FAKULTA
ELEKTROTECHNICKÁ
ČVUT V PRAZE

Fakulta elektrotechnická
Katedra radioelektroniky

Bakalářská práce

# Návrh laboratorních úloh v prostředí TIA Portal

Jakub Špicar

Praha, Květen 2023

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Špicar** Jméno: **Jakub** Osobní číslo: **492101**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Elektronika a komunikace**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Návrh laboratorních úloh v prostředí TIA Portal**

Název bakalářské práce anglicky:

**Design of Laboratory Tasks in TIA Portal**

Pokyny pro vypracování:

Cílem práce je návrh laboratorních úloh, které budou sloužit k výuce programování PLC kontrolerů v prostředí TIA portal. Součástí práce bude seznámení s technologií PLC, funkčními bloky a používanými programovacími jazyky. Hlavní částí práce budou zadání laboratorních úloh, ve kterých bude možné simulovat dílčí částí vzorové výrobní linky. Ke každé úloze bude k dispozici zadání, vzorový kód a schéma.

Seznam doporučené literatury:

[1] SIEMENS, Programming Guideline for S7-1200/1500, 2014
[2] HOOPER, Jay. Introduction to PLCs. 2. Carolina Academic Press. ISBN 978-1594603310.
[3] TUBBS, Stephen Philip. Programmable Logic Controller (PLC) Tutorial, Siemens Simatic S7-1200. 2016. ISBN 978-0981975368.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

**doc. Ing. Stanislav Vítek, Ph.D. katedra radioelektroniky FEL**

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

Datum zadání bakalářské práce: **13.02.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

| doc. Ing. Stanislav Vítek, Ph.D.<br>podpis vedoucí(ho) práce | doc. Ing. Stanislav Vítek, Ph.D.<br>podpis vedoucí(ho) ústavu/katedry | prof. Mgr. Petr Páta, Ph.D.<br>podpis děkana(ky) |
|---|---|---|

## III. PŘEVZETÍ ZADÁNÍ

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

. | Datum převzetí zadání

Podpis studenta

# Čestné prohlášení

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

V Praze, 25.05.2023

# Poděkování

Rád bych poděkoval doc. Ing. Stanislavovi Vítkovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

# Abstrakt

Tato bakalářská práce se zabývá navrhem a implementací virtuální sladovny pivovaru pro potřeby laboratorních úloh zabývajících se uvedením do problematiky programování programovatelných logických automatů. V teoretické část obsahuje bakalářská práce popis chování procesorů programovatelných logických automatů a uvádí základní metody programování a práce s daty. Praktická část práce se věnuje vlastnímu návrhu virtuální sladovny v prostředí TIA Portal. Virtuální sladovna pak slouží jako prostředí pro testování a aplikaci teoretických znalostí studenta.

**Klíčová slova:** Siemens, Programovatelný logický automat, TIA portal

# Abstract

This bachelor's thesis deals with the design and implementation of a virtual brewery malt house for the purposes of laboratory exercises focusing on introducing programmable logic controller programming. The theoretical part of the thesis provides a description of the behavior of programmable logic controller processors and introduces basic methods of programming and working with data. The practical part of the thesis is dedicated to the actual design of the virtual malt house in the TIA Portal environment. The virtual malt house serves as a testing environment and application platform for the student's theoretical knowledge.

**Keywords:** Siemens, Programmable logic controller, TIA portal

# Seznam zkratek

**ALU** ... Aritmeticko-logická jednotka
**CPU** ... Procesor
**DB** ... Datový blok
**FB** ... Funkční blok
**FBD** ... Diagram funkčních bloků
**FC** ... Funkce
**HMI** ... Uživatelské rozhraní
**LAD** ... Příčkový diagram
**OB** ... Organizační blok
**PLC** ... Programovatelný logický automat
**SCL** ... Strukturovaný jazyk řízení
**STL** ... Seznam prohlášení

# Obsah

# Úvod

Programovatelný logický automat (PLC) je důležitým prvkem v automatizačních systémech. Skládá se z procesoru (CPU) a periferií, přičemž existují dva hlavní typy PLC systémů - kompaktní a modulární. Procesor je považován za hlavní modul řídicího systému PLC a je nezbytný v každém automatizačním systému.

Procesory PLC systémů pracují v cyklicky orientovaném chodu, který se skládá ze čtyř fází. Doba cyklu se obvykle pohybuje mezi 1 až 10 milisekundami, ale při použití výkonných procesorů nebo krátkých programů se doba jednoho cyklu může snížit až do řádu mikrosekund.

V PLC systémech se k programování využívají různé programové bloky. Organizační bloky (OB) slouží jako rozhraní mezi operačním systémem a uživatelským programem a jsou volány v závislosti na událostech, jako jsou časové události, cyklické události nebo chybové stavy. Další programové bloky zahrnují funkce (FC), funkční bloky (FB) a datové bloky (DB). Funkce se využívají pro popis a řízení jednodušších procesů, například matematických výpočtů nebo řízení otevírání a zavírání ventilů. Funkční bloky mají svou vlastní lokální paměť a mohou být využity pro komplexnější procesy, nebo procesy u kterých je potřeba uchovávat data po delší dobu, než je doba jednoho programového cyklu. Datové bloky slouží k ukládání proměnných a umožňují strukturování dat v PLC procesoru.

Programování PLC systémů je prováděno pomocí různých programovacích jazyků, přičemž syntaxi a sémantiku programovacích jazyků sjednocuje mezinárodní norma IEC 61131-3. Norma definuje 5 programovacích jazyků, včetně příčkového diagramu, diagramu funkčních bloků, strukturovaného textu, sekvenčního funkčního diagramu a seznamu instrukcí. V rámci normy IEC 61131-3 jsou také definovány různé datové typy, které slouží k popisu a manipulaci s proměnnými v programování PLC. Mezi běžné datové typy patří celočíselné typy (například INT nebo DINT), desetinné typy (například REAL nebo LREAL), logický typ (BOOL), řetězcový typ (STRING) a další.

Programování PLC systémů se obvykle provádí pomocí specializovaných vývojových prostředí, která poskytují nástroje pro vytváření, ladění a testování PLC programů. Tato prostředí umožňují vytváření grafických blokových diagramů, psaní strukturovaného textového kódu a případně i kombinaci různých programovacích jazyků. Kromě samotného programování umožňují tyto prostředí provádět rozsáhlé diagnostické testy, sledovat stav vstupů a výstupů, monitorovat chod PLC systému, spravovat komunikaci s ostatními zařízeními a mnoho dalšího. Tímto způsobem je umožněna komplexní a efektivní práce s PLC systémy v průmyslové automatizaci.

PLC systémy vykazují široké uplatnění v průmyslové automatizaci pro řízení a sledování rozličných zařízení a procesů. Jejich využití je rozšířeno v různých odvětvích, včetně výroby, energetiky, dopravy, zpracování surovin a dalších. Díky své spolehlivosti, odolnosti vůči rušení a schopnosti operovat v náročných provozních podmínkách se PLC systémy stávají preferovanou volbou pro automatizaci průmyslových aplikací.

# I Teoretická část

## 1 Procesor programovatelného logického automatu

Programovatelný logický automat (PLC) se skládá z procesoru (CPU) a periferií, přičemž rozlišujeme dva typy PLC systémů. Pokud jsou všechny periferie (např. vstupní a výstupní moduly) integrovány ve stejném zařízení, ve kterém se nachází procesor, pak mluvíme o kompaktním PLC. Pokud jsou periferie realizovány jako samostatné moduly, které se k modulu procesoru připojují pomocí sběrnice, pak mluvíme o modulárním PLC. V obou případech je procesor považován za hlavní modul řídicího systému PLC, který musí být v každém automatizačním systému.


Obrázek 1.1: Modulární PLC systém SIMATIC S7-300 [1]

### 1.1 Architektura procesoru

Základem celého PLC systému je procesor - *central processing unit* (CPU). Jeho úkolem je vykonávat kód programu a zpracovávat vstupní signály, aby mohl ovládat výstupní periferie. Správnou funkci procesoru zajišťuje řadič, aritmeticko-logická jednotka (ALU), registry, paměť, vstupní a výstupní rozhraní [2].

Řadič slouží k řízení celého procesoru. Jeho hlavním úkolem je interpretovat a provádět instrukce uložené v paměti a koordinovat činnost ostatních komponentů procesoru.

ALU se stará o aritmetické a logické operace nad daty. Je schopná provádět sčítání, odčítání, násobení, dělení, porovnávání, logické operace a další. Zpracování dat je řízeno instrukcemi, které jsou předány z řadiče a výsledky operací jsou poté uloženy do paměti procesoru nebo registru.

Registry jsou malé paměťové jednotky, které slouží k rychlému ukládání a přístupu k datům v procesoru PLC. Mají obvykle omezenou kapacitu, ale jsou velmi rychlé a umožňují efektivní manipulaci s daty. Existují různé typy registrů, jako jsou obecné registry pro

ukládání běžných dat, registry pro adresování, kam se ukládají paměťové adresy nebo registry pro ukládání výsledků operací ALU.

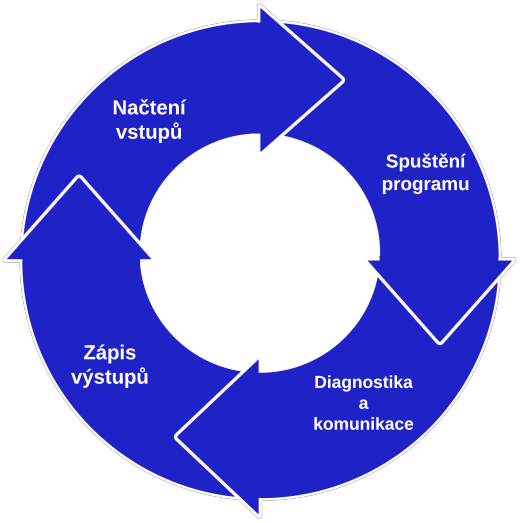
Vstupní a výstupní rozhraní (I/O) propojuje procesor s vnějšími zařízeními a umožňuje přenos dat mezi PLC a okolním světem. I/O rozhraní zajišťuje komunikaci se senzory, motory, HMI panely nebo jinými PLC.

## 1.2 Chod procesoru

Procesory PLC systémů pracují v tzv. *cyklicky orientovaném chodu* [3]. Ten se skládá ze smyčky, která obsahuje čtyři fáze:

1. Načtení vstupů
2. Spuštění programu
3. Diagnostika a komunikace
4. Zápis výstupů

Typicky se doba cyklu pohybuje mezi 1 až 10 milisekundami [3]. V případě výkonných procesorů, nebo při použití krátkých programů se doba jednoho cyklu může snížit až do řádu mikrosekund.




Obrázek 1.2: Cyklicky orientovaný chod CPU

### 1.2.1 Načtení vstupů

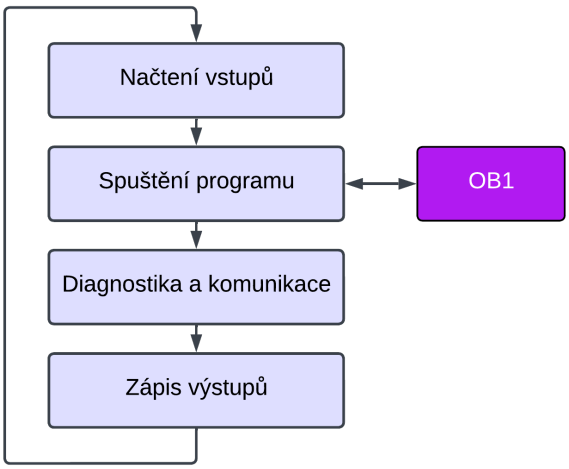
V první fázi vytvoří procesor obraz všech adres vstupních periferií [4]. V případě digitálních vstupů logické jedničky a nuly. V případě analogových vstupů hodnoty napětí nebo proudu. Tento obraz je pak uložen do paměti procesoru pro další zpracování uživatelským programem.

Obraz se vytváří z toho důvodu, aby během zbylých fází cyklu nevznikl hazardní stav, kdy by se změnila hodnota adresy vstupní periferie během cyklu. Takto se případná změna projeví až v cyklu dalším [4].

### 1.2.2 Spuštění programu

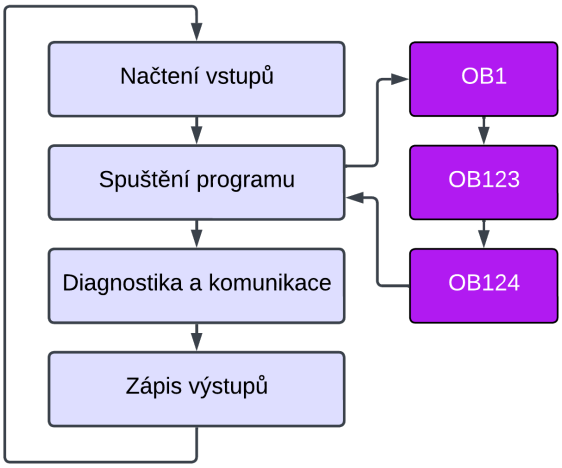
V druhé fázi načtená data zpracovává uživatelský program. Mezi tři hlavní přístupy k tvorbě programu patří *lineární program, program s blokovou strukturou a strukturovaný program* [5].

V případě lineárního programování se celý program nachází v jednom souvislém bloku, a to v programovém bloku OB1. Tato metoda je vhodná pro malé projekty. Tím, že je uživatelský program uložen pouze v jednom organizačním bloku, může docházet ke ztrátě přehlednosti s rostoucí velikostí programu.




Obrázek 1.3: Schéma lineárního přístupu k programování

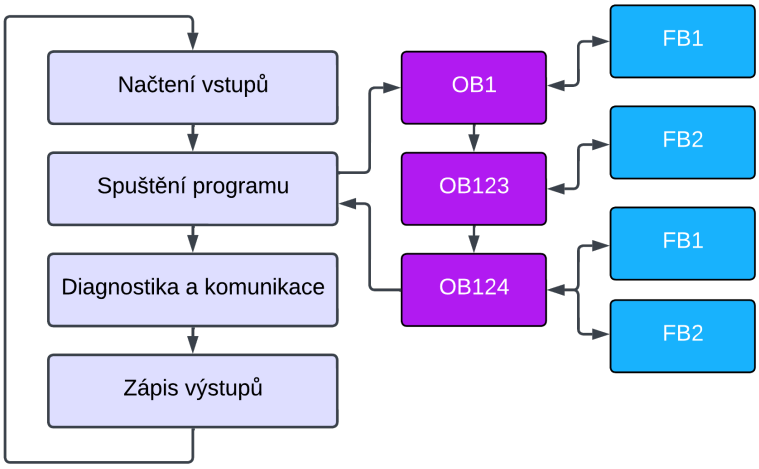
Program s blokovou strukturou je rozdělen do více programových bloků, přičemž každý programový blok obsahuje pouze program pro řešení specifické úlohy. V takovém případě se používají cyklicky volané organizační bloky, které operační systém PLC volá sám, bez nutnosti volání bloku v hlavní sekci programu.




Obrázek 1.4: Schéma programu s blokovou strukturou

Strukturovaný program obsahuje bloky s přiřaditelnými parametry, které jsou naprogramovány tak, aby bylo bloky možné používat univerzálně. Při volání takového bloku

mu jsou zároveň předány hodnoty parametrů např. v podobě adresy vstupního nebo výstupního zařízení. V tomto případě je nutné definovat v kódu programu kdy a s jakými parametry se mají tyto bloky volat.




Obrázek 1.5: Schéma strukturovaného programu

### 1.2.3 Diagnostika a komunikace

Ve třetí fázi se provádějí vnitřní funkce, mezi které patří diagnostika a komunikace. Operační systém procesoru je vybaven diagnostickou sadou nástrojů, která slouží k včasnému odhalení událostí ohrožující funkčnost celého automatizačního systému [6]. Pomocí těchto nástrojů je PLC schopno varovat obsluhu v případě neočekávaného odpojení od napájecího napětí nebo detekovat chyby připojených periferií, a to jak vlastních modulů, tak i zařízení, které jsou k těmto modulům připojeny.

### 1.2.4 Zápis výstupů

Jakmile uživatelský program dokončí zpracování obrazu hodnot vstupních periferií a diagnostika PLC nezjistí žádné hrozby ohrožující další chod automatizačního systému, pak se hodnoty, které jsou v obrazu hodnot výstupních periferií, zapíší na reálné adresy výstupních modulů.

# 2 Organizační bloky

Organizační bloky (OB) jsou programovými bloky, které tvoří rozhraní mezi operačním systémem a uživatelským programem. Organizační bloky jsou operačním systémem označovány jako bloky řízené událostmi [6]. Události mohou být cyklické, časově závislé, hardwarově závislé, přechod RUN-STOP, nebo chybové. Pokud některá z událostí nastane, je zavolán příslušný organizační blok.

