,0,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToVac_MountVac:=[[-144.27,1226.64,1141.6],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToVac_PosAfterStand:=[[-144.28,1226.65,1320.7],
[0.704314,0,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToVac_PosAfterStand_b:=[[-142.3,976.65,1320.7],
[0.704314,0,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_FlangeToVac_HOME_b:=[[0,-0.02,199.1],[1,0.000001,0,0.000001],
[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------
    !--Trajectory points declaration for tool change VAC to Flange
    !-----------------------------------------------------------------
    LOCAL CONST robtarget TC_VacToFlange_HOME_a:=[[0,-0.02,199.1],[1,0.000001,0,0.000001],
[0,0,-1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToFlange_PosBeforeStand_a:=[[-142.3,976.65,1220.7],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,0,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToFlange_PosBeforeStand_b:=[[-144.27,1226.64,1220.7],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToFlange_ReleaseVac:=[[-144.27,1226.64,1340.7],
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToFlange_PosAfterStand:=[[-144.27,1226.64,950],
```

```
[0.704314,0.000001,-0.000001,0.709889],[1,-1,1,0],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
    LOCAL CONST robtarget TC_VacToFlange_HOME_b:=[[0,-0.02,0],[1,0,0,0.000001],[1,0,0,0],[0,9E+09,9E↵
+09,9E+09,9E+09,9E+09]];

    !-----------------------------------------------------------------------
    !--Signal variables declaration
    !-----------------------------------------------------------------------
    VAR signalai OPERATION_NUMBER;
    VAR signalao MIRROR_NUMBER;
    VAR signaldo OPERATION_DONE;
    VAR signaldi START_ROBOT;
    VAR signaldo ROBOT_WORKING;

    !xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

    !-----------------------------------------------------------------------
    !--Main routine
    !-----------------------------------------------------------------------

    PROC main()
        CellProgram;
    ENDPROC

    !xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
    !xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

    !--Procedure CellProgram where all required subprocedures are called
    PROC CellProgram()
        TEST AInput(OPERATION_NUMBER)
        !--Function Called by variable declaration
        !--Called operations are dependent on the OPERATION_NUMBER input signal
        CASE 1:
            !--This cases are representing called operation's numbers
            IF DInput(START_ROBOT)=high THEN
                !--Operation can be called only when input signal START_ROBOT is set
                SetDO ROBOT_WORKING,high;
                SetAO MIRROR_NUMBER,OPERATION_NUMBER;
                ToolChangeFlangeToSchunk;
                !--Before an operation is called ROBOT_WORKING and MIRROR_NUMBER signals are set
                SetDO ROBOT_WORKING,low;
                !--After operation ends a ROBOT_WORKING signal is set to false
            ENDIF
        !--The same approach is used in another cases
        CASE 2:
            IF DInput(START_ROBOT)=high THEN
                SetDO ROBOT_WORKING,high;
                SetAO MIRROR_NUMBER,OPERATION_NUMBER;
                ToolChangeFlangeToSMC;
                SetDO ROBOT_WORKING,low;
            ENDIF
        CASE 3:
            IF DInput(START_ROBOT)=high THEN
                SetDO ROBOT_WORKING,high;
                SetAO MIRROR_NUMBER,OPERATION_NUMBER;
                ToolChange_FlangeToVac;
                SetDO ROBOT_WORKING,low;
            ENDIF
        CASE 10:
```

```
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        ToolChange_SchunkToFlange;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 12:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        ToolChange_SchunkToSMC;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 13:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        ToolChange_SchunkToVac;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 20:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        ToolChange_SMCToFlange;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 21:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        ToolChange_SMCToSchunk;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 23:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        ToolChange_SMCToVac;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 30:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        ToolChange_VacToFlange;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 31:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        ToolChange_VacToSchunk;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 32:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
```

