ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA ŘÍDICÍ TECHNIKY

Virtuální řízená soustava diskrétního nebo hybridního typu se systémy PLC a SCADA

Bc. Šiška Matěj

DIPLOMOVÁ PRÁCE

2010

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Matěj Šiška**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Virtuální řízená soustava diskrétního nebo hybridního typu se systémy PLC a SCADA**

Pokyny pro vypracování:

Seznamte se s programovatelnými automaty Tecomat Foxtrot, s jejich vývojovým systémem Mosaic, vizualizačním systémem Reliance a se zásadami programování PLC podle standardu IEC EN 61 131-3.
Zhodnoťte vlastnosti obvyklých technologických procesů a soustav diskrétního a hybridního typu a zvolte typického představitele řízené soustavy, vhodné pro realizaci názorného modelu pro výuku.
Analyzujte funkce zvolené soustavy a navrhněte její algoritmický popis, jako podklad pro vytvoření modelu.
Realizujte model soustavy programem systémů PLC Tecomat a SCADA Reliance 4, včetně ovládacího panelu.
Navrhněte typické algoritmy a program PLC pro řízení modelu soustavy v typických režimech a pro její technickou diagnostiku, využijte komunikační funkce obou systémů, rozhodněte o fyzické realizaci a navrhněte metodickou příručku.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Martin Hlinovský, Ph.D.

Platnost zadání: do konce letního semestru 2009/10

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 27. 2. 2009

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 5.5.2010 .....

Podpis

Poděkování

Zde bych rád vyjádřil poděkování lidem, kteří přispěli ke zdárnému dokončení této diplomové práce.

V prvé řadě rodině, která se mnou měla trpělivost a podporovala mne, čímž jsem se mohl maximálně věnovat této práci a díky nimž mohu studovat a i úspěšně dokončit tuto školu.

Dále lidem působícím ve školství.

- pan Ing. Martin Hlinovský, Ph.D., vedoucí diplomové práce - jenž mi nechal volné ruce při realizaci vlastního řešení
- pan Ing. Ladislav Šmejkal, CSc., oponent této práce - který mi věnoval dostatek konzultací včetně několika společných cest do Sezimova Ústí, kde mě rovněž uvedl a představil a za zasvěcené rozhovory nad problematikou
- pan Ing. Jan Fuka, učitel odborných předmětů VOŠ, SŠ, COP Sezimovo Ústí - za jeho čas věnovaný našim návštěvám, dodaným technickým podkladům, cenným připomínkám při realizaci a další spolupráci při nasazování výsledků práce do výuky

Název práce

Virtuální řízená soustava diskrétního nebo hybridního typu se systémy PLC a SCADA

Anotace

Tato diplomová práce se zabývá virtualizací fyzického modelu procesu obrábění. Na začátku práce popisuji používané vývojové nástroje (SW Mosaic, SW Reliance, norma IEC EN 61 131-3). Dále rozebírám typy technologických procesů a jejich příklady. Na základě toho vybírám vhodný model a popisuji jeho funkčnost. Hlavní částí jsou pak kapitoly věnující se virtualizaci sestavy a jejímu následnému řízení. Zpracování logiky sestavy a řízení realizuji v SW Mosaic, ovládací obrazovky v SW Reliance. Na závěr provedu zhodnocení splnění požadavků zadání a přínos této práce.

Klíčová slova

programovatelný automat (PLC), vizualizační systém (SCADA), řízená soustava, diskrétní proces, model, učební pomůcka, řídicí algoritmus

Topic of the thesis

Virtual Discrete or Hybrid Controlled Plant with systems PLC or SCADA

Abstract

This thesis deals with the virtualization of a physical model of the machining process. At the beginning of the work I describe how to use development tools (Mosaic, SW, SW Reliance, IEC EN 61 131-3). Further, I analyze the types of technological processes and their examples. On this basis, I select the appropriate model and describe its functionality. The main parts are the chapters devoted to virtualization report and its subsequent management. Logic processing and assembly procedure implemented in an SW Mosaic software controls the screen in Reliance. In conclusion, conduct an evaluation of compliance with the assignment and contribution of this paper.

Keywords

programmable logic controller (PLC), visual system (SCADA), control of the system, discrete process, model, learning tool, the control algorithm

Obsah

1 Úvod

Trendem ve výuce je co nejvíce se přiblížit reálným problémům, tj. praxi. V případě technických škol toto dobře zastupují nejrůznější modely reálných fyzikálních soustav, např. modely výrobních linek, obráběcích procesů apod. Často se jedná o vysoce kvalitní a profesionální modely dodávané přímo od výrobců daných technologií. Od toho se rovněž i odvíjí nemalé finanční částky spojené s pořízením těchto učebních pomůcek

Proto je potřeba se k nim chovat tak, aby co nejvíce posloužily a měly co nejdelší životnost. Jedním z možných řešení je přenést je v přesné kopii do počítače, tj. virtualizovat je a veškeré nezbytné úkony realizovat na této virtuální sestavě. Studenti tak pracují s věrnou virtuální kopií fyzické sestavy, kde si mohou vyzkoušet jednotlivé funkce sestavy a zároveň ladit své algoritmy realizující její řízení. Nedochází tak ke zbytečným havarijním stavům fyzických sestav, což značně prodlužuje jejich využitelnost. Rovněž to umožňuje pracovat na daném problému většímu počtu studentů, jelikož reálný model není tak vytížen.

Tyto virtuální soustavy jsou však rozšířené spíše na úrovních matematických modelů, využívající např. SW Matlab Simulink apod. Mým cílem je ubírat se cestou průmyslového řízení, tj. simulaci sestavy realizovat na virtuálním PLC a k ovládání použít technologické obrazovky. Toto vše budu provozovat na jednom PC, resp. notebooku.

2 Prostředky pro řešení

Tuto kapitolu věnuji prostředkům, které jsou doporučeny k řešení tématu. Lze si je rozdělit na:

- PLC Tecomat
- Mosaic – vývojové prostředí pro programování PLC Tecomat
- Reliance - vývojové prostředí pro tvorbu vizualizačních obrazovek
- Norma 61 131 – standart pro programování řídících systémů

2.1 PLC Tecomat

Řízení technologií, linek a strojů ve všech oblastech průmyslu, mimo jiné např. v dopravě, energetice – zde všude jsou nasazovány a využívány programovatelné automaty Tecomat firmy Teco a. s. Systémy Tecomat jsou svým kvalitním provedením a širokými programovými možnostmi lehce zařaditelné do standardní úrovně světových PLC. Řada Tecomat je tvořena různě výkonnými systémy. Od malých kompaktních systémů TC400, *viz Obr. č. 2-1*, mající řádově jednotky vstupů a výstupu přes větší kompaktní systémy typu TC500, TC600 a TC650. Vrcholem a pro nasazení ve velkých aplikací jsou určeny modulární systémy NS950 a TC700, *viz Obr. č. 2-2*. Shrnutí jednotlivých zástupců řady Tecomat je přehledně zpracováno v následující tabulce, *viz Tab. č. 2-1*.

Obr. č. 2-1 – PLC Tecomat TC400

Obr. č. 2-2 – PLC Tecomat TC700

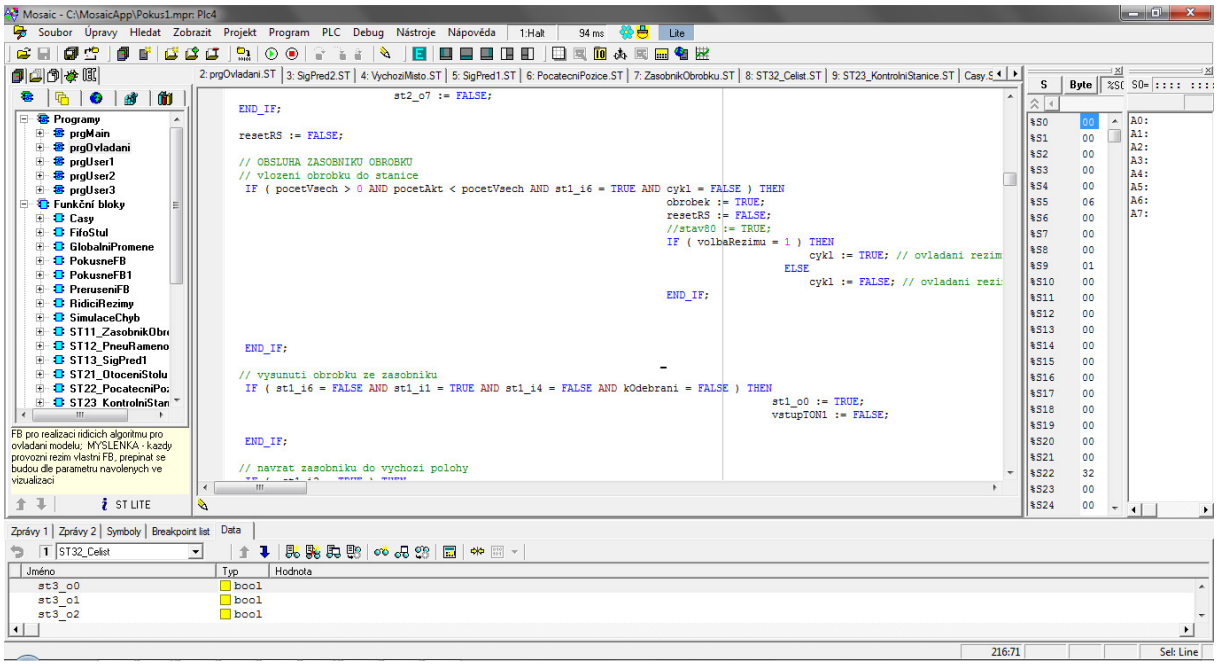
Typová řada	*Počet binárních I/O*	*Počet analogových I/O*	*Komunikační kanály*	*Display / klávesnice*	*Paměť programu*
TC400	6 / 4	2 / -	2x serial	externí	32 kB
TC500	20 / 20	4 / 4	2x serial	integrovaný	32 kB
TC600	48 / 44	24 / 8	3x serial	externí	32 kB
TC650	48 / 44	24 / 8	3x serial, 1x Ethernet	externí	64 + 64 kB
Foxtrot	až 128 DI / DO	až 80 AI / 80 AO	2x seriál, 1x CIB, 1x Ethernet	externí	192 + 64 kB
TC700	přes 6500	až 3300 / 700	až 10x seriál, 2x Ethernet, 1x USB	externí	192 + 64 kB

Tab. č. 2-1 – Typové řady PLC Tecomat a jejich základní parametry

Všechny systémy Tecomat je možné programovat z vlastního vývojového prostředí Mosaic, podrobnější popis *viz kapitola 2.2*. Základním a pro všechny systémy Tecomat společným programovacím jazykem je firemní mnemokód (odpovídá jazyku typu IL). Ten zajišťuje vzájemnou kompatibilitu programování a umožňuje přenositelnost kódu v rámci celé řady Tecomat. Novější typy PLC, jenž jsou již založené na 32 bitových procesorech, je možné programovat na úrovni jazyka strukturovaného textu (ST) dle mezinárodní normy IEC EN 61131-3. Tvorba programu dle této normy zpřehledňuje a zefektivňuje programování. Zároveň rozšiřuje možnost použít nové podpůrné nástroje na rozdíl od předchozích starších typů CPU, kde ani tyto možnosti nebyly podporovány.

Další technické detaily viz [1].

2.2 Vývojové prostředí Mosaic



Obr. č. 2-3 – Ukázka vývojového prostředí Mosaic

Vývojové prostředí Mosaic, *viz Obr. č. 2-3*, umožňuje vytvářet aplikační programy pro PLC Tecomat. Programování je možné v jazyce instrukcí (mnemokódu). Pro novější systémy s 32 bitovými procesory (Tecomat TC650 a TC700) lze vytvářet programy v jazycích podle normy IEC EN 61131-3:

- **IL** – jazyk instrukcí
- **ST** – strukturovaný text
- **LD** – reléová schémata
- **FBD** – funkční bloky

Součástí prostředí Mosaic je i řada nástrojů usnadňujících vývoj a ladění aplikací:

- **IEC manažer** - grafickou deklaraci všech prvků programu PLC
- **Inspektor POU** - nástroj pro ladění programu PLC
- **Simulátor PLC** - dovoluje ladit programy bez nutnosti připojení reálného hardwaru, simulovat lze všechny typy PLC Tecomat a Tecoreg
  - k simulátoru lze připojit i vizualizační software Reliance a ladit celou aplikaci na jednom PC
- **PanelMaker** - nástroj na tvorbu dialogů pro operátorské panely

- **PanelSim** - simulátor operátorských panelů
- **PIDMaker** - nástroj pro ladění a návrh PID regulátorů
- **GraphMaker** - nástroj pro podporu ladění a diagnostiku řízeného systému
- **Softwarová konfigurace PLC** - konfigurační nástroj umožňující výběr typu PLC a definici konkrétní sestavy
- **Definice sítě PLC** - nástroj umožňuje grafickou formou vytvořit vazby mezi PLC v rámci projektu
- **GraphMaker** - nástroj pro podporu ladění a diagnostiku řízeného systému
- **Projektový manažer** - komfortní správa projektu, archivace a zálohování projektu
- **hypertextová a kontextová nápověda** - kompletní dokumentace k systémům Tecomat a Tecoreg ve formátu pdf

Vývojové prostředí umožňuje provádět on-line úpravy programu PLC bez zastavení řízení. Tato funkce vyžaduje podporu ze strany CPU a týká se pouze systémů Tecomat TC700 a TC650. Kromě změn řídícího algoritmu lze přidávat a mazat proměnné, měnit jejich datový typ apod. Přepnutí mezi starým a novým programem je velmi rychlé, typicky méně než desetinu doby potřebné pro zpracování programu. Společně s možností vyměňovat I/O moduly PLC bez zastavení řízení je on-line změna programu důležitou podmínkou pro minimalizaci ztrát vzniklých odstavením řídícího systému při údržbě SW i HW PLC.

Komunikace Mosaicu se řídícím systémem je možná pomocí:

- sériové linky
- Ethernetu
- USB

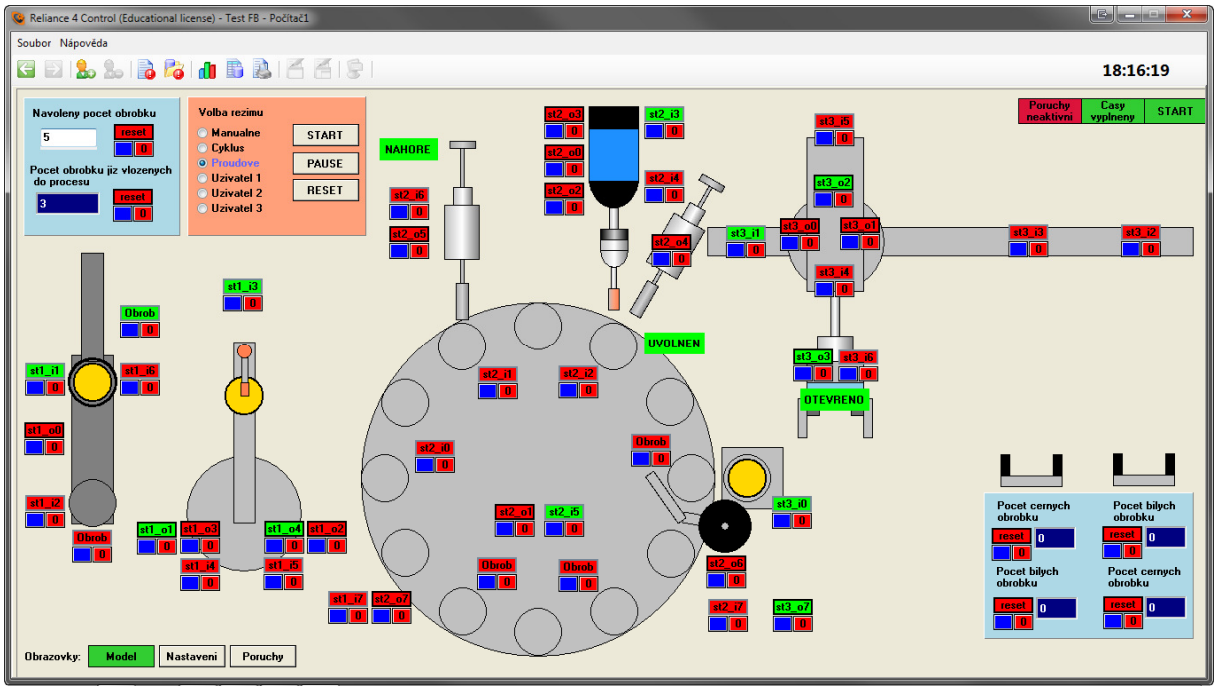
Dále je zahrnuta i podpora pro:

- vytáčené připojení přes telefonní nebo GSM modem
- Wi-fi

Pro své potřeby využiji především moduly pro podporu ladění a simulátor PLC.

Podrobnosti viz [1].

2.3 Vývojové prostředí Reliance



Obr. č. 2-4 – Ukázka vývojového prostředí Reliance

Reliance, *viz Obr. č. 2-4*, je moderní bohatě škálovatelný, bezpečný a robustní SCADA/HMI systém určený pro monitorování a ovládání i velmi rozsáhlých průmyslových technologií. Vývojové prostředí RAD nabízí řadu základních funkcí již předdefinovaných, slouží pro správu rozsáhlých aplikací, umožňuje projekt vyexportovat do webového formátu apod.

Vývojový balík Reliance se skládá z dílčích modulů:

- **Reliance Design** - vývojové prostředí určené pro tvorbu vizualizačních projektů
- **Reliance Runtime (modul)**
  - zajišťuje běh vizualizačního projektu na počítači koncového uživatele
  - získává data z komunikačních driverů, z jiných runtime modulů apod.
  - zobrazuje tato data v podobě vizualizačních obrazovek, zobrazení a kvitaci (potvrzení) aktuálních alarmů
  - zobrazení a tisk historických dat
- **Reliance Server** - umožňuje připojení a obsluhu webových klientů Reliance J
- **Reliance Runtime Server** – obsahuje funkce mající Reliance Runtime

a Reliance Server

- **Reliance J (Webový klient)** - applet napsaný v jazyce Java, který umožňuje spustit vizualizační projekt v prostředí webového prohlížeče

Z pohledu koncového uživatele přináší Reliance zvýšení kvality a produktivity výrobního procesu tím, že minimalizuje výpadky technologie včasným varováním obsluhy (obsluha alarmů) a zabudovanou redundancí datových toků. V případě poruchy je zde možnost zpětně analyzovat příčiny výpadku nebo poruchy technologie (funkce Postmort). Správa reportů dokáže automaticky generovat reporty z výroby a ty následně zasílat na předem nastavené e-mailové schránky. Dálkový přístup a dohled nad technologií je možný v podstatě nepřetržitě pomocí moderních komunikačních kanálů – např. internet či přes GSM – SMS.

Pro své potřeby využiji především modul Reliance Design a Runtime (modul)

Tento SW je licencovaný. Bez licence je možné použít max. 25 proměnných, což pro mé potřeby absolutně nedostačuje. Toto jsem vyřešil zapůjčením si školní licence, která umožňuje pracovat až s 500ti proměnnými, což již postačuje.

Podrobnosti viz [3].

2.4 Programování dle normy IEC EN 61 131-3

2.4.1 Norma IEC 61 131

Norma IEC 61 131 se zaměřuje na programovatelné řídící systémy. Skládá se z pěti základních částí, , *viz Tab. č. 2-2*, které definují požadavky na moderní řídící systém. Norma je nezávislá na výrobcích a má širokou mezinárodní podporu.

V ČR byly přijaty jednotlivé části této normy pod následujícími čísly a názvy:

Číslo normy	Název normy
ČSN EN 61 131-1	Programovatelné řídící jednotky - Část 1: Všeobecné informace
ČSN EN 61 131-2	Programovatelné řídící jednotky - Část 2: Požadavky na zařízení a zkoušky
ČSN EN 61 131-3	Programovatelné řídící jednotky - Část 3: Programovací jazyky
ČSN EN 61 131-4	Programovatelné řídící jednotky - Část 4: Podpora uživatelů
ČSN EN 61 131-5	Programovatelné řídící jednotky - Část 5: Komunikace
ČSN EN 61 131-7	Programovatelné řídící jednotky - Část 7: Programování fuzzy řízení

Tab. č. 2-2 – Tabulka se seznamem částí normy IEC 61 131 pro ČR

V Evropské unii jsou tyto normy přijaty pod číslem EN IEC 61 131

2.4.2 Základní myšlenka normy 61 131-3

Standardizuje programovací jazyky pro průmyslovou automatizaci, tj. specifikuje syntaxe a sémantiky souboru programovacích jazyků, včetně obecného modelu a strukturujícího jazyka. Je výsledkem práce sedmi mezinárodních společností a ve svém souhrnu obsahuje cca. 200 stran a cca. 60 tabulek. Tato norma byla přijata jako směrnice u většiny významných výrobců PLC. Základní prvky normy lze rozdělit na:

- Datové typy
- Proměnné
- Konfigurace, zdroje a úlohy
- Programové organizační jednotky
    - Funkce
    - Funkční bloky
    - Programy
- Programovací jazyky

2.4.2.1 Datové typy

Běžné datové typy jsou Bool, Byte, Word, Int, Real, Date, Time, String atd. Uživatelské datové typy pak vznikají odvozením z těchto běžných typů. Datové typy rovněž slouží jako prevence chyb při samotném počátku tvorby projektu

2.4.2.2 Proměnné

Z důvodu vysokého stupně hardwarové nezávislosti se proměnné přiřazují k hardwarovým adresám explicitně, tj. pouze v konfiguracích, zdrojích nebo programech.

Proměnné jsou běžně lokálního typu, tj. nejsou viditelné mimo organizační jednotku, ve které byly deklarovány. Tentýž název proměnné může být tedy použit bez obav v jiné organizační jednotce. To rovněž eliminuje možnost vzniku chyb. Mají-li mít proměnné globální působnost, musí být nedeklarovány jako globální *VAR_GLOBAL*.

2.4.2.3 Konfigurace, zdroje a úlohy

Konfigurace (Configuration) je na nejvyšší úrovni SW řešení problému řízení. Je závislá na konkrétním řídícím systému, typu a uspořádání HW.

Jeden nebo více tzv. zdrojů (Resource) je pak definováno v rámci jedné konfigurace. Zdroj můžeme považovat za zařízení, které je schopno vykonávat IEC programy.

Zdroje se skládají z jedné nebo více úloh (Task). Úlohy obstarávají řízení provádění souboru programů a/nebo funkčních bloků. Úlohy mohou být prováděny buď periodicky nebo po vzniku spouštěcí události.

Program je složen z různých SW prvků (funkce, funkční bloky apod.), které jsou zapsány v některém z jazyků definovaném v normě. Funkce a funkční bloky jsou základní stavební kameny obsahující datové struktury a algoritmus.

2.4.2.4 Programové organizační jednotky

Programové organizační jednotky (Program Organization Units, POU) je souhrnný název pro:

- funkce (function, FUN)

- funkční bloky (function block, FB)
- programy (program, PROG)

2.4.2.4.1 Funkce

Funkce mohou byt standardní (ADD pro sčítání, SQRT pro odmocninu apod.), jejž definuje norma, a uživatelem definované.

Pokud je funkce volána se stejnými vstupními parametry, musí dávat vždy stejný výsledek (hodnotu). Funkce vrací pouze jeden výsledek.

2.4.2.4.2 Funkční bloky

Mají definované rozhranní (vstupní a výstupní proměnné), skryté vnitřní proměnné a na rozdíl od funkcí si pamatují informace. Navenek tedy působí jako tzv. černá skříňka. Funkční bloky lze psát v libovolném programovacím jazyku v rámci normy. Po definici funkčního bloku může být používán opakovaně a např. i mimo projekt. Funkční blok může vracet více než jeden výsledek.

2.4.2.4.3 Programy

Program je složen z různých SW prvků (funkce, funkční bloky), které jsou zapsány v některém z jazyků definovaném v normě. Funkce a funkční bloky jsou základní stavební kameny obsahující datové struktury a algoritmus.

