VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

AUTOMATICKÉ GENEROVÁNÍ PLC PROGRAMU POMOCÍ TIA PORTAL OPENNESS
AUTOMATIC CREATION OF PLC PROGRAM USING TIA PORTAL OPENNESS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Pavel Janeček
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jakub Arm
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Pavel Janeček ***ID:*** 186098
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Automatické generování PLC programu pomocí TIA Portal Openness

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvoření programu pro PLC S7-1500 a vizualizace pro model "Kuličky" automatizovaně pomocí nástroje TIA Portal Openness dle XML předpisu.

1. Seznamte se s nástrojem TIA Portal Openness.
2. Vytvořte PLC program s vizualizací v prostředí TIA Portal pro ovládání modelu "Kuličky" a rozdělte je na jednotlivé části.
3. Nadefinujte vhodně schéma XML souboru sloužící jako předpis pro generování PLC programu pomocí jednotlivých bloků co nejuniverzálněji.
4. Navrhněte a realizujte desktopovou aplikaci, která generuje PLC program dle XML předpisu.
5. Zdokumentujte postup generování a konfiguraci PLC programu pomocí TIA Openness.

DOPORUČENÁ LITERATURA:

Firemní dokumentace Siemens pro TIA Portal Openness.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: Ing. Jakub Arm
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Tato bakalářská práce se zabývá automatickým generováním PLC programu s vizualizací pro model "Kuličky" pomoci TIA Portal Openness. Pro tento účel byla napsána desktopová aplikace "TIAOpennessGenerator" v programovacím jazyku C#, která generuje PLC program podle vhodně navrženého schématu XML souboru, který slouží jako předpis. Schéma i aplikace jsou napsány takovým způsobem, aby generování bylo co nejuniverzálnější.

KLÍČOVÁ SLOVA

TIA Portal Openness, TIA Portal, PLC Simatic S7-1500, XML

ABSTRACT

This bachelor's thesis deals with automatic generation of PLC program with visualization for model 'Kuličky' through TIA Portal Openness. For this purpose a desktop application "TIAOpennessGenerator", by using programming language C#, was developed. This application generates PLC program according to suitably designed XML schema which works as a "recipe". Both schema and application are written in order to make data generating as universal as possible.

KEYWORDS

TIA Portal Openness, TIA Portal, PLC Simatic S7-1500, XML

JANEČEK, Pavel. *Automatické generování PLC programu pomocí TIA Portal Openness*. Brno, 2018, 62 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Jakub Arm

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Automatické generování PLC programu pomocí TIA Portal Openness“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Jakubu Armovi za odborné vedení, konzultace, věnovaný čas, trpělivost, ochotu a podnětné návrhy k práci.

Brno ....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

V současné době jde automatizační průmysl neustále kupředu a jsou kladeny stále vyšší nároky na efektivitu a rychlost uvádění výrobních linek do provozu. V tomto rozvíjejícím se průmyslu má pomoci TIA Portal Openness, jenž umožňuje efektivní spravování programovatelných automatů a díky němuž je uvádění výrobních linek do provozu rychlejší.

Obsah této práce je věnován problematice programovatelných automatů, kterými se řídí výrobní linky a prostředí TIA Portal, pomocí něhož se tyto automaty programují. Převážná část se pak věnuje TIA Portal Openness, kde je přímo vysvětleno, jaký formát má soubor určený pro generování programů do programovatelných automatů. Dále je zmíněno v jakém programovacím jazyku se programuje TIA Portal Openness a jsou popsány a ukázány funkce, kterými se díky TIA Portal Openness generuje program.

Cílem této práce je generování PLC programu s vizualizací pomocí nástroje TIA Portal Openness pro laboratorní model "Kuličky". Pro generování programu pomocí TIA Portal Openness je nutné nejdříve navrhnout vhodné schéma XML souboru a vytvořit desktopovou aplikaci v programovacím jazyku C#. XML soubor i aplikace mají být napsány tak, aby co nejuniverzálněji generovaly PLC program s vizualizací pro libovolný počet válců z modelu "Kuličky".

1 PROGRAMOVATELNÉ AUTOMATY

Programovatelný logický automat, dále jen PLC z anglické zkratky Programmable Logic Controller, je určen pro řízení automatizovaných procesů ve výrobě, čímž můžeme rozumět řízení výrobních linek. *PLC zpracovává a vyhodnocuje signály přicházející z technologického procesu na jeho vstupy (vstupní signály) a na základě naprogramovaných sekvenčních logických a časových funkcí vydává na své výstupy (výstupní signály) povely, kterými se ovládají stykače motorů, klapky, ventily, topná tělesa apod. [2].* Jejich funkčnost je zajištěna cyklickým opakováním programu, což zajišťuje operační systém PLC. Pomocí programovatelných automatů se již řeší téměř veškeré průmyslově řízené aplikace. [1] [2]

Tyto automaty nahradily převážně reléové systémy a jejich použití značně zrychluje proces, šetří náklady a čas. Největší výhodou PLC je možnost diagnostiky, modifikace stávajících úloh a především rychlá realizace algoritmu. U reléového systému byly úpravy velmi složité, jelikož jejich algoritmus byl dán složitým fyzickým propojením, což komplikovalo i určování chyb. Dalšími nedostatky reléových systémů je zabírající místo pro uložení řídící logiky a poruchovost. Reléové systémy byly uloženy v obrovských halách a pro jejich velkou poruchovost bylo zapotřebí několik servisních techniků. PLC systémy je možno uložit v prostoru srovnatelného s malou ledničkou, kdy jejich obsluhu zvládá jeden vyučený technik. [2]

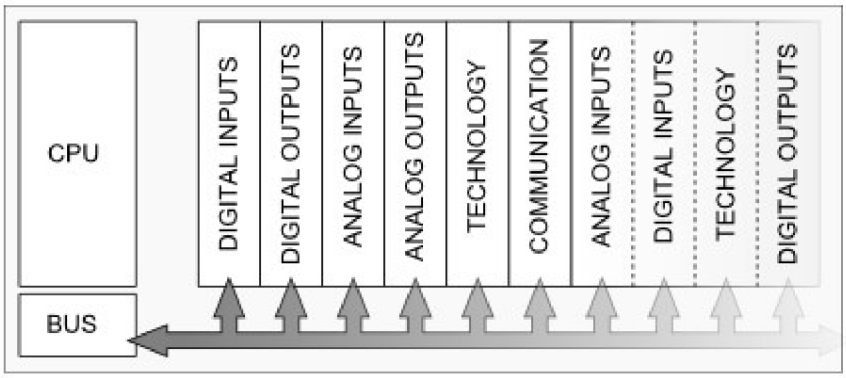
Podle druhu konstrukce můžeme rozdělit automaty na modulární a kompaktní.

1.1 Modulární automaty

Modulární automat, ostatně jako každý jiný PLC musí obsahovat centrální procesorovou jednotku, dále jen CPU, ke které se připojují ostatní moduly. Jejich největší výhodou je tedy volnost a velká rozmanitost. To je zapříčiněno možností do soustavy zasouvat různé moduly, jako jsou např. komunikační moduly a analogové nebo digitální moduly, a to s různorodou kombinací počtu vstupů a výstupů. Příklad zapojení modulárního automatu je zobrazen na Obr. 1.1. Díky této pružnosti propojení CPU s ostatními moduly se tento typ automatu využívá ve většině procesů. [3] [4]

1.2 Kompaktní automaty

Jedná se o programovatelný automat, který je konstruován jako jeden modul obsahující jak CPU, tak i předem nadefinovaný počet analogových nebo digitálních vstupů a výstupů. [4]



Obr. 1.1: Příklad uspořádání modulárního automatu [4].

Jejich výhodou je nižší pořizovací cena než u modulárních automatů a jejich vysoká rychlost přenosů signálů ze vstupů a zpátky na výstupy. Vyšší rychlost umožňuje skutečnost, že vstupy a výstupy jsou ve stejném modulu jako CPU, a proto se signál nezpožďuje průchodem přes sběrnice.[4]

Nevýhodou je, že často nedisponují možností připojit další moduly, a tak je jejich využití omezeno pouze pro malé výroby a podniky. Pokud mají možnost připojit moduly, tak jejich počet je často omezen typem automatu. K těmto automatům lze připojit maximálně desítky modulů, zatímco u modulárních automatů to mohou být až stovky. [3] [4]

1.3 Siemens Simatic S7-1500

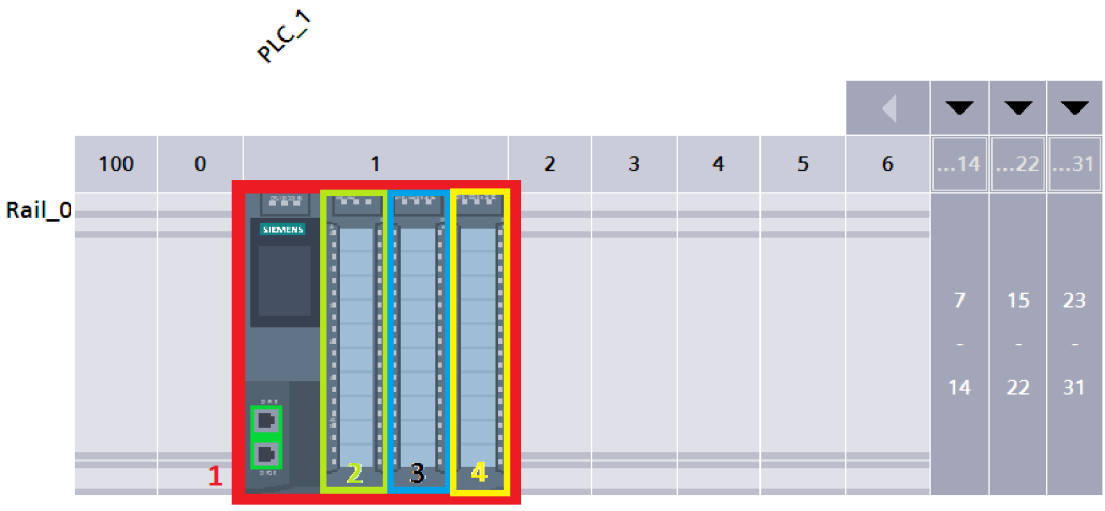
Řídící systém Simatic S7-1500 je zkonstruován s ohledem na výkonnost a efektivitu. Pro tyto systémy jsou výrazně rozšířeny technologické funkce a zdokonaleny funkce zabezpečení a ochrany dat a je zvýšen výkon jádra systému. Pro zvýšení efektivity byla vyvinuta řada vylepšení, zejména v oblastech konstrukce, ovládání a diagnostiky systému přičemž byl řídící systém S7-1500 začleněn do vývojového prostředí TIA Portal. [5]

Vysoký výkon systému S7-1500 umožňuje dosáhnout rychlého zpracování signálů, mimořádně krátkých dob odezvy a vysoké kvality řízení. Systém Simatic S7-1500 má nebývale rychlou systémovou sběrnici s velkou rychlostí přenosu dat a efektivním komunikačním protokolem. [5]

Řídicí jednotky Simatic S7-1500 jsou vybavené displejem, který umožňuje pohodlnou obsluhu a zobrazení podrobné textové informace poskytující zcela transparentní pohled na zařízení. [5]

V této práci bylo použito právě PLC řady Simatic s7-1500 s CPU 1512C-1 PN,

který je zobrazen na Obr. 1.2. CPU je ohraničen červeným rámečkem. Toto CPU obsahuje celkem 3 moduly. První, na obrázku označený číslem 2, má pět analogových vstupů a dva analogové výstupy. Druhý a třetí modul je na obrázku označený číslem 3 a 4. Tyto moduly jsou shodné a mají 16 digitálních vstupů i výstupů. Celkem tedy toto CPU disponuje již v základu 32 digitálními vstupy a výstupy.



Obr. 1.2: PLC Simatic s7-1500 s CPU 1512C-1 PN.

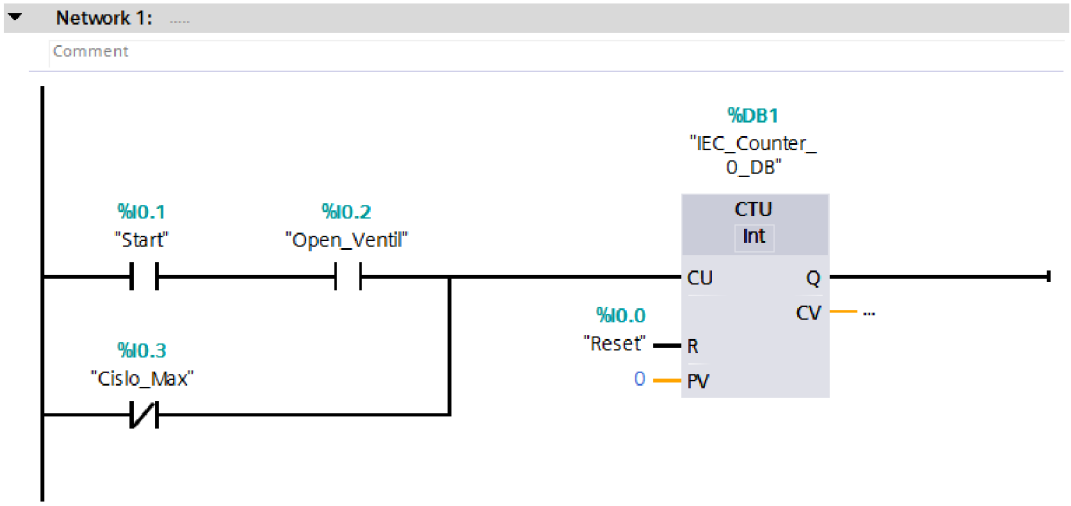
1.4 Programovací jazyky

Každý výrobce má pro jejich vývojové prostředí své programovací jazyky určené pro programování PLC, které jsou sice podobné, ale v žádném případě nejsou stejné a tudíž nejsou ani přenositelné mezi jednotlivými vývojovými prostředími od různých výrobců. Existuje však mezinárodní norma IEC 61 131-3, která sjednocuje typy jazyků do čtyř základních typů s přesně definovanou syntaxí. Podle této normy se jazyky tedy dělí na LD - Ladder Diagram, FBD - Function Block Diagram, IL - Instruction list a ST - Structured Text. Jako další lze uvést programovací jazyk SFC - Sequential Function Chart. Další dělení se používá dle formy jazyka a dělí se na grafické a textové. Mezi grafické jazyky patří LD, FBD, SFC. Mezi textové pak patří jazyky ST a IL. [6] [7]

1.4.1 Jazyk LD - Ladder Diagram

Jazyk příčkového diagramu spadá mezi grafické programovací jazyky. LD je založen na principu reléové logiky. Program se píše na jednotlivé příčky, přidáváním

prvků z instrukční sady, jako jsou spínací a rozpínací kontakty, funkce a funkční bloky. Tento jazyk je pro svou jednoduchost vhodný pro začátečníky. Při složitějších aplikacích se však stává nevyhovujícím a nepřehledným. [1] [6]



Obr. 1.3: Ukázka jazyka LAD z vývojového prostředí TIA Portal.

Na Obr. 1.3 je ukázka jazyka LAD reprezentující normovaný jazyk LD, vytvořeného v prostředí TIA Portal od firmy Siemens. Na tomto obrázku jsou zobrazeny spínací a rozpínací kontakty a pak funkční blok pro čítač čítající vpřed. Z obrázku je patrné, že se jedná o velmi přehledný programovací jazyk.

