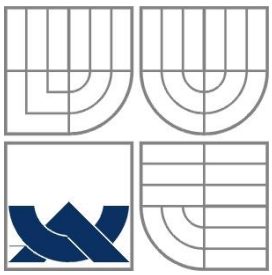
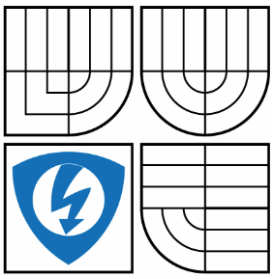


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

KOMPILÁTOR ZDROJOVÉHO KÓDU PRO SIMATIC PLC

DIPLOMOVÁ PRÁCE

AUTOR PRÁCE
AUTHOR

bc. ZDENĚK KUBÁT

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAN PÁSEK, CSc.

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Zdeněk Kubát	***ID:***	125509
Ročník:	2	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Kompilátor zdrojového kódu pro PLC SIMATIC

POKYNY PRO VYPRACOVÁNÍ:

Úkolem je vytvoření nezávislého kompilátoru, který zpracuje zdrojový kód pro Simatic PLC a vygeneruje informace použitelné pro tvorbu HMI aplikace.
Cíle práce:
a. Analýza možností načtení zdrojového kódu PLC ve formátu AWL / STL pomocí mezisouboru anebo jinak (např. vlastní databáze)
b. Vytvoření kompilátoru, který ze zdrojového kódu zapíše relevantní informace do listu Excelu anebo jiného vhodného editoru
c. Vytvoření dalšího modulu kompilátoru, který data z Excelu (anebo jiného editoru) podle parametrů zpracuje a vytvoří data pro Simatic WinCC, včetně generování textů výstrah, alarmů apod. pro objekty z listů zdrojů.

DOPORUČENÁ LITERATURA:

1. Firemní dokumentace společnosti Siemens AG
2. Firemní dokumentace společnosti COMPAS S.R.O.

Termín zadání:	10.2.2014	***Termín odevzdání:***	8.8.2014

Vedoucí práce: Ing. Jan Pásek, CSc.
Konzultanti diplomové práce:

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

UPOZORNĚNÍ:

Abstrakt

Tato diplomová práce popisuje nezávislý kompilátor pro aplikace STEP 7 a WinCC V7.0 společnosti SIEMENS. Kompilátor zpracovává zdrojový soubor generovaný ze STEP 7 a následně ukládá zpracovaná data do mezisouboru typu .xls. Data mezisouboru slouží jako zdrojová data pro funkce generující soubory s tagy, připravenými na import do WinCC V7.0. Pro vytvoření kompilátoru v prostředí Visual Studio 2010 byl použit programovací jazyk C#.

Klíčová slova

Kompilátor, AWL, PLC SIMATIC, STEP 7

Abstract

The thesis discusses an independent compiler for STEP 7 and WinCC V7.0 applications of the SIEMENS company. The compiler processes source files generated in STEP 7 and subsequently saves the processed data in .xls type intermediatefiles. The data of the intermediatefiles serve as source data for functions generating files with tags to be imported into WinCC V7.0. The compiler was created using the C# programming language in Visual Studio 2010.

Keywords

Compiler, AWL, PLC SIMATIC, STEP 7

Bibliografická citace:

KUBAT, Z. *Kompilátor zdrojového kódu pro SIMATIC PLC*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 56s. Vedoucí diplomové práce byl Ing. Jan Pásek, CSc.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Kompilátor zdrojového kódu pro PLC SIMATIC jsem vypracoval samostatně pod vedením vedoucího diplomové 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 diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **8. srpna 2014** ……

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Janu Páskovi, CSc. a odbornému konzultantovi Ing. Martinu Miervovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **8. srpna 2014**

……
podpis autora

Obsah

1 Úvod 10
1.1 Cíle práce 10
2 Používaný software 11
2.1 STEP 7 Professional 11
2.2 WinCC V7.0 11
2.3 Microsoft Visual Studio 2010 12
3 Programovací Jazyky 14
3.1 Jazyky pro programování programových bloků STEP 7 14
3.1.1 AWL/STL 14
3.1.2 LAD 14
3.1.3 FBD 15
3.1.4 S7 – SCL 15
3.1.5 S7 – GRAPH 15
3.2 Programovací jazyk C# 15
4 Formáty používných souborů 17
4.1 Zdrojový soubor STEP 7(formát .awl) 17
4.2 Mezisoubor (formát .xls) 17
4.3 Zdrojové soubory WinCC (formát .csv) 17
5 Typy programových bloků používaných v projektech STEP 7 18
5.1 OB 18
5.2 FC 19
5.3 FB 19
5.4 DB 19
5.5 UDT 20
5.6 SFB 20
6 Mezisoubor pro kompilátor GeneratorTagu 21
6.1 List Param 21
6.2 List Zdroj 22
6.3 List ZdrojOut 23
6.4 List Lang 24
6.5 List TagTypes 25
6.6 List Obj> 26

6.7 List KorekceTextu ... 26
6.8 List Dialog ... 27
7 Předzpracování projektu STEP 7 pro kompilátor *GeneratorTagu* ... 28
7.1 Zpracování programových bloků v jazyce SCL ... 28
7.2 Podporovaná struktura bloků v projektu STEP 7 ... 28
7.3 Označování dat ... 29
7.4 Syntaxe pojmenovávání a komentování programových bloků ve STEP 7 ... 29
7.4.1 FB pro definici zařízení ... 29
7.4.2 FB pro skupinu zařízení ... 30
7.4.3 Volání FB zařízení ze skupinového FB bez možnosti instancování ... 31
7.4.4 Volání FB zařízení ze skupinového FB s možnosti instancování ... 32
7.4.5 Přímé volání FB zařízení (bez volání v instančním FB) ... 33
7.4.6 DB s parametry ... 34
8 Kompilátor GeneratorTagu ... 36
8.1 Generování zdrojového souboru STEP 7 ... 36
8.2 Instalace kompilátoru GeneratorTagu ... 38
8.3 Spuštění kompilátoru GeneratorTagu ... 39
8.4 Obsluha kompilátoru GeneratorTagu ... 39
8.5 Import tagů do WinCC V7.0 ... 40
9 Zdrojový kód Kompilátoru ... 41
9.1 Třída GeneralForm ... 41
9.1.1 AwlReadAwl (region VytvoreniMezisouboru) ... 42
9.1.2 TagGenerateTag (region GenerovaniDatWinCC) ... 43
9.2 Třídy s definicí datového typu ... 43
9.2.1 TypeArea ... 43
9.2.2 TypeAreaAlarmNumber ... 43
9.2.3 TypeAwlDb ... 43
9.2.4 TypeAwlFb ... 43
9.2.5 TypeAwlFbDev ... 44
9.2.6 TypeAwlNextObj ... 44
9.2.7 TypeAwlOneDb ... 44
9.2.8 TypeAwlOneFb ... 44
9.2.9 TypeAwlOneFbDev ... 45
9.2.10 TypeAwlOneNextObj ... 45
9.2.11 TypeAwlOneType ... 45

9.2.12 TypeAwlType 45
9.2.13 TypeCTAlarmOne 45
9.2.14 TypeCTTagOne 45
9.2.15 TypeDb 45
9.2.16 TypeDbOne 46
9.2.17 TypeFbFb 46
9.2.18 TypeFbFbOne 46
9.2.19 TypeCheckStringAddress 46
9.2.20 TypeObject 46
9.2.21 TypeObjectAlarmRow 46
9.2.22 TypeOneObject 47
9.2.23 TypeOneSymbol 47
9.2.24 TypeOutDexFile 47
9.2.25 TypeOutDexFileRow 48
9.2.26 TypeOutVexFile 48
9.2.27 TypeOutVexFileRow 48
9.2.28 TypeRowZdroj 48
9.2.29 TypeSimaticAddress 48
9.2.30 TypeSourceOneLine 48
9.2.31 TypeStruct 49
9.2.32 TypeStructOneLine 49
9.2.33 TypeStructTag 49
9.2.34 TypeSymbolTable 49
9.2.35 TypeTableZdroj 50
9.2.36 TypeTableZdrojAddTag 50
9.2.37 TypeTagParam 50
9.2.38 TypeTagType 51
9.2.39 TypeUpdateDefaultValue 51
10 Závěr 52

1 ÚVOD

Výměna dat mezi vizualizační aplikací a PLC je založena na předávání tagů. Formát těchto dat silně závisí na výrobci PLC i vizualizačního software. Nicméně mnoho firem má na formát předávaných dat vlastní nároky a proto jsou nuceni vytvořit vlastní nástroje pro zpracování těchto dat. Vzniká tak množství firemních řešení, které zpravidla slouží pouze pro interní potřeby dané firmy. Tyto firemní kompilátory načítají data z PLC, umožňují jejich doplnění, úpravu a následně vygenerování zdrojových dat pro vizualizační aplikaci.

1.1 Cíle práce

Prvním cílem této práce bylo zpracovat postupy pro vytvoření vstupních dat pro kompilátor a postupy pro načtení zkompilovaných dat do vizualizační aplikace. Druhým cílem bylo vytvoření a popsání mezisouboru ve formátu .xls. Tento mezisoubor bude využit pro uložení a manuální úpravu dat před vytvořením vstupních dat pro WinCC V7.0. Posledním cílem práce bylo vytvořit kompilátor zdrojového kódu pro PLC řady SIMATIC od firmy Siemens.

2 POUŽÍVANÝ SOFTWARE

2.1 STEP 7 Professional

STEP 7 Professional je programovací a konfigurační software určený pro profesionální použití spolu s řídicími systémy SIMATIC. Zajišťuje podporu uživatele ve všech fázích vývoje projektu.

STEP 7 Professional včetně všech programovacích jazyků odpovídá mezinárodnímu standardu IEC 61131-3 čímž podporuje všeobecnou standardizaci a napomáhá k úspoře nákladů na tvorbu projektu. Se STEP 7 Professional je možné programovat jak řídicí systémy založené na PLC tj. SIMATIC S7 a C7 tak i řídicí systémy založené na PC, tj. WinAC. Tímto je dána uživateli svobodná volba výběru mezi použitím hardwarové platformy nebo smíšené softwarové konfigurace.

STEP 7 Professional je programovací a konfigurační software určený pro profesionální použití spolu s řídicími systémy SIMATIC. Zajišťuje podporu uživatele ve všech fázích vývoje projektu.

STEP 7 Professional se skládá z následujících částí:

- *STEP 7 Basic včetně osvědčených jazyků LAD, FBD, STL,*
- *S7-GRAPH pro grafické programování sekvenčních řízení,*
- *S7-SCL vyšší programovací jazyk pro realizací komplexnějších úloh,*
- *S7-PLCSIM simulátor reálného hardware. Odladění programu v kanceláři bez spojení se skutečným automatem.*

Navíc jsou k dispozici další programové balíčky přímo založené na STEP 7, které dále zefektivňují specifické úlohy programování konkrétních systémů a aplikací. V katalogu je lze nalézt pod označením Inženýrské nástroje (Engineering tools) a Runtime software.[1]

2.2 WinCC V7.0

SCADA software, pracující na platformě Windows XP/7, který je vhodný pro všechny náročné aplikace ve všech oblastech průmyslu. Může být nasazen jako jedna stanice případně jako více stanic (až 12 redundantních serverů) s architekturou server-klient (až 32 klientů na server). WinCC je k dispozici v různých „velikostech" dle počtu procesních proměnných, které se přenášejí z řídicích prvků do vizualizace a opačně.

WinCC je vybudováno na relační databázi (MS SQL Server), v níž jsou uložena konfigurační i archivní data. Tato koncepce umožňuje přístup ke zmíněným datům metodami ODBC (Open Data-Base Connectivity), SQL (Structured Query Language). Aplikace běžící paralelně s WinCC, např. MS Excel, mohou spolupracovat na datech prostřednictvím DDE (Dynamic Data Exchange), OLE (Object Linking and

Embedding) a OPC. Svou otevřeností je sytém WinCC připraven na komunikační propojení s vnitropodnikovými systémy (např. systém SAP).

Z hlediska současných požadavků odpovídá systém WinCC standardu FDA 21 CFR část 11, který vyžaduje důslednou archivaci všech operátorských zásahů v běžící aplikaci WinCC. Nezbytnou součástí aplikace je systém správy jednotlivých operátorů a jejich přístupových práv. Tento systém je možno ovládat z centrální stanice, čímž je zajištěna nejvyšší ochrana nastavených hodnot. Pokud to okolnosti vyžadují je možno archivovat i projekční změny vlastní WinCC aplikace.

Sledovací a ovládací úlohy aplikace WinCC je možné provádět přes Intranet/Internet rozhraní. Díky tomu je možné z jakéhokoli místa na světě sledovat požadované hodnoty a případně provést operátorský zásah. Veškeré zásahy jsou i zde plně zabezpečeny proti zneužití neoprávněnou osobou.

Relační databáze MS SQL Server plně zabezpečuje možnost dlouhodobého archivování požadovaných hodnot. Systém umožňuje záložní archivaci na vybrané médium s volbou zpětného vložení do WinCC.

Nadstavba Basic Process Control přináší progresivní a standardizované řešení pro vlastní vzhled WinCC aplikace, který je možný vytvořit v několika minutách, čímž dojde k podstatnému snížení projekčních nákladů. Součástí nadstavby je varianta pro aplikace s více monitory.

Pro zabezpečení integrity WinCC aplikace s více stanicemi je možno nakonfigurovat tzv. inženýrskou stanici, kde se veškeré změny provádějí. Systém v této stanici sleduje realizované úpravy, které je možné následně on-line přenášet do ostatních stanic. [2]

2.3 Microsoft Visual Studio 2010

Visual Studio obsahuje editor kódu podporující např. intelliSense a refaktorování. Integrovaný debugger pracuje jak na úrovni kódu, tak na úrovni stroje. Další vestavěné nástroje zahrnují designer formulářů pro tvorbu GUI (grafického rozhraní) aplikací, designer webu, tříd a databázových schémat (formulář lzejednoduše vytvořit přetahováním grafických prvků na formulář). Je možné přidávat rozšíření, což vylepšuje funkčnost na téměř každé úrovni – od přidání podpory pro verzovací systémy (jako Subversion a Visual SourceSafe) do přidání nových nástrojů jako editory a vizuální designery pro jazyky specifické pro obor nebo nástroje pro další aspekty návrhu programu (jako klient Team Foundation Serveru: Team Explorer). [3]

IntelliSense je funkce, která automaticky během psaní programu vyhledává a doplňuje jména proměnných, tříd, metod a datových typů.

Refaktorování slouží k hromadnému přejmenování proměnných z jediného místa v programu.

Visual Studio 2010, zvané "Dev10", bylo uvedeno 12. února 2010. Uživatelské prostředí bylo přepracováno s použitím Windows Presentation Foundation a Managed

Extensibility Framework. Visual Studio 2010 také obsahuje vylepšené nástroje pro vývoj a ladění vícevláknových aplikací a jejich profilování, vizualizaci vláken, úkolů (TPL) a jejich aktuálních zásobníků.

Nástroj Call Hierarchy umožňuje pro vybranou metodu v kódu zobrazit, jak metody které ji volají, tak i metody které samá volá. Technologie Quick Search dovoluje inkrementální vyhledání mezi symboly v otevřených projektech. Editor také podporuje přístup consume-first, kdy programátor nejprve v kódu použije proměnnou nebo metodu a Visual Studio ji na základě tohoto použití vygeneruje.

Visual Studio Ultimate 2010, zvané "Rosario", je další verze Visual Studio Team Systemu a je nabízeno jako nástroj pro "integrovanou správu životního cyklu aplikace". Záměrem je umožnit a rozšířit vývoj v jakékoliv fázi životního cyklu aplikace od návrhu po vydání a udržování. Pro řízený kód obsahuje Visual Studio Ultimate 2010 i historický debugger IntelliTrace, který kromě aktuálního stavu programu ukládá informace (např. hodnoty parametrů, volání metod, výskyt událostí) průběžně a dovoluje je zobrazit zpětně (lze krokovat program nazpět). [3]

3 PROGRAMOVACÍ JAZYKY

Tato kapitola shrnuje stručné charakteristiky programovacích jazyků zmíněných v textu této diplomové práce.

3.1 Jazyky pro programování programových bloků STEP 7

Programové bloky pro řídicí systémy SIMATIC v programovacím software STEP 7 lze vytvářet v několika programovacích jazycích. Může se jednat o jazyky grafické či textové.

3.1.1 AWL/STL

Jedná se o jazyk založený na vykonávání sekvence instrukcí, přičemž každá instrukce začíná na novém řádku.

Zkratka AWL vychází z německého označení *Anweisungslist*, v překladu do češtiny tedy *Seznam instrukcí*. V anglické lokalizaci je jazyk znám jako *Statement list*, zkráceně STL. Jazyky AWL a STL jsou sice funkčně stejné, ale názvy jednotlivých instrukcí se liší, neboť vychází z německých a anglických názvů dané instrukce. Jako příklad poslouží instrukce logického součinu, v STL je využívána instrukce *A* (z anglického and) v AWL je jejím ekvivalentem ovšem instrukce *U* (z německého und).

STL je v porovnání s grafickými programovacími jazyky méně přehledný ale zároveň umožňuje provádět i operace v grafických jazycích LAD a FBD nedostupné (např. přepis struktury o velikosti 32bitů do proměnné typu Double Word). Proto se v praxi používá programových bloků, kde je část Networků napsána v STL a zbývající část například v FBD.

Program lze převádět mezi jazyky STL, FBD a LAD.