```
            SetAO MIRROR_NUMBER,OPERATION_NUMBER;
            ToolChange_VacToSMC;
            SetDO ROBOT_WORKING,low;
        ENDIF
    CASE 101:
        IF DInput(START_ROBOT)=high THEN
            SetDO ROBOT_WORKING,high;
            SetAO MIRROR_NUMBER,OPERATION_NUMBER;
            PartInsert;
            SetDO ROBOT_WORKING,low;
        ENDIF
    CASE 102:
        IF DInput(START_ROBOT)=high THEN
            SetDO ROBOT_WORKING,high;
            SetAO MIRROR_NUMBER,OPERATION_NUMBER;
            PartTakeOut;
            SetDO ROBOT_WORKING,low;
        ENDIF
    CASE 201:
        IF DInput(START_ROBOT)=high THEN
            SetDO ROBOT_WORKING,high;
            SetAO MIRROR_NUMBER,OPERATION_NUMBER;
            Cubes_1_1_1;
            SetDO ROBOT_WORKING,low;
        ENDIF
    CASE 202:
        IF DInput(START_ROBOT)=high THEN
            SetDO ROBOT_WORKING,high;
            SetAO MIRROR_NUMBER,OPERATION_NUMBER;
            Cubes_2_1_1;
            SetDO ROBOT_WORKING,low;
        ENDIF
    CASE 203:
        IF DInput(START_ROBOT)=high THEN
            SetDO ROBOT_WORKING,high;
            SetAO MIRROR_NUMBER,OPERATION_NUMBER;
            Cubes_3_1_1;
            SetDO ROBOT_WORKING,low;
        ENDIF
    CASE 204:
        IF DInput(START_ROBOT)=high THEN
            SetDO ROBOT_WORKING,high;
            SetAO MIRROR_NUMBER,OPERATION_NUMBER;
            Cubes_1_2_1;
            SetDO ROBOT_WORKING,low;
        ENDIF
    CASE 205:
        IF DInput(START_ROBOT)=high THEN
            SetDO ROBOT_WORKING,high;
            SetAO MIRROR_NUMBER,OPERATION_NUMBER;
            Cubes_2_2_1;
            SetDO ROBOT_WORKING,low;
        ENDIF
    CASE 206:
        IF DInput(START_ROBOT)=high THEN
            SetDO ROBOT_WORKING,high;
            SetAO MIRROR_NUMBER,OPERATION_NUMBER;
            Cubes_3_2_1;
```

```
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 207:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        Cubes_1_1_2;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 208:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        Cubes_2_1_2;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 209:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        Cubes_3_1_2;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 210:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        Cubes_1_2_2;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 211:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        Cubes_2_2_2;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 212:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        Cubes_3_2_2;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 250:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        Cubes_Kart_1;
        SetDO ROBOT_WORKING,low;
    ENDIF
CASE 251:
    IF DInput(START_ROBOT)=high THEN
        SetDO ROBOT_WORKING,high;
        SetAO MIRROR_NUMBER,OPERATION_NUMBER;
        Cubes_Kart_2;
        SetDO ROBOT_WORKING,low;
    ENDIF
```

```
    ENDTEST
ENDPROC

!-----------------------------------------------------------------------
!--Procedures where robot operating machine MCV
!-----------------------------------------------------------------------

!--Procedure where robot puts part into the machine MCV
!--Procedure has the same process as procedure before (ToolChangeFlangeToSchunk)
!--With only one small difference - Pulse signal (DO3) is used to close jaws on the gripper
!--and the Pulse signal (DO4) is used to open jaws on the defined gripper
PROC PartInsert()
    SetDO OPERATION_DONE,low;
    MoveJ PartInsert_HOME,v1000,z200,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ PartInsert_PosBeforaPick_a,v1000,z200,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ PartInsert_PosBeforePick_b_z_100,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveL PartInsert_Pick,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;

    !--Close tool
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;

    MoveJ PartInsert_PosAfterPick_z_100,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ PartInsert_PosBeforeMCV,v1000,z10,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveL PartInsert_PosBeforePlace_z_100,v500,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveL PartInsert_Place,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;