1.4.2 Jazyk IL - Instruction list

Tento jazyk patří do skupiny textových jazyků. Jeho syntaxe může být podobná jazyku assembler. Je tvořen seznamem instrukcí, které se vždy píší na samostatný řádek. Instrukce obsahují operátor, v případě potřeby modifikátor a jeden nebo více operandů. Kód se dělí do jednotlivých networků podle jejich funkce. Toto dělení zajišťuje alespoň malou přehlednost programu. Pro složité aplikace je ale opět nevyhovující, protože programátor je nucen znát velké množství instrukcí a program je pak příliš rozsáhlý a nepřehledný. [1] [6]

Na Obr. 1.4 je ukázka jazyka IL z vývojového prostředí Simatic Manager, z kterého je patrné, že se jednotlivé instrukce píší vždy na samostatný řádek a že kód je dělen do networků. Pro složité funkce je tento jazyk velmi nevhodný a stává se velmi nepřehledným.

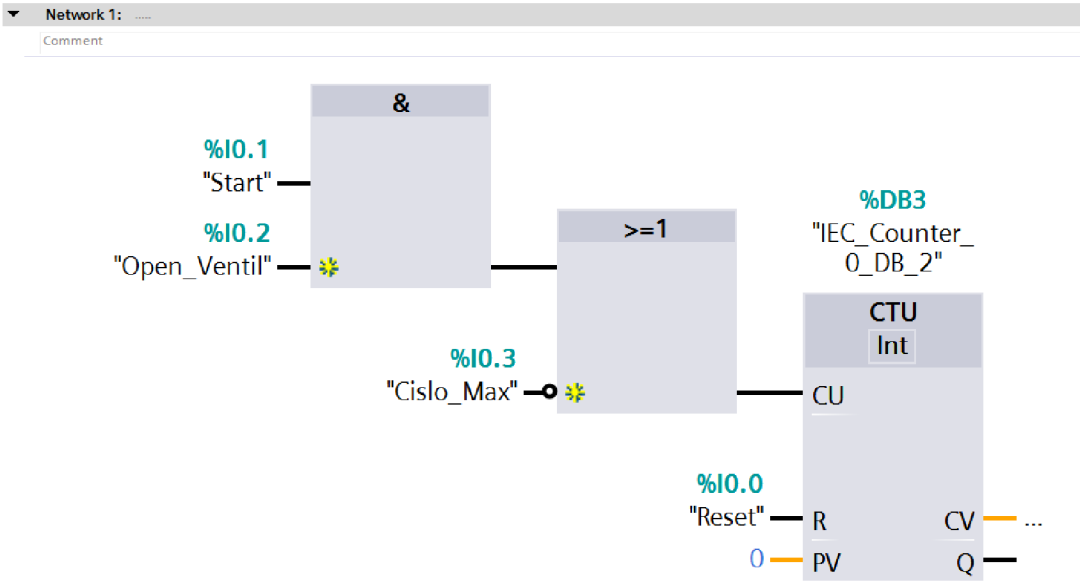
```
Network: 6                          Network: 7

A     T      1                      A     T      2
S     Q     40.1                    R     Q     40.1
L     S5T#1S                       R     M     50.1
SD    T      2                     R     M     50.0
```

Obr. 1.4: Ukázka jazyka IL z vývojového prostředí Simatic Manager.

1.4.3 Jazyk FBD - Function Block Diagram

V českém překladu funkční blokové schéma. Jak název říká, jedná se o jazyk tvořený pomocí funkčních bloků, kdy jednotlivé operace jsou prováděny zapojováním bloků v paralelních či sériových kombinacích. Funkční bloky mohou obsahovat logické, aritmetické operace či funkce. Z tohoto popisu vyplývá, že se jedná o grafický typ jazyka a v principu je podobný jazyku LD. [1]



Obr. 1.5: Ukázka jazyka FBD z vývojového prostředí TIA Portal.

Na Obr. 1.5 je ukázka jazyka FBD, jenž znázorňuje stejný program jako v příkladu na Obr. 1.3. Program vypadá podobně a je opět přehledný.

1.4.4 Jazyk ST - Structured Text

Strukturovaný text, jak již název napovídá, patří mezi textové programovací jazyky. Syntaxe je podobně jako např. v jazyce C dána definovanými výrazy a příkazy. Oproti jazyku IL nemusí programátor znát velké množství instrukcí a program je i přehlednější. Jazyk Structured text je pokládán za výkonný programovací jazyk. Z důvodu přehlednosti je využíván pro složitější programy. [1] [6]

```
1
2
3  IF "Povoleni" THEN
4      // Statement section IF
5      "IEC_Counter_0_DB_1".CTU(CU := "Start"
6                               AND "Open_Ventil"
7                               OR NOT "Cislo_Max",
8                               R := "Reset",
9                               PV := 0);
10                              //Q=>_bool_out_,
11                              //CV=>_int_out_);
12
13
14 END_IF;
```

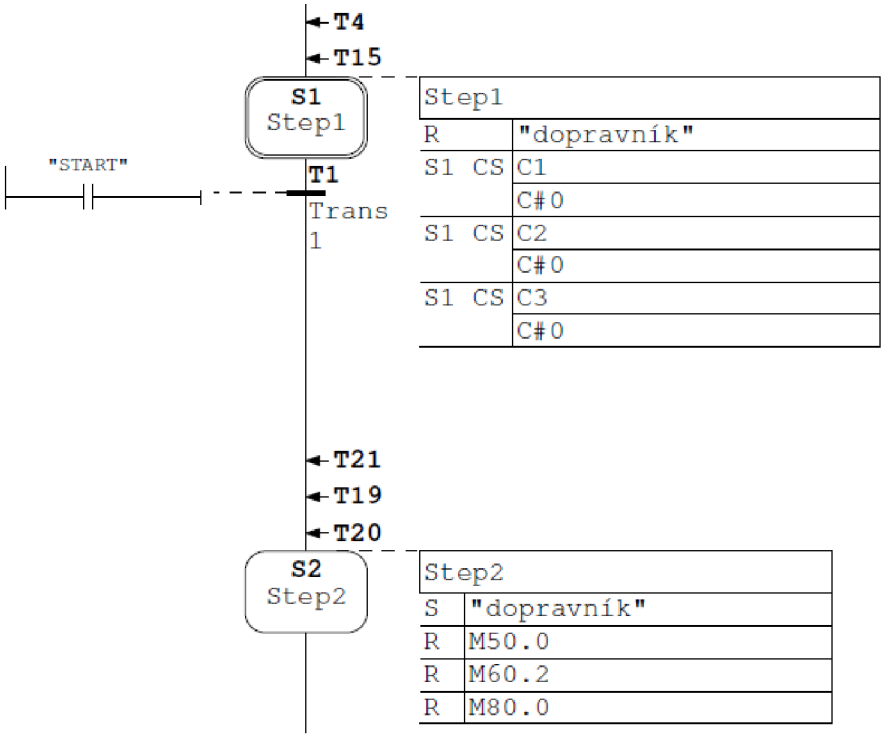
Obr. 1.6: Ukázka jazyka SCL z vývojového prostředí TIA Portal.

Na Obr. 1.6 je ukázka jazyka SCL z vývojového prostředí TIA Portal od firmy Siemens, který je postaven na základě normy jazyka ST. Opět se jedná o stejný program jako na obrázcích 1.3 a 1.5. Tento program sice není vizuálně tak přehledný, ale pro složitější programy, kdy by byl zápis při použití jazyka LD či FBD velmi složitý, např. pro psaní velkého množství výpočtů, je tento jazyk výhodnější.

1.4.5 Jazyk SFC - Sequential Function Chart

V českém překladu sekvenční funkční diagram, častěji nazýván GRAPH. I z tohoto označení plyne, že se jedná o grafický programovací jazyk. Tento program je vhodný pro programování stavových automatů, protože tak se i tento jazyk chová. Program je složen z tzv. startujícího bloku, mezibloků, které se mohou dělit do několika větví, z konečného bloku, kterých může být více, a z přechodů mezi jednotlivými bloky. Bloky pak reprezentují stav, ve kterém se automat nachází, a ke každému bloku je přiřazena akce, která se vykoná. Přechod do dalšího bloku je podmíněn nadefinovanou podmínkou. Každý prvek, tzn. přechod i blok akcí, může být naprogramován

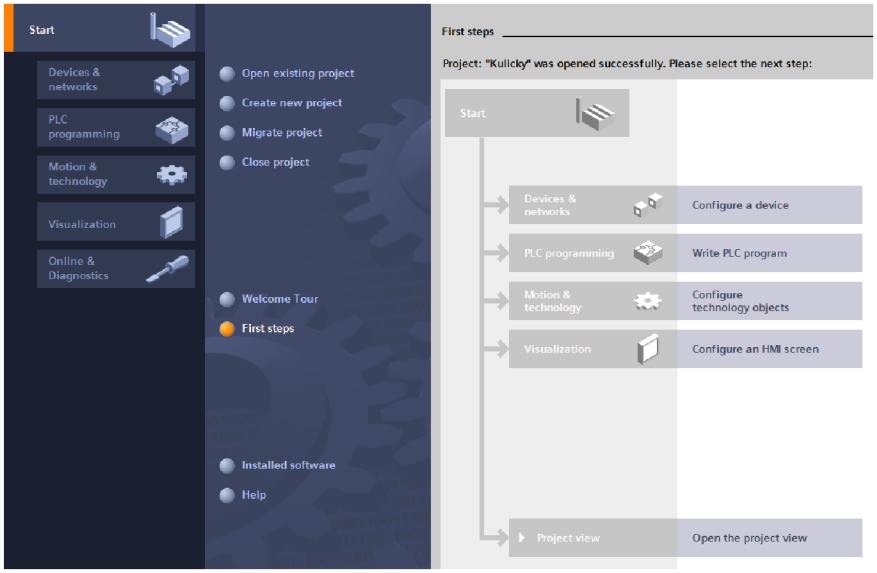
v libovolném jazyce definovaném v normě, včetně vlastního SFC. Ukázka jazyka SFC je na Obr. 1.7. [6]



Obr. 1.7: Ukázka jazyka SFC z vývojového prostředí Simatic Manager.

2 TIA Portal

Totally Integrated Automation Portal, dále jen TIA Portal, je software od firmy Siemens pro programování, konfiguraci a diagnostiku řídících systémů. V současné době nemá na trhu téměř konkurenci. Jeden z důvodu je, že jako jediný program obsahuje společnou platformu pro vývoj řídících aplikací a vytváření vizualizací. Jinými slovy TIA Portal sdružuje psaní programu pro PLC systémy a vytváření vizualizací, tedy vytváření operátorských rozhraní pro stroje a zařízení jak s použitím operátorských panelů HMI (Human Machine Interface – dále jen HMI), tak i pro dispečerské systémy kategorie SCADA (Supervisory Control And Data Acquisition – dále SCADA). Když je vše v jednom programu, pracuje se lépe, než když jsou části uloženy zvlášť. Vývojové prostředí navíc působí velmi příjemným dojmem. Vše je přehledně uspořádané, všechny prvky nastavení od konfigurace PLC až po vizualizace mají stejný princip a přidávání je velmi snadné. Na Obr. 2.1 je základní obrazovka projektu, kde lze přes jednotlivé záložky přistupovat ke všem částem projektu okamžitě a jednoduše. [8] [9]



Obr. 2.1: Vývojové prostředí TIA Portal.

2.1 Programovací jazyky

Vývojové prostředí TIA Portal disponuje všemi jazyky dle normy IEC 61 131-3, která já popsána výše, včetně jazyka SFC. Na výběr jsou zde jazyky LAD, FBD, STL, SCL a GRAPH. Pro ujasnění jazyk LAD odpovídá normovanému LD, STL jazyku IL, SCL jazyku ST a GRAPH jazyku SFC.

2.2 Bloky

Program se dělí do bloků, které jsou nadále rozděleny podle jejich funkcí. V popisu bloků je čerpáno ze zdroje [10].

2.2.1 OB – Organizační blok

Je základním blokem v programu. Při vytvoření projektu se automaticky vytvoří i blok Main (OB1). OB jsou volány cyklicky, odkud se provádí volání bloků a dochází k jejich zpracování. OB je několik druhů. [10]

- OB Startup, který se vykoná jen jednou při uvedení PLC do provozu z režimu STOP.
- OB, které jsou cyklicky volány pro vykonávání programu.
- OB pro zpracování alarmů a chyb.

2.2.2 DB – Datový blok

Slouží k uložení dat programu. Datové bloky obsahují hodnoty proměnných, které jsou využívány v programu. Každý funkční blok v programu může do DB zapisovat data nebo z něj číst. [10]

2.2.3 FB – Funkční blok

Funkčními bloky jsou tvořeny podprogramy, které jsou vždy vykonány, když je funkční blok volán. Takovéto volání se nazývá instance funkčního bloku. Pro každé volání funkčního bloku je přidělen datový blok, do kterého se ukládají zpracovaná data a parametry funkčního bloku. Informace obsažené v tomto datovém bloku jsou uchovány i po vykonání FB. Funkční bloky se používají tehdy, když pro vykonání úlohy nestačí funkce, jinými slovy, je potřeba uchovat informace i po skončení volání funkčního bloku, pro další běh programu, které se uchovají v přiděleném datovém bloku. [10]

2.2.4 FC – Funkce

Funkce obsahuje program, který se vykoná po volání z jiného bloku. K funkci se neváže žádná přidělená paměť, po vykonání funkce jsou použitá data ztracena. Funkci je možné volat několikrát z různých částí programu. Proto je funkce vhodná pro napsání jednoduchých částí programu, které se často opakují. [10]

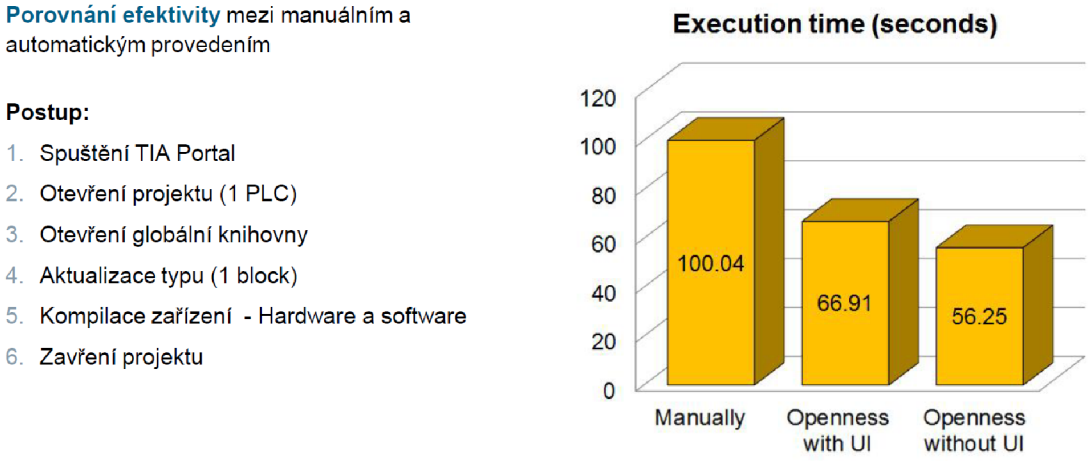
3 TIA PORTAL OPENNESS

TIA Portal mimo jiné obsahuje i rozhraní TIA Portal Openness. TIA Portal Openness je vylepšením hlavně v řešeních pro digitalizaci. *Vývoj směřuje k tomu, že činnosti jako mechanický a elektrický návrh stroje, programování a testování řídicího systému, vlastní výroba stroje a další, budou probíhat stále více paralelně, čímž se může dosáhnout podstatného zrychlení a větší efektivity. K tomu, aby to fungovalo, je třeba zajistit perfektní koordinaci činností a provázanost všech dat. Digitalizace v pojetí TIA Portal verze V14 znamená pracovat s otevřenými standardy a vizualizovanými a maximálně propojenými systémy [11].*