3.1.2 LAD

Grafický programovací jazyk Ladder Logic (LAD) je v češtině označován jako reléový, liniový či žebříčkový diagram. Program je složen z linií, na kterých programátor definuje podmínky sepnutí daného výstupu či funkce. Podmínky jsou založeny na výrokové logice (reprezentováno jako kontakt se stavy sepnuto/rozepnuto) a vyhodnocují se směrem z leva doprava na každé linii. Ukončit linii lze pouze výstupem, či vstupem bloku (čítače, časovače, FB, FC, atd.). Pro zvýšení přehlednosti, je program členěn do částí nazývaných Network. Výhodou tohoto programovacího jazyka je jeho jednoduchost, ale při použití rozsáhlých logických podmínek se stává program nepřehledným.

3.1.3 FBD

Jazyk funkčního blokového schématu (zkratka FBD je odvozena z názvu Function Block Diagram) používá k reprezentaci logických podmínek bloky s definovatelným počtem vstupů. Dochází tak k zvýšení přehlednosti programu oproti LAD (logický součin pěti proměnných lze namísto pěti kontaktů v sérii reprezentovat jediným pětivstupovým blokem AND). Logický signál z výstupu logické funkce je přiváděn na blok výstupu nebo vstupy následujících bloků.

3.1.4 S7 – SCL

Vyšší programovací jazyk vycházející z jazyka Pascal a odpovídající strukturovanému textu (ST) podle DIN EN/IEC 61131-3, vhodný spíše pro rozsáhlejší algoritmy s matematickými funkcemi nebo úlohy zpracování dat.[4]

3.1.5 S7 – GRAPH

Jazyk pro grafické programování algoritmů sekvenčního řízení: díky standardizovanému uživatelskému rozhraní (IEC 61131-3, DIN EN 61131) lze sekvence programovat a parametrizovat rychle a jednoznačně.[5]

S7 – GRAPH je jednou z obdob jazyka SFC (Sequential Function Chart) popisuje sekvenční chování řídicího programu. Je odvozen ze symboliky Petriho sítí. SFC umožňuje rozložit úlohu řízení na zvládnutelné části a zachovat přitom přehled o chování celku. Sekvenční funkční diagram se skládá z kroků a přechodů. Každý krok reprezentuje stav řízeného systému a má k sobě přiřazen blok akcí. Přechod je spojen s podmínkami, které musí být splněny, aby mohl být deaktivován krok, který přechodu předchází, a naopak aktivován krok, který následuje. Každý prvek, tzn. přechod i blok akcí, může být naprogramován v libovolném jazyce, včetně vlastního SFC. Jazyk umožňuje i větvení programu se spojením alternativních větví a paralelní souběh více větví s jejich následnou synchronizací.[6]

3.2 Programovací jazyk C#

Jazyk C# vyvinula firma Microsoft. Byl představen spolu s celým vývojovým prostředím .NET. Jak název napovídá, vychází tento jazyk v mnohém z programovacího jazyka C/C++, ale v mnoha ohledech je daleko bližší programovacímu jazyku Java. Základní charakteristiky jazyka jsou:

- *Jazyk C# je čistě objektově orientovaný.*
- *Obsahuje nativní podporu komponentového programování.*
- *Podobně jako Java obsahuje pouze jednoduchou dědičnost s možností násobné implementace rozhraní.*

- *Vedle členských dat a metod přidává vlastnosti a události.*
- *Správa paměti je automatická. O korektní uvolňování zdrojů aplikace se stará garbage collector.*
- *Podporuje zpracování chyb pomocí výjimek.*
- *Zajišťuje typovou bezpečnost a podporuje řízení verzí.*
- *Podporuje atributové programování.*
- *Zajišťuje zpětnou kompatibilitu se stávajícím kódem jak na binární tak na zdrojové úrovni.*

Většina uvedených vlastností vychází přímo s funkcionality vývojového rámce .NET. Jazyk C# je také integrován do vývojového prostředí Visual Studio .NET.

Překladače jazyka C# jsou case sensitive. Rozlišují tedy velká a malá písmena. Podobně jako v jiných programovacích jazycích, i v jazyce C# bylo zavedeno několik konvencí. Jména balíků, tříd, rozhraní a většiny dalších položek začínají velkým písmenem. Malým začínají privátní a chráněné (protected) atributy, lokální proměnné a parametry. [7]

4 FORMÁTY POUŽÍVNÝCH SOUBORŮ

Kapitola je zaměřena na popis formátů souborů použitých při práci s vytvořeným kompilátorem dat. Konkrétně se jedná o popis souboru formátu .awl, .xls, .sdf a .csv.

4.1 Zdrojový soubor STEP 7(formát .awl)

Soubory typu .awl využívá společnost SIEMENS pro ukládání programů vytvořených v programovacím jazyce AWL. K otevření souborů typu .AWL je třeba využít textový editor (například Microsoft Word).

4.2 Mezisoubor (formát .xls)

Binární formát souboru aplikací Excel 97 až Excel 2003 je sbírkou záznamů a struktur, které specifikují obsah konkrétního souboru (nazývaného sešit). Sešit může obsahovat nestrukturované či částečně strukturované tabulky čísel, textů, vzorců, externích dat, grafů a obrázků. [8]

4.3 Zdrojové soubory WinCC (formát .csv)

CSV je jednoduchý souborový formát určený pro výměnu tabulkových dat. Soubor ve formátu CSV sestává z řádků, ve kterých jsou jednotlivé položky odděleny znakem čárka.

CSV z anglického Comma-separated values, tedy hodnoty oddělené čárkami, je jednoduchý souborový formát určený pro výměnu tabulkových dat. Hodnoty položek mohou být uzavřeny do uvozovek, což umožňuje, aby text položky obsahoval čárku. Pokud text položky obsahuje uvozovky, jsou tyto zdvojeny.

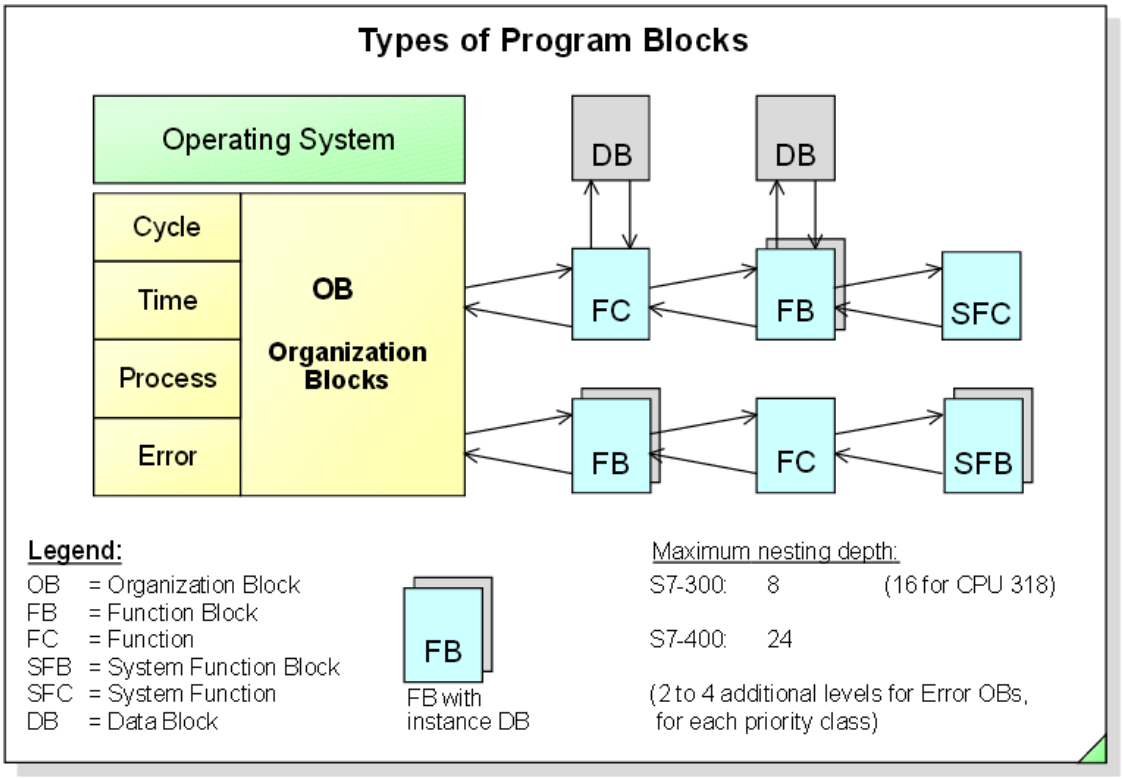
Jelikož se v některých jazycích včetně češtiny čárka používá v číslech jako oddělovač desetinných míst, existují varianty, které používají jiný znak pro oddělování položek než čárku, nejčastěji jde o středník nebo tabulátor (taková varianta se pak někdy označuje jako TSV, Tab-separated values).

Díky jednoduchosti, nenáročnosti a čitelnosti i bez specializovaného softwaru se tento formát používá pro výměnu informací mezi různými systémy. Ke stejnému účelu se dnes používá i modernější a univerzálnější (ale složitější) formát XML. [9]

5 TYPY PROGRAMOVÝCH BLOKŮ POUŽÍVANÝCH V PROJEKTECH STEP 7

Program vykonávaný PLC řady SIMATIC je členěn do programových bloků, které jsou nahrány do paměti PLC. Koncepce bloků je výhodná díky své modularitě, protože pokud je blok naprogramován univerzálně, lze ho používat i v dalších projektech.

Programové bloky jsou označeny zkratkou reprezentující typ bloku a číslem bloku. V jednom projektu se nemůže vyskytovat více bloků se stejným jménem. Počet bloků v projektu je závislý na velikosti uživatelské paměti, kam jsou ukládány. Přehledové schéma typů programových bloků zobrazuje Obr. 5-1



Obr. 5-1 Přehled typů programových bloků [10]

5.1 OB

Organizační bloky (OB) tvoří rozhraní, mezi operačním systémem CPU a uživatelským programem. V OB lze definovat pouze lokální proměnné, které nemají při volání bloku předem definovanou hodnotu. Součástí každého programu je základní organizační blok OB1. Tento programový blok cyklicky vykonává program, který je do něj zapsán. PLC vykoná program uložený v blocích FC a FB, pouze pokud jsou tyto bloky volány z OB1 nebo z bloků volaných OB1. Samotné organizační bloky volat z jiných bloků nelze. K volání OB operačním systémem dochází například na základě následujících událostí:

- Spuštění CPU
- Restart CPU
- Uplynula nastavená časová perioda
- Nastala nastavená denní doba
- Vyskytla se chyba software
- Došlo k hardwarovému přerušení

5.2 FC

Programové bloky FC reprezentují funkce s definovatelnými počty vstupních, výstupních, vstupně-výstupních a lokálních parametrů. Funkce neumožňuje definovat statické parametry. Data lze ve funkci ukládat pouze do adresových slov nebo do datové oblasti v bitové paměti PLC. O udržování takto uložených dat musí ošetřit sám programátor. Volání funkce lze provést v blocích OB, FB a FC.

5.3 FB

Funkční blok FB na rozdíl od FC pracuje se statickými proměnnými (kromě vstupních, výstupních, vstupně-výstupních a lokálních parametrů definujeme i statické parametry). To znamená, že blok lokálních dat je přiřazen k funkčnímu bloku. Vzniká tak instanční datový blok. Při volání FB je nutné definovat s jakým DB (datovým blokem) je volaný blok svázán. Do přiřazeného DB se pak při práci s daným FB zapisují aktuální hodnoty proměnných. Programátor tudíž nemusí věnovat zvýšenou pozornost ukládání dat. Zároveň je snížena pravděpodobnost omylu programátora, kdy v důsledku vícenásobného přiřazení adresové nebo datové oblasti v bitové paměti dochází k nežádoucímu přepisování ukládaných dat.

Jednomu FB lze přiřadit i více DB. Na místě volání FB je však určeno, z jakého DB bude přijímat data.

5.4 DB

Datové bloky (DB) uchovávají data využívaná uživatelským programem. Každé DB je datovou strukturou, složenou z jednotlivých proměnných. Definice proměnné obsahuje adresu, jméno, datový typ a počáteční hodnotu. Přičemž adresa v DB je určena typem a počtem proměnných definovaných před nově vkládanou proměnnou. Nepovinou součástí definice proměnné je komentář. Přístup uživatelského programu k datům je prováděn prostřednictvím bitových a bytových operací, slova a dvojslova.

Dle struktury dat jsou DB rozděleny na dva typy a to na DB s uživatelem definovanou strukturou dat a DB s instanční datovou strukturou.

Do DB s uživatelsky vytvořenou strukturou může uživatel libovolně zasahovat (přidávat, odebírat i upravovat proměnné). Datové bloky tohoto typu zakládá vždy uživatel. Typickým využitím je například DB, které obsahuje vybraná data od jednotlivých zařízení na výrobní lince.

Naopak instanční DB je založen automaticky při zanesení volání FB do programu. Datový blok je instancí volaného FB a jeho struktura přesně odpovídá struktuře FB. Uživatel nemůže do struktury instančního DB nijak zasahovat. Pokud došlo ke změně FB je nutné dříve vygenerované DB aktualizovat, jinak dochází k chybě během kompilace uživatelského programu. Datový blok lze přiřadit pouze jedinému FB.

5.5 UDT

Blok s uživatelsky definovaným datovým typem (UDT) slouží k vytváření struktury pro data. V OB, FB, FC a DB potom namísto standardních datových typů přiřazujeme k proměnné název vybraného UDT. Využití UDT je výhodné při vícenásobném použití stejné datové struktury.

5.6 SFB

Systémové funkční bloky (SFB) a systémové funkce (SFC) jsou bloky (většinou je používáno souhrnné označení SFB), definované výrobcem CPU. Tyto bloky není třeba nahrávat do PLC, jelikož jsou v něm přímo zabudovány. Typ a počet SFB i SFC obsažených v PLC je závislý na použitém typu CPU. Proto není možná ani editace či vytváření vlastních bloků SFC a SFB. Do programu OB, FB a FC lze bloky SFC a SFB vkládat v editoru LAD/STL/FBD z panelu Program *elements/Libraries/Systém Function Blocks*.

Bloky SFC a SFB poskytují uživateli možnost, zasahovat do CPU, která není v programu jinak možná. Například s pomocí bloku SFC0 lze nastavit systémový čas CPU, SFC12 slouží k aktivaci/deaktivaci slave zařízení decentralizované periferie a SFB19 vyvolá studený nebo horký restart na vzdáleném zařízení. [11]

6 MEZISOUBOR PRO KOMPILÁTOR GENERATORTAGU

Kompilátor využívá pro práci s daty mezisouboru typu *.xls*. S mezisouborem bude nakládáno dle pokynů shrnutých v následujícím odstavci.

Ke každému zpracovávanému projektu bude vytvořen vlastní mezisoubor podle šablony Data.xls, která byla zkonstruována v rámci této diplomové práce. Přičemž jméno mezisouboru bude vždy odpovídat jménu konkrétního projektu ve STEP 7. Uživatelé budou do mezisouboru doplňovat vlastní data, dle vzorů popsaných v této kapitole. Před zpracováním dat z vybraného projektu bude připraven odpovídající mezisoubor.

Práci s mezisouborem lze popsat pěti kroky. V prvním kroku byl kopírováním a přejmenováním šablony Data.xls vytvořen mezisoubor k vybranému STEP 7 projektu. Během druhého kroku byla do mezisouboru doplněna uživatelská data (parametry generovaných souborů, typy objektů použité v projektu STEP 7, atd.). Ve třetím kroku byla generována kompilátorem *GeneratorTagu* data ze zdrojového souboru AWL do mezisouboru. Editace dat před generováním souborů pro import dat do WinCC V7.0 byla možná ve čtvrtém kroku prostřednictvím tabulkového editoru (uživatelský zásah přímo v mezisouboru). Posledním krokem bylo zpracování mezisouboru při generování souborů pro import tagů do WinCC V7.0.

Šablona mezisouboru *Data.xls* má pevně danou strukturu, tvořenou šesti listy, z nichž každý má jiný účel. Popis jednotlivých listů byl uveden v této kapitole.

6.1 List Param

List parametrů obsahuje údaje o členění dat do jednotlivých skupin zařízení a alarmů. Adresace zde závisí na množství zadaných dat. Mezi jednotlivé sekce byl pro zvýšení přehlednosti vložen jeden volný řádek. Na Obr. 6-1je příklad listu *Param*.

Sekce *Connection* obsahuje název projektu ve STEP 7, který je zároveň použit jako prefix (předpona) u názvu všech tagů.

V sekci *SymbolArea* jsou uvedeny názvy oblastí (*Area*) podle kterých je přiřazeno DB k zařízení. Pokud není přiřazeno žádné jméno oblasti, platí daná adresa DB pro objekty ve všech skupinách *Area*. Ke každému *SymbolArea* je přiřazeno číslo alarmu *AlarmNumber*, které udává, od jakého čísla jsou pro danou *Area* generovány alarmy.

Následující sekce obsahuje údaje o SQL serveru s databází WinCC. Pokud je jako server používán lokální počítač, zůstává název serveru nevyplněn. Parametr *User* udává uživatelské jméno, parametr *Password* pak odpovídající heslo.

Sekce *StructureEmpty* obsahuje jméno prázdné struktury, které je použito pro vygenerování instance. Poté je jméno objektu dostupné v linkovacím nástroji grafického editoru WinCC V7.0. Každá struktura má unikátní identifikační číslo (*ID*).

V sekci *AreaName* je definován seznam jmen *Area* včetně indexu jména *Text ID* a jeho symbolického označení *SymbolArea*.

Alarmy jsou rozděleny do tříd (*AlarmClass*), přičemž každá třída má vlastní identifikační číslo (*AlarmClassID*). Jednotlivé třídy mohou obsahovat několik typů (*AlarmType*), každý typ má opět vlastní identifikační číslo (*AlarmTypeID*).

Pokud jsou v projektu použity SFB (systémové funkční bloky) je třeba definovat jejich délky v sekci *AdditionalObjects*. Délka SFB je udána v bytech.