    !--Open tool
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;

    MoveL PartInsert_PosAfterPlace_z_100,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveL PartInsert_WaitingForMachining,v500,z10,toolGripper_PGN_200\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

 !--Procedure where robot takes part out of the machine MCV
PROC PartTakeOut()
    SetDO OPERATION_DONE,low;
    MoveJ PartTakeOut_WaitingForMachining,v500,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveL PartTakeOut_PosBeforePick_z_100,v500,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveL PartTakeOut_Pick,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL PartTakeOut_PosAfterPick_z_100,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveL PartTakeOut_PosAfterMCV,v500,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ PartTakeOut_PosBeforePlace_z_100,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveL PartTakeOut_Place,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL PartTakeOut_PosAfterPlace_a,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ PartTakeOut_PosAfterPlace_b,v1000,z200,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ PartTakeOut_HOME,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

!xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
!-----------------------------------------------------------------------
!--Procedures where cubes are manipulated
!-----------------------------------------------------------------------

PROC Cubes_1_1_1()
    SetDO OPERATION_DONE,low;
    MoveJ C_111_HOME,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_111_bPick,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_111_pick,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL C_111_aPick,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_111_bPP,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_111_bPlace,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_111_place,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL C_111_aPlace,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_111_bPP_2,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_111_bPick_2,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

PROC Cubes_2_1_1()
    SetDO OPERATION_DONE,low;
    MoveJ C_211_bPick,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_211_pick,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL C_211_aPick,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_211_bPP,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_211_bPlace,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_211_place,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL C_211_aPlace,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_211_bPP_2,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_211_bPick_2,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

PROC Cubes_3_1_1()
    SetDO OPERATION_DONE,low;
    MoveJ C_311_bPick,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_311_pick,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL C_311_aPick,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_311_bPP,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_311_bPlace,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_311_place,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL C_311_aPlace,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_311_bPP_2,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_311_bPick_2,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC
```

```
PROC Cubes_1_2_1()
    SetDO OPERATION_DONE,low;
    MoveJ C_121_bPick,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_121_pick,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL C_121_aPick,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_121_bPP,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_121_bPlace,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_121_place,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL C_121_aPlace,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_121_bPP_2,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_121_bPick_2,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

PROC Cubes_2_2_1()
    SetDO OPERATION_DONE,low;
    MoveJ C_221_bPick,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_221_pick,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL C_221_aPick,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_221_bPP,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_221_bPlace,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_221_place,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL C_221_aPlace,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_221_bPP_2,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_221_bPick_2,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

PROC Cubes_3_2_1()
    SetDO OPERATION_DONE,low;
    MoveJ C_321_bPick,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_321_pick,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL C_321_aPick,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_321_bPP,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_321_bPlace,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_321_place,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL C_321_aPlace,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_321_bPP_2,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_321_bPick_2,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

PROC Cubes_1_1_2()
    SetDO OPERATION_DONE,low;
    MoveJ C_112_bPick,v1000,fine,toolGripper\WObj:=Workobject_1;
```

```
    MoveL C_112_pick,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL C_112_aPick,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_112_bPP,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_112_bPlace,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_112_place,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL C_112_aPlace,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_112_bPP_2,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_112_bPick_2,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

PROC Cubes_2_1_2()
    SetDO OPERATION_DONE,low;
    MoveJ C_212_bPick,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_212_pick,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL C_212_aPick,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_212_bPP,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_212_bPlace,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_212_place,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL C_212_aPlace,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_212_bPP_2,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_212_bPick_2,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

PROC Cubes_3_1_2()
    SetDO OPERATION_DONE,low;
    MoveJ C_312_bPick,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_312_pick,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL C_312_aPick,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_312_bPP,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_312_bPlace,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_312_place,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL C_312_aPlace,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_312_bPP_2,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_312_bPick_2,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