TIA Portal Openness umožňuje programátorům vytvořit nadstavbu, která je schopná automaticky generovat objekty a části programů jak pro PLC, tak i pro vizualizaci na základě informací uložených v textovém souboru nebo ve vhodně konstruovaném XML souboru. Namísto manuální práce, kde jsme pomocí myši a klávesnice pracovali s programem TIA Portal, tak díky Openness, můžeme automaticky spouštět požadované úkony, aniž bychom se nacházeli v blízkosti pracoviště. *Díky TIA Portal Openness jsme schopni komunikovat a zadávat příkazy v TIA Portalu prostřednictvím volně programovatelného rozhraní, tzv. API (Application Programming Interface) [11].* TIA Portal Openness tedy zprostředkovává API do TIA Portalu. API, jak vyplývá z anglické zkratky, představuje rozhraní pro programování aplikací, kterými se ovládají jiné aplikace. V tomto případě TIA Portal. Openness přistupuje k funkcím TIA Portalu přes knihovny *Siemens.Engineering.dll* a *Siemens.Engineering.Hmi.dll*. Tyto knihovny obsahují jednotlivé třídy, které jsou rozděleny podle toho, k jakému účelu slouží. Pomocí těchto tříd je umožněno např. spustit TIA Portal na pozadí (WithoutUserInterface) nebo s uživatelským rozhraním (WithoutUserInterface). Spuštění na pozadí je efektivnější, protože jednotlivé úkony se poté provádějí rychleji, jak je zobrazeno na Obr. 3.1. Po spuštění můžeme vykonávat běžné úkony s projekty, jako je jejich vytvoření, otevření, uložení a zavření. Mimo jiné lze i spravovat zařízení, vizualizace, jednotlivé bloky, tagy a využívat mnoho dalších funkcí TIA Portalu. TIA Portal Openness dále nabízí možnost exportovat či importovat jednotlivé položky jako např. tabulky tagů, datové bloky, funkční bloky do/z XML souborů. [11] [12] [13]

Pomocí TIA Portalu Openness lze výrazně automatizovat procesy, realizovat příkazy v TIA Portalu na dálku, díky čemuž je používání efektivnější a rychlejší. Jeho prostřednictvím dochází k minimalizaci chyb, rychlejšímu provádění úloh, vyšší konkurenceschopnosti, s čímž souvisí i účinnější využívání strojů. Největší předností je zkrácení doby uvádění strojů do provozu. Nejčastějším použitím je například automatické generování projektů pro více strojů, protože většina programů bude mít vždy stejný základ. Důležitým aspektem je možnost pomocí Openness převést

knihovny všech projektů na nejnovější verzi, a to u všech najednou a automaticky. [11] [12] [13]



Obr. 3.1: Porovnání efektivity mezi manuálním a automatickým provozem [12]

3.1 Použití

TIA Openness se používá hlavně pro přístup k projektům a k projektovým datům. Převážně se používá pro:

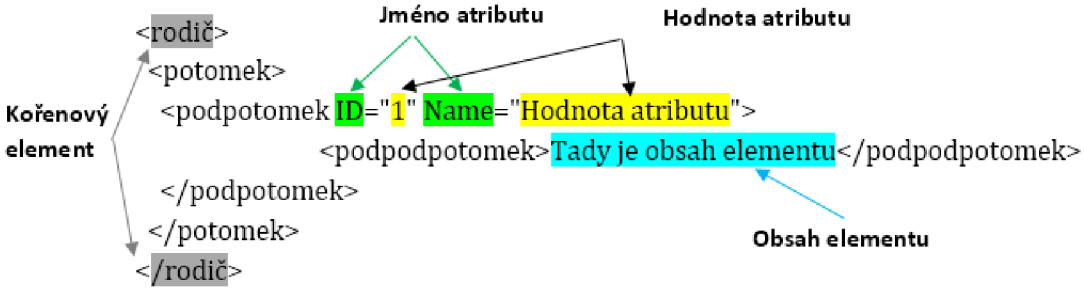
- Otevření, uložení a zavření projektů
- Vyhledávání a vyjmenovávání objektů v projektu jako např. tabulka tagů, PLC a HMI zařízení aj.
- Vytváření objektů.
- Mazání objektů.

První volání funkce z TIA Openness aplikace trvá déle než volání další funkce. Např. pokud se provádí více exportů konfiguračních dat za sebou, první export může trvat déle než následující. [14]

TIA Portal Openness má API do všech jazyků s koncovkou .NET Framework. Nejčastěji se používá jazyk C#, jakožto jednoduchý, moderní a objektově orientovaný jazyk. Protože je v této době C# velmi populární a i Siemens podporuje psaní API v tomto jazyku, je i v této práci použit tento programovací jazyk. Jazyk C# je založený na kostrách jazyka Java a C++. Syntaxi je podobný spíše C++. K vytváření aplikací se používá převážně program Microsoft Visual Studio. Pro programování stačí znát jednoduché programovací příkazy a práci se soubory typu XLM. Pro vytvoření API do TIA Portalu je vhodné používat např. okenní/formulářové aplikace, tzv. WinFomrs. [13]

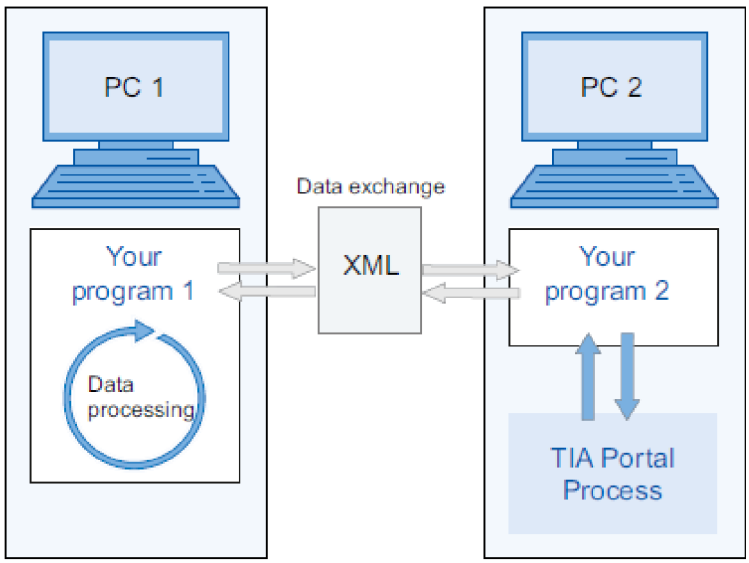
3.2 Podklad pro generování PLC programů pomocí Openness

Jako podklad pro generování jednotlivých PLC programů pomocí TIA Openness se používají XML soubory. Tyto soubory jsou vhodné pro výměnu dat i mezi dvěma různorodými systémy. XML je formát dokumentů stejně tak, jako např. HTML nebo DOC. Jedná se o značkovací jazyk, což znamená, že se užívá značek, u XML jsou jimi elementy, k oddělení jednotlivých prvků. Elementy se skládají většinou z počátečního a koncového tagu. Jednotlivé elementy mohou obsahovat další elementy, tzv. potomky, a ty zas další elementy, tzv. podpotomky, jak je zobrazeno na Obr. 3.2. Vnořených elementů může být neomezený počet, ale při velkém množství dochází k nepřehlednosti programu. Takto vzniká tzv. stromová struktura programu. Každý dokument musí obsahovat tzv. kořenový element, který značí začátek a konec dokumentu. Jednotlivé elementy mohou, ale nemusí mít obsah a atribut. Co to je obsah a atribut je zřejmé z Obr. 3.2. [15] [16]



Obr. 3.2: Vnořené elementy, jejich název, obsah a atributy

Projektová data z prostředí TIA Portal je možné exportovat do souboru XLM, který je možné pozměnit a doplnit novými objekty a následně importovat zpět do prostředí TIA Portal [13]. Jedna z možností, jak pracovat s XML soubory pomocí TIA Openness, je výměna dat mezi dvěma počítači, jak je zobrazeno na Obr. 3.3. Výměna může probíhat tak, že data exportována z projektu TIA Portal na PC 2 mohou být změněna na PC 1 a zpětně importována do projektu v PC 2. Import a export se provádí pomocí předem naprogramované TIA Openness API aplikace, která běží na PC 2. Výhodou je, že PC 1 nemusí obsahovat TIA Portal, ani Openness aplikaci, protože se zde pracuje pouze a jenom s XML souborem. [14]



Obr. 3.3: Výměna dat pomocí XML souborů [14]

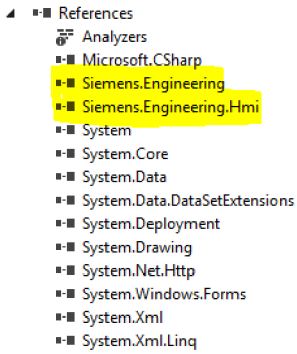
3.2.1 Koncept této práce

Koncept této práce je takový, že bude vytvořen jeden hlavní zdrojový XML soubor jako předpis, podle kterého se poté bude generovat celý program. Tento přepis by měl obsahovat všechny náležitosti, které jsou potřebné pro správné generování programu. Měl by být logicky členěn. Jedno ze základních členění by mělo být rozdělení na PLC program s jeho daty a na HMI program se všemi potřebnými informacemi.

Podle hlavního zdrojového XML souboru by se pak měly přepisovat či jinak upravovat ostatní soubory, z kterých se bude následně generovat program do aplikace TIA Portal.

4 Programování Openness v jazyce C#

Pro využití TIA Openness je nejdříve potřeba přidat knihovny *Siemens.Engineering* a *Siemens.Engineering.Hmi* mezi reference viz Obr. 4.1. Tyto knihovny se nachází v adresáři s nainstalovaným TIA Portalem, přesněji *Siemens\Automation\Portal V14\PublicAPI\V14 SP1*.



Obr. 4.1: Knihovny Siemens

Pro kapitoly 4.2 až po kapitolu 4.7 včetně podkapitol se vychází ze zdroje [14].

4.1 Vysvětlení pojmů z jazyka C#

V následujících podkapitolách budou vysvětleny základní pojmy z jazyka C#, které jsou použity v této práci.

4.1.1 Třída

Třída je základní konstrukční prvek objektově orientovaného programování [17]. Třídu si můžeme představit jako vzor, podle kterého se vytvářejí objekty. Prvek vytvořený podle třídy se tedy nazývá objekt. Třída obsahuje data a operace, které jsou určeny pro práci s těmito daty. *Zároveň slouží jako bezpečnostní prvek, kterým je řízení přístupových práv k datům a metodám [18].* Rozlišují se tři přístupová práva, a to sice veřejná - *public*, která jsou přístupná všem ostatním třídám, soukromá - *private*, která jsou přístupná pouze dané třídě, kde se vyskytují a poslední možností jsou dědičná - *protected*, která jsou přístupná pouze potomkům dané třídy. Metody jsou funkce dané třídy, sloužící k práci s daty, které daná třída obsahuje. Každý nový objekt vytvořený podle dané třídy se pak nazývá instance. Prvek se vytváří buď konstruktorem, který je vždy typu public a jmenuje se stejně jako název třídy, nebo implicitním konstruktorem, který je definován automaticky překladačem. [17] [18] [19]

4.1.2 Výjimky

Výjimky (exceptions) slouží jako nový mechanizmus ošetření chyb. *Jedná se o ošetření „předpokládaných“ chyb, tedy chyb, o kterých programátor ví, že mohou nastat, a tímto se na ně připraví [18].* V místě, kde vznikne chyba, je vygenerována, tzv. vyhozena výjimka. Tato výjimka putuje programem až do místa, kde jí je možné „odchytit“ a vyřešit bez toho, aniž by uživatel přišel o rozpracovaná data. [18]

4.1.3 Vytvoření DirectoryInfo/FileInfo objektu

Tyto objekty jsou obsaženy v knihovně System.IO. Slouží k zadání absolutní cesty k požadované složce, kde se např. nachází projekt, který chceme otevřít, nebo na místo, kam se mají uložit exportované soubory či odkud se mají importovat soubory. Pokud cesta není absolutní, nebo pokud odkazuje na místo, které neexistuje, program vygeneruje výjimku. Ukázka, jak se používají tyto objekty je znázorněno ve výpisu 4.1. [14] [20]

Třída FileInfo poskytuje všechny informace o souboru. Pomáhá tedy pracovat se soubory, jako je čtení existujícího souboru, vytvoření nového souboru, který je specifikován v cestě FileInfo, přidání obsahu do existujícího souboru, zkopírování souboru do jiného nového souboru, odstranění vytvořeného souboru, přesunutí souboru. [20]

Třída DirectoryInfo obsahuje všechny informace o adresáři a některých operacích, jako je odstranění, vytvoření nových adresářů a podadresářů. [20]

```
//Vytvoření DirectoryInfo objektu
string directoryPath = @"D:\Test\Project";
DirectoryInfo directoryInfo = new DirectoryInfo(directoryPath);

//Vytvoření FileInfo objektu
string fileName = @"D:\Test\Project\Project.ap14";
FileInfo fileInfo = new FileInfo(fileName);
```

Výpis 4.1: Vytvoření DirectoryInfo a FileInfo objektu

V této práci bylo použito těchto tříd např. pro otevření TIA projektu či založení projektu nového.

4.2 Start TIA Portalu

Pro start TIA Portalu slouží třída TiaPortal. Při startu programu se vytvoří nová instance této třídy s volbou TiaPortalMode, jak je znázorněno ve výpisu 4.2. TIA Portal je možno spustit s módem WithUserInterface, dále jen with UI, nebo WithoutUserInterface, dále without UI. Pokud se zvolí mód with UI, tak se spustí TIA

Portal v uživatelském prostředí tak, jak by se spustil obvykle bez Openness aplikace. V tomto režimu je možné pozorovat změny, které se provádí, popřípadě některé věci měnit manuálně. Při spuštění s módem without UI se program spustí na pozadí, což přináší větší rychlost provádění úloh tak, jak je znázorněno na Obr. 3.1 viz str. 22. K informaci, zda se požadovaný úkon vykonal, slouží dialogové okno v API Openness aplikaci. [14]

```
//Spuštění TIA Portalu s módem Without UI
TiaPortal MyTiaPortal =
    new TiaPortal(TiaPortalMode.WithoutUserInterface);

//Spuštění TIA Portalu s módem With UI
TiaPortal MyTiaPortal =
    new TiaPortal(TiaPortalMode.WithUserInterface);
```

Výpis 4.2: Start TIA Portalu

Pokud se TIA Portal spustí s metodou without UI, tak pro jeho ukončení slouží metoda Dispose, která zavře aktivní instanci třídy TiaPortal. [14]

4.3 Práce s projektem

Pro práci s projektem slouží třída Project. Pro otevření již existujícího projektu se využívá metoda Project.Open, pomocí níž se vytvoří nová instance třídy Project. Použití demonstruje výpis 4.3. Otevřít lze však jen projekty, které byly vytvořeny s aktuální verzí TIA Portalu, tedy s verzí V14 SP1, nebo projekty, které byly upgradovány na aktuální verzi. Pokud by bylo požadováno otevřít projekt, který byl vytvořen z předchozí verze TIA Portal, tak by metoda Project.Open vygenerovala výjimku. Pro otevření takového projektu se použije metoda OpenWithUpgrade. Tato metoda vytvoří nový upgradovaný projekt a otevře jej. Pokud by se otevíral projekt z verze starší než je předchozí, opět by se vygenerovala výjimka. Pro otevření již existujícího projektu je potřeba nejdříve získat cestu k tomuto projektu. To se nejlépe udělá pomocí otevřeného Průzkumníku Windows, kde jednoduše vybereme požadovaný projekt. [14]

```
//Získání cesty k projektu
OpenFileDialog fileSearch = new OpenFileDialog();
string ProjectPath = fileSearch.FileName.ToString();

//Otevření existujícího projektu
MyProject = MyTiaPortal.Projects.Open(new FileInfo(ProjectPath));

//Získání nové instance třídy ProjectComposition
ProjectComposition projectComposition = MyTiaPortal.Projects;
```

```
//Adresář, kam se uloží nový projekt
DirectoryInfo targetDirectory = new DirectoryInfo("Adresář");

//Vytvoření projektu
MyProject = projectComposition.Create(targetDirectory,NameProject);
```

Výpis 4.3: Otevření a vytvoření projektu TIA Portal

Pro vytvoření nového projektu se používá metoda Create ze třídy ProjectComposition. Tato metoda se volá s parametrem targetDirectory, který představuje cestu k adresáři, respektive místo k uložení projektu, a s parametrem Name, což je jméno projektu. Založení nového projektu je vyobrazeno opět ve výpisu 4.3.