	A	B	C	D	E
1	**Connection**				
2	Test				
3					
4	**SymbolArea**	**AlarmNumber**			
5	Oblast	10000			
6	DefaultAlarmNumber	100000			
7					
8	**SqlServer**	**User**	**Password**		
9	XKUZ	WinCCConnect	heslo heslo		
10					
11	**StructureEmpty**	**ID**			
12	Empty	1032			
13					
14	**AreaName**	**Text ID**	**SymbolArea**		
15	Ovládání		LINKA1		
16	Navažovna		NAV		
17					
18	**AlarmClass**	**AlarmClassID**	**AlarmType**	**AlarmTypeId**	
19	Alarm	1	Alarm High	1	
20	Alarm	1	Alarm Low	2	
21	Warning	2	Warning High	19	
22	Warning	2	Warning Low	20	
23					
24	**AdditionalObjects**	**Bytes**			
25	ANY	10			
26	TIMER	2			
27					

Obr. 6-1 Mezisoubor Data.xls list Param

6.2 List Zdroj

Sloupce v tomto listu lze rozdělit do třech kategorií. V závorkách za označením jednotlivých sloupců bude vždy uvedena adresa sloupce například sloupec *Ofset* (sloupec B).

První kategorií jsou sloupce obsahující údaje o DB, ve kterém je tag uložen. Sloupec *DB* (A) obsahuje číslo datového bloku. Ve sloupci *Ofset* je zanesena adresa v daném DB. Jméno projektu, jehož je DB součástí, je uvedeno ve sloupci *Connection* (C). Tagy jsou rozděleny do skupin, dle části technologie do které patří, toto označení je zaneseno ve sloupci *Area* (D), Symbolická adresa tagu je uložena ve sloupci *Označení* (E).

V druhé kategorii jsou sloupce s počátečními hodnotami tagů, jako jsou komentáře ve sloupci *Popis funkce* (F), *Typ* (G), *Area* (H). Sloupce A až H jsou povinné.

Sloupce následující po sloupci *Area* (H) slouží pro úpravu alarmových hlášení. Úpravu jednoho alarmového hlášení je možné zapsat vždy ve třech sloupcích. U alarmu je možné nahradit text, třídu nebo typ. Program funguje tak, že prohledává list zdrojů a hledá v něm extenze tagů pro úpravu inicializační hodnoty (viz druhá kategorie sloupců v listu *Zdroj*). Pokud je nalezen sloupec, který ve třetím řádku začíná textem *$$ALARM$$*, je tento sloupec vyhodnocen jako počáteční oblast pro definici úprav alarmů.

V souboru má smysl zapsat za sebou tolik trojic sloupců pro úpravu alarmu, kolik má nejvíce alarmů libovolný objekt. Při zpracovávání jednoho objektu jsou jeho alarmy zpracovávány v pořadí, v jakém jsou zapsané při definici objektu. K náhradě dojde u toho alarmu, který má v listu zdrojů něco napsaného v příslušném pořadí sloupců.

Pokud jsou v poli textu anebo třídy alarmu libovolného sloupce napsány dva znaky mínus -- (Excel takto zapsaný text přeformátovává na ‘--), nebude daný alarm pro objekt vygenerován vůbec.

6.3 List ZdrojOut

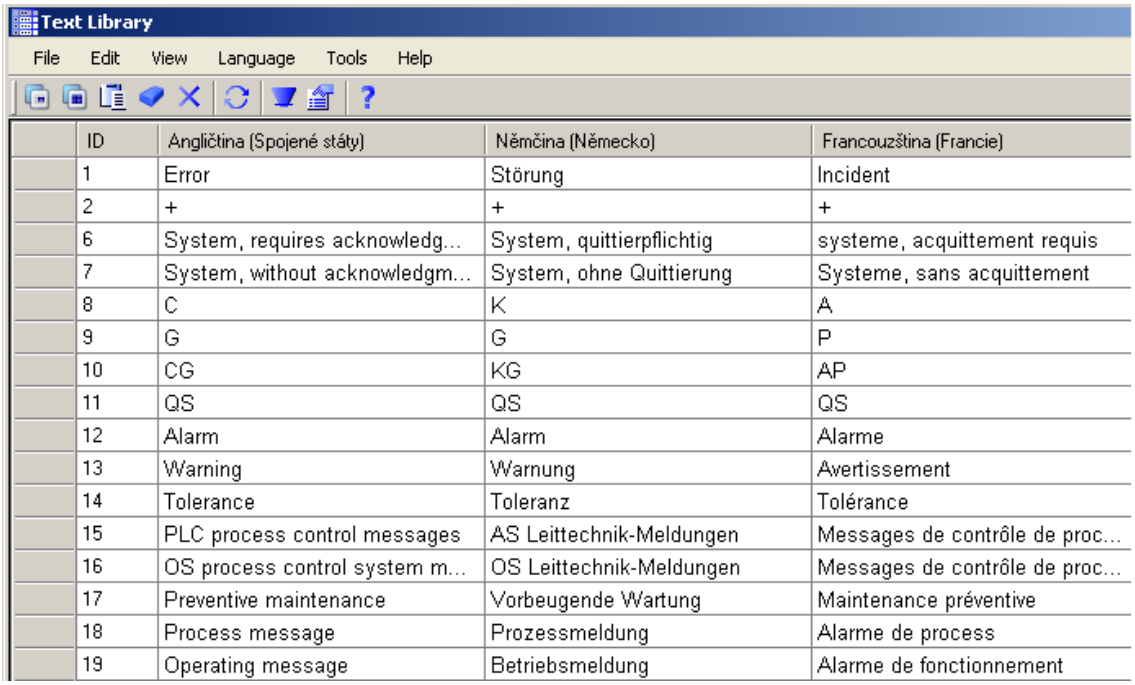
Jednou z funkčností grafického designeru WinCC V7.0 je generování objektů na obrazovku na základě souboru se seznamem objektů. List *ZdrojOut* bude sloužit jako soubor dat pro funkci, která vytvoří soubor ve formátu požadovaném WinCC V7.0 pro generování objektů na obrazovku. List má strukturu dle Obr. 6-2. Ve sloupci *Picturename* je uvedeno jméno cílové grafiky (soubory s příponou .pdl). Jméno objektu ve STEP 7 je zaneseno do sloupce *Objecttype*. Ve sloupci *Link* je třeba vyplnit jméno tagu, který bude nalinkován na daný objekt v grafice. Následující sloupce definují vlastnosti konkrétního objektu na vybrané obrazovce. Jméno objektu je ve sloupci *Objectname*, sloupce *X-Pos* a *Y-Pos* definují pozici na obrazovce a hodnoty ve sloupcích *Tag* a *Tagname* jsou zapsány do stejnojmenných vlastností objektu. V případě, že tag má být napojen na trend, je název tohoto trendu uveden ve sloupci *Trend*.

	A	B	C	D	E	F	G	H	I
1	Picturename	Objecttype	Link	Objectname	X-Pos	Y-Pos	Tag	Trend	Tagname
2	LINKA1.pdl	FB_AI	Zdroj/LI1001	Zdroj/LI1001	10	10	Zdroj/LI1001		Zdroj/LI1001
3	LINKA1.pdl	FB_AI	Zdroj/LINKA1_REG	Zdroj/LINKA1_REG	10	70	Zdroj/LINKA1_REG		Zdroj/LINKA1_REG
4	NAV.pdl	FB_AI	Zdroj/NAV_REG	Zdroj/NAV_REG	10	130	Zdroj/NAV_REG		Zdroj/NAV_REG
5	LINKA1.pdl	FB_MOTOR	Zdroj/LINKA1_REG	Zdroj/LINKA1_REG	10	190	Zdroj/LINKA1_REG		Zdroj/LINKA1_REG
6	NAV.pdl	FB_MOTOR	Zdroj/NAV_REG	Zdroj/NAV_REG	10	250	Zdroj/NAV_REG		Zdroj/NAV_REG
7	LINKA1.pdl	FB_MOTOR	Zdroj/LINKA1_MOT001	Zdroj/LINKA1_MOT001	10	310	Zdroj/LINKA1_MOT001		Zdroj/LINKA1_MOT001
8	NAV.pdl	FB_MOTOR	Zdroj/NAV_MOT002	Zdroj/NAV_MOT002	10	370	Zdroj/NAV_MOT002		Zdroj/NAV_MOT002
9	LINKA1.pdl	FB_PUMP	Zdroj/M002	Zdroj/M002	10	430	Zdroj/M002		Zdroj/M002
10									

Obr. 6-2 Mezisoubor Data.xls list ZdrojOut

6.4 List Lang

Součástí zadání projektu je často požadavek na vícejazyčné provedení. WinCC V7.0 umožňuje přepínání mezi jazyky uvedenými v textové tabulce (*Text Library*), příklad tabulky je na Obr. 6-3. Do tabulky lze přidávat další jazyky z databáze WinCC V7.0. Kompilátor bude obsahovat funkci pro vyčtení textové tabulky z WinCC V7.0. Vyčtené texty je třeba v listu přeložit do požadovaného jazyka (manuálně či pomocí funkce využívající webového překladače). List s doplněnými překlady lze exportovat nazpět do textové tabulky WinCC V7.0.



	ID	Angličtina (Spojené státy)	Němčina (Německo)	Francouzština (Francie)
	1	Error	Störung	Incident
	2	+	+	+
	6	System, requires acknowledg...	System, quittierpflichtig	systeme, acquittement requis
	7	System, without acknowledgm...	System, ohne Quittierung	Systeme, sans acquittement
	8	C	K	A
	9	G	G	P
	10	CG	KG	AP
	11	QS	QS	QS
	12	Alarm	Alarm	Alarme
	13	Warning	Warnung	Avertissement
	14	Tolerance	Toleranz	Tolérance
	15	PLC process control messages	AS Leittechnik-Meldungen	Messages de contrôle de proc...
	16	OS process control system m...	OS Leittechnik-Meldungen	Messages de contrôle de proc...
	17	Preventive maintenance	Vorbeugende Wartung	Maintenance préventive
	18	Process message	Prozessmeldung	Alarme de process
	19	Operating message	Betriebsmeldung	Alarme de fonctionnement

Obr. 6-3 Textová knihovna projektu ve WinCC V7.0

Hlavička listu *Lang* se skládá ze dvou řádků. V prvním řádku je zapsáno jméno jazyka a jeho kód v databázi WinCC V7.0. Ve druhém řádku je zapsán pouze kód jazyka v databázi WinCC V7.0.

První sloupec listu obsahuje čítač použití textu. Čítače jsou vynulovány při spuštění funkce, které textovou tabulku používají. Následně v průběhu práce příslušné funkce jsou čítače inkrementovány v případě použití daného textu. Po skončení funkce je možné vidět, které texty už nejsou v projektu využívány. Řádky s nepoužitými texty lze z textové tabulky odstranit.

Druhý sloupec listu obsahuje *Referenční jazyk*. Je nutné, aby tento jazyk byl i druhým sloupcem v textové tabulce WinCC V7.0. Funkce pro export bude postupně číst texty z referenčního jazyka. Tyto texty bude vyhledávat v databázi WinCC a pokud je najde, tak do ostatních jazyků daného textu v databázi WinCC přepíše texty, které jsou zapsané v listu Lang mezisouboru *Data.xls*. Ukázka listu Lang je na Obr. 6-4.

	A	B	C	D	E
1	Jazyk:	Angličtina 1033	Němčina 1031	Francouzština 1036	
2	WinCC ID:	1033	1031	1036	
3	13	Warning	Warnung	Avertissement	
4					
5					

Obr. 6-4 Mezisoubor Data.xls list Lang

6.5 List TagTypes

V listu *TagTypes* byly zaneseny tabulky s definicemi formátu tagů ve WinCC V7.0 dle jejich datového typu. Tabulka na řádcích číslo jedna až dvacet jedna definuje formát dat tagu s klasickým datovým typem. Druhá tabulka (řádky dvacet sedm až čtyřicet jedna) obsahuje definice pro tagy zanořené ve strukturách (datový typ STRUCT). Tabulky zobrazené na Obr. 6-5 a Obr. 6-6ukazují pouze datové typy pro tagy ve WinCC V7.0 a několik jejich parametrů. Do kompletní tabulky lze nahlédnout v Příloze č. 1 (šablona mezisouboru *Data.xls*)

	B	C	D	E	F	G	H	I
1	'Configuration Param	Decimal separ	Layout Version V5.1					
2	'Varname	Conn	Group	Spec	Flag	CTyp	CLen	CPro
3	ValueBool	Conn	Group	*	0	1	1	4
4	ValueByte	Conn	Group	*	0	3	1	4
5	ValueDint	Conn	Group	*	0	6	4	4
6	ValueDword	Conn	Group	*	0	7	4	4
7	ValueReal	Conn	Group	*	0	8	4	4
8	ValueReal	Conn	Group	*	0	8	4	4
9	ValueInt	Conn	Group	*	0	4	2	4
10	ValueString	Conn	Group	*	0	10	255	4
11	VaueStruct	*	Struct	*	0	1025	0	2
12	ValueTextRef	Conn	Group	*	0	18	4	4
13	ValueWord	Conn	Group	*	0	5	2	4
14	ValueString[4]	Conn	Group	*	0	10	4	4
15	ValueString[12]	Conn	Group	*	0	10	12	4
16	ValueString[16]	Conn	Group	*	0	10	16	4
17	ValueDint[3]	Conn	Group	*	0	8	4	4
18	ValueDint[3]	Conn	Group	*	0	8	4	4
19	ValueDint[3]	Conn	Group	*	0	8	4	4
20	ValueRAW	Conn	*	RAW_BSEND(DE	0	12	36	4
21	ValueBlockDb	Conn	Group	*	0	4	2	4
22								

Obr. 6-5 Definice datových typů pro tagy ve WinCC V7.0

27	'Structure Name	Type-ID	Creator-ID	Project path				
28	StructMie	1026	0					
29	'Varname	C.Vartype	C.CreatorI	C.VarLength	C.VarProperty	C.Format	C.OsOffse	C.AsOffse
30	ValueBool	1	0	2	2	0	0	0
31	ValueByte	3	0	1	2	0	0	0
32	ValueDint	6	0	4	2	0	0	0
33	ValueDword	7	0	4	2	0	0	0
34	ValueReal	8	0	4	2	0	0	0
35	ValueReal	9	0	8	2	0	0	0
36	ValueInt	4	0	2	2	0	0	0
37	ValueString	10	0	254	2	0	0	0
38	ValueWord	5	0	2	2	0	0	0
39	ValueTextRef	18	0	0	2	255	0	0
40	ValueDint	6	0	4	2	0	0	0
41	ValueBlockDb	4	0	2	2	0	0	0

Obr. 6-6 Definice datových typů pro WinCC V7.0 tagy zanořené v strukturách

6.6 List Obj>

List *Obj*> byl naplněn definicemi jednotlivých typů objektů. Definice objektu začíná řádkem, kde první buňka obsahuje text *Configuration Parameter:*. Druhá buňka byla vyhrazena pro název objektu, třetí buňka obsahuje unikátní identifikační číslo struktury tagů (objekt bylo pro WinCC V7.0 nutné reprezentovat strukturou tagů) daného objektu. Následující řádek listu obsahuje hlavičku tabulky pro definici parametrů objektu (tagů obsažených ve struktuře tagů). Tag reprezentující parametru objektu ve WinCC V7.0 byl navržen podle šablony *objekt.parametr* (k parametru FP_STAT objektu FB_MOTOR přistupuje ve WinCC V7.0 tag FB_MOTOR.FP_STAT. Dle typu parametru bylo nutné nadefinovat další vlastnosti jako délku (parametr typu *STRING*) nebo adresu (u tagů přenášejících hodnoty typu *DWORD, REAL, BOOL,FLOAT*, atd.). K objektu lze definovat do vyhrazené oblasti i alarmy. Definice alarmu zahrnuje jméno (*Name*), zobrazovaný text (*Text*), třídu (*AlarmClass*), typ (*AlarmType*) a názvy tagů pro jednotlivé parametry. Příklad definice objektu je zobrazen na Obr. 6-7.

	A	B	C	D	E
1	'Configuration Parameter:	FB_MOTOR	1002		
2	Varname	Typ	Length	Value	Address
3	.#text	STRING	20		
4	.#area	TEXT-REF			
5	.#comment	STRING	1		
6	.#areaname	TEXT-REF			
7	.FP_STAT	DWORD			DD2
8	.FP_CMD	DWORD			DD4
9					
10	AlarmList	Bit	Text	AlarmType	PV2
11	$$OBJNAME$$.FP_STAT	28	@10%s@ - Aktivní simulace	#ODKAZ!	
12					
13	'Configuration Parameter:	FB_AI	1003		
14	Varname	Typ	Length	Value	Address
15	.#text	TEXT-REF	20		
16	.Q_PV#unit	STRING	20		
17	.#area	TEXT-REF			
18	.#comment	STRING	1		
19	.#areaname	TEXT-REF			
20	.AI	STRING	1		
21	.Q_PV	FLOAT			DD2
22	.LL_ALM	FLOAT			DD6
23					
24	AlarmList	Bit	Text	AlarmType	PV2
25	$$OBJNAME$$.FP_STAT	5	@10%s@ - Alarm LL (PV=@2%g@ LL	#ODKAZ!	$$OBJNAME$$.Q_PV
26	$$OBJNAME$$.FP_STAT	5	$$OBJNAME$$ $$COMMENT$$ $$DE	#ODKAZ!	$$OBJNAME$$.Q_PV
27	$$OBJNAME$$.FP_STAT	5	$$OBJNAME$$ $$VALUE$$Text1$$ -	#ODKAZ!	$$OBJNAME$$.Q_PV

Obr. 6-7 Mezisoubor Data.xls list Obj>

6.7 List KorekceTextu

Kompilátor obsahuje funkce pro tvorbu pomocných tagů (interní tagy), které budou využívány například k ukládání textů příkazových tlačítek ve faceplate. Funkce bude možné použít i pro modifikaci alarmových hlášení.

