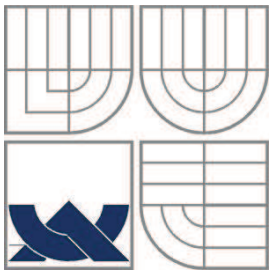
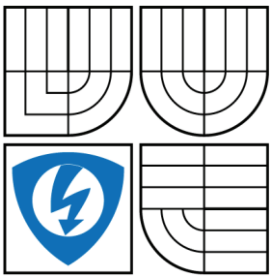


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

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

AUTOMATIZACE TVORBY SW ŘÍDICÍCH SYSTÉMŮ

AUTOMATION OF SW CONTROL SYSTEMS

DIPLOMOVÁPRÁCE
DIPLOMA'S THESIS

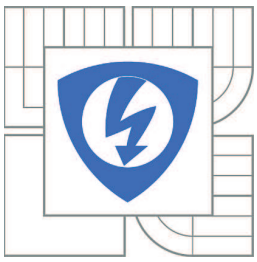
AUTOR PRÁCE
AUTHOR

Bc. Radek Sysel

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Radek Štohl, Ph.D.

BRNO 2015



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. Radek Sysel	***ID:***	133850
Ročník:	2	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Automatizace tvorby SW řídicích systémů

POKYNY PRO VYPRACOVÁNÍ:

1. Definujte základní objekty – objekt měření, spotřebiče aj.
2. Nalezněte vhodného zdroje dat pro objekty měření a spotřebičů.
3. Vytvořte vhodné nástroje pro naplnění databáze měření a objektů.
4. Proveďte popis, jak se pracuje s WinCC a jak se v něm přistupuje k objektům.
5. Vytvořte testovací aplikaci nad základní množinou objektů.

DOPORUČENÁ LITERATURA:

ISA-5.5-1985. Graphic Symbols for Process Displays. North Carolina: Instrument Society of America, 1985.

Dle vlastního literárního průzkumu a doporučení vedoucího práce.

Termín zadání:	9.2.2015	***Termín odevzdání:***	18.5.2015

Vedoucí práce: Ing. Radek Štohl, Ph.D.
Konzultanti diplomové práce: ing. Vladislav Konečný

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

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být 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.

Abstrakt

Diplomová práce pojednává o průběhu návrhu složitých objektů v automatizaci a jejich rychlého zařazení do celé procesní technologie. Ukazuje průběh tvorby od databázového modelu po generování do SCADA aplikace. Cílem práce je ukázat, co vše je potřeba vytvořit, aby vznikl funkční vizualizační model napojený na řídicí systém v PLC. Je zde pojednáváno o automatizovaném systému řízení změn, kdy již je procesní technologie vytvořena a my do ní musíme na základě revizí a připomínek zákazníka přidat objekty nové.

Klíčová slova

SCADA, spotřebič, měření, WinCC, PLC, MS SQL, databáze, Visual Basic, Simatic Step7, Tag, Alarm, Trend.

Abstract

The diploma thesis is about the design process of complex objects in automation and their rapid inclusion in the entire process technology. It shows the proces of creating a database model to generate into the SCADA application. The aim of the thesis is to show what all is needed to create, to produce a functional visualization model connected to the control system PLC. There is the analysis of the automated system change management, when it is processing technologies developed and we must do it based on customer feedback and revisions add new objects.

Keywords

SCADA, appliance, measurement, WinCC, PLC, MS SQL, database, Visual Basic, Step7, Tag, Alarm, Trend.

Bibliografická citace:

SYSEL, R. *Automatizace tvorby SW řídicích systémů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 60s. Vedoucí diplomové práce byl Ing. Radek Štohl, Ph.D.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Automatizace tvorby SW řídicích systémů 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: **18. května 2015** ……
podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Radku Štohlovi, Ph.D., Ing. Vladislavovi Konečnému a celému oddělení řídicích systémů firmy Siemens za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **18. května 2015**

……
podpis autora

Obsah

1 ÚVOD

Řídicí systémy dnes zasahují do všech odvětví průmyslu. Jsou čím dál více propracovanější, rychlejší a spolehlivější. Dnes se klade i velký důraz na vizualizaci technologických procesů. Vizualizace má působit přehledně, moderně a jednoduše. Musí splňovat požadavky na intuitivnost prostředí, dodržování obecných standardů ovládání SW počítačů, ergonomii, tak aby doba potřebná na zaškolování pracovníků obsluhy byla zkrácena na minimum. Celý proces návrhu řídicího systému se všemi funkčními komponentami je samozřejmě složitý.

V dnešním světě je celý tento proces návrhu čím dál více tlačen časem. Času na celý projekt je méně, ale práce je stejně nebo i více. Proto je vhodné určité části tvorby automatizovat. Najít řešení, které dovolí určité části návrhu výrazně zrychlit, aby bylo možno věnovat více času důležitějším věcem.

Navíc celá automatizace dílčích procesů musí být správně propojena s celým systémem řízení. A to nejen s programem v PLC (programovatelný automat), ale veškeré informace o našich objektech mohou být uloženy v databázi. V databázi být uloženy nemusí, nicméně pro automatizaci tvorby SW se bez toho neobejdeme. Je totiž nesmírně nutné, aby každá část, která bude vytvořena, správně zapadala do celého procesu. Při splnění těchto podmínek se pak projeví výhody daného řešení - snížení chybovosti v opakovaných činnostech až po úplné odstranění chyb vzniklých programátorem, dále možná opakovatelnost projektů, rychlá možnost úprav, zkrácení potřebného času k vytvoření a zprovoznění aplikace atd.

Firma, pro kterou je daná práce vytvořena, má vyřešený systém návrhu celého projektu v počátku zadání. Co už ale vyřešeno nemá, je řízení změn, které nastávají v procesu, než se celý projekt zprovozní finálně. Nastává zde totiž záležitost přidávání změn do hotového projetu. Proces řízení změn je velice důležitý a složitý. V této práci bude ukázáno, jak lze zautomatizovat tvorbu složitých grafických objektů. Od jejich tvorby v databázi, po vytvoření v grafickém nástroji, následně na možné pozdější vygenerování a napojení na řídicí systém, který již běží a dané objekty se musí zaintegrovat.

Pokud se zde bavíme o řídicím systému, máme na mysli řídicí systém pro parní turbíny. Je zde nutná znalost celého procesu řízení a to jak v programovatelném automatu, tak ve vizualizačním programu. Stejně tak je vhodná znalost principů činností snímačů použitých v hardwarovém návrhu pro celý proces. Zde už se počítá s danými znalostmi a celá práce je nadstavbou pro danou technologii.

2 ZÁKLADNÍ OBJEKTY V AUTOMATIZACI

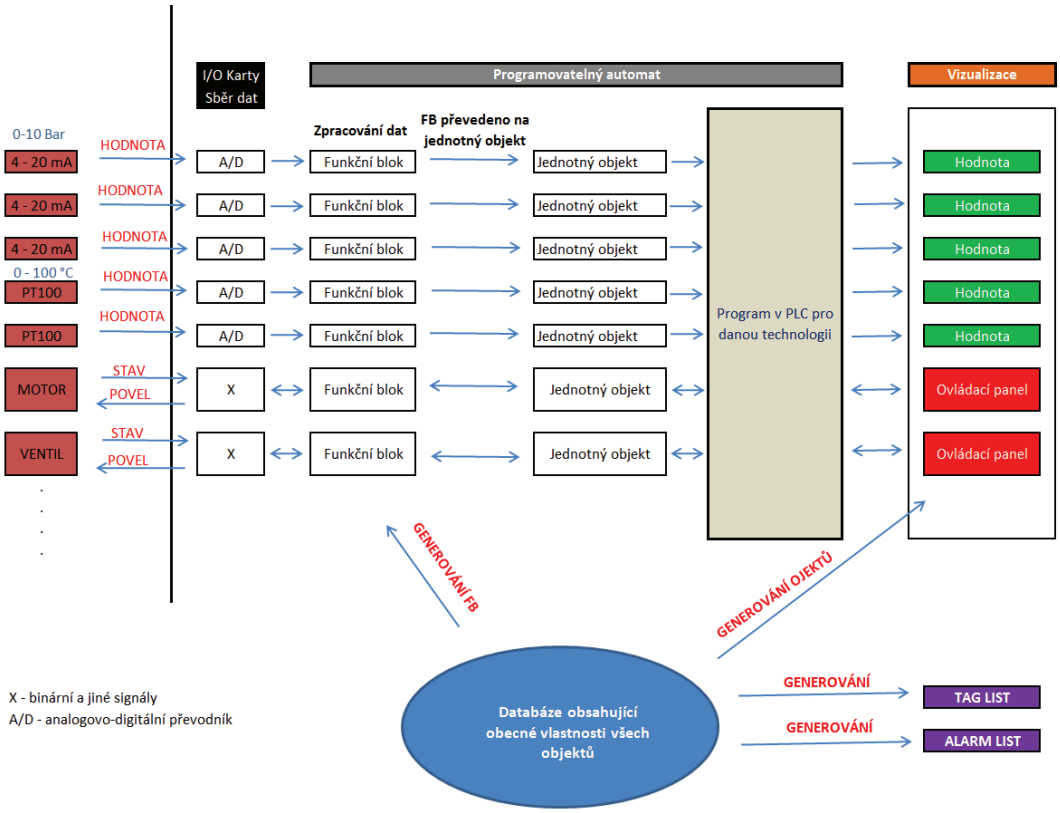
Automatizace má lidem usnadňovat život. K tomu, aby dosáhla tohoto poslání, potřebuje abstrahovat z procesu automatizace jednotlivé objekty, tak aby nabyly co nejobecnějšího charakteru. A mohly být pro účely automatizace typizovány v co nejmenším počtu druhů. Objektem máme na mysli vstupní a výstupní objekty automatizačního řídicího systému. Vstupní objekty – senzory (čidla), snímající fyzikální veličinu, s převodníkem na elektrický signál, 4 – 20 mA, odpor (PTx00), zvané též měření. Zajišťují pro automatizaci procesu vstupní informace o stavu řízené technologie. Nejsou jenom analogové, ale také binární. Výstupní objekty – akční členy – motor, ventil, regulovaný ventil – ovládaný buď binárními, nebo analogovým signálem, zvané taktéž spotřebič. Zde je potřeba se zamyslet, k čemu nám dané objekty jsou, jaké využití mají v našem procesu, jak pomůžou usnadnit život a celý proces vytváření objektů více zautomatizovat. V této diplomové práci probírané objekty, jejich popis a vlastnosti, budou brány s ohledem na potřeby průmyslu v oboru parních turbín, jejich následného řízení a v neposlední řadě vizualizace celého procesu. Dalo by se však říci, že nezáleží na technologii použití zde uvedených objektů. Můžeme používat objekty nejen v technologii, která se zabývá správným řízením parních turbín, ale i v automatizační technologii různých výrobních linek. Mnohé vlastnosti zde uvedené jsou natolik obecné, že není problém je využít jinde.