2.4.2.5 Programovací jazyky

Norma definuje čtyři programovací jazyky, *viz Obr. č. 2-5*. Ty mají přesně definovanou sémantiku a syntaxi. Lze je dělit na:

Textové jazyky

- **IL – Instruction List** – jazyk seznamu instrukcí
- **ST – Structured Text** – jazyk strukturovaného textu

Grafické jazyky

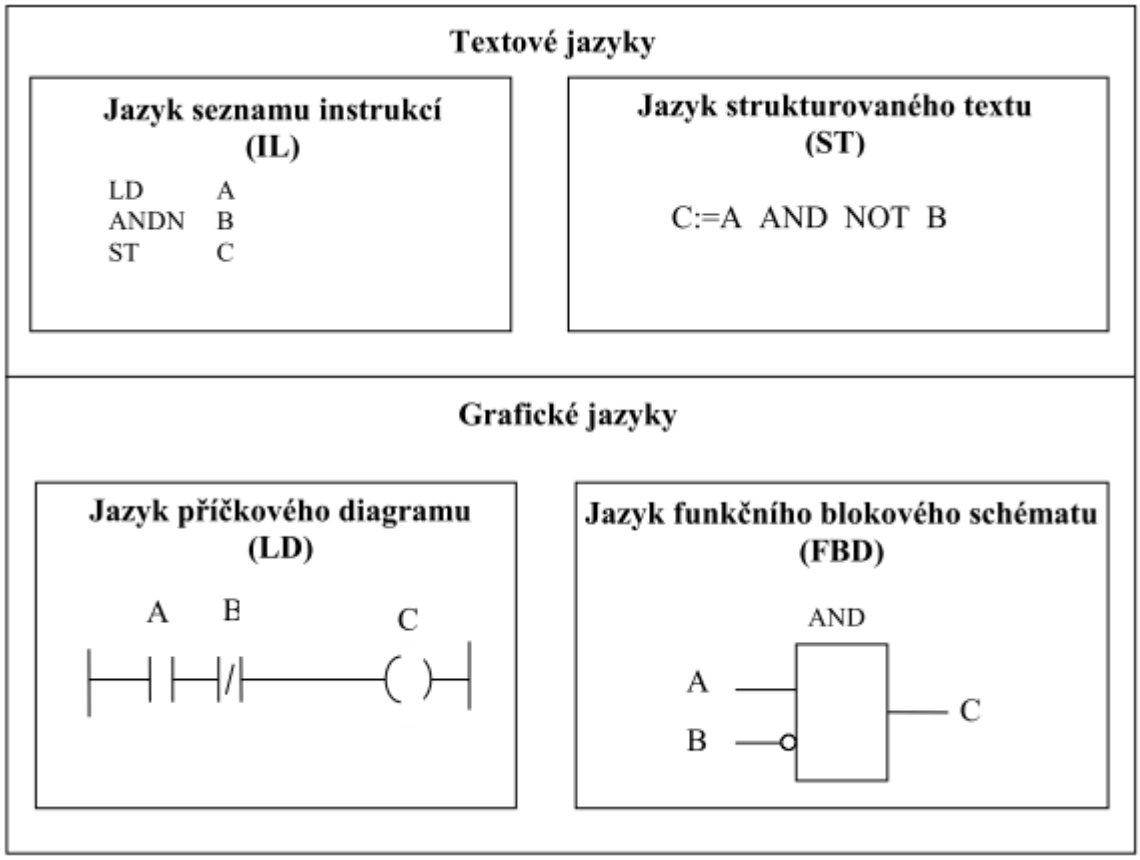
- **LD – Ladder Diagram** – jazyk příčkového diagramu (jazyk kontaktních schémat)
- **FBD – Function Block Diagram** – jazyk funkčního blokového schématu

IL (Instruction List) - evropský protějšek LD
- připomíná textový programovací jazyk assembler

ST (Structured Text) - výkonný vyšší programovací jazyk
- vychází z jazyků Ada, Pascal a C
- obsahuje prvky moderního programovacího jazyka (IF-THEN-ELSE, CASE OF, FOR, WHILE, REPEAT apod.)
- vhodný pro definování komplexních funkčních bloků

LD (Ladder Diagram) - má původ v USA
- vychází z grafické reprezentace reléové logiky

FBD (Function Block Diagram) - blízký procesnímu průmyslu
- vyjadřuje chování POU pomocí grafických vazeb

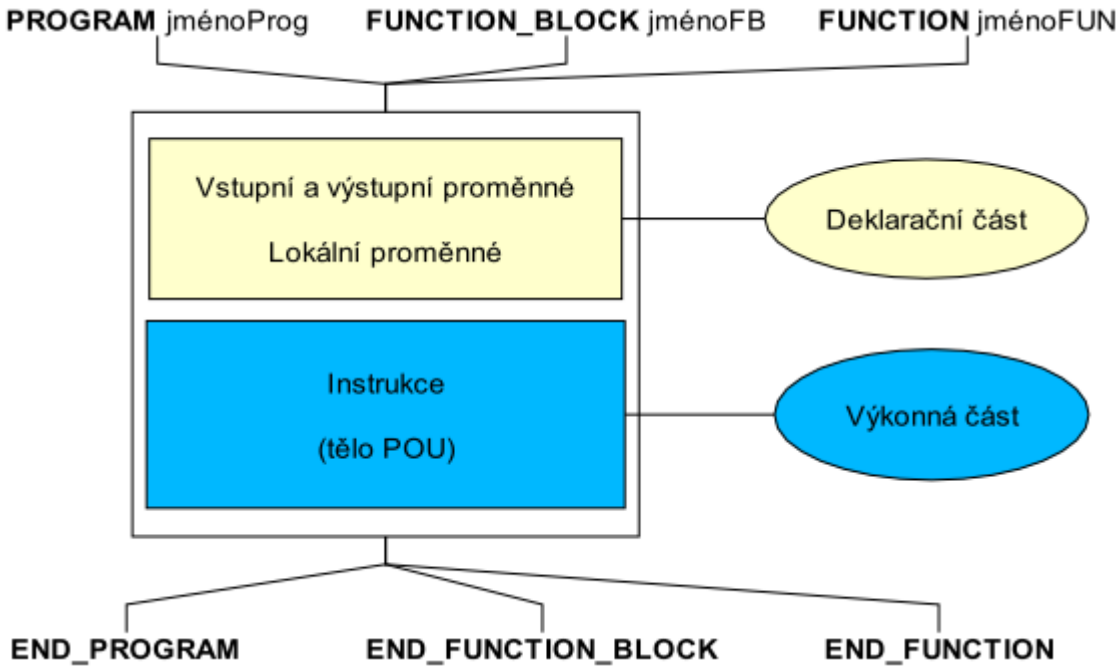


Obr. č. 2-5 – Příklad logické funkce AND NOT ve 4 základních jazycích normy

2.4.3 Stavební bloky POU

Každá POU, *viz Obr. č. 2-6*, se skládá ze dvou základních částí:

- Část deklarační – definice proměnných pro činnost POU
- Část výkonná – příkazy pro realizaci algoritmu



Obr. č. 2-6 – Základní bloková struktura POU

2.4.3.1 Deklarační část POU

Obsahuje seznam proměnných potřebných pro činnost POU, *viz Obr. č. 2-7*. Proměnné slouží pro ukládání a zpracování informací a jsou definovány jménem proměnné a jejím datovým typem.

Kromě datových typů můžeme proměnné ještě dělit na globální a lokální. Globální proměnné jsou definovány mimo POU a jsou viditelné, tj. použitelné v rámci celého projektu. Lokální proměnné jsou definovány uvnitř POU a je možné je využívat pouze v tomto POU.

Z pohledu předávání parametrů z/do POU, rozlišujeme proměnné na vstupní a výstupní. Pomocí vstupních proměnných do POU data a informace vstupují, pomocí výstupních z ní vystupují.

```
FUNCTION_BLOCK PrikladDeklaraceProm

  VAR_INPUT                          (* vstupní proměnné *)
    logPodminka          : BOOL;     (* binární hodnota *)
  END_VAR
  VAR_OUTPUT                         (* výstupní proměnné *)
    vysledek             : INT;      (* celočíselná hodnota se znaménkem *)
  END_VAR
  VAR                                (* lokální proměnné *)
    kontrolniSoucet      : UINT;     (* celočíselná hodnota *)
    mezivysledek         : REAL;     (* reálná hodnota *)
  END_VAR

END_FUNCTION_BLOCK
```

Obr. č. 2-7 – Příklad deklarační části POU – deklarace proměnných

2.4.3.2 Výkonná část POU

Obsahuje příkazy a instrukce realizující logiku algoritmu, *viz Obr. č. 2-8*. Rovněž může obsahovat volání dalších POU, kdy mohou být předávány parametry pro volané funkce a funkční bloky.

```
PROGRAM ExampleStrings
  VAR
    message      : STRING := '';  // empty string
    value        : INT;
    valid        : BOOL;
  END_VAR

  IF valid THEN
    message := 'Temperature is ';
    message := CONCAT(IN1 := message, IN2 := INT_TO_STRING(value));
    message := message + ' [C]';
  ELSE
    message := 'Temperature is not available !';
  END_IF;
  message := message + '$0D$0A';
END_PROGRAM
```

Obr. č. 2-8 – Příklad výkonné části POU – realizace logiky algoritmu

Podrobnosti viz [2].

3 Technologické procesy

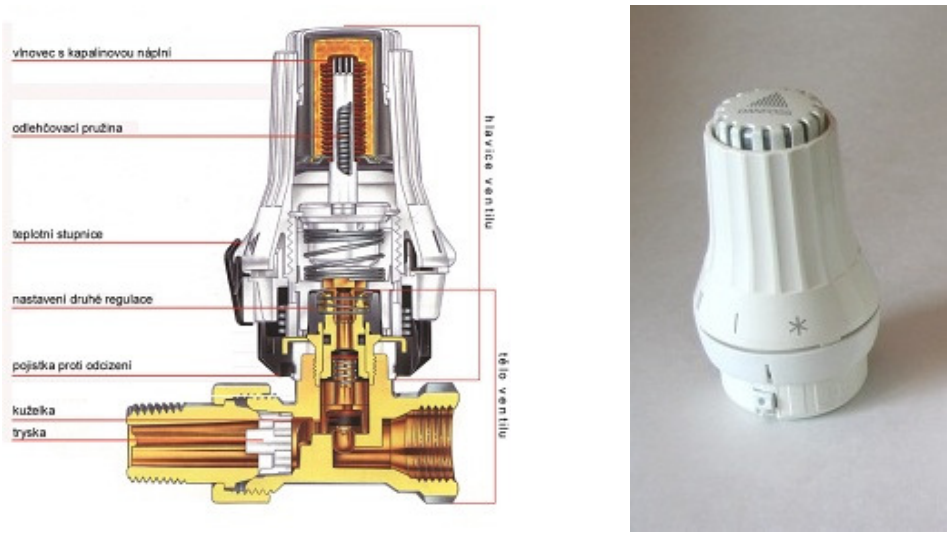
Technologický proces je označení pro popis posloupnosti (kroků), jak se mění stav (přechod ze stavu do stavu) daného zařízení (stroje apod.). Můžeme dělit na tři základní typy:

- Spojitý proces
- Diskrétní proces
- Hybridní proces

3.1 Spojitý proces

V tomto procesu se vyskytuje fyzikální veličiny (např. teplota, průtok, tlak) mající více než dva stavy, tj. plynule přecházejí ze současného stavu do následujícího. Čistě kompletních spojitých procesů moc není, jedná se spíše o jednoduché procesy nebo o dílčí procesy komplexnějších systému.

Jako příklad spojitého procesu může být regulace teploty v místnosti. Regulace teploty je realizována termostatem, *viz Obr. č. 3-1*. Termostaty jsou mechanické a elektrické. Princip funkce je takový, že se využívá roztažnosti kapalin, plynu či pevných materiálů. V případě regulace teploty v místnosti je řídící veličinou teplota vzduchu v místnosti. Roste-li teplota vzduchu, roztahuje se pozvolna materiál uvnitř termostatu, který postupně uzavírá přívod teplé vody do topného tělesa, teplota v místnosti se dál nezvyšuje. Začne-li se vzduch v místnosti ochlazovat, materiál se začne smršťovat a otevírá přívod teplé vody do topného tělesa, teplota v místnosti dál neklesá.



Obr. č. 3-1 – Struktura termostatu a reálný výrobek

3.2 Diskrétní proces

Popisuje proces, který má pouze omezený počet stavů (obvykle dva), např. zapnuto/vypnuto, otevřeno/zavřeno, přítomnost/nepřítomnost. Tyto procesy jsou o něco typičtější a jsou takto realizovány celé komplexní systémy.

Zástupcem tohoto procesu je např. použití vrtačky ve výrobním procesu, *viz Obr. č. 3-2*. Je-li přítomen obrobek na pracovišti vrtačky, zapne se vřeteno vrtačky. Na pneumatickém pístu sjede až do své dolní polohy, zde se zastaví a vyvrtá daný otvor. Po vyvrtání vyjede zpět do své horní polohy a vypne vřeteno. Obrobek je uvolněn k přesunu na další pracoviště. Z popisu plyne, že nás vždy zajímají jen krajní polohy vrtačky, přítomnost obrobku.

Obr. č. 3-2 – Ukázka výrobní linky s pracovištěm vrtačky

3.3 Hybridní proces

Je to kombinace obou předchozích typů procesů, tj. procesu spojitého a diskrétního. Většina reálných procesů patří právě do této skupiny.

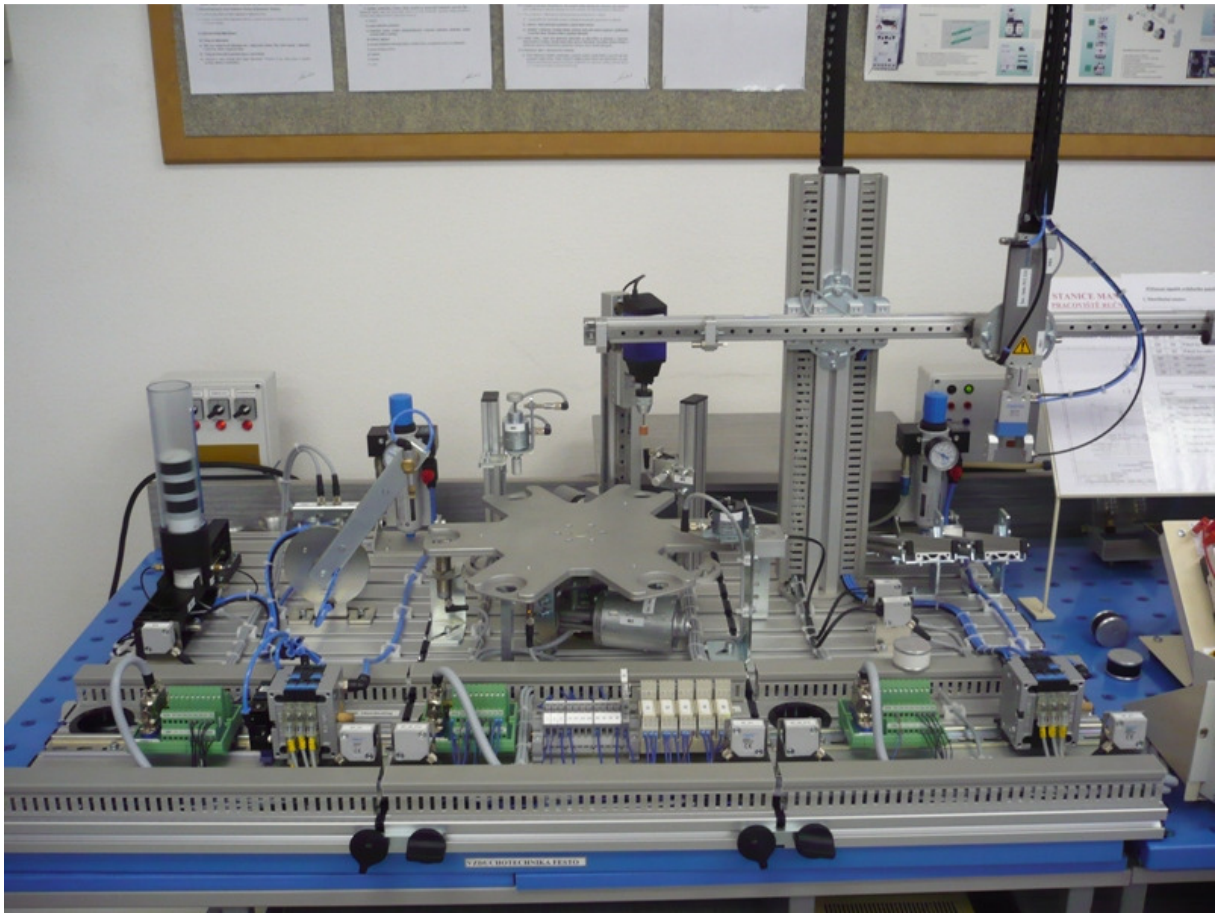
Mezi tyto procesy patří např. automatická pračka. Jako spojitý proces zde funguje regulace teploty vody v pračce na principu termostatu, popis výše. Diskrétní proces je zde reprezentován jako přechod mezi jednotlivými stavy pračky (napouštění vody, ohřev, praní, ždímání, vypouštění vody atd.).

4 Popis zvolené soustavy

Dle konzultace s vedoucím své diplomové práce jsem si vybral soustavu popisující čistě diskrétní proces. Jedná se o výukový model technologického procesu obrábění umístěný v Centru odborné přípravy Sezimovo Ústí (COPSU), *viz Obr. č. 4-1*.

Abych sestavu lépe pochopil, zavítal jsem osobně do COPSU. Zde jsem si sestavu prohlédl, pořídil detailní fotografie a rovněž zhotovil video záznam z chodu sestavy. Dále jsem prostudoval podklady dodané p. Ing. Janem Fukou, který má v COPSU soustavu při výuce na starosti.

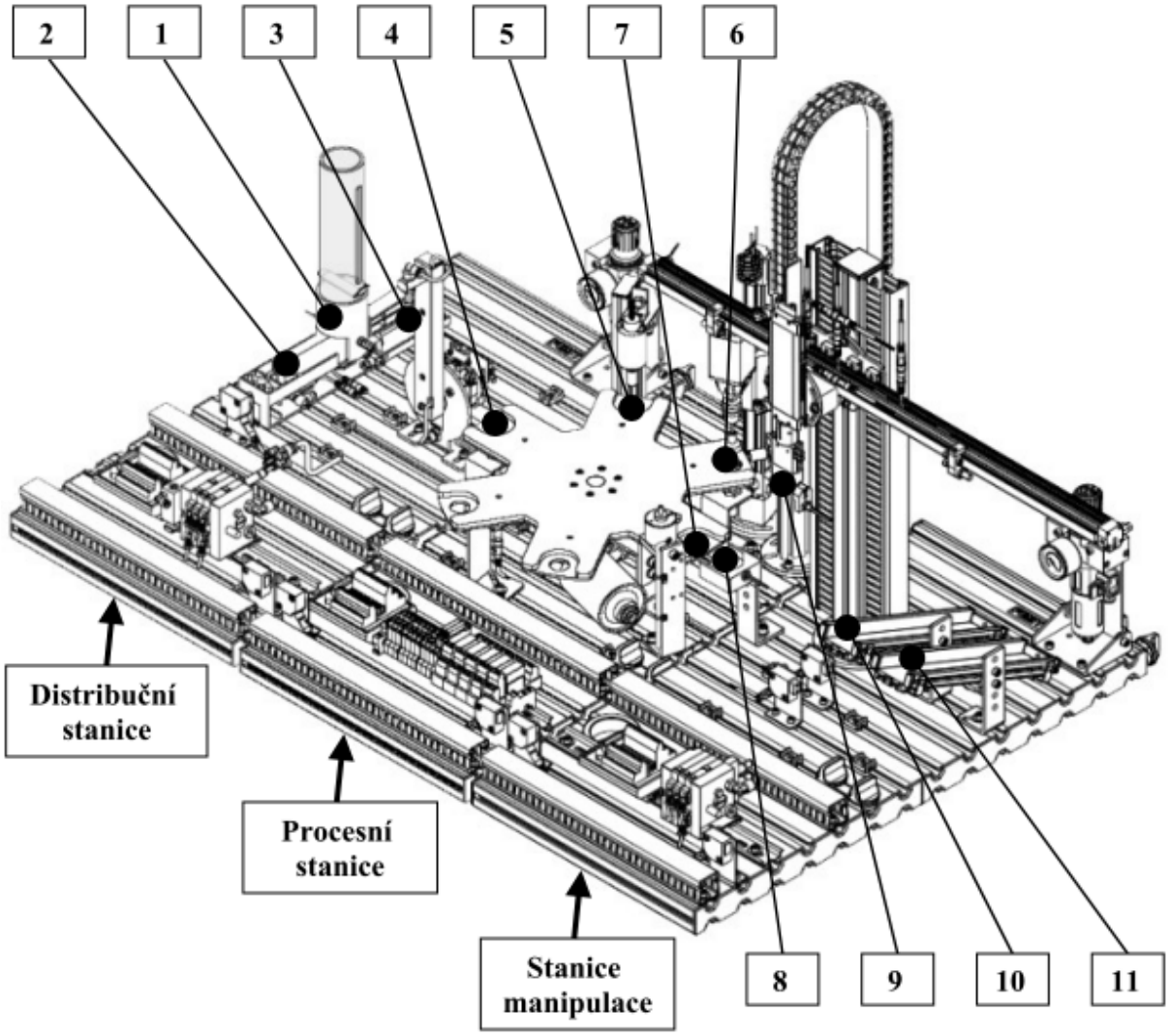
Podrobnosti viz [4].



Obr. č. 4-1 – Výukový model technologického procesu obrábění umístěný v COPSU

Sestava se skládá ze tří základních mechatronických elektropneumatických stanic firmy Festo, *viz Obr. č. 4-2*:

- Distribuční stanice (ST1) - elektropneumatická
- Procesní stanice (ST2) - elektrická
- Manipulační stanice (ST3) - elektropneumatická



Obr. č. 4-2 – Výukový model technologického procesu obrábění umístěný v COPSU

Celou sestavou „protéká“ předmět, jenž si pojmenujme jako **obrobek**. Proces se skládá z klíčových bodů, které jsou očíslovány v pořadí odpovídající postupnému průchodu obrobku sestavou.

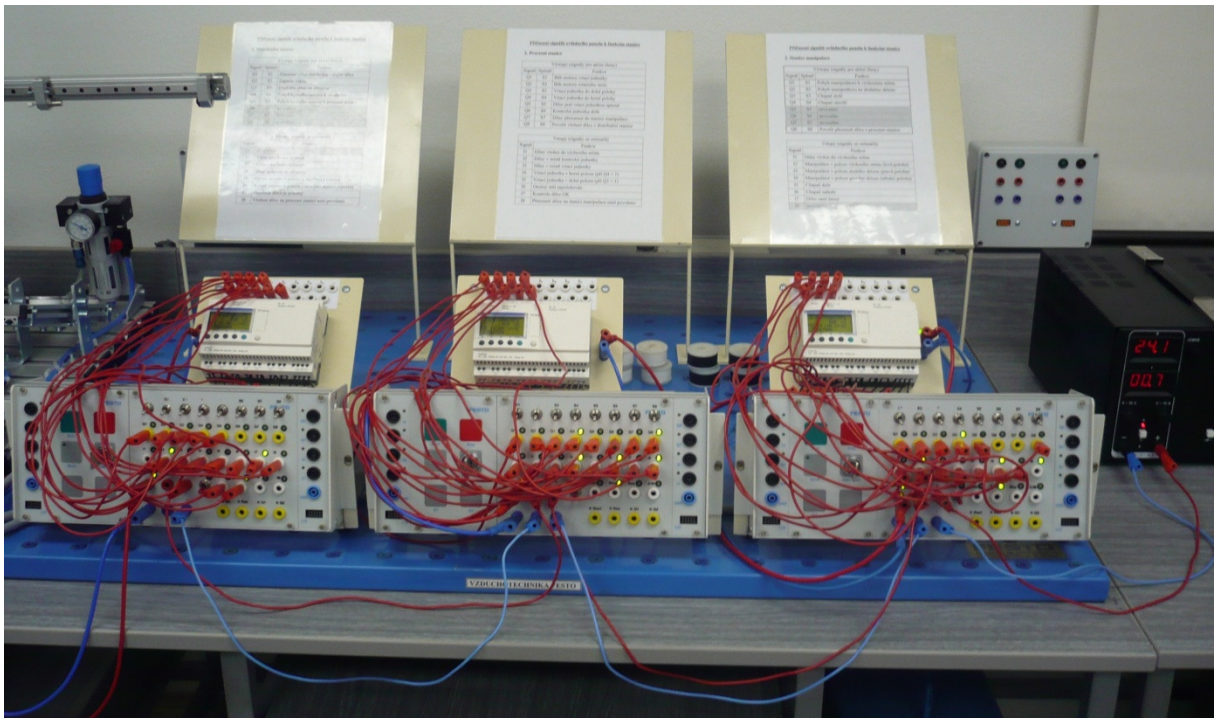
Začátek procesu je v zásobníku distribuční stanice, pozice **1**. Odtud je obrobek vysunut do pozice **2**. Pomocí pneumatického ramene s přísavkou **3** je přenesen na počáteční

polohu procesu rotačního stolu **4**, již procesní stanice. Zde je postupně rotován jednotlivými pracovišti - v pozici **5** je obrobek testován, na pozici **6** je pak obráběn vrtáním a z poslední polohy **7** je přesunut na výchozí místo stanice manipulace **8**. Zde je uchopen dalším pneumatickým ramenem a zároveň je mu rozeznána barva (bílý/černý) **9**. Dle barvy je pak dopraven nad správný zásobník **10**, **11** a následně nad ním uvolněn.

Komunikaci mezi jednotlivými sousedícími stanicemi je realizována opticky. Je přenášen pouze jeden signál a to o připravenosti následující stanice.

Festo k této soustavě dodává i svůj originální řídící systém. Ten však z důvodu vysoké pořizovací ceny nebyl koupen a školou byl realizován vlastní řídící systém.

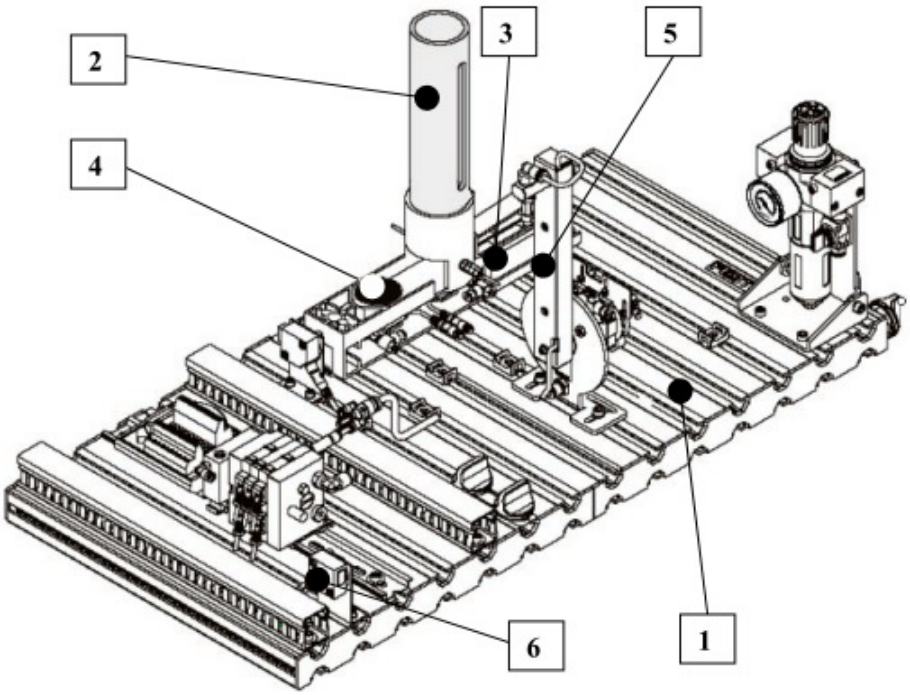
V současné době se tedy k řízení této sestavy používají programovatelná relé firmy Schneider Electric, řady Zelio Logic, *viz Obr. č. 4-3*. Tato relé svým výkonem plně postačují, omezením jsou zde počty vstupů/výstupů. Proto se přistoupilo k myšlence, že každé relé má na starosti řízení právě jedné stanice.



Obr. č. 4-3 – Současný řídící systém včetně ovládacího panelu

4.1 Distribuční stanice (ST1)

Distribuční stanice (ST1) má za úkol vložení obrobku do procesu. *Obr. č. 4-4* přestavuje technický náčrt s označením jednotlivých částí stanice a *Obr. č. 4-5* již představuje foto reálného modelu.