Tyto funkce používají jako databázi list *KorekceTextu*. V prvním sloupci je zapsáno jméno skupiny. Následuje libovolné množství sloupců pro definici tagů. První řádek takového sloupce je označen řetězcem *TAG*, druhý řádek obsahuje název tagu. Skupina je definována na jednom řádku listu. Tagy pro skupinu budou vytvořeny jen pro sloupce, kde je vyplněn text. Za sloupci definujícími tagy následují sloupce pro definici

úpravy textu alarmu. Alarm je definován dvěma sloupci. První řádky obou sloupců jsou shodně označeny jako *ALARM*, druhý řádek prvního sloupce obsahuje text alarmu pro objekt definovaný v listu *Obj*>. Ve druhém sloupci je zapsán nový text alarmu.

6.8 List Dialog

Tento list slouží pro ukládání dat, která kompilátor *GeneratorTagu* načítá do interface během spouštění. Zejména se jedná o jména symbolické tabulky a zdrojového souboru AWL. Dalšími hodnotami jsou jména souborů pro import tagů do WinCC V7.0. Rovněž sem byly zavedeny parametry pro nastavení kompilátoru *GeneratorTagu* (například parametr určující zda budou výstupní soubory ukládány). List *Dialog* pro navrženou šablonu je zobrazen na Obr. 6-8.

	A	B
1	Folder:	.\Zdroj\
2	Read sources:	PRAVDA
3	AWL sources:	zdroj.AWL
4	Symbol table ASC	symbol.sdf
5	Save output files:	PRAVDA
6	Compare output files:	PRAVDA
7	Tags:	Data_vex.csv
8	Alarms:	Data_alarm.txt
9	Structures:	Data_dex.csv
10	Generate structures:	PRAVDA
11	Style SMS:	NEPRAVDA
12	Don't use Area as text Ref.:	NEPRAVDA
13	Don't use point in DBB tag:	NEPRAVDA
14	Tag Master Type:	.#MasterType
15	Tag konfigurace faceplate:	AI
16	Rozsirena konfigurace faceplate:	PRAVDA
17	Korekce příkazových textů:	1
18	Soubor pro lbu:	PRAVDA
19	Skupina pro NE MULTI instanci:	4

Obr. 6-8 Šablona listu Dialog

7 PŘEDZPRACOVÁNÍ PROJEKTU STEP 7 PRO KOMPILÁTOR *GENERATORTAGU*

Jako zdrojový soubor kompilátor *GeneratorTagu* využívá zdrojový soubor AWL s celým projektem vygenerovaný v STEP 7. *GeneratorTagu* z tohoto AWL souboru zpracovává zdrojový kód FB, ve kterých jsou definované zařízení a dále všechny vnořené FB a UDT, které jsou do těchto prvních FB vnořené (vnoření na úrovni VAR_INPUT, VAR_OUTPUT, VAR).

7.1 Zpracování programových bloků v jazyce SCL

Automatická funkce STEP 7 do zdrojového AWL souboru není schopna generovat bloky, které jsou napsány v SCL. Proto žádný z bloků (FB, UDT), který je kompilátorem *GeneratorTagu* používán nesmí být napsán v programovacím jazyce SCL. Pokud jsou přesto součástí STEP 7 projektu bloky napsané v SCL, potom si zdrojový blok s kódem v SCL, musí uživatel uchovat v knihovně, kterou následně přidá k celému projektu. V samotné struktuře projektu musí být proveden převod bloku do jazyka STL. Tento převod lze provést v LAD/STL/FDB editoru. Otevřením daného bloku dojde k automatickému přepisu SCL kódu do jazyka STL. Takto převedený blok bude následně uložen standardním postupem.

7.2 Podporovaná struktura bloků v projektu STEP 7

Kompilátor *GeneratorTagu* během načtení zdrojového souboru AWL funguje rekurzivně a z obecného pohledu detekuje AWL zdrojový kód STEP 7 programu. Základní objektovou jednotkou je typ STRUCT. Objekt typu STRUCT může obsahovat:

- Základní datové typy (BOOL, INT, REAL, STRING)
- Vložení bloku (UDT, FB s definicí zařízení)
- ARRAY ze základních datových typů (kromě STRING[x])
- ARRAY z vložených bloků UDT, FB

STRUCT dále může obsahovat zanoření:

- Zanořený obyčejný STRUCT (tento zanořený STRUCT opět má všechny vlastnosti tak, jak bylo obecně definováno pro STRUCT)
- ARRAY složené ze STRUCT (opět tento zanořený STRUCT má všechny vlastnosti objektu STRUCT)

GeneratorTagu potom používá při načítání:

- UDT – načítá obyčejný STRUCT
- FBdev – jednotlivé karty VAR_xxx jsou objekty typu STRUCT
- DB – načítá STRUCT

Parametr *WCC* používaný pro označení proměnné jako tag pro WinCC V7.0 funguje rovněž i v zanořené struktuře. To znamená, že lze označit potřebné tagy v UDT parametrem *WCC*. Pokud potom toto UDT bude voláno s parametrem *WCC*, tak budou jednotlivé tagy z tohoto UDT vygenerovány pro WinCC V7.0.

7.3 Označování dat

Předávání veškerých proměnných použitých v projektu STEP 7 do aplikace vytvořené ve WinCC V7.0 by bylo náročné jak hardwarově, tak i finančně. Cena licence WinCC V7.0 byla totiž odvozena od množství předaných proměnných (tagů). Z tohoto důvodu vybírá programátor data určená pro přenos do WinCC V7.0 ručně. V kompilátoru *GeneratorTagu* bylo k označování vybraných dat použito klíčového slova *WCC*. Během zpracování zdrojového souboru AWL *GeneratorTagu* detekoval klíčové slovo *WCC* a označená data byla zanesena do listu Zdroj v mezisouboru.

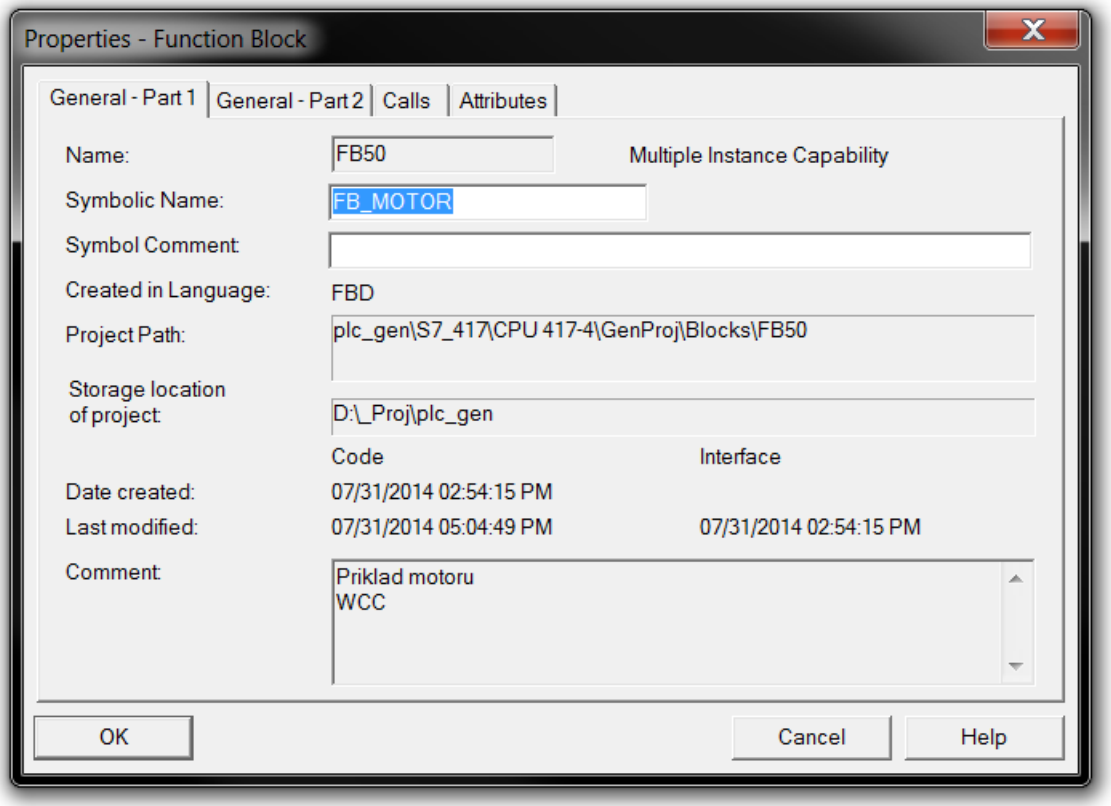
7.4 Syntaxe pojmenovávání a komentování programových bloků ve STEP 7

Kompilátor *GeneratorTagu* byl navržen tak aby čerpal veškerá data potřebná pro generování tagů (určených k importu do WinCC V7.0) z mezisouboru typu .xls. Část dat pro tento úkon doplňuje do mezisouboru uživatel, zbylá data do mezisouboru byla generována kompilátorem *GeneratorTagu*. Aby byla data vyčtená ze zdrojového souboru AWL (kompilátorem *GeneratorTagu*) kompatibilní s daty která do mezisouboru zadal uživatel, bylo třeba zavést syntaxi pojmenovávání programových bloků v zpracovávaném STEP 7 projektu.

7.4.1 FB pro definici zařízení

Jednotlivá zařízení byla popsána funkčním blokem, který obsahuje definici parametrů zařízení a program pro ovládání zařízení (podmínky chodu, přepis příkazového slova do stavového slova, přepínání mezi režimy daného zařízení, atd.). Každému FB s definicí zařízení odpovídá záznam v mezisouboru (definice typu objektu v listu Obj>). Spojení mezi FB a definicí objektu v mezisouboru bylo navázáno kompilátorem *GeneratorTagu* prostřednictvím shodného názvu objektu a symbolickým jménem (*Symbolic Name*) FB s definicí zřízení. Komentář FB zařízení (*Comment*) nesmí obsahovat na začátku

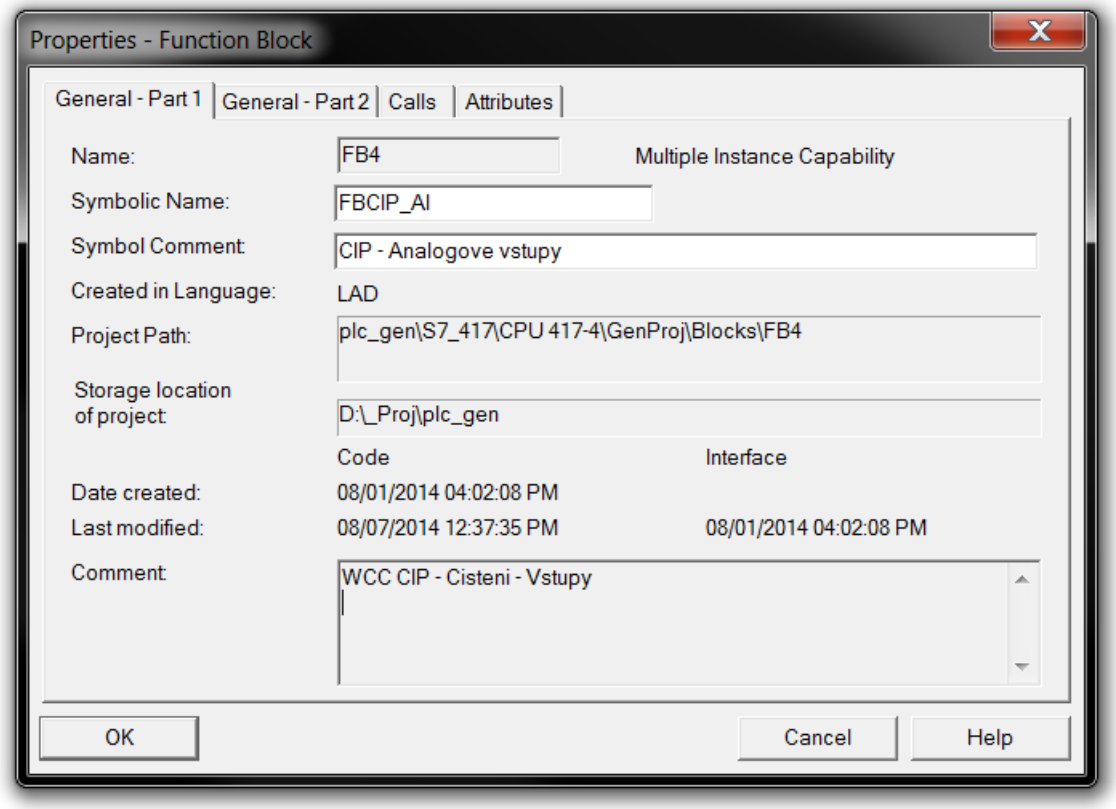
prvního řádku klíčové slovo *WCC* (vyhrazeno pro identifikaci FB, které obsahuje volání FB s definicí zařízení. Příklad nastavení byl uveden na Obr. 7-1.



Obr. 7-1 Příklad nastavení FB s definicí zařízení

7.4.2 FB pro skupinu zařízení

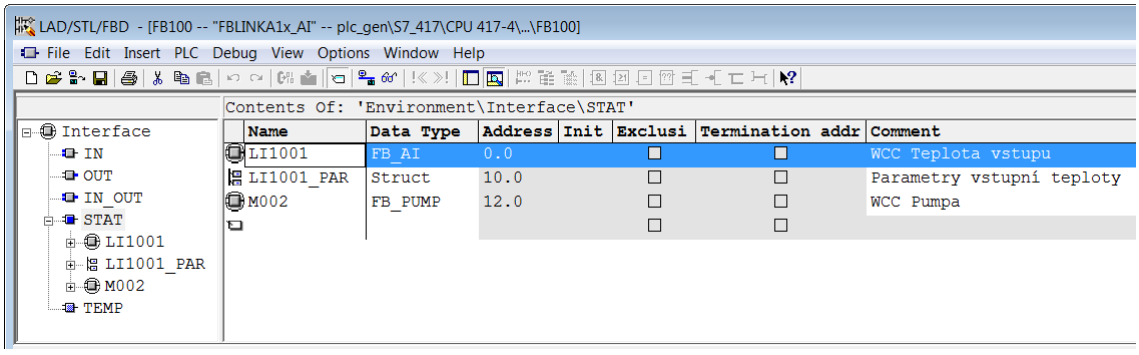
Funkční blok, který obsahuje volání více FB zařízení. Zařízení byla rozdělena podle typu (například souhrnný FB pro všechny motory v projektu STEP 7) nebo dle vybraného úseku technologie (např. ventily a čerpadla na jedné nádrži čističky odpadních vod). Symbolické jméno volíme libovolně, kompilátor *GeneratorTagu* totiž pracuje s připojeným DB (v případě instancování i s více DB). Jméno DB musí být vytvořeno dle šablony *DBx_y*, kde *x* představuje jméno skupiny a *y* je volitelná část jména. Dle jména *x* je proveden převod jména skupiny na jméno Area (mezisoubor, list *Param*, sekce *AreaName).* Pole komentáře *Comment* FB pro skupinu zařízení začíná vždy klíčovým slovem *WCC*. Pokud byl k bloku zároveň přiřazen běžný komentář, pak byl od klíčového slova *WCC* oddělen pomlčkou. Příklad nastavení FB pro skupinu zařízení je uveden na Obr. 7-2.



Obr. 7-2 Příklad nastavení FB pro skupinu zařízení

7.4.3 Volání FB zařízení ze skupinového FB bez možnosti instancování

Obrázek Obr. 7-3 zobrazuje příklad volání FB zařízení z FB pro skupinu zařízení (bez možnosti instancování). Dle nastavených parametrů v zobrazeném příkladu bude do mezisouboru (list *Zdroj*) vygenerován objekt jménem *LI1001* s datovým typem *FB_AI* a objekt *M002* typu *FB_PUMP*.



Obr. 7-3 Příklad volání FB zařízení z FB pro skupinu zařízení (bez možnosti instancování)

Výraz *WCC* na začátku komentáře identifikuje, že se jedná o objekt, ke kterému mají být generovány tagy a alarmy. Z komentáře vytvořeného objektu, který je kompilátorem dále využíván jsou znaky *WCC* vypuštěny. Pokud komentář nebude na

začátku obsahovat klíčové slovo *WCC*, ke generování tagů a alarmů od daného objektu nedojde. Kompilátor *GeneratorTagu* ale vyhodnotí velikost datové oblasti zabrané tímto objektem a u případných dalších objektů adresy posune.

V poli komentáře *Comment* mohou být znaky *WCC* doplněny dalším textem dle vzoru *WCCx*. Část *x* je ukončena mezerou. Kompilátor následně rozšíří jméno typu objektu v seznamu zdrojů v mezisouboru o text *x*. Objekt bude mít tedy jméno typu *FB_AIx*.

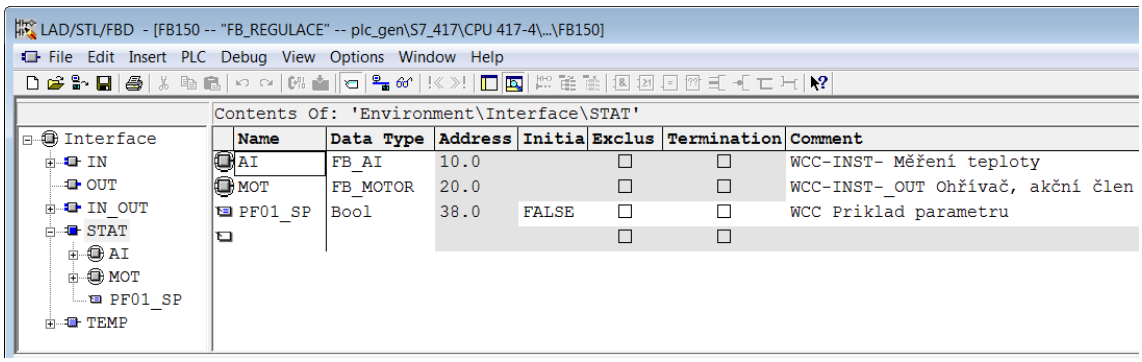
V seznamu objektů lze definovat i proměnnou standardního datového typu (INT, REAL, STRING, atd.). Takováto proměnná je vygenerovaná jako parametr.