## 2.1 Systém priorit organizačních bloků

Každý organizační blok má své číslo, které mu je přiděleno buď automaticky programovacím prostředím, nebo manuálně uživatelem. Toto číslo pak znázorňuje prioritu organi-

začního bloků, která je důležitá při volání jednotlivých organizačních bloků [5].

Například organizační blok s označením OB1, který má nejvyšší prioritu, je volán cyklicky a v případě lineárního programu je v něm uložen celý uživatelský program. Z tohoto důvodu je tento organizační blok často považován za hlavní část programu, stejně jako je v paradigmatu procedurálního a objektově orientovaného programování považovaná funkce main() za hlavní část programu. Existuje mnoho různých organizačních bloků, které se volají podle různých podmínek a priorit. Existují ale 4 hlavní skupiny, do kterých se organizační bloky rozdělují [5].

## 2.2 Spouštěcí organizační bloky

První skupinou jsou organizační bloky, které se volají bezprostředně po spuštění PLC. Tyto organizační bloky se volají po spuštění PLC, a to plánovaném i neplánovaném. Volají se jako první programové bloky ještě před hlavním, cyklicky volaným organizačním blokem. Pomocí těchto organizačních bloků se běžně provádí vnitřní nastavení komunikačních spojení nebo inicializace proměnných.

V PLC SIMATIC od společnosti Siemens jsou tyto programy uloženy v organizačním bloku s označením OB100. Pokud by chtěl uživatel přidat další organizační bloky, které by se volaly při spouštění PLC, pak by musely mít číslo, které je větší nebo rovno 123 [5].

## 2.3 Cyklické organizační bloky

Druhou skupinou jsou cyklicky volané organizační bloky. Programy, které jsou uloženy v takovém organizačním bloku, se provádí v nepřetržité smyčce - cyklu. Při tomto cyklickém chodu programu je reakční doba výsledkem doby běhu operačního systému procesoru a součtu dob chodu všech prováděných instrukcí. Reakční doba, tj. jak rychle lze změnit hodnotu adresy výstupní periferie v závislosti na změně hodnoty adresy vstupní periferie, je minimálně stejná jako doba jednoho cyklu. Maximální reakční doba je dvojnásobek doby cyklu.

Siemens udává pro PLC SIMATIC jeden hlavní cyklicky volaný organizační blok, který má označení OB1. Pokud by chtěl uživatel přidat další organizační bloky tohoto typu, pak musí být jejich číslo větší nebo rovno 123 [5]. Jakmile se v projektu nachází více než jeden cyklicky volaný organizační blok, pak se volají postupně, tj. nejdříve se spustí OB1, pak OB123, OB124, atd.

## 2.4 Periodické organizační bloky

Další skupinou jsou periodicky volané organizační bloky. Tyto organizační bloky lze rozdělit do dvou skupiny podle charakteru události, kterou jsou řízeny - přerušení v denní dobu a přerušení v intervalech [5].

V PLC SIMATIC se v rozmezí organizačních bloků OB10 až OB17 nachází přerušení v denní dobu. Tyto organizační bloky umožňují spouštět programy v nich uložené v předem specifikovaný denní čas. Toho se dá využít například při nastavování zálohování. Program pro zálohování totiž může být uložen v organizačním bloku OB10, ve kterém bude specifikované, že tento organizační blok se zavolá každý den v 20:00.

Druhou skupinou periodicky volaných organizačních bloků tvoří přerušení v intervalech, které se nachází se v rozmezí organizačních bloků OB30 až OB38. Organizační blok

tohoto typu operační systém procesoru volá v předem specifikovaných intervalech. Toho lze využít v kontrolních programech, které se v tomto organizačním bloku mohou spouštět například každých 5 minut.

## 2.5 Organizační bloky řízené událostí

Poslední skupinou organizačních bloků jsou takové organizační bloky, které jsou řízené událostmi. Tyto události jsou rozděleny do čtyř skupin - přerušení se zpožděním, přerušení hardwarem, asynchronní chyby a synchronní chyby [5].

Organizační bloky v rozsahu OB20 až OB23 jsou přerušení se zpožděním. Po uplynutí uživatelsky definovatelné doby přeruší chod programu uloženého v jakémkoli cyklicky volaném organizačním bloku.

Organizační bloky typu hardwarové přerušení v rozsahu organizačních bloků OB40 až OB47 přeruší cyklicky volaný organizační blok v reakci na signál z hardwarové události.

Organizační bloky typu asynchronní chyba (OB121 až OB122) se volají v případech, kdy dojde k chybě v programování, nebo k chybnému přístupu v adresaci.

Organizační bloky typu synchronní chyby s rozsahem OB80 až OB87 jsou pak vyvolány pokud dojde k selhání hardwaru, rozvaděče, napájení nebo komunikace.

Tabulka 2.1: Přehled typů organizačních bloků

| **Skupina** | **Podskupina** | **Rozsah** |
|---|---|---|
| Cyklické | (nemá) | OB1 |
| Spouštěcí | | OB100 - OB102 |
| Periodické | Přerušení v denní dobu | OB10 - 0B17 |
| | Přerušení v intervalech | OB30 - OB38 |
| Řízené událostí | Přerušení se zpožděním | OB20 - OB23 |
| | Přerušení hardwarem | OB40 - OB47 |
| | Asynchronní chyby | OB80 - OB87 |
| | Synchronní chyby | OB121 - OB122 |

# 3 Funkce, funkční bloky, datové bloky

Mimo organizačních bloků existují další programové bloky, a to funkce (FC), funkční bloky (FB) a datové bloky (DB) [4]. Stejně jako v případě organizačních bloků má každý z těchto programových bloků své číslo. Nicméně vzhledem k tomu, že funkce ani funkční bloky nejsou volány automaticky operačním systémem procesoru, ale programátor musí definovat volání v kódu programu organizačního bloku, tak toto číslo slouží pouze jako identifikátor bloku a neurčuje pořadí v jakém pořadí jsou tyto programové bloky volané. To stejné platí i pro datové bloky s tím rozdílem, že v datovém bloku není uložen program, ale jedná se o strukturované místo v paměti.

## 3.1 Popis funkcí a jejich využití

Kromě organizačních bloků lze program ukládat do bloku funkce (FC). Na rozdíl od organizačních bloků nemá tento typ programových bloků dedikovanou paměť. Z toho důvodu je možné používat pouze dočasné nebo globální proměnné. Funkce se využívají při strukturovaném programování na popis a řízení jednodušších procesů. Například matematický výpočet, řízení otevírání a zavírání regulačního ventilu, řízení startérů. Veškeré parametry musí být funkci přiřazeny společně s voláním funkce.

## 3.2 Popis funkčních bloků a jejich využití

Na rozdíl od funkce má funkční blok (FB) alokovanou svou lokální paměť, která musí být inicializována při svém volaní [4]. Jeden segment dat lokální paměti pro funkční blok nazýváme *instance*, která je vytvářena automaticky operačním systémem při kompilaci a stažení programu do PLC.

Existují dva různé druhy instancí: *single-instance, multi-instance* [6]. Jednodušší metoda vyžití single-instance znamená, že volaný funkční blok ukládá svá data do svého instančního datového bloku. Multi-instance znamená, že volané funkční bloky mohou ukládat svá data do datového bloku instance volajícího funkčního bloku, tj. pokud je ve funkčním bloku volán jiný funkční blok, pak volaný funkční blok uloží svá data do instančního datového bloku nadřazeného funkčního bloku.

## 3.3 Popis datových bloků a jejich využití

Datové bloky (DB) představují mimo obecné datové paměti další způsob jak ukládat proměnné. Na rozdíl od obecné datové paměti není nutné definovat se symbolickým jménem proměnné také její adresu. Při vytvoření proměnné, tj. pojmenování a určení datového typu, vytvoří a alokuje operační systém adresu v paměti sám, automaticky. Další výhodou oproti obecné datové paměti je možnost strukturovatelnosti dat. Uvnitř datových bloků je totiž možné vytvářet struktury a pole dat [5].

Datové bloky může vytvářet vývojové prostředí, nebo uživatel. Pokud uvnitř organizačního nebo funkčního bloku chceme zavolat další funkční blok, pak vývojové prostředí může vytvořit datový blok pro danou instanci funkčního bloku samo. Nicméně je možné i specifikovat již existující datový blok pro instanci funkčního bloku manuálně. Mimo instance funkčních bloků je vhodné používat datové bloky i pro ukládání interních dat, pro které je výhodné uložení do struktur nebo polí.

# 4 Programování PLC systémů

## 4.1 Jazyky programování PLC

Za úspěchem prvního PLC Modicon 084, při uvedení na trh v roce 1969, stálo především rozhodnutí použít pro programování programovací jazyk příčkového diagramu, který graficky interpretuje reléovou logiku [7]. Prvním důvodem úspěchu Modiconu 084 a příčkového diagramu bylo to, že grafické programování bylo mnohem rychlejší a snadnější než předešlé řešení, kterým bylo manuální propojování reléových řídicích systémů. Druhým důvodem byla uživatelská přívětivost oproti konkurenci, která k programování používala

booleovské příkazy. Inženýři, kteří automatizované systémy navrhovali, byly totiž zvyklí z reléových řídicích systémů na reléovou logiku, která byla implementována do Modiconu 084 [8].

S technologickým rozvojem automatizačních systémů a příchodem nových výrobců PLC se začaly vyvíjet i programovací jazyky určené pro programování PLC. Nejdříve bylo běžným zvykem, aby každý výrobce používal pro programování svých PLC proprietární programovací jazyk. To se změnilo až v roce 1993 s příchodem IEC 61131-3 [9].

Mezinárodní norma IEC 61131-3 specifikuje 5 programovacích jazyků - příčkový diagram LD, diagram funkčních bloků FBD, strukturovaný text ST, sekvenční funkční diagram SFC a seznam instrukcí IL [7]. Zavedením normy tak byla sjednocena syntaxe a sémantika jazyků pro programování PLC.

### 4.1.1 Příčkový diagram - LAD

Příčkový diagram, v některé literatuře také označovaný jako kontaktní schéma [10], je grafický programovací jazyk interpretující reléovou logiku. Jeho označení LD, případně LAD, vychází z anglického označení *Ladder Diagram*.

Program je zapsán ve schématu, který tvoří dvě svislé příčky označované jako napájecí vedení (*power line*) nebo napájecí sběrnice (*power bus*). Mezi tato vedení programátor vkládá instrukční bloky na tzv. příčky. Každá příčka může být rozvětvena do několika dalších příček a rozvětvená příčka může být spojena s příčkou nad ní. Vkládané instrukční bloky mohou být jednoduché logické instrukce, například spínací a rozpínací kontakty, které přeruší (obnoví) vedení v příčce pokud má tag hodnotu 1 (0). Pokročilejšími instrukčními bloky mohou být instrukce pracující s tagy, které mají velikost několika bitů, například *word* nebo *integer*. Mezi tyto instrukční bloky patří instrukce *Add* a *Sub* které sečtou, případně odečtou hodnoty libovolného počtu tagů. Další instrukční bloky mohou vyžadovat pro správnou funkci vlastní instanci datového bloku. To je nutné v případě použití časovačů. Kromě instrukčních bloků definovaných výrobcem lze používat i vlastní bloky v podobě funkcí, nebo funkčních bloků.

### 4.1.2 Diagram funkčních bloků - FBD

Kromě příčkového diagramu je možné programovat PLC dalším grafickým jazykem, kterým je diagram funkčních bloků. Jeho označení FBD opět vychází z anglického pojmenování *Function Block Diagram*.

Program je v tomto případě interpretován samostatnými grafickými instrukčními bloky, připomínající volání funkčních bloků [10]. Instrukční bloky jsou navzájem propojeny bez jakýchkoli dalších prvků jakými byly v příčkovém diagramu napájecí vedení.

Možnosti instrukčních bloků, které je možné použít jsou stejné jako v případě příčkového diagramu. Rozdíl mohou tvořit některé základní logické operace. Zatímco v příčkovém diagramu bylo možné vytvořit logickou funkci typu AND nebo OR jen pomocí specifického uspořádání instrukčních bloků do příček, tak v diagramu funkčních bloků jsou i pro tyto instrukce vytvořeny vlastní instrukční bloky.

### 4.1.3 Seznam instrukcí - IL a Seznam prohlášení - STL

Prvním z textových programovacích jazyků je seznam instrukcí (*Instruction List*). Jedná se o nízkoúrovňový programovací jazyk podobající se jazyku symbolických adres [11].

U tohoto jazyku je časté, že výrobci PLC přidávají další funkce nad rámec normy IEC 61131-3. Například v automatizačních systémech Siemens je tento jazyk označen jako STL (*Statement List*) [6] a je optimalizovaný pro procesory SIMATIC. Díky nízkoúrovňovosti a optimalizaci jazyka přímo pro SIMATIC procesory se tak jedná o nejrychlejší programovací jazyk určený pro SIMATIC PLC [6].

### 4.1.4 Strukturovaný text - ST a strukturovaný jazyk řízení SCL

Strukturovaný text je vyšší programovací jazyk, který je založený na syntaxi jazyka Pascal [12]. Stejně jako v případě seznamu instrukcí tento jazyk výrobci doplňují o další funkcionality. V systémech Siemens se tak programuje v strukturovaném jazyku řízení (SCL) [6].

## 4.2 Proměnné a práce s daty

Data jsou základním stavebním kamenem výpočetní techniky a mají klíčový význam pro správnou funkci automatizačních systémů. Jsou uloženy v paměti PLC procesoru a slouží k tomu, aby PLC mohlo správně vykonávat svou činnost.

Data v PLC jsou různorodá a zahrnují celou škálu informací o automatizovaném procesu. Může se jednat o jednotlivé bity, které představují logické hodnoty nula a jedna, nebo o větší datové struktury jako Byty, wordy, double wordy a další. Podle povahy a využití dat můžeme data kategorizovat na procesní, interní a externí data [13].

Procesní data představují informace získané ze sledovaného procesu. Tato data zahrnují hodnoty adres vstupních periferií, jako jsou snímače, tlačítka, přepínače, či jiné ovládací prvky.

Interní data jsou vytvářena a používána uvnitř programu během zpracovávání algoritmů řízení. Mezi interní data patří stavová data, která slouží převážně k uchování konkrétního stavu programu, procesu, nebo připojeného zařízení. Dále do interních dat patří data výpočtů, která obsahují mezivýsledky a výsledky matematických a logických operací. Dalšími interními daty jsou data receptur, které definují předpisy pro konkrétní výrobní výrobní nebo zpracovatelské procesy. Archivní data jsou využívána pro ukládání záznamů z procesů a statistiku. Dalším typem interních dat jsou data vizualizační. Tato data jsou určena pro snadnou vizualizaci, ať už uživatelským rozhraním HMI, nebo SCADA systémy.

Externí data slouží k výměně informací mezi různými automatizačními systémy a umožňují integraci PLC s dalšími zařízeními nebo řídicími systémy.

### 4.2.1 IEC 61131-3 a datové typy

V rámci normy IEC 61131-3 jsou definovány různé datové typy, které slouží k reprezentaci a manipulaci s daty v PLC. Mezi nejběžnější datové typy patří bit, který reprezentuje binární stav zapnuto / vypnuto. Dále existují datové typy jako Byte, word, double word. Tato norma také poskytuje podporu pro číselné datové typy jako *integer*, *float* nebo *decimal*, které slouží k reprezentaci celých nebo desetinných čísel.

Kromě číselných datových typů norma IEC 61131-3 definuje také datové typy pro textové řetězce *string*, logické hodnoty *bool* a časová data *time*, *date*. Celkový přehled základních datových typů podporovaným ekosystémem TIA Portal je v tabulce níže.