Kvůli složitosti zde probíraných objektů se zaměříme v následujících kapitolách na vytvoření a ovládání dvou hlavních objektů. Těmi budou objekt měření a objekt spotřebič (např. motor).

Všechny objekty zde uvedené byly vytvořeny a odzkoušeny na SCADA systému a nástrojem pro programování PLC. Jako SCADA nástroj byl použit program WinCC 7.0 SP3 update 5 a nástrojem pro programování PLC byl použit Simatic Step7 SP3 V5.5.

Dříve než budou vysvětleny vlastnosti jednotlivých objektů, je třeba uvést, co bude výsledkem celé práce. To nám nejlépe ukáže následující obrázek (Obrázek 1). Je na něm ukázáno, jak se dostane informace od hardwarového přístroje (ať už senzor nebo motor, ventil) do vizualizace. Od samotného zapojení do I/O karty, kdy prochází přes specializovaný funkční blok v PLC, který z něho převede informaci na obecnou informaci (stejný číselný formát) a tím dostaneme obecný jednoduchý objekt. Informace takto získaná je důležitá pro samostatný proces celé technologie, který je naprogramován v PLC a až následně je vše posíláno do vizualizace.

My se zde budeme zabývat tvorbou generování a ukládání dat do databáze. Pomocí vložených dat přes aplikaci budeme generovat pro každý objekt vizualizační podklad, alarmová hlášení, tagovou strukturu proměnných a v neposlední řadě vše udržovat aktuální v projektové databázi pro daný projekt.



Obrázek 1: Schéma celého procesu přenosu dat

2.1 Objekt senzor

Ať už měříme teplotu, tlak, výšku hladiny oleje v nádrži popřípadě jinou požadovanou veličinu, nastává tu pár problémů. Pokud opomeneme princip samotného čidla, převádějící příslušnou fyzikální veličinu (teplota, tlak, vibrace, zrychlení, frekvenci apod.) na elektrický signál (oddělení instrumentace nám dodalo správné čidlo pro dané měření), musíme brát v úvahu to, že dané čidlo je zapojené na určitý vstup našeho zařízení (rozvaděče), ze kterého programově odečítáme hodnotu. Samozřejmě daná hodnota se musí přepočítat. V praxi se nejčastěji na vstup posílá hodnota 4 – 20mA nebo se přímo připojí PT100. AD převod se provádí přímo na periferii automatu a potom se programem filtruje a přepočítává na skutečnou hodnotu. A tady se vynořuje náš „Objekt měření“. Pod tímto pojmem si můžeme představit virtuální obraz objektu, který spojuje signál ze senzoru až do samotné vizualizace, kde je vše zobrazeno. Je v něm zakomponována celá funkčnost celého procesu. Než se však začneme zaobírat vizuální stránkou objektu měření a normami, jak má takový objekt vypadat, podíváme se na obecné vlastnosti tohoto objektu. Z hlediska vizualizace ve WinCC má tří hlavní vlastnosti, které jsou důsledkem požadavku na to, co chceme, aby objekt dělal. Objekt je popsán ve WinCC i Windows pomocí dvou skupin parametrů – *Properties* a *Events* (Vlastnosti a události). Tyto parametry můžeme měnit či ovládat v procesu návrhu nebo v RunTime režimu. Podle toho, jak s těmi vlastnostmi pracujeme, je z hlediska návrhu dělíme do tří skupin: generované, skriptové (programové), manuální.

2.1.1 Vlastnosti generované z databáze

Tyto vlastnosti se generují na základě vyplněné databáze. Protože každý objekt měření má své individuální vlastnosti, je třeba, aby tyto vlastnosti byly zaznamenány v databázi. Výčet vlastností:

- Index – Udává unikátní index daného měření a umístění v DB21.
- Objektové jméno – Měření musí obsahovat objektové jméno. Využívá se to při spouštění skriptů.
- Kód 1 – Jedná se o tzv. KKS (z německého Kraftwerk-Kennzeichen-System). Každé měření musí mít své unikátní označení, toto označení pomáhá rychle identifikovat, o jaké měření se jedná. Daný KKS se vyskytuje i na projektových plánech k technologii, kde je měření použito.
- Kód 2 - Zákazník si velice často přeje, aby mohl k měřícím prvkům přistupovat a dohledávat pod svým označením. Objekt tedy musí obsahovat i tuto vlastnost: možnost zobrazit kód zákazníka.
- Formát – Udává, v jakém číselném formátu se bude zobrazovat výsledná hodnota a zároveň počet desetinných míst.
- Minimum – Jakou minimální hodnotu náš objekt ještě zobrazí.
- Maximum – Jakou maximální hodnotu náš objekt zobrazí.
- Jednotka – Jednotka k měřené veličině.
- Poznámka – Název vybraného objektu měření – většinou max 80 znaků.
- Hodnota – Na tomto místě je napojení na tag, ve kterém je uložena hodnota z PLC.
- Technologická skupina – Mazací olej, kontrolní olej, hlavní pára, ucpávková para atd.
- Sdružené měření – Některá měření jsou typu dva ze tří. Nemusí tomu tak být vždy.
- Typ měření – parametry potřebné ke generování awl souborů.
- A mnohé další parametry potřebné pro danou technologii.

Všechny tyto hodnoty je třeba mít vyplněné v databázi. V jaké databázi a jak se objekt měření na danou databázi napojí, následně vygeneruje, to vše bude vysvětleno v následujících kapitolách.

2.1.2 Vlastnosti spojené se skriptem

Tyto vlastnosti jsou napojené na naprogramovaný skript. Ať už napsaný v jazyce C nebo ve Visual Basicu. Jedná se o předpřipravené skripty, které berou jako vstupní parametr jméno objektu. Podle názvu objektu pak provedou danou operaci. Ačkoliv se v této práci budeme zaměřovat na objekty automatizace vytvořené ve vizualizačním programu WinCC, je třeba připomenout, že tyto vlastnosti by bylo možné nastavit i v jiných vizualizačních programech (např. Citec).

Události spojené se skriptem jsou většinou ukazatelem stavu daného objektu. Skript totiž dokáže změnit kteroukoliv vlastnost objektu na základě kritérií, které budou v našem skriptu obsaženy. Například při překročení minima nebo maxima objektu se může objekt jinak zbarvit, vstupní pole může vypsat *varovnou hlášku* apod. Další významnou událostí spojenou se skriptem může být vykreslení tzv. *faceplate* (Windows okno) s podrobnějšími informacemi o daném objektu měření.

Každý skript si ale musíme naprogramovat sami. Vizualizační programy sice nabízejí základní funkce, ale ty jsou ve většině případů nedostačující. Proto je vhodné ovládat programovací jazyk C, popřípadě Visual Basic. O tom, jak se napojuje skript na danou vlastnost a o samotném programování jednoduché vlastnosti ve WinCC, pojednává má minulá bakalářská práce [4].