PROC Cubes_1_2_2()
    SetDO OPERATION_DONE,low;
    MoveJ C_122_bPick,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_122_pick,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL C_122_aPick,v100,fine,toolGripper\WObj:=Workobject_1;
```

```
    MoveJ C_122_bPP,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_122_bPlace,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_122_place,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL C_122_aPlace,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_122_bPP_2,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_122_bPick_2,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

PROC Cubes_2_2_2()
    SetDO OPERATION_DONE,low;
    MoveJ C_222_bPick,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_222_pick,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL C_222_aPick,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_222_bPP,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_222_bPlace,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_222_place,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL C_222_aPlace,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_222_bPP_2,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_222_bPick_2,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

PROC Cubes_3_2_2()
    SetDO OPERATION_DONE,low;
    MoveJ C_322_bPick,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_322_pick,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveL C_322_aPick,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_322_bPP,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_322_bPlace,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL C_322_place,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL C_322_aPlace,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ C_322_bPP_2,v1000,z200,toolGripper\WObj:=Workobject_1;
    MoveJ C_322_bPick_2,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

!xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

!-----------------------------------------------------------------------
!--Procedures where cardboards are manipulated
!-----------------------------------------------------------------------

PROC Cubes_Kart_1()
    SetDO OPERATION_DONE,low;
    MoveJ Cubes_Kart_1Home,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveJ Cubes_Kart_1bPick,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveL Cubes_Kart_1pick,v100,fine,toolVacuum\WObj:=Workobject_1;
```

```
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveJ Cubes_Kart_1aPick,v100,z200,toolVacuum\WObj:=Workobject_1;
    MoveJ Cubes_Kart_1bPlace,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveL Cubes_Kart_1place,v100,fine,toolVacuum\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL Cubes_Kart_1aPlace,v100,fine,toolVacuum\WObj:=Workobject_1;
    MoveJ Cubes_Kart_1Home2,v1000,fine,toolVacuum\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

PROC Cubes_Kart_2()
    SetDO OPERATION_DONE,low;
    MoveJ Cubes_Kart_2Home,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveJ Cubes_Kart_2bPick,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveL Cubes_Kart_2pick,v100,fine,toolVacuum\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO3;
    WaitTime 0.5;
    MoveJ Cubes_Kart_2aPick,v100,z200,toolVacuum\WObj:=Workobject_1;
    MoveJ Cubes_Kart_2bPlace,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveL Cubes_Kart_2place,v100,fine,toolVacuum\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO4;
    WaitTime 0.5;
    MoveL Cubes_Kart_2aPlace,v100,fine,toolVacuum\WObj:=Workobject_1;
    MoveJ Cubes_Kart_2Home2,v1000,fine,toolVacuum\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

!xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

!-----------------------------------------------------------------------
!--Procedures where tools are changed
!-----------------------------------------------------------------------

!--Procedure where robot mounts Schunk Tool
PROC ToolChangeFlangeToSchunk()
    SetDO OPERATION_DONE,low; !--Setting output signal OPERATION_DONE as false
    MoveJ TC_FlangeToSch_HOME_a,v1000,fine,toolChanger\WObj:=Workobject_1;
    MoveJ TC_FlangeToSch_PosBeforeStand,v1000,z1,toolChanger\WObj:=Workobject_1;
    MoveL TC_FlangeToSch_MountSch,v100,fine,toolChanger\WObj:=Workobject_1;
    !--Several move instructions
    !--Point to point or linear movement, velocity, zone, tool with reference frame

    PulseDO Local_IO_0_DO1;
    WaitTime 0.5;
    !--Pulse signal to attach tool

    MoveL TC_FlangeToSch_PosAfterStand_a,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_FlangeToSch_PosAfterStand_b,v300,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_FlangeToSch_PosAfterStand_c,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_FlangeToSch_PosAfterStand_d,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_FlangeToSch_HOME_b,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    !--Next movement instructions with the same attributes as on the beginning
    SetDO OPERATION_DONE,high; !--Setting output signal OPERATION_DONE as false
ENDPROC