Mezi statickými parametry FB mohou být definovány i pole základních datových typů (ARRAY) a proměnné typu UDT. Z těchto proměnných nebudou kompilátorem *GeneratorTagu* generovány tagy, dojde pouze k posunu adresy v závislosti na délce dané proměnné.

7.4.4 Volání FB zařízení ze skupinového FB s možnosti instancování

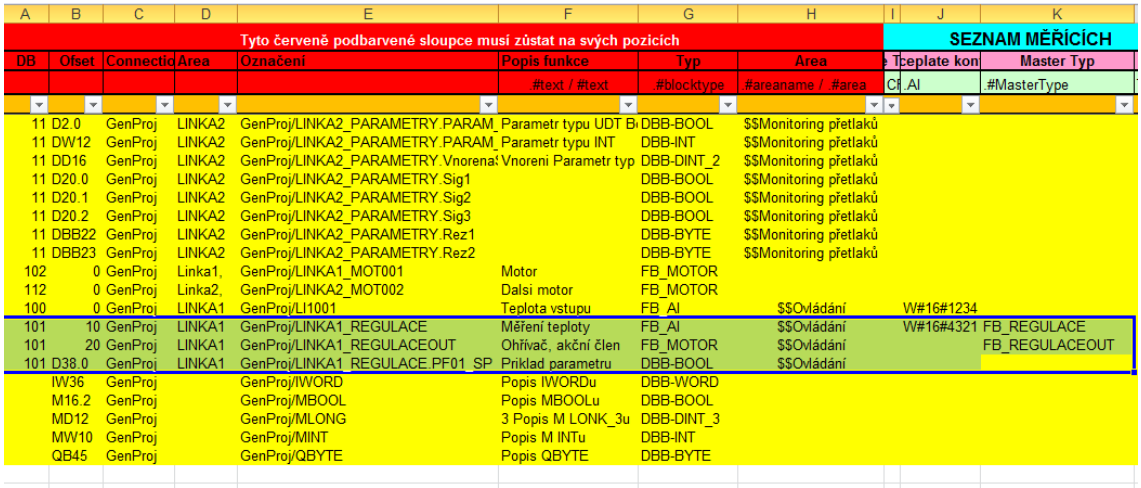
Při využití FB s možností instancování je třeba rozlišovat, ze kterého DB byla proměnná volána. Z tohoto důvodu nelze odvozovat jméno tagu od jména FB (docházelo by k vytváření tagů s duplicitními jmény). Pro vytvoření jména tagu bylo tedy využito jméno instančního DB.

Klíčovým slovem pro generování tagu z proměnné uložené v instančním DB je slovo *WCC-INST-*. Do pole *Comment* lze zapisovat dle vzoru *WCC-INST-x y*, kde *x* odpovídá textu, který bude přidán na konec jména tagu. Text *x* je nepovinnou částí jména tagu. Standardní komentář parametru tvoří část *y*. V příkladu uvedeném na Obr. 7-4 byl k uvedenému *FB150* přiřazen blok *DB101* (symbolické jméno *LINKA1_REGULACE*). Dojde tedy ke generování tagů LINKA1_REGULACE (komentář *Měření teploty*) a *LINKA1_REGULACE_OUT* (komentář *Ohřívač, akční člen*). Parametru PF01_SP se instancování netýká (datový typ BOOL), proto bude tag vygenerován jako běžný parametr, tedy *LINKA1_REGULACE.PF01_SP*.



Obr. 7-4 Příklad volání FB zařízení z FB pro skupinu zařízení s možností instancování

Jméno FB, který zavolal instanční DB, bylo uloženo kompilátorem *GeneratorTagu* do mezisouboru (list *Zdroj* sloupec *Master Typ*). Záznam odpovídající příkladu z Obr. 7-4 je zobrazen na Obr. 7-5 (kurzorem zvýrazněné řádky tabulky zdrojů).



A	B	C	D	E	F	G	H	I	J	K
Tyto červeně podbarvené sloupce musi zůstat na svých pozicích									SEZNAM MĚŘÍCÍCH	
DB	Ofset	Connectio	Area	Označení	Popis funkce	Typ	Area		Teplate kon	Master Typ
					#text / #text	#blocktype	#areaname / .#area		Cl .AI	#MasterType
11	D2.0	GenProj	LINKA2	GenProj/LINKA2_PARAMETRY.PARAM_	Parametr typu UDT Bi	DBB-BOOL	$$Monitoring přetlaků			
11	DW12	GenProj	LINKA2	GenProj/LINKA2_PARAMETRY.PARAM_	Parametr typu INT	DBB-INT	$$Monitoring přetlaků			
11	DD16	GenProj	LINKA2	GenProj/LINKA2_PARAMETRY.Vnorena	Vnoreni Parametr typ	DBB-DINT_2	$$Monitoring přetlaků			
11	D20.0	GenProj	LINKA2	GenProj/LINKA2_PARAMETRY.Sig1		DBB-BOOL	$$Monitoring přetlaků			
11	D20.1	GenProj	LINKA2	GenProj/LINKA2_PARAMETRY.Sig2		DBB-BOOL	$$Monitoring přetlaků			
11	D20.2	GenProj	LINKA2	GenProj/LINKA2_PARAMETRY.Sig3		DBB-BOOL	$$Monitoring přetlaků			
11	DBB22	GenProj	LINKA2	GenProj/LINKA2_PARAMETRY.Rez1		DBB-BYTE	$$Monitoring přetlaků			
11	DBB23	GenProj	LINKA2	GenProj/LINKA2_PARAMETRY.Rez2		DBB-BYTE	$$Monitoring přetlaků			
102	0	GenProj	Linka1,	GenProj/LINKA1_MOT001	Motor	FB_MOTOR				
112	0	GenProj	Linka2,	GenProj/LINKA2_MOT002	Dalsi motor	FB_MOTOR				
100	0	GenProj	LINKA1	GenProj/LI1001	Teplota vstupu	FB_AI	$$Ovládání		W#16#1234	
101	10	GenProj	LINKA1	GenProj/LINKA1_REGULACE	Měření teploty	FB_AI	$$Ovládání		W#16#4321	FB_REGULACE
101	20	GenProj	LINKA1	GenProj/LINKA1_REGULACEOUT	Ohřívač, akční člen	FB_MOTOR	$$Ovládání			FB_REGULACEOUT
101	D38.0	GenProj	LINKA1	GenProj/LINKA1_REGULACE.PF01_SP	Priklad parametru	DBB-BOOL	$$Ovládání			
	IW36	GenProj		GenProj/IWORD	Popis IWORDu	DBB-WORD				
	M16.2	GenProj		GenProj/MBOOL	Popis MBOOLu	DBB-BOOL				
	MD12	GenProj		GenProj/MLONG	3 Popis M LONK_3u	DBB-DINT_3				
	MW10	GenProj		GenProj/MINT	Popis M INTu	DBB-INT				
	QB45	GenProj		GenProj/QBYTE	Popis QBYTE	DBB-BYTE				

Obr. 7-5 Příklad záznamu dat pro proměnnou uloženou v instančním DB

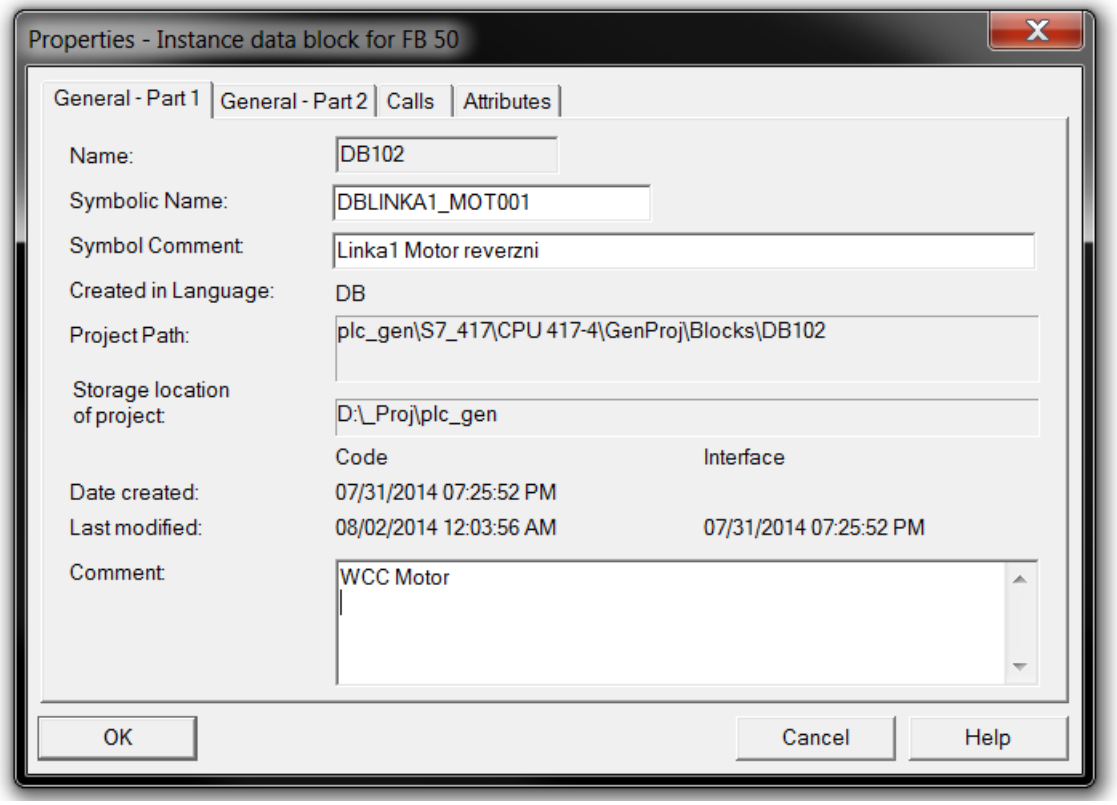
7.4.5 Přímé volání FB zařízení (bez volání v instančním FB)

Funkční blok s definicí zařízení lze volat z FC tak aby byly generovány tagy při dodržení následujících zásad.

Komentář přidruženého DB odpovídal šabloně *WCCx y*, kde *x* tvořilo rozšíření jména typu objektu v seznamu zdrojů mezisouboru (list *Zdroj*, sloupec *Typ*) a *y* obsahovalo standardní komentář pro DB.

Jméno pro tag ve WinCC V7.0 bylo kompilátorem *GeneratorTagu* odvozeno ze symbolického jména DB (symbolické jméno bez prvních dvou znaků). Z pole symbolického komentáře *Symbol Comment* DB bylo vyčteno jméno skupiny (text od prvního znaku po první mezeru).

Příklad korektního nastavení FB zařízení pro přímé volání z FC je uveden na Obr. 7-6. Dle příkladu bude vytvořen tag *LINKA1_MOT001* ve skupině *Linka.1*



Obr. 7-6 Příklad přímého volání FB zařízení z FC

7.4.6 DB s parametry

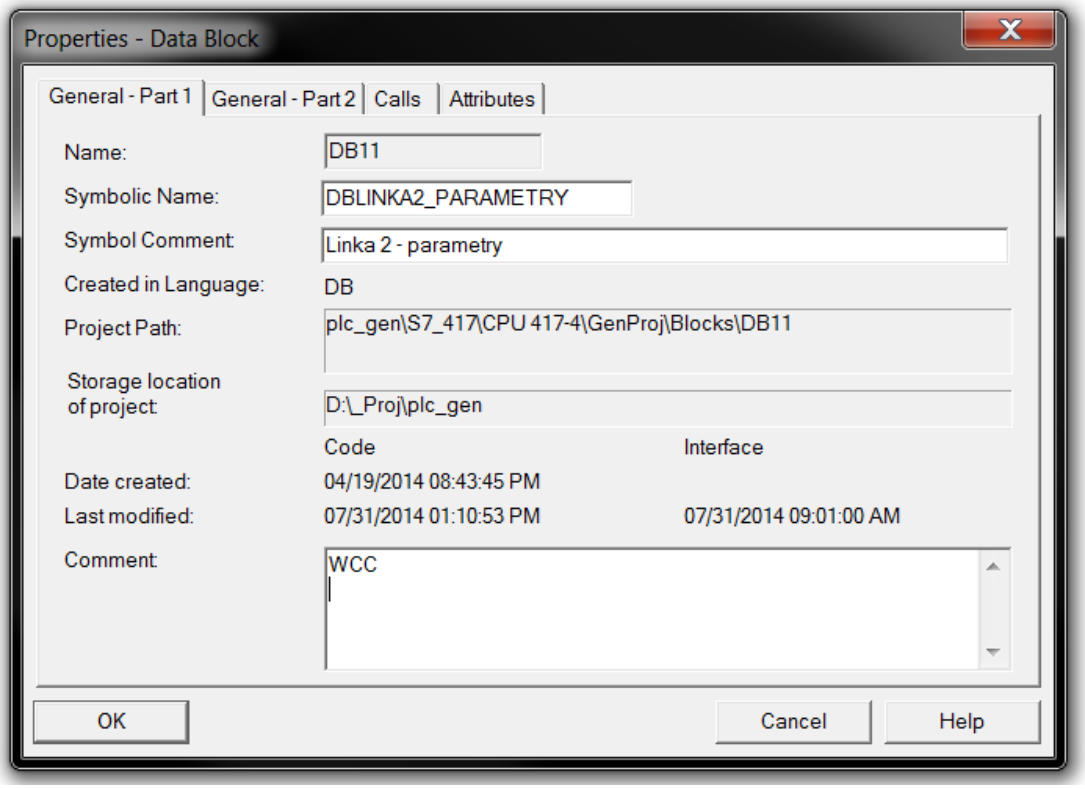
Syntaxe pro symbolické jméno DB s parametry byla stanovena vzorem *DBa_b*. Text označen písmenem *a* byl definován jako jméno skupiny parametrů. Kombinace textů *a* a *b* (včetně podtržítka) byla využita pro jméno tagu jednoho parametru ve WinCC V7.0.

Komentář DB (textové pole *Comment*) musí začínat klíčovým slovem *WCC*, tím je identifikováno DB, které obsahuje parametry, ze kterých se mají generovat kompilátorem *GeneratorTagu* tagy pro WinCC V7.0. Z komentáře vytvořeného objektu, který je kompilátorem dále využíván, byly znaky *WCC* vypuštěny.

Příklad syntaxe pro DB s parametry je zobrazen na Obr. 7-7 a Tagy vytvořené tímto DB budou umístěny do skupiny LINKA2 a jejich jména budou začínat textem LINKA2_PARAMETRY.

Pokud jsou v DB definovány parametry s datovým typem STRUCT budou jména tagů ve struktuře generována dle vzoru *Jméno_DB.cd*. Část *c* obsahuje jméno struktury, do které byl tag zanořen a podtržítko. Pokud je zanořeno více struktur do sebe, obsahuje tato část jména všech struktur oddělených podtržítkem. Text *d* představuje jméno zanořené proměnné. Na Obr. 7-8 je uveden příklad DB s proměnnými typu STRUCT. Jméno tagu pro parametr *DW01* ze struktury *VnorenaStruktura* definované v uvedeném příkladu DB bude *LINKA2_PARAMETRY.VnorenaStruktura_DW1*.

Datový typ tagu je odvozen z komentáře proměnné dekódovaného podle vzoru *WCCg h*. V tomto vzoru představuje *h* standardní komentář proměnné a *g* odpovídá textu, který bude doplněn za jméno tagu a zároveň tvoří část datového typu proměnné dle vzoru *DBB-fg*. Znak *f* ve vzoru datového typu bude nahrazen datovým typem proměnné.