Zde je výčet základních vlastností spojených se skriptem:

- Barva pozadí – Při speciální události dojde ke změně barvě pozadí. Objekt nás upozorní, že je něco jinak, než má být. Patří sem i zobrazení varovných symbolů hlášení – červené blikající zarámování, HH, H, L, LL (vše jsou ukazatele překročení limit měření), ComFail atd.
- Povolení aktivity objektu – Tato vlastnost upozorní na ztrátu komunikace mezi programovatelným automatem a vizualizací, kde je daný objekt zobrazen.
- Kliknutí na tlačítko myši – Po kliknutí je často požadována další informace v podobě podrobnějších informací o daném měření. Dojde také k výpisu vlastností objektu. To už záleží na nás, co budeme chtít zobrazit.

2.1.3 Vlastnosti manuální

Každý objekt je svým způsobem individuální. Můžeme chtít měřit teplotu na deseti místech, devět měření bude vizualizačně úplně stejné, ale u 10. měření budeme chtít jinou barvu, jinou průhlednost objektu, obecně jiné vlastnosti. Občas každý objekt automatizace potřebuje po automatickém vygenerování změnu určité vlastnosti. Opět zde budou uvedeny nejčastější vlastnosti:

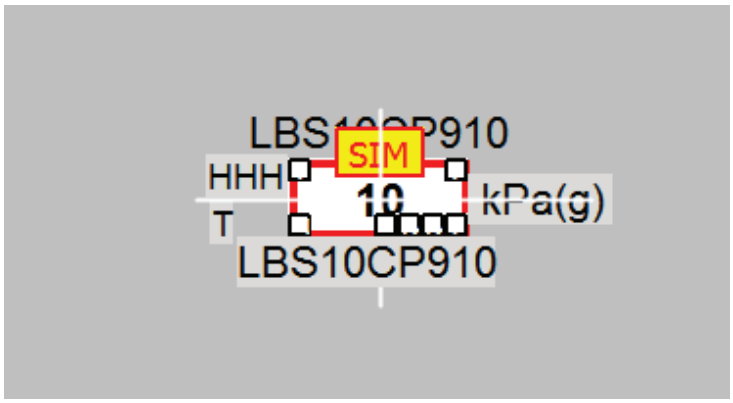
- Průhlednost – Objekt měření může změnit průhlednost.
- Flash animace – Jedná se o různě barevné blikání. Dělá se centrálně pro daný typ objektu. Daná vlastnost se nastavuje centrálně pro všechny objekty stejného druhu. Z důvodu jednotnosti zobrazované informace.
- Individuální popisek - Při překročení maximální či minimální hodnoty objektu měření budeme chtít zobrazit speciální hlášení.

2.1.4 Objekt měření vizualizace

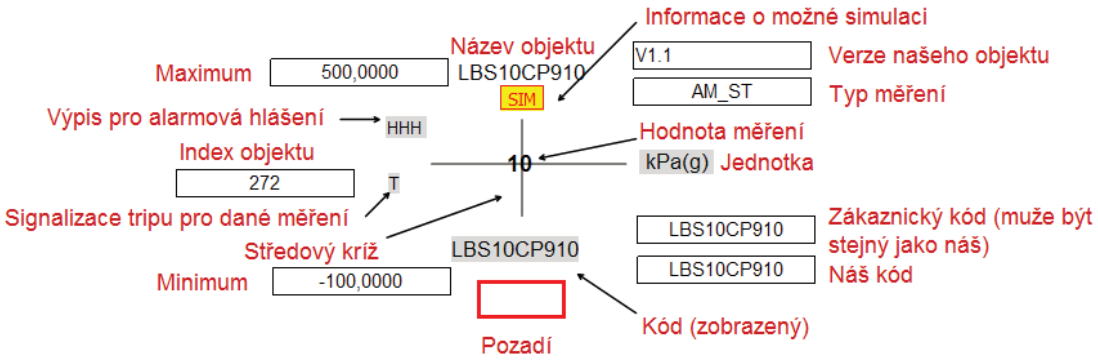
Před vytvořením samotného objektu měření je důležité si zjistit, zda takový objekt použitý v řídicím systému v průmyslu nepodléhá vizuálně určitým normám. Teprve pak

je vhodné začít takový objekt tvořit. Ovšem takovou normou se často můžeme pouze inspirovat. V našem případě byla počáteční inspirace normou ISA-5.5-1985 [1]. Protože poté, co objekt vytvoříme, ověříme funkčnost, přichází na řadu slovo zákazníka. Ten může mít své normy na vizualizaci průmyslových objektů u řídicího systému. Potom tedy rozhodují o výsledném vzhledu normy a zvyklosti zákazníka. Všechny tyto aspekty se nesmí při tvorbě vizualizace zanedbat. Mohlo by se taky stát, že se vygeneruje a upraví přes 100 objektů měření, spotřebičů, aby se následně zjistilo, že je vše jinak, protože normy zákazníka nesouhlasí s normami firmy.

Na obrázku (Obrázek 2) vidíme samotný objekt měření. Podíváme se na něho podrobněji. Na dalším obrázku (Obrázek 3) vidíme jeho podrobnější popis.



Obrázek 2: Objekt měření



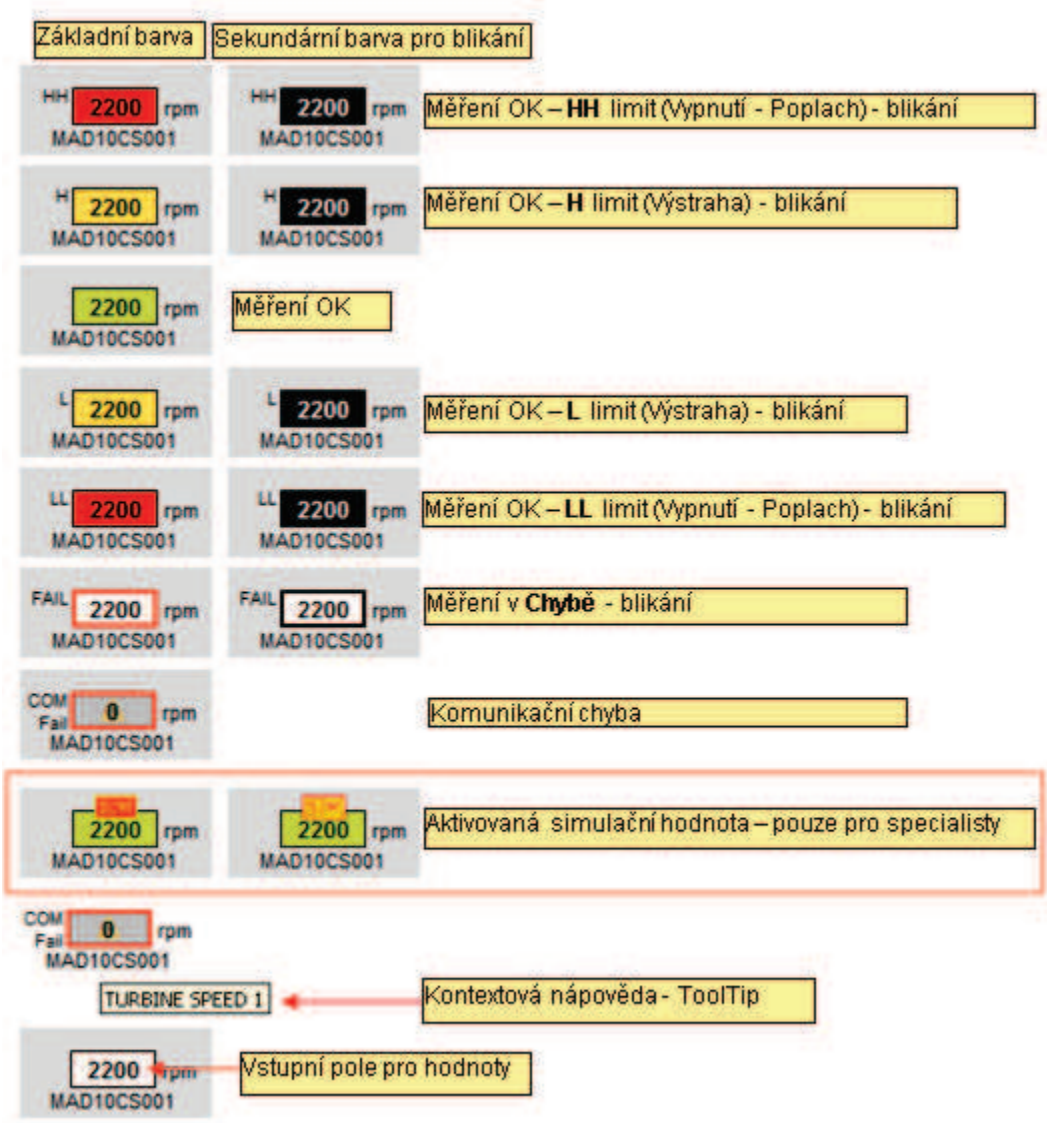
Obrázek 3: Podrobnější popis měření

Z podrobnějšího popisu si můžeme všimnout *středového kříže*. Ten slouží k vycentrování objektu. Dále si můžeme všimnout, že na původním obrázku (Obrázek 2) nejsou vidět velká bílá vstupní pole. Jsou tam, ale zmenšená. V režimu RT jsou neviditelná - skrytá. Slouží k uložení dat (vlastností) k objektu, které jsou generované z databáze. Jedná se o trvalé informace o objektu, tímto jsou uložena v grafickém návrhu. Například Min, Max hodnota.