Obr. č. 4-4 – Distribuční stanice (ST1) - 1. část sestavy – popis částí

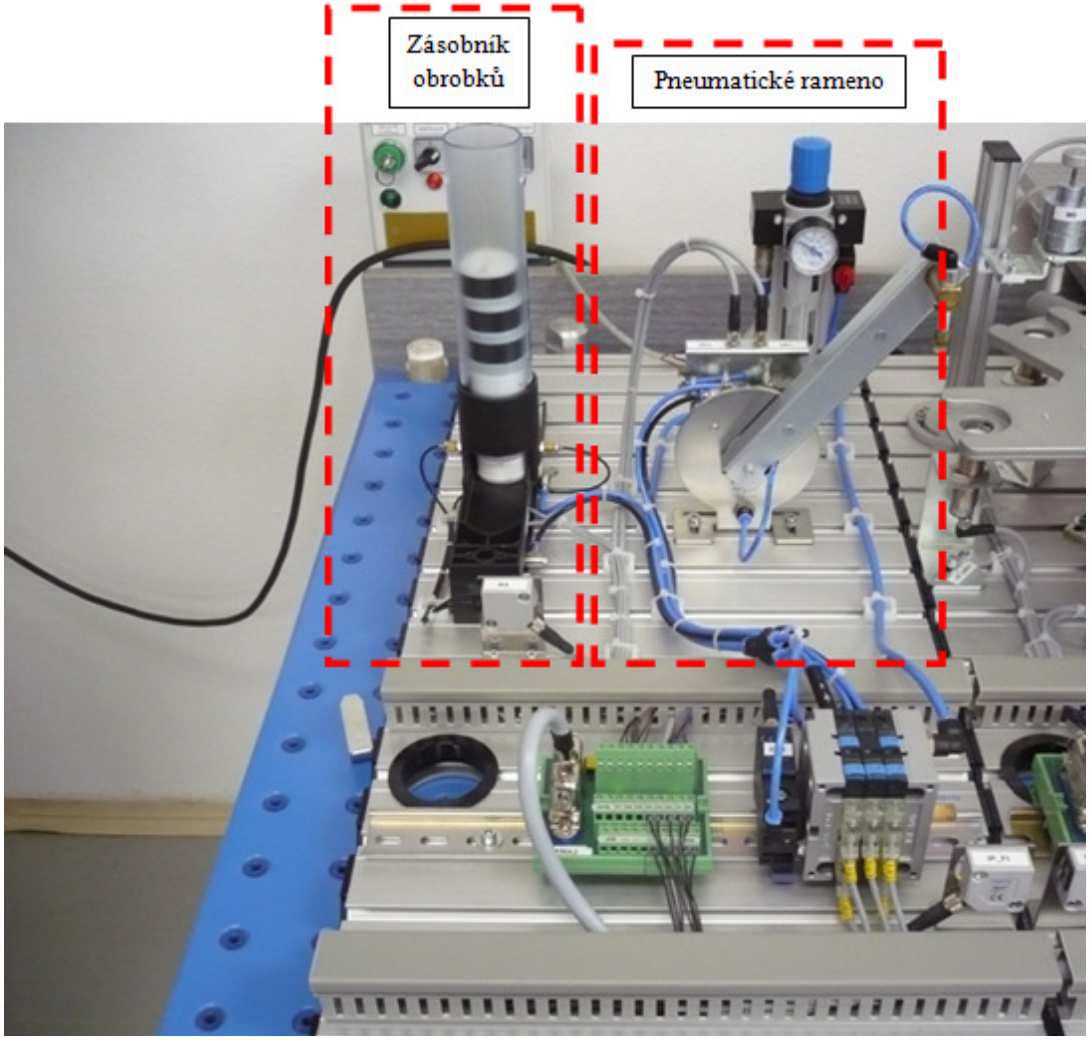
Je tvořena spádovým zásobníkem obrobku **2**, kde jsou jednotlivé obrobky v tubusu umístěny nad sebou. Indikace prázdného zásobníku je pomocí optického senzoru dávající signál **I6**. Pomocí lineárního pneumotoru **3** je obrobek povelem **O0** přesunut na místo předání **4** pro pneumatické rameno. Krajní polohy pneumotoru jsou osazeny magnetickými snímači dávající signál **I1** (píst vysunutý – klidová poloha) a **I2** (píst zasunutý – obrobek přesunut na místo předání **4**).

Je-li tedy obrobek na místě předání **4**, pneumatické kyvné rameno s přísavkou **5** je povelem **O3** ze své výchozí středové polohy (není signalizováno žádným čidlem) přesunuto na levý doraz **I4**, odpovídá zároveň poloze **4**. Podmínkou přichycení obrobku na přísavku na levém dorazu je neobsazené místo na cílové pozici přesunu. To je přenášeno optickým signálem připravenosti **I7** na přijímač světelné závory **6**. K přísavce je hadičkou přiveden podtlak z generátoru vakua (vývěvy), jenž je ovládána signálem **O1**. Korektní přisátí obrobku na přísavku je detekováno snímačem podtlaku **I3** v pneumatickém obvodu. Povelem pro

pohyb vpravo **O4** je obrobek přenesen na rotační stůl, pravý doraz. Po dosažení pravého dorazu **I5** je vypnut podtlak a otevřen ventil vzduchu **O2** s cílem zanechat obrobek na rotačním stole, který již představuje další stanici.

Výchozí stav stanice je:

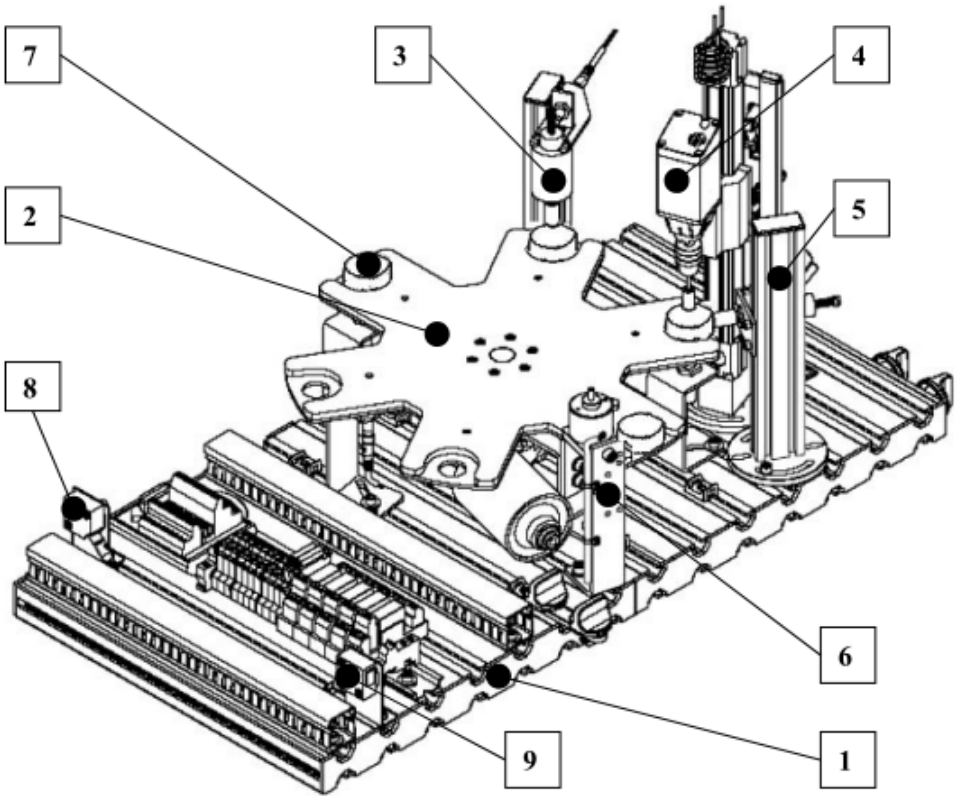
- válec zásobníku je vysunut
- vakuum je vypnuto
- na přísavce není obrobek
- kyvné rameno je ve střední poloze (I4 = 0, I5 = 0)
- v zásobníku je alespoň jeden obrobek
- vložení obrobku na procesní stanici je povoleno



Obr. č. 4-5 – Distribuční stanice (ST1) - 1. část sestavy – reálné foto

4.2 Procesní stanice (ST2)

Procesní stanice (ST2) simuluje technologické zpracování obrobku. *Obr. č. 4-6* přestavuje technický náčrt s označením jednotlivých částí stanice a *Obr. č. 4-7* již představuje foto reálného modelu.



Obr. č. 4-6 – Procesní stanice (ST2) - 2. část soustavy – popis částí

Na základě požadavku na dodání obrobku pomocí signálu **O7** přes optický vysílač **8** je na procesní stanici z předchozí, distribuční stanice, vložen obrobek na výchozí místo **7**. Přítomnost obrobku v této poloze je detekována čidlem **I0**.

Druhá poloha stolu je kontrolní a je detekována čidlem přítomnosti obrobku **I1**. Zde je umístěn modul testování **3.** Povelem **O5** je pomocí elektromagnetu spuštěn dotek dolů a ve své dolní poloze aktivuje snímač **I6,** jeho sepnutí představuje pozitivní kontrolu.

Třetí poloha symbolizuje obrábění pomocí vrtání **4**. Přítomnost obrobku je detekována čidlem **I2**. Vřeteno vrtačky se spouští signálem **O0** a roztáčí brusný váleček uchycený ve

sklíčidle. Pohyb vrtačky je ve svislé ose realizován vodícím šroubem a stejnosměrným motorem. Signál **O2** ovládá pohyb dolu až na dolní čidlo dorazu **I4**, **O3** zas nahoru s horním dorazovým čidlem **I3**

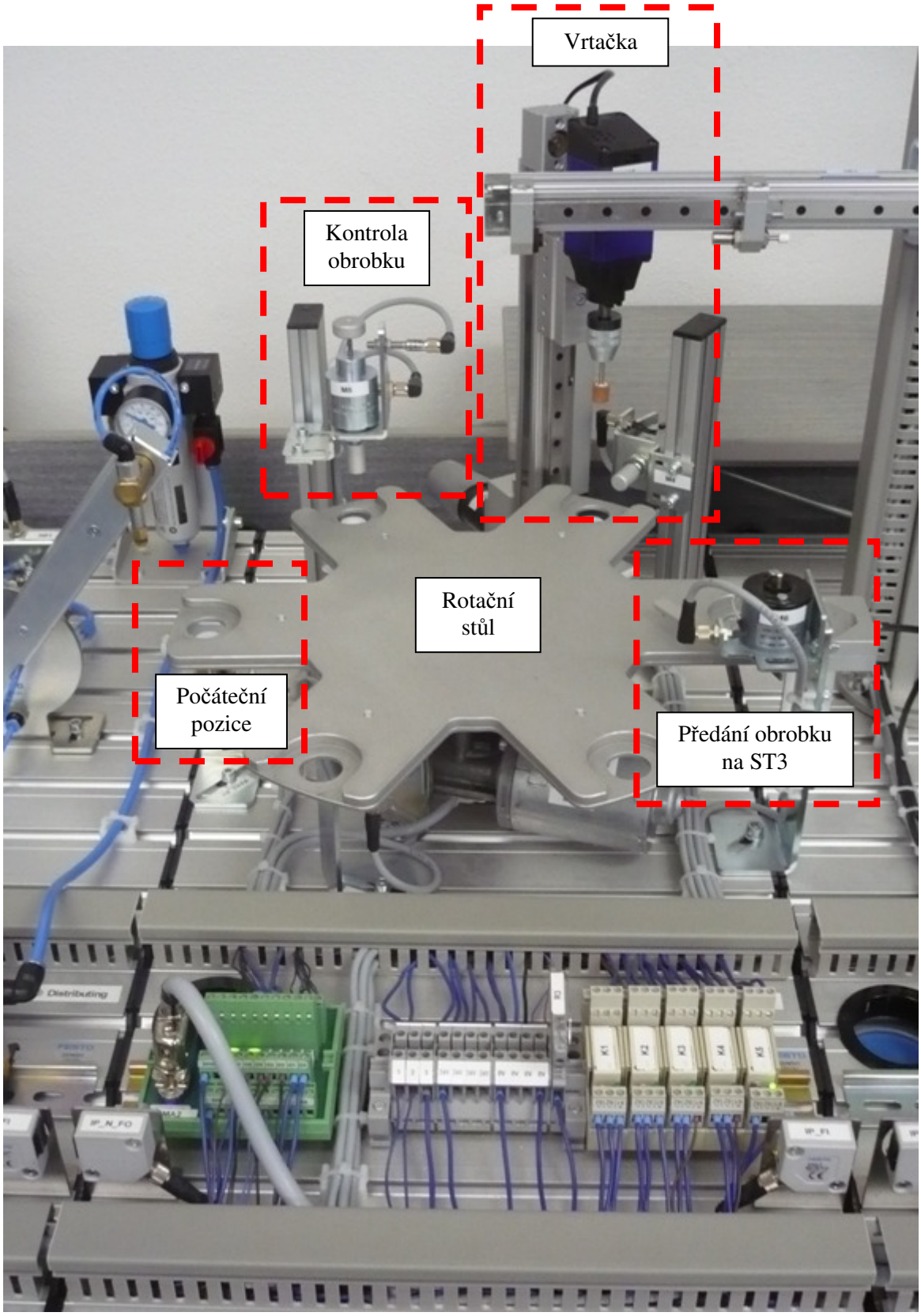
Při obrábění je třeba obrobek pevně uchytit – upnout. Toto realizuje stojan s elektromagnetem **5**, jehož kotvička ovládaná signálem **O4**, vyjíždí vodorovně proti obrobku a fixuje ho ve vybrání otočného stolu.

Nakonec je potřeba obrobek přesunout na další stanici, což má na starosti část **6**. Přesun je realizován pomocí povelu **O6** na elektromagneticky ovládané raménko.

I zde je potřeba znát požadavek na přesun obrobku na další stanici. Opět je toto řešeno optickým signálem **I7** předávaným z následující stanice na přijímač světelné závory **7**.

Výchozí stav stanice je:

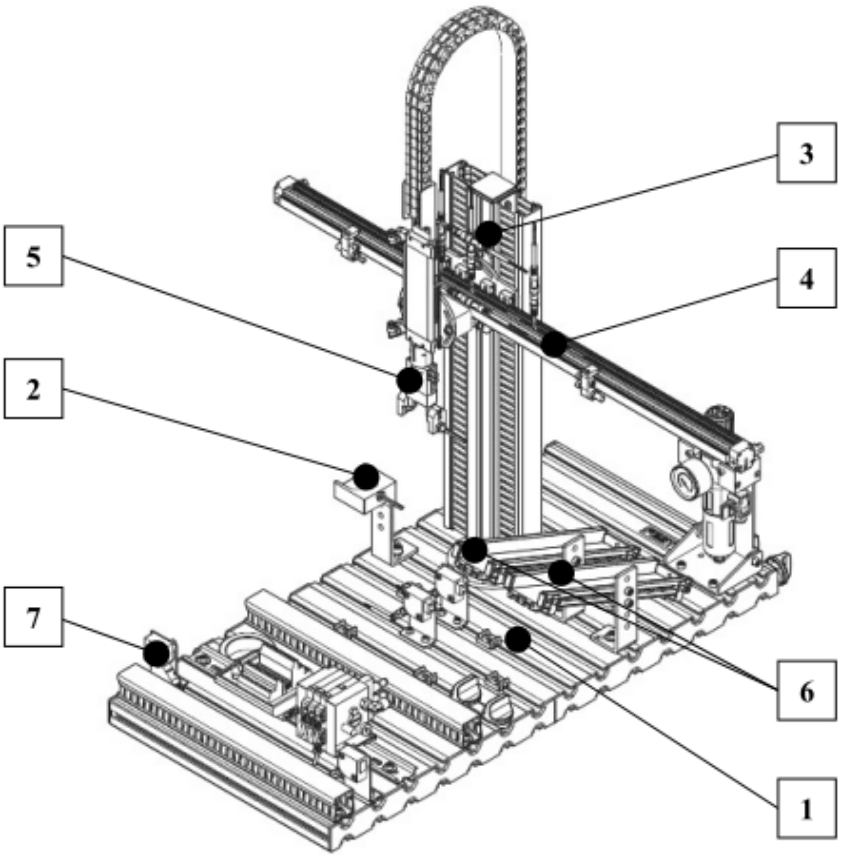
- všechny polohy rotačního stolu jsou prázdné
- vrtačka je nahoře
- vřeteno vrtačky je zastaveno
- rotační stůl je v klidu a zapolohován
- kontrolní dotek je nahoře
- kyvné rameno je ve střední poloze (I4 = 0, I5 = 0)
- v zásobníku obrobků je alespoň jeden obrobek



Obr. č. 4-7 – Procesní stanice (ST2) - 2. část sestavy – reálné foto

4.3 Stanice manipulace (ST3)

Stanice manipulace (ST3) obstarává třídění obrobků do zásobníků dle jejich barvy. *Obr. č. 4-8* přestavuje technický náčrt s označením jednotlivých částí stanice a *Obr. č. 4-9* již představuje foto reálného modelu.



Obr. č. 4-8 - Stanice manipulace (ST3) - 3. část sestavy – popis částí

Na základě požadavku na dodání obrobku pomocí signálu **O7** přes optický vysílač **7** je na stanici manipulace z předchozí, procesní stanice, vložen obrobek na výchozí místo **2**. Přítomnost obrobku v této poloze je detekována čidlem **I0.**

Manipulátor se skládá z hlavního stojanu **3**, na kterém je připevněno vodorovné rameno – bezpístnicový lineární pneumotor **4** nesoucí saně se svisle (pneumaticky) vysouvatelným úchopem **5**.

Vodorovný posuv saní se ovládá signály **O0** (pohyb vlevo, nad výchozí místo) a **O1** (pohyb vpravo, třídění dle barvy). Poloha saní je definována třemi nastavitelnými

magnetickými čidly – **I1** (vlevo, nad výchozí polohou), **I3** (uprostřed, nad levým skluzem) a **I2** (vpravo, nad druhým skluzem).

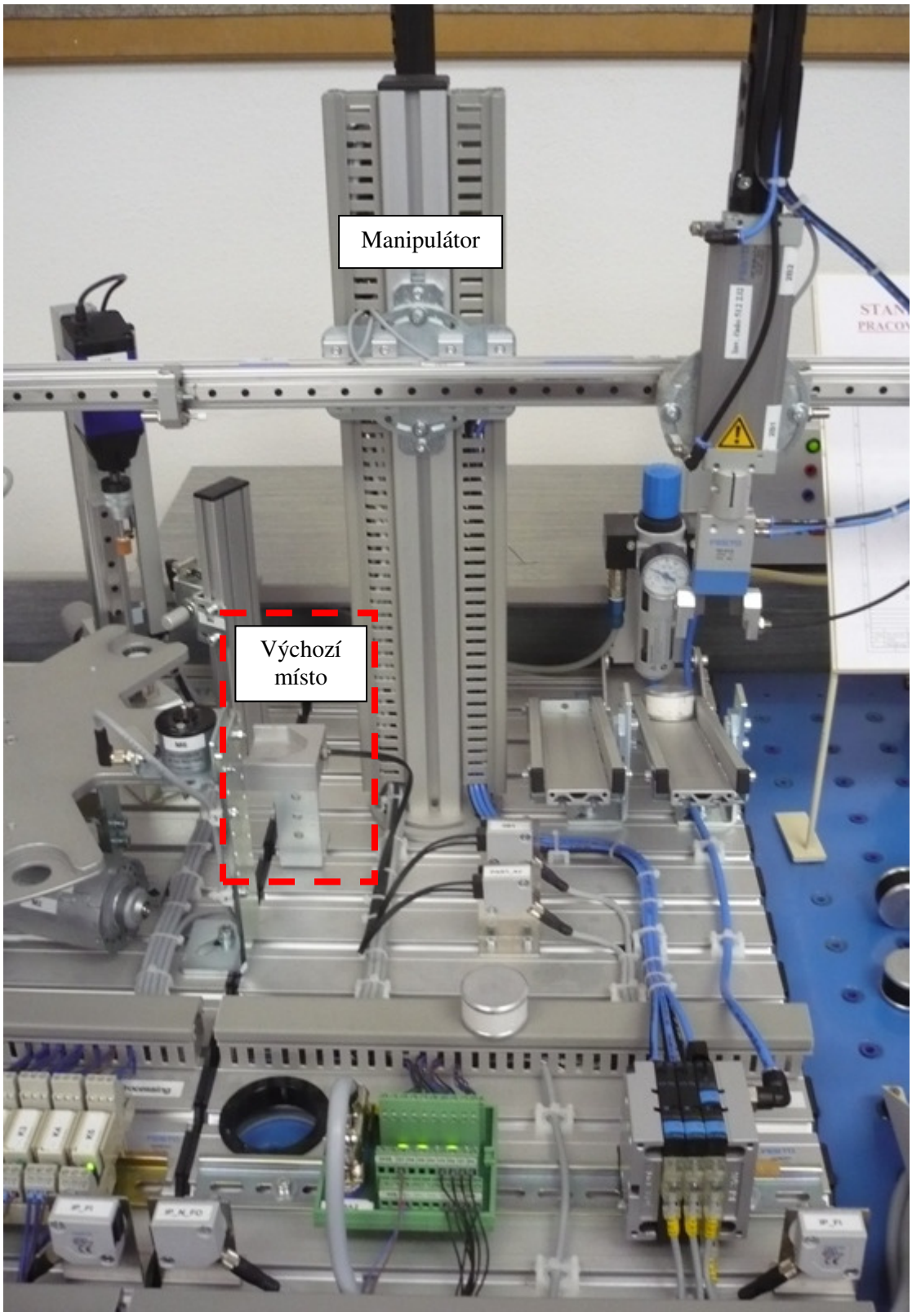
Svislý posuv úchopu je ovládán signálem **O2** a pracuje s koncovými čidly – **I5** (úchop nahoře) a **I4** (úchop dole).

Obrobek je uchopen pomocí signálu **O3**, který řídí zavírání/otevírání úchopu. V čelisti je přítomen i optický **I6** snímač rozpoznávající barvu obrobku.

Dva skluzy **6** pro roztříděné obrobky dle barvy jsou mechanicky stavitelné, neobsahují žádný akční člen ani snímač.

Výchozí stav stanice je:

- přesunutí obrobku z procesní stanice je povoleno
- saně manipulátoru (osa X) jsou v poloze nalevo, nad modulem odběru obrobku
- chapač (osa Z) je nahoře
- čelisti úchopu jsou otevřeny



Obr. č. 4-9 - Stanice manipulace (ST3) - 3. část sestavy – reálné foto

4.4 Souhrn signálů celé sestavy

Na základě předchozího popisu fyzického modelu jsem vytvořil seznam HW signálů (vstupů i výstupů) rozdělený dle stanic, viz *Tab. č. 4-1*:

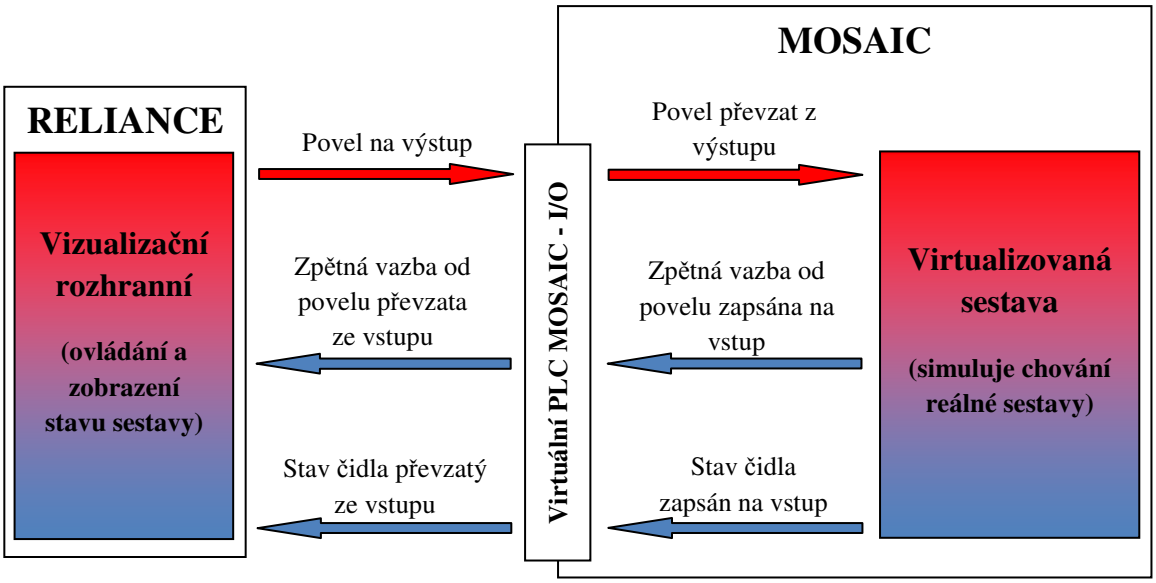
	VSTUPY (signály ze snímačů)			VÝSTUPY (signály pro akční členy)	
	Sig.	**Funkce**		**Sig.**	**Funkce**
Distribuční stanice ST1	I1	Válec zásobníku vysunut		O0	Zasunutí válce zásobníku – vyjetí obrobku
	I2	Válec zásobníku zasunut		O1	Zapnutí vakua
	I3	Obrobek uchycen na přísavce		O2	Uvolnění obrobku na přísavce
	I4	Kyvné rameno v poloze u zásobníku (vlevo)		O3	Pohyb kyvného ramena k zásobníku (vlevo)
	I5	Kyvné rameno v poloze u procesní stanice (vpravo)		O4	Pohyb kyvného ramena k procesní stanici (vpravo)
	I6	Zásobník obrobků je prázdný			
	I7	Vložení obrobku na procesní stanici není povoleno			
Procesní stanice ST2	I0	Obrobek v počáteční pozici		O0	Běh vřetena vrtačky
	I1	Obrobek v poloze kontrolní stanice		O1	Točení rotačního stolu
	I2	Obrobek v poloze vrtání		O2	Vrtačka dolů
	I3	Vrtačka nahoře		O3	Vrtačka nahoru
	I4	Vrtačka dole		O4	Upnutí obrobku
	I5	Rotační stůl zapolohován		O5	Kontrolní dotek dolů
	I6	Kontrola obrobku OK		O6	Obrobek přesunout na stanici manipulace
	I7	Vložení obrobku na stanici manipulace povoleno		O7	Procesní stanice obsazena
Manipulační stanice ST3	I0	Obrobek vložen do výchozího místa		O0	Pohyb manipulátoru k výchozímu místu (vlevo)
	I1	Manipulátor v poloze výchozího místa (levá poloha)		O1	Pohyb manipulátoru ke druhému skluzu (vpravo)
	I2	Manipulátor v poloze druhého skluzu (pravá poloha)		O2	Úchop dolů (osa Z)
	I3	Manipulátor v poloze prvního skluzu (střední poloha)		O3	Čelisti otevřít
	I4	Úchop dole		O7	Výchozí místo stanice manipulace obsazeno
	I5	Úchop nahoře			
	I6	Obrobek není černý			
	počet binárních vstupů: 22			počet binárních výstupů: 18	

Tab. č. 4-1 – Seznam HW I/O signálů dělený dle stanic

Se všemi těmito signály budu pracovat i ve virtuální variantě, jelikož je nutné, aby chování virtuální verze včetně shodnosti signálů odpovídalo realitě. Takto bude zachována přenositelnost algoritmu mezi virtuálním a fyzickým PLC.