Tabulka 4.1: Přehled datových typů [14]

| **Datový typ** | **Velikost v bitech** | **Popis** | **Rozsah** |
|---|---|---|---|
| BOOL (Bit) | 1 | Boolean | TRUE/FALSE |
| BYTE (Byte) | 8 | Hexadecimální číslo | B#16#0 až B#16#FF |
| WORD (Word) | 16 | Binární číslo | 2#0 až 2#1111_1111_1111_1111 |
| | | Hexadecimální číslo | W#16#0 až W#16#FFFF |
| | | BCD | C#0 až C#999 |
| | | Decimální číslo (bez znaménka) | B#(0,0) až B#(255,255) |
| DWORD (Double word) | 32 | Binární číslo | 2#0 až 2#1111_1111_1111_1111_ 1111_1111_1111_1111 |
| | | Hexadecimální číslo | W#16#0000_0000 až W#16#FFFF_FFFF |
| | | Decimální číslo (bez znaménka) | B#(0,0,0,0) až B#(255,255,255,255) |
| INT (Integer) | 16 | Decimální číslo (se znaménkem) | -32768 až 32767 |
| DINT (Double integer) | 32 | Decimální číslo (se znaménkem) | L#-2147483648 až L#2147483647 |
| REAL (Floating-point number) | 32 | IEEE Číslo s plovoucí desetinnou čárkou | ±3.402824e+38 až ±1.175495e-38 |
| S5TIME (SIMATIC Time) | 16 | S7 čas s kroky po 10ms | T#0S_10MS až T#9S_999MS |
| TIME (IEC time) | 32 | IEC čas s kroky po 1ms | T#24D_20H_31M_23S_648MS až T#24D_20H_31M_23S_647MS |
| DATE (IEC date) | 16 | IEC datum s kroky po 1 dni | D#1990-1-1 až D#2168-12-31 |
| TIME_OF_DAY (Time) | 32 | Čas s kroky po 1ms | TOD#0:0:0.0 až TOD#23:59:59.999 |
| CHAR (Character) | 8 | ASCII charakter | 0 až 127 |

### 4.2.2 Ukládání dat

V paměti procesoru jsou uloženy nejen proměnné (tagy), ale i celá hardwarová konfigurace , tj. jaké moduly a periferie jsou k procesoru připojeny a uživatelský program. Je tedy výhodné, aby byla paměť rozdělena do dílčích oblastí. Paměť procesoru je rozdělena do třech hlavních oblastí - načítací paměť, pracovní paměť a zachovávací paměť [4].

Načítací a zachovávací paměť jsou tzv. nevolantní paměti [15]. Tento typ je na rozdíl od volantních typů pamětí nezávislý na napájecím napětí. To znamená, že jejich obsah není po ztrátě napájení ztracen, či poškozen. Pro volantní typy paměti totiž platí, že se po vypnutí napájecího napětí jejich obsah ztratí nebo poškodí v řádech jednotek až desítek milisekund po ztrátě napájení.

V načítací paměti jsou ukládány uživatelské programy, data a konfigurace. Jakmile je program stažen do procesoru PLC, tak je nejprve uložen do této oblasti, kde je uchováván pro případné vypnutí a opětovné zapnutí PLC. V případě potřeby většího úložiště, je možné tuto oblast rozšířit pomocí instalace paměťové karty, jejímž slotem disponuje většina procesorů z rodin SIMATIC S7-1500, S7-1200, S7-400 a S7-300 [6].

Pracovní paměť je dočasná paměť, ve které jsou ukládána data potřebná pro správný běh programu. Kdykoli je volaný programový blok, tak procesor alokuje v této oblasti místo pro dočasná data. Svým chováním tak tato oblast připomíná paměť RAM v běžném osobním počítači. Jakmile se procesor odpojí od napájecího napětí, veškerá data uchovávaná v této paměti ztrácí.

Zachovávací paměť je určená pro omezené množství hodnot pracovní paměti. Oblast zachovávací paměti slouží k ukládání hodnot vybraných paměťových adres. Běžně se do této paměti ukládají důležitá interní data, u kterých je potřeba, aby byla uložena jejich hodnota pro případ výpadku napájecího napětí.

### 4.2.3 Adresace paměťových oblastí

Většina vývojových prostředí, mezi které patří i SIMATIC STEP 7 [6] umožňují tzv. symbolické programování. To znamená, že je možné vytvářet symbolická jména pro adresy dat. A to ať už se jedná o adresaci vstupních a výstupních periferií, nebo lokálních proměnných používaných v rámci uživatelského programu.

Z hlediska adresace rozlišujeme 3 základní typy proměnných - adresy vstupních zařízení (I), adresy výstupních zařízení (Q) a adresy bitové paměti (M) [4]. Každá proměnná musí mít jednoznačně definovanou a unikátní adresu s uvedením příslušného typu I, Q nebo M.

# II Praktická část

## 5 TIA Portal

Celý projekt byl vytvořen v softwaru Totally Integrated Automation Portal - zkráceně TIA Portal. Jedná se o inženýrský ekosystém pro automatizaci vyvíjený společností Siemens. Hlavními funkcemi tohoto prostředí je konfigurace, programování, diagnostika a údržba PLC systémů, HMI panelů a pohonů.

TIA Portal uvedl Siemens na trh v roce 1996 [16] jako reakci na složitost automatizovaných systémů a potřebu sjednocení softwarových nástrojů pro správu těchto systému. Před příchodem TIA Portalu byly nástroje pro automatizaci řízení rozděleny do několika samostatných programů:

- SIMATIC STEP 7 - pro programování PLC
- SIMATIC STEP 7 Safety - pro programování fail-safe PLC
- WinCC - pro konfiguraci vizualizací procesů
- SINAMICS Startdrive - pro parametrizace pohonných systémů

Takto nejednotný přístup měl řadu nevýhod, mezi které patří náročnost na výkon počítače na kterém jsou programy spuštěny, složitost nastavení komunikace mezi jednotlivými programy a omezené sdílení dat.

### 5.1 Využití TIA Portalu v projektu

Pro účely seznámení s problematikou návrhu automatizovaných systémů stačí funkcionalita, kterou nabízí sjednocení programů SIMATIC STEP 7 a WinCC. Mimo tyto základní funkce TIA Portalu byl v projektu použit software SIMATIC S7-PLCSIM Advanced pro simulaci PLC procesoru.

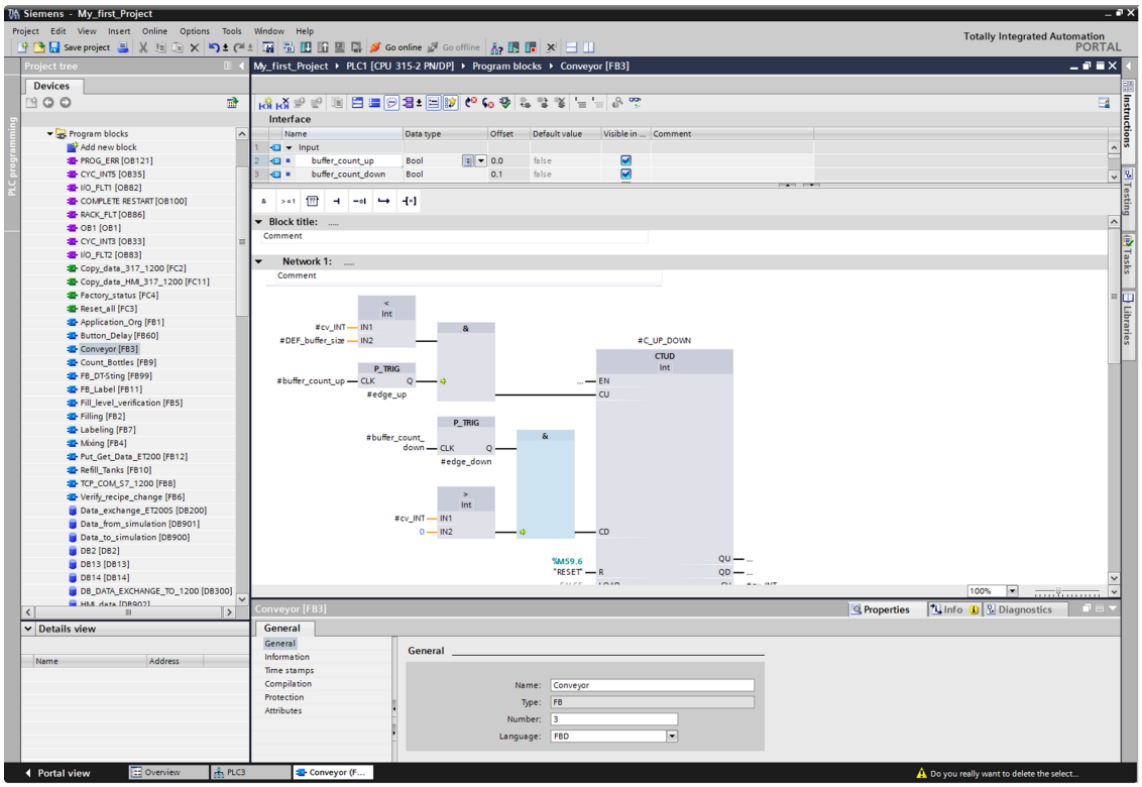
### 5.2 SIMATIC STEP 7

SIMATIC STEP 7 je vývojové prostředí pro programování a konfiguraci PLC společnosti Siemens. Jeho podpora je omezená pouze na produkty společnosti Siemens, mezi které patří zejména procesory z rodiny SIMATIC S7-1500, S7-1200, S7-400 a S7-300 [6].

V rámci tohoto prostředí je možné vytvářet programy pro PLC pomocí většiny standardně používaných jazyků, tj. kontaktní zobrazení LAD, diagram funkčních bloků FBD, seznam prohlášení STL, strukturovaný jazyk řízení SCL. To poskytuje programátorům flexibilitu při vývoji. Navíc STEP 7 umožňuje přepínat zobrazení programu i po vytvoření programového bloku. Tato funkce je dostupná pouze pro přepínání mezi jednotlivými grafickými programovacími jazyky, jako je kontaktní zobrazení LAD a diagram funkčních bloků FBD a pro přepínání mezi jednotlivými programovacími jazyky, které jsou textové, jako je seznam prohlášení STL a strukturovaný jazyk řízení SCL. To je výhodné například v takovém případě, kdy programátorovi, který daný systém udržuje nevyhovuje programovací jazyk, který zvolil programátor, který systém navrhoval.

Klíčovou funkcí STEP 7 je jeho možnost online spojení s PLC. To umožňuje programátorovi sledovat v reálném čase program, který je spuštěný na PLC procesoru, diagnostikovat problémy s hardwarem a aktualizovat firmware hardwaru. Taková funkce zkracuje čas potřebný pro ladění programu a umožňuje rychlejší reakci na změny nebo problémy v řídicím systému.

Kromě trvalého online spojení s PLC, pro účely sledování programu v reálném čase, je možné i krátkodobé spojení za účelem nahrání nového nebo aktualizovaného programu nebo stažení programu z PLC do STEP 7.




Obrázek 5.1: Vývojové prostředí STEP 7 [17]

## 5.3 WinCC

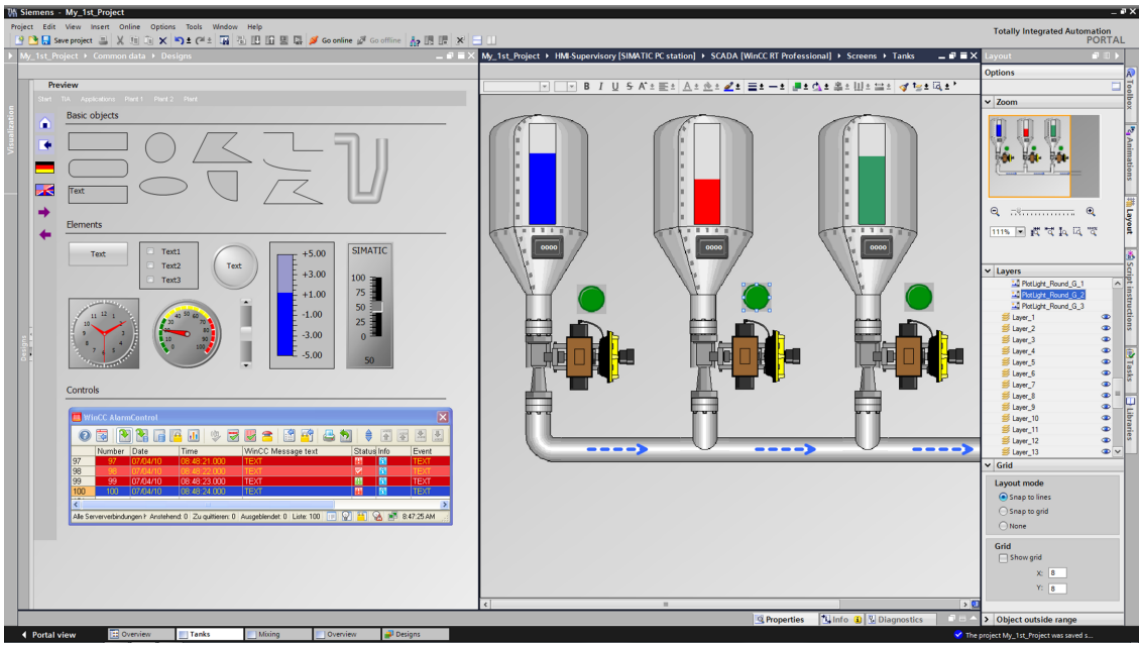
WinCC je prostředí pro návrh vizualizací a ovládání průmyslových procesů. Prostřednictvím uživatelského rozhraní HMI lze na základě konfigurace ve WinCC monitorovat stav automatizovaného systému, zobrazovat hodnoty ze senzorů, nebo ovládat některá zařízení a procesy.

Uživatelská rozhraní vytvořená ve WinCC mohou obsahovat různé prvky, jako jsou tlačítka, přepínače, grafy a textová pole. Je možné také využít pokročilých funkcí vizualizace, jako jsou animace, trendy, alarmy a historické záznamy, které umožňují sledovat a analyzovat chování procesů v reálném čase.

Kromě vlastních obrazovek lze ve WinCC také vytvářet šablony pro obrazovky, knihovny a vlastní objekty. HMI panely mají vlastní paměť, ve které se nacházejí mimo jiné i HMI tagy. Tyto tagy fungují obdobně jako PLC tagy a jejich hodnoty jsou obvykle v nějaké podobě zobrazovány na HMI panelu. Například pro správnou funkčnost přepínače musí existovat HMI tag, který bude uchovávat hodnotu přepínače, tj. 0 nebo 1.

Všechny prvky na obrazovce se skládají do vrstev. Na každé obrazovce je 31 vrstev a v každé vrstvě lze vytvářet skupiny a podskupiny. Platí, že prvky ve vrstvě s nižším číslem jsou na obrazovce níže než prvky ve vrstvě s vyšším číslem.

Integrace WinCC do TIA Portalu pak velmi zjednodušuje proces připojení panelu k PLC, což může být realizováno libovolnou průmyslovou sítí, například průmyslovým ethernetem s protokoly Profibus nebo Profinet. Jakmile je nastaveno spojení HMI panelu a PLC, pak je možné spojit HMI tagy s PLC tagy společně s intervalem synchronizace hodnot tagů.




Obrázek 5.2: Prostředí pro vizualizace WinCC [18]

## 5.4 SIMATIC S7-PLCSIM Advanced

SIMATIC S7-PLCSIM Advanced je simulátor PLC procesorů vyvíjený společností Siemens. Jedná se o součást ekosystému TIA Portal, která je ovšem samostatně instalovatelná.

Hlavní funkcí simulátoru je poskytnutí virtuálního prostředí, ve kterém je možné simulovat chování PLC bez nutnosti mít fyzické zařízení.

Vzhledem k tomu, že se jedná o software vyvíjený společností Siemens pouze pro potřeby ekosystému TIA Portal, tak je možné simulovat pouze PLC procesory společnosti Siemens. To zahrnuje PLC z rodin SIMATIC S7-1500, S7-1200, S7-300 a 400 [19].

Je plně integrován do ekosystému TIA Portal, což zjednodušuje přenos projektu mezi vývojovým prostředím a virtuálním prostředím simulátoru. Díky tomu je možné připojení k dalším simulátorům, které mohou například simulovat uživatelská rozhraní HMI panelů. To umožňuje testovat celkové chování systému a následné ověření jeho správného fungování.

# 6 Struktura projektu virtuální sladovny

Projekt laboratorních úloh má podobu zjednodušené sladovny pivovaru. Studentovi bude poskytnut soubor s projektem, ve kterém je již naprogramována funkční sladovna, která je ovládána manuálně. Součástí projektu je jedna obrazovka v HMI panelu, která bude simulována a na které se nacházejí všechny ovládací prvky sladovny, jako jsou ventily, pásy a tanky.

Pro všechny simulované objekty bude předem vytvořena logika v procesoru tak, aby se simulované objekty chovaly co nejvíce realisticky. Kromě toho budou předem naprogramovány události, které zajistí ještě větší realističnost. Například bude vytvořeno hlášení o přetížení dopravníkového pásu, pokud bude na dopravníkový pás aplikovaná příliš velká hmotnost.

## 6.1 Cíl laboratorních úloh

Laboratorní úlohy mají za úkol přiblížit studentům základní prvky návrhu automatizačních systémů a naučit je, jak takové systémy programovat. V průběhu výuky se studenti nejprve seznámí se základy adresace, osvojí si základní datové typy, logické instrukce a operace. Dále se naučí jaké jsou možnosti správy dat a jak používat pokročilé instrukční bloky, jako jsou časovače a čítače.

Napříč všemi laboratorními úlohami budou studenti pracovat na tvorbě organizačních bloků, funkcí, funkčních bloků a datových bloků, což jsou základní stavební prvky pro programování logiky automatizovaného systému.