2.1.5 Objekt měření v RunTime vizualizaci

Už při návrhu je třeba myslet na to, jak se bude daný objekt chovat při různých stavech v RunTime režimu. RunTime režimem je myšleno online sledování technologického procesu, je to vlastně běžící vizualizace. Tedy jak bude naše měření signalizovat události, o kterých by měl operátor vědět. Proto zde bude ukázáno, jak náš objekt mění barvu pozadí a celkově ukazuje signalizaci při různých stavech. Pro přehlednost je vše zobrazeno na ilustračním obrázku i s popisem (Obrázek 4).

Všechny barvy jsou měněné skriptem. Pro každý stav je definována barva. Při splnění podmínky je pak zavolán skript a barva změněna.



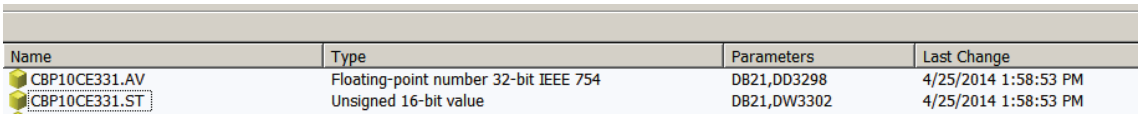
Obrázek 4: Celkový přehled všech možných barevných variací objektu měření

2.1.6 Struktura tagu pro objekt měření

Daná informace patří spíše do kapitoly týkající WinCC. Přesto považuji za vhodné, aby byla daná problematika rozepsána již zde. Jak již bylo uvedeno, tag je proměnná, ve které je uložená hodnota. Jinými slovy je tag vlastně označení přenášené jednotkové informace mezi PLC a vizualizací, může být v různém datovém formátu – Bit, Byte, Word, Float, Double Word aj. V našem případě potřebujeme u našeho objektu měření přenášet hodnotu měřené veličiny a dále bitově vyhodnocovat stav měření. Statusem měření myslíme nejenom chybu zapojení, ale i tzv. limity překročení určitých hodnot, na základě kterých začne vizualizace spouštět například alarmový systém WinCC. Limity se vyhodnocují ve funkčních blocích v PLC a do vizualizace se předávají ve stavovém slově, kde každý bit má svůj význam.

To, na jaké místo bude náš tag odkazovat, vyžaduje velice dobrou znalost řídícího programu v PLC pro danou technologii. Program v PLC je totiž velice silně provázaný s následnou vizualizací.

Naše struktura pro každé měření se tedy bude skládat ze dvou tagů. Tag pro aktuální hodnotu – typ tagu je *Floating-point number 32-bit IEEE 754* a pro vyhodnocování samotného stavu tag *Unsigned 16-bit value*.



Name	Type	Parameters	Last Change
CBP10CE331.AV	Floating-point number 32-bit IEEE 754	DB21,DD3298	4/25/2014 1:58:53 PM
CBP10CE331.ST	Unsigned 16-bit value	DB21,DW3302	4/25/2014 1:58:53 PM

Obrázek 5: Ukázka vytvořených tagů pro objekt měření

Stavové slovo se vyhodnocuje bitově. Je třeba zde uvést, v jaké posloupnosti bity jdou po sobě a co jednotlivé bity znamenají. V následující tabulce (Tabulka 1) je možnost vidět rozpis celé struktury objektu a to jak na úrovni ve WinCC, tak v PLC. Stojí za všimnutí, že při posílání stavového slova o 16-ti bitech z PLC do WinCC, dochází k přehození bytu. To znamená, že první byte v PLC je ve výsledku druhým bytem ve WinCC a naopak.

Všechny položky mají svůj význam, jak je ukázáno v tabulce. Položky LIM1 až LIM10 jsou obsazovány a používány velice individuálně. Záleží na použitém měření a na případu, ve kterém je použito.

Tabulka 1: Rozpis stavového slova po bitech

PLC	WinCC	Význam	Popis
Bit			
0.0	1.0	FAULT	Indikátor chyby měření.
0.1	1.1	Alarm_HH	Bit indikující překročení maximální možné hodnoty
0.2	1.2	Warning_H	Bit indikující překročení vyšší alarmové hodnoty
0.3	1.3	Warning_L	Bit indikující překročení nižší alarmové hodnoty.
0.4	1.4	Alarm_LL	Bit indikující překročení minimální možné hodnoty.
0.5	1.5	LIM1	Měření může obsahovat různé limity, které indikují překročení různých hodnot. Tyto limity jsou pak uplatňovány v algoritmech procesní technologie ke spouštění dalších akčních členů.
0.6	1.6	LIM2	
0.7	1.7	LIM3	
1.0	0.0	LIM4	
1.1	0.1	LIM5	
1.2	0.2	LIM6	
1.3	0.3	LIM7	
1.4	0.4	LIM8	
1.5	0.5	LIM9	
1.6	0.6	LIM10	
1.7	0.7	SIM_ON	Bit indikující, že měření je momentálně v simulačním módu.

2.2 Objekt spotřebič

Objekt spotřebič je vhodné nejdříve zobecnit z pohledu akčního členu. Můžeme si ho rozdělit na motor a ventil. Má své funkční bloky v PLC, které se starají o ovládání akčního členu na základě dat z měření a z příkazů z vizualizace. Tento objekt slouží k tomu, aby se určitý akční člen dal do pohybu. Skoro ve všech případech se objekt spotřebiče dá do pohybu na základě dat z objektu měření. Není vhodné, aby se spotřebič spouštěl bez předem vymezených spouštěcích podmínek. Dostáváme se znovu k objektu měření, na základě kterého dostáváme informaci, že se spotřebič může spustit, protože podmínky byly splněny.

Budeme se zabývat dvěma objekty spotřebičů – motorem a ventilem. Každý má svou specifickou značku, ale způsob ovládání těchto objektů zůstává stejný. Stejně tak vlastnosti jsou stejné. V následujících podkapitolách si danou problematiku u obou objektů blíže popíšeme.

2.2.1 Vlastnosti generované z databáze

U objektu měření jsme se bavili o důležitosti databáze a možnosti čerpat vlastnosti pro daný objekt z předem vyplněné databáze. Dává nám to obrovské možnosti. Zde u

objektu spotřebiče to platí stejně tak. Vlastnosti a parametry objektu spotřebiče je třeba mít uložený ve stejném stylu. Vlastnosti, které zde budou uvedeny, jsou takřka totožné s vlastnostmi objektu měření.

- Index – Udává unikátní index daného spotřebiče a umístění v DB512/515.
- Objektové jméno – Měření musí obsahovat objektové jméno. Využívá se to při spouštění skriptů.
- Kód 1 – Jedná se o tzv. KKS (z německého Kraftwerk-Kennzeichen-Systém, jako u objektu měření). Každý spotřebič musí mít své unikátní označení, toto označení pomáhá rychle identifikovat, o jaký objekt se jedná. Daný KKS se vyskytuje i na projektových plánech k technologii, kde je spotřebič použit.
- Kód 2 - Zákazník si velice často přeje, aby mohl ke spotřebičům přistupovat a dohledávat pod svým označením. Objekt tedy musí obsahovat i tuto vlastnost: možnost zobrazit kód zákazníka.
- Typ – Spotřebič může mít více podob. Práce se zabývá typem motor a ventil. Na základě typu se zobrazí určitá značka spotřebiče a zároveň se vhodně větví ovládací skript značky spotřebiče.
- Formát – Udává, v jakém číselném formátu se bude zobrazovat výsledná hodnota a zároveň počet desetinných míst. Používá se jen u regulovaných spotřebičů, které mají analogovou hodnotu.
- Poznámka – Krátký popis vybraného objektu.
- Médium – V závislosti na tom, co spotřebič ovládá (pára, olej atd.), takové číslo je zde obsaženo. Na základě čísla se pak udává barevné schéma do vizualizace pro samotný objekt.

2.2.2 Vlastnosti spojené se skriptem

Funkčnost celého procesu je velice obdobná jako u objektu měření. Naprogramovali jsme si skript, ať už v jazyce C nebo Visual Basicu. Tento skript napojíme na náš objekt. Jedná se nejčastěji o hlídání stavu objektu. Stejně tak se může jednat o upozornění na událost. Popřípadě zde u objektu spotřebiče se bude jednat o zpřístupnění ovládacího panelu a indikačních diod na *faceplatu*.