Obr. 7-7 Příklad syntaxe pro DB s parametry

DB11 -- "DBLINKA2_PARAMETRY" -- plc_gen\S7_417\CPU 417-4\...\DB11

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	PARAM_UDT	"UDT_B8"		WCC Parametr typu UDT
+4.0	DTx	DATE_AND_TIME	DT#90-1-1-0:0:	
+12.0	PARAM_INT	INT	0	WCC Parametr typu INT PARAM_INT : INT ; //WCC Parametr typu INT
+14.0	VnorenaStruktura	STRUCT		WCC Vnoreni
+0.0	B01	BOOL	FALSE	
+2.0	DW01	DINT	L#0	WCC_2 Parametr typ DWORD
=6.0		END_STRUCT		
+20.0	Sig	ARRAY[1..3]		WCC
*0.1		BOOL		
+22.0	Rez	ARRAY[1..2]		WCC
*1.0		BYTE		
=24.0		END_STRUCT		

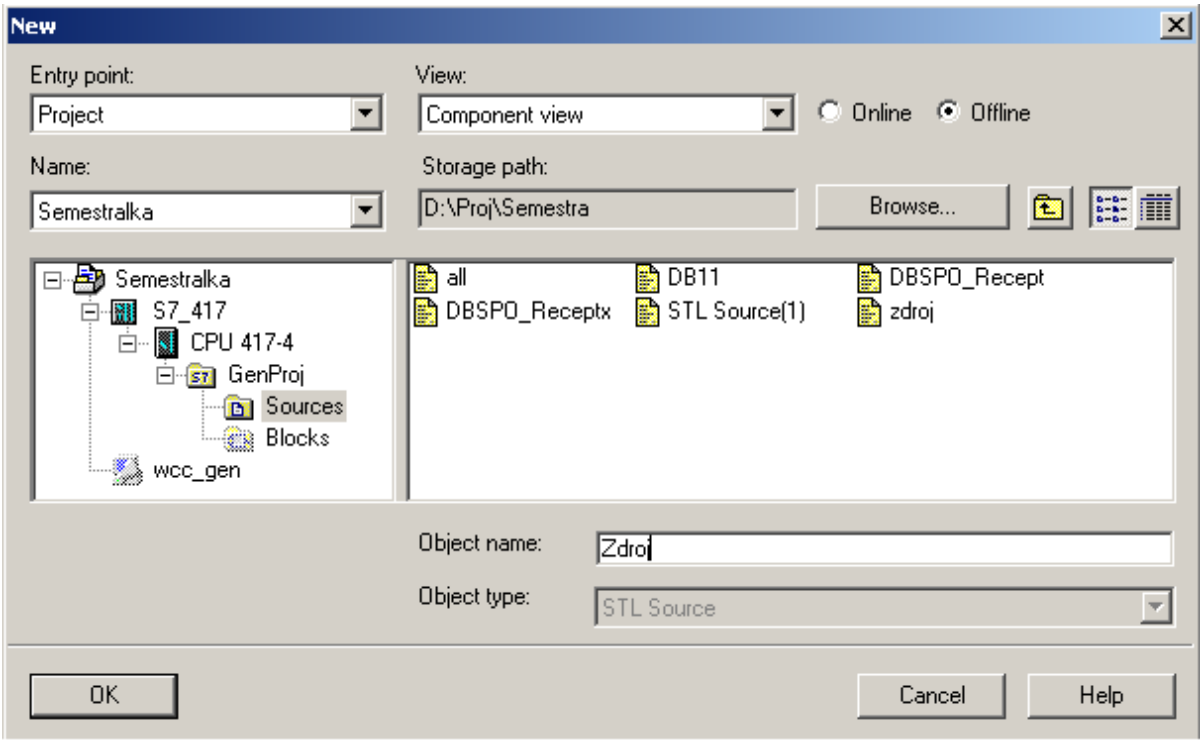
Obr. 7-8 Příklad datové struktury DB s parametry

8 KOMPILÁTOR GENERATORTAGU

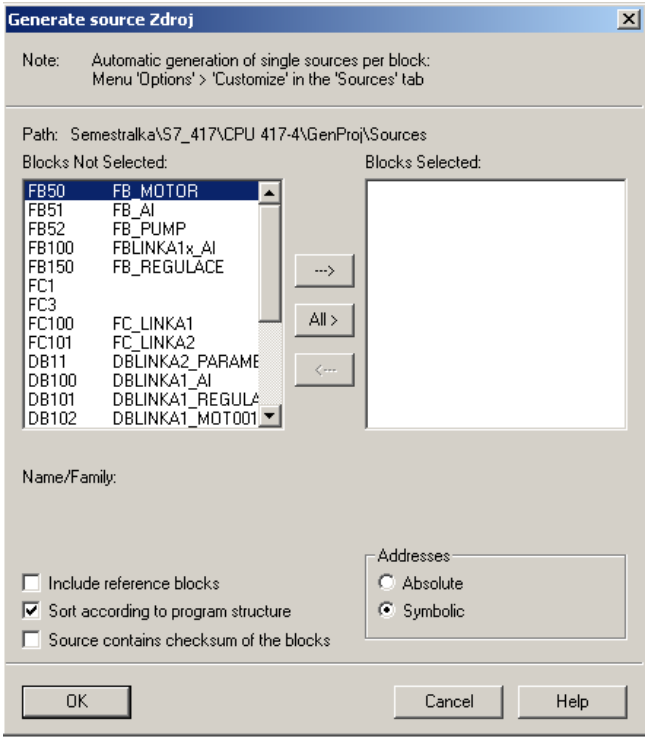
Kapitola obsahuje popis a návod pro obsluhu kompilátoru zdrojového kódu pro SIMATIC PLC. Kromě práce s kompilátorem *GeneratorTagu* byly popsány i postupy generování zdrojových souborů z projektu STEP 7 a importování tagů do WinCC V7.0.

8.1 Generování zdrojového souboru STEP 7

V *SIMATIC Manageru* je třeba otevřít projekt, ze kterého budou exportovány zdrojové soubory pro kompilátor. Spustíme *LAD/STL/FDB editor* (otevření libovolného programového bloku v projektu STEP 7). V *LAD/STL/FDB editoru* prostřednictvím klávesové zkratky *Ctrl+T*, nebo volbou *File/Generate Source…* otevřeme dialog pro generování zdrojového souboru. V dialogovém okně zobrazeném na Obr. 8-1. lze nastavit název souboru se zdrojovým kódem. Po zvolení názvu souboru stiskneme tlačítko *OK*. Tímto přejdeme k dialogovému oknu, které bylo zachyceno na Obr. 8-2. Dialogové okno slouží pro výběr bloků, ze kterých bude generován zdrojový soubor. V našem případě je vhodné vybrat všechny bloky (tlačítko *All* >). Z dalších dostupných voleb vybereme *Sort according to program structure* a adresování *Symbolic*. Stiskem tlačítka *OK* provedeme vlastní generování zdrojového souboru.

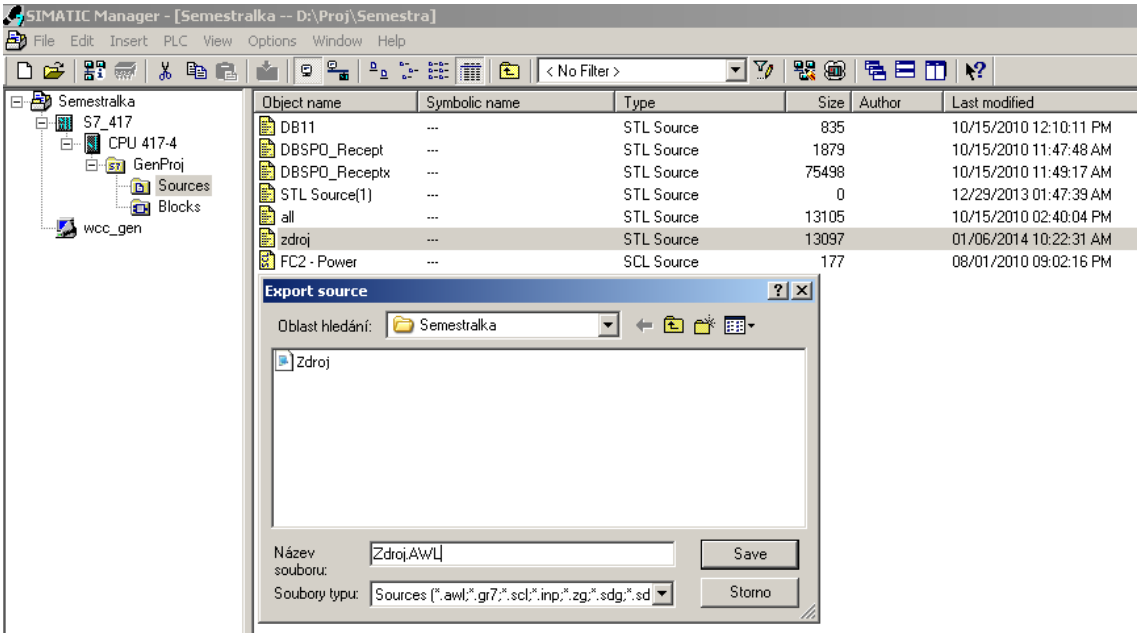


Obr. 8-1 Založení zdrojového souboru AWL v projektu STEP 7



Obr. 8-2 Výběr bloků pro generování zdrojového kódu AWL

Vygenerovaný zdrojový soubor byl začleněn do struktury daného projektu STEP 7. Lze k němu přistoupit přes záložku *Sources* ve stromu projektu. Zdrojový soubor je nutné ze SIMATIC Manageru exportovat do souboru formátu .awl (jméno souboru volíme libovolně). Stiskem pravého tlačítka myši na vybraném zdrojovém souboru zobrazíme nabídku, z níž vybereme volbu *Export Source*. Dojde tak k vyvolání dialogového okna zachyceného na Obr. 8-3. Zde vybereme umístění, kam bude soubor ve formátu *jméno.awl* uložen. Export bude dokončen stiskem tlačítka *Save*.



Obr. 8-3 Export zdrojového souboru z projektu STEP 7

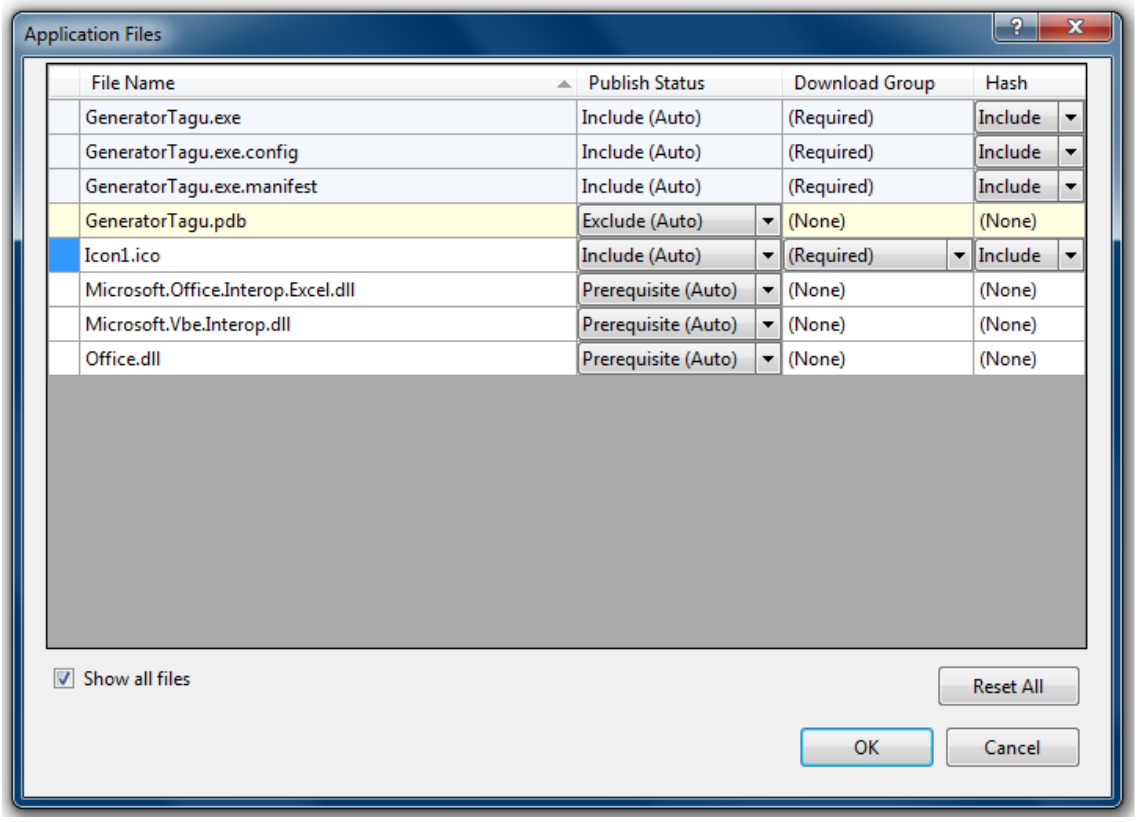
8.2 Instalace kompilátoru GeneratorTagu

GeneratorTagu je aplikací typu windows form application. Před prvním spuštěním musí uživatel aplikaci nainstalovat na PC (testováno na operačních systémech Windows XP a Windows 7, kde bude využívána.

Pro instalaci kompilátoru *GeneratorTagu* nejprve zkopírujeme z Přílohy č.1 archiv *GeneratorTagu.zip*. Tento archiv na cílovém PC rozbalíme a nově vytvořenou složku *GeneratorTagu* otevřeme. Uvnitř složky jsou umístěny kromě samotného instalátoru *setup.exe* i knihovny (seznam uveden na Obr. 8-4). Spustíme instalátor *setup.exe* spolu s aplikací *GeneratorTagu* budou instalovány následující součásti rozhraní .NET Framework (pokud již nejsou instalovány):

- .NET Framework 3.5 SP1
- Microsoft .NET Framework 4 (verze 32bit a 64bit)
- Windows Installer 3.1

Po instalací součástí .NET Framework bude zobrazeno dialogové okno, ve kterém stiskneme tlačítko *Instal*. Tímto byl *GeneratorTagu* nainstalován.



Obr. 8-4 Seznam souborů pro instalátor setup.exe

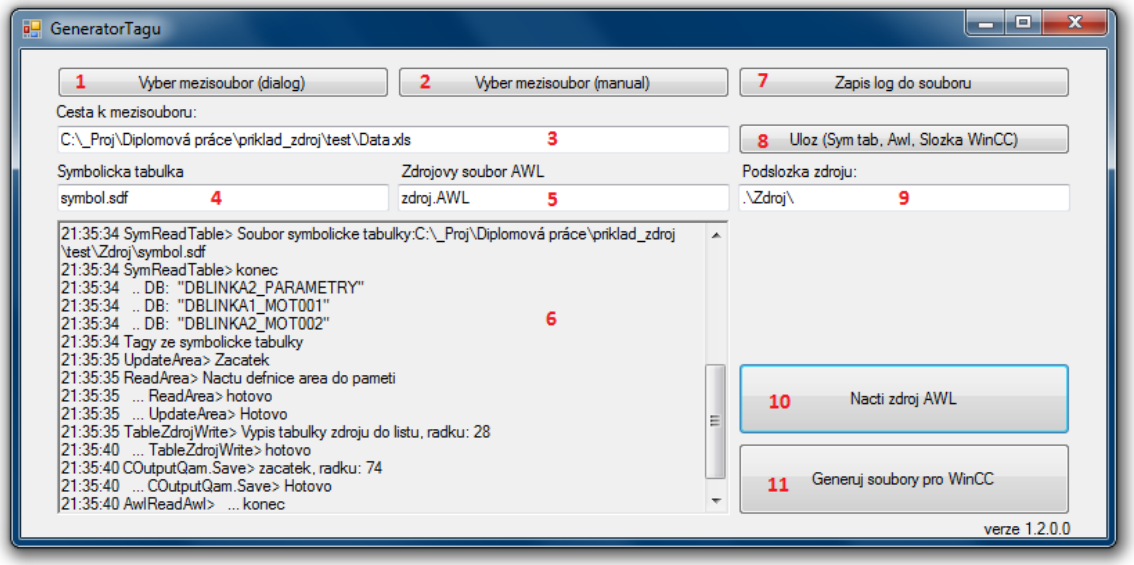
8.3 Spuštění kompilátoru GeneratorTagu

Spustíme program *GeneratorTagu.exe*. Na monitoru bude zobrazeno dialogové okno pro výběr mezisouboru, vybereme tedy mezisoubor předpřipravený dle kapitoly 6. Pokud dosud nebyl mezisoubor vytvořen, vybereme libovolný soubor a po vytvoření mezisouboru upravíme v kompilátoru GeneratorTagu cestu k novému mezisouboru). Po vykonání tohoto postupu je GeneratorTagu spuštěn.

8.4 Obsluha kompilátoru GeneratorTagu

Uživatelské rozhraní kompilátoru zdrojového kódu pro SIMATIC PLC ve verzi 1.2 je popsáno prostřednictvím Obr. 8-5, kde byly jednotlivé ovládací prvky pro účely popisu červeně očíslovány. Tlačítko označené číslem 1 spouští dialog pro nastavení cesty k připravenému mezisouboru. V případě manuálního zadání cesty k mezisouboru (textové pole značené číslem 3) je nutné následně stisknout tlačítko číslo 2. V obou případech jsou po zadání nové cesty do textových polí číslo 4, 5 a 9 zapsány hodnoty z mezisouboru (list *Dialog*). Text zapsaný v textovém poli číslo 9 určuje jméno složky, kde musí být uloženy veškeré zdrojové soubory vygenerované z projektu STEP 7. Jména zdrojových souborů jsou zapsána v textových polích s čísly 4 (jméno symbolické tabulky) a 5 (jméno zdrojového souboru AWL). Pokud došlo k úpravě polí 4, 5 nebo 9, provedeme zápis jejich nových hodnot do mezisouboru (list *Dialog*).

Textové pole označené číslem 6 slouží jako výpis o chodu kompilátoru. Výpis je ukládán do souboru se jménem dle vzoru *JménoMezisouboru_Tagy.txt*. K ukládání dochází automaticky po vykonání funkcí pro načtení dat do mezisouboru nebo po úspěšném vygenerování zdrojových souborů pro import tagů do WinCC V7.0. Popřípadě lze zapsat aktuální výpis do téhož souboru ručně stiskem tlačítka s číslem 7. Soubor s výpisem (v našem případě *Data_Tagy.xls*) je uložen do složky zdrojů.

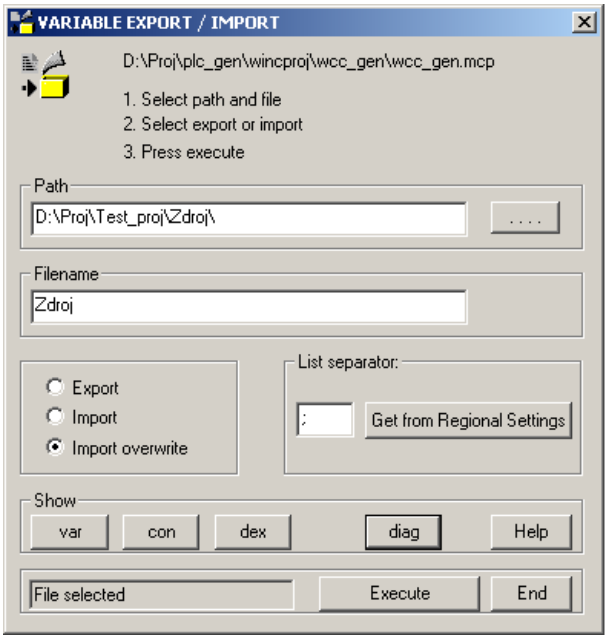


Obr. 8-5 Uživatelské rozhraní kompilátoru *GeneratorTagu*

Kompilaci zdrojového kódu provedeme stiskem tlačítka *Nacti zdroj AWL* (tlačítko označené číslem 10). Soubory potřebné pro přenos tagů do WinCC V7.0 vygenerujeme z mezisouboru stisknutím tlačítka *Generuj soubory pro WinCC* (tlačítko číslo 11).