## 6.2 Schéma projektu

Celý projekt je rozdělen celkem do 4 částí - programové části, vizualizační části, datové části a simulační části.

### 6.2.1 Programová část

Programová část se stará především o celkovou logiku projektu a je zde uložen veškerý kód. Tuto část lze rozdělit do 3 sekcí - sekce ovládání, sekce realističnosti a studentská sekce.

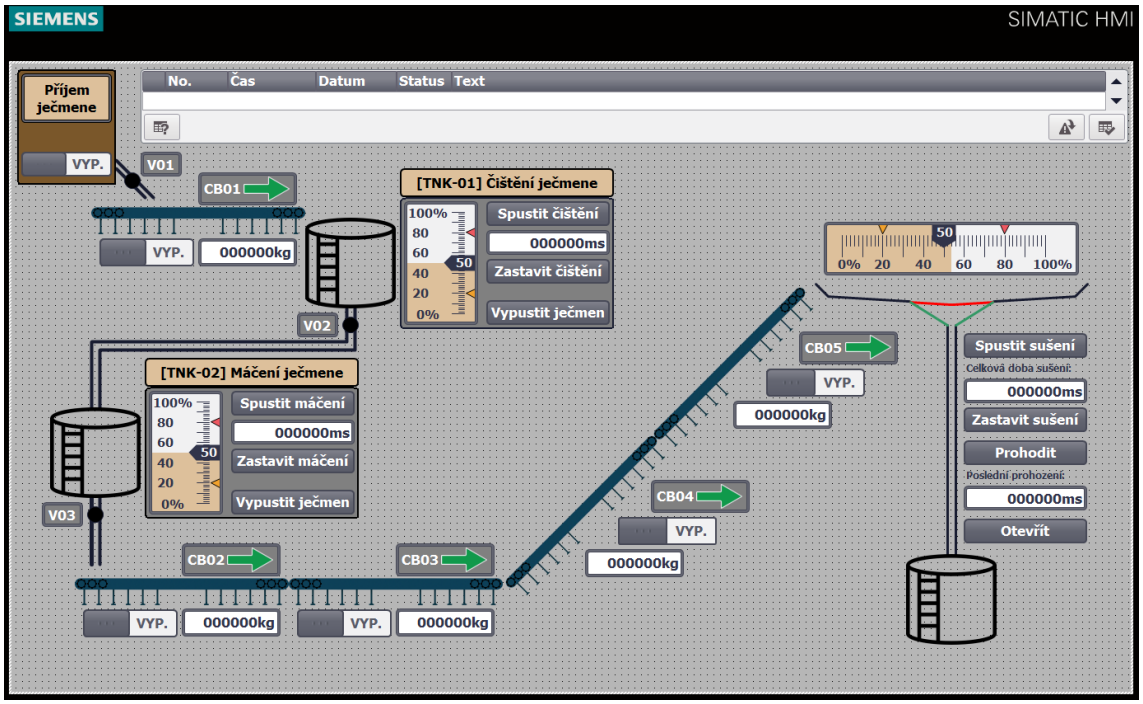
Sekce ovládání se stará především o chod a ovládání virtuálních zařízení sladovny. Příkladem této sekce může být program, který obsluhuje dopravníkový pás s přepínačem. Pokud je přepínač v poloze 1, virtuální dopravníkový pás se rozjede. Pokud je přepínač v poloze 0, dopravníkový pás se zastaví.

Sekce realističnosti se stará o to, aby se virtuální prvky sladovny ze simulační části chovaly realisticky. Příkladem je program, který ukončí činnost celé sladovny pokud je na dopravníkovém pasu překročena nosnost dopravníkového pásu, která je definovaná v simulační části. Zároveň se s ukončením činnosti sladovny vyvolá upozornění, které dá zpětnou vazbu studentovi.

K tomu, aby se student nemusel zabývat tím jak funguje program uložený v sekci ovládání a realističnosti, tak byly tyto sekce navrženy tak, aby je nebylo potřeba jakkoli upravovat a student si tak může vytvářet vlastní programové bloky nezavisle na zbylém programu. Takové bloky pak tvoří studenskou sekci.

### 6.2.2 Vizualizační část

O vizualizaci se stará WinCC prostředí, ve kterém je nakonfigurovaná jedna obrazovka na které je vyobrazena celá sladovna. Prostřednictvím této obrazovky se ovládají a monitorují všechna virtuální zařízení sladovny. Součástí je i zobrazení alarmů, které zjednoduší studentovi hledání chyb v programu.




Obrázek 6.1: Vizualizace virtuální sladovny

Od studenta se očekává, že bude pracovat pouze se simulovaným HMI panelem a nijak nebude do paměti HMI zasahovat. Veškeré prvky v uživatelském rozhraní jsou nakonfigurovány společně s HMI tagy a alarmy. Paměť HMI panelu je v již spojena se simulovaným PLC procesorem.

### 6.2.3 Datová a simulační část

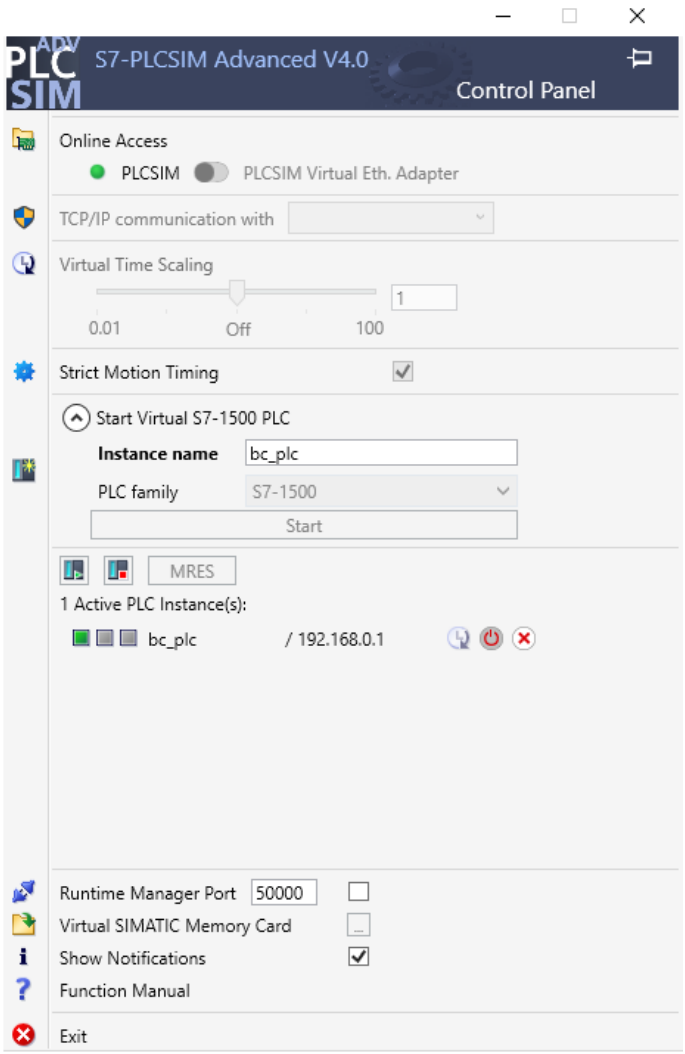
Data v projektu jsou rozdělena do několika tabulek tagů a datových bloků. Hlavní tabulkou tagů je I/O ve kterém se nacházejí všechny tagy, které by byly potřebné v případě programování reálného PLC se vstupními a výstupními moduly. Nacházejí se zde například tagy pro všechny přepínače, tlačítka, pásy, apod.

Hlavním datovým blokem je datový blok simulace, ve kterém se nacházejí data pro definování simulačních podmínek sladovny, jako je například maximální nosnost pásů nebo rychlost vypouštění tanků. Tento blok tak představuje zajištění škálovatelnosti obtížnosti úloh, protože je možné nastavit podmínky simulace velmi benevolentní, či naopak velmi striktní.

## 6.3 Vytvoření projektu a nastavení simulace CPU

Pro projekt byla zvolena simulace procesoru SIMATIC S7-1500 CPU 1513-1 PN. Rodina SIMATIC S7-1500 byla zvolena za účelem plné využitelností prostředí STEP 7. Nejvýznamnější rozdíly mezi procesory z rodiny SIMATIC S7-1200 a S7-1500 jsou především v hardwaru. Takové rozdíly jsou ale při simulaci zanedbatelné. Hlavním rozdílem z hlediska softwaru je edice prostředí STEP 7. Zatímco jsou procesory SIMATIC S7-1200 programovány v prostředí Step 7 Basic, tak procesory S7-1500 jsou programovány v prostředí STEP 7 Professional. Důležitým rozdílem mezi edicemi Basic a Professional je z hlediska tohoto projektu to, že v edici Professional lze programovat v jazyce STL, zatímco v edici Basic nikoli.

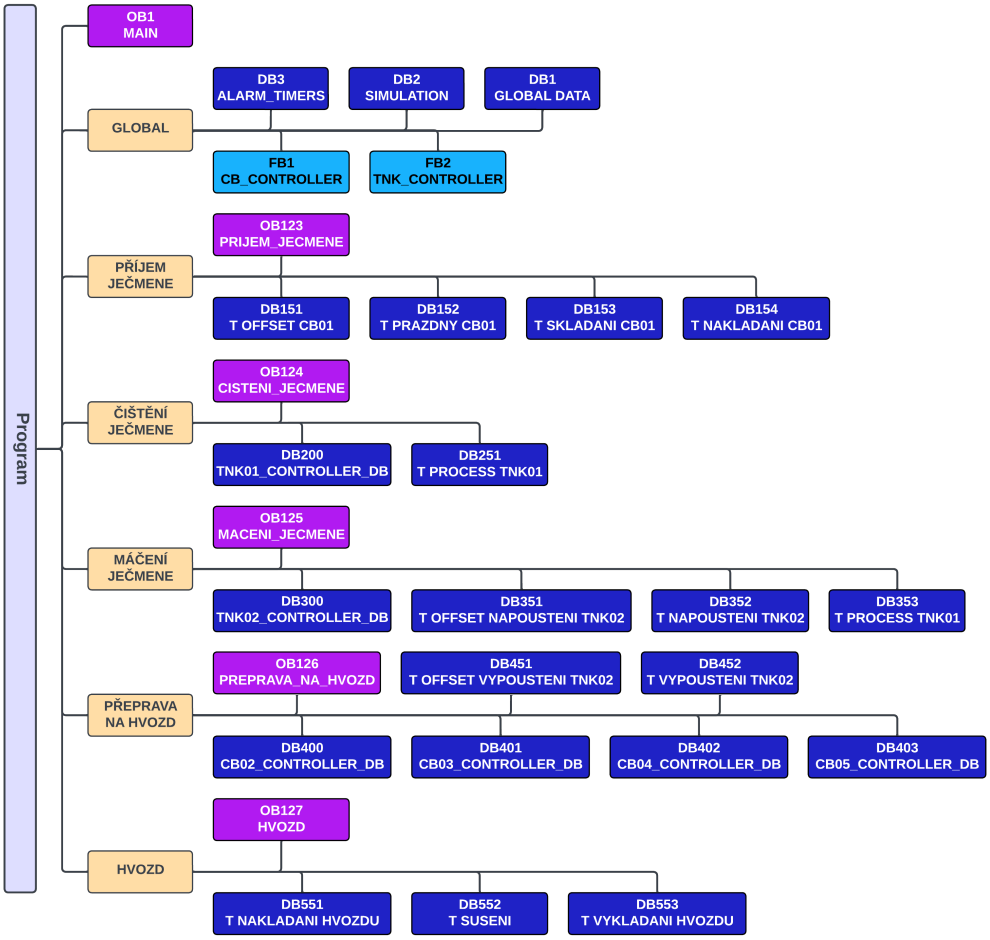
Virtuální prostředí SIMATIC S7-PLCSIM Advanced umožňuje dva typy komunikace s prostředím STEP 7. Jednak s využitím proprietárního protokolu Softbus a jednak s využitím protokolu TCP/IP [19]. Pro projekt byla zvolena komunikace s protokolem Softbus. Softbus byl navržen přímo pro simulaci PLC díky čemuž se minimalizuje latence komunikace mezi STEP 7 a PLC. Nevýhodou protokolu Softbus oproti TCP/IP je ta, že při využití protokolu TCP/IP může virtuální prostředí simulace PLC běžet na jiném stroji, než na kterém je spuštěn STEP 7.




Obrázek 6.2: Simulace PLC S7-1500

# 7 Programování logiky PLC

Pro lepší orientaci v kódu jsou programové bloky uloženy ve struktuře složek dle následujícího schématu.




Obrázek 7.1: Schéma rozdělení programových bloků

## 7.1 Nahrazení externích dat

Vzhledem k tomu, že je v projektu pouze simulované PLC a ne reálné, tak není vhodné používat k adresaci oblasti adres vstupních periferií I a adresy výstupních periferií Q. V těchto případech očekává TIA portal, že existují vstupní a výstupní moduly ke kterým je jsou připojená vstupní a výstupní zařízení. Pokud bychom hodnotu takového tagu chtěli upravit přímo v programu, nebo by tuto hodnotu upravoval prvek v uživatelském rozhraní HMI, tak by se tato hodnota nastavila pouze na jeden cyklus. V dalším cyklu by byla tato hodnota automaticky nastavena na hodnotu 0, protože k simulovanému PLC nejsou připojeny žádné vstupní nebo výstupní moduly.

Tomuto problému lze předejít pomocí použití adres bitové paměti M, které jsou ukládány, čteny a zapisovány do obecné datové paměti. Hodnoty adres typu I a Q se oproti hodnotám adres typu M neukládají do pamětí na déle než jeden programový cyklus.

Dalším řešením je místo obecné datové paměti použít datový blok. Vzhledem k povaze tagů, které mají představovat tagy reálných vstupních a výstupních periferií, byla tato

možnost zavrhnuta, protože v případě reálného návrhu automatizovaného systému se tagy vstupních a výstupních periferií ukládají do obecné datové paměti.

Výsledná tabulka tagů pro vstupní a výstupní prvky vypadá následovně:

Tabulka 7.1: Tabulka tagů simulovaných vstupů a výstupů

| **Název** | **Adresa** | **Datový typ** | **Popis** |
|---|---|---|---|
| TNKPJ_SW | %M0.0 | Bool | Přepínač ventilu příjmu ječmene |
| CB01_SW | %M0.1 | Bool | Přepínač pohonu dopravníkového pásu CB01 |
| CB02_SW | %M0.2 | Bool | Přepínač pohonu dopravníkového pásu CB02 |
| CB03_SW | %M0.3 | Bool | Přepínač pohonu dopravníkového pásu CB03 |
| CB04_SW | %M0.4 | Bool | Přepínač pohonu dopravníkového pásu CB04 |
| CB05_SW | %M0.5 | Bool | Přepínač pohonu dopravníkového pásu CB05 |
| HVOZD_BTN03 | %M0.6 | Bool | Tlačítko otevření hvozdu |
| HVOZD_BTN04 | %M0.7 | Bool | Tlačítko prohození hvozdu |
| TNK01_BTN01 | %M1.0 | Bool | Tlačítko spuštění procesu v tanku TNK01 |
| TNK01_BTN02 | %M1.1 | Bool | Tlačítko zastavení procesu v tanku TNK01 |
| TNK01_BTN03 | %M1.2 | Bool | Tlačítko vypuštění tanku TNK01 |
| TNK02_BTN01 | %M1.3 | Bool | Tlačítko spuštění procesu v tanku TNK02 |
| TNK02_BTN02 | %M1.4 | Bool | Tlačítko zastavení procesu v tanku TNK02 |
| TNK02_BTN03 | %M1.5 | Bool | Tlačítko vypuštění tanku TNK02 |
| HVOZD_BTN01 | %M0.6 | Bool | Tlačítko spuštění hvozdu |
| HVOZD_BTN02 | %M0.7 | Bool | Tlačítko zastavení hvozdu |
| CB01_M | %MW2 | Int | Váha na dopravníkovém pásu CB01 |
| CB02_M | %MW4 | Int | Váha na dopravníkovém pásu CB02 |
| CB03_M | %MW6 | Int | Váha na dopravníkovém pásu CB03 |
| CB04_M | %MW8 | Int | Váha na dopravníkovém pásu CB04 |
| CB05_M | %MW10 | Int | Váha na dopravníkovém pásu CB05 |
| TNK01_LVL | %MW12 | Int | Úroveň hladiny v tanku TNK01 |
| TNK02_LVL | %MW14 | Int | Úroveň hladiny v tanku TNK02 |
| HVOZD_LVL | %MW16 | Int | Úroveň hladiny ve hvozdu |
| TNKPJ_V_OUT | %M50.0 | Bool | V01 Výstupní ventil tanku příjmu ječmene |
| TNK01_V_OUT | %M50.1 | Bool | V02 Výstupní ventil tanku TNK01 |
| TNK02_V_OUT | %M50.2 | Bool | V03 Výstupní ventil tanku TNK02 |
| TNK03_D_OUT | %M50.3 | Bool | Sklápěcí dveře hvozdu |
| CB01 | %M51.0 | Bool | Pohon dopravníkového pásu CB01 |
| CB02 | %M51.1 | Bool | Pohon dopravníkového pásu CB02 |
| CB03 | %M51.2 | Bool | Pohon dopravníkového pásu CB03 |
| CB04 | %M51.3 | Bool | Pohon dopravníkového pásu CB04 |
| CB05 | %M51.4 | Bool | Pohon dopravníkového pásu CB05 |

## 7.2 Hlavní bloky

### 7.2.1 OB1 Main

Ze schématu rozdělení programových bloků na obrázku 7.1 je patrné, že každá část sladovny má svůj organizační blok, který danou část ovládá.