5 Virtualizace zvolené sestavy

Na základě předchozího popisu fungování sestavy a cíle diplomové práce jsem navrhl následující blokové schéma virtualizované sestavy, *viz Obr. č. 5-1*.



Obr. č. 5-1 – Blokové schéma virtualizované soustavy pomocí SW Mosaic a Reliance

V této fázi řešení mi bude Mosaic představovat virtualizovaný model sestavy. Tj. na základě pozorování a dostupné dokumentace přenesu chování sestavy a jejich logických vazeb a souvislostí (tj. dám-li povel na dynamický prvek, očekávám příslušnou reakci a hlášení o stavu jiod čidel) do algoritmů, které následně poběží na simulátoru PLC.

Jelikož nemám k dispozici fyzické PLC, budu v SW Mosaic používat modul *Simulátor PLC* . Vstupy a výstupy tohoto simulátoru budu jako globální proměnné zároveň používat pro přenos proměnných mezi Mosaic a Realiance. Simulátor PLC plně nahradí reálné PLC a jsem tak schopen kompletně rozběhnout budoucí sestavu jen za pomocí SW nástrojů a vše spustit na jednom PC.

Abych mohl k takto vytvořené sestavě pohodlně a komfortně přistupovat, vytvořím v Reliance grafické rozhraní. Základní obrazovka představuje blokově znázorněný reálný model. Z ní budu schopen pracovat se všemi signály, jenž využívá i fyzická sestava. Pro potřeby dokonalejší představy o stavu sestavy jsem vytvořil i řadu pomocných signálů informující např. o aktuální poloze mechaniky, obrobku apod.

Dle potřeby vytvořím i další obrazovky pro rozšíření možností celé virtuální sestavy.

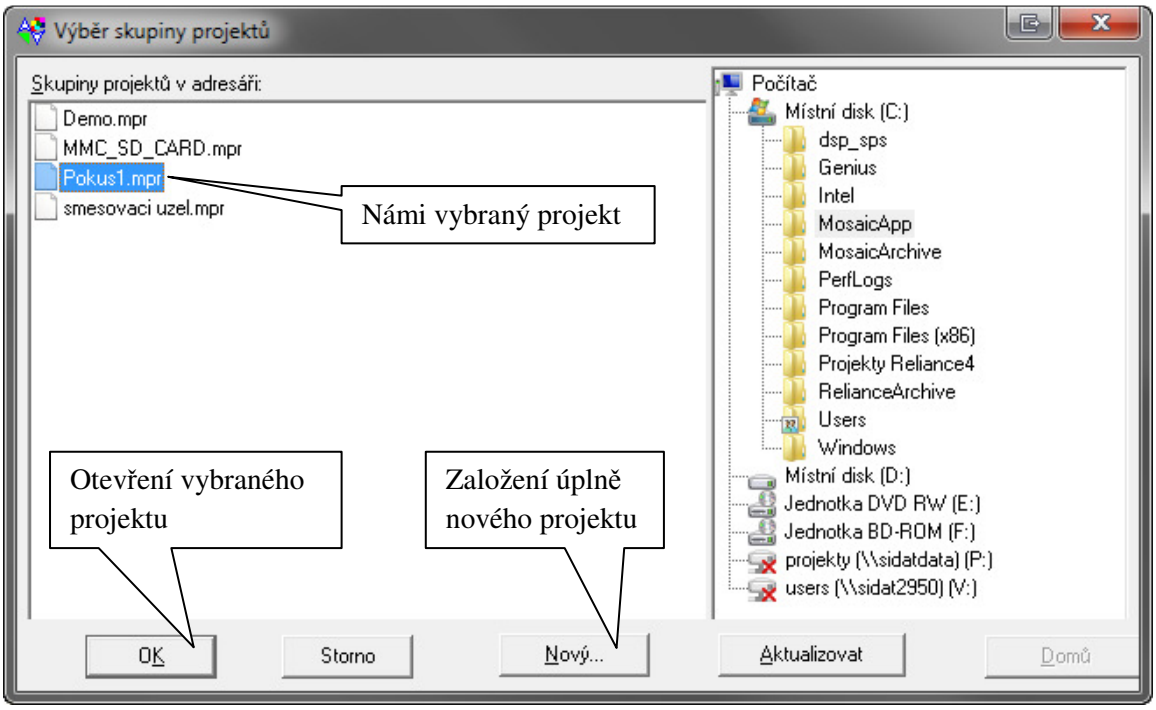
5.1 Realizace soustavy v Mosaic

5.1.1 Spuštění prostředí

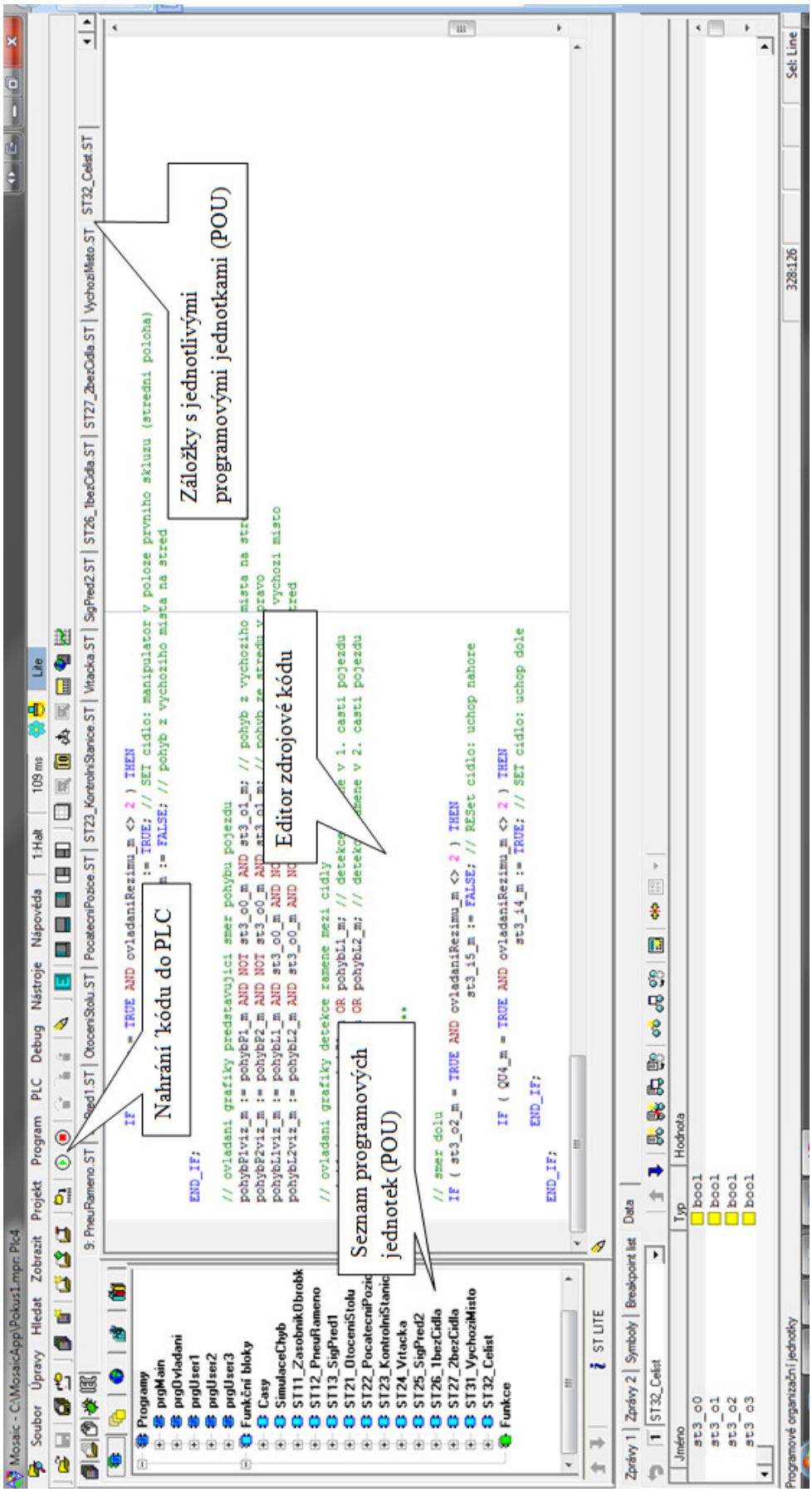
Po instalaci SW Mosaic verze 2.20.0 jej spustíme ikonou . Hned v úvodu nás program informuje o verzi a o nepřítomnosti HW klíče, *viz Obr. č. 5-2*.

Obr. č. 5-2 – Spouštění Mosaic – informace o licenci a verzi SW

Odsouhlasením získáme další okno, kde je možnost otevřít již stávající skupinu projektů či vytvořit úplně novou, *viz Obr. č. 5-3*. Naše skupina projektů se neprakticky jmenuje *Pokus1.mpr*, projekt pak *Plc4*. Tím již otevřeme vývojové prostředí, *viz Obr. č. 5-4*.



Obr. č. 5-3 – Spouštění Mosaic – otevření existující projektu nebo založení úplně nového

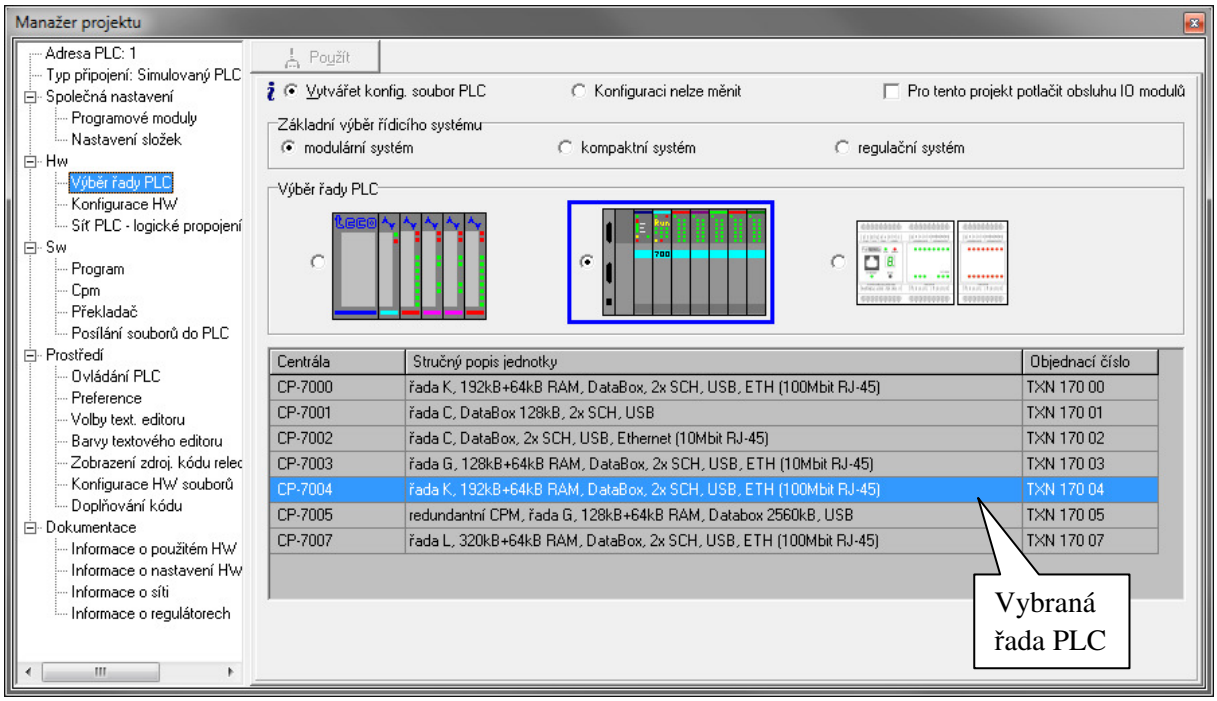


Obr. č. 5-4 – Vývojové prostředí Mosaic – popis jednotlivých částí

5.1.2 HW konfigurace

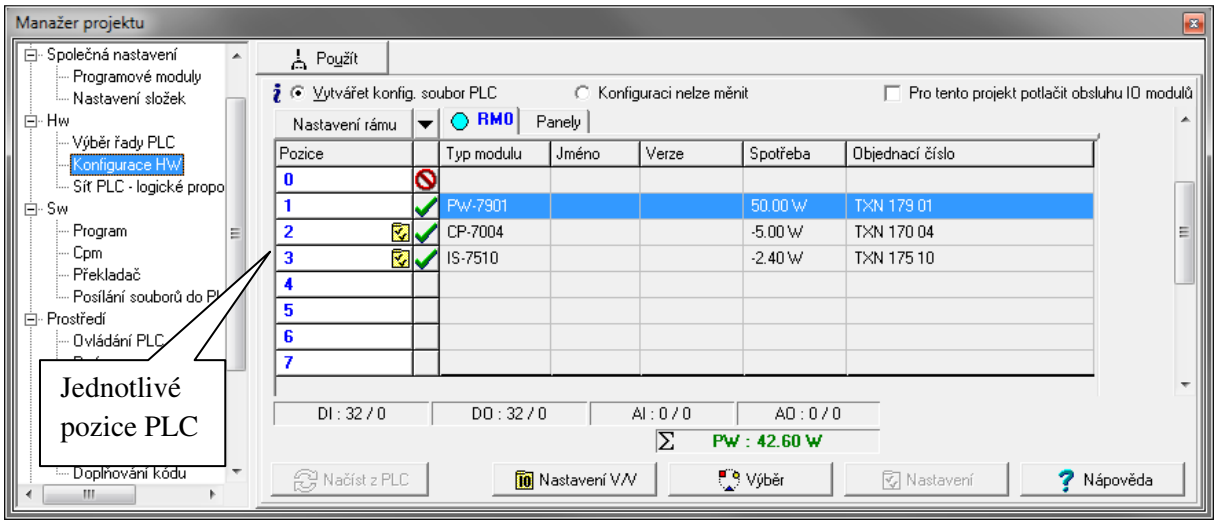
Základem každého SW pro programování PLC je vytvoření tzv. *HW konfigurace*. HW konfigurace odpovídá hardwarovému popisu PLC, tj. jaký používá napájecí zdroj, CPU jednotku a rozšiřující karty. V případě, že je PLC zapojeno v komunikační sítí, pak je zde i popis této sítě. Tato nastavení se pak nahrají do paměti PLC, které tak ví, s jakými periferiemi bude pracovat a jaký je stav jeho okolí.

V Mosaic se k této konfiguraci dostaneme přímo z hlavního okna přes ikonu *Manažer projektu* . Prvním krokem je vybrání řady PLC. Volím řadu 700, konkrétně centrálu CP-7004. Disponuje pamětí 192kB+64kB RAM, rozhraním USB a Ethernet, *viz Obr. č. 5-5.*



Obr. č. 5-5 – Manažer projektu – Výběr řady PLC

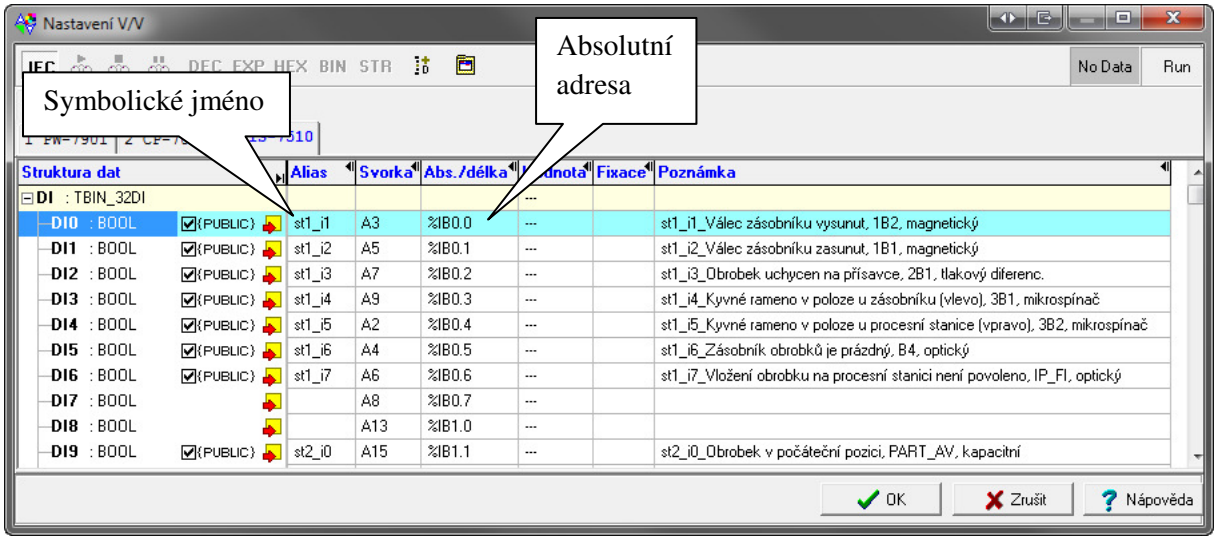
Nyní je potřeba k této centrále přidat periferie, *viz Obr. č. 5-6.* Jako napájecí zdroj jsem vybral PW-7901, což je zdroj 24V AC/DV bez UPS. Původní myšlenka bylo použít více vstupně/výstupních karet, např. tři I/O karty, čímž bychom docílili rozložení signálů jako je současná podoba řízení. Bohužel licenční politika programu, kdy využívám LITE verzi, toto nedovolila. V této verzi mohu použít pouze jednu I/O kartu. Zvolil jsem tedy kartu IS-7510. Tato karta obsahuje 32 vstupů 24V DC a 32 výstupů 24V/50mA DC, což pro naše potřeby plně postačuje.



Obr. č. 5-6 – Manažer projektu – Konfigurace HW

Abychom mohli při programování používat symbolická jména (např. st1_i1) a ne nepraktické absolutní adresy (IB0.0), je nutné tuto vlastnost nastavit.

Děje se tak přes ikonu Nastavení V/V, nejdříve je však nutné mít v okně *Konfigurace HW* vybránu potřebnou rozšiřující kartu, pro nás tedy *IS-7510*. Tím se dostáváme do okna *Nastavení V/V*, *viz Obr. č. 5-7,* kde jednotlivým vstupním/výstupním svorkám přiřazujeme symbolická jména, např. původní absolutní adrese IB0.6 přiřadíme symbolický název st1_7, což při tvorbě programu tvůrci řekne mnohem více. Kód je pak snáze pochopitelný i pro své okolí, nejen pro tvůrce a rovněž pozdější úprava a editace je tak snazší a efektivnější.

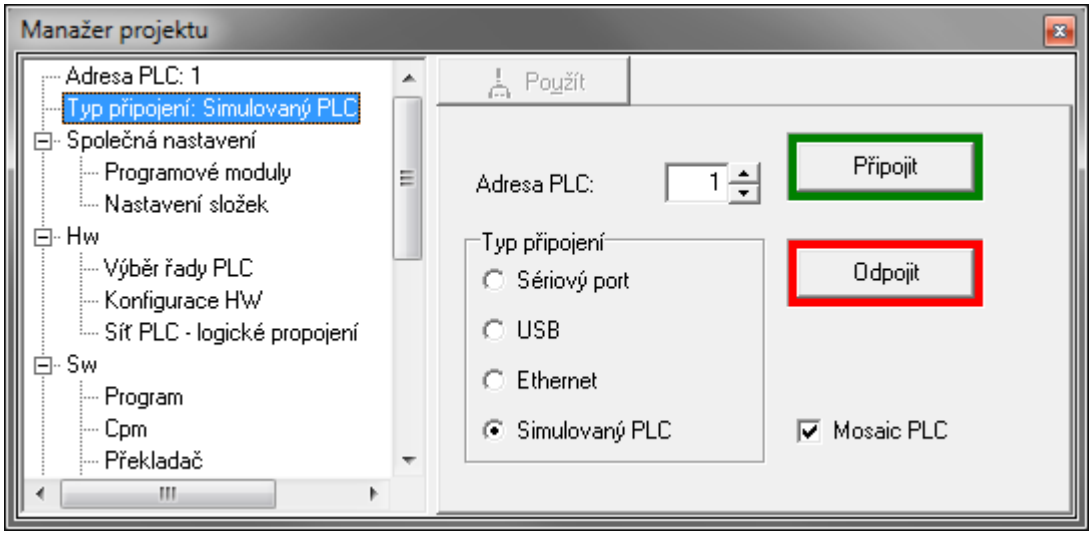


Obr. č. 5-7 – Manažer projektu – Konfigurace HW

5.1.3 Simulované PLC

Jak již bylo řečeno v úvodu, pracuji pouze s virtuálními nástroji, tj. v současné době nemám k dispozici ani reálné PLC.

Tento nedostatek vyřeším pomocí modulu *Simulátor PLC*, jenž *Mosaic* obsahuje, *viz Obr. č. 5-8*. Pomocí tohoto jsem schopný plně nahradit reálné PLC a vše realizovat na jednom výkonném počítači. K nastavení modulu se dostanu opět přes *Manažera projektu*.

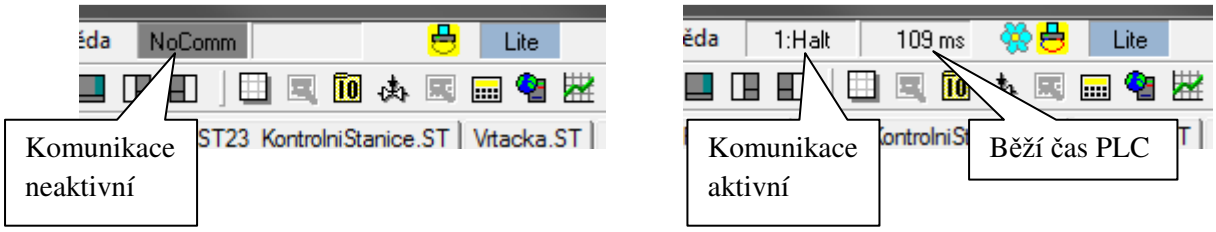


Obr. č. 5-8 – Manažer projektu – Simulátor PLC

Zde vyberu položku *Typ připojení*, která mi nabízí řadu možností. Pro nás důležitá je varianta *Simulovaný PLC*. Vyplníme adresu PLC, zaškrtneme volbu „Mosaic PLC“ a přes tlačítko 

aktivujeme toto připojení.

Že aktivace připojení simulovaného PLC proběhla správně, si můžeme ověřit v hlavním okně prostředí. Zde se po korektním připojení rozběhne čítač času, *viz Obr. č. 5-9.*



Obr. č. 5-9 – Kontrola připojení simulovaného PLC – vlevo: nepřipojen, vpravo: připojen

5.1.4 Rozdělení virtuální sestavy

Jako programovací jazyk pro celý projekt jsem si zvolil Strukturovaný text (ST). Má nejblíže ke klasickému programování (Java, C, Pascal…), na které jsem zvyklý ze školy.

Sestavu jsem si nejprve rozdělil na jednotlivé stanice a ty dále na základní celky, které budu softwarově realizovat, *viz Tab. č. 5-1*.

Název stanice	Název celku	Název funkčního bloku
Distribuční (ST1)		
	Zásobník obrobků	ST11_ZasobnikObrobku
	Pneumatické kyvné rameno	ST12_PneuRameno
	Signál připravenosti 1	ST13_SigPred1
Procesní (ST2)		
	Otočení stolu	ST21_OtoceniStolu
	Výchozí místo	ST22_PocatecniPozice
	Kontrolní stanice	ST23_KontrolniStanice
	Vrtačka	ST24_Vrtacka
	Signál připravenosti 2	ST25_SigPred2
	1. místo na stole bez čidla	ST26_1bezCidla
	2. místo na stole bez čidla	ST27_2bezCidla
Manipulační (ST3)		
	Výchozí místo	ST31_VychoziMisto
	Manipulátor	ST32_Celist

Tab. č. 5-1 – Seznam jednotlivých funkčních bloků

Jak plyne z tabulky, sestava se skládá z celkem 12ti základních celků. Tyto celky je nutno nyní detailně navrhnout a realizovat.

5.1.5 Hlavní program a globální proměnné

Každý základní celek je realizován jedním funkčním blokem. Jeho lokální proměnné slouží pro interní potřeby bloku (práce s čítači času, pomocná koncová čidla apod.). Globální proměnné slouží pro vstup (povely na akční členy) a výstup (stavy čidel) hodnot z bloku do celého projektu.

Všechny funkční bloky jsou volány z hlavního programu *prgMain*, jenž je neustále cyklicky volán.

U hlavního programu jsou nadeklarovány rovněž všechny globální proměnné. Ty, které je potřeba předávat dále do vizualizace, mají přidán parametr {PUBLIC}. Po přeložení programu v Mosaicu je vygenerován soubor *.PUB, který obsahuje právě všechny tyto proměnné a pomocí něj je pak možné je importovat do vizualizace.