- Povolení aktivity objektu – Mezi PLC a vizualizací může dojít ke ztrátě komunikace. V takovém případě je důležité na to upozornit a znepřístupnit operátorovi povolení ovládat objekt.
- Zbarvení pozadí – Objekt spotřebiče je spouštěn velice často na základě podmínek, které jsou dány aktuálními hodnotami z objektu měření. Nicméně po spuštění sledování stavu spotřebiče nekončí. Může dojít k překročení hodnot dalších podmínek, chyb (stavová, časová, obecná) a tehdy musí náš spotřebič správně zareagovat, zbarvit své pozadí, aby operátor věděl, co přesně se děje.

- Kliknutí na tlačítko myši – v daném případě se nám zobrazí panel (*faceplate*) pro ovládání daného spotřebiče. Je velice důležité, aby objekt spotřebič měl možnost ovládání.

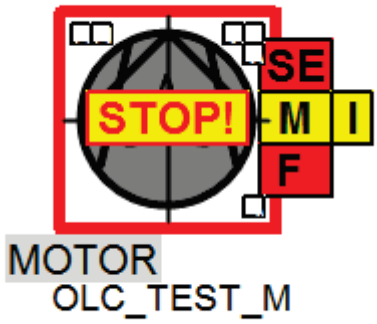
2.2.3 Vlastnosti manuální

Jak již bylo uvedeno u objektu měření, můžeme mít deset stejných spotřebičů v zadání projektu, databázi vyplněnou jejich parametry, ale následně při uvádění do provozu se zjistí jisté nesrovnalosti. Může se jednat o cokoliv. Například spotřebič bude potřebovat přidat jeden ovládací parametr, popřípadě při reakci na událost v daném technologickém procesu se má vizualizačně zbarvit jinak. Pak je důležité mít možnost danou vlastnost nastavit a parametrizovat ručně. Jedná se tedy o vlastnosti grafické převážně. Málokdy je třeba změnit odkaz na místo v paměti, protože došlo ke změně hardwarové dokumentace k projektu. Taková úprava vyžaduje i změnu v rozvaděči, na který je napojený náš spotřebič. Nemluvě o změně v samotném programu do PLC, ve kterém se musí také dané změny provést.

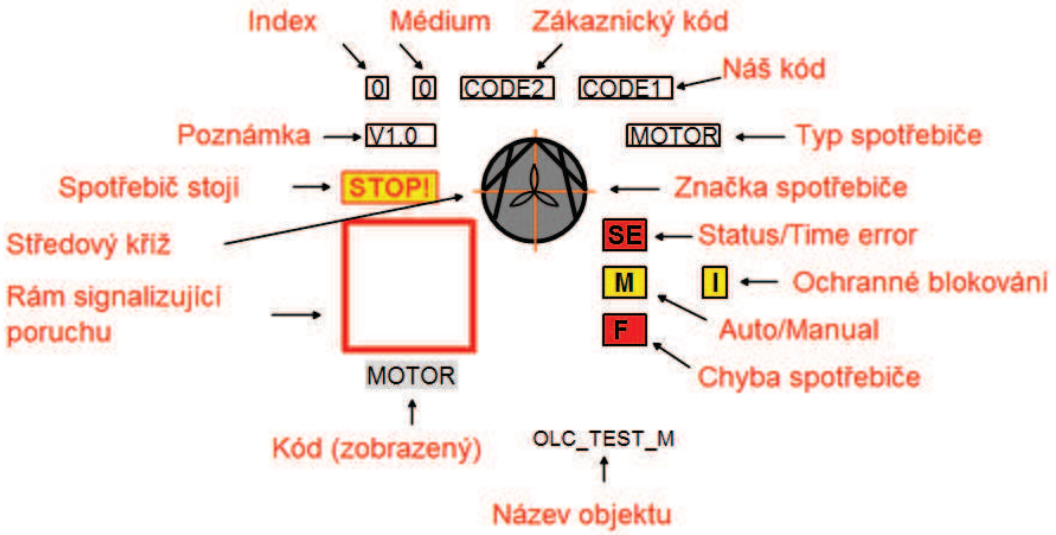
2.2.4 Objekt spotřebiče vizualizace

Ve vizualizaci spotřebiče také platí, že než ho začneme vytvářet, je třeba se podívat na normy, dle jakých by se mělo přibližně postupovat. A samozřejmě vycházet z podnikových norem každé firmy, kde takový objekt bude použit. Zde daný vizualizovaný spotřebič je podle této normy [2]. Na obrázku (Obrázek 6) vidíme objekt spotřebič – motor. Na dalším obrázku (Obrázek 7) jeho podrobnější popis.

Když se podíváme blíž na normy, je patrno, že daný objekt z nich přesto vychází. V čem dochází ke změnám, jsou hlavně barvy, ve kterých konečný objekt je vykreslen.



Obrázek 6: Objekt spotřebič - motor

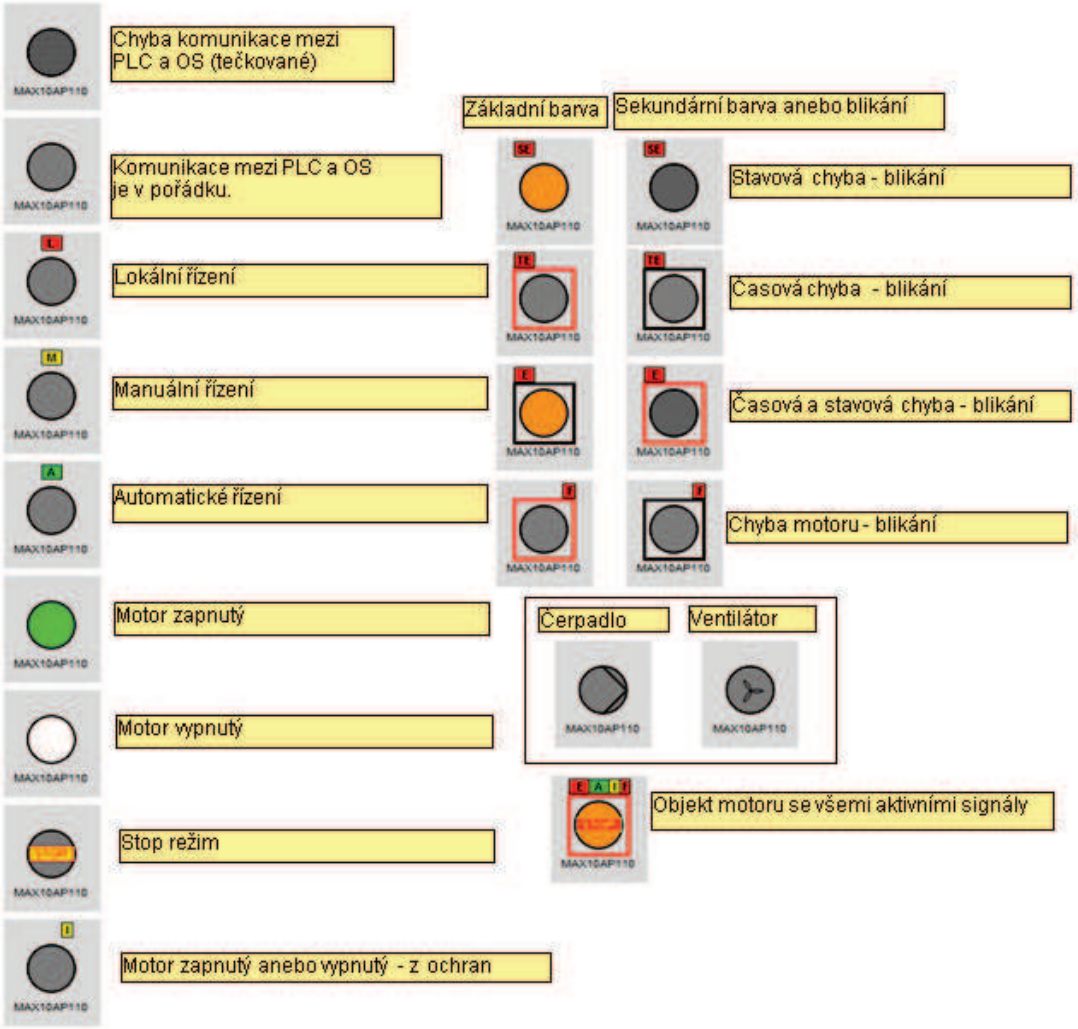


Obrázek 7: Objekt spotřebič - podrobnější popis

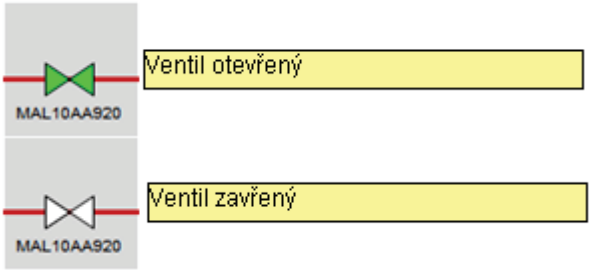
2.2.5 Objekt spotřebiče v RunTime vizualizaci

I zde budou zobrazeny všechny možné stavy pro náš spotřebič. Budeme se zabývat opět všemi možnými stavy pro motor. Ty jsou ukázány na obrázku níže (Obrázek 8). Stejně jak u objektu měření, i tady se o změnu barev stará skript napojený na barvu objektu. Jinak tu navíc ukážeme, jak vypadá ventil v RunTime režimu. Ale nebudou už ukázány všechny jeho možné stavy.