V hlavním organizačním bloku OB1 se tady nachází pouze takový kód, který obsluhuje taková data, která jsou společná pro všechny části sladovny. Jedná se hlavně o obnovení dat do výchozích hodnot v případě, kdy nastane chyba při programování, nebo ovládání sladovny.

### 7.2.2 FB1 Kontroler dopravníkového pásu

Vzhledem k četnému použití dopravníkových pásů, které se všechny chovají stejně byl pro účely ovládání dopravníkových pásů vytvořen funkční blok, který lze použít na libovolný dopravníkový pás.

Ze *static* sekce tabulky interních tagů funkčního bloku FB1 je patrné, že v tomto funkčním bloku byla použita metoda multi-instance datových bloků pro datové bloky časovačů.

Tabulka 7.2: Interní tagy FB1 - Kontroler dopravníkového pásu

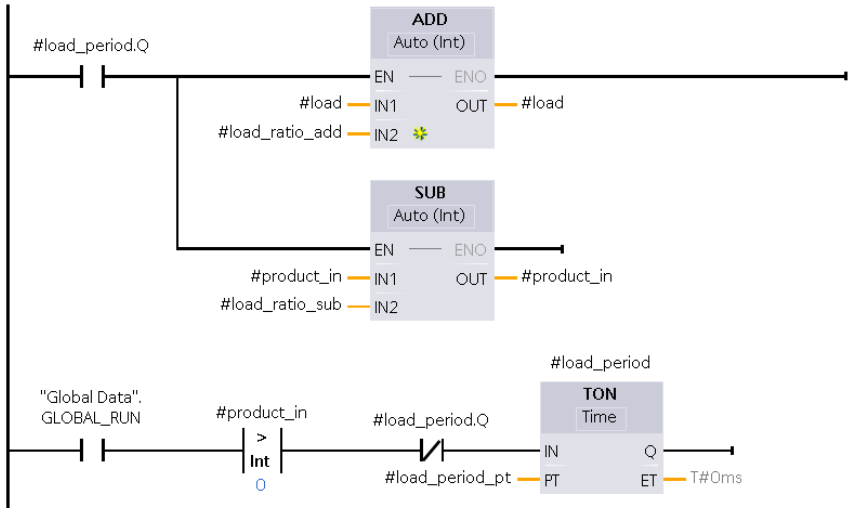
| **Skupina** | **Název** | **Datový typ** | **Popis** |
|---|---|---|---|
| Vstup | cb_sw | Bool | Přepínač ovládání dopravníkového pásu |
| | load_ratio_sub | Int | Množství produktu odečteného od vstupního produktu při nakládání |
| | load_ratio_add | Int | Množství produktu přičteného k aktuálnímu zatížení pásu při nakládání |
| | load_period_pt | Time | Perioda nakládání |
| | unload_ratio_sub | Int | Množství produktu odečteného od aktuálního zatížení pásu při vykládání |
| | unload_ratio_add | Int | Množství produktu přičteného k výstupnímu produktu při vykládání |
| | unload_period_pt | Time | Perioda vykládání |
| Výstup | cb | Bool | Motor dopravníkového pásu |
| | product_out | Int | Množství produktu na výstupu |
| Vstup - Výstup | product_in | Int | Množství produktu na vstupu |
| | load | Int | Aktuální zatížení dopravníkového pásu |
| Static | load_period | TON_TIME DB | |
| | ↳ PT | Time | Naprogramovaný čas časovače |

*Pokračování na další stránce*

Tabulka 7.2: Interní tagy FB1 - Kontroler dopravníkového pásu *(Pokračování)*

| Skupina | Název | Datový typ | Popis |
|---|---|---|---|
| Static | ↳ ET | Time | Uběhlý čas časovače |
| | ↳ IN | Bool | Vstup časovače |
| | ↳ Q | Bool | Výstup časovače |
| | unload_period | TON_TIME DB | |
| | ↳ PT | Time | Naprogramovaný čas časovače |
| | ↳ ET | Time | Uběhlý čas časovače |
| | ↳ IN | Bool | Vstup časovače |
| | ↳ Q | Bool | Výstup časovače |

O nakládání a vykládání dopravníkového pásu se stará algoritmus periodického přičítání a odečítání vyobrazený na obrázku níže.




Obrázek 7.2: Program nakládání pásu

Obdobný program je následně aplikován pro vykládání pásu s pozměněnými tagy.

### 7.2.3 FB2 Kontroler tanku

Ačkoli je prvek tanku použit pouze dvakrát, stále se vyplatí vytvoření kontroleru. Z hlediska ovládání jsou důležité vstupní prvky, které tvoří 3 tlačítka - spuštění procesu, zastavení procesu a vypuštění tanku. Dalším důležitým prvkem je hlídání hladiny. To je realizováno tagy `TNK01_LVL` na adrese %MW12 a `TNK02_LVL` na adrese %MW14. Jediným výstupním ovládacím prvkem jsou ventily `TNK01_V_OUT` a `TNK02_V_OUT`, které simulují ventily vypouštění tanků.

Důležitými daty z hlediska realističnosti je struktura `state`, která je vytvořena pro oba tanky v datovém bloku DB1. Tato struktura popisuje v jakém stavu procesu se momentálně tank nachází. Jedná se o 5 stavů přičemž aktivní, tj. stav s hodnotou 1, může být pouze jeden z nich. Přepínání stavů dělá program automaticky. Tím pádem nebude nutné, aby se student zabýval změnou stavu při programování vlastních úloh. Jedná se tak o simulaci reálného prostředí kdy je například skutečné vypouštění tanku patrné

z ubývající hladiny a nenulovým prutokem v potrubí za tankem. Toto chování je zde nahrazeno tagem stavu a odpovídajícími programy - pro příklad vypouštění je v kontroleru algoritmus, který zajišťuje odečítaní hodnoty z tagu `lvl` za předpokladu, že je otevřen výstupní ventil tanku.

Stejně jako v případě funkčního bloku FB1 - Kontroler pásů, byla i zde použita metoda multi-instance datových bloků. Všechny interní tagy kontroleru tanků jsou v následující tabulce.

Tabulka 7.3: Interní tagy FB2 - Kontroler tanků

| **Skupina** | **Název** | **Datový typ** | **Popis** |
|---|---|---|---|
| Vstup | lvl_max | Int | Maximální hladina tanku |
| | lvl_min | Int | Minimální hladina tanku |
| | btn_start | Bool | Tlačítko spuštění |
| | btn_stop | Bool | Tlačítko zastavení |
| | btn_open | Bool | Tlačítko vypuštění |
| | process_time_min | Time | Minimální doba potřebná úspěšnému dokončení procesu |
| | process_time_max | Time | Maximální doba potřebná úspěšnému dokončení procesu |
| | unload_period | Time | Maximální doba běhu pásu naprázdno |
| Výstup | valve | Bool | Ventil vypuštění tanku |
| | product | Int | Množství produktu na výstupu |
| | process_time_et | Time | Uběhlá doba procesu |
| | state | Struct | Struktura stavů procesu |
| | ↳ ready_to_fill | Bool | Tank je připraven na plnění |
| | ↳ filling | Bool | Tank se plní |
| | ↳ running | Bool | Probíhá proces |
| | ↳ ended | Bool | Proces byl úspěšně dokončen |
| | ↳ emptying | Bool | Tank se vypouští |
| | overload | Bool | Tank přetekl |
| Vstup - Výstup | gr | Bool | GLOBAL RUN |
| | lvl | Int | Hlídání hladiny tanku |
| Static | emptying | TON_TIME DB | Časovač periody vypouštění |
| | ↳ PT | Time | Naprogramovaný čas časovače |
| | ↳ ET | Time | Uběhlý čas časovače |
| | ↳ IN | Bool | Vstup časovače |
| | ↳ Q | Bool | Výstup časovače |
| | process | TON_TIME DB | Časovač procesu |
| | ↳ PT | Time | Naprogramovaný čas časovače |
| | ↳ ET | Time | Uběhlý čas časovače |

*Pokračování na další stránce*

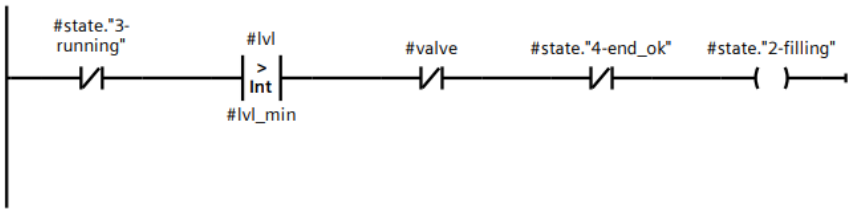
Tabulka 7.3: Interní tagy FB2 - Kontroler tanků *(Pokračování)*

| Skupina | Název | Datový typ | Popis |
|---|---|---|---|
| Static | ↳ IN | Bool | Vstup časovače |
| | ↳ Q | Bool | Výstup časovače |

Výchozí stav tanku je takový, že je prázdný, tj. `lvl=0` a je připraven k naplnění `state.ready_to_fill=1`. Předpokládá se, že nejdříve je potřeba tank naplnit. Vzhledem k tomu, že simulované tanky nemají žádnou regulaci ječmene vstupujícího do tanku, tak program, který obsluhuje plnění tanku je vždy umístěn v takovém organizačním bloku, za kterým je tank umístěn. Program plnění tanku TNK01 pro čištění ječmene je umístěn v organizačním bloku OB123 pro příjem ječmene. Ječmen zde padá z pásu CB01 přímo do tanku TNK01 a jediná regulace na vstupu tanku je přepínač ovládající pohon dopravníkového pásu CB01. V případě tanku TNK02 pro máčení ječmene padá vyčištěný ječmen přímo z potrubí mezi tanky TNK01 a TNK02. Regulace na vstupu tanku je tak výstupní ventil tanku TNK01.

Následující program zajišťuje automatickou detekci toho, že je do tanku přidáván ječmen. V takovém případě je tedy potřeba změnit stav daného tanku. Stav tanku se změní z `state.ready_to_fill` na `state.filling` a setrvá v tomto stavu tak dlouho, dokud budou platit všechny tyto podmínky:

- tank není ve stavu `state.running`,
- hladina v tanku `lvl` je vetší, než minimální hladina `lvl_min` definovaná v datovém bloku DB2,
- výstupní ventil tanku je uzavřený a tank není ve stavu ended

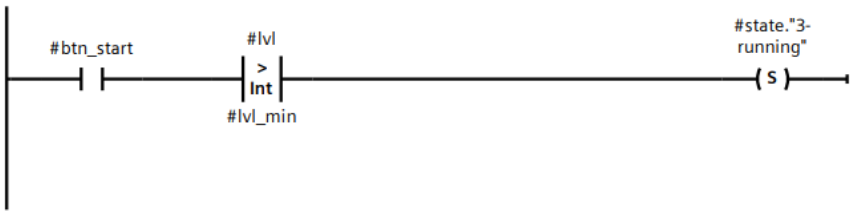



Obrázek 7.3: Program detekce plnění tanku

Pokud nebude včas vypnut zdroj ječmene dodávaného do tanku, stane se, že tank přeteče. To je realizováno porovnáním aktuální hladiny v tanku `lvl` a maximální možnou hladinou `lvl_max`, která je definovaná v datovém bloku DB2. Pokud se stane, že je aktuální hladina v tanku větší než maximální možná hladina v tanku, pak se program resetuje se odpovídajícím alarmem, který se zobrazí v uživatelském rozhraní.

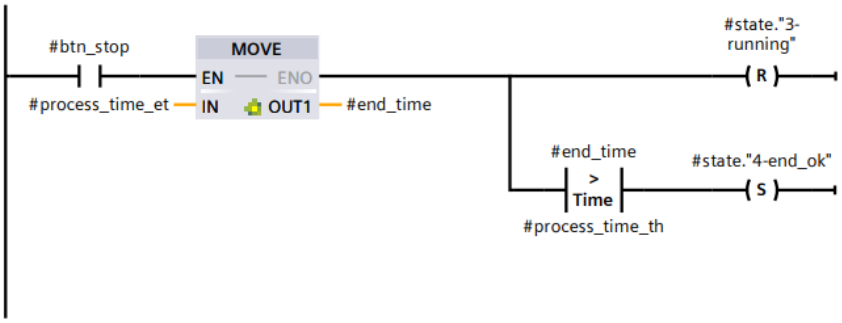

Obrázek 7.4: Program detekce přetečení tanku

Jakmile je tank naplněn, je možné pomocí tlačítka `btn_start` spustit takový proces, pro který je tank určen. Po stisknutí tohoto tlačítka se nastaví `state.running` na hodnotu 1. To poruší první z podmínek programu detekce plnění tanku a hodnota stavu `state.filling` je automaticky nastavena zpět na 0.




Obrázek 7.5: Program spuštění procesu

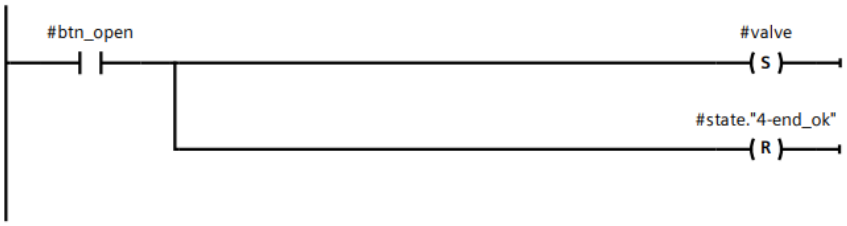
Pokud student, nebo jeho program zastaví proces tanku tlačítkem `btn_stop`, konečná doba běhu procesu tanku `end_time` se porovná s minimální akceptovatelnou dobou běhu `process_time_min` a status `state.running` se nastaví na hodnotu 0. Minimální akceptovatelná doba běhu je uložena v interním tagu `process_time_min` a pro oba tanky je tato doba definovaná v datovém bloku DB2. Pokud je doba běhu procesu větší než minimální doba, tak se nastaví status `state.ended`. Pokud je menší, program se resetuje s odpovídajícím alarmem, který se zobrazí v uživatelském rozhraní.




Obrázek 7.6: Program zastavení procesu

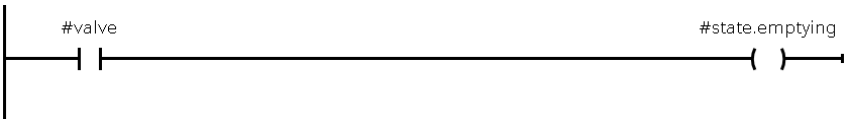
Pokud byl proces úspěšně dokončen, pak je možné tank vypustit. Vypouštění není možné regulovat a vypustí se vždy vše co je uvnitř tanku. Pokud se tedy nastaví poměry v datovém bloku DB2 tak, že z tanku TNK01 vystoupí méně ječmene, než kolik do něj vstoupilo (což se v reálných podmínkách při čištění ječmene děje), pak se automatizační úloha stane velmi náročnou. V takovém případě totiž není možné aby oba tanky byly spuštěné vždy se stoprocentně využitou kapacitou a při automatizaci plnění tanku TNK02 by bylo nutné počítat s takto upraveným poměrem.

Vypouštění se spouští tlačítkem `btn_open`. Po spuštění vypouštění se otevře ventil vypouštění tanku a hodnota stavu `state.ended` se změní na 0. S otevřením ventilu se zároveň nastaví stav `state.emptying`.




Obrázek 7.7: Program otevření ventilu

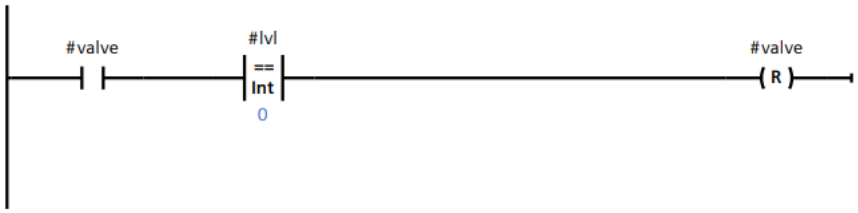
Po dobu, co je ventil otevřený je nastaven stav `state.emptying` na hodnotu 1.