Pro uložení a zavření projektu se používají metody *Save()* a *Close()* třídy Project. [14]

4.4 Práce s PLC a HMI zařízením

Pro všechny práce se zařízením je nutností nejdříve otevřít projekt. Přidat PLC zařízení do projektu je možné pomocí dvou metod obsažených ve třídě Device. Jednou z metod je Creat. Druhou metodou, používanou v této práci, je pak CreateWithItem, která má jako parametry OrderNumber a Verzi požadovaného CPU, jméno stanice např. PLC_1 a jméno zařízení. Jak lze přidat zařízení do projektu je zobrazeno ve výpisu 4.4. [14]

Pokud je zapotřebí jakkoli zasahovat do projektu, ať už jde o export, import objektů nebo pro přidávání tagů, je třeba nejdříve vyhledat požadované PLC nebo HMI zařízení, kterých se má daná operace týkat. Vyhledání se provádí pomocí smyčky foreach, která prochází všechna zařízení, která jsou obsažena ve třídě DeviceComposition. [14]

Dále je třeba vyhledat podle typu zařízení PLC nebo HMI software, označováno také jako PLC/HMI target. K nalezení se využívá funkce, jejímž vstupním parametrem je dané zařízení. Tato funkce obsahuje pak v návratové hodnotě nalezený PlcSoftwar/HMISoftware. Postup, jak vyhledat PLC software ,je ve výpisu 4.4. [14]

```
//Přidání zařízení
MyDevice = MyProject.Devices.CreateWithItem(MLFB, name, devname);

//Nalezení PlcSoftware(PLC target)
private PlcSoftware GetPlcSoftware(Project project)
{
  Device device = EnumerateDevicesInProject(project, "PLC");
```

```
  DeviceItemComposition deviceItemComposition = device.DeviceItems;
  foreach (DeviceItem deviceItem in deviceItemComposition)
  {
    SoftwareContainer softwareContainer =
      deviceItem.GetService<SoftwareContainer>();
    if (softwareContainer != null)
    {
      Software softwareBase = softwareContainer.Software;
      if (softwareBase is PlcSoftware)
      {
        PlcSoftware plcSoftware = softwareBase as PlcSoftware;
        return plcSoftware;
      }
      else
      {
        HmiTarget hmiTarget = softwareBase as HmiTarget;
        return hmiTarget;
      }
    }
  }
  return null;
}
```

Výpis 4.4: Práce s PLC a HMI zařízením

4.5 Práce s PLC tagy

Tato třída obsažená v prostoru (namespace) *Siemens.Engineering.SW.Tags* slouží pro práci s tag tabulkou a PLC tagy v projektu. Mezi nejužitečnější metody z této třídy patří vyhledávání dostupných tag tabulek v projektu či vytvoření nové tag tabulky, čehož bude využito v této práci. Další velmi důležité metody se týkají správy tagů, jako je jejich mazání, ale zejména při generaci programu je důležitější metoda pro přidávání PLC tagů, k čemuž slouží metoda Create obsažená ve třídě tagComposition. Parametry této metody jsou jméno tagu, typ a logická adresa. Nejdříve je však zapotřebí najít nebo vytvořit konkrétní tabulku tagů, do které má být daný tag vložen. Použití je demonstrováno ve výpisu 4.5. [14]

```
//Vytvoření Tag Tabulky s názvem TagTable
PlcTagTable table =
  plcSoftware.TagTableGroup.TagTables.Create("TagTable");
//Vytvoření nového tagu se jménem Zarazka, typu Bool a adresou Q4.0
PlcTagComposition tagComposition = table.Tags;
PlcTag tag = tagComposition.Create(Zarazka, Bool, Q4.0);
```

Výpis 4.5: Práce s PLC tagy

Existuje mnoho dalších funkcí pro práci s tagy a s tabulkami tagů jako například vyhledání časů posledních úprav v dané tabulce. Tyto metody však nejsou v této práci používané.

4.6 Export a import souborů

Vytváření objektů pomocí Openness je otázkou víceméně jen PLC tagů a zařízení. Toto však nestačí pro generování nových projektů. Pro téměř veškeré operace pro přidávání objektů do projektu slouží importování z XML souborů. Stejně tak získávání potřebných objektů z projektu, např. pro přenos s ostatními zařízeními se používá takřka jen metoda export, která vyexportuje daný objekt do XML souboru, obsahující všechny informace.[14]

Funkce import a export se používají převážně pro následující účely: [14]

- Pro výměnu dat
- Kopírování částí projektu
- Externí zpracování konfiguračních dat pro nové projekty založené na stávajících konfiguracích
- Import externě vytvořených konfiguračních dat
- Poskytování dat projektu pro externí aplikace

Podporované bloky pro export a import jsou: [14]

- Funkční bloky (FB)
- Funkce (FC)
- Organizační bloky (OB)
- Datové bloky (DB)

Podporované jazyky pro export jsou: [14]

- STL
- FBD
- LAD
- GRAPH
- SCL

XML soubor vytvořený po exportu z bloku napsaného jazykem SCL neobsahuje kód programu, ale pouze rozhraní nastavení bloku, jako je programovací jazyk aj. Tudíž ve vyexportovaném XML souboru funkce psané jazykem SCL nelze měnit psaný kód a upravovat tak program. Jsou však způsoby, jak upravovat i bloky psané

jazykem SCL. Jeden ze způsobů je demonstrován i v této práci v kapitole 5 Programové řešení. [14]

V Tab. 4.1 jsou označena data, která lze z projektu exportovat, respektive importovat.

Tab. 4.1: Tabulka dat, která mohou být exportována a importována

Objekty	**Export**	**Import**
Bloky	**ANO**	**ANO**
Systémové bloky	**ANO**	**NE**
PLC Tag Tabulky	**ANO**	**ANO**
PLC Tagy a Konstanty	**ANO**	**ANO**
Obrazovky	**ANO**	**ANO**
HMI Tag Tabulky	**ANO**	**ANO**
HMI Tagy	**ANO**	**ANO**

4.6.1 Export

Pro export souborů slouží metoda Export. Prvním parametrem této metody je FileInfo, což zahrnuje cestu, kam má být exportovaný soubor uložen a zároveň i jméno tohoto souboru. Druhým parametrem je pak ExportOptions, což udává způsob, jak mají být data exportována, respektive která data má exportovaný soubor obsahovat. Jsou tři nastavení exportu, přičemž druhé a třetí nastavení exportu může být nastaveno současně. [14]

1. ExportOptions.None
   Toto nastavení exportuje pouze upravená data nebo data, která se liší od výchozích. Exportovaný soubor obsahuje také všechny hodnoty, které jsou povinné pro následné importování dat.
2. ExportOptions.WithDefaults
   Exportovány jsou navíc i standartní hodnoty.
3. ExportOptions.WithReadOnly
   Exportovány jsou i hodnoty chráněné proti zápisu.

Exportovaný XML soubor je velmi obsáhlý. Například pro jednoduchou funkci, která má pouze 14 networků má tento soubor zhruba 1500 řádek.

Ve výpisu 4.6 je zobrazena ukázka části XML kódu exportované funkce psané v programovacím jazyku LAD, která obsahuje elementy, jenž se budou nejčastěji přepisovat při generování nových funkcí. Kód funkce v XML souboru začíná elementem *SW.Blocks.FC*, který má mimo jiné podpotomky *Name* a *Number* představující

jméno a číslo bloku. Pokud se tyto dva elementy nebudou přepisovat, dojde při importování tohoto XML souboru k přepsání funkce, která bude mít stejné jméno, popřípadě číslo.

Networky v dané funkce jsou členěny do elementů *SW.Blocks.CompileUnit*. Jednotlivé proměnné, které se vyskytují v daném networku, jsou obsaženy v elementech *Component*, které jsou zanořeny hluboko ve stromové struktuře XML souboru. V ukázce jsou vidět pouze tři elementy, ve kterých je *Component* zanořený. Jméno proměnné je poté obsaženo v atributu *Name* tohoto elementu.

Změny jména proměnných, jména a čísla funkce se mohou provádět podle schématu XML. Tyto změny zachovají funkčnost programu, nemění se tedy bloky, ale pouze jejich jména či vstupy a výstupy.

```
<SW.Blocks.FC ID="0">
  <AttributeList>
    <Name>Valec1</Name>
    <Number>1</Number>
  </AttributeList>
...
  <SW.Blocks.CompileUnit>
  ...
    <Parts>
      <Access Scope="GlobalVariable" UId="21">
        <Symbol>
          <Component Name="BCD_0" />
        </Symbol>
      </Access>
    ...
    </Parts>
  ...
  </SW.Blocks.CompileUnit>
...
</SW.Blocks.FC>
```

Výpis 4.6: Ukázka kódu exportovaného XML souboru

4.6.2 Import

Konfigurační data jsou importována z dříve exportovaného a upraveného souboru XML nebo z XML souboru, který byl vytvořen dříve. Údaje obsažené v tomto souboru jsou kontrolovány během importu. Data obsažené v XML souboru musí dodržovat stanovená pravidla. Importovaný soubor nesmí obsahovat chyby v syntaxi a struktuře. Pro komplexní změny je vhodné použít XML editor, který tyto kritéria kontroluje. Pokud jsou během importu zjištěny tyto chyby, dojde k vygenerování

výjimky. Při importu bloků je třeba brát na vědomí, že instance DB a použité tagy, které se vyskytují v tomto bloku, nejsou vytvořeny automaticky. Pokud v souboru XML není zadáno žádné číslo bloku, tak je přiřazeno automaticky. [14]

K importu souborů slouží metoda Import a způsob je pak velmi podobný, jako tomu je u exportu. Do prvního parametru se udává cesta, odkud má být soubor importován, a jméno souboru. Druhým parametrem je pak ImportOptions, což udává, jakým způsobem mají být soubory importovány. Toto nastavení importu udává, jakým způsobem se mají objekty importovat, pokud již v projektu existují. Jsou dvě nastavení importu. [14]

1. ImportOptions.None
   Při použití tohoto nastavení budou data importována bez přepsání. Pokud je objekt importován ze souboru XML, který již existuje v projektu, import je přerušen a vygeneruje se výjimka.
2. ImportOptions.Override
   Při použití tohoto nastavení budou data konfigurace importována s automatickým přepisováním. Pokud má objekt v projektu stejný název jako ten importovaný, tak je tento objekt přepsán. Respektive takový objekt je před importem smazán a vytvoří se nový s hodnotami obsaženými z importovaného souboru.

4.7 Externí zdroje

V předchozí kapitole 4.6 bylo zmíněno, že exportovaný blok psaný programovacím jazykem SCL neobsahuje kód a nelze jej tedy měnit. Pro demonstraci, jak upravovat i funkce psané v tomto jazyce, bylo v této práci použito externích zdrojů. Generovat externí zdroj lze pouze z datových bloků a z bloků psaných v jazyce SCL nebo STL. Datové bloky jsou pak uloženy s koncovkou *.db, SCL bloky *.scl a STL bloky *.awl. S každým z těchto bloků je možné pracovat jako s textovým souborem. V programu lze jednoduše tento soubor otevřít jako textový a následně jej přepisovat dle potřeby.

Pro generování upravených souborů do PLC programu je zapotřebí nejdříve přidat tyto soubory do právě otevřeného projektu v TIA Portalu mezi *External source files*, tedy externí zdrojové soubory. Nadále je nutné projít takto přidané externí zdrojové soubory a ty následně generovat do programových bloků. Celý postup, jak vygenerovat programové bloky, je uveden ve výpisu 4.7.

```
//Přidání upravených souborů mezi External source files s
    parametrem jméno bloku a cestou k adresáři, kde se nachází
PlcExternalSource externalSource =
plcSoftware.ExternalSourceGroup.ExternalSources.CreateFromFile("
    Name","Cesta");
```

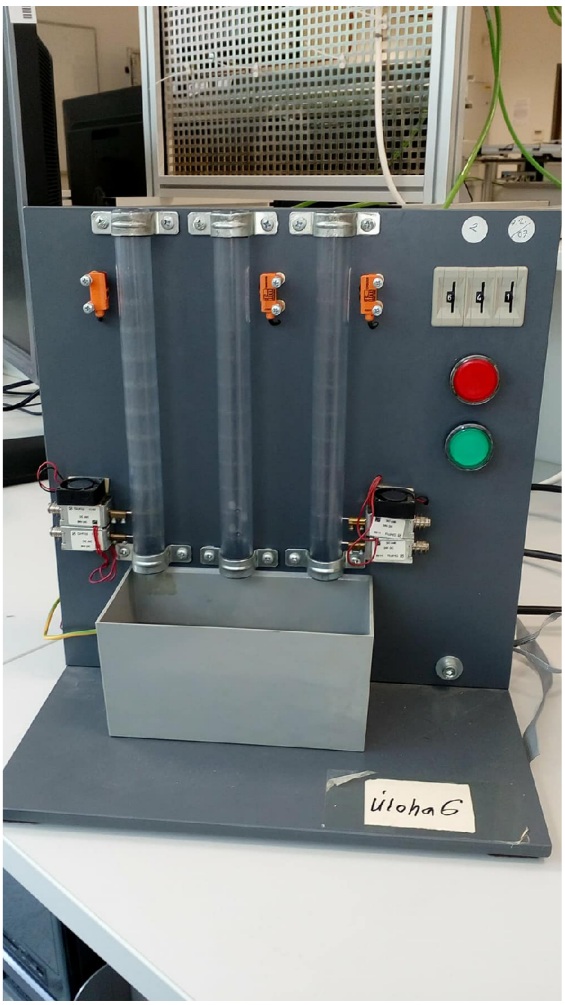
```
//Procházení všech externích souborů
foreach (PlcExternalSource plcExternalSource in
  plcSoftware.ExternalSourceGroup.ExternalSources)
{
  //Generování jednotlivých programových bloků z External source
     files
  plcExternalSource.GenerateBlocksFromSource();
}
```

Výpis 4.7: Generování externích zdrojových souborů

5 Programové řešení

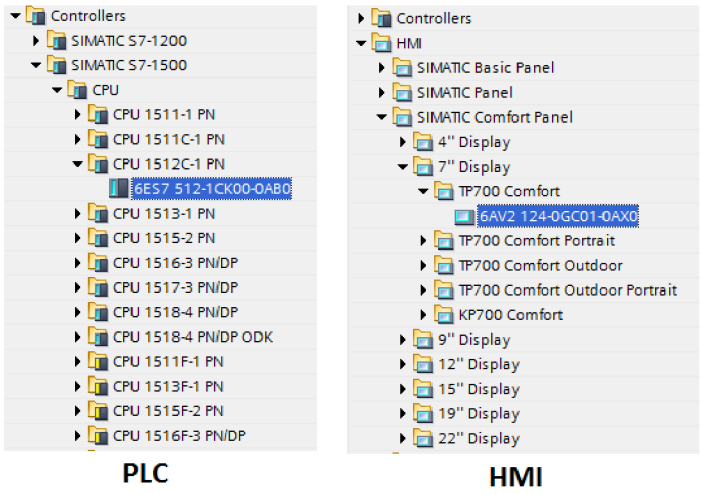
5.1 Programování PLC a HMI

Jako první krok bylo nutné napsat PLC program s vizualizací v prostředí TIA Portal pro model "Kuličky", jež je zobrazený na Obr. 5.1. Byl založen projekt, do kterého je zapotřebí nejdříve přidat PLC hardware. Jak již bylo zmíněno v kapitole 1.3, jednalo se o CPU 1512C-1 PN verze 6ES7 512-1CK00-0AB0 firmware 2.1.