Objekt spotřebič je v počtu stavů na tom podobně jako objekt měření. Musí se pokrýt všechny chybové stavy, ukazatele automatického či manuálního režimu, objektový kód a jiné další důležité ukazatele. Už kvůli tomu, že spotřebič je akčním členem celé procesní technologie, není radno při návrhu opomenout kterýkoliv stav. Vše, co připojený motor pošle za informaci do PLC, je nutné pak i ve vizualizaci zachytit a zobrazit.



Obrázek 8: Celkový přehled všech možných barevných variací spotřebiče – motoru



Obrázek 9: Ventil v RunTime režimu

2.2.6 Struktura tagů pro objekt spotřebič

Objekt spotřebič již vyžaduje složitější strukturu tagů pro samotné ovládání. Je to dáno tím, že objekt spotřebič zpravidla obsahuje kromě podmínek, v jakém stavu se nachází,

ještě povely pro samotné ovládání. A každý povel je ještě ošetřen potvrzovacím bitem, který se posílá z ovládacího panelu k danému spotřebiči ve vizualizaci.

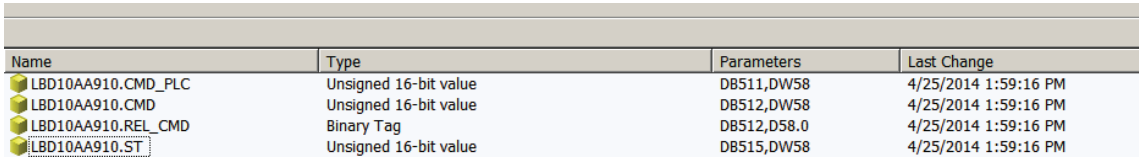
Také zde se vychází ze znalosti programu v PLC, kde už je datová struktura pro náš objekt připravena a my se napojíme pomocí tagů na ni. Struktura tagů pro objekt spotřebič vypadá následovně:

.CMD_PLC – Povelový word pro přenos dat do alarmového sytému.

.CMD – Command - Povelový word.

.REL_CMD - Potvrzovací binární povel. Používá se z hlediska bezpečnosti. I kdyby se operátor v příkazech *uklikl* myší, musí je ještě potvrdit.

.ST – Status – Stavová informace o spotřebiči.



Name	Type	Parameters	Last Change
LBD10AA910.CMD_PLC	Unsigned 16-bit value	DB511,DW58	4/25/2014 1:59:16 PM
LBD10AA910.CMD	Unsigned 16-bit value	DB512,DW58	4/25/2014 1:59:16 PM
LBD10AA910.REL_CMD	Binary Tag	DB512,D58.0	4/25/2014 1:59:16 PM
LBD10AA910.ST	Unsigned 16-bit value	DB515,DW58	4/25/2014 1:59:16 PM

Obrázek 10: Struktura tagů pro objekt spotřebič

Povelové slovo pro motor musí pokrýt všechny ovládací prvky, které objekt může nabízet. Ale povelových příkazů není tolik. Důležitý je příkaz RELEASE, bez něhož by se žádný příkaz neprovedl. Stejně tak příkaz ACKNOWLEDGE, který naopak dovoluje resetovat právě zvolený příkaz a zadat nový. Automatický a manuální mód je běžný objektů tohoto druhu. Jsou situace, kdy je lepší, že je automatický mód, protože se například opakuje stále stejný proces. A jindy zase může nastat výjimečná situace, kdy se naopak hodí ovládat motor manuálně.

Nyní se podíváme na jednotlivé bity povelového slova[4]:

Tabulka 2: Struktura povelového slova pro spotřebič

PLC	WinCC	Význam	Popis
Bit			
0.0	1.0	RELEASE	Potvrzovací příkaz ->Slouží k provedení příkazů od 0.2 (PLC, WinCC 1.2) bitu výš.
0.1	1.1	ACKNOWLEDGE	Zrušení příkazu ->Slouží ke zrušení příkazů od 0.2 bitu výš. Také slouží k resetu chyby na spotřebiči.
0.2	1.2	ON_CMD	Zapne motor.
0.3	1.3	OFF_CMD	Vypne motor.
0.4	1.4	AUTO_CMD	Automatický mód motoru.
0.5	1.5	MANUAL_CMD	Manuální mód motoru.
0.6	1.6	REZERVA1	Rezerva.
0.7	1.7	REZERVA2	
1.0	0.0	REZERVA3	
1.1	0.1	REZERVA4	
1.2	0.2	REZERVA5	
1.3	0.3	REZERVA6	
1.4	0.4	REZERVA7	
1.5	0.5	REZERVA8	
1.6	0.6	REZERVA9	
1.7	0.7	REZERVA10	

Dále budou probrány jednotlivé bity ve stavovém slově, které má svou strukturu v datovém bloku v PLC[4]. Ten má o poznání více obsazených bitů. Už kvůli tomu, že na objekt spotřebič jsou kladeny větší nároky a je potřeba více věcí kontrolovat. Zvláštní význam má vlastnost PROT_ON a PROT_OFF. PROT_ON slouží k zapnutí motoru, ať už jsou okolnosti a příkazy jakékoliv. Spotřebič se musí zapnout. A PROT_OFF má naopak opačný význam. Ten naopak dovoluje spotřebiči vypnout.

Tabulka 3: Struktura stavového slova pro spotřebič

PLC	WinCC	Význam	Popis
Bit			
0.0	1.0	FEEDBACK_ON	Zpětná indikace, že motor běží.
0.1	1.1	FEEDBACK_OFF	Zpětná indikace, že motor neběží.
0.2	1.2	AUTO_MODE	Automatický mód.
0.3	1.3	MANUAL_MODE	Manuální mód.
0.4	1.4	ON_CMD_MSG	Je aktivní povel Zapnout.
0.5	1.5	OFF_CMD_MSG	Je aktivní povel Vypnout
0.6	1.6	FAULT	Chyba.
0.7	1.7	LOCAL_MODE	Lokální mód.
1.0	0.0	STATUS_ERROR	
1.1	0.1	TIME_ERROR	Časová chyba.
1.2	0.2	PERMIT_ON	Podmínka, kdy se může zapnout motor.
1.3	0.3	PERMIT_OFF	Podmínka, aby šel motor vůbec vypnout. Musí být TRUE.
1.4	0.4	PROT_ON	Přednostní zapnutí
1.5	0.5	PROT_OFF	Přednostní vypnutí.
1.6	0.6	MSG_Reserve_16	Rezerva.
1.7	0.7	MSG_Reserve_17	Rezerva.

Jak již bylo uvedeno, motor (případně ventil), musí mít svou vlastní datovou strukturu v datovém bloku v PLC.

3 ZDROJ DAT

Ať chceme jako technici navrhovat (programovat) cokoliv, vždy je se třeba opírat o kvalitní podklady. Ty bývají nejčastěji uložené v databázi. Databáze je propracovaný systém pro ukládání dat, uložen na paměťovém médiu [3]. Součástí tohoto systému je i programovací jazyk na čtení/zápis do tohoto systému. Sice by tu byla možnost všechny naše data číst a zapisovat ze souboru, ale to je značně neefektivní a pomalé. Databáze je přímo stavěná a optimalizována na práci s velkým počtem dat. Stejně tak obsahuje mnoho ochranných prvků na přístup k ní.

3.1 Rozdělení databází

Dle způsobu ukládání dat dělíme databáze následovně:

- Hierarchické
- Síťová
- Relační
- Objektové
- Objektově relační

Nejčastěji se používají webové relační databáze. Více samotné rozdělení databází rozebírat nebudeme z důvodů, že databázové systémy jsou věda sama o sobě a není to hlavním předmětem této diplomové práce. Důležité je zde vědět umět pracovat s danou databází.

3.2 Využití

Dnes se můžeme setkat s databázemi na každém kroku. V mobilech je databáze SQLite. Tato databáze, ačkoliv k tomu nebyla původně určena, je vestavěná v systému Android. Na webovém poli se setkáme hlavně s databázemi MySQL, PostgreSQL, Oracle, MSSQL. MySQL i PostgreSQL jsou *open source*, tedy zdarma a běží na Linuxu. MSSQL je databáze od firmy Microsoft, běží tedy na počítačích s operačním systémem Windows. Databáze Oracle je dle použité technologie nejkvalitnější, zato je placená. Patří však k tomu nejlepšímu, co můžete získat. Hodí se pro opravdu náročné aplikace a složité webové portály. Nicméně technik (programátor) si bohatě vystačí s open source produkty MySQL a PostgreSQL. MS SQL neuvádím, protože naprostá většina *hostingových* serverů běží na Linuxu.