!--Procedure where tool is changed from Schunk To Vac
```

```
!--Signal pulses (DO2) are used to detach the tool
!--Signal pulses (D01) are used to attach the tool
PROC ToolChange_SchunkToVac()
    SetDO OPERATION_DONE,low;
    MoveJ TC_SchToVac_HOME_a,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SchToVac_MoveToStand_a,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SchToVac_MoveToStand_b,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SchToVac_PosBeforeStand_a,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SchToVac_PosBeforeStand_b,v300,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveL TC_SchToVac_ReleaseSchunk,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO2;
    WaitTime 0.5;
    MoveL TC_SchToVac_PosBetweenStand_a,v100,fine,toolChanger\WObj:=Workobject_1;
    MoveL TC_SchToVac_PosBetweenStand_b,v300,fine,toolChanger\WObj:=Workobject_1;
    MoveL TC_SchToVac_MountVac,v100,fine,toolChanger\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO1;
    WaitTime 0.5;
    MoveL TC_SchToVac_PosAfterStand_a,v100,fine,toolVacuum\WObj:=Workobject_1;
    MoveJ TC_SchToVac_PosAfterStand_b,v300,fine,toolVacuum\WObj:=Workobject_1;
    MoveJ TC_SchToVac_HOME_b,v1000,fine,toolVacuum\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

!--Procedure where tool is changed from Vac To Schunk
PROC ToolChange_VacToSchunk()
    SetDO OPERATION_DONE,low;
    MoveJ TC_VacToSch_HOME_a,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveJ TC_VacToSch_PosBeforeStand_a,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveJ TC_VacToSch_PosBeforeStand_b,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveL TC_VacToSch_ReleaseVac,v100,fine,toolVacuum\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO2;
    WaitTime 0.5;
    MoveL TC_VacToSch_PosBetweenStand_a,v100,fine,toolChanger\WObj:=Workobject_1;
    MoveL TC_VacToSch_PosBetweenStand_b,v300,fine,toolChanger\WObj:=Workobject_1;
    MoveL TC_VacToSch_MountSch,v100,fine,toolChanger\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO1;
    WaitTime 0.5;
    MoveL TC_VacToSch_PosAfterStand_a,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_VacToSch_PosAfterStand_b,v300,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_VacToSch_MoveToHome_a,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_VacToSch_MoveToHome_b,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_VacToSch_HOME_b,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

!--Procedure where tool is changed from Schunk To SMC
PROC ToolChange_SchunkToSMC()
    SetDO OPERATION_DONE,low;
    MoveJ TC_SchToSMC_HOME_a,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SchToSMC_MoveToStand_a,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SchToSMC_MoveToStand_b,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SchToSMC_PosBeforeStand_a,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveL TC_SchToSMC_PosBeforeStand_b,v300,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveL TC_SchToSMC_ReleaseSchunk,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO2;
    WaitTime 0.5;
    MoveL TC_SchToSMC_PosBetweenStand_a,v100,fine,toolChanger\WObj:=Workobject_1;
    MoveL TC_SchToSMC_PosBetweenStand_b,v300,fine,toolChanger\WObj:=Workobject_1;
```

```
    MoveL TC_SchToSMC_MountSMC,v100,fine,toolChanger\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO1;
    WaitTime 0.5;
    MoveL TC_SchToSMC_PosAfterStand_a,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ TC_SchToSMC_PosAfterStand_b,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveJ TC_SchToSMC_HOME_b,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