8.5 Import tagů do WinCC V7.0

Nástroj pro export a import tagů zobrazený na Obr. 8-6 je součástí instalačního balíčku WinCC V7.0. Při standardní instalaci je nástroj *Var_Exim.exe* instalován do složky *C:\Program Files\Siemens\WinCC\SmartTools\VarExpImp*.



Obr. 8-6 Nástroj pro export/import tagů do WinCC V7.0

Nástroj pro import/export tagů do WinCC V7.0 vyžaduje/generuje tři soubory. Tyto soubory jsou ve formátu CSV. Jména souborů byla tvořena dle vzorce *JménoProjektu_Označení.csv*. Soubor s označením *vex* obsahuje seznam tagů. Jednotlivá spojení odkazující na tagy uložené ve *vex* souboru byly uloženy v souboru s označením *cex*. Pokud jsou součástí projektu struktury tagů, byly informace o nich obsaženy v souboru označeném *dex*.

Aby mohli být soubory vygenerované kompilátorem *GeneratorTagu* importovány do WinCC V7.0 je třeba nastavit v nástroji pro export a import tagů cestu k vygenerovaným souborům. Cestu lze zadat do textového pole *Path*, název projektu (projektu ve STEP 7) do pole *Filename*. Stejného výsledku lze dosáhnout stisknutím tlačítka pro vybrání cesty (označeno "….") a následným vybráním souboru *Zdroj_dex* nebo *Zdroj_vex* v zobrazeném dialogovém okně. Do pole *Filename* je automaticky přepsán název projektu. V sekci *List separator* zvolíme oddělovací znak v CSV souborech (*GeneratorTagu* používá jako oddělovací znak *středník*).

Zatrhneme volbu *Import overwrite* a tlačítkem *Execute* provedeme import souborů do WinCC V7.0.

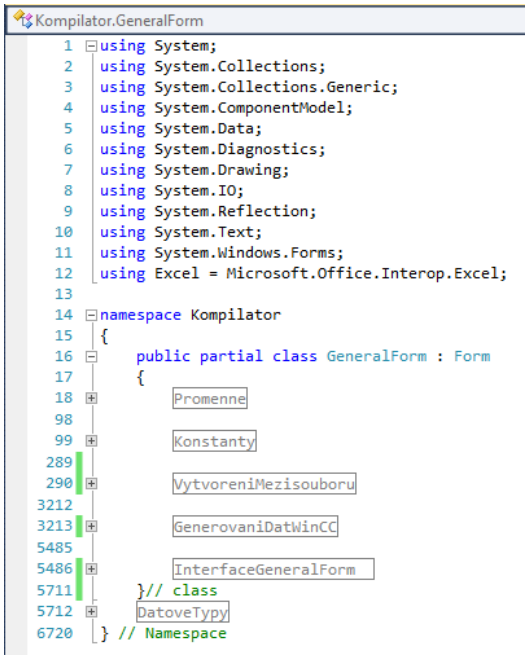
9 ZDROJOVÝ KÓD KOMPILÁTORU

Kapitola obsahuje popis tříd a jednotlivých metod kompilátoru *GeneratorTagu*. Datový typ proměnných uvedených v popisu tříd a metod byl doplněn do závorky za názvem proměnné. Pokud se jednalo o proměnnou standardního datového typu jazyka C#, bylo před název proměnné přidáno malé písmeno (s pro STRING, i pro INTEGER).

9.1 Třída GeneralForm

Tato třída dědí vlastnosti třídy *Form*, která byla určena pro práci s uživatelským prostředím vytvářené aplikace (typ windows form apllication) v prostředí Microsoft Visual Studio 2010. Metody v třídě *GeneralForm* byly rozděleny pro zvýšení přehlednosti kódu do pěti regionů. Rozdělení kódu na jednotlivé regiony je zobrazeno na Obr. 6-1.

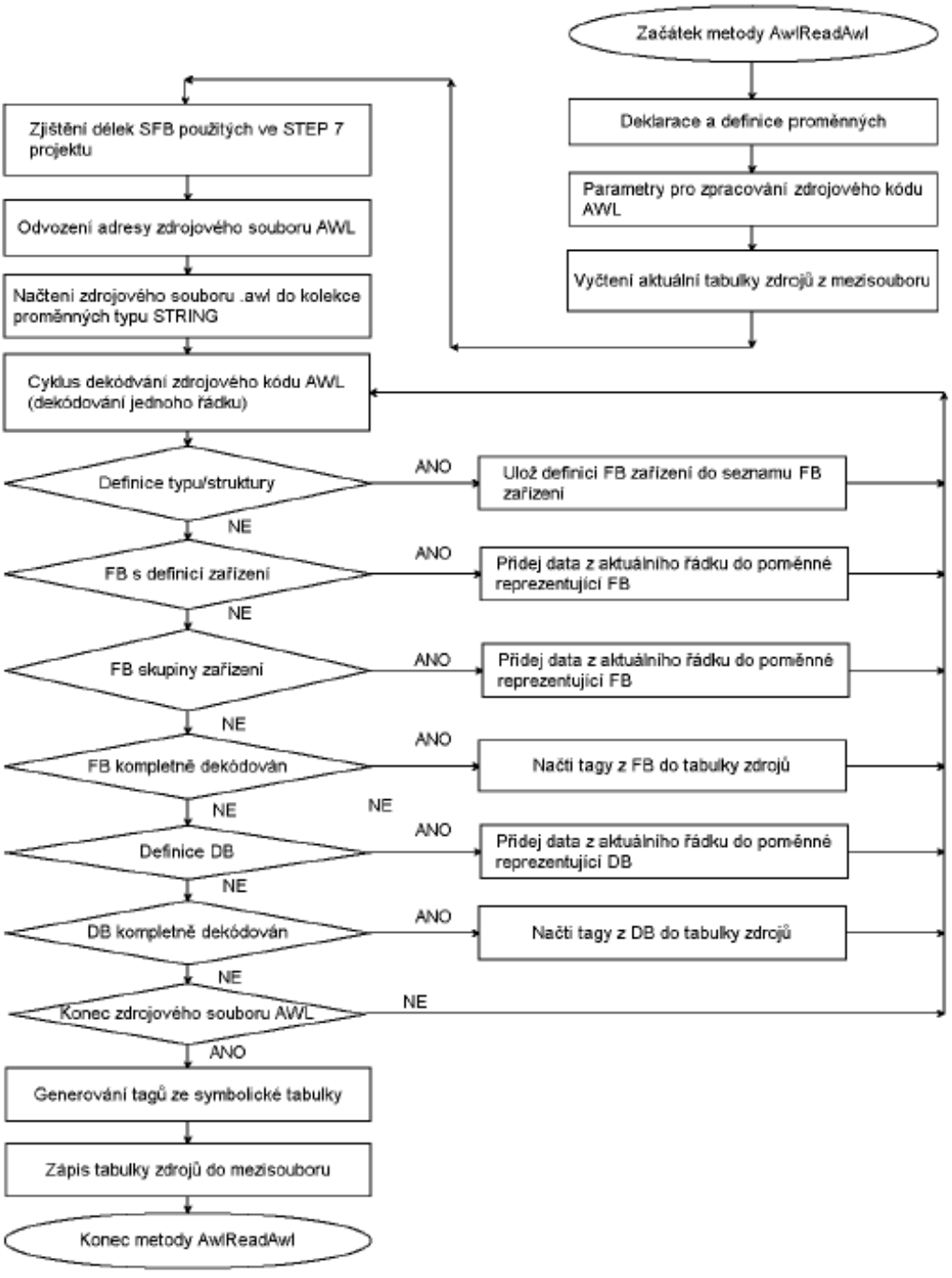
V prvním regionu označeném *Promenne* byly deklarovány a částečně i definovány globální proměnné třídy *GeneralForm*. V následujícím regionu Konstanty byly definovány konstanty (převážně je jednalo o klíčová slova pro WinCC V7.0 a indexy sloupců pro listy mezisouboru). Metody kompilátoru *GeneratorTagu*, které byly navrženy pro zpracování zdrojového kódu AWL a záznam relevantních informací do listů mezisouboru byly shrnuty v regionu *VytvoreniMezisouboru*. Region s metodami pro generování souborů určených k importu do WinCC V7.0 (tagy a alarmy) byl nazván *GenerovaniDatWinCC*. Metody přímo zasahující do navržené aplikace (práce s tlačítky, textboxy, atd.) byly zapsány v regionu *InterfaceGeneralForm*.



Obr. 9-1 Rozdělení kódu kompilátoru GeneratorTagu na regiony

9.1.1 AwlReadAwl (region VytvoreniMezisouboru)

AwlReadAwl byla navržena k vykonání kompletního dekódování zdrojového souboru AWL. Metoda postupně prochází zdrojový soubor a dle klíčových slov vybírá data relevantní pro zápis do mezisouboru. Zjednodušený popis metody byl popsán, prostřednictvím vývojového diagramu na Obr. 9-2. Kompletní vývojový diagram nebyl vytvořen z důvodu nízké přehlednosti (metoda *AwlReadAwl* obsahuje jedenáct set řádků kódu).



Obr. 9-2 Vývojový diagram metody AwlReaAwl

9.1.2 TagGenerateTag (region GenerovaniDatWinCC)

Pro zpracování dat zapsaných v mezisouboru byla vytvořena metoda TagGenerateTag, která data z mezisouboru převede do formátu požadovaného nástrojem pro import a export tagů do WinCC V7.0 Posloupnost operací prováděných metodou byla zaznamenána ve vývojovém diagramu na.

9.2 Třídy s definicí datového typu

V kompilátoru *GeneratorTagu* byly využívány k uchovávání dat kromě proměnných standardních datových typů (pro jazyk C#) také objekty (instance) vytvořené konstruktory navržených tříd. Každá třída obsahuje dva konstruktory. První konstruktor je volán bez parametrů a nastavuje proměnné na jejich výchozí hodnoty (např. hodnota nula pro typ INT). Druhý konstruktor disponuje jedním vstupním parametrem na každou jednu proměnnou třídy, které předává svou hodnotu. Těmto třídám by vyhrazen region *DatoveTypy*.

9.2.1 TypeArea

Instance této třídy byly využívány pro vyčtení seznamu skupin z listu Param. Definice skupiny obsahuje název *sWinCCArea*, symbolický název *sSymbolArea*, a identifikační číslo skupiny *iTxtId*, které je však nepovinné.

9.2.2 TypeAreaAlarmNumber

V instancích třídy *TypeAreaAlarmNumber* jsou uloženy jména *sAreaSymbol* a odpovícající čísla *iAlarmNumber* pro alarmy využívané v HMI aplikaci.

9.2.3 TypeAwlDb

Třída slouží jako seznam datových bloků vyčtených ze zdrojového souboru typu .AWL. Kromě celočíselné proměnné *iMax* pro sledování počtu vyčtených datových bloků, obsahuje třída kolekci instancí třídy *TypeAwlOneDb*. Kolekce nese označení *obj*.

9.2.4 TypeAwlFb

Funkční bloky vyčtené ze zdrojového souboru typu .AWL (dále označován jako zdrojový soubor) jsou reprezentovány kolekcí *obj*, která obsahuje instance třídy *TypeAwlOneFb*. Součástí třídy *TypeAwlFb* je proměnná *iMax* sloužící jako počítadlo počtu vyčtených funkčních bloků.

9.2.5 TypeAwlFbDev

Funkční bloky obsahující definice jednotlivých zřízení elektrovýzbroje technologického procesu jsou načteny ze zdrojového souboru do třídy *TypeAwlFbDev* obsahující kolekci *obj* (instance třídy *TypeAwlOneFbDev*), proměnnou *iMax* reprezentující maximální počet objektů v kolekci *obj* a proměnnou *iAct* zaznamenávající aktuální pozici při procházení kolekce *obj*.

9.2.6 TypeAwlNextObj

Třída obsahuje seznam objektů *obj* (typ *TypeAwlOneNextObj*) a proměnnou *iMax* (datový typ INT), která reprezentuje aktuální počet objektů v seznamu *obj*. Proměnná typu *TypeAwlNextObj* byla v kompilátoru *GeneratorTagu* využívána k uložení seznamu SFB použitých v STEP 7 projektu. Seznam použitých SFB byl vyčten z mezisouboru (list Param, sekce *AdditionalObjects*), kam byl zapsán uživatelem.

9.2.7 TypeAwlOneDb

Tato třída byla vytvořena pro reprezentaci datového bloku v kompilátoru *GeneratorTagu*. Číslo datového bloku bylo uloženo do proměnné *iDb*. Symbolické jméno DB bylo zpracováno v proměnných *sName, sTagName, sTagNameWithoutArea*. Do *sName* bylo zapsáno celé symbolické jméno DB a *sTagName* (jméno tagu včetně skupiny) obsahovala symbolické jméno DB po oříznutí prvních dvou znaků. Odstraněním názvu skupiny z *sTagName* byl získán text uložený v proměnné *sTagNameWithoutArea*. Dále byl DB popsán proměnnými *sComment* (kompletní komentář DB), *sDetail* (komentář DB bez klíčových slov pro kompilátor), *sAreaPlc* (jméno Area v programu PLC) a *sWccFolder* (jméno složky pro tagy ve WinCC V7.0).

9.2.8 TypeAwlOneFb

Funkční bloky pro skupiny zařízení byly kompilátorem *GeneratorTagu* převedeny na instance třídy *TypeAwlOneFb*. V rámci třídy byly definovány proměnné:

- *sName* (symbolické jméno FB),
- *sComment* (komentář FB),
- *db* (instance třídy TypeDb obsahující seznam instančních DB pro toto FB)
- *fb* (instance třídy TypeFbFb do které byl uložen seznam FB zařízení volaných skupinovým FB).

9.2.9 TypeAwlOneFbDev

Tato třída byla určena k popisu FB s definicí zařízení. Symbolické jméno FB bylo zapsáno do proměnné *sName* a komentář FB do *sComment*. Struktury dat byly uloženy do proměnné *obj* (instance třídy *TypeStruct*).

9.2.10 TypeAwlOneNextObj

Instance třídy *TypeAwlOneNextObj* obsahuje údaje o SFB, který byl použit v projektu STEP 7. Do proměnné *sName* bylo zapsáno jméno vybraného SFB a v proměnné *iLen* byl uložen údaj o délce SFB (v Bytech). Oba uvedené údaje byly čerpány z mezisouboru (list *Param*, sekce *AdditionalObjects*).

9.2.11 TypeAwlOneType

Třída představuje jeden objekt vyčtený ze zdrojového souboru. Objekt je definován jménem *sName* a vlastnostmi *objStruct* (instance třídy *TypeStruct*).

9.2.12 TypeAwlType

Seznam instancí třídy *TypeAwlOneType* označený jako *obj*, obsahuje objekty, které reprezentují funkční bloky pro nadefinování jednoho zařízení. Kromě kolekce *obj* obsahuje třída počítadla *iMax* a *iAct*, jejichž funkce byla popsána v kapitole 9.2.5.

9.2.13 TypeCTAlarmOne

Třída s vlastnostmi objektu pro korekci textů u příkazových tlačítek a alarmů. Obsahuje jméno skupiny objektů *sName*, seznam alarmu *alarms* a seznam tagů *tags*. Pro každý seznam byla zavedena jedna proměnná typu INTEGER (*iAlarmMax* a *iTagMax)*, jejíž hodnota odpovídá počtu prvků v daném seznamu.

9.2.14 TypeCTTagOne

Jedná se o třídu obsahující jméno (*sTagName*) a hodnotu (*sValue*) jednoho tagu nutného pro předání upraveného textu příkazového tlačítka objektu faceplate z knihovny *Compas Faceplate* [12]

9.2.15 TypeDb

Datové bloky vyčtené ze zdrojového souboru jsou ukládány do kolekce *obj* složené z instancí třídy *TypeDbOne.* Počet instancí v kolekci je reprezentován proměnnou *iMax.*

9.2.16 TypeDbOne

Každý datový blok definovaný touto třídou byl popsán třemi údaji. Jedná se o číslo *iDb*, jméno *sName* a jméno skupiny *sArea.*

9.2.17 TypeFbFb

Kolekce *obj* složená z instancí třídy *TypeFbFbOn* obsahuje seznam funkčních bloků vložených jako instance do druhých funkčních bloků. Počet instancí v kolekci reprezentuje proměnná *iMax.*

9.2.18 TypeFbFbOne

Funkční blok definovaný touto třídou je charakterizován jménem funkčního bloku v kódu AWL (*sNameAwl*), jménem používaným ve WinCC (*sNameWcc*) a jménem typu (*sType*) funkčního bloku.

9.2.19 TypeCheckStringAddress

Třída vytvořená k rozkladu adresy z formátu *string* na jednotlivé komponenty. Každá adresa je složena maximálně ze tří komponent. První komponentou je textová část *sPrefix*, následuje adresa byte *iByte* a od ní tečkou oddělená adresa bitu *iBit.*

9.2.20 TypeObject

Je třída obsahující seznam objektů *obj* s definicí všech parametrů (*kolekce instancí třídy TypeOneObject*).