Obr. 5.1: Model Kuličky.

Dále měl být připojen sedmipalcový HMI panel TP700 Comfort verze 6AV2 124-0GC01-0AX0 firmware 14.0.1.0. Objevil se však problém, že tento HMI panel je dostupný pouze ve verzi WinCC RT Advanced. Vedoucím práce bylo zapůjčeno instalační CD a doinstalováno potřebné rozšíření pro TIA Portal. Použitý hardware pro PLC i HMI je uveden na Obr. 5.2.



Obr. 5.2: PLC a HMI hardware.

Po přidání všech zařízení se přistoupilo k samotnému psaní programu. Pro univerzálnost generování byla napsána funkce pro jeden válec s kuličkami a přidány všechny vstupy a výstupy do tabulky tagů. Funkce byla napsána v jazyce SCL, protože při upravování funkce v tomto jazyce se musí postupovat jinak, než k funkcím psaným v jiných jazycích, vyjma STL, viz kapitola 4.6. K této funkci připadá datový blok, který obsahuje všechny potřebné proměnné, viz Obr 5.3.

	Static							
	Cislo	Int	0		☑	☑	☑	
	PrvniStart	Bool	false		☑	☑	☑	
	Citac	Int	0		☑	☑	☑	
	Odecti	Bool	false		☑	☑	☑	
	Stav	Int	0		☑	☑	☑	
	PrvniCasovac	Bool	false		☑	☑	☑	
	DruhyCasovac	Bool	false		☑	☑	☑	

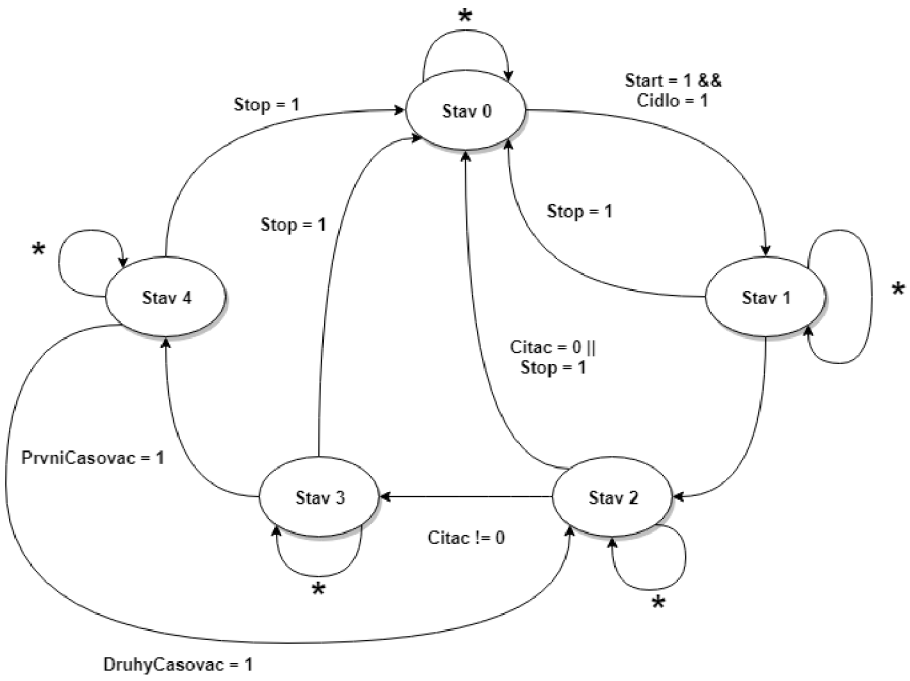
Obr. 5.3: Datový Blok.

Při psaní programu se nejprve převedlo BCD číslo z číselníku pro nastavení počtu vypouštějících se kuliček na číslo desítkové soustavy. Program byl psán dle stavového diagramu uvedeného na Obr. 5.4. V inicializačním stavu 0 se nastaví všechny proměnné do výchozího stavu. Přechod do dalšího stavu je podmíněn podmínkou, že musí být stisklé tlačítko start a zároveň musí být naplněn válec s kuličkami, což indukuje sepnuté čidlo. Ve stavu 1 se nastaví čítač na hodnotu získanou z BCD čísla a dále program čeká na spuštění tlačítka start. Jinými slovy nedojde k přechodu do stavu 2, jestliže bude uživatel stále držet tlačítko start. Ve stavu 2 se kontroluje podmínka, jestli je čítač již na nule (již se vypustil nastavený počet kuliček) nebo ne. Pokud čítač není na hodnotě nula, pokračuje se do stavu 3. Pakliže čítač dočítal,

program se dostane opět do stavu 0. Ve stavu 3 a 4 dojde k vypuštění kuličky a odečtení hodnoty z čítače. Vypouštění kuličky je realizováno pomocí dvou časovačů, kterými se postupně setují a resetují dolní a horní zarážka, které drží kuličky ve válci. Přechod ze stavu 3 do stavu 4 je podmíněn uplynutím nastaveného času prvního časovače. Poté, co "dočasuje"i druhý časovač, program skočí opět do stavu 2. Po stisku tlačítka stop se program dostane do stavu 0 z jakéhokoli stavu.

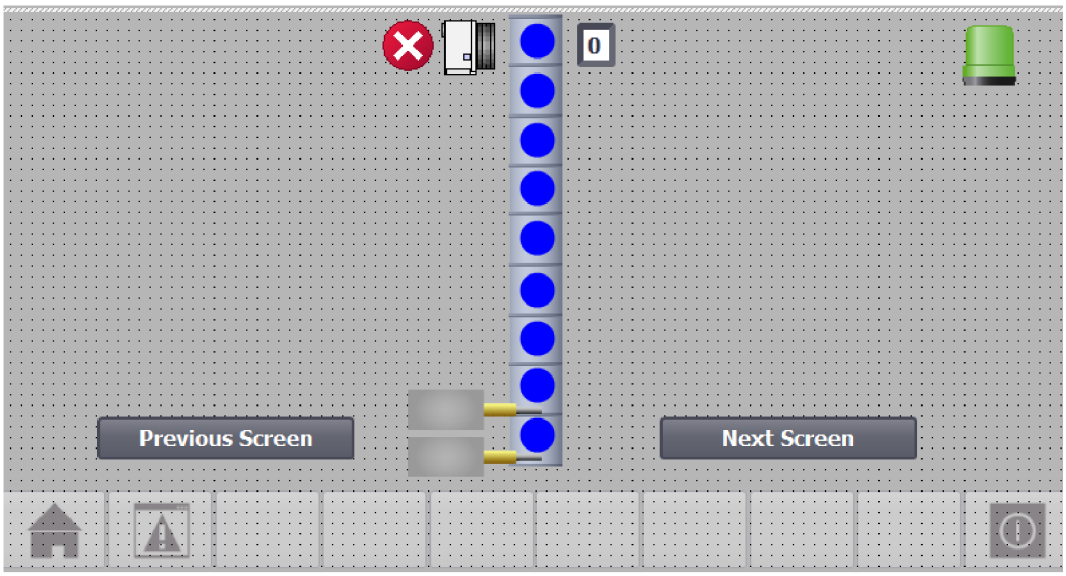
Tento stavový automat je programován pomocí příkazu CASE, který vybírá jednotlivé stavy a vykoná úkon daný vybraným stavem. Jelikož se většina PLC programů píše v jazyce LAD, tak díky neznalosti programovacího jazyka SCL se vyskytl problém s příkazem CASE. U tohoto příkazu je zapotřebí, aby po každém provedení sekvencí úloh v dané podmínce následoval středník. Pokud se tak neučiní, dochází k tomu, že celý příkaz probíhá stále dokola bez ohledu na podmínky. Nakonec i tato nepříjemná záležitost byla vyřešena.

Kompletní SCL funkce je obsažena v příloze A.



Obr. 5.4: Stavový diagram.

Na závěr bylo nutné udělat k programu vizualizaci. Otázkou bylo, jak vytvořit vizualizaci tak, aby bylo možné program následně co nejuniverzálněji generovat pomocí TIA Openness. Na základě této podmínky bylo rozhodnuto, že ke každé funkci bude sestaven displej s obrazovkou, která bude obsahovat pouze jeden válec s kuličkami a přepínači na další obrazovky. Výsledná vizualizace je zobrazena na Obr. 5.5.



Obr. 5.5: Vizualizace.

5.2 Definování XML souboru

Pro nadefinování XML souboru, který má sloužit jako předpis pro generování PLC programu pomocí TIA Openness, bylo nejdříve nutné shrnout, co vše bude nezbytné k vytvoření tohoto programu a nastavit koncept, jak bude tento soubor rozdělen na logické části. Základní dělení je na PLC a HMI údaje. Ukázka navrženého XML souboru je v příloze B ve výpisu B.1. Kompletní XML předpis je pak v příloze E pod názvem Source.xml.

Jedny ze základních informací pro generování PLC programu jsou vstupy a výstupy. Z tohoto důvodu se do PLC dat přidal *TagList*, který má v atributu uvedeno jméno tabulky tagů, která se má vytvořit. *TagList* dále obsahuje všechny potřebné vstupy a výstupy v položkách *TagItem*. Tyto položky obsahují vše potřebné pro generování tagů. Atributy obsahují adresu a typ, obsah elementu a jméno tagu.

Element PLC bude dále obsahovat údaje o blocích, podle nichž se budou jednotlivé bloky generovat. Jednotlivé bloky budou v atributech nést informace o typu funkce, jménu a pořadovém čísle. Bloky dále obsahují data o použitých vstupech *(Inputs)* a výstupech *(Outputs)*, které obsahují v atributu název tagu, jenž je uveden v *TagListu*. Jako obsah elementu je uveden název proměnné, který odpovídá jménu tagu ze vzorového souboru pro daný blok - v tomto případě pro funkci. Pomocí tohoto pojmenování program pozná, co má přepsat. Poslední element v bloku *Dependencies* obsahuje všechny další závislosti, které jsou nutné ke generování programu. Zde jsou zahrnuty převážně použité systémové bloky jako jsou časovače a čítače. Řadí se sem i datové bloky.

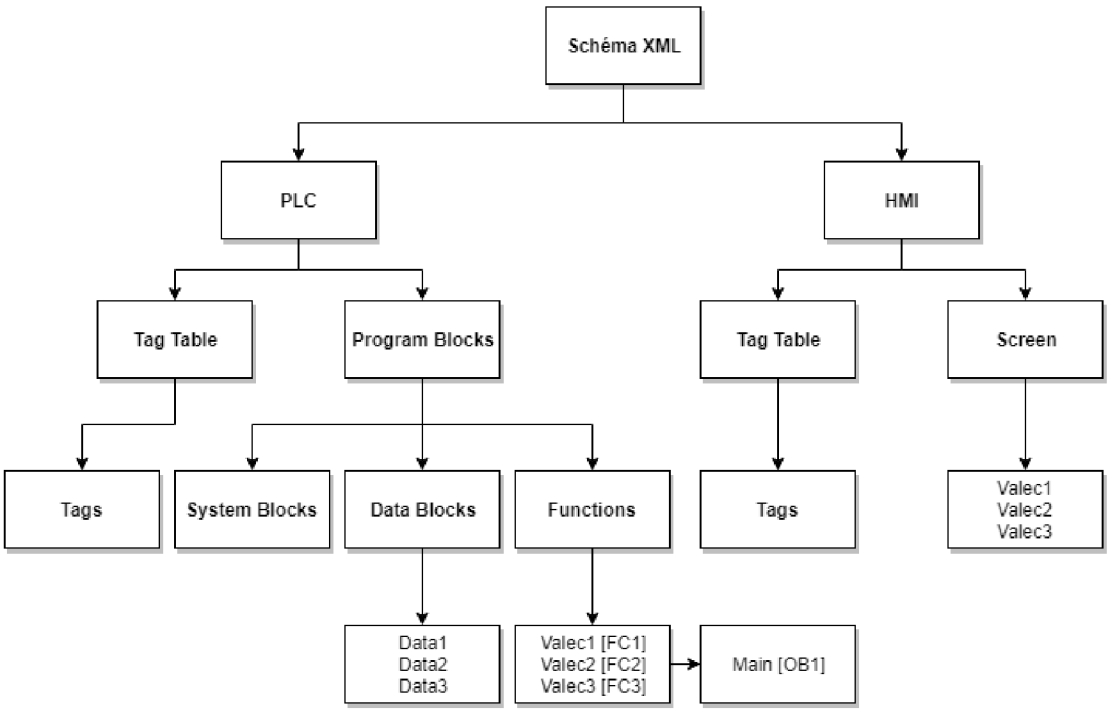
Dále se musí nadefinovat údaje pro HMI. Podobně jak tomu bylo u PLC, tak i HMI nejdříve obsahuje *TagList* s názvem HMI tag tabulky. V tomto *TagListu* se opět nachází seznam tagů. Je zde však rozdíl v atributech. HMI tagy mají v atributech informace o PLC tagu, ke kterému mají být připojeny, a typ tagu. V obsahu elementu pak jeho jméno.

HMI musí disponovat ještě údaji o displejích. Ty jsou obsažené v elementech *Screen*, které jsou zanořeny v *Layouts*. Screeny budou obsahovat *Variables*, tedy jednotlivé komponenty v daném displeji, kterým naleží tagy, dochází k jejich změnám nebo změny vyvolávají. Statické komponenty, které tvoří pouze obraz a nepodléhají událostem, zde uvedeny být nemusí.

Takto nadefinovaný XML soubor je kompletní a může sloužit ke generování kompletního PLC programu.

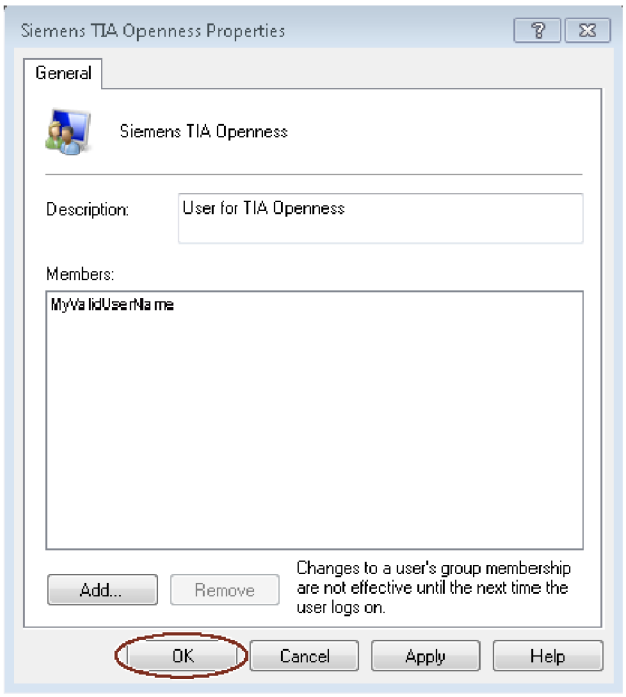
5.3 Generování programu

PLC Program s vizualizací se generuje dle schéma XML souboru viz blokové schéma 5.6.