Program *prgMain* a definice globálních proměnných tvoří *prgMain.ST*, názorná struktura *viz Obr. č. 5-10.*

prgMain.ST

```
VAR_GLOBAL  // deklarace globalnich promennych

    Deklarace všech globálních proměnných (pohyb modelu a obrobku, tlačítka z vizualizace

END_VAR  // konec deklarace

PROGRAM prgMain  // zacatek programu

VAR  // deklarace lokalnich proměnných

    Deklarace instancí všech použitých funkčních bloků (simulace modelu, nastavení času,
    maskování sig.)

END_VAR  // konec deklarace

    // FB simulujici model
    ST11_ZasobnikObrobku1 (…);
    ST12_PneuRameno1 (…);
    ST13_SigPred1_1 (…);
    ST21_OtoceniStolu1 (…);
    ST22_PocatecniPozice1 (…);
    ST23_KontrolniStanice1 (…);
    ST24_Vrtacka1 (…);
    ST25_SigPred2_1 (…);
    ST26_1bezCidla1 (…);
    ST27_2bezCidla1 (…);
    ST31_VychoziMisto1 (…);
    ST32_Celist1 (…);

    // FB rozsirujici
    Casy1 (…);  // obsluha casu predstavujici dynamicka zpozdeni modelu
    SimulaceChyb1 (…);  // obsluha maskovani signalu

END_PROGRAM  // konec programu
```

Obr. č. 5-10 – Blokové struktura prgMain.ST - hlavní program prgMain a globální proměnné

5.1.6 Princip tvorby simulačních FB

Na první části sestavy, tj. *Zásobník obrobků* (funkční blok *ST11_ZasobnikObrobku*) vysvětlím algoritmus řešení a následně i blokově popíši strukturu tohoto funkčního bloku.

Při pozorování funkce zásobníku obrobků lze rozeznat jednotlivé stavy, které mají pro své vykonávání nutné podmínky a tento stav tak jednoznačně definují. Této vlastnosti jsem využil při tvorbě všech simulačních FB.

Zásobník obrobků lze dále rozdělit na stavy:

- **Počáteční stav** – definuje počáteční stav, např. je-li některé čidlo ve výchozím stavu aktivní
- **Založení obrobku do zásobníku** – obsluhuje založení (přítomnost) obrobku do (v) zásobníku
- **Odebrání obrobku ze zásobníku** - obsluhuje odebrání (nepřítomnost) obrobku ze (v) zásobníku
- **Zasunutí válce zásobníku (vyjetí přítomného obrobku)** – posun obrobku do procesu
- **Zasunutí válce zásobníku (není přítomen obrobek) –** válec se pohybuje naprázdno, bez obrobku
- **Vysunutí válce zásobníku (pohyb do základní polohy)** – válec se vrací do výchozí polohy

Každý tento stav jsem realizoval pomocí příkazu **IF**, *viz Obr. č. 5-11,* jehož syntaxe je

```
IF podmínka THEN
                    kód - vykoná se po splnění podmínky
END_IF;
```

```
//POCATECNI STAV

IF ( st1_o0_m = FALSE AND st1_i2_m = FALSE AND obrobek_m = FALSE AND
obrobek1_m = FALSE AND obrobek2_m = FALSE AND QD1_m = TRUE AND
ovladaniRezimu_m <> 2 AND stuj = FALSE ) THEN

    st1_i1_m := TRUE; // pocatecni stav cidla - po zapnuti stroje je cidlo st1_i1_m SETovano
    obrobek1_m := FALSE; // ovladani viditelnosti obrobku ve vizualizaci
    obrobek2_m := FALSE; // ovladani viditelnosti obrobku ve vizualizaci
    CD1_m := FALSE; // ovlada sestupny vstup citace CTUD1
    CU1_m := FALSE; // ovlada vzestupny vstup citace CTUD1
    st1_i6_m := TRUE; // SET cidlo: zasobnik obrobku je prazdny
    resetCTUD := FALSE; //reset citace CTUD2

END_IF;
```

Obr. č. 5-11 – Příklad kódu FB ST11_ZasobnikObrobku – počáteční stav

Cílem tedy bylo popsat jednotlivé stavy pomocí podmínek tak, aby byl vždy vykonáván jen jeden z nich a to buď v logickém pořadí za sebou nebo po nasimulování podmínek i nezávisle na předchozím vykonaném stavu. Na základě toho je např. možné vložit obrobek přímo na místo předání na pneumatické kyvné rameno, aniž by bylo nutné předtím zasunout válec zásobníku. Tím se při seznamování se sestavou šetří čas, jelikož není nutné si projít celý předchozí proces, než se dostaneme do stavu, který nás zajímá.

Aby simulace co nejvíce odpovídala realitě, bylo nutné rovněž do algoritmu zapracovat zpoždění simulující dynamiku jednotlivých pohybů. Původní myšlenkou bylo zpoždění realizovat pomocí časovače (fce. Timer), to se však z hlediska ovládání ukázalo jako těžko realizovatelné (časovač nelze zastavit či jeho čas snižovat). Místo toho jsem použil čítač (fce. Counter). Na jeho vstupy tiká proměnná odvozená od systémových registrů s periodou 100ms, tím jsem docílil jakési funkce čítače času. Čítač lze zastavit a i dekrementovat, což plně vyhovuje mým potřebám.

Pro potřeby nastavování těchto času jsem vytvořil samostatný funkční blok *Casy*, kde jsou pevně uloženy defaultní odzkoušené hodnoty včetně možnosti editace uživatelem.

Bloková struktura FB *ST11_ZasobnikObrobku.ST, viz Obr. č. 5-12*.

ST11_ZasobnikObrobku.ST

```
FUNCTION_BLOCK ST11_ZasobnikObrobku  // zacatek programu

VAR_INPUT  // deklarace vstupnich proměnných

    Deklarace vstupních systémových registrů tikající s periodou 100ms pro simulaci
    časovače pomocí čítače

END_VAR  // konec deklarace

VAR  // deklarace lokalnich promennych

    Deklarace lokálních proměnných, jsou použitelné pouze pro toto FB

END_VAR  // konec deklarace

VAR_IN_OUT  // deklarace vstupnich/vystupnich proměnných

    Deklarace vstupních/výstupních proměnných pro předávání parametrů do/z FB

END_VAR  // konec deklarace

    Obsluha čítačů pro simulaci dynamicky

    Obsluha jednotlivých stavu Zásobníku obrobků

    Obsluha tlačítka RESET – proměnné vráceny do původního stavu

END_PROGRAM  // konec programu
```

Obr. č. 5-12 – Blokové struktura ST11_ZasobnikObrobku.ST

Přímo zdrojový kód je k dispozici v příloze 9.2.

Zdrojové kódy zbylých funkčních bloků jsou z důvodu jejich značné délky přiloženy pouze v elektronické podobě na přiloženém DVD, v adresáři *Výsledky diplomové práce\ Kódy ze SW Mosaic v souborech TXT*.

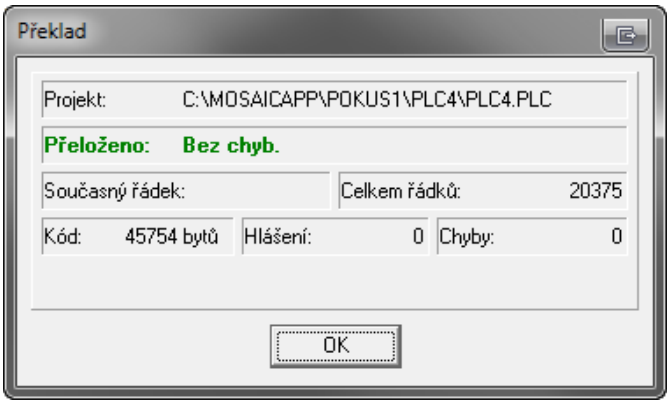
5.1.7 Režim Manuálně

Jak vyplyne z dalšího popisu, současný stav projektu (algoritmus v Mosaicu a vizualizace v Reliance) lze nazvat režimem *Manuálně*. Zatím není k dispozici žádný panel pro ovládání a volbu režimů, tudíž je tento stav jako výchozí. K jeho spuštění není nutno splnit žádné další podmínky.

V rámci tohoto režimu je možné si sestavu kompletně vyzkoušet, tj. není nutno přesně dodržovat tok obrobku sestavou, ale lze si nezávisle na předchozím stavu za splnění podmínek vyzkoušet libovolný stav. Toto má nespornou výhodu v tom, že uživatel (student) nebude nucen tento zdlouhavý proces seznamování se s funkčností modelu realizovat na reálně sestavě, kde v tomto případě hrozí i větší nebezpečí poškození, ale vše si právě otestuje na virtuální kopii. Jednoduše si dle technických podkladů nastaví sestavu do požadovaného stavu (povely na akční členy a zpětná hlášení od čidel) a může si analyzovat její chování. Takto si bude uživatel moci projít jednotlivé stavy, včetně těch kolizních a dovést tak své budoucí řídící algoritmy k dokonalosti.

5.1.8 Spuštění projektu v PLC

Máme-li již algoritmus zapsán v Mosaic, je nutné jej nechat zkompilovat. To se činí přes klávesnici F9. Je nutné, aby kompilace proběhla bez chyby, *viz Obr. č. 5-13*



Obr. č. 5-13 – Okno se stavem kompilace – kompilace proběhla bez chyby

Po úspěšné kompilaci a aktivním spojení se simulovaným PLC nahrajeme kód do tohoto PLC. K tomu slouží . Po jeho stisku jsme dotázáni na vyslání programu do PLC, potvrzením se kód nahraje PLC. Poté ještě určíme typ restartu PLC. Nyní je projekt již plně spuštěn a vykonává naprogramovaný algoritmus.

5.2 Realizace soustavy v Reliance

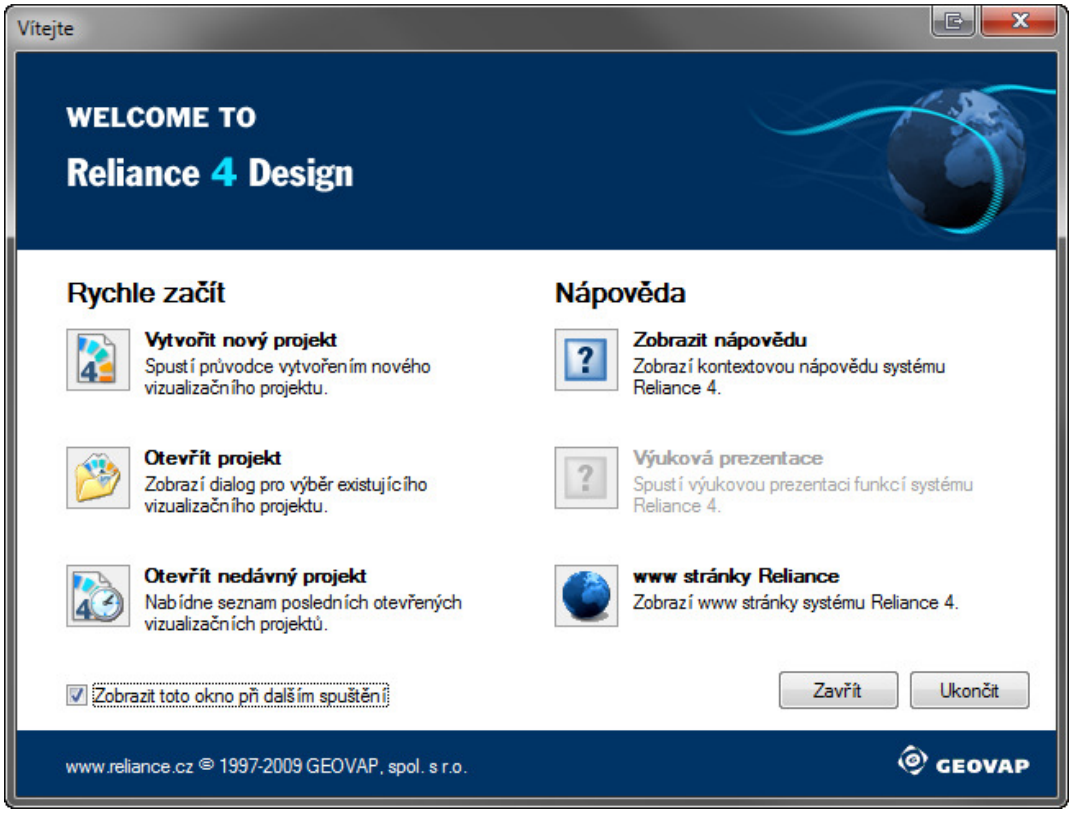
V tomto nástroji jsem realizoval grafické uživatelské rozhraní. Základ vytvoří dvě obrazovky

- Obrazovka modelu sestavy
- Obrazovka pro nastavení časů simulující dynamiku sestavy

5.2.1 Spuštění prostředí

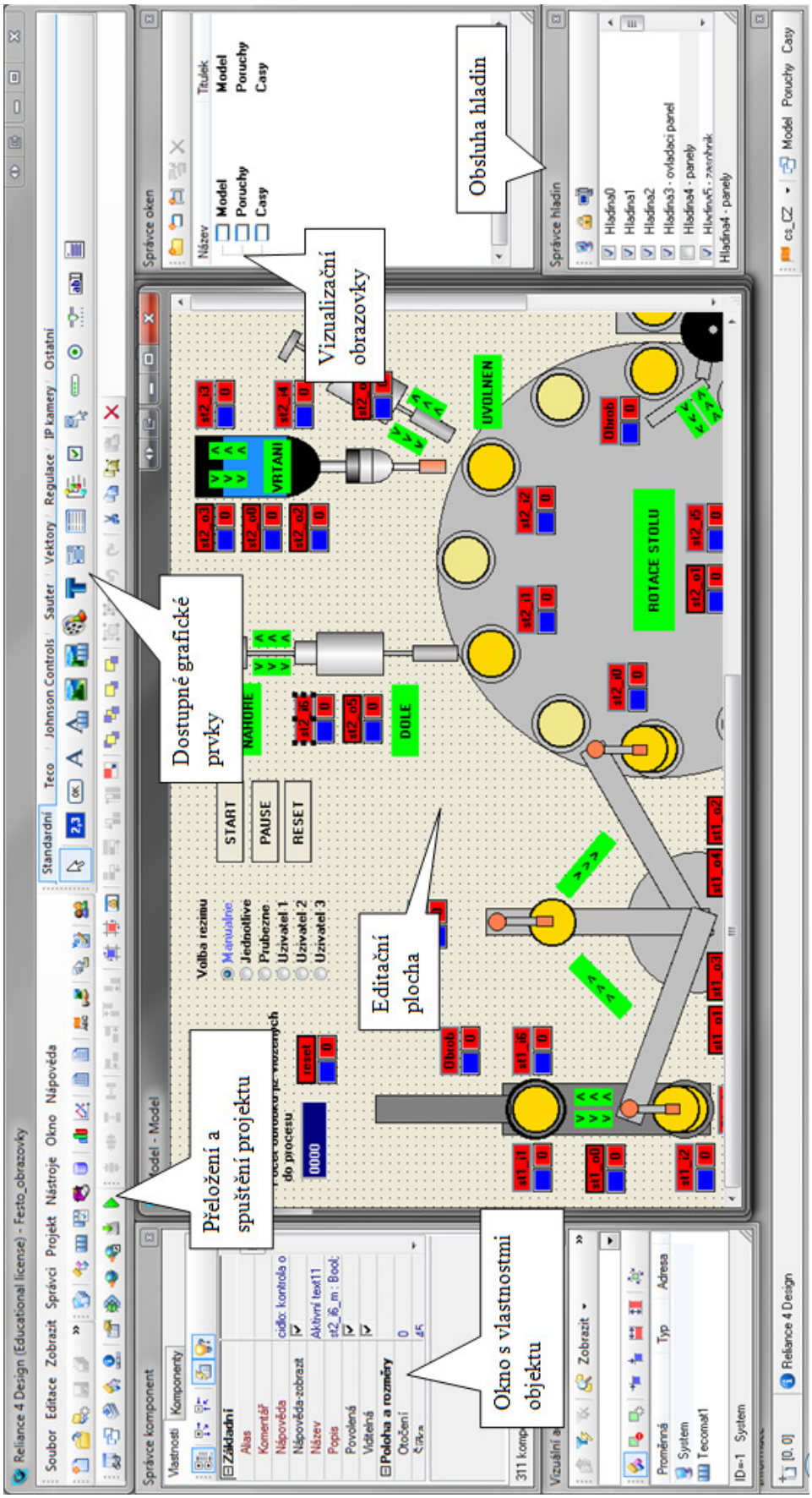
Vývojové prostředí Reliance 4 Design spustíme přes ikonu . Hned po startu programu je kontrolována licence. Bez ní je možné použít max. 25 proměnných, což je skutečně málo. K dispozici mám zapůjčenou školní licenci ve formě USB Flash disku. Tato školní licence umožňuje pracovat až s 500ti proměnnými, což již plně postačuje.

V úvodním okně, *viz Obr. č. 5-14,* vybereme položku *„Otevřít nedávný projekt“*, kde je nám automatický nabídnout naposledy otevřeny projekt. Náš projekt je *Festo_obrazovky*.



Obr. č. 5-14 – Úvodní okno

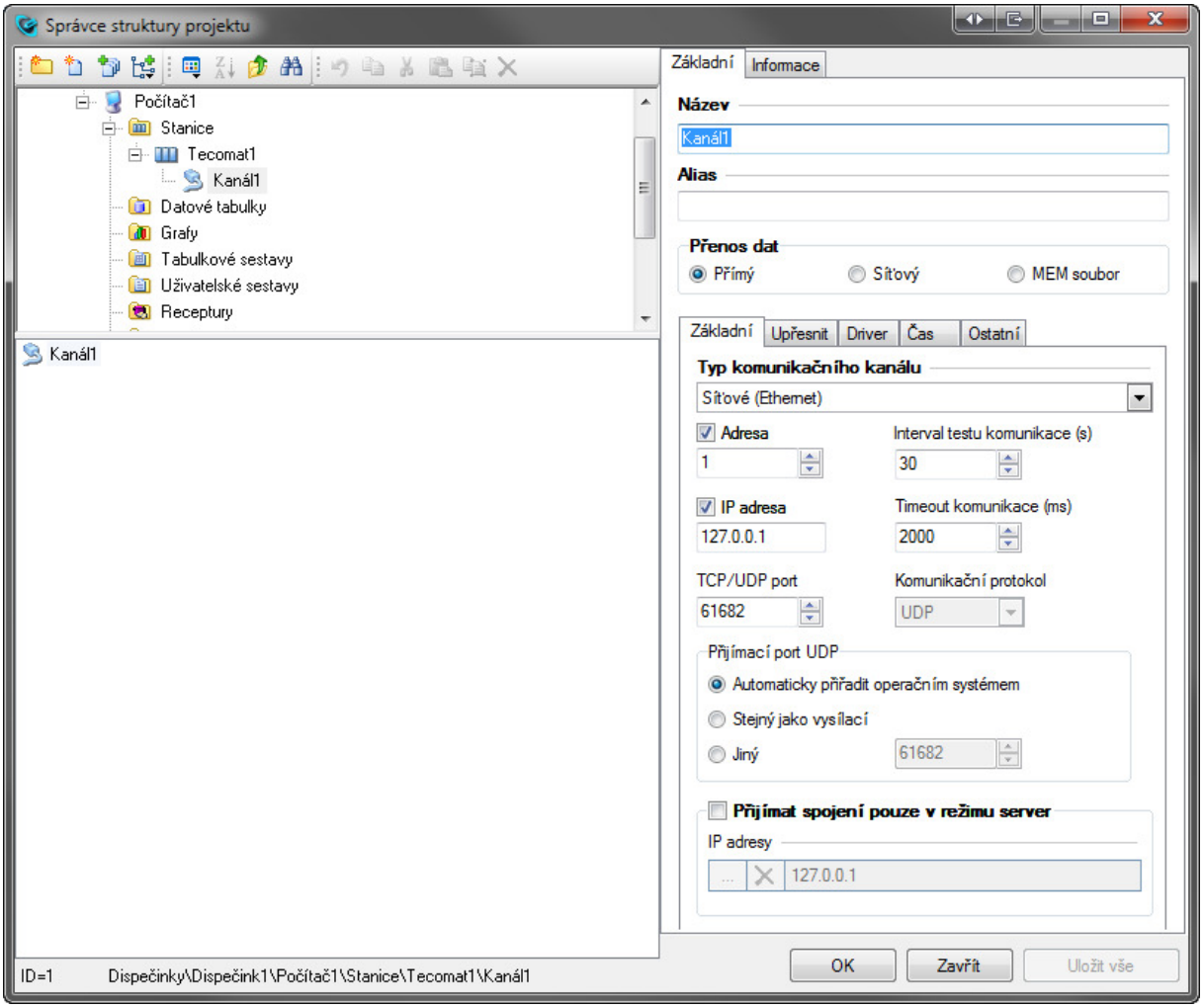
Následně již startuje vývojové prostředí Reliance, *viz Obr. č. 5-15*.



Obr. č. 5-15 – Vývojové prostředí Reliance 4 Design

5.2.2 Komunikace a proměnné

O pomoc s nastavením komunikace mezi oběma SW provozovanými na jednom PC jsem požádal technickou podporu obou produktů. Na základně jejich rad jsem realizoval v Reliance nastavení ve *Správci struktury projektu*  , *viz Obr. č. 5-16*.

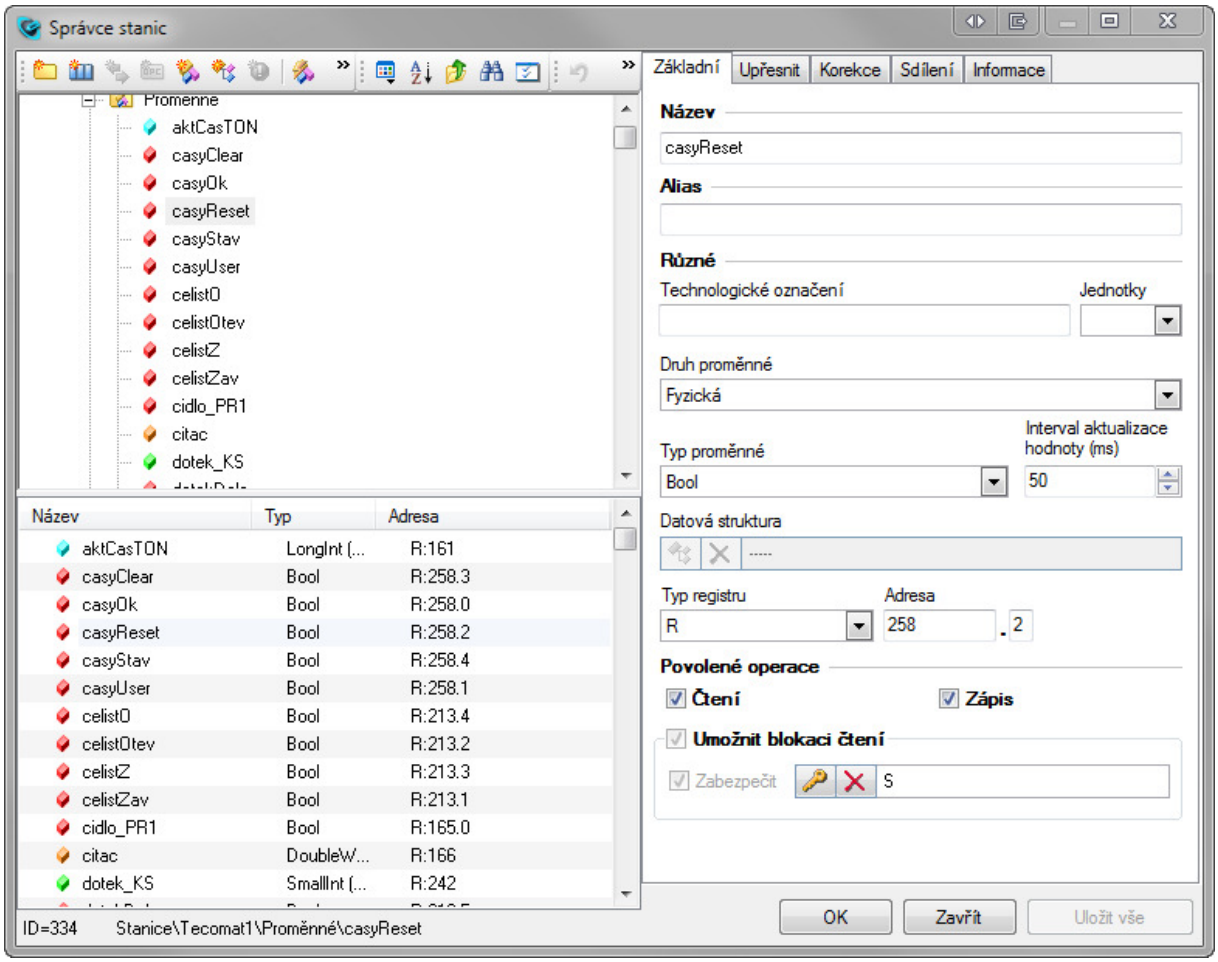


Obr. č. 5-16 – Nastavení komunikace mezi SW Reliance a Mosaic

Nastavil jsem komunikační kanál na Ethernet. Adresa odpovídá té, co jsem nastavil v Mosaicu, tj. 1.

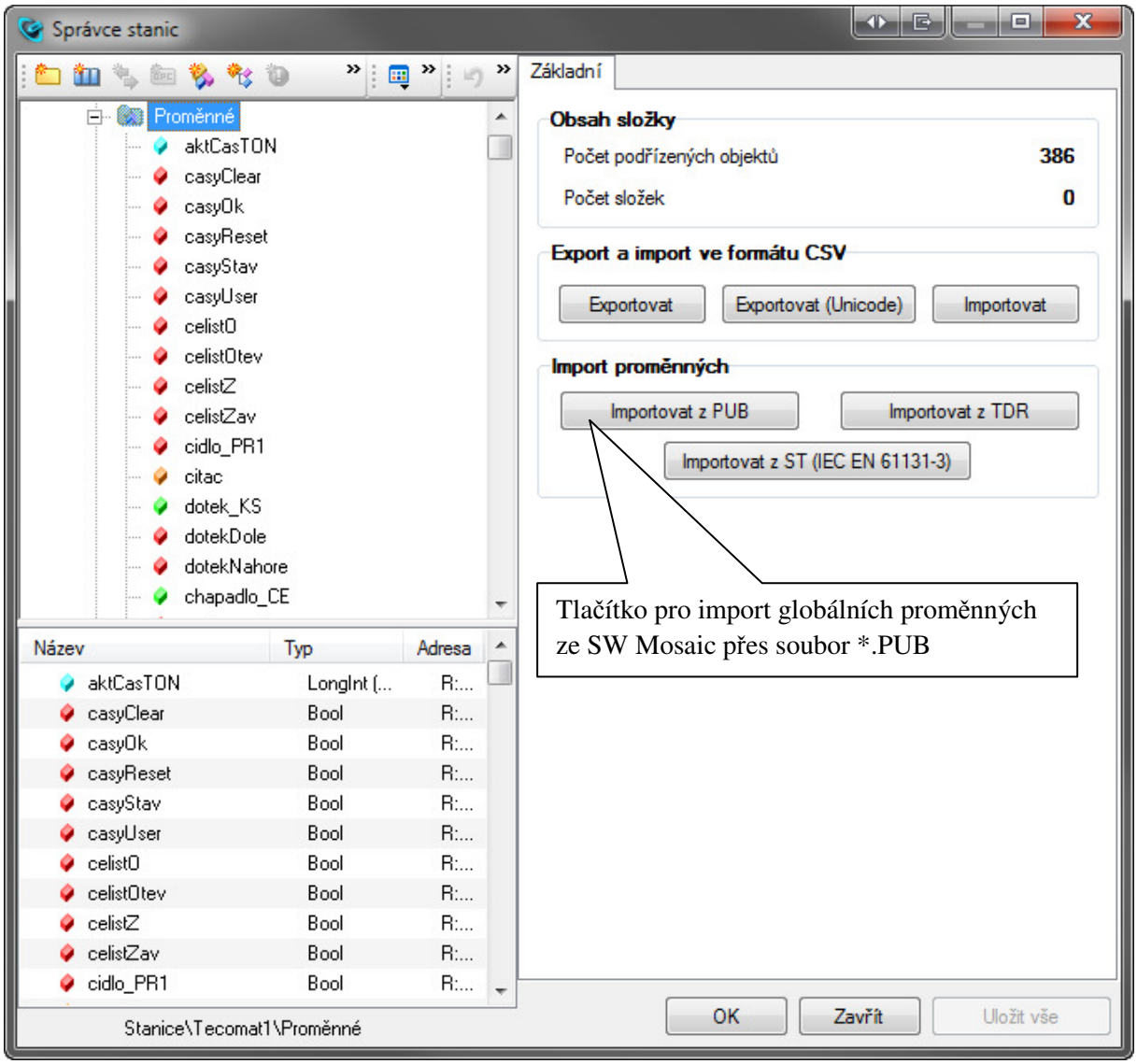
Máme-li nadefinován a funkční komunikační kanál, je potřeba dále provázat proměnné pro předávání hodnot mezi vizualizací a simulátorem.