!--Procedure where tool is changed from SMC To Schunk
PROC ToolChange_SMCToSchunk()
    SetDO OPERATION_DONE,low;
    MoveJ TC_SMCToSch_HOME_a,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveJ TC_SMCToSch_PosBeforeStand_a,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveJ TC_SMCToSch_PosBeforeStand_b,v300,fine,toolGripper\WObj:=Workobject_1;
    MoveL TC_SMCToSch_ReleaseSMC,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO2;
    WaitTime 0.5;
    MoveL TC_SMCToSch_PosBetweenStand_a,v100,fine,toolChanger\WObj:=Workobject_1;
    MoveL TC_SMCToSch_PosBetweenStand_b,v300,fine,toolChanger\WObj:=Workobject_1;
    MoveL TC_SMCToSch_MountSch,v100,fine,toolChanger\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO1;
    WaitTime 0.5;
    MoveL TC_SMCToSch_PosAfterStand_a,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SMCToSch_PosAfterStand_b,v300,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SMCTOSch_MoveToHome_a,v500,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SMCTOSch_MoveToHome_b,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SMCToSch_HOME_b,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

!--Procedure where tool is changed from Schunk To Flange
PROC ToolChange_SchunkToFlange()
    SetDO OPERATION_DONE,low;
    MoveJ TC_SchToFlange_HOME_a,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SchToFlange_MoveToStand_a,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SchToFlange_MoveToStand_b,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SchToFlange_PosBeforeStand_a,v1000,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveJ TC_SchToFlange_PosBeforeStand_b,v300,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    MoveL TC_SchToFlange_ReleaseSchunk,v100,fine,toolGripper_PGN_200\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO2;
    WaitTime 0.5;
    MoveL TC_SchToFlange_PosAfterStand,v100,fine,toolChanger\WObj:=Workobject_1;
    MoveJ TC_SchToFlange_HOME_b,v1000,fine,toolChanger\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

!--Procedure where tool is changed from Flange To SMC
PROC ToolChangeFlangeToSMC()
    SetDO OPERATION_DONE,low;
    MoveJ TC_FlangeToSMC_HOME_a,v1000,fine,toolChanger\WObj:=Workobject_1;
    MoveJ TC_FlangeToSMC_PosBeforeStand,v1000,fine,toolChanger\WObj:=Workobject_1;
    MoveL TC_FlangeToSMC_MountSMC,v100,fine,toolChanger\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO1;
    WaitTime 0.5;
    MoveL TC_FlangeToSMC_PosAfterStand_a,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ TC_FlangeToSMC_PosAfterStand_b,v300,fine,toolGripper\WObj:=Workobject_1;
    MoveJ TC_FlangeToSMC_HOME_b,v1000,fine,toolGripper\WObj:=Workobject_1;
```

```
    SetDO OPERATION_DONE,high;
ENDPROC

!--Procedure where tool is changed from SMC To VAC
PROC ToolChange_SMCToVac()
    SetDO OPERATION_DONE,low;
    MoveJ TC_SMCToVac_HOME_a,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveJ TC_SMCToVac_PosBeforeStand_a,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL TC_SMCToVac_PosBeforeStand_b,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL TC_SMCToVac_ReleaseSMC,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO2;
    WaitTime 0.5;
    MoveL TC_SMCToVac_PosBetweenStand_a,v100,fine,toolChanger\WObj:=Workobject_1;
    MoveL TC_SMCToVac_PosBetweenStand_b,v300,fine,toolChanger\WObj:=Workobject_1;
    MoveL TC_SMCToVac_MountVac,v1000,fine,toolChanger\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO1;
    WaitTime 0.5;
    MoveL TC_SMCToVac_PosAfterStand_a,v100,fine,toolVacuum\WObj:=Workobject_1;
    MoveL TC_SMCToVac_PosAfterStand_b,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveJ TC_SMCToVac_HOME_b,v1000,fine,toolVacuum\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