Obr. 5.6: Blokové schéma generování PLC programu s vizualizací dle schéma XML

Pro generování programu pomocí TIA Openness je nejdříve zapotřebí přidat uživatelský účet Windows do skupiny Siemens TIA Openness viz Obr. 5.7. Podrobný návod je popsán v použité literatuře [14]. Nepatrnou nevýhodou je fakt, že pro přidání účtu do této skupiny je nutné mít prostředí Windows verze Pro. Ve verzi Windows Home tedy nelze TIA Openness používat.



Obr. 5.7: Přidání uživatele do skupiny TIA Openness [14]

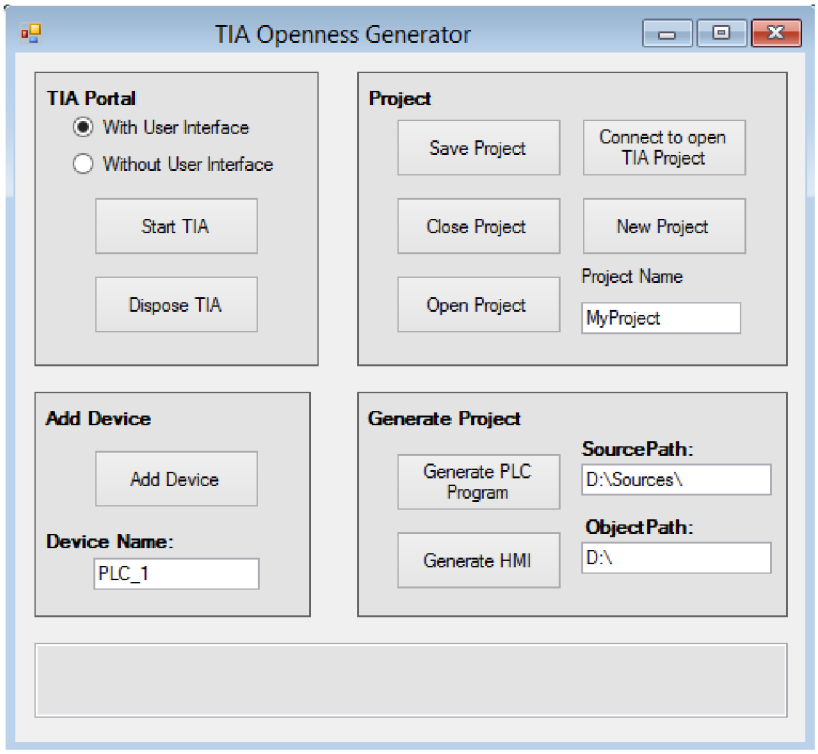
5.3.1 Start TIA Portalu

Program API Openness byl napsán ve vývojovém prostředí Visual Studio 2017 v programovacím jazyce C#, jako desktopová aplikace. Jednotlivé úkony pro ovládaní TIA Portalu se pak budou provádět pomocí jednotlivých tlačítek. Výsledek zvoleného úkonu vždy bude napsán v dialogovém okně, které se nachází dole v této okenní aplikaci. Výsledná API Openness je zobrazena na Obr. 5.8.

5.3.2 Otevření projektu

Ke generování programu je nezbytné mít otevřen projekt v TIA Portalu. To lze udělat dvěma způsoby.

1. Spuštěním TIA Portalu a následným otevřením nebo vytvořením projektu
2. Připojením již k otevřenému projektu.

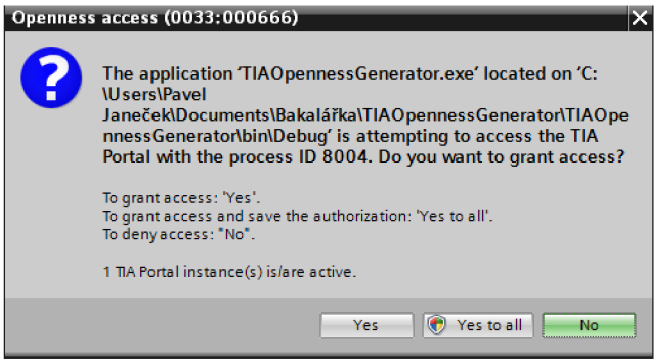


Obr. 5.8: TIA Openness Generator

Prvním způsobem je spustit TIA Portal s vybraným módem, s kterým má být spuštěn buď to with UI, nebo without UI. Automaticky je vybrána možnost with UI. Po tomto kroku je třeba kliknout na tlačítko *Start TIA*. Po kliknutí na toto tlačítko musí být potvrzen přístup API Openness k TIA Portalu, viz Obr. 5.9. a musí se počkat, dokud se v dialogovém okénku neobjeví nápis *TIA Portal started without user interface* nebo *TIA Portal started with user interface*. Výpis signalizuje s jakým módem byl TIA Portal spuštěn, a že spuštění proběhlo úspěšně.

K otevření projektu jsou nyní k dispozici dvě možnosti. První možností je otevření již existujícího projektu pomocí tlačítka *Open Project*. Druhou možností je pak založení nového projektu, kde je nejdříve zapotřebí zadat jeho název do kolonky *Project Name*. Po zadání jména už stačí jen kliknout na tlačítko *New Project*. Projekt se následně vytvoří v adresáři uvedeném v kolonce *ObjectPath* a otevře se.

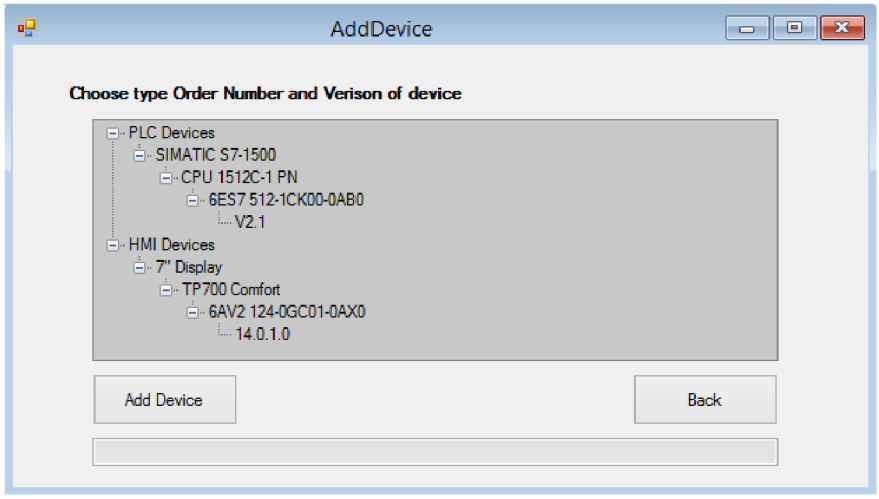
Druhým způsobem otevření projektu je připojení již k otevřenému projektu. Pokud je spuštěný projekt v TIA Portalu, nemusí se zavírat a startovat znovu skrze tlačítko *Start TIA* a následně jej otevírat postupem uvedeným výše. K tomuto účelu zde slouží tlačítko *Connect to open TIA Project*. Při této možnosti se TIA Openness připojí k právě otevřenému projektu. Opět je nezbytné potvrdit přístup API Openness k TIA Portalu.



Obr. 5.9: Přístup k TIA Portalu

5.3.3 Přidání zařízení

Pokud se bude program generovat do nově založeného projektu nebo do projektu, který neobsahuje hardware pro PLC a HMI, je nutností jej nejdříve přidat. Pro tento účel je zde kolonka pro zvolení jména daného zařízení a tlačítko *Add Device*, po jehož stisknutí se otevře nové okno, zobrazeno na Obr. 5.10. V tomto okně se ve stromové struktuře vybere požadovaný hardware a opět se výběr potvrdí tlačítkem *Add Device*.



Obr. 5.10: Okno pro výběr PLC a HMI hardwaru

5.3.4 Přehled tříd a jejich metod

V otevřeném projektu, který obsahuje PLC i HMI hardware, je možné přejít k samotnému generování projektu. Program je členěn na několik tříd podle účelu. Přehled tříd je uveden v příloze na Obr. C.1. Tyto třídy jsou popsány níže.

Třída Receipt

Třída Receipt slouží pro práci se zdrojovým XML souborem, který slouží jako předpis pro generování programu. Zpracuje z tohoto souboru seznam bloků, screenů a TagList HMI a PLC tagů. Pro získání bloků obsahuje tato třída veřejnou metodu *GetBlocks*, která má jako návratovou hodnotu List třídy bloků. Obdobně je koncipována i metoda *GetHMIs*, která vrací List třídy HMI. Ve třídě Receipt se nachází ještě dvě veřejné metody *GetTagList* a *GetHMITagList*, a to sice pro získání HMI a PLC tagů. Tyto dvě metody mají jako návratovou hodnotu XmlNode, jenž obsahuje všechny potřebné tagy vytažené ze zdrojového XML souboru. V neposlední řadě obsahuje tato třída dvě statické proměnné, a to *PathObject* a *PathSource*, ve kterých jsou uloženy adresy vytvořených respektive zdrojových souborů.

Třída Block

Tato třída je určená pro vytváření potřebných bloků, které se budou následně přidávat do TIA Portálového projektu. K tomuto účelu obsahuje tato třída čtyři privátní metody, které se volají z veřejné metody *Proccess()*.

Třída HMI

Jedná se o nejjednodušší třídu, která obsahuje pouze jednu veřejnou, nepočítaje konstruktor, a jednu privátní metodu. Veřejná metoda *Proccess* slouží pouze k volání privátní metody *WriteScreen*, která vytváří jednotlivé screeny.

Třída Generator

V této třídě jsou použity knihovny od Siemensu, které jsou určeny pro komunikaci s TIA Portalem. Obsahuje metody potřebné pro importování a exportování objektů do/z projektu. Vyskytují se zde metody pro získání PLC a HMI zařízení a jejich softwaru. Pro generování programu pak slouží dvě veřejné metody *GeneratePLC* a *GenerateHMI*, ve kterých se volají mimo jiné privátní metody *WriteTags* a *WriteHMITagTable*, které slouží pro přidání tagů do projektu.

Form TIAGenerator a AddDevice

Jedná se o Windows formulářové aplikace. TIAGenerator je hlavní okno, které je na Obr. 5.8 viz str. 41. Z tohoto okna se ovládá celá TIA Openness aplikace. AddDevice je pak podružné okno, které se otevře po stisku tlačítka *AddDevice* a je zobrazeno na Obr. 5.10. Toto okno obsahuje pouze stromovou strukturu s vybranými PLC a HMI softwary.

5.3.5 Generování PLC programu

Ke generování PLC programu dojde po stisku tlačítka *Generate PLC Program*, kdy se zavolá metoda *GeneratePLC* ze třídy Generator, která dále používá metody pro vytvoření tagů a objektů, jejichž funkce jsou popsány v následujících podkapitolách.

Vytvoření Tagů

Jak již bylo zmíněno v kap. 5.3.4, PLC tagy se přidávají ve třídě Generator metodou *WriteTags*, která nejdříve využije metodu *GetTagList* ze třídy Receipt k načtení TagListu z XML předpisu. Tímto způsobem se všechny tagy uloží do listu elementů tzv. XmlNodeListu. Následně se z tohoto listu prochází jednotlivé elementy, tzv. XmlNody, což jsou požadované PLC tagy. Ty se přidávají jednotlivě do projektu, jak je popsáno v kap. 4.5.

Vytvoření funkcí, datových a systémových bloků

Pro funkci, datový blok, časovač a čítač jsou ve zdrojových souborech připraveny předdefinované soubory obsažené v příloze E. Podle typu bloku se otevře odpovídající soubor jako textový a nadále se s i ním takto pracuje. Tento soubor se pak prochází po jednotlivých řádcích, ve kterých se pomocí metody *Contains* vyhledá proměnná obsažená v XML předpisu a dle něj se přepíše. Následně se takto přepsané řádky vloží do nově vytvořeného textového souboru, tím se zajistí zachování původního zdrojového souboru. V příloze A je zobrazen kód SCL funkce, který je obsažen ve zdrojovém souboru pro funkci, kde jsou červeně vyznačeny všechny proměnné, podléhající přepsání, vyjma proměnných *Zelena, Cervena, Stop a Start*, které jsou společné pro všechny funkce. U datových a systémových bloků se pak přepisuje pouze jejich jméno. Všechny vytvořené soubory se přidají do projektu pomocí externích zdrojů, jejichž přidávání je popsáno v kap. 4.7.

V neposlední řadě dojde k přepsání funkce Main pomocí metody *WriteMainBlock*. Otevře se předpřipravený zdrojový XML soubor s touto funkcí, který je součástí přílohy E pod názvem Main.xml. na přiloženém CD. Část XML kódu, který je nutný k přepsání, je zobrazen ve výpisu 5.1. Vytvoří se klony XmlNodů *Parts* a *NameCon*, obsahující volání funkce. Atribut UId elementu *Call* a *NameCon* se zvětší o jedničku, tak aby nebyl duplicitní. Následně se přepíše v elementu *CallInfo* atribut Name a BlockType. Nakonec se takto přepsané klony vloží do XML souboru s funkcí Main, která se za pomoci metody *ImportBlocks* importuje do projektu s nastavením přepsání viz 5.1.

Postup volání všech metod pro generování PLC programu je zobrazen v levé větvi vývojového diagramu uvedeného v příloze D. Schéma PLC programu se nachází na Obr. 5.11.

```
<Parts>
  <Call UId="21">
     <CallInfo Name="Valec1" BlockType="FC" />
  </Call>
</Parts>
<Wires>
  <Wire UId="24">
    <Powerrail />
      <NameCon UId="21" Name="en" />
  </Wire>
</Wires>

//Import bloků
PlcBlockGroup blockGroup = plcSoftware.BlockGroup;
IList<PlcBlock> blocks = blockGroup.Blocks.Import
  (new FileInfo(path), ImportOptions.Override);
```

Výpis 5.1: XML Main

5.3.6 Generování HMI programu

Ke generování HMI programu dojde po stisku tlačítka *Generate HMI*, díky čemuž se zavolá metoda *GenerateHMI* ze třídy Generator, která používá další metody pro vytvoření HMI tagů a displejů, jejichž fungování je popsáno v následujících podkapitolách.

Vytvoření HMI tagů

Vytváření HMI tagů je složitější, než tomu bylo u PLC tagů. Opět k vytváření tagů dochází ve třídě Generator, a to sice v metodě *WriteHMITagTable*. Složitější je to z toho důvodu, že pro vytvoření HMI tagů neexistuje metoda *Create* jako u PLC tagů. Proto bylo nutné najít jiné řešení. Jsou dvě možnosti, jak tento problém vyřešit.

První možností bylo exportovat HMI tag tabulku pro jednu funkci odpovídající fungování jednoho válce modelu kuliček. Ta by měla sloužit jako vzor, podle něhož se budou přepisovat další tabulky. Tímto řešením se měla získat ke každé funkci jedna HMI tag tabulka. Problém však byl, že proměnné, které se v těchto tabulkách vyskytovaly opakovaně, neboť byly pro všechny funkce společné, způsobovaly chyby v programu.