Obrázek 7.8: Program změny stavu na emptying

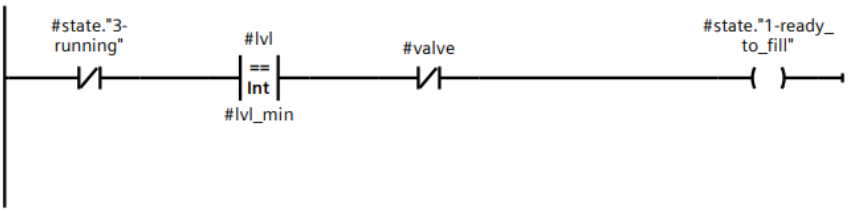
Vypouštění následně probíhá stejným algoritmem, jaký byl použit ve funkčním bloku FB01 - Kontroler dopravníkového pasu pro nakládání a skládání pásu.

Jakmile je tank vypuštěný, tj `lvl=0`, tak se ventil automaticky zavře. Tím se zruší podmínka otevřeno ventilu pro stav `state.emptying` a tak se hodnota tohoto stavu vrátí na hodnotu 0.




Obrázek 7.9: Program uzavření ventilu

Po dokončení vypouštění tanku se nastaví opět stav `state.ready_to_fill`. Ten je nastavený za podmínky, že v tanku neběží proces, tank je prázdný a výstupní ventil tanku je zavřený.




Obrázek 7.10: Detekce stavu ready_to_fill

### 7.2.4 DB1 Globální data

V datovém bloku DB1 jsou uloženy všechny interní data mimo alarmy a simulace. Hlavním tagem je tag `GLOBAL_RUN`, který uchovává informaci o tom, zda simulace sladovny běží v pořádku bez narušení sekce realističnosti. Dale se zde nacházejí data stavová a výpočetní pro všechny prvky sladovny.

Pro většinu interních tagů platí, že je výhodné používat strukturovanou paměť místo obecné. To je dáno zejména tím, že ve strukturované datové paměti lze vytvářet struktury. Ty jsou v tomto datovém bloku vytvořeny pro každý prvek sladovny a v některých případech jsou vytvořeny i struktury uvnitř struktur prvků, např. stavové struktury tanků.

Tabulka 7.4: Datový blok DB1 - Globální data

| **Název** | **Datový typ** | **Popis** |
|---|---|---|
| GLOBAL_RUN | Bool | GLOBAL RUN |
| tank01 | Struct | Tank TNK01 |
| ↳ lvl_ref | Int | Referenční hladina tanku |
| ↳ process_time_et | Time | Uběhlý čas procesu tanku |
| ↳ product_out | Int | Výstupní produkt |
| ↳ state | Struct | Stav tanku |
| ↳ ready_to_fill | Bool | Stav "připraven k plnění" |
| ↳ filling | Bool | Stav "plnění" |
| ↳ running | Bool | Stav "běh procesu" |
| ↳ ended | Bool | Stav "proces dokončen" |
| ↳ emptying | Bool | Stav "vypouštění tanku" |
| tank02 | state | Tank TNK02 |
| ↳ lvl_ref | Int | Referenční hladina tanku |
| ↳ process_time_et | Time | Uběhlý čas procesu tanku |
| ↳ product_out | Int | Výstupní produkt |
| ↳ state | Struct | Stav tanku |
| ↳ ready_to_fill | Bool | Stav "připraven k plnění" |
| ↳ filling | Bool | Stav "plnění" |
| ↳ running | Bool | Stav "běh procesu" |
| ↳ ended | Bool | Stav "proces dokončen" |
| ↳ emptying | Bool | Stav "vypouštění tanku" |
| hvozd | Struct | HVOZD |
| ↳ lvl_ref | Int | Referenční hladina hvozdu |
| ↳ process_time_et | Time | Uběhlý čas sušení |
| ↳ turn_time_et | Time | Uběhlý čas bez prohození |
| ↳ state | Struct | Stav hvozdu |
| ↳ ready_to_fill | Bool | Stav "připraven k plnění" |
| ↳ filling | Bool | Stav "plnění" |
| ↳ running | Bool | Stav "běh procesu" |
| ↳ ended | Bool | Stav "proces dokončen" |
| ↳ emptying | Bool | Stav "vypouštění hvozdu" |

### 7.2.5 DB2 Simulace

Datový blok DB2 slouží pro ukládaní a nastavování simulačních podmínek. Jedná se o poměry a periody nakládání a skládání dopravníkových pásu a tanků, maximální nosnosti dopravníkových pásů, minimální a maximální hladiny tanků a jiné.

Tabulka 7.5: Datový blok DB2 - Simulace

| **Název** | **Datový typ** | **Výchozí hodnota** | **Popis** |
|---|---|---|---|
| conveyor_belt01 | Struct | | Dopravníkový pás CB01 |
| ↳ load_ratio_add | Int | 1 | Poměr nakládání - přičítání |
| ↳ load_period | Time | T#2S | Perioda nakládání pásu |
| ↳ unload_time_offset | Time | T#5S | Prodleva mezi naložením a skládáním |
| ↳ unload_period | Time | T#2S | Perioda skládání pásu |
| ↳ empty_belt | Time | T#15S | Maximální doba běhu naprázdno |
| ↳ max_load | Int | 10 | Maximální nosnost pásu |
| conveyor_belt02 | Struct | | Dopravníkový pás CB02 |
| ↳ load_ratio_sub | Int | 1 | Poměr nakládání - odečítání |
| ↳ load_ratio_add | Int | 1 | Poměr nakládání - přičítání |
| ↳ load_period | Time | T#2S | Perioda nakládání pásu |
| ↳ unload_ratio_sub | Int | 1 | Poměr skládání - přičítaní |
| ↳ unload_ratio_add | Int | 1 | Poměr skládání - odečítaní |
| ↳ unload_period | Time | T#2S | Perioda skládání pásu |
| ↳ empty_belt_th | Time | T#15S | Maximální doba běhu naprázdno |
| ↳ max_load | Int | 10 | Maximální nosnost pásu |
| conveyor_belt03 | Struct | | Dopravníkový pás CB03 |
| ↳ load_ratio_sub | Int | 1 | Poměr nakládání - odečítání |
| ↳ load_ratio_add | Int | 1 | Poměr nakládání - přičítání |
| ↳ load_period | Time | T#2S | Perioda nakládání pásu |
| ↳ unload_ratio_sub | Int | 1 | Poměr skládání - přičítaní |
| ↳ unload_ratio_add | Int | 1 | Poměr skládání - odečítaní |
| ↳ unload_period | Time | T#2S | Perioda skládání pásu |
| ↳ empty_belt_th | Time | T#15S | Maximální doba běhu naprázdno |
| ↳ max_load | Int | 10 | Maximální nosnost pásu |
| conveyor_belt04 | Struct | | Dopravníkový pás CB04 |
| ↳ load_ratio_sub | Int | 1 | Poměr nakládání - odečítání |
| ↳ load_ratio_add | Int | 1 | Poměr nakládání - přičítání |
| ↳ load_period | Time | T#2S | Perioda nakládání pásu |
| ↳ unload_ratio_sub | Int | 1 | Poměr skládání - přičítaní |
| ↳ unload_ratio_add | Int | 1 | Poměr skládání - odečítaní |
| ↳ unload_period | Time | T#2S | Perioda skládání pásu |

*Pokračování na další stránce*

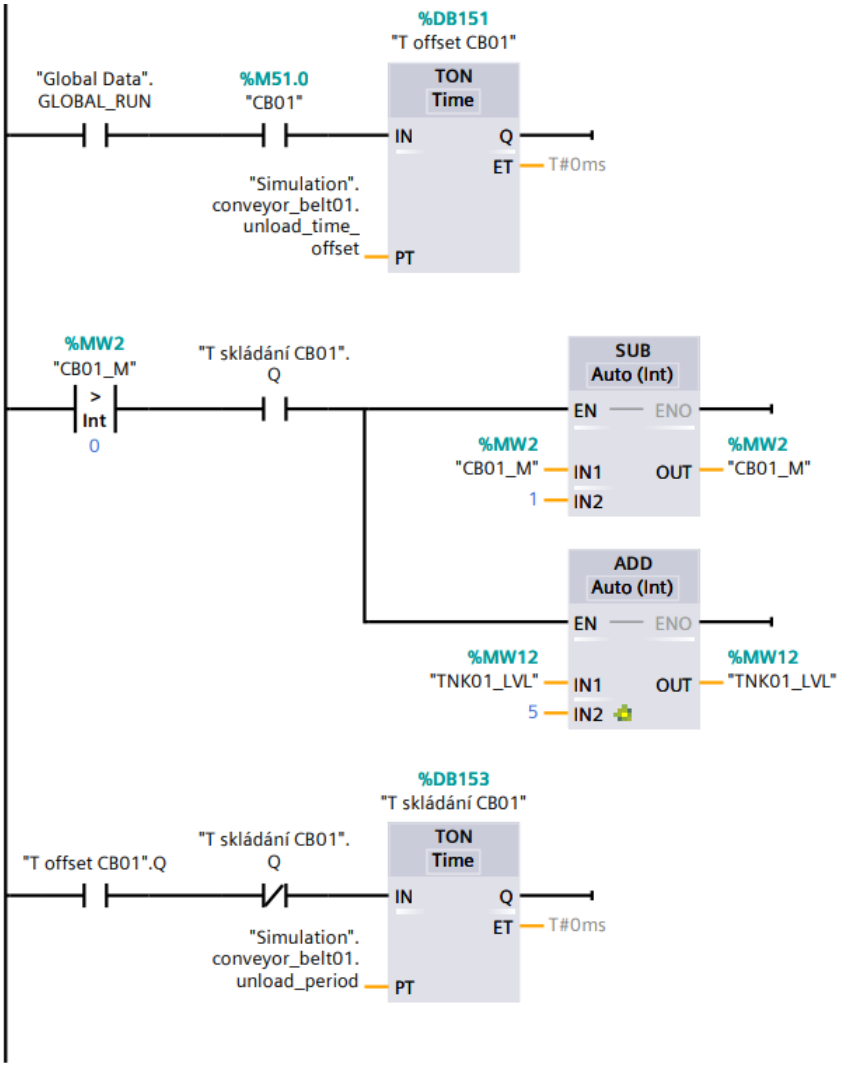
Tabulka 7.5: Datový blok DB2 - Simulace *(Pokračování)*

| **Název** | **Datový typ** | **Výchozí hodnota** | **Popis** |
|---|---|---|---|
| ↳ empty_belt_th | Time | T#15S | Maximální doba běhu naprázdno |
| ↳ max_load | Int | 10 | Maximální nosnost pásu |
| conveyor_belt05 | Struct | | Dopravníkový pás CB05 |
| ↳ load_ratio_sub | Int | 1 | Poměr nakládání - odečítání |
| ↳ load_ratio_add | Int | 1 | Poměr nakládání - přičítání |
| ↳ load_period | Time | T#2S | Perioda nakládání pásu |
| ↳ unload_ratio_sub | Int | 1 | Poměr skládání - přičítaní |
| ↳ unload_ratio_add | Int | 1 | Poměr skládání - odečítaní |
| ↳ unload_period | Time | T#2S | Perioda skládání pásu |
| ↳ empty_belt_th | Time | T#15S | Maximální doba běhu naprázdno |
| ↳ max_load | Int | 10 | Maximální nosnost pásu |
| tank01 | Struct | | Tank TNK01 |
| ↳ lvl_min | Int | 0 | Minimální hladina tanku |
| ↳ lvl_max | Int | 100 | Maximální hladina tanku |
| ↳ process_time_max | Time | T#22S | Maximální doba běhu procesu |
| ↳ process_time_min | Time | T#18S | Minimální doba běhu procesu |
| ↳ unload_period | Time | T#1S | Perioda vypouštění tanku |
| tank02 | Struct | | Tank TNK02 |
| ↳ lvl_min | Int | 0 | Minimální hladina tanku |
| ↳ lvl_max | Int | 100 | Maximální hladina tanku |
| ↳ process_time_max | Time | T#22S | Maximální doba běhu procesu |
| ↳ process_time_min | Time | T#18S | Minimální doba běhu procesu |
| ↳ unload_period | Time | T#1S | Perioda vypouštění tanku |
| hvozd | Struct | | HVOZD |
| ↳ lvl_min | Int | 0 | Minimální hladina hvozdu |
| ↳ lvl_max | Int | 100 | Maximální hladina hvozdu |
| ↳ turn_time_max | Time | T#12S | Maximální doba bez prohození |
| ↳ process_time_max | Time | T#62S | Maximální doba běhu sušení |
| ↳ process_time_min | Time | T#58S | Minimální doba běhu sušení |
| ↳ unload_period | Time | T#1S | Perioda vypouštění hvozdu |

## 7.3 Příjem ječmene

Na začátku simulace sladovny se nachází část příjmu ječmene. V reálných podmínkách se jedná o jeden či více tanků, které jsou plněny ječmenem, který je přivážen v kamionech, vlacích, či jiným způsobem přepravy. V simulaci sladovny je toto nahrazeno jedním, nekonečně velkým tankem s neomezeným množstvím ječmene.

Přivezený ječmen je potřeba vyčistit. To se děje v čistícím tanku kam je ječmen dopraven prostřednictvím dopravníkových pásů nebo potrubí. Nejprve je ale potřeba ječmen dostat z tanku příjmu ječmene na dopravníkový pás. To je zajištěno ventilem na výstupu tanku. jakmile je tento ventil otevřen přepínačem `TNKPJ_SW`, tak se na základě periodicity nastavené v datovém bloku DB2 začne inkrementovat zatížení na dopravníkovém pásu CB01. Simulace doby přepravy je vyřešena přidáním časového offsetu před algoritmus skládání.




Obrázek 7.11: Program skládání pásu CB01

Pro tuto sekci jsou definované dva alarmy. První alarmem je *přetížení pásu* a vyvolá se, pokud zatížení pásu `CB01_M` je větší než maximální nosnost pásu definovaná v datovém bloku DB2 `Simulation.conveyor_belt01.max_load`. Druhý alarm *neefektivita pásu* se vyvolá pokud běží pohon dopravníkového pásu CB01 déle než je definováno v tagu `Simulation.conveyor_belt01.empty_belt`. Jedná se o opatření, které se v reálných podmínkách často používá pro zefektivnění spotřeby energií.

## 7.4 Čištění ječmene

Jakmile je ječmen přepraven do tanku TNK01 pro čištění ječmene, je ovládací sekce daná instancí funkčního bloku FB2 - Kontroler tanků. Použitím funkčního bloku FB2 je zaveden první z alarmů části čištění ječmene - *přetečení tanku*.

Stejně jako se řeší energetická efektivita dopravníkových pásů, tak se řeší i energetická efektivita tanků. Pokud se tedy student, nebo jeho program pokusí spustit proces čištění ječmene pokud je tank prázdný, tj. `TNK01_LVL:=Simulation.tank01.lvl_min`, vyvolá se alarm *spuštění prázdného tanku*. V reálných podmínkách je zvykem zároveň implementovat program, který dovoluje spuštění procesu tanku až jakmile je tank zaplněný na předem definovanou hladinu. Pro účely simulace sladovny je implementovana pouze podmínka, že tank před spuštěním nesmí být prázdný.

Jeden z hlavních alarmů - *neoprávněné spuštění čištění*, se vyvolá pokud se student, nebo jeho program pokusí o spuštění procesu čištění ječmene v době, kdy je tank TNK01 ve stavu `state.ended` nebo `state.emptying`.

Důležitým alarmem je *neoprávněné přidání ječmene*. Je to ochrana proti přidávání ječmene v průběhu procesu čištění ječmene, přidávání ječmene do již vyčištěného ječmene a přidávání ječmene v době, kdy se tank TNK01 vypouští. Každý cyklus se porovnává hodnota tagů hladiny v tanku `TNK01_LVL` a referenční hodnota hladiny v tanku, která byla uložena při spuštění čištění `"Global Data".tank01.lvl_ref`. Jakmile nastane situace kdy je skutečná hladina v tanku TNK01 vetší než referenční hodnota hladiny v tanku TNK01 a zároveň je tank TNK01 v jakémkoli jiném stavu než `state.filling`, tak se vyvolá alarm *neoprávněné přidání ječmene*. Pokud se skutečná hladina liší od referenční, ale tank TNK01 je ve stavu `state.filling`, pak se pouze aktualizuje referenční hodnota hladiny v tanku TNK01.