Databází je samozřejmě mnohem víc. Jsou zde uvedeny jen ty nejpopulárnější.

3.3 Volba databáze

Opravdu je velice důležité si říct, k čemu daná databáze bude použita. Podle toho si zvolíme. V mém případě byl mimo jiné požadavek na databázi, která bude fungovat již s modelem, který je zavedený ve firmě. Zde mají počítače Windows a využívají pro ukládání dat databázi MS SQL. MS SQL je databáze relační.

Ačkoliv je MS SQL databáze od firmy Microsoft, je velice podobná databázi MySQL stylem přístupu do databáze. Pro zápis do databáze se používá stejný jazyk SQL. Zvládnutí umění programovat v SQL je velice žádoucí pro práci s databází MS SQL.

3.4 Návrh tabulek

Volba databáze byla poměrně jednoduchou záležitostí. U tabulky vycházel proces návrhu také z potřeb firmy a jejich zavedených procesů. Jak bylo v minulé kapitole vypsáno, vlastnosti každého objektu je třeba z databáze nejen načítat, ale i ukládat. Každá vlastnost tedy musí mít v tabulce svůj sloupec a určení sloupce, o jaký datový typ se jedná. Například zda se jedná o text nebo číslo, popřípadě ještě jiný typ (datum, enum atd.). Od toho se pak odvíjí celý návrh tabulky. Zde se vyplatí vzít nejdříve tužku, papír a napsat si, co je očekáváno od naší tabulky, jaké vlastnosti mají opravdu být ukládány. A je běžné, že v první fázi tvorby se několikrát návrhový vzor naší budoucí databáze změní.

Důležité je však navrhovat model tabulek tak, aby se později, kdy už celý systém běží, nemusely dělat zásahy do tabulek. Navrhnout správný databázový model je základem každé složitější aplikace. Bez správného databázového modelu pak nemůžeme požadovat, aby naše aplikace běžela dlouhodobě spolehlivě.

Na základě požadavků firmy a obecně platných pravidel jsem došel k následující tabulce tabObjects (Tabulka 4). Tabulka je společná pro všechny typy objektů vizualizace. Musí vycházet z toho, jaké veškeré informace musíme u objektů shromáždit, abychom je mohli automaticky generovat do různých datových a grafických výstupů.

Kromě této tabulky se používají i další pomocné tabulky:

- tabPredefinedSignals – v této tabulce jsou informace o tom, jakou strukturu mají objekty, sufixy signálů pro alarmový systém a typ každého signálu.
- tabSettings – zde jsou uloženy informace o projektu, prefix ke všem signálům, rozlišení obrazovek na operátorských stanicích atd.
- tabObjectType – každá struktura objektu má své místo v alarmovém systému. Pomocí této tabulky víme, odkud můžeme začít generovat objekty do alarmového systému.

Tabulka 4: Navržená tabulka pro načítání a ukládání dat

Název pole	Datový typ	Popis
DB_object	Číslo	Datový blok v PLC, na který bude odkazovat náš objekt.
Address_object	Číslo	Adresa objektu v samotném DB.
Tag_group	Text	Upřesnění, do které skupiny se uloží TAG objektu.
Obj_name	Text	Primární klíč. Jméno objektu.
Code1	Text	Firemní kód objektu.
Code2	Text	Zákaznický kód objektu.
Tooltip_EN	Text	Popis objektu v angličtině.
Tooltip_2	Text	Popis objektu v jazyku zákazníka.
ObjectType	Číslo	Id typu objektu. Definuje, jakou tagovou strukturu bude mít náš objekt.
Alarm_Group	Text	Rozlišení alarmového hlášení.
Format	Text	Definuje číselný formát, který bude zobrazený ve vizualizaci.
R_Min	Číslo	Pro objekt měření – minimální hodnota.
R_max	Číslo	Pro objekt měření – maximální hodnota.
Eu	Text	Jednotka měřené veličiny.
Medium	Číslo	Pro objekt spotřebič – podle zvoleného media se pak vykreslí barevné schéma.
Name	Text	Pro objekt spotřebič – jméno ventilu.
PictureWindow	Text	Objekt pro zobrazení okna.
FaceplateName	Text	Faceplate pro zvolený objekt.
HH_Trip	Číslo	Objekt měření – Tripová hodnota
HH_Lim	Číslo	Objekt měření – vyšší alarmová limita.
H_Lim	Číslo	Objekt měření – alarmová limita.
L_Lim	Číslo	Objekt měření – alarmová limita.
LL_Lim	Číslo	Objekt měření - nižší alarmová limita.
LL_Trip	Číslo	Objekt měření – Tripová hodnota
HW_addess	Text	PIW adresa pro načítání dat.
SignalType	Text	Objekt měření - rozlišení, zda se jedná o PT100 nebo smyčku 4-20 mA.

Všechny pomocné tabulky vycházely z již zavedeného modelu dokumentace k procesní technologii. Celkově je celý tabulkový systém v databázi poměrně rozsáhlý. Ale každá tabulka a sloupec v ní má své opodstatnění. A mohlo by se očekávat, proč není databáze ještě rozsáhlejší, ale opravdu není důležité, abychom dosáhli složitého uspořádání, které by za rok od vytvoření bylo nepoužitelné, protože by mu již nikdo nerozuměl.

Často jsou vytvářeny skvělé aplikace, moduly do již vytvořených programů, které však fungují jen pro aktuální problém a zapomíná se, že trh a automatizace obecně je velice dynamický, proto programy musí být navrhovány už v základu univerzálně,

přehledně a musí být jednoduché na úpravu pro pozdější požadavky. Už kvůli tomu, že ti, kdo navrhovali jakoukoliv aplikaci do průmyslu, nemusí pracovat pro firmu na stálo, a proto musí své počínání ve fázi vývoje řádně dokumentovat. Aby jejich nástupci nezačínali od znova, naopak aby mohli pokračovat v již existující myšlence a tu někam posunout.

Je vhodné, aby tedy se už od počátku myslelo na to, že se vývoj aplikace časem předal novým a mladším nástupcům. Je opravdu nežádoucí, aby se v důsledku jakýkoliv změn na pracovišti, začínalo od začátku.

4 PLNĚNÍ A GENEROVÁNÍ

Firma, pro kterou je daná práce vytvářena má samozřejmě databázový nástroj, pomocí kterého lze plnit námi zvolenou databázi MS SQL. Tento nástroj slouží nejenom k plnění dat, ale i ke generování všech grafických objektů do vizualizace, stejně tak funkčních bloků do Step7, dále alarmového systému, tagových struktur a výsledné dokumentace pro zákazníka. Tento nástroj není předmětem zkoumání této práce.

Jelikož ale SCADA systém k řídicímu systému v PLC běží na operátorské stanici, která má v sobě nainstalované pouze WinCC, vznikl požadavek, zda by nebylo možné využít možnosti tohoto SCADA nástroje, k plnění dat a i ke generování složitých objektů. Často se totiž na stavbě zjistí, že je třeba doplnit ještě nějaké měření, popřípadě spotřebič. A to celé se musí zakomponovat do vizualizace a alarmového systému, databázového systému k celému projektu, nemluvě o programu do PLC.

Musel tedy vzniknout takový nástroj, který bude na ovládání velice jednoduchým, ale velice mocný pro správu již vytvořené databáze, s možností generovat složité objekty do WinCC. Nástroj musel být spustitelný na kterékoliv operátorské stanici, kde je nainstalované WinCC 7.0.1.

Předně je třeba říci, že bylo využito ve WinCC možnosti programovat ve Visual Basicu. Stejně tak Visual Basic dovoluje vytvářet formuláře, pracovat s nimi, napojovat na ně naše naprogramované funkce.

Byl vytvořený pdl soubor (ve WinCC je to typ grafického souboru, se kterým daný program pracuje), který po otevření ve WinCC dovoloval spouštět naše funkce.

Ještě poznamenám, že bylo čerpáno ze systémového manuálu k WinCC [5], kde jsem našel důležité rady, jak implementovat určité funkce řešící náš problém.

4.1 Co vše se bude generovat

Než bude ukázán formulář na plnění dat, rozepíšeme si zde, co vše se bude generovat. Aplikace musí umět zvládat nejen generování grafického objektu do vizualizace se všemi zadanými vlastnostmi, ale dále bude generovat všechny důležité části potřebné ke správnému chodu objektu v celé procesní technologii. Jedná se o alarm list, tag list a samozřejmě uložení do všech tabulek do hlavní databáze k projektu.