Vzhledem k popsaným problémům volba nakonec spočinula na druhé variantě. V tomto případě se nejdříve exportovala prázdná HMI tag tabulka a jeden samostatný HMI tag. Tyto dva exportované XML soubory sloužily jako vzory a jsou obsaženy v příloze E. Následně se pro každý tag vytvořil vždy klon celého XML souboru

obsahující individuální HMI tag, který se následně přepisoval o všechny potřebné náležitosti, viz výňatek z XML souboru *SingleTag.xml* ve výpisu 5.2. Konkrétně se jedná o elementy *Length* (velikost zabírající paměti podle typu proměnné) a *Name* (jméno proměnné), které jsou potomkem *AttributeList*. Dále se musely upravit tři elementy se shodným jménem *Name*. Jeden byl potomkem *ControllerTag* (příslušný PLC tag), druhý *DataType* (datový typ PLC tagu) a třetí *HmiDataType* (datový typ HMI tagu). U každého tagu se musely přepsat ještě atributy ID u elementů *MultilingualText* a *MultilingualTextItem*, které se vyskytují třikrát v XML souboru pro samostatný HMI tag a všechny musí být přepsány. Upravit se musely tak, aby nebyly totožné s dalším přidávaným tagem. Proto se k těmto atributům, které jsou označeny písmeny abecedy, vždy připsalo číslo displeje obsažené v XML předpisu v atributu Postfix. Tímto se zajistilo, že každý tag bude mít odlišné atributy a nebude docházet k chybám.

```
<AttributeList>
  <Length>1</Length>
  <Name>TagItem</Name>
</AttributeList>
...
<LinkList>
  <ControllerTag TargetID="@OpenLink">
    <Name>TagItem</Name>
  </ControllerTag>
  <DataType TargetID="@OpenLink">
    <Name>Bool</Name>
  </DataType>
  <HmiDataType TargetID="@OpenLink">
    <Name>Bool</Name>
  </HmiDataType>
</LinkList>
//Vystřižený element MultilingualText a MultilingualTextItem
<MultilingualText ID="A" CompositionName="Comment">
<MultilingualTextItem ID="B" CompositionName="Items">
```

Výpis 5.2: Část XML souboru SingleTag podlehající změnám

Takto přepsaný XmlNode se vložil do XML souboru obsahující prázdnou HMI tag tabulku. Vyskytl se však problém, že se nemohou jednoduše vkládat XmlNody z jednoho dokumentu do druhého. Proto bylo zapotřebí použití metody pro import daného XmlNodu do jiného XML dokumentu. Tento proces se opakoval pro všechny HMI tagy obsažené v XML předpisu. Po vyplnění celého XML souboru s HMI tag tabulkou všemi tagy se tento soubor importoval do TIA Portalového projektu.

Vytváření Displejů

Závěrečným krokem bylo vyřešit generování displejů. V tomto případě se přepisoval vzorový XML soubor pro jeden displej, který je součástí přílohy E pod názvem Screen.xml. Změny se prováděly v závislosti na údajích v XML předpisu. Bylo nezbytné přepsat všechny komponenty, které v daném displeji vytvářejí události a jsou tedy součástí XML předpisu. Pro jejich přepsání se musely vyhledat ve vzorovém XML souboru pro displej elementy *Name*, které byly potomkem buď elementu *Tag*, nebo *Value* v závislosti na typu komponenty, viz vytažený úsek kódu ve výpisu 5.3. Nakonec se musel změnit název a číslo daného displeje. Pro tyto změny musely být nalezeny elementy *Name* a *Numer*, které jsou potomkem *AttributeList*.

Takto přepsaný soubor se importoval do projektu pomocí metody import použité na daný displej.

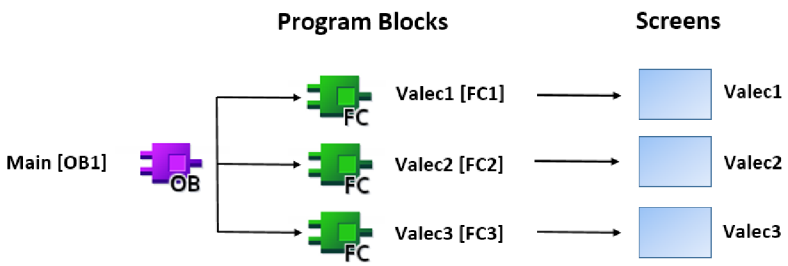
Postup volání všech metod pro generování HMI programu je zobrazen v pravé větvi vývojového diagramu uvedeného v příloze D. Schéma PLC programu se nachází na Obr. 5.11.

```
//Komponenta: tlačítko Next Screen
<Value TargetID="@OpenLink">
  <Name>Next Screen</Name>
</Value>

//Komponenta: IOField počítadlo
<Tag TargetID="@OpenLink">
  <Name>Data_Citac</Name>
</Tag>

//Název a číslo displeje
<AttributeList>
  <Name>MainScreen</Name>
  <Number>1</Number>
</AttributeList>
```

Výpis 5.3: Část XML souboru HMITagTable podlehající změnám



Obr. 5.11: Schéma PLC programu s vizualizací

5.4 Použité třídy a funkce z TIA Portal Openness

Třída TiaPortal

- TiaPortal.TiaPortal(TiaPortalMode)
  Konstruktor třídy TiaPortal pro start TIA Portalu, s parametrem módu spuštění.
- TiaPortal.Dispose()
  Ukončení aktivní instance Tia Portalu, bez parametrů.
- TiaPortal.Projects
  Výpis otevřených projektů TIA Portalu. Vrací typ třídy ProjectComposition.

Třída ProjectComposition

- ProjectComposition.Open(FileInfo)
  Otevření projektu s parametrem path typu FileInfo (soubor projektu). Návratovou hodnotou je třída Project.
- ProjectComposition.Create(DirectoryInfo, string)
  Vytvoření projektu s parametry targetDirectory typu DirectoryInfo (adresář pro uložení projektu) a name typu string(jména projektu). Návratovou hodnotou je třída Project.
- ProjectComposition.Count
  Zjištění počtu aktuálně spuštěných projektů.

Třída Project

- Project.Save()
  Uložení projektu, bez parametrů.
- Project.Close()
  Zavření projektu, bez parametrů.
- Project.Devices
  Výpis všech PLC a HMI v daném projektu. Vrací typ třídy DeviceComposition.

Třída DeviceComposition

- DeviceComposition.CreateWithItem(string, string, string)
  Přidání nového zařízení do projektu. Návratovou hodnotou je třída Device. S parametry:
  - typeIdentifier: Identifikační typ (OrderNumber a verze).
  - name: Pojmenování zařízení.
  - deviceName: Typové jméno zařízení.

Třída PlcExternalSource a PlcExternalSourceComposition

- PlcExternalSource.Delete()
  Smazání souboru z externích zdrojových souborů, bez parametrů.
- PlcExternalSource.GenerateBlocksFromSource()
  Generování bloku z externích zdrojových souborů, bez parametrů.
- PlcExternalSourceComposition.CreateFromFile(string, string)
  Přidání souboru mezi externí zdrojové soubory s parametry name a path označující jméno a cestu k souboru.

Třída PlcBlockComposition a PlcBlock

- PlcBlockComposition.Import(FileInfo, ImportOptions)
  Import PLC bloku s parametry path typu FileInfo (soubor XML a cesta k němu) a ImportOptions (způsob importu bloku).
- PlcBlock.Export(FileInfo, ExportOptions)
  Export PLC bloku s parametry path typu FileInfo (jméno XML souboru a cesta k němu) a ExportOptions (způsob exportu bloku).

Třída PlcTagTableComposition, PlcTagComposition a PlcTagTable

- PlcTagTable.Delete()
  Smazání tabulky tagů, bez parametrů.
- PlcTagTableComposition.Find(string)
  Vyhledání PLC tag tabulky s parametrem name typu string (jméno tabulky). Vrací typ třídy PlcTagTable.
- PlcTagTableComposition.Create(string)
  Vytvoření PLC Tag tabulky s parametrem name typu string (jméno tabulky). Vrací typ třídy PlcTagTable.
- PlcTagComposition.Create(string, string, string)
  Vytvoření PLC tagu s parametry:
  - name: Jméno tagu.
  - dataTypeName: Datový typ tagu.
  - logicalAddress: Logická adresa tagu.

Třída TagComposition, TagTableComposition a TagTable

- Tag.Export(FileInfo, ExportOptions)
  Export jednoho HMI tagu s parametry path typu FileInfo (jméno XML souboru a cesta k němu) a ExportOptions (způsob exportu tagu).

- TagTable.Export(FileInfo, ExportOptions)
  Export HMI tag tabulky s parametry path typu FileInfo (jméno XML souboru a cesta k němu) a ExportOptions (způsob exportu tabulky).
- TagTableComposition.Import(FileInfo, ImportOptions)
  Import HMI tag tabulky s parametry path typu FileInfo (soubor XML a cesta k němu) a ImportOptions (způsob importu tabulky).

Třída Screen a ScreenComposition

- Screen.Export(FileInfo, ExportOptions)
  Export HMI displeje s parametry path typu FileInfo (jméno XML souboru a cesta k němu) a ExportOptions (způsob exportu displeje).
- ScreenComposition.Import(FileInfo, ImportOptions)
  Import HMI displeje s parametry path typu FileInfo (soubor XML a cesta k němu) a ImportOptions (způsob importu displeje).

6 Závěr

Cílem práce bylo vytvoření programu v prostředí TIA Portal s PLC S7-1500 a HMI panelem TP700 Comfort pro školní model "Kuličky", realizující vypouštění kuliček a jeho následné co nejuniverzálnější automatizované generování pomocí nástroje TIA Portal Openness. Před samotnou prací bylo důležité se seznámit s XML formátem souboru a s nástrojem TIA Portal Openness, zjistit, jak fungují základní funkce a osvojit si funkce nové, potřebné pro generování programu z XML souboru.

První část práce se zabývá popisem programovatelných automatů, jejich dělením a popsáním PLC, které je použito v této práci. Dále tato část popisuje programovací jazyky pro programovatelné automaty a jejich dělení podle normy.

V druhé části je stručně představeno vývojové prostředí TIA Portal, jsou popsány jeho bloky sloužící k programování a programovací jazyky, které se v tomto prostředí používají.

Třetí část se věnuje TIA Portal Openness, jeho výhodám a použití. Je představen vhodný formát souboru podle nějž se generují programy pomocí TIA Openness.

Ve čtvrté části jsou nejprve představeny základní pojmy z jazyka C#, které se vyskytují v této práci v hojné míře. Následně je proveden rozbor základních funkcí obsažených v knihovnách *Siemens.Engineering* a *Siemens.Engineering.Hmi*, které jsou zapotřebí k úspěšnému generování pomocí nástroje TIA Openness.

Poslední část se věnuje způsobu realizace okenní aplikace, pomocí níž lze generovat PLC program s vizualizací pro model s kuličkami o libovolném počtu válců. Jejich počet závisí na formulaci schématu XML předpisu, podle nějž se program generuje. Je podrobně sepsáno, jaký byl postup psaní tohoto programu a jakých funkcí se využívá. Jsou zde popsány problémy, se kterými bylo nutné se vypořádat a přijít na vhodná řešení.

V přílohách, které jsou součástí práce je uvedena ukázka schématu XML souboru sloužící jako předpis pro generování. Dále jsou v přílohách obsaženy obrázky, které znázorňují vytvořené třídy, jejich metody a postup jednotlivých volání, díky kterým dochází k vytváření programu. V přiloženém CD se nachází program včetně zdrojového souboru a všechny zdrojové soubory potřebné ke generování programu.

Literatura

[1] KADLENČÍK, Pavel. *Možnosti využití programovatelných automatů Simatic řady S7-1200 od firmy Siemens* [online]. Zlín, 2017 [cit. 2018-05-03]. Dostupné z: <https://digilib.k.utb.cz/bitstream/handle/10563/41283/kadle%C4%8D%C3%ADk_2017_dp.pdf?sequence=1> Bakalářská práce. Univerzita Tomáše Bati. Vedoucí práce Ing. Pavel Nevrátil, Ph.D.

[2] PÁSEK, Jan. *Programovatelne automaty v řízení.* Brno: FEKT Vysokého učení technického v Brně, 2007. [cit. 2018-05-03].

[3] HEMALA, Vojtěch. *Řízení modulu strojní linky průmyslovým automatem* [online]. Brno, 2012 [cit. 2018-05-03]. Dostupné z: <http://docplayer.cz/33523206-Rizeni-modulu-strojni-linky-prumyslovym-automatem.html> Bakalářská práce. Mendelova univerzita. Vedoucí práce Dr. Ing. Radovan Kukla.

[4] *SESTAVA PLC* [online]. [cit. 2018-05-03]. Dostupné z: <http://www.plc-automatizace.cz/knihovna/plc/plc-hw-sestava.htm>

[5] SIEMENS. *Vlastnosti PLC SIMATIC S7-1500* [online]. [cit. 2018-05-03]. Dostupné z: <http://stest1.etnetera.cz/ad/current/index.php?ctxnh=7eaed34950&ctxp=home>

[6] *Programovací jazyky pro PLC* [online]. [cit. 2018-05-03]. Dostupné z: <https://coptkm.cz/portal/?doc=3905>

[7] KOVÁŘ, Josef, Zuzana PROKOPOVÁ a Ladislav ŠMEJKAL, CSC. *Programování PLC* [online]. Praha [cit. 2018-05-03]. Dostupné z: <http://www.spszl.cz/soubory/plc/programovani_plc.pdf>

[8] OPÍCHAL, Antonín. *Řízení a automatizace procesů pomocí distribuovaných sítí systémů firmy Siemens* [online]. Olomouc, 2016 [cit. 2018-05-03]. Dostupné z: <http://docplayer.cz/27199566-Diplomova-prace-univerzita-palackeho-v-olomouci-rizeni-/a-automatizace-procesu-pomoci-distribuovanych-siti-systemu-firmy/-siemens.html> DIPLOMOVÁ PRÁCE. Univerzita Palackého. Vedoucí práce Doc. RNDr. Jiří Pechoušek, Ph.D.

[9] *Siemens TIA Portal – jednotné vývojové prostředí pro automatizaci v průmyslu* [online]. Ústí nad labem: Časopis Automa pro automatizační techniku, 2011, 11(3) [cit. 2018-05-03]. Dostupné z: <http://automa.cz/cz/casopis-clanky/siemens-tia-portal-jednotne-vyvojove-prostredi-pro-automatizaci-/v-prumyslu-2011_03_43212_6058/>

[10] SIEMENS. *SCE Training Curriculum pro Integrované Automatizační Řešení Totally Integrated Automation (TIA): Blokové typy pro SIMATIC S7-1200* [online]. 2012 [cit. 2018-05-03]. Dostupné z: <http://www1.siemens.cz/ad/current/index.php?ctxnh=5dc8474325&ctxp=doc_manualy>

[11] *Nový TIA Portal vám urychlí a zjednoduší práci* [online]. Praha: TT | Technický týdeník, 2017 [cit. 2018-05-03]. Dostupné z: <https://www.technickytydenik.cz/rubriky/archiv-technik/novy-tia-portal-vam-urychli-a-zjednodusi-praci_40558.html>

[12] KVÁČ, Josef. *TIA Portal Openness Generování projektu* [online]. Praha, 2016 [cit. 2018-05-03]. Dostupné z: <https://w5.siemens.com/web/cz/cz/corporate/portal/home/produkty_a_sluzby/IADT/tia_na_dosah/Documents/2016_zari/TIAPortalOpenness.pdf> Prezentace.