Proměnné můžeme definovat jednotlivě, tj. např. definovat její jedinečné jméno, datový typ a typ a adresu registru v PLC ke kterému je připojena apod., *viz Obr. č. 5-17*.



Obr. č. 5-17 – Definice proměnných v SW Reliance

Pro zjednodušení je možné použít export proměnných z Mosaicu a do Realince importovat soubor *.PUB, který již obsahuje deklarace proměnných, *viz Obr. č. 5-18* .

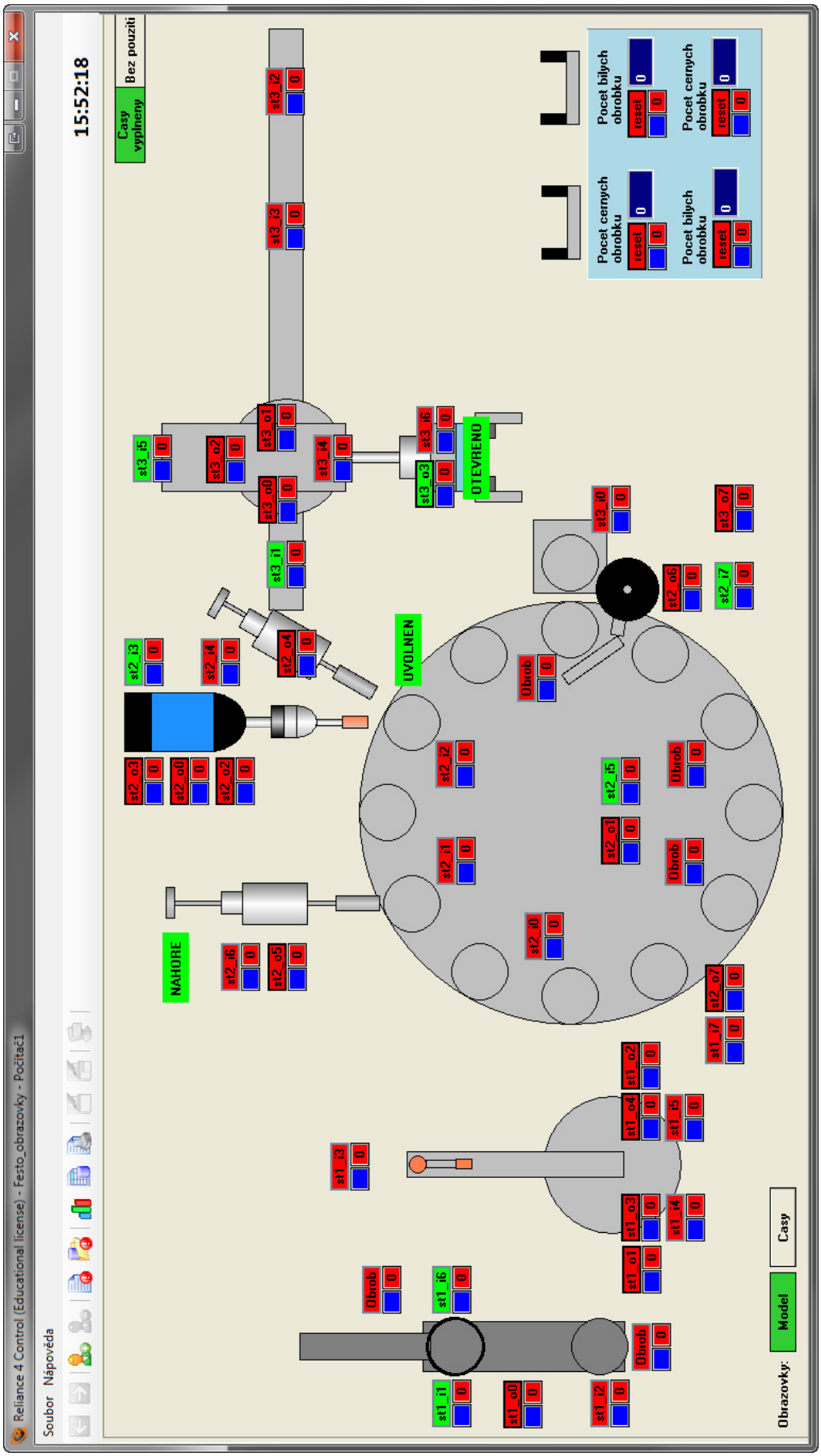


*Obr. č. 5-18 – Import globálních proměnných pomocí souboru *.PUB ze SW Mosaic*

Finální verze projektu pracuje s 325 proměnnými pro vzájemnou výměnu hodnot.

5.2.3 Obrazovka Model

Představuje hlavní obrazovku pro ovládání simulace, *viz Obr. č. 5-19*. Zobrazuje přehledně celou část sestavy v kombinaci pohledů zpředu a shora.



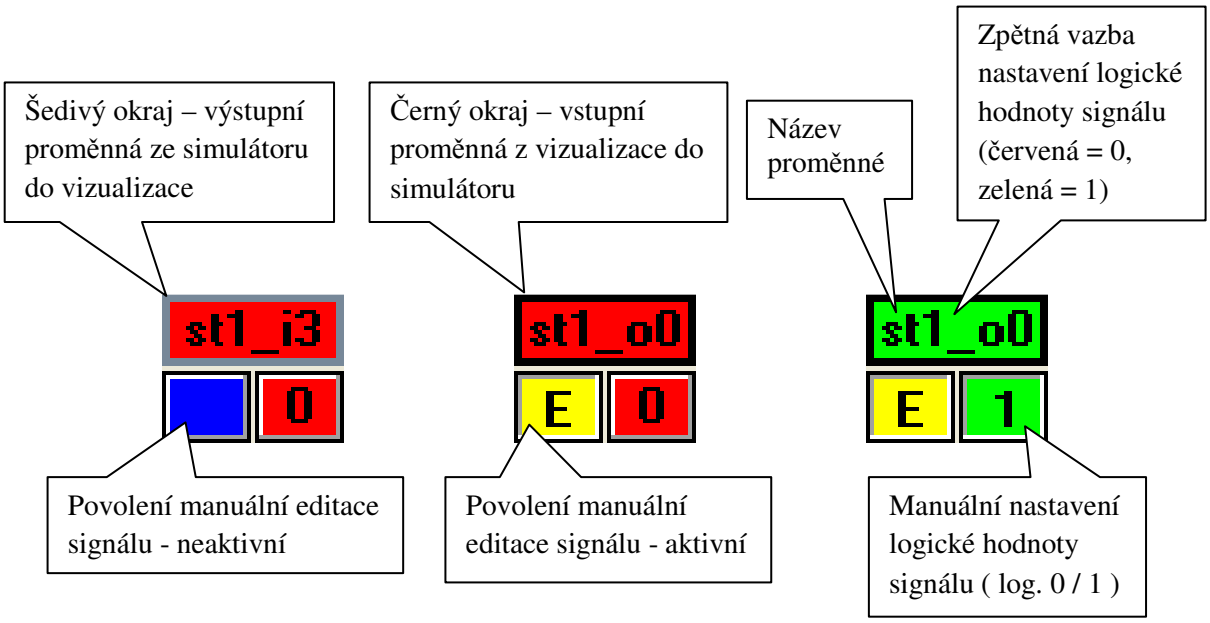
Obr. č. 5-19 – Obrazovka modelu sestavy – Model

Název obrazovky, na které jsme přítomni, je detekován zeleným podbarvením tlačítka pro volby obrazovek, které se nacházejí v levém dolním rohu obrazovky. Může nabývat stavů:

- Model (Model) – není vybrána obrazovka *Model*
- Model (Model) – je vybrána obrazovka *Model*

Z obrazovky je možné ovládat všechny signály tak, jak byly popsány v popisu fyzické sestavy, *viz Obr. č. 5-20,* včetně těch pomocných (simulace obrobku tam, kde není fyzické čidlo přítomnosti apod.).

Signály mající HW obraz jsou značeny složeninou čísla stanice a jejího signálu (vychází z *Tab. č. 4-1 – Seznam HW I/O signálů dělený dle stanic*), např. *st1_i3*, *st1_o0* atd.



Obr. č. 5-20 – Ovládání signálů na obrazovce modelu

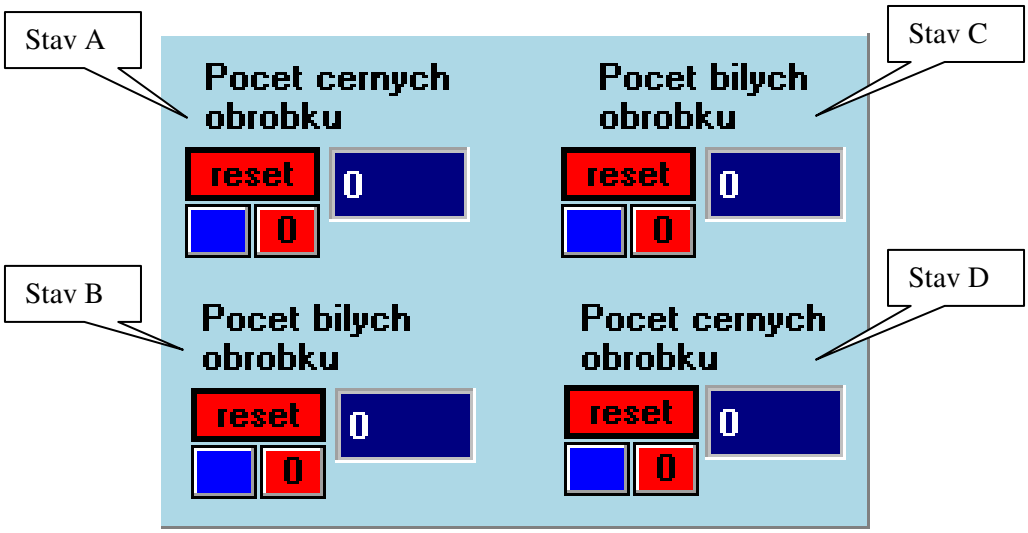
Dalším typem signálů jsou tzv. pomocné, *viz Obr. č. 5-21.* Ty nelze ovládat z vizualizace, jelikož jsou ovládány pouze logikou simulace a slouží k dodatečné informaci o stavu modelu. Většinou se jedná o detekce pomocných krajních poloh, šipky určující směr pohybu a podobné informační údaje.

Obr. č. 5-21 – Příklady využití pomocných signálů

Tyto všechny signály jsem se snažil na obrazovce umístit dle jejich skutečného rozložení na fyzickém modelu.

Dojde-li obrobek až na konec celé sestavy, je zde tříděn do zásobníků podle své barvy, *viz Obr. č. 5-22.* Toto je obsluhováno pomocí čtveřice čítačů na světle modrém podkladu, umístěny jsou vpravo dole na obrazovce. Jakmile je obrobek uvolněn nad zásobníkem, tak dle jeho barvy je inkrementována hodnota správného čítače. Celkem mohou nastat tyto 4 stavy:

A. Černý obrobek uvolněn na místě pro černou barvu – v pořádku
B. Bílý obrobek uvolněn na místě pro černou barvu - chyba
C. Bílý obrobek uvolněn na místě pro bílou barvu – v pořádku
D. Černý obrobek uvolněn na místě pro bílou barvu - chyba



Obr. č. 5-22 – Čítače obsluhující třídění obrobků dle barev

Čítače je možné resetovat manuálně tlačítkem či programově přes globální proměnné. Programové resetování by neměli používat uživatelské řídící algoritmy, jelikož to nejsou fyzické signály od sestavy. V případě přenesení do fyzického PLC pro řízení reálného sestavy nebude resetování fungovat.

5.2.4 Obrazovka Časy

Slouží pro přehledné nastavování časů simulujících dynamiku sestavy, *viz Obr. č. 5-23*. Je tvořena jediným panelem modrého podbarvení, nacházející se v pravé části této obrazovky.

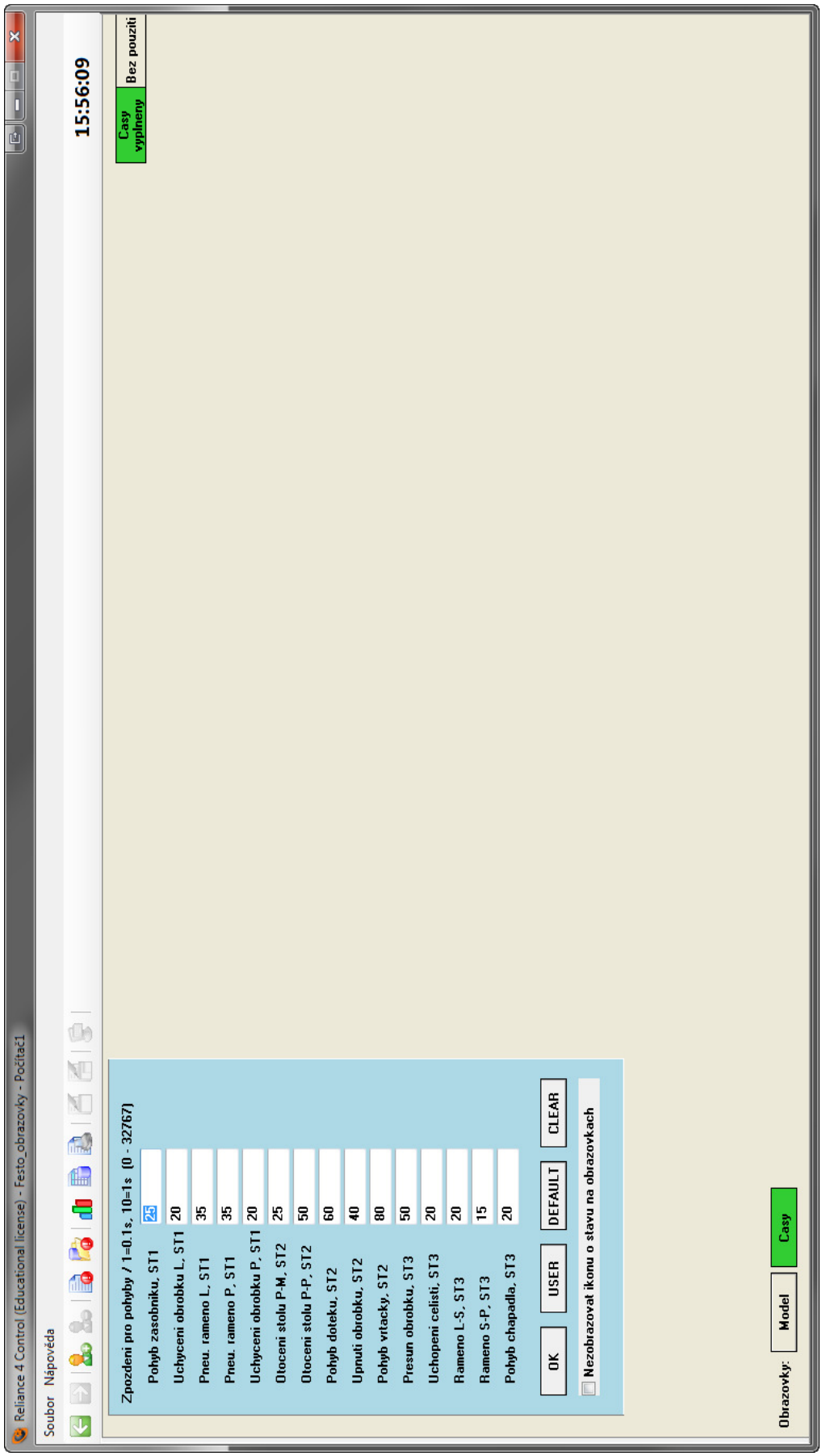
K dispozici je celkem 15 časových zpoždění, které lze ovlivňovat:

- Pohyb zásobníku, ST1 - Čas vysunutí/zasunutí zásobníku
- Uchycení obrobku L, ST1 - Zpoždění pro uchycení obrobku na levém dorazu pneu. ramene
- Pneu. rameno L, ST1 - Čas pohybu pneu. ramene ze středu doleva nebo zpět
- Pneu. rameno P, ST1 - Čas pohybu pneu. ramene ze středu doprava nebo zpět
- Uchycení obrobku P, ST1 - Zpoždění pro uchycení obrobku na pravém dorazu pneu. ramene
- Otočení stolu P-M, ST2 - Čas otočení stolu z pozice do mezipozice
- Otočení stolu P-P, ST2 - Čas otočení stolu z pozice do pozice
- Pohyb doteku, ST2 - Čas pohybu kontrolního doteku dolu/nahoru
- Upnutí obrobku, ST2 - Čas pro upnutí/uvolnění obrobku na vrtačce
- Pohyb vrtačky, ST2 - Čas představující pohyb vrtačky dolu/nahoru
- Přesun obrobku, ST3 - Přesun obrobku z ST2 na ST3
- Uchopení čelistí, ST3 - Čas zavření/otevření čelisti
- Rameno L-S, ST3 - Čas pohybu ramene z levé polohy do středové nebo zpět
- Rameno S-P, ST3 - Čas pohybu ramene ze středové polohy do pravé nebo zpět
- Pohyb chapadla, ST3 - Čas pohybu chapadla dolů/nahoru

Časy do těchto vstupních polích jsou zadávány dekadicky a to v následujícím přepočtu: zadám-li 1, odpovídá to času 0.1 vteřiny, tj. pro zadání 10 získám čas 1 vteřina. Z toho tedy plyne, že po zadání čísla získám čas desetkrát menší.

Rozsah takto zadávaných hodnot je nastaven na 0-32767. V případě, že zadám číslo mimo rozsah, objeví se varovné okno upozorňující na nekorektní zadání.

Tyto hodnoty je třeba nastavovat s uvážením. Nevhodnou kombinací časů, které spolu interně souvisí, je možné model zcela rozhodit a nebude pak vykonávat svou původní funkci. Řešením je načíst zpět výchozí hodnoty předdefinované systémem.



Obr. č. 5-23 – Obrazovka Časy

Dále jsou k dispozici 4 ovládací tlačítka:

- OK (OK) – Potvrzení aktuálně nastavených hodnot času
- USER (USER) – Návrat k původnímu (před potvrzením) nastavení hodnot časů
- DEFAULT (DEFAULT) – Vrátí hodnoty časů na hodnoty předdefinované systémem
- CLEAR (CLEAR) – Vynuluje všechny hodnoty časů

a zaškrtávací volba, umožňující skrýt ikonu o stavu časů na obrazovkách. Tato ikona nabývá stavů:

- Časy nevyplněny (Casy nevyplneny) – žádný z časů není vyplněn
- Časy vyplněny (Casy vyplneny) – alespoň jeden čas je vyplněn

Název obrazovky, na které jsme přítomni, je detekován zeleným podbarvením tlačítka pro volby obrazovek, jenž se nachází v levém dolním rohu obrazovky. Může nabývat stavů:

- Časy (Casy) – není vybrána obrazovka *Časy*
- Časy (Casy) – je vybrána obrazovka *Časy*

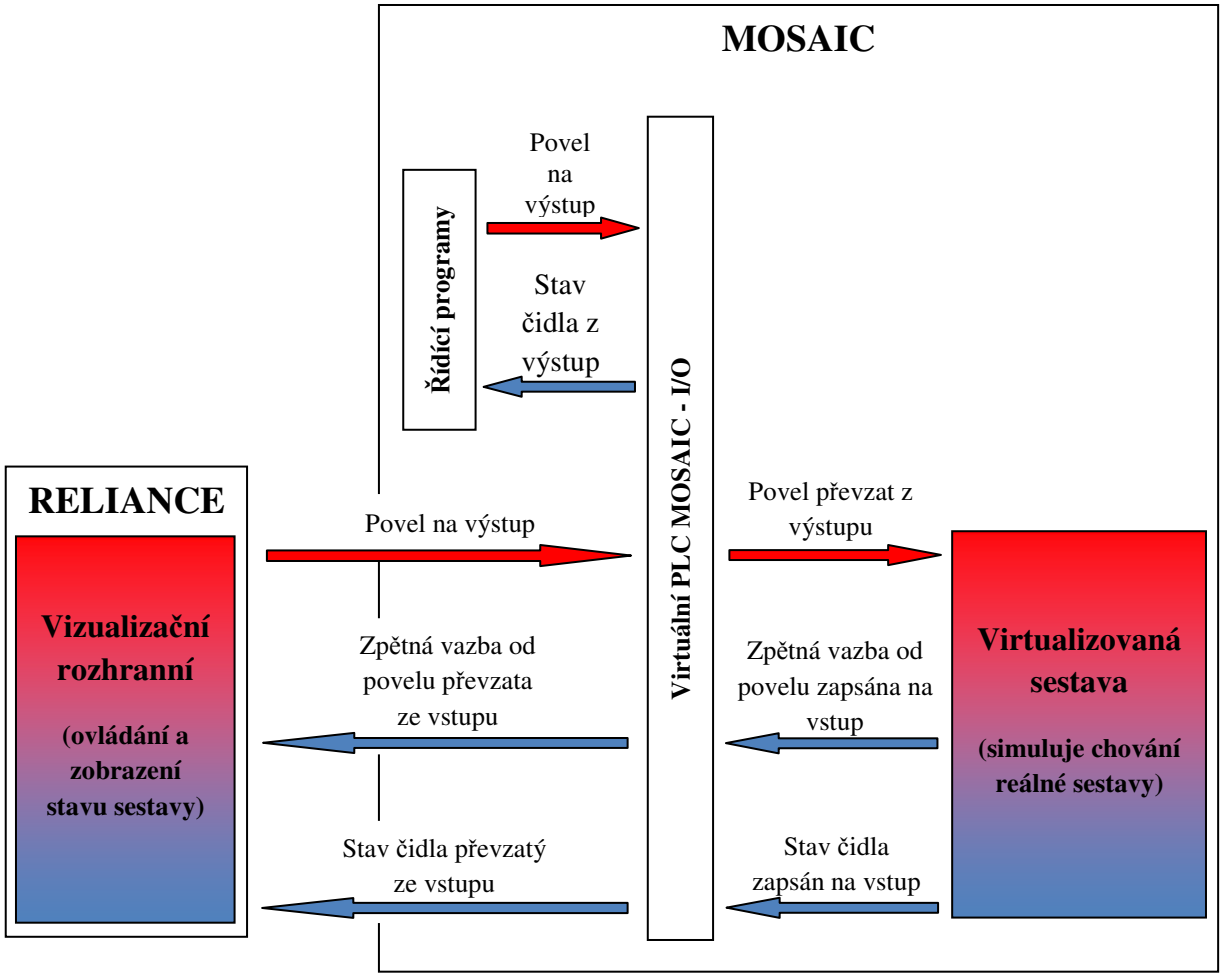
Tato obrazovka není plně obsazena. V případě potřeby a dalšího rozšiřování funkčnosti či nastavování nových parametrů sestavy je možno využít právě tohoto volného místa.

5.2.5 Spuštění projektu vizualizačních obrazovek

Máme-li hotové vizualizační obrazovky – vloženy grafické prvky a jim přiřazeny proměnné, můžeme tento celý projekt nechat přeložit a spustit pomocí jediného tlačítka . Je-li spuštěn a vykonáván kód i v Mosaic, tak při spouštění vizualizačního projektu v Reliance se naváže komunikace s Mosaic a jsou vyměňovány hodnoty proměnných, které jsou následně interpretovány vizualizačními obrazovkami.

6 Řízení soustavy

Při realizaci řízení soustavy vycházím z doposud vytvořeného projektu. Ten nyní vhodně rozšířím o část řídících algoritmů v Mosaicu a nutných ovládacích prvků v Reliance, *viz Obr. č. 6-1*. Původní blokové schéma upravím do tvaru:



Obr. č. 6-1 – Blokové schéma virtualizované soustavy pomocí SW Mosaic a Reliance rozšířené o řídící algoritmy

Řídící algoritmy jsou realizovány v prostředí Mosaic, stejně tak jako celý model. Algoritmy simulují chování uživatele a dle logického vyhodnocování stavu modelu na základě hodnot z čidel nastavují povely na akční členy. Původní režim *Manuálně* nenahrazují, ale rozšiřují možnosti virtuální soustavy o další pracovní režimy, jenž si pak bude moci uživatel volit.

Pro potřeby ovládání řídících režimů rozšířím původní obrazovku *Model* o ovládací panel. Zde bude možnost volby režimu a jeho základní obsluha. Rovněž do téže obrazovky

vznikne panel pro obsluhu obrobků, tj. aby mohly automatické režimy probíhat téměř nepřetržitě.

Dalším rozšířením původní verze bude realizace logiky a k tomu ovládací obrazovky pro maskování signálů. To umožní simulovat chyby technologie – upadlý kontakt, nefunkční akční prvek či čidlo apod. Toto bude k dispozici jen pro signály, které obsahuje i skutečný model.