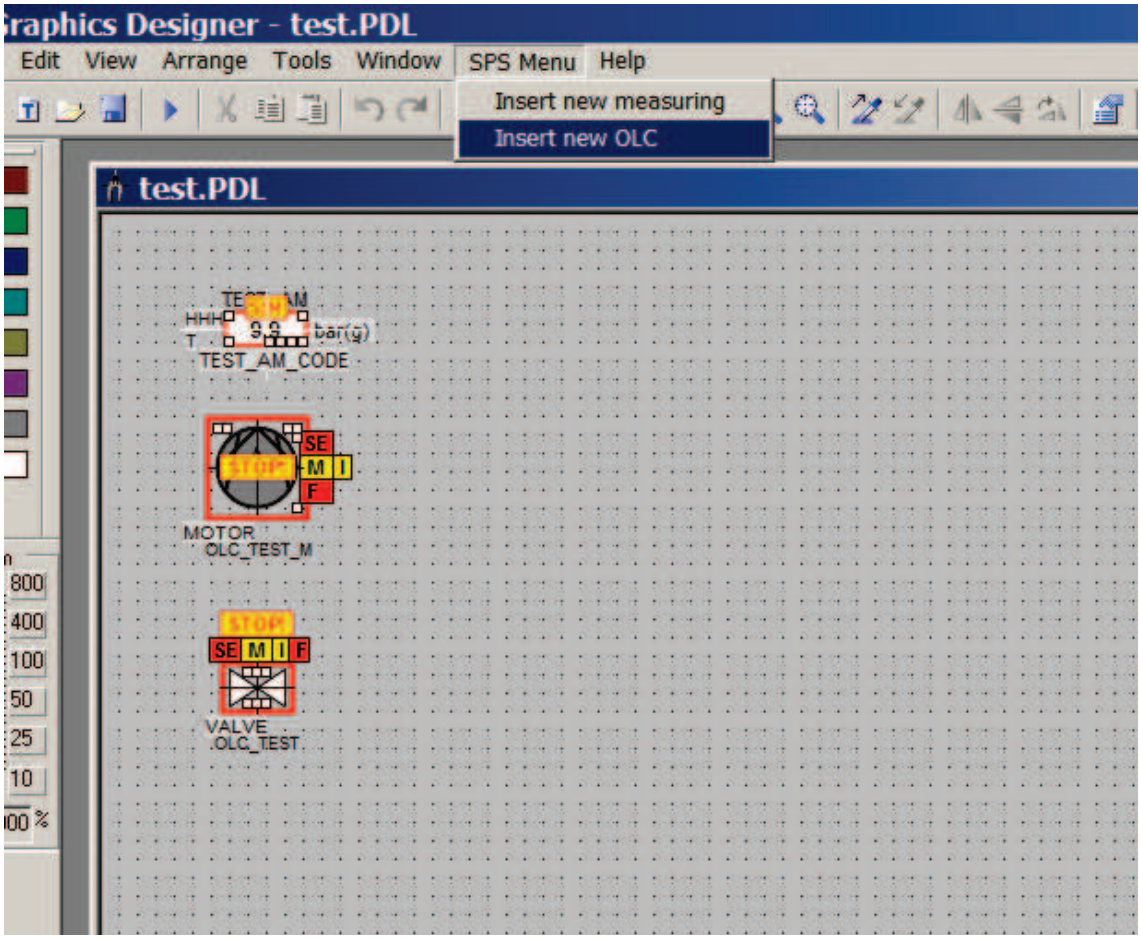
Ideové schéma bylo ukázáno a popsáno v kapitole 2 na obrázku (Obrázek 1).

4.2 Menu ve WinCC

Na začátek je výhodné si vytvořit menu v grafickém nástroji Graphics Designer ve WinCC. Menu je důležité z toho hlediska, že přes něj budeme přistupovat k našim funkcím a formulářům. Naštěstí Graphics Designer dovoluje přistupovat ke svému menu pomocí příkazů ve Visual Basicu. Proto byla naprogramována funkce

InsertMenuItems(), která mi vždy při spuštění našeho pdl souboru integruje menu (Obrázek 11) do grafického nástroje. Následně po zavření souboru se menu odstraní. K tomu slouží funkce *DeleteMenu()*.

Nutno podotknout, že dané funkce (jejich kód) nejsou uloženy mezi klasickým kódem ve Visual Basicu (v Modules), ale přímo ve vlastnostech daného grafického objektu. Zde máme na mysli soubor *test.pdl*. Po jeho otevření v Graphics Designeru se nám zobrazí i naše menu pro ovládání a generování.



Obrázek 11: Ukázka vytvořeného menu

4.3 Formulář pro objekt měření

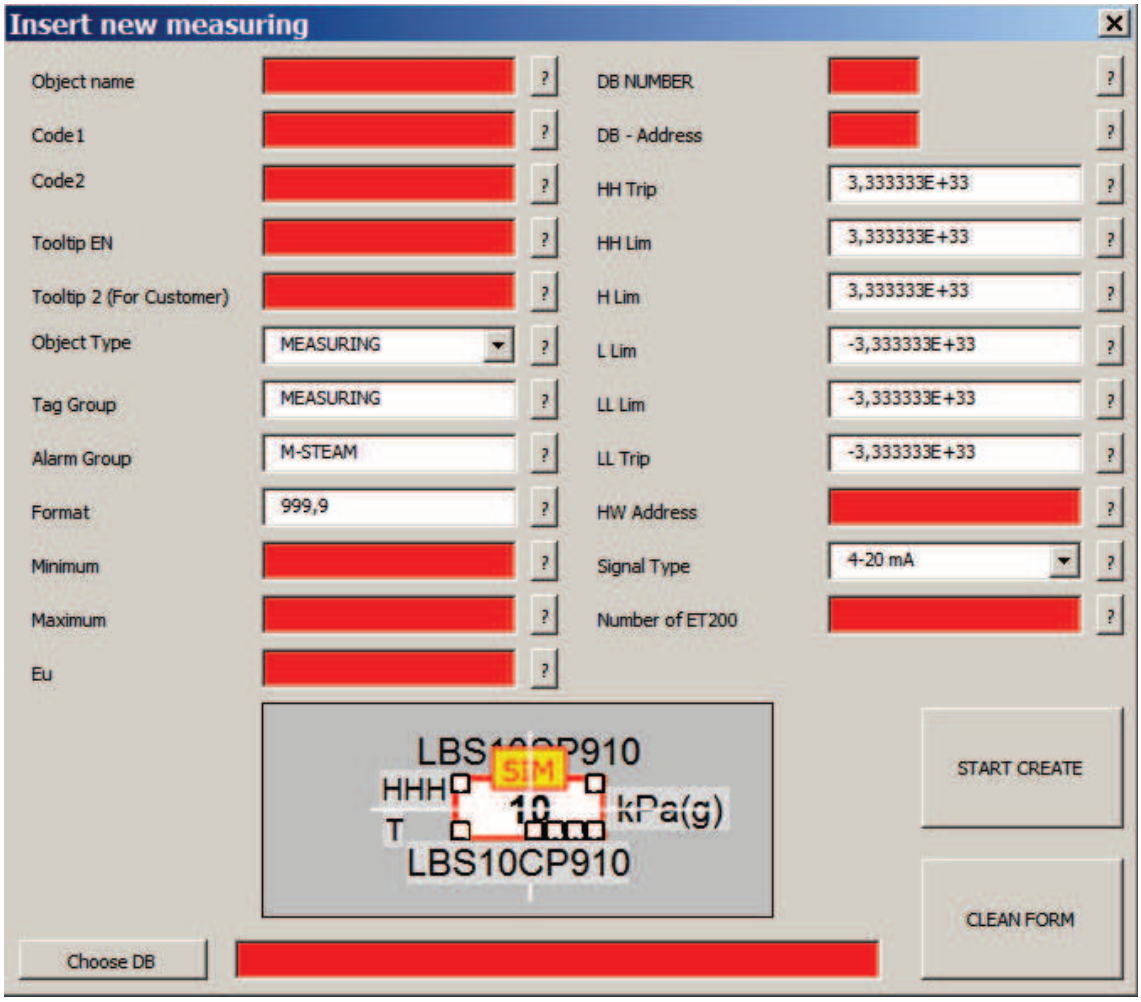
Po vytvoření menu pokračovala práce na samotném formuláři (Obrázek 12). Na formulář se položily všechny prvky, které budou ukládány do našeho databázového modelu, který jsme si navrhli v předchozí kapitole.

Každé pole, které se nevyplní, svítí červeně. A formulář nedovolí uložit data. Stejně tak je třeba vždy vybrat databázi, na kterou se chcete napojit a do ní uložit data.

Na formuláři je jedno hlavní tlačítko START CREATE, jež zastupuje tři hlavní funkce, které budou rozepsány v pozdější fázi. Jedná se o následující funkce:

- Export alarmů – Exportuje k danému objektu měření alarmové zprávy do WinCC.

- Vytvoření tagu – Vytvoří k objektu speciální soubor, který se naimportuje do WinCC a vytvoří speciální tagovou strukturu.
- Vytvoření grafického objektu – Vytvoří grafický objekt na základě vyplněných dat.