[13] *Siemens představuje vizualizační software WinCC v rámci nové verze TIA Portal V14* [online]. Ústí nad labem: Časopis Automa pro automatizační techniku, 2016, 16(11) [cit. 2018-05-03]. Dostupné z: <http://automa.cz/cz/casopis-clanky/siemens-predstavuje-vizualizacni-software-wincc-v-ramci-nove-ver\ze-tia-portal-v14-2016_11_0_9197/>

[14] SIEMENS. *SIMATIC Automating projects with scripts: System Manual* [online]. Mnichov: SIEMENS, 2017 [cit. 2018-05-03]. Dostupné z: <https://cache.industry.siemens.com/dl/files/163/109477163/att_926042/v1/TIAPortalOpennessenUS_en-US.pdf>

[15] Co je XML? *Interval.cz* [online]. [cit. 2018-05-03]. Dostupné z: <https://www.interval.cz/clanky/co-je-xml/>

[16] Základy jazyka XML. *Domovská stránka Jirky Koska – "VŠE O WWW"* [online]. Jiří Kosek, 2001 [cit. 2018-05-03]. Dostupné z: <http://www.kosek.cz/clanky/swn-xml/syntaxe.html>

[17] Třída (programování). *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2017 [cit. 2018-05-03]. Dostupné z: <https://cs.wikipedia.org/wiki/T%C5%99%C3%ADda_(programov%C3%A1n%C3%AD)>

[18] RICHTER, Miroslav. *Praktické programování v C++* [online]. Brno, 2017 [cit. 2018-05-03]. Dostupné z: <http://www.uamt.feec.vutbr.cz/~richter/vyuka/XPPC/spolecne/prednes.pdf>

[19] 1. díl - Úvod do objektově orientovaného programování v C#. *It-network.cz - Ajťácká sociální síť a materiálová základna pro C#, Java, PHP, HTML, CSS, JavaScript a další* [online]. [cit. 2018-05-03]. Dostupné z: <https://www.itnetwork.cz/csharp/oop/c-sharp-tutorial-uvod-do-objektove-orientovaneho-programovani>

[20] Working with FileInfo and DirectoryInfo classes. *C# Corner - A Social Community of Developers and Programmers* [online]. 2009 [cit. 2018-05-03]. Dostupné z: <http://www.c-sharpcorner.com/UploadFile/skumaar_mca/working-with-fileinfo-and-directoryinfo-classes/>

Seznam symbolů, veličin a zkratek

TIA	Totally Integrated Automation
CPU	Central Processing Unit/Centrální procesorová jednotka
OB	Organization Block/Organizační blok
FB	Function Block/Funkční blok
FC	Functions/Funkce
DB	Data Block/Datový blok
PLC	Programmable Logic Controller/Programovatelný logický automat
LD	Ladder Diagram
IL	Instruction list
FBD	Function Block Diagram
ST	Structured Text
SFC	Sequential Function Chart
HMI	Human-Machine Interface/Rozhraní člověk-stroj
SCADA	Supervisory Control And Data Acquisition
XML	eXtensible Markup Language
WinForm	Windows form
API	Application Programming Interface
BCD	Binary Coded Decimal/Dvojkově reprezentované dekadické číslo

Seznam příloh

A Funkce SCL

```
FUNCTION "Valec" : Void
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1

BEGIN

  //Ziskani DEC cisla z ciselniku
  "Data1".Cislo := BOOL_TO_INT("BCD_0") +
  BOOL_TO_INT("BCD_1") * 2 +
  BOOL_TO_INT("BCD_2") * 2 * 2 +
  BOOL_TO_INT("BCD_3") * 2 * 2 * 2;


  IF "Stop" THEN
      "Data1".Stav := 0;
  END_IF;



  CASE "Data1".Stav OF
      0:                                  //Uvodni inicializace
          "Horni" := true;
          "Dolni" := false;
          "Zelena" := false;
          "Cervena" := true;

          IF "Start" AND "Cidlo" THEN              // ceka na start
              "Data1".Stav := 1;
          END_IF;
          ;
      1:
          IF "Start" = 0 THEN          // ceka na pusteni startu
              "Data1".Stav := 2;
          END_IF;
          "Zelena" := TRUE;
          "Cervena" := FALSE;
          ;
      2:
          IF "Data1".Citac = 0 THEN
              "Data1".Stav := 0;
          ELSE
              "Data1".Stav := 3;
          END_IF;
          ;
```

```
        3:                                  //Vypusteni kulicky
            "Horni" := false;
            "Dolni" := true;
            "Data1".Odecti := true;


            IF "Data1".PrvniCasovac = true THEN
                "Data1".Stav := 4;
            END_IF;
            ;
        4:
            "Horni" := true;
            "Dolni" := false;
            "Data1".Odecti := false;


            IF "Data1".DruhyCasovac = true THEN
                "Data1".Stav := 2;
            END_IF;
        ELSE  // Statement section ELSE
            ;
    END_CASE;

    //Deklarace Citace
    "CounterDB1".CTD(CD := "Data1".Odecti,
                     LD := "Start",
                     PV := "Data1".Cislo,
                     CV => "Data1".Citac);

    "Timer1".TON(IN := "Data1".Stav = 3,
                  PT := t#500ms,
                  Q => "Data1".PrvniCasovac
    );

    "Timer2".TON(IN := "Data1".Stav = 4,
                  PT := t#500ms,
                  Q => "Data1".DruhyCasovac);

END_FUNCTION
```

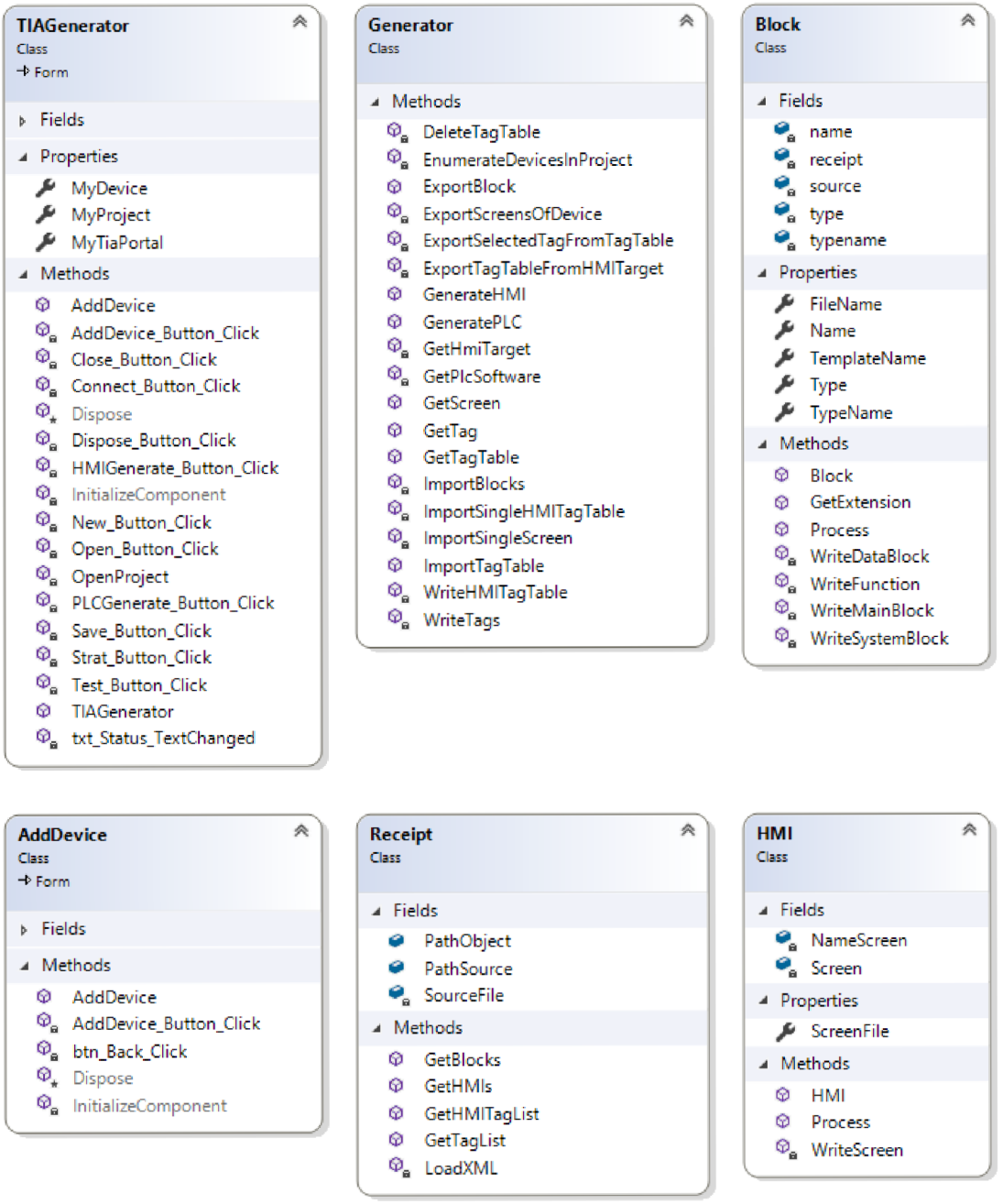
Výpis A.1: Funkce SCL

B Ukázka XML předpisu

```
<PLC Name="PLC_1">
  <TagList Name="TagTable">
    <TagItem Adress="Q4.0" Type="Bool">Horni1</TagItem>
  </TagList>
  <Blocks>
    <Block Type="Function" Name="Valec" Postfix="1">
      <Outputs>
        <Output TagName="Horni1">Horni</Output>
      </Outputs>
      <Inputs>
        <Input TagName="Cidlo1">Cidlo</Input>
      </Inputs>
      <Dependencies>
        <Dependency Type="Timer" Name="Timer_1_DB1">Timer1</
            Dependency>
      </Dependencies>
    </Block>
  </Blocks>
</PLC>
<HMI>
  <TagList Name="HMITagTabel">
    <TagItem PLCTag="Data1.Citac" Type="Int">Data1_Citac</TagItem
        >
  </TagList>
  <Layouts>
    <Screen Name="Valec" Postfix="1">
      <Variables>
        <Variable Component="Pocitadlo" TagName="Data1_Citac">
            Data_Citac</Variable>
      </Variables>
    </Screen>
  </Layouts>
</HMI>
```

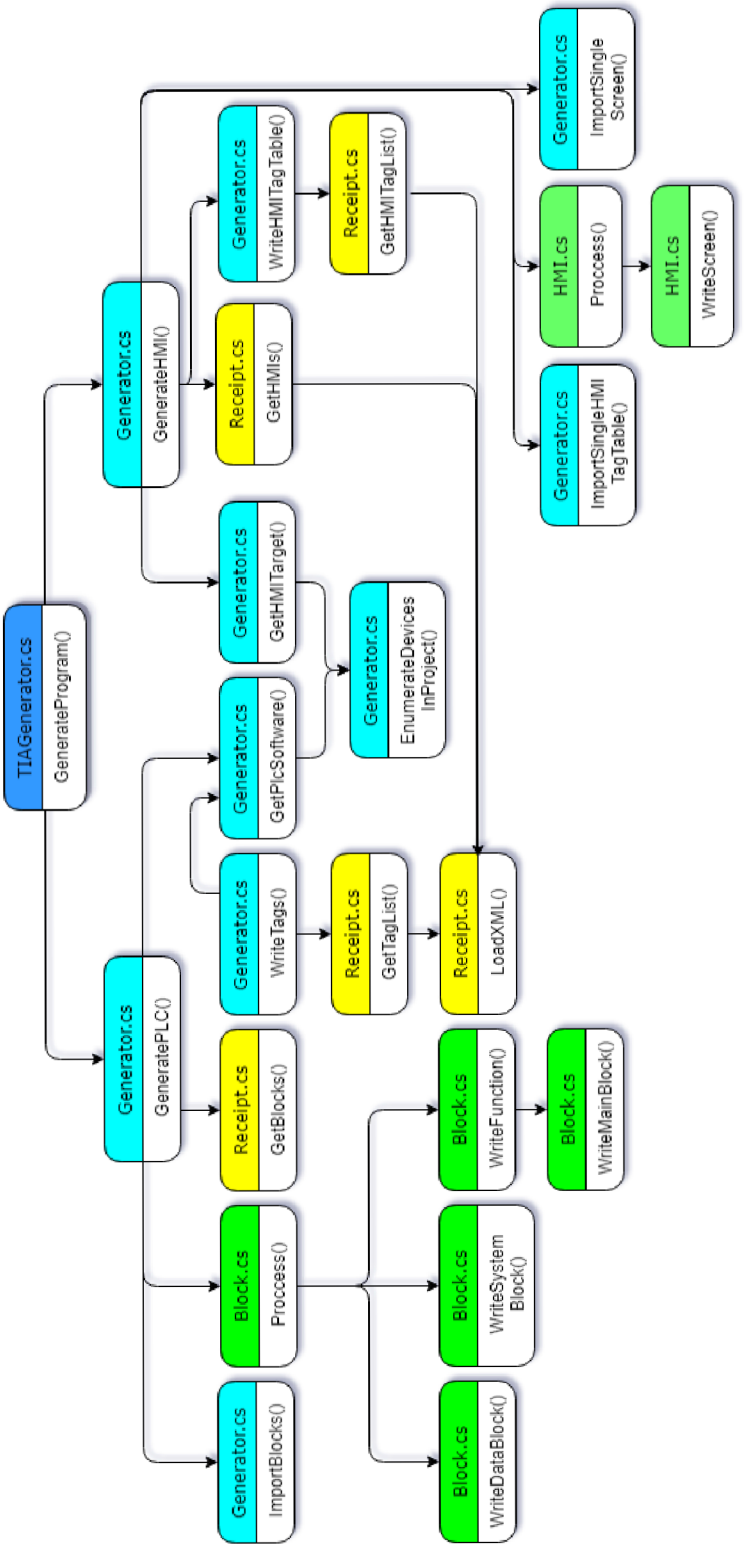
Výpis B.1: XML soubor

C Přehled tříd a jejich metod



Obr. C.1: Přehled Tříd

D Flow Diagram



Obr. D.1: Flow Diagram

E Obsah přiloženého CD

- `PRILOHA_BP_JANECEK_PAVEL` ..... Kořenový adresář přiloženého CD
  - `Janecek_BP.pdf` ..... Elektronická verze bakalářské práce
  - `Kulicky` ..... Projekt TIA Portal
  - `TIAOpennessGenerator` ..... Openness program
  - `Zdroje` ..... Zdrojové soubory
    - `Counter.db` ..... Předpřipravený datový blok pro čítač
    - `DataBlock.db` .... Předpřipravený datový blok obsahující proměnné pro funkci
    - `Main.xml` ..... XML soubor pro generování Main (OB1)
    - `Screen.xml` ..... XML soubor pro generování Displejů u HMI panelu
    - `SingleTag.xml` XML soubor pro jeden HMI tag, určen k přepsání a vložení do HMI Tag Table
    - `Source.xml` ..... XML soubor sloužící jako předpis pro generování programu
    - `TagTable.xml` ..... XML soubor prázdné HMI Tag Table, do které se vkládají individuální tagy
    - `Timer.db` ..... Předpřipravený datový blok pro časovač
    - `Valec.scl` ..... Předpřipravená vzorová funkce

Program je psán pro verzi TIA Portal V14 SP1.