Na závěr jsou v části čištění ječmene další dva hlavní alarmy: *neoprávněné zastavení čištění* a *neoprávněné vypouštění tanku*. Stisknutí tlačítka zastavení procesu čištění ječmene `TNK01_BTN02` vyvolá alarm v případě, kdy je tank TNK01 ve stavu `state.ready_to_fill`, `state.filling` nebo `state.emptying`. Stiskuntí tlačítka vypuštění tanku `TNK01_BTN03` pak vyvolá alarm pokud je tank TNK01 ve stavu `state.ready_to_fill`, `state.filling` nebo `state.running`.

## 7.5 Máčení ječmene

Po otevření výstupního ventilu tanku TNK01 `TNK01_V_OUT` se ječmen přesune z tanku do potrubí mezi tankem čištění ječmene TNK 01 a tankem máčení ječmene TNK02. Ječmen se pak automaticky dostane skrze potrubí do TNK02. To je vyřešeno obdobným algoritmem který řídí přesun ječmene z dopravníkového pásu CB01 do tanku TNK01. Hlavním rozdílem je absence regulace příjmu ječmene do tanku TNK02. Pohon dopravníkového pásu CB01 lze kdykoli vypnout přepínačem `CB01_SW`, díky čemuž se zastaví inkrementace hladiny tanku TNK01. Oproti tomu hladina tanku TNK02 je inkrementována dokud se tank TNK01 zcela nevypustí a potrubí mezi tanky TNK01 a TNK02 nevyprázdní.

Dále je třeba zavolat další instanci funkčního bloku FB2 - Kontroler tanků, který se stará o ovládací sekci tanku TNK02. Složení alarmů je pak stejné jako v části čistění ječmene.

## 7.6 Přeprava na hvozd

Přepravu na hvozd tvoří celkem 4 dopravníkové pásy. Ovládání každého z těchto dopravníkových pásu je řešeno instancemi funkčního bloku FB1 - Kontroler dopravníkového pásu. Zároveň jsou pro každý z těchto dopravníkových pásů zavedeny alarmy *přetížení pásu* a *neefektivita pásu*.

## 7.7 Hvozd

Hvozd je část sladovny, ve které se suší naklíčený mokrý ječmen. V reálném prostředí trvá proces klíčení a sušení několik desítek dní. Z toho důvodu je třeba průběžně ječmen prohazovat. V simulaci sladovny je v datovém bloku DB2 specifikovaný potřebný čas sušení pomocí intervalu `Simulation.hvozd.process_time_min` až `Simulation.hvozd.process_time_max`. Prohazování nemá stanovený žádný minimální interval, nicméně maximální doba mezi dvěma prohazováními je specifikovaná v datovém bloku DB2 jako `Simulation.hvozd.turn_time_max`.

Mimo funkci prohazování je logika simulovaného hvozdu totožná s logikou tanků. A to včetně složení alarmů. Program prohazování spustí časovač pokud je hvozd ve stavu `state.running`. Pokud student, nebo jeho program inicializuje prohození ječmene kliknutím na tlačítko `HVOZD_BTN04`, časovač se resetuje. Pokud ale časovač dříve převáží hodnotu `Simulation.hvozd.turn_time_max`, vyvolá se alarm *nedostatečné prohazování*. Dalším řádným způsobem jak časovač zrušit je dokončení procesu sušení kliknutím na tlačítko `HVOZD_BTN02`.

Po úspěšném dokončení sušení je možné vypustit výsledný slad do sila. Běžně se mezi silem a hvozdem nachází odkličovací stroj. Vzhledem k zjednodušenému přístupu ke strojům by ale simulace odkličovacího stroje nijak výrazně nenavýšila vzdělávací hodnotu projektu.

# 8 Vizualizace

O vizualizační část projektu se stará WinCC prostředí ekosystému TIA Portal. Je zde vytvořena jedna obrazovka, která zobrazuje všechny komponenty simulované sladovny. Celý proces sladovny začíná v levém horním rohu obrazovky, kde se nachází tank příjmu ječmene. Z něho vede potrubí s ventilem V01, které vypouští ječmen na dopravníkový pás CB01. Dopravníkový pás CB01 dopraví ječmen do tanku čištění ječmene TNK01. Na výstupu tanku čištění ječmene TNK01 se pak nachází potrubí s výstupním ventilem V02 tanku čištění ječmene TNK01. Toto potrubí pak spojuje tanky čištění ječmene TNK01 a máčení ječmene TNK02. Po vypuštění tanku máčení ječmene TNK02 se namočený ječmen dostane na dopravníkový pás CB02, který je první ze čtyř dopravníkových pásů přepravujících namočený ječmen na hvozd.

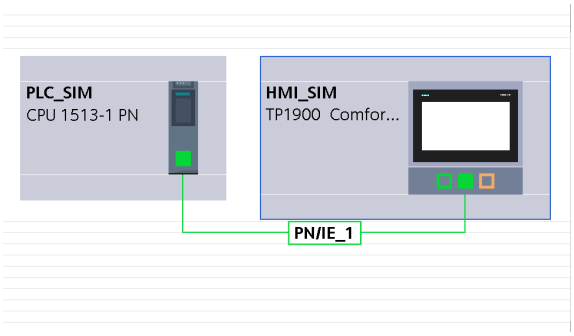
Ventily i dopravníkové pásy vizuálně reagují na aktuální stav programu. Pokud je hodnota tagu ventilu rovna 0, pak se ventil zobrazuje červeně. Pokud je jeho hodnota rovna 1, pak je zelený. Podobné chování je i u dopravníkových pásů. Pokud je hodnota tagu pohonu dopravníkového pásu rovna 1, tak se napravo od označení pásu zobrazí zelená šipka. Pokud se hodnota tagu pohonu dopravníkového pásu změní na 0, pak šipka zmizí.

## 8.1 Spojení s PLC

Stejně jako je při programování v SIMATIC STEP-7 potřeba definovat adresaci dat pomocí tabulek tagů, případně datových bloků, tak je ve WinCC potřeba definovat adresaci dat pomocí HMI tabulek tagů. HMI tabulky tagů fungují obdobně jako PLC tabulky tagů v SIMATIC STEP-7 s tím rozdílem, že lze u každého tagu zvolit, zda má být lokální, nebo spojený s PLC tagem.

Veškerá data, jako jsou například obrazovky, HMI tabulky tagů a alarmy, se stahují do paměti HMI panelu. Pro řízení zobrazení barvy ventilu je tedy potřeba mít vytvořený tag v paměti HMI panelu. Tento tag bude uchovávat hodnotu 0 nebo 1, tedy zda má být ventil zobrazovaný červeně nebo zeleně. Takový tag je výhodné spojit s tagem téhož ventilu, který již byl vytvořen v programu a jeho hodnota se nachází v paměti PLC procesoru. Tím, že zvolíme, že má být HMI tag spojený s PLC tagem docílíme toho, že pokud se hodnota tagu v paměti PLC procesoru změní, pak se změní i hodnota odpovídajícího tagu v paměti HMI panelu. Stejně tak pokud máme na obrazovce HMI panelu prvek, který umožňuje měnit hodnotu HMI tagu, pak se změna HMI tagu propíše i do odpovídajícího PLC tagu.

Pro správnou funkčnost je nutné, aby byl HMI panel s PLC propojený. To je realizované pomocí průmyslového ethernetu s využitím protokolu PROFINET. Obecné nastavení spojení PLC a HMI panelu se konfiguruje v SIMATIC STEP-7. Konkrétní nastavení tagů a alarmů se spravuje ve WinCC konkrétního panelu. Schéma propojení simulovaného PLC a HMI panelu je vidět na obrázku 8.1.




Obrázek 8.1: Schéma propojení PLC a HMI panelu

## 8.2 Alarmy

Mimo prvky sladovny se na obrazovce HMI panelu nachází tzv. *alarm view*. Jedná se o tabulku sloužící k zobrazovaní a správě alarmů. Alarmy se vytvářejí ve WinCC pomocí speciální tabulky tagů. Pro vytvoření alarmu je třeba vytvořit tag datového typu *integer*. Tím se v paměti HMI panelu alokuje 16 bitů, přičemž každý z bitů může být využit jako alarm. V projektu je vytvořeno celkem 5 tagů pojmenovaných jako AG01, AG02, AG03, AG04 a AG05. To vytváří celkem 80 možných diskrétních alarmů. V AG01 se nacházejí pouze alarmy pro dopravníkové pásy. Pro každý dopravníkový pás existují dva typy alarmů. Prvním typem je *přetížení pásu*, který je vyvolaný pokud je na dopravníkový pás aplikovaná větší hmotnost, než je maximální nosnost pásu specifikovaná v DB2. Druhým typem alarmu je *neefektivita pásu*. Tento alarm je vyvolán pokud je zapnut pohon dopravníkového pásu a po dobu specifikovanou v DB2 není zatížen žádnou hmotností. Alarmy dopravníkového pásu CB01 se nachází tedy v prvních dvou bitech tagu AG01, alarmy dopravníkového pásu CB02 jsou ve 3. a 4. bitu AG01, a tak dále.

Pro tanky existuje celkem 9 typů alarmů. *Přetečení tanku* se vyvolá, pokud je do tanku vloženo větší množství ječmene než jaká je maximální kapacita tanku definovaná v DB2. Jakmile se student, nebo jeho program pokusí spustit proces tanku i když je tank prázdný, pak se vyvolá alarm *spuštění prázdného tanku*. *Neoprávněné spuštění* tanku znamená, že se student, nebo jeho program pokusil spustit proces tanku když byl již proces dokončen, nebo když se tank vypouštěl. Alarm *neoprávněné přidání ječmene* je vyvolán pokud je do tanku přidán ječmen v průběhu běžícího procesu tanku, po dokončení procesu tanku, nebo během vypouštění tanku. *Neoprávněné zastavení* tanku je vyvoláno pokud se student, nebo jeho program pokusí zastavit proces tanku v době, kdy není spuštěn, tj. pokud je ve stavu, kdy je tank připraven k plnění, během plnění, po dokončení procesu tanku, nebo během vypouštění. *Brzké zastavení* a *příliš dlouhý chod* souvisí s časem, po který je proces tanku spuštěn. Pokud student, nebo jeho program zastaví proces příliš brzy, tj. konečná doba běhu procesu tanku je menší než definovaná minimální doba běhu procesu tanku v DB2, pak se vyvolá alarm *brzké zastavení* tanku. Pokud student, nebo jeho program nezastaví proces tanku včas, tj. doba běhu procesu tanku překročí maximální dobu běhu procesu tanku definovanou v DB2, pak se vyvolá alarm *příliš dlouhý chod* tanku. Posledním alarmem pro tanky je *neoprávněné vypuštění* tanku. Tento alarm je vyvolán pokud student, nebo jeho program spustí vypouštění tanku v době, kdy je tank prázdný, plní se nebo je spuštěn proces tanku. Seznam všech alarmů se nachází v následující tabulce.

Tabulka 8.1: Definované alarmy

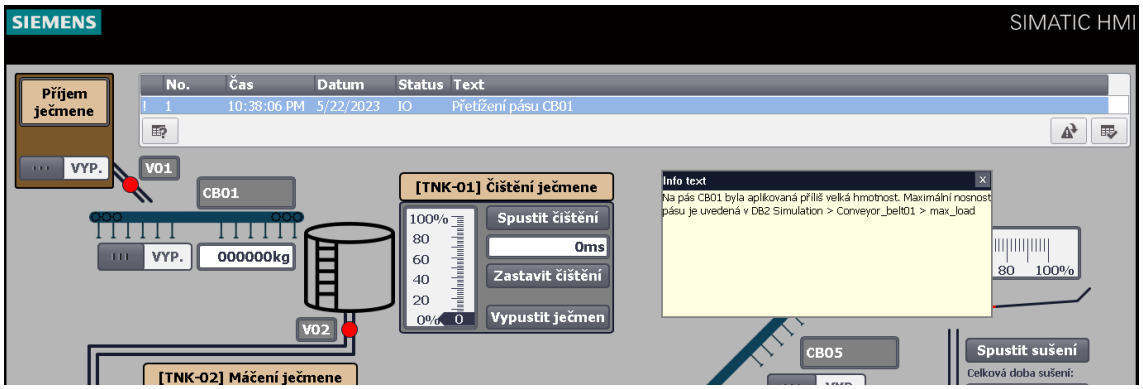
| Tag (adresa) | Trigger bit (adresa) | Popis |
|---|---|---|
| AG01 (%MW200) | AG01ERR0 (%M201.0) | Přetížení dopravníkového pásu CBO1 |
| | AG01ERR1 (%M201.1) | Neefektivita dopravníkového pásu CB01 |
| | AG01ERR2 (%M201.2) | Přetížení dopravníkového pásu CBO2 |
| | AG01ERR3 (%M201.3) | Neefektivita dopravníkového pásu CB02 |
| | AG01ERR4 (%M201.4) | Přetížení dopravníkového pásu CBO3 |
| | AG01ERR5 (%M201.5) | Neefektivita dopravníkového pásu CB03 |
| | AG01ERR6 (%M201.6) | Přetížení dopravníkového pásu CBO4 |
| | AG01ERR7 (%M201.7) | Neefektivita dopravníkového pásu CB04 |
| AG02 (%MW202) | AG02ERR0 (%M203.0) | Přetížení dopravníkového pásu CBO5 |
| | AG02ERR1 (%M203.1) | Neefektivita dopravníkového pásu CB05 |
| | AG02ERR2 (%M203.2) | Přetečení tanku TNK01 |
| | AG02ERR3 (%M203.3) | Tank TNK01 spuštěn prázdný |
| | AG02ERR4 (%M203.4) | Neoprávněné spuštění tanku TNK01 |
| | AG02ERR5 (%M203.5) | Neoprávněné přidání ječmene do tanku TNK01 |
| | AG02ERR6 (%M203.6) | Neoprávněné zastavení tanku TNK01 |
| | AG02ERR7 (%M203.7) | Brzké zastavení tanku TNK01 |
| AG03 (%MW204) | AG03ERR0 (%M205.0) | Příliš dlouhý chod tanku TNK01 |
| | AG03ERR1 (%M205.1) | Neoprávněné vypuštění tanku TNK01 |
| | AG03ERR2 (%M205.2) | Přetečení tanku TNK02 |
| | AG03ERR3 (%M205.3) | Tank TNK02 spuštěn prázdný |
| | AG03ERR4 (%M205.4) | Neoprávněné spuštění tanku TNK02 |

*Pokračování na další stránce*

Tabulka 8.1: Definované alarmy *(Pokračování)*

| Tag (adresa) | Trigger bit (adresa) | Popis |
|---|---|---|
| | AG03ERR5 (%M205.5) | Neoprávněné přidání ječmene do tanku TNK02 |
| | AG03ERR6 (%M205.6) | Neoprávněné zastavení tanku TNK02 |
| | AG03ERR7 (%M205.7) | Brzké zastavení tanku TNK02 |
| AG04 (%MW206) | AG04ERR0 (%M207.0) | Příliš dlouhý chod tanku TNK02 |
| | AG04ERR1 (%M207.1) | Neoprávněné vypuštění tanku TNK02 |

Po tom, co je vyvolán libovolný alarm, tak se v alarm view zobrazí časová známka a popis alarmu. Pro každý alarm je také sepsán detailní popis chyby která alarm vyvolala. Tento popis lze zobrazit kliknutím na ikonu v levém spodním rohu alarm view. Kliknutím se vyvolá nové okno, ve kterém se nachází text detailního popisu chyby.




Obrázek 8.2: Detailní popis alarmu

Program při každém vyvolání alarmu uvede sladovnu do výchozího stavu. Pro zajištění dostatečného času pro analýzu chyby, která vyvolala alarm lze záznam alarmu z alarm view smazat pouze manuálním odbavením. Odbavení alarmu se provádí kliknutím na tlačítko v pravém spodním rohu alarm view.

# Závěr

Bakalářská práce se zaměřuje na návrh a implementaci virtuální sladovny pivovaru pomocí softwaru Totally Integrated Automation Portal (TIA Portal) od společnosti Siemens. TIA Portal je inženýrský ekosystém pro automatizaci, který poskytuje nástroje pro konfiguraci, programování, diagnostiku a údržbu PLC systémů, HMI panelů a pohonů.

Přestože existuje několik dalších populárních platforem pro automatizaci, jako je Allen-Bradley RSLogix, Beckhoff TwinCAT, Mitsubishi GX Works nebo Omron CX-One, rozhodnutí zvolit TIA Portal pro tento konkrétní projekt virtuální sladovny pivovaru bylo založeno na různých faktorech a zvažování jak obecných, tak i osobních. Hlavní nevýhodou volby TIA portálu je jeho cena. Ačkoli je konkurenceschopná s uvedenými alternativami, tak by implementace pro vzdělávací účely mohla být považovaná za velmi nákladnou. Ačkoli existuje mnoho softwaru pro programování a simulace PLC procesorů, tak žádný z nich nenabízí jednoduchost a uživatelskou přívětivost při tvorbě vizualizací, které jsou pro tento projekt zásadní.