9.2.21 TypeObjectAlarmRow

Definice jednoho alarmu je reprezentována instancí této třídy. Proměnné třídy jsou:

- *sTag* (obsahuje jméno tagu přiřazeného v listu alarmů)
- *sBit* (udává číslo bitu v tagu)
- *sEvent* (popisuje událost vzniku alarmu)
- *sClass* (odpovídá třídě alarmu)
- *sType(* přiřazuje alarmu jeho typ)
- *sPv* (kolekce proměnných typu STRING obsahuje názvy tagů navázaných na procesní hodnoty)
- *sAckTag* (obsahuje název tagu pro potvrzení alarmu)
- *sAckBit* (udává adresu potvrzovacího bitu v tomto tagu, status alarmu je určen bitem *sStatusBit* v tagu *sStatusTag).*

9.2.22 TypeOneObject

Třída byla zavedena jako souhrn parametrů jednoho typu objektu (zařízení). V třídě byly definovány parametry:

- *sName* (jméno objektu z mezisouboru list Obj>)
- s*XlsSheet* (jméno listu mezisouboru s definicí objektu)
- *sGroup* (skupina tagů)
- *idStruct* (číslo struktury objektu)
- *iRow* (číslo řádku, kde začíná v mezisouboru definice objektu)
- *iColumn* (číslo sloupce v mezisouboru, kde se nachází jméno objektu)
- *tag* (kolekce instancí třídy *TypeTagParam* pro definici parametrů jednotlivých tagů objektu)
- *alarm* (kolekce instancí třídy *TypeObjectAlarmRow*, určená k záznamu alarmů přidělených k objektu).

9.2.23 TypeOneSymbol

Pro práci s daty symbolické tabulky (soubor formátu .sdf exportovaný z projektu STEP 7) byla zavedena tato třída. Do proměnných třídy byly načteny data popisující jeden symbol v symbolické tabulce (jeden řádek souboru .sdf). Třída obsahuje proměnné:

- *sName* (jméno symbolu)
- s*Address* (adresa, ke které byl symbolický název přiřazen)
- *sDataTyp* (datový typ proměnné na adrese s*Address*)
- *sComment* (uživatelský komentář symbolu)
- *sDetail* (upřesnění typu uvedené v komentáři *sComment* mezi znaky WCC a mezera)
- *bWccComment* (příznak generování tagu). Pokud bylo na začátku komentáře *sComment* nalezeno klíčové slovo WCC, příznak *bWccComment* byl nastaven na hodnotu TRUE (generuj tag pro symbol).

9.2.24 TypeOutDexFile

Instance této třídy reprezentuje soubor *dex*, který obsahuje informace o strukturách tagů použitých v programu STEP 7. Jakmile byly do instance třídy zaneseny údaje o všech použitých strukturách tagů, byl kompilátorem *GeneratorTagu* vytvořen soubor *dex* (soubor typu .csv).

Třída obsahuje proměnné *iMaxRow* (maximální počet řádků s daty v souboru *dex*) a *row* (kolekce instancí třídy *TypeOutDexFileRow*, která odpovídá řádkům v souboru *dex*).

9.2.25 TypeOutDexFileRow

Třída definuje jeden řádek výstupního souboru *dex*(soubor typu .csv). Řádek byl vytvořen kolekcí *column* (kolekce proměnných datového typu STRING). Proměnné z kolekce *column* představují položky oddělené čárkou v souboru *dex* (při otevření souboru *dex* tabulkovým editorem Microsoft Excel odpovídají proměnné v kolekci buňkám na jednom řádku otevřeného listu).

9.2.26 TypeOutVexFile

Seznam tagů byl před zapsáním do souboru *vex* uložen v instanci třídy *TypeOutVexFile*. V třídě byly definovány proměnné *iMaxRow* (počet řádků zapsaných do souboru *vex*), *iMaxColumn* (počet sloupců, který byl zanesen do souboru *vex*) a *row* (kolekce instancí třídy *TypeOutVexFileRow*, která obsahuje vlastní seznam tagů).

9.2.27 TypeOutVexFileRow

Proměnná *column* (kolekce proměnných typu STRING) popisuje jeden řádek s definicí tagu pro soubor *vex*.

9.2.28 TypeRowZdroj

Třída obsahuje definici jednoho řádku v listu *Zdroj* v mezisouboru. Kromě vlastní definice řádku (kolekce proměnných typu STRING, představující jednotlivé sloupce v tabulce zdrojů) byly v třídě definovány příznaky *bEdited* (řádek byl pozměněn oproti originálu) a *bDuplicity* (jméno tagu v řádku bylo použito u dříve definovaného řádku).

9.2.29 TypeSimaticAddress

Třída byla navržena pro uložení číselné části adresy používané v programu STEP 7 (číselná část adresy odpovídá vzoru *byte.bit*), adresa použitého byte byla zanesena do proměnné *iByte*. K proměnné *iBit* byla přiřazena adresa daného bitu.

9.2.30 TypeSourceOneLine

K dekódování jednoho řádku zdrojového souboru typu .awl, byla vytvořena třída *TypeSourceOneLine*. Třída obsahuje příznaky:

- bInst (jedná se o instanci, jméno tagu bude odvozeno od jména DB namísto jména FB)

- bWccComment (na řádku byly nalezeny data určená pro WinCC V7.0).

Zbývající proměnné slouží k uložení textu z dekódovaného řádku. Dle stavu příznaku *bInst* mohou nastat dva případy. Pokud hodnota příznaku *bInst* odpovídá hodnotě TRUE, pak se jedná o tag uložený v instančním DB a řádek byl do proměnných *sLine*, *sComment* a *sInstExt* rozložen dle klíče *sLine;//WCC-INST-sInstExt sComment*. V opačném případě došlo k rozkladu řádku dle klíče *sLine;//WCCsDetail sComment*.

9.2.31 TypeStruct

Popis vlastností datové struktury obsažené ve FB byl popsán touto třídou. Proměnná *iDbLen* obsahuje délku struktury v bytech, *iRow* slouží jako počítadlo řádků struktury a *objLine* (kolekce instancí třídy *TypeStructOneLine*) obsahuje seznam řádků struktury.

9.2.32 TypeStructOneLine

Třída definuje jeden řádek datové struktury, která tvoří proměnnou funkčního bloku. *TypeStructOneLine* obsahuje proměnné:

- *sName* (jméno tagu zanořeného ve struktuře).
- *sType* (datový typ tagu zanořeného ve struktuře).
- *sComment* (komentář tagu).
- *address* (adresa tagu v DB přiřazeném k FB s parametrem typu STRUCT).
- *bWccComment* (příznak je aktivní, pokud v komentáři struktury nalezne klíčové slovo *WCC*).

9.2.33 TypeStructTag

Do instance této třídy byly ukládány jednotlivé řádky s definicí parametrů tagu z vybrané struktury tagů (využíváno během převodu objektů definovaných v mezisouboru (List *Obj*>) na strukturu tagů (soubor *dex* pro import do WinCC V7.0)). Jméno tagu bylo zapsáno do proměnné *sName* a parametry byla naplněna kolekce *sPar*.

9.2.34 TypeSymbolTable

Instance třídy reprezentuje v kompilátoru *GeneratorTagu* tabulku symbolických jmen exportovanou z projektu STEP 7. Proměnná *obj* (kolekce instancí třídy *TypeOneSymbol*) byla naplněna všemi symboly vyčtenými se souboru symbolické tabulky (soubor typu .sdf). Do proměnné *iMax* byl uložen celkový počet symbolů v tabulce. Příznak *bInit* je nastaven na hodnotu TRUE, pokud byla celá symbolická tabulka načtena do instance třídy *TypeSymbolTable*.

9.2.35 TypeTableZdroj

V kompilátoru *GeneratorTagu* byla naplněna instance třídy *TypeTableZdroj* daty a následně byl proveden přepis této instance do mezisouboru (list *Zdroj*). Třída obsahuje proměnné:

- *row* (kolekce instancí třídy *TypeRowZdroj*, která obsahuje všechny řádky tabulky zdrojů)
- *iMaxColumn* (počet sloupců tabulky zdrojů)
- *iMaxRow* (počet řádků v tabulce zdrojů)
- *iWccFolder* (číslo sloupce se jménem složky pro založení tagu ve WinCC V7.0)
- *iFpConfig* (číslo sloupce, který obsahuje konfigurační slovo pro Compas Faceplate [12])
- *sTagFpConfig* (jméno tagu, ve kterém byla zapsána konfigurace Compas Faceplate [12]).
- *bActual* (signalizuje stav kdy instance třídy *TypeTableZdroj* odpovídá listu Zdroj v mezisouboru)

9.2.36 TypeTableZdrojAddTag

V této třídě jsou shrnuty všechny parametry pro jeden objekt. Z instancí třídy *TypeTableZdrojAddTag* byla kompilátorem *GeneratorTagu* čerpána data pro zápis tagů do listu *Zdroj* v mezisouboru. Třída obsahuje proměnné:

- *sName* (jméno objektu)
- *sConnection* (jméno spojení definované pro WinCC V7.0)
- *sAreaPlc* (jméno skupiny v STEP 7 programu)
- *sTyp* (datový typ objektu)
- *sComment* (komentář k objektu)
- *sWccFolder* (jméno složky pro tagy ve WinCC V7.0)
- *iDb* (číslo DB)
- *address* (instance třídy *TypeSimaticAddress* ve které byla uložena adresa objektu)
- *sIoTyp* (identifikace oblasti adresace tagu).

Poslední jmenované proměnné byla hodnota přiřazena, dle použité oblasti adresace (“I“ pro vstup, “Q“ pro výstup, “M“ pro merkr a “ “ pro DB).

9.2.37 TypeTagParam

Třída obsahuje proměnné potřebné k popisu jednoho parametru (tagu) pro objekt, který byl definován v mezisouboru (list *Obj*>). Parametr byl identifikován názvem objektu

rozšířeným dle vzoru *název objektu.sVarname* (kde *sVarname* byla proměnná se jménem parametru). Dále byl parametr definován proměnnými:

- *sType* (udává datový typ parametru)
- *iLength* (obsahuje délku tagu)
- *sConn* (jméno použitého *Connection* ve WinCC V7.0)
- *sGroup* (jméno skupiny tagů)
- *sDefaultValue* (výchozí hodnota parametru)
- *sAddressPrefix* (počátek adresy tagu, včetně zkratky označující datový typ)
- *iAddressByte* (Byte na kterém začíná adresa parametru)
- *sAddresBit* (bitová část adresy parametru).

9.2.38 TypeTagType

Proměnné této třídy reprezentují tag pro WinCC V7.0 a veškeré jeho parametry. Jméno tagu bylo zapsáno v proměnné *sName* a parametry byly uloženy do *sPar* (kolekce proměnných typu STRING).

9.2.39 TypeUpdateDefaultValue

Třída uchovává výchozí hodnoty jednoho tagu (řádku) z mezisouboru (list *Zdroj*). Proměnná *sTagName* (kolekce proměnných typu STRING) obsahuje jména tagů, pro které platí výchozí hodnoty. Číslo uložené v proměnné *iColumn* odpovídá sloupci s přednastavenou hodnotou v listu *Zdroj*. Proměnná *sDefaultValue* obsahuje přednastavenou hodnotu.

10 ZÁVĚR

V prvních pěti kapitolách jsem se věnoval popisu prostředků a vědomostí potřebných k zpracování kompilátoru zdrojových dat pro PLC řady SIMATIC. Při volbě vhodného softwarového vybavení jsem hledal jednoduché a přístupné řešení. Zadaný vizualizační software WinCC a PLC řady SIMATIC byly navrženy společností SIEMENS. Proto jsem považoval za nejvýhodnější využít software STEP 7 od stejného výrobce.

Při návrhu mezisouboru byly zvažovány dvě varianty, a to použití databáze a tabulkového editoru. Z mého pohledu převýšili výhody tabulkového editoru nad výhodami databází. Tabulku vytvořenou tabulkovým editorem lze velice snadno zakomponovat přímo do projektu STEP 7 i WinCC V7.0 a přesouvat ji tak spolu s celým projektem. Navíc toto řešení neklade vyšší nároky na uživatele, pro obsluhu kompilátoru i mezisouboru bohatě dostačuje průměrná znalost práce na PC.

Zpracování dat ze zdrojového souboru AWL kompilátorem GeneratorTagu probíhalo při testech zcela korektně. Data vygenerovaná pro import tagů do WinCC V7.0 jsou rovněž v pořádku do WinCC importována. Navrženou koncepci kódu pro generování alarmů z mezisouboru se bohužel nezdařilo uvést do provozu. Během generování alarmů byla celá aplikace nestabilní, proto jsem metody ve finální verzi kompilátoru zablokoval. Nicméně jsou tyto metody přístupné k nahlédnutí v Příloze č. 1 (výpis zdrojového kódu kompilátoru GeneratorTagu).

Pro reálné nasazení kompilátoru v praxi je třeba zapracovat do kódu aplikace algoritmus schopný zpracovat data i pro aplikaci TIA Portal (aktuálně ve verzi 13). Společnost SIEMENS totiž postupně modernizuje PLC a OP, přičemž nová zařízení nejsou ve STEP 7 ani WinCC V7.0 podporována. Nevýhodou nezávislého kompilátoru GeneratorTagu je nutnost stále jej aktualizovat dle změn ve WinCC a STEP 7. Tuto nevýhodu však vyvažuje vysoká míra flexibility, které u komerčně dostupných kompilátorů nelze dosáhnout.

Literatura

[1] STEP 7 Professional- Industry Automation & Drive Technologies - Siemens:. SIEMENS, s.r.o. *Siemens Česká republika:* [online]. 2014 [cit. 2014-08-08]. Dostupné z: http://stest1.etnetera.cz/ad/current/index.php?vw=0&ctxnh=bfadced791&ctxp=home

[2] Procesní vizualizační systém SIMATIC WinCC V7.x- Industry Automation & Drive Technologies - Siemens:. SIEMENS, s.r.o. *Siemens Česká republika:* [online]. 2014 [cit. 2014-08-08]. Dostupné z: http://stest1.etnetera.cz/ad/current/index.php?vw=0&ctxnh=525faea5c2&ctxp=home

[3] Microsoft Visual Studio – Wikipedie:. *Www.wikipedia.org* [online]. 2014, 2014-07-07 [cit. 2014-08-08]. Dostupné z: http://cs.wikipedia.org/wiki/Microsoft_Visual_Studio#Visual_Studio_201

[4] S7-SCL- Industry Automation & Drive Technologies - Siemens:. SIEMENS, s.r.o. *Siemens Česká republika:* [online]. 2014 [cit. 2014-08-08]. Dostupné z: http://stest1.etnetera.cz/ad/current/index.php?vw=0&ctxnh=b82d918097&ctxp=home

[5] S7-GRAPH- Industry Automation & Drive Technologies - Siemens:. SIEMENS, s.r.o. *Siemens Česká republika:* [online]. 2014 [cit. 2014-08-08]. Dostupné z: http://stest1.etnetera.cz/ad/current/index.php?vw=0&ctxnh=8d98b7aeda&ctxp=home

[6] KOCHANÍČEK, Ludvík. Programovací jazyky pro PLC – Internetový portál COPTEL – Elektrotechnika:. *Internetový portál COPTEL – rozcestník:* [online]. 2010 [cit. 2014-08-08]. Dostupné z: http://coptel.coptkm.cz/index.php?doc=3905

[7] BĚHÁLEK, Marek. Kapitola 2. Základní charakteristika jazyka C#:. *Programovací jazyk C#:* [online]. 2007 [cit. 2014-08-08]. Dostupné z: http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/ch02.html

[8] Formát CSV (přípona souboru) - IT Slovník:. *IT Slovník - počítačový slovník:* [online]. 2003 [cit. 2014-01-06]. Dostupné z: http://it-slovnik.cz/pojem/csv

[9] [MS-XLS]: Introduction:. MICROSOFT. *MSDN – the Microsoft Developer Network:* [online]. 2014 [cit. 2014-01-06]. Dostupné z: http://msdn.microsoft.com/en-us/library/dd906173(v=office.12).aspx

[10] Register-Deckblatt, 20-er Einteilung Version 1.0 - siemens-simatic-s7-st-7pro1-course.pdf:. *Http://unitedelectricalengineers.wordpress.com/* [online]. 2012 [cit. 2014-08-08]. Dostupné z: https://unitedelectricalengineers.files.wordpress.com/2012/06/siemens-simatic-s7-st-7pro1-course.pdf

[11] S7 Library Functions | PLCdev:. *PLCdev | Tools for PLC programming:* [online]. 2005 [cit. 2014-08-08]. Dostupné z: http://www.plcdev.com/s7_library_functions

[12] Compas Faceplate- Industry Automation & Drive Technologies - Siemens. SIEMENS. *Siemens Česká republika* [online]. 2014 [cit. 2014-05-19]. Dostupné z: http://www1.siemens.cz/ad/current/index.php?ctxnh=bdc912c32d&ctxp=home

Seznam zkratek

AWL - Anweisungsliste (seznam instrukcí)
LAD - Ladder Diagram (žebříkový diagram)
FBD - Function Block Diagram (diagram funkčních bloků)
STL - Statement List (seznam instrukcí)
PLC - Programmable Logic Controller
HMI - Human Machine Interface
SFB - System Function Block (systémový funkční blok)
SFC - System Function (systémová funkce)
OB - Organizační blok
FB - Funkční blok
FC - Funkce
UDT - Uživatelsky definovaný datový typ

Seznam příloh

Příloha č. 1 - CD-R

Příloha 1

- Kopie diplomové práce ve formátu PDF
- Zdrojový soubor zdroj.AWL použitý pro testování kompilátoru *GeneratorTagu*
- Šablonu mezisouboru Data.xls
- Instalátor kompilátoru *GeneratorTagu* setup.exe
- Výpis zdrojového kódu kompilátoru GeneratorTagu