6.1 Algoritmus řízení soustavy v Mosaic

Řízení je tvořeno algoritmem, který postupně vykonává povely, tj. na akční členy sestavy posílá log. hodnoty povelů na základě aktuálního stavu sestavy, který získá ze stavu sledovaných čidel. V tomto případě na rozdíl od režimu *Manuálně*, který byl úplně nezávislý na průběhu obrobku jednotlivými stanicemi, je nutno dodržet technologický proces. Obrobek tedy postupně proplouvá sestavou a prochází všemi stavy a stanicemi od začátku až do konce technologického procesu. Tento problém řízení odpovídá konečnému automatu, tj. množina stavů je konečná.

Toto lze modelovat např. pomocí Petriho sítí. Tj. máme soustavu míst (stavů), která jsou mezi sebou propojeny předáváním si obrobku (hrany). V každém místě může být max. jeden obrobek (token).

Na základě této analýzy jsem realizoval dva automatické řídící algoritmy

- režim Jednotlivě
- režim Průběžně

Tyto algoritmy jsou realizovány programem *prgOvladani.ST*. Jelikož mají některé části kódu společné, realizoval jsem v tomto programu oba tyto režimy. Pomocí volby režimu z vizualizace vhodně nastavím podmínky a je vykonán pouze žádaný automatický řídící režim. Jeho bloková struktura je, *viz Obr. č. 6-2.*

prgOvladani.ST

```
PROGRAM prgOvladani  // zacatek programu

VAR  // deklarace lokalnich promennych

    Deklarace lokálních proměnných, jsou použitelné pouze pro toto FB

END_VAR  // konec deklarace

    Obsluha časovačů pracující s dynamikou sestavy

    Spuštění programu tlačítkem START s navolením jednoho z automatických režimů

    Postupné prochází sestavy s nastavováním povelů na základě stavů čidel

    Obsluha rotačního stolu pro režim Jednotlivě

    Obsluha rotačního stolu pro režim Průběžně

    Postupné prochází sestavy s nastavováním povelů na základě stavů čidel

    Tlačítkem RESET nastavení lokálních proměnných na původní hodnoty

END_PROGRAM  // konec programu
```

Obr. č. 6-2 – Blokové schéma programu automatických řídících režimů, prgOvladani.ST

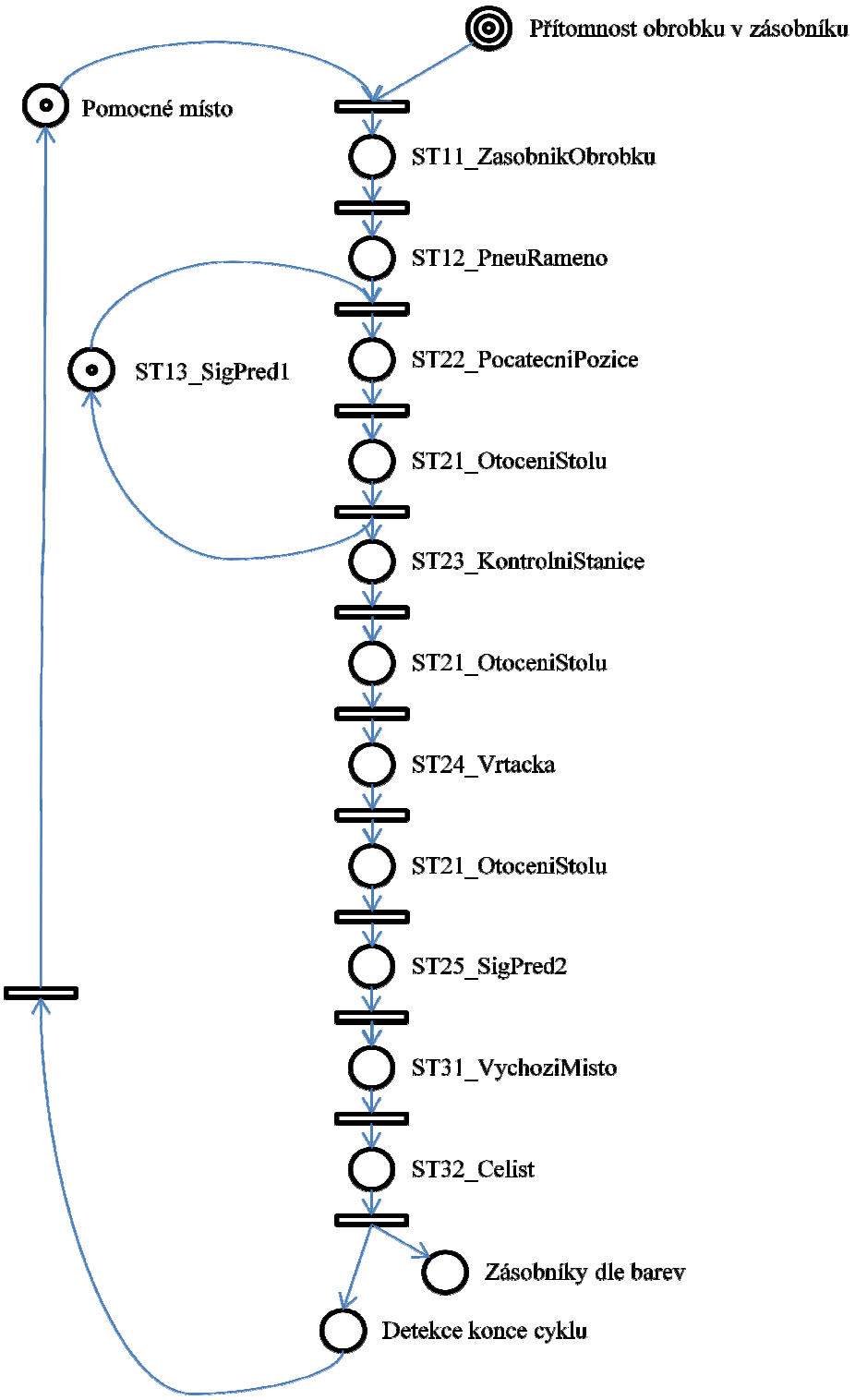
6.1.1 Režim Jednotlivě

Jak už název vypovídá, obrobky se budou v sestavě vyskytovat jednotlivě. Je tím myšleno, že sestava bude zpracovávat právě jeden obrobek, nebo-li další obrobek bude vložen do procesu až po uložení obrobku do koncového zásobníku a nastavení manipulátoru do výchozí polohy.

Tento stav bude cyklicky probíhat, než bude vyčerpán počet zadaných obrobků. Po posledním obrobku se nastaví sestava do výchozí polohy.

Nutnou podmínkou pro spuštění tohoto režimu je počet obrobků ve vstupním zásobníku větší než nula.

Petriho síť k tomuto režimu, *viz Obr. č. 6-3*

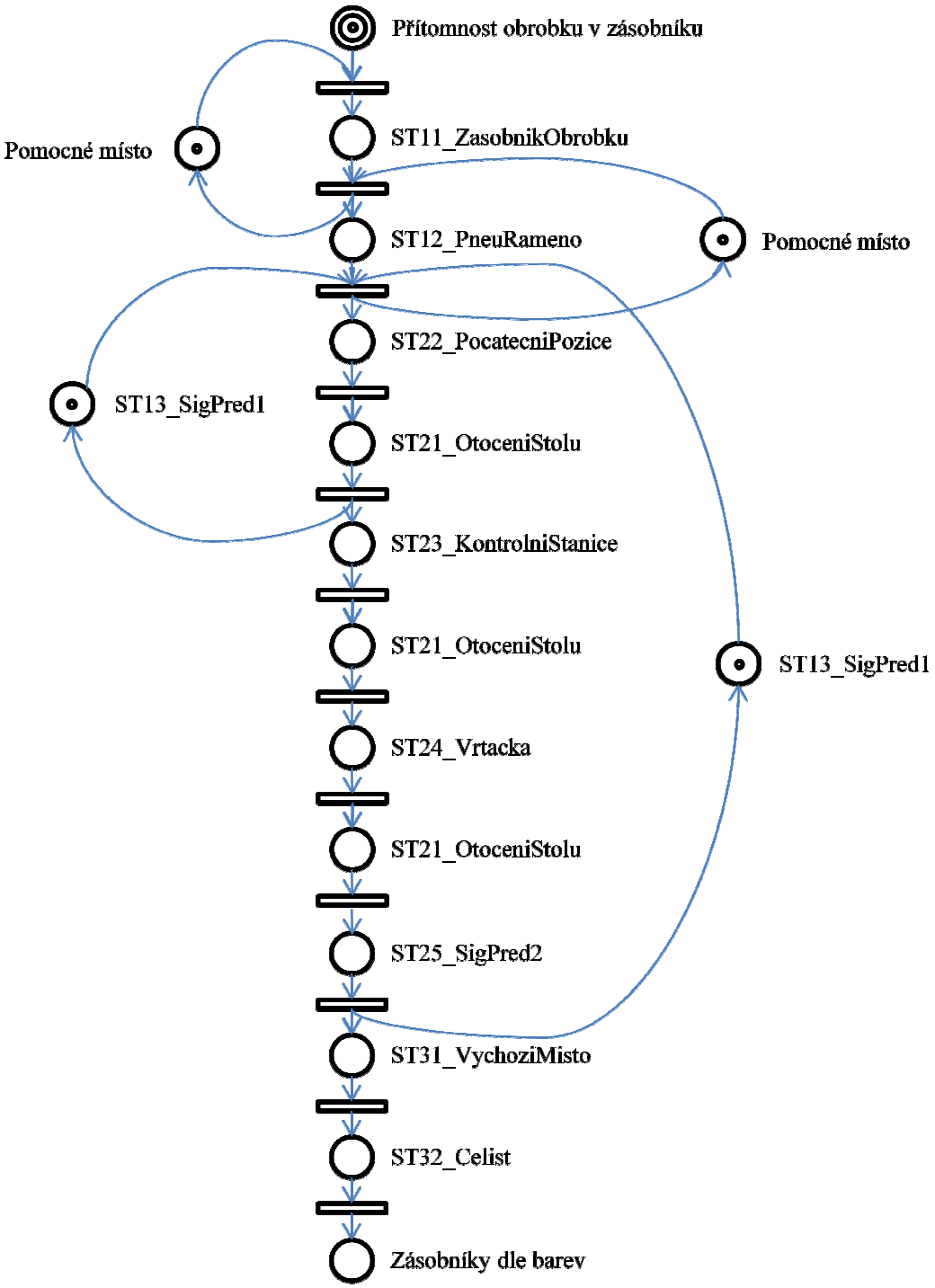


Obr. č. 6-3 – Petriho síť režimu Jednotlivě

6.1.2 Režim Průběžně

Tento režim využívá sestavu efektivněji. V sestavě se zároveň vyskytují až 4 obrobky současně. Ke zpracování stejného množství obrobků tedy dojde mnohem dříve než v režimu *Jednotlivě*.

Původní myšlenku, aby obrobky byly zpracovávány hned za sebou, jsem byl nucen opustit. Fyzický model nebyl při svém pořizování z finančních důvodů osazen takovým množstvím čidel, jaký by byl ideální a nebylo tak možné na rotačním stole zpracovávat více než jeden obrobek současně. Rotační stůl patří mezi úzké místo procesu, které mě omezilo při realizaci ještě efektivnějšího režimu. Petriho síť k tomuto režimu, *viz Obr. č. 6-4*



Obr. č. 6-4 – Petriho síť režimu Průběžně

Tento stav bude cyklicky probíhat, než bude vyčerpán počet zadaných obrobků. Po posledním obrobku se nastaví sestava do výchozí polohy.

Nutnou podmínkou pro spuštění tohoto režimu je počet obrobků ve vstupním zásobníku větší než nula.

6.1.3 Režim Uživatel

Slouží k realizaci uživatelského kódu. Programy pro tyto tři uživatelské režimy jsou již v projektu předdefinovány (nelze je generovat automaticky) *prgUser1*, *viz Obr. č. 6-5, prgUser2* a *prgUser3*. Vygenerovány jsou v jazyce ST, jenž se volí při generování každého

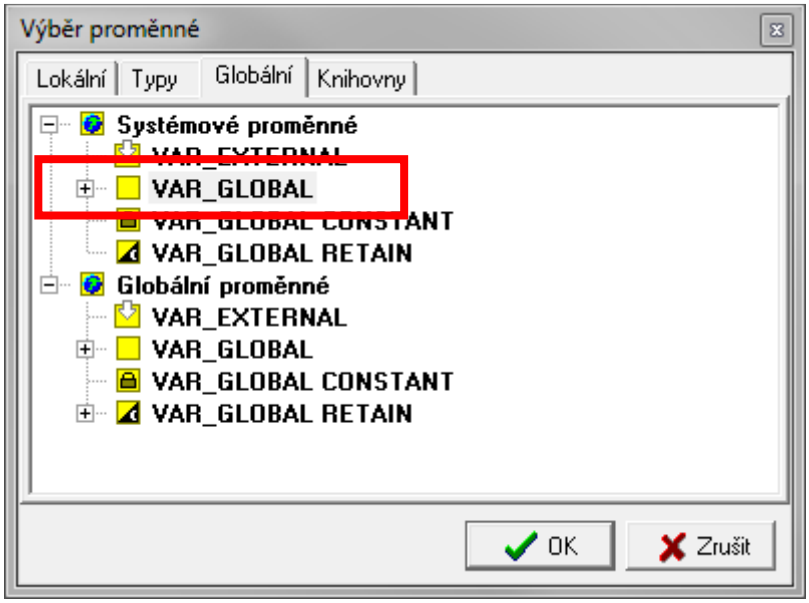
```
PROGRAM prgUser1
(* FB pro realizaci uzivateleskeho programu, v menu vizualizace jako volba
"Uzivatel 1"
*)

  VAR_INPUT
  END_VAR
  VAR
  END_VAR
  VAR_OUTPUT
  END_VAR
  VAR_TEMP
  END_VAR

  // PO VYBRANI REZIMU "UZIVATEL 1" A STISKU TLACITKA "START" JE SPLNENA...
  // NASLEDUJICI PODMINKA A JEJI VYKONAVAN JEJI KOD

  IF ( ovladaniRezimu = 1 AND volbaRezimu = 3 ) THEN

  // ZDE BUDE UZIVATLUV KOD JENZ SE MA VYKONAT PO STISKU VYBERU TOHOTO REZIMU...
  // A STISKU TLACITKA START

  END_IF;

  // PO STISKU TLACITKA "RESET" JE SPLNENA NASLEDUJICI PODMINKA, VE KTERE JE...
  // POTREBA  NASTAVIT VSECHNY LOKALNE POUZITE PROMENNE DO PUVODNIHO (VYCHOZIHO)...
  // STAVU

  IF ( ovladaniRezimu = 3 ) THEN

  // ZDE BUDOU VSECHNY UZIVATELEM POUZITE LOKALNI PROMENE NASTAVENY NA VYCHOZI...
  // HODNOTU

  END_IF;


END_PROGRAM
```

Obr. č. 6-5 – Předdefinovaný program pro uživatelský kód, prgUser1.ST

nového programu. Není tedy problém při vytváření dalšího uživatelského programu tento jazyk změnit na jeden ze čtyř definovaných normou. V rámci projektu je tak možné kombinovat bloky tvořené různými programovacími jazyky.

Celkový počet uživatelských programu je omezen velikosti paměti PLC.

Uživateli jsou k dispozici všechny globální proměnné včetně těch pomocných pro vizualizaci a simulaci. Je důrazně doporučeno vybírat globální proměnné pouze ze skupiny **Systémových proměnných,** *viz Obr. č. 6-6*. Tyto proměnné odpovídají fyzickým I/O signálům PLC, resp. reálné sestavy.



Obr. č. 6-6 – Práce s předdefinovanými globálními proměnnými

6.2 Rozhranní řízení soustavy v Reliance

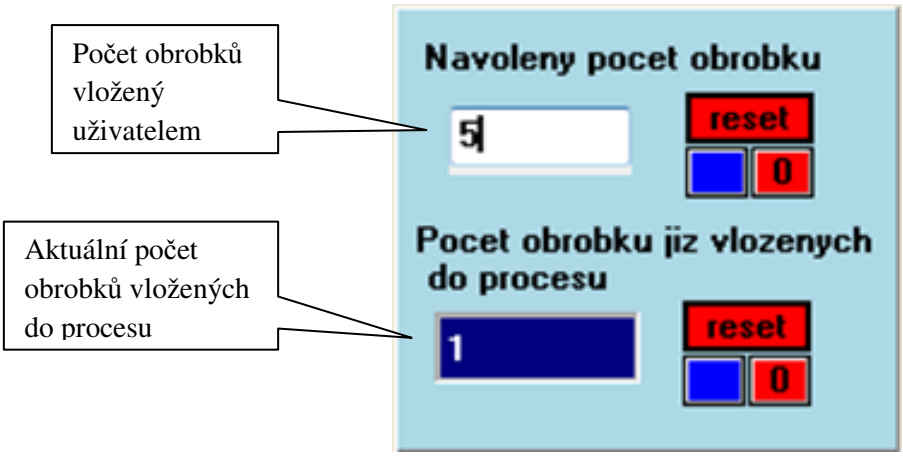
I zde jsem vycházel z původního projektu, který jsem vhodně rozšířil. Původní obrazovku modelu jsem rozšířil o obsluhu řídících režimů a správu obrobků. Dále jsem přidal i další obrazovku – *Poruchy*.

6.2.1 Obrazovka modelu sestavy rozšířená o obsluhu řízení

Jak už bylo zmíněno, v Mosaicu jsem realizoval dva automatické režimy: *Jednotlivě* a *Průběžně*. Pro tyto potřeby jsem i přidal do původní obrazovky modelu panel na volbu a ovládání režimů a čítač obrobků, *viz Obr. č. 6-9 a 6-10*.

Panel pro obsluhu obrobků, *viz Obr. č. 6-7,* se nachází v levé horní části obrazovky modelu a je světle modrého podkladu. Skládá se ze vstupního pole, kde uživatel zadá počet obrobků, které se mají zpracovat sestavou. Dolní, modře podbarvené pole, zobrazuje počet obrobků již vložených do procesu. Programové resetování by neměly používat uživatelské řídící algoritmy, jelikož to nejsou fyzické signály od sestavy. V případě přenesení do fyzického PLC pro řízení reálného sestavy nebude resetování fungovat.

Volba a obsluha režimu je realizována panelem nacházejícím se obrazovce hned vedle předchozího panelu, *viz Obr. č. 6-8*. Barva pokladu je oranžová. Levá část panelu slouží k výběru režimu chodu sestavy. Výchozím stavem je režim *Manuálně*. Ten jako jediný je spuštěn hned po jeho navolení, tj. pro jeho chod nejsou nutné žádné další spouštěcí podmínky. Pro předdefinované automatické režimy platí podmínka, že musí být navolen počet obrobků větší než nula a poté se režim spustí stiskem tlačítka START. Tlačítko PAUSE slouží k pozastavení režimu v jeho aktuálním stavu a RESET vrátí sestavu do výchozího stavu, včetně nulování čítačů obrobků. Zůstane jen navolený režim a nastavení ostatních obrazovek.

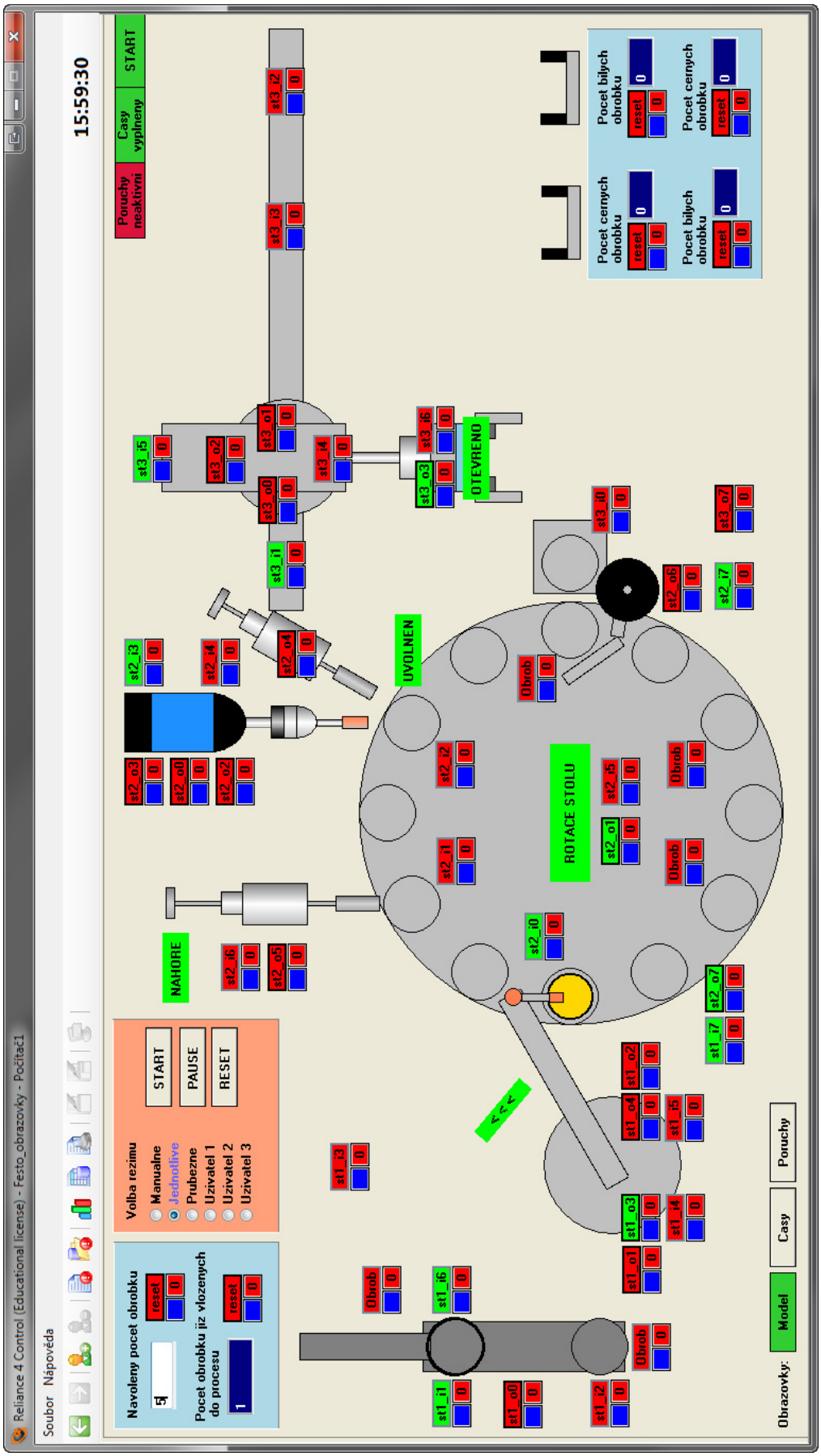


Obr. č. 6-7 – Panel čítače obrobků

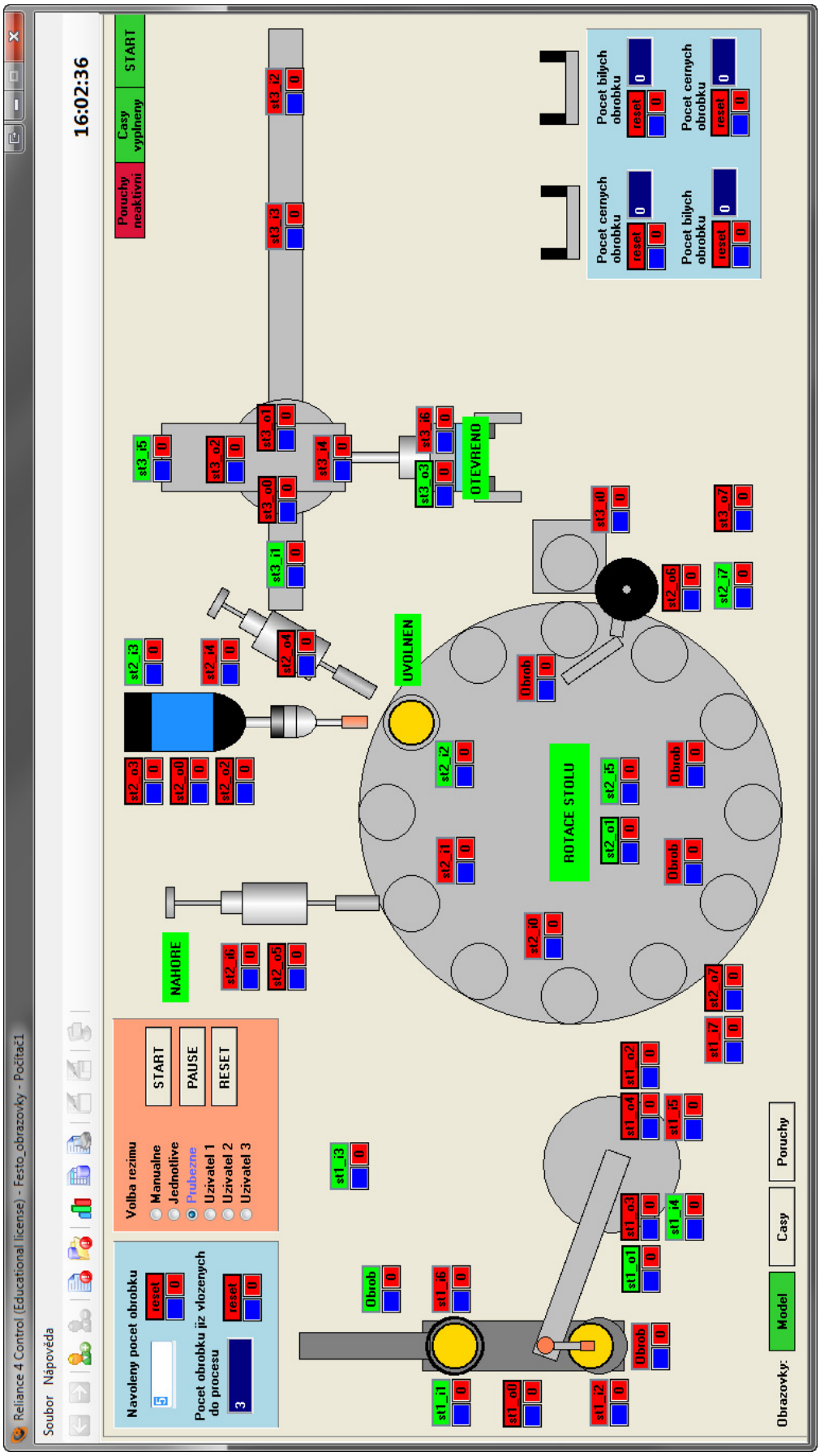
Obr. č. 6-8 - Panel volby a ovládání režimů