!--Procedure where tool is changed from Vac To SMC
PROC ToolChange_VacToSMC()
    SetDO OPERATION_DONE,low;
    MoveJ TC_VacToSMC_HOME_a,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveJ TC_VacToSMC_PosBeforeStand_a,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveJ TC_VacToSMC_PosBeforeStand_b,v1000,fine,toolVacuum\WObj:=Workobject_1;
    MoveL TC_VacToSMC_ReleaseVac,v100,fine,toolVacuum\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO2;
    WaitTime 0.5;
    MoveL TC_VacToSMC_PosBetweenStand_a,v100,fine,toolChanger\WObj:=Workobject_1;
    MoveL TC_VacToSMC_PosBetweenStand_b,v300,fine,toolChanger\WObj:=Workobject_1;
    MoveL TC_VacToSMC_MountSMC,v100,fine,toolChanger\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO1;
    WaitTime 0.5;
    MoveL TC_VacToSMC_PosAfterStand_a,v100,fine,toolGripper\WObj:=Workobject_1;
    MoveJ TC_VacToSMC_PosAfterStand_b,v300,fine,toolGripper\WObj:=Workobject_1;
    MoveJ TC_VacToSMC_HOME_b,v1000,fine,toolGripper\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

!--Procedure where tool is changed from SMC To Flange
PROC ToolChange_SMCToFlange()
    SetDO OPERATION_DONE,low;
    MoveJ TC_SMCToFlange_HOME_a,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveJ TC_SMCToFlange_PosBeforeStand_a,v1000,fine,toolGripper\WObj:=Workobject_1;
    MoveL TC_SMCToFlange_PosBeforeStand_b,v300,fine,toolGripper\WObj:=Workobject_1;
    MoveL TC_SMCToFlange_ReleaseSMC,v100,fine,toolGripper\WObj:=Workobject_1;
    PulseDO Local_IO_0_DO2;
    WaitTime 0.5;
    MoveL TC_SMCToFlange_PosAfterStand,v100,fine,toolChanger\WObj:=Workobject_1;
    MoveJ TC_SMCToFlange_HOME_b,v1000,fine,toolChanger\WObj:=Workobject_1;
    SetDO OPERATION_DONE,high;
ENDPROC

!--Procedure where tool is changed from Flange To VAC
```

```
    PROC ToolChange_FlangeToVac()
        SetDO OPERATION_DONE,low;
        MoveJ TC_FlangeToVac_HOME_a,v1000,fine,toolChanger\WObj:=Workobject_1;
        MoveJ TC_FlangeToVac_PosBeforeStand,v1000,fine,toolChanger\WObj:=Workobject_1;
        MoveL TC_FlangeToVac_MountVac,v100,fine,toolChanger\WObj:=Workobject_1;
        PulseDO Local_IO_0_DO1;
        WaitTime 0.5;
        MoveL TC_FlangeToVac_PosAfterStand,v100,fine,toolVacuum\WObj:=Workobject_1;
        MoveL TC_FlangeToVac_PosAfterStand_b,v300,fine,toolVacuum\WObj:=Workobject_1;
        MoveJ TC_FlangeToVac_HOME_b,v1000,fine,toolVacuum\WObj:=Workobject_1;
        SetDO OPERATION_DONE,high;
    ENDPROC

    !--Procedure where tool is changed from Vac To Flange
    PROC ToolChange_VacToFlange()
        SetDO OPERATION_DONE,low;
        MoveJ TC_VacToFlange_HOME_a,v1000,fine,toolVacuum\WObj:=Workobject_1;
        MoveJ TC_VacToFlange_PosBeforeStand_a,v1000,fine,toolVacuum\WObj:=Workobject_1;
        MoveL TC_VacToFlange_PosBeforeStand_b,v300,fine,toolVacuum\WObj:=Workobject_1;
        MoveL TC_VacToFlange_ReleaseVac,v100,fine,toolVacuum\WObj:=Workobject_1;
        PulseDO Local_IO_0_DO2;
        WaitTime 0.5;
        MoveL TC_VacToFlange_PosAfterStand,v100,fine,toolChanger\WObj:=Workobject_1;
        MoveJ TC_VacToFlange_HOME_b,v1000,fine,toolChanger\WObj:=Workobject_1;
        SetDO OPERATION_DONE,high;
    ENDPROC

    !xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

ENDMODULE
```