Pro účely laboratorních úloh byl vytvořen komplexní projekt sladovny se simulovanými objekty, jako jsou ventily, pásy a tanky. Pro tyto objekty byla implementována realistická logika chování, která zajišťuje jejich správné fungování v simulovaném prostředí. Kromě toho byly naprogramovány události, které přidávají další úroveň realističnosti, například hlášení o přetížení dopravníkového pásu.

Struktura projektu virtuální sladovny je rozdělena do čtyř částí: programová část, vizualizační část, datová část a simulační část. Programová část obsahuje veškerou logiku projektu a je rozdělena do sekcí ovládání, realističnosti a studentské sekce. Vizualizační část je zajištěna pomocí prostředí WinCC, které poskytuje obrazovku HMI panelu sladovny a umožňuje ovládání a monitorování virtuálních zařízení. Datová a simulační část obsahují tabulky tagů a datové bloky, které definují vstupy, výstupy a simulované podmínky sladovny.

Cílem laboratorních úloh je přiblížit studentům základní prvky návrhu automatizačních systémů a naučit je, jak takové systémy programovat. Studenti se seznámí se základy adresace, datovými typy, logickými instrukcemi, časovači, čítači a dalšími pokročilými funkcemi TIA Portalu. Během výuky pracují s organizačními bloky, funkcemi, funkčními bloky a datovými bloky, což jsou základní stavební prvky pro programování logiky automatizovaného systému.

Výuka v prostředí virtuální sladovny pivovaru umožňuje studentům praktickou aplikaci teoretických znalostí a získání cenného know-how v oblasti automatizace. Tímto způsobem se studenti aktivně zapojují do procesu učení a mají možnost experimentovat s různými scénáři a nastaveními. Virtuální prostředí umožňuje opakované testování a ladění programů bez nutnosti fyzického zařízení, což je výhodné z hlediska času a nákladů.

# Bibliografie


1. SIEMENS. *SIMATIC S7-300*. [B.r.]. Dostupné také z: https://assets.new.siemens.com/siemens/assets/api/uuid:ccf8b374a4e750320264ac2e0f38d1663961d89c/width:1125/quality:high/p-st70-xx-05782v-bg.jpg.
2. KULISZ, Józef; CHMIEL, Mirosław; KRZYZYK, Adrian; ROSÓŁ, Marcin. A hardware implementation of arithmetic operations for an FPGA-based programmable logic controller. *IFAC-PapersOnLine*. 2015, roč. 48, č. 4, s. 460–465.
3. NIEDERMAIER, Matthias; MALCHOW, Jan-Ole; FISCHER, Florian; MARZIN, Daniel; MERLI, Dominik; ROTH, Volker; BODISCO, Alexander von. *12th USENIX Workshop on Offensive Technologies (WOOT 18)*. You Snooze, You Lose: Measuring PLC Cycle Times under Attacks. Baltimore, MD: USENIX Association, 2018. Dostupné také z: https://www.usenix.org/conference/woot18/presentation/niedermaier.
4. SIEMENS. *SIMATIC S7-1200 Easy Book*. 2015. 01/2015. A5E02486774-AG.
5. SIEMENS. *STRAIN Digital Industry Academy: TIA-PRO2*. SIMATIC programming 2 in the TIA Portal. 17.00.00. vyd. Siemens AG, 2021.
6. SIEMENS. *Programming Guideline for S7-1200/1500*. Siemens AG, 2018. Ver. 1.6. Entry ID: 81318674.
7. WODOSLAWSKY, Ted. *PLC PROGRAMMING THEN AND NOW: THE HISTORY OF PLC'S*. 2019. Dostupné také z: https://www.c3controls.com/white-paper/history-of-programmable-logic-controllers/.
8. KŘENA, Michal. Historie, současnost a budoucnost programovatelných automatů Modicon. *AUTOMA*. 2015, roč. 21, č. 2-2015, s. 32–33. ISSN 1210-9592.
9. COMMISSION, International Electrotechnical. *Programmable controllers - Part 3: Programming languages*. 2013. Ver. 3.0. Tech. zpr. IEC 61131-3:2013.
10. *Programovací jazyky pro PLC* [Web Page]. Střední škola - Centrum odborné přípravy technické Kroměříž, 2019-11. AUT - Automatizace. Dostupné také z: https://coptel.cz/mod/page/view.php?id=6737.
11. BARESI, Luciano; CARMELI, Stefania; MONTI, Antonello; PEZZÈ, Mauro. PLC programming languages: A formal approach. *Proc. Autom.* 1998, roč. 98.
12. MASLAR, Mark. PLC standard programming languages: IEC 1131-3. In: *Conference Record of 1996 Annual Pulp and Paper Industry Technical Conference*. IEEE, 1996, s. 26–31.
13. EDER, Johann; LEHMANN, Marek. Uniform access to data in workflows. In: *E-Commerce and Web Technologies: 5th International Conference, EC-Web 2004, Zaragoza, Spain, August 31-September 3, 2004. Proceedings 5*. Springer, 2004, s. 66–75.
14. PLCDEV. *Step 7 Elementary Data Types*. [B.r.]. Dostupné také z: http://www.plcdev.com/step_7_elementary_data_types.
15. TIŠNOVSKÝ, Pavel. Nevolantní paměti. *Co se děje v počítači*. 2008. Dostupné také z: https://www.root.cz/clanky/nevolatilni-pameti/.
16. ARMENTA, Antonio. Siemens SIMATIC PLCs - Hardware History. 2022. Dostupné také z: https://control.com/technical-articles/siemens-simatic-plcs-hardware-history/.

17. SIEMENS. *PLC Programming with SIMATIC STEP7*. [B.r.]. Dostupné také z: https://www.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal/software/step7-tia-portal.html.

18. SIEMENS. *HMI visualization with SIMATIC WinCC*. [B.r.]. Dostupné také z: https://www.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal/software/simatic-wincc-tia-portal.html.

19. SIEMENS. *SIMATIC S7-1500 S7-PLCSIM Advanced*. 2016. 09/2016. Dostupné také z: https://cache.industry.siemens.com/dl/files/153/109739153/att_895955/v1/s7-plcsim_advanced_function_manual_en-US_en-US.pdf. A5E37039512-AA.

# A Laboratorní úloha č. 1 Seznámení s prostředím

## Cíle úlohy

1. Import a otevření projektu, nastavení simulace
2. Vytvoření tabulky tagů, vytvoření datového bloku
3. Vytvoření prvního organizačního bloku
4. Seznámení s bitovými instrukčními bloky

## Zadání úlohy

**Automatizace příjmu ječmene:** Navrhněte program, který bude zajišťovat to, že přepínač ventilu tanku příjmu ječmene TNKPJ_SW bude zároveň s otevíráním a zavíráním ventilu tanku příjmu ječmene i zapínat a vypínat pohon dopravníkového pásu CB01.

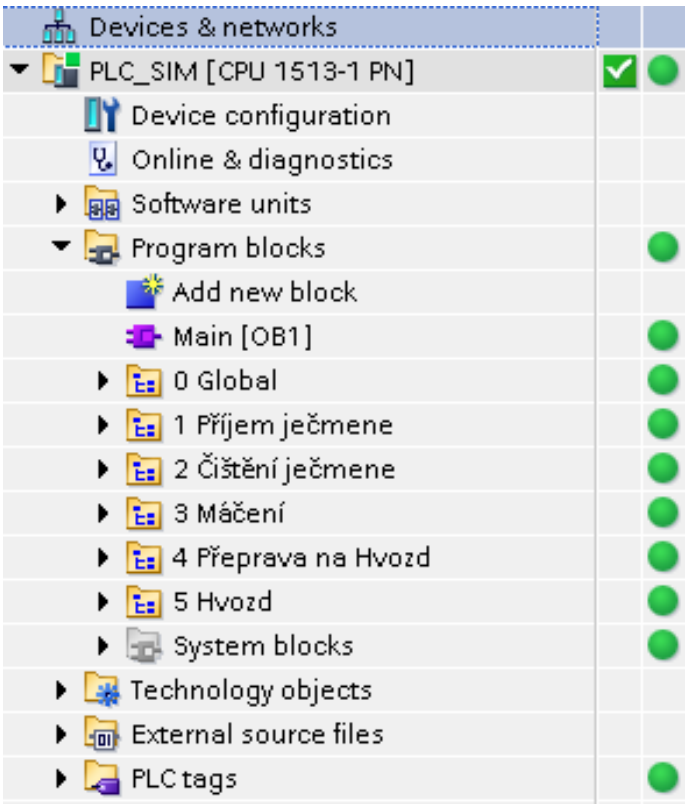
## Teoretický základ

Po spuštění TIA Portalu je třeba zkontrolovat, zda je nastavené zobrazení projektu. V levém spodním rohu obrazovky se nachází šipka, vedle které je napsáno buď "Portal view", nebo "Project view". Pokud je zde napsáno "Project view", tak je potřeba na tento text kliknout.

Po nastavení zobrazení projektu otevřeme projekt: *Project -> Open . . . -> Browse*. Zadáme cestu k souboru *project.zap17* a klikneme na *Open*. V okně pro výběr cílového adresáře vybereme kam se má projekt otevřít (např. *C://project/*).

Po otevření projektu nastavíme simulaci PLC procesoru. V programu S7-PLCSIM Advanced vytvoříme novou instanci procesoru. Ke komunikaci použijeme PLCSIM Softbus, tj. přepínač *Online Access* bude v poloze vlevo. Povinným polem je *Instance name* které může být libovolne (např. project_cpu). Kliknutím na tlačítko Start se vytvoří nová instance simulace procesoru.

Po vytvoření instance procesoru spojíme procesor s TIA Portalem. V levé části TIA portálu - *Project tree* - otevřeme *Devices & Networks*. Označíme symbol PLC procesoru a dolní část obrazovky přepneme na záložku *Properties*. Zkontrolujeme, zda se shoduje IP adresa (*PROFINET interface X1 -> Ethernet addresses -> IPv4 -> Set IP address in the project*) s IP adresou simulovaného procesoru. Pokud se neshodují, pak je třeba přepsat IP adresu v TIA Portalu. Nakonec stiskneme tlačítko *Go online* v horní liště. V seznamu zařízení, ke kterým se chceme připojit zvolíme pouze PLC_SIM - CPU 1513-1 PN. Po úspěšném přepnutí do online stavu, a tedy i úspěšném propojení procesoru a TIA Portalu, se objeví v hierarchii projektu zelená indikace.




Obrázek A.1: Online stav

### Vytvoření tabulky tagů a vytvoření datového bloku

**Vytvoření tabulky tagů:** Project tree -> PLC_SIM -> PLC Tag -> Add new tag table.

**Vytvoření datového bloku:** Project tree -> PLC_SIM -> Program blocs -> Add new block -> Data block.

### Tvorba organizačních bloků

Project tree -> PLC_SIM -> Program blocs -> Add new block -> Organization block.

### Bitové instrukční bloky

Mezi základní bitové instrukční bloky patří: Normally open contact, Normally closed contact, Assignment, Reset output, Set output.

| | | |
|---|---|---|
| Normally open contact | ... | Propouští signál pokud je hodnota přiřazeného bitu 1 |
| Normally closed contact | ... | Propouští signál pokud je hodnota přiřazeného bitu 0 |

| | | |
|---|---|---|
| Assignment | ... | Dokud je k dispozici signál, nastaví přiřazený bit na hodnotu 1 |
| Set output | ... | Nastaví přiřazený bit na hodnotu 1 |
| Ret output | ... | Nastaví přiřazený bit na hodnotu 0 |

Pomocí těchto instrukčních bloků již lze naprogramovat operace typu AND a OR:


Obrázek A.2: Operace AND

Na obrázku A.2 je vidět program operace AND. *lab1.Output* má hodnotu 1 pokud mají hodnotu 1 *lab1.input1* i *lab1.input1*.


Obrázek A.3: Operace OR

Program operace OR udává, že *lab1.Output* má hodnotu 1 pokud má hodnotu 1 *lab1.input1* nebo *lab1.input2*.

# B Laboratorní úloha č. 2 Pokročilé instrukční bloky

## Cíle úlohy

1. Operace s číselnými datovými typy
2. Operace s časovými datovými typy

## Zadání úlohy

**Dokončení automatizace příjmu ječmene:** Navrhněte program, který bude plně automatizovat příjem ječmene. Přepínačem TNKPJ_SW se otevře ventil a automaticky se spustí pohon pásu CB01. Jakmile bude tank TNK01 zcela naplněn, pohon CB01 se vypne a ventil TNKPJ_V_OUT se zavře.

**Bonus č. 1:** Optimalizujte program tak, aby po naplnění TNK01 nezbyl na CB01 žádný ječmen.

**Bonus č. 2:** Zacyklete program automatizace tak, aby se po vypuštění TNK01 opět spustil.

## Teoretický základ

### Instrukční bloky pro číselné datové typy

**Matematické instrukce:** Pro práci s číselnými datovými jsou definované matematické instrukce Add, Sub, Mul, Div, Mod, . . . Pro pokročilejší funkce je dostupná také instrukce Calculate, do které lze zapisovat předpisy funkcí. Zásadní limitace je v tom, že je nutné aby všechny parametry těchto instrukcí byly stejného datového typu.

**Move:** Instrukce Move zkopíruje hodnotu vstupního tagu a vloží ji do výstupního tagu.

**Comparator:** Instrukce typu Comparator slouží pro vzájemné porovnání dvou hodnot. Jsou dostupné instrukce, které propustí signál pokud jsou dvě hodnoty stejné, pokud se liší, pokud je jedna menší než druhá, . . .

### Instrukční bloky pro časové datové typy

**TON, TOF časovače:** Instrukční bloky časovačů vyžadují při svém použití instanci. Může se jednat jak o single-instanci, tak i o multi-instanci. TON propustí signál pokud je na vstupu TON instrukce přítomný signál po dobu stanovenou tagem PT v instanci TON instrukce. TOF generuje signál na výstupu TOF instrukce po detekci sestupné hrany vstupního signálu. Výstupní signál je generován po dobu stanovenou tagem PT v instanci TOF instrukce.

# C Laboratorní úloha č. 3 Strukturalizace programu

## Cíle úlohy

1. Vytvoření první funkce
2. Vytvoření prvního funkčního bloku

## Zadání úlohy

**Automatizace přepravy na hvozd:** Pomocí vhodné volby programových bloků navrhněte minimální program pro automatizaci skupiny dopravníkových pásu v sekci přepravy na hvozd.

**Bonus č. 1:** Optimalizujte program tak, aby po naplnění hvozdu nezbyl na žádném dopravníkovém pásu žádný ječmen.

**Bonus č. 2:** Optimalizujte program tak, aby se po naplnění hvozdu seskupoval ječmen nejdříve na posledním dopravníkovém pásu. Po zatížení posledního pásu na jeho maximální nosnost se bude seskupovat ječmen na předposledním pásu, a tak dále.

## Teoretický základ

### Funkce

Kromě organizačních bloků lze program ukládat do bloku funkce (FC). Na rozdíl od organizačních bloků nemá tento typ programových bloků dedikovanou paměť. Z toho důvodu je možné používat pouze dočasné nebo globální proměnné. Funkce se využívají při strukturovaném programování na popis a řízení jednodušších procesů. Například matematický výpočet, řízení otevírání a zavírání regulačního ventilu, řízení startérů. Veškeré parametry musí být funkci přiřazeny společně s voláním funkce.

**Vytvoření funkce:** Project tree -> PLC_SIM -> Program blocs -> Add new block -> Function.

**Volání funkce:** Přetáhnutím myší názvu funkce z Project tree do kódu volajícího programového bloku (Drag and Drop).

### Funkční blok

Na rozdíl od funkce má funkční blok (FB) alokovanou svou lokální paměť, která musí být inicializována při svém volaní. Jeden segment dat lokální paměti pro funkční blok nazýváme *instance*, která je vytvářena automaticky operačním systémem při kompilaci a stažení programu do PLC.

**Vytvoření funkčního bloku:** Project tree -> PLC_SIM -> Program blocs -> Add new block -> Function block.

**Volání funkčního bloku:** Přetáhnutím myší názvu funkce z Project tree do kódu volajícího programového bloku (Drag and Drop).

### Minimální program

Minimálním programem se rozumí program, který zabírá v paměti co nejméně místa. Pro zobrazeni obsazenosti paměti klikněte na *Tools -> Resources* (v online režimu). Pro lepší přehlednost zpracovávejte tuto úlohu v samostatném organizačním bloku. Velikost tohoto programu je pak dána součtem velikosti tohoto organizačního bloku, všech programových bloků, které byly v tomto organizačním bloku volány a všech datových bloků.