Stav modelu je zobrazen pomocí indikátoru stavu, který se nachází v pravé horní části obrazovky a je viditelný i na všech ostatních obrazovkách. Indikátor stavu může nabývat:

- **Bez použití** (Bez pouziti) - po zapnutí simulace či po tlačítku RESET
- **START** (START) - sestava běží, tj. provádí se aktuálně zvolený řídící režim (neplatí pro režim *Manuálně*)
- **PAUSE** (PAUSE) - aktuálně prováděný řídící režim je pozastaven (neplatí pro režim *Manuálně*)
- **RESET** (RESET) - aktuálně prováděný řídící režim je pozastaven a veškeré signály jsou navráceny do výchozí hodnoty (neplatí pro režim *Manuálně*)



Obr. č. 6-9 – Obrazovka modelu sestavy rozšířená o obsluhu řízení – režim Jednotlivě

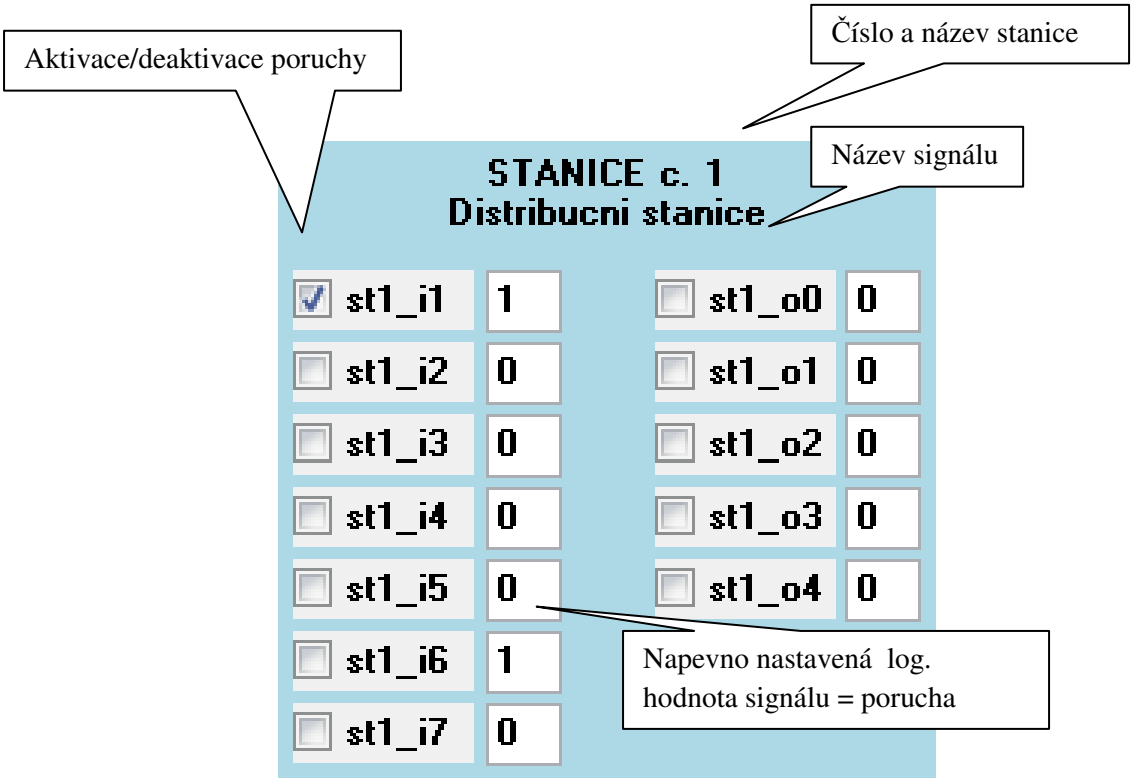


Obr. č. 6-10 – Obrazovka modelu sestavy rozšířená o obsluhu řízení – režim Průběžně

6.2.2 Obrazovka Poruchy

Tuto obrazovku můžeme také zahrnout do obrazovek podporujících řízení sestavy, *viz Obr. č. 6-12*. Slouží k simulování chyb sestavy, tj. můžeme napevno nastavit logickou hodnotu povelu či čidla, kterou není možné řídícím algoritmem přepsat. Tím simulujeme např. utržení kontaktu od čidla – dostáváme jinou logickou hodnotu, než jaká by měla být.

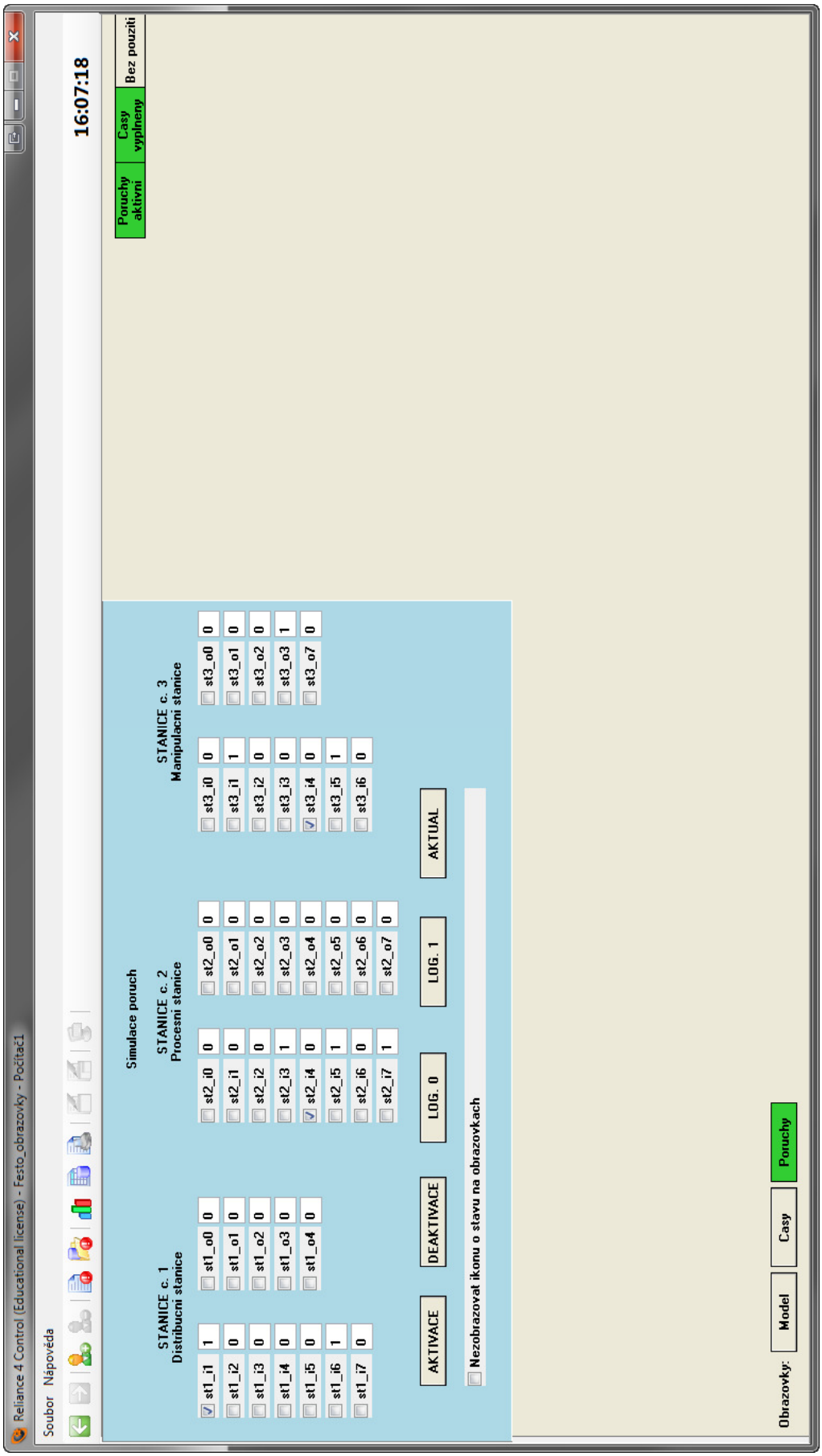
Princip ovládání je takový, že nejdříve si uživatel vybere signál, který chce maskovat. U tohoto signálu je zaškrtávací políčko, aktivující jeho maskování. Poté již jen stačí nastavit požadovanou logickou hodnotu (log. 0 nebo log.1), *viz Obr. č. 6-11*.



Obr. č. 6-11 – Panel čítače obrobků, panel volby a ovládání režimů

K ovládání této obrazovky máme k dispozici 5 tlačítek

- AKTIVACE (AKTIVACE)- aktivuje všechny poruchy
- DEAKTIVACE (DEAKTIVACE) - deaktivuje všechny poruchy
- LOG. 0 (LOG. 0)- všechny log. hodnoty signálů nastaví do log. 0
- LOG. 1(LOG. 1) - všechny log. hodnoty signálů nastaví do log. 1



Obr. č. 6-12 – Nově přidaná obrazovka Poruchy

- AKTUAL(AKTUAL) - všechny log. hodnoty signálů nastaví dle aktuálního stavu modelu

a zaškrtávací volba, umožňující skrýt ikonu o stavu poruch na obrazovkách. Tato ikona nabývá stavů:

- Poruchy neaktivní (Poruchy neaktivni) - žádná porucha není aktivní
- Poruchy aktivní (Poruchy aktivni) - alespoň jedna porucha je aktivní

Název obrazovky, na které jsme přítomni, je detekován zeleným podbarvením tlačítka pro volby obrazovek, jenž se nacházejí v levém dolním rohu obrazovky. Může nabývat stavů:

- Poruchy (Poruchy) – není vybrána obrazovka *Poruchy*
- Poruchy (Poruchy) – je vybrána obrazovka *Poruchy*

7 Zhodnocení práce

V základu jsem se seznámil s doporučenými vývojovými nástroji, Mosaic pro programování PLC a Realiance pro tvorbu vizualizačních obrazovek. Jelikož jsem s nimi doposud nepracoval, bylo úvodní seznamování o to náročnější.

Dalším krokem bylo vybrat vhodného zástupce soustavy, jenž mám virtualizovat. Na základě konzultace s oponentem mé diplomové práce jsme vybrali soustavu diskrétního typu - sestavu výukového modelu firmy Festo, jenž je umístěn v Centru odborné přípravy Sezimovo Ústí (COPSU). Zde je použita jako výuková pomůcka a studenti pro ni realizují řídící algoritmy.

S touto sestavou jsem se důsledně seznámil. Osobně jsem několikrát navštívil COPSU, kde mi byl velmi nápomocen pan Ing. Jan Fuka, který má danou sestavu na starosti. Poskytl mi dokumentaci a ukázal mi sestavu v činnosti. Z toho jsem pořídil řadu fotografií a pro osobní potřeby i videozáznam z běhu sestavy.

Pokračování práce spočívalo ve virtualizaci celé této sestavy, což bylo nejtěžší částí této diplomové práce. Přímo s virtualizací sestavy pomocí PLC jsem se doposud nesetkal, tudíž jsem vyzkoušel několik přístupů, než jsem nalezl ten správný a ten dále rozvíjel do postupné virtualizace celé sestavy. Logiku jsem realizoval v Mosaic na virtuálním PLC Tecomat řady 700 řady, obrazovky byly tvořeny v Reliance s výchozím širokoúhlým rozlišení 1366x768 pixelů. Tento model si je možné z vizualizace postupně vyzkoušet a seznámit se s jeho funkcí bez nutnosti mít k dispozici fyzický model.

Po dokončení virtuální sestavy jsem realizoval její řízení. Tj. původní koncepci jsem rozšířil o vhodně zakomponované řízení. Logika řízení je opět realizována v Mosaic, v Reliance jsem rozšířil původní obrazovky a přidal nové. K dispozici jsou ode mne dodané dva automatické řídící režimy a předpřipravené bloky pro vložení uživatelských kódů (v MOSAICU i RELIANCE).

Výsledkem této práce je tedy virtuální sestava původního fyzického modelu. Rozšířena je i o řídící algoritmy a grafické rozhranní tvořené několika obrazovkami pro kompletní správu sestavy.

Virtualizovaná sestava přináší nové možnosti do výuky. Přínosů této diplomové práce je hned několik.

- Zabránění mechanickému poškození modelu, ladění probíhá na virtuálním modelu
- Umožnění studentům pracovat na svém řídícím algoritmu i mimo výuku, tj. v rámci vývoje algoritmu není nutný fyzický přístup modelu. Nevznikají typické problémy s časovým využitím modelu mezi více studenty
- Bude-li fyzická sestava rozšířena o fyzické PLC podporující normu programování 61 131-3, bude možné totožný kód nahrát do reálného PLC a výsledný algoritmus řízení předvést na fyzickém modelu - výší verze řízení než je nyní ve výuce realizována
- Bude-li fyzický model v budoucnu rozšířen o potřebná čidla, bude možné po drobných úpravách v Realiance používat vizualizaci i ke sledování reálného modelu

Výsledný projekt jsem byl rovněž osobně předvést panu Ing. Janu Fukovi v COPSU. S výsledky byl velmi spokojen a jeho písemné vyjádření je k dispozici v příloze 9.3. Nyní spolu řešíme fázi zavádění mé práce do jejich výuky.

I po oficiálním odevzdání práce na ČVUT FEL jsem COPSU přislíbil svou podporu v oblastech:

- Na základě vzájemné dohody a potřeb vypracuji podrobný uživatelský manuál
- Předvedu projekt ostatním vyučujícím a zaškolím je
- Dle potřeb provedu lehké modifikace projektu, tak aby byl skutečně použitelný do výuky
- Vypracuji verzi projektu pro vyučující a pro studenty
- Budu spolupracovat na vytvoření zadání školních úloh pro studenty

8 Zdroje

[1] Teco a.s. – výrobce PLC Tecomat a SW Mosaic, http://www.tecomat.cz

- dokumentace k SW Mosaic

technická podpora přes e-mail

Začínáme v prostředí MOSAIC.pdf

Webová prezentace firmy

[2] Teco a.s. – výrobce PLC Tecomat a SW Mosaic, http://www.tecomat.cz

- dokumentace k normě 61 131-3

Programování PLC TECOMAT podle IEC 61131-3.pdf

Knihovny pro programování PLC TECOMAT podle IEC 61131-3.pdf

[3] Geovap s.r.o. – výrobce SW Reliance, http://www.reliance.cz

- dokumentace k SW Reliance

technická podpora přes e-mail

FirstSteps_CSY.pdf

Reliance 4 Design.pdf

Runtime_CSY.pdf

[4] COPSU – VOŠ, SŠ, Centrum odborné přípravy Sezimovo Ústí, http://www.copsu.cz

- dokumentace k fyzické soustavě Festo

Ing. Jan Fuka – Návod k obsluze, učební text.pdf

[5] Bakalářská práce Bc. Martina Cicvárka – Knihovna funkčních bloků pro analýzu tvaru a predikci průběhu číslicových signálů v PLC

9 Přílohy

9.1 Obsah přiloženého DVD

1) Obsah DVD v souboru *Obsah_DVD.txt*
2) Elektronická podoba této práce v souboru *DP_2010_Siska_Matej.pdf*
3) Adresář *Dokumentace*
    a. Model Festo
    b. Norma IEC 61 131
    c. SW Mosaic
    d. SW Reliance
4) Adresář *Pomocné Office soubory*
5) Adresář *Programové vybavení*
6) Adresář *Výsledky diplomové práce*
    a. SW Mosaic
    b. SW Reliance
    c. Kódy ze SW Mosaic v souborech TXT

9.2 Zdrojový kód funkčního bloku ST11_ZasobnikObrobku

```
FUNCTION_BLOCK ST11_ZasobnikObrobku

(*FB simulujici zasobnik obrobkuna ST1
vstupy: O0
vystupy: I1, I2, I6
*)

  VAR_INPUT

    po100msR_m : Bool R_EDGE; // detekce vzestupne hrany signalu po100ms
    po100msF_m : Bool F_EDGE; // detekce spadove hrany signalu po100ms

  END_VAR
  VAR_OUTPUT
  END_VAR
  VAR

    CTUD1 : CTUD; // instance bloku obousmerneho citace
    QU1_m : Bool; // vystup citace - je dosazena pozadovana hodnota citace
    QD1_m : Bool; // vystup citace - je dosazena nulova hodnota citace
    PV1_m : INT := 25; // mezni hodnota citace - pozdeji nastavovano z vizualizace
    CTUD2 : CTUD; // instance bloku obousmerneho citace
    dojel : Bool; //
    resetCTUD : Bool; // univerzalni reset citacu

  END_VAR
  VAR_IN_OUT

  //IN
    st1_i6_m : Bool; // cidlo: zasobnik obrobku je prazdny
    st1_o0_m : Bool; // povel: zasunuti valce zasobniku - vyjeti obrobku
    ovladaniRezimu_m : Int; // resetovani a uvedeni do puvodniho stavu

  //OUT
    st1_i1_m : Bool; // cidlo: valec zasobniku vysunut
    st1_i2_m : Bool; // cidlo: valec zasobniku zasunut
```

```
  // VIZU
    obrobek_m : Bool; // vlozeni / vyjmuti obrobku z procesu
    obrobek1_m : Bool; // ovladani viditelnosti obrobku ve vizualizaci
    obrobek2_m : Bool; // ovladani viditelnosti obrobku ve vizualizaci
    st1_i6_t_m : Bool; // promenna od tlacitka aktivace ovladani promenne st2_i6 z vizualizace
    CU1_m : Bool; // ovlada vzestupny vstup citace
    CD1_m : Bool; // ovlada sestupny vstup citace

  // POCITANI OBROBKU
    pocetVsechR_m : Bool; // RESet nastaveny pocet obrobku ktere jsou ve vychozim zasobniku
    pocetAktR_m : Bool; // RESet aktualni pocet obrobku ktere byli vlozeny do procesu
    pocetAkt_m : Int; // aktualni pocet obrobku ktere byly vlozeny do procesu
    pocetVsech_m : Int; // nastaveny pocet obrobku ktere jsou ve vychozim zasobniku

  // HODNOTY CITACE
    zasobnik_ZO_m : Int; // Cas vysunuti/zasunuti zasobniku

  END_VAR
  VAR_TEMP
  END_VAR


// **** VERZE S VYUZITIM CITACU PRO VYPOCET POLOHY A SIMULACI ZPOZDENI - bez pohybu ve vizu ****
      // CITACE
      // Obsluha pohybu
      CTUD1 ( CU := ( CU1_m AND (po100msR_m OR po100msF_m)), CD := ( CD1_m AND (po100msR_m OR po100msF_m)), PV := zasobnik_ZO_m,
      R := resetCTUD, QU => QU1_m, QD => QD1_m ); // citac simulujici pohyb akcniho clenu

      // Obsluha pocitani obrobku
      CTUD2 ( CU :=  NOT st1_i6_m, R := pocetAktR_m, PV := pocetVsech_m, R := resetCTUD, CV => pocetAkt_m ); // citac poctu jiz...
                                                                                                            // zpracovanych obrobku

      //POCATECNI STAV
      IF ( st1_o0_m = FALSE AND st1_i2_m = FALSE AND obrobek_m = FALSE AND obrobek1_m = FALSE AND obrobek2_m = FALSE AND QD1_m = TRUE
      AND ovladaniRezimu_m <> 2 ) THEN
                                    st1_i1_m := TRUE; // pocatecni stav cidla - po zapnuti stroje je cidlo st1_i1_m SETovano
                                    obrobek1_m := FALSE; // ovladani viditelnosti obrobku ve vizualizaci
```

```
                                obrobek2_m := FALSE; // ovladani viditelnosti obrobku ve vizualizaci
                                CD1_m := FALSE; // ovlada sestupny vstup citace CTUD1
                                CU1_m := FALSE; // ovlada vzestupny vstup citace CTUD1
                                st1_i6_m := TRUE; // SET cidlo: zasobnik obrobku je prazdny
                                resetCTUD := FALSE; //reset citace CTUD2
END_IF;

// ZALOZENI/ODEBRANI OBROBKU Z POCATECNIHO MISTA
// zalozeni obrobku do zasobniku
IF ( obrobek_m = TRUE AND obrobek1_m = FALSE AND QD1_m  = TRUE AND ovladaniRezimu_m <> 2 ) THEN
                                            st1_i6_m := FALSE;// RESet cidlo: zasobnik obrobku je prazdny
                                            obrobek1_m := TRUE; // ovladani viditelnosti obrobku ve vizualizaci
END_IF;

// odebrani/nepritomnost obrobku v zasobniku
IF ( obrobek_m = FALSE AND QD1_m = TRUE AND ovladaniRezimu_m <> 2 ) THEN
                                            st1_i6_m := TRUE;// SET cidlo: zasobnik obrobku je prazdny
                                            obrobek1_m := FALSE; // RESet ovladani viditelnosti obrobku ve vizualizaci
END_IF;

// OBSLUHA POHYBU ZASOBNIKU
// zasunuti valce zasobniku - vyjeti pritomneho obrobku
IF ( st1_o0_m = TRUE AND obrobek1_m = TRUE AND ovladaniRezimu_m <> 2 ) THEN
                                               st1_i1_m := FALSE; // RESet cidla detekce vysunuti valce zasobniku
                                               CD1_m := FALSE;  // ovlada sestupny vstup citace
                                               CU1_m := TRUE; // ovlada vzestupny vstup citace

   IF ( QU1_m = TRUE AND ovladaniRezimu_m <> 2 ) THEN
                       st1_i2_m := TRUE; // SET cidla detekce zasunuti valce zasobniku
                       CD1_m := FALSE;  // ovlada sestupny vstup citace
                       CU1_m := TRUE; // ovlada vzestupny vstup citace
                       obrobek_m := FALSE; // vlozeni / vyjmuti obrobku z procesu
                       obrobek1_m := FALSE; // ovladani viditelnosti obrobku ve vizualizaci
                       obrobek2_m := TRUE; // ovladani viditelnosti obrobku ve vizualizaci
   END_IF;

END_IF;
```

```
    // zasunuti valce zasobniku - neni pritomen obrobek
    IF ( st1_o0_m = TRUE AND obrobek1_m = FALSE AND ovladaniRezimu_m <> 2 ) THEN
                                                st1_i1_m := FALSE; // RESet cidla detekce vysunuti valce zasobniku
                                                CD1_m := FALSE;  // ovlada sestupny vstup citace
                                                CU1_m := TRUE; // ovlada vzestupny vstup citace
                                                st1_i6_m := FALSE;// RESet cidlo: zasobnik obrobku je prazdny

       IF ( QU1_m = TRUE AND ovladaniRezimu_m <> 2 ) THEN
                           st1_i2_m := TRUE; // SET cidla detekce zasunuti valce zasobniku
                           CD1_m := FALSE;  // ovlada sestupny vstup citace
                           CU1_m := TRUE; // ovlada vzestupny vstup citace
       END_IF;

    END_IF;

    // vysunuti valce zasobniku - pohyb do zakladni pozice
    IF ( st1_o0_m = FALSE AND ovladaniRezimu_m <> 2 ) THEN
                              st1_i2_m := FALSE; // RESet cidla detekce zasunuti valce zasobniku
                              CD1_m := TRUE;  // ovlada sestupny vstup citace
                              CU1_m := FALSE; // ovlada vzestupny vstup citace

       IF ( QD1_m = TRUE AND ovladaniRezimu_m <> 2 ) THEN
                           st1_i1_m := TRUE; // SET cidla detekce vysunuti valce zasobniku
                           CD1_m := FALSE;  // ovlada sestupny vstup citace
                           CU1_m := FALSE; // ovlada vzestupny vstup citace
       END_IF;

    END_IF;

// RESETOVANI a uvedeni do puvodniho stavu
IF ( ovladaniRezimu_m = 3 ) THEN
                                dojel := FALSE; //
                                resetCTUD := TRUE; //
                                st1_i6_m := FALSE; //
                                st1_o0_m := FALSE; //
                                st1_i1_m := FALSE; //
                                st1_i2_m := FALSE; //
```

```
                    obrobek_m := FALSE; //
                    obrobek1_m := FALSE; //
                    obrobek2_m := FALSE; //
                    st1_i6_t_m := FALSE; //
                    CU1_m := FALSE; //
                    CD1_m := FALSE; //
                    pocetVsechR_m := FALSE; //
                    pocetAktR_m := FALSE; //
                    pocetAkt_m := 0; //
                    pocetVsech_m := 0; //
 END_IF;

END_FUNCTION_BLOCK
```

9.3 Hodnocení předvedených výsledků práce od pana Ing. Jana Fuky

Hodnocení diplomové práce

Autor: Bc. Matěj Šiška

Téma diplomové práce: Virtuální řízená soustava diskrétního nebo hybridního typu se systémy PLC a SCADA

Předvedení možností virtuálního modelu bylo na VOŠ, SŠ, COP Sezimovo Ústí autorem předvedeno 8. 4. 2010.

Mohu konstatovat, že práce splňuje mnou očekávané přínosy. Virtuální model distribuční, procesní a manipulační stanice Festo vhodně znázorňuje reálnou sestavu i její činnost. Jsou k dispozici všechny signály senzorů a fungují řídicí zásahy směrem k akčním členům. Systém dovoluje simulovat chod stanice podle zapsaného programu řízení.

Z tohoto důvodu doporučím, aby výsledky diplomové práce mohly být využívány ve výuce automatizace na naší střední škole (především obor Mechatronika) a vyšší odborné škole. Ke splnění tohoto cíle jsme s autorem předběžně dohodli další postup:

- autor podle svých možností provede předvedení výsledků práce na VOŠ, SŠ, COP Sezimovo Ústí s cílem informovat vyučující předmětu automatizace a mechatronika
- autor specifikuje potřebné SW vybavení tak, aby bylo možno simulaci využívat na PC učitele i žáků (studentů)
- autor specifikuje potřebné SW a HW vybavení pracoviště na VOŠ, SŠ, COP Sezimovo Ústí tak, aby bylo možno využívat výsledků diplomové práce k simulaci a řízení reálných stanic Festo

Zpracoval Ing. Jan Fuka, učitel odborných předmětů VOŠ, SŠ, COP Sezimovo Ústí.

V Sezimově Ústí 28. 4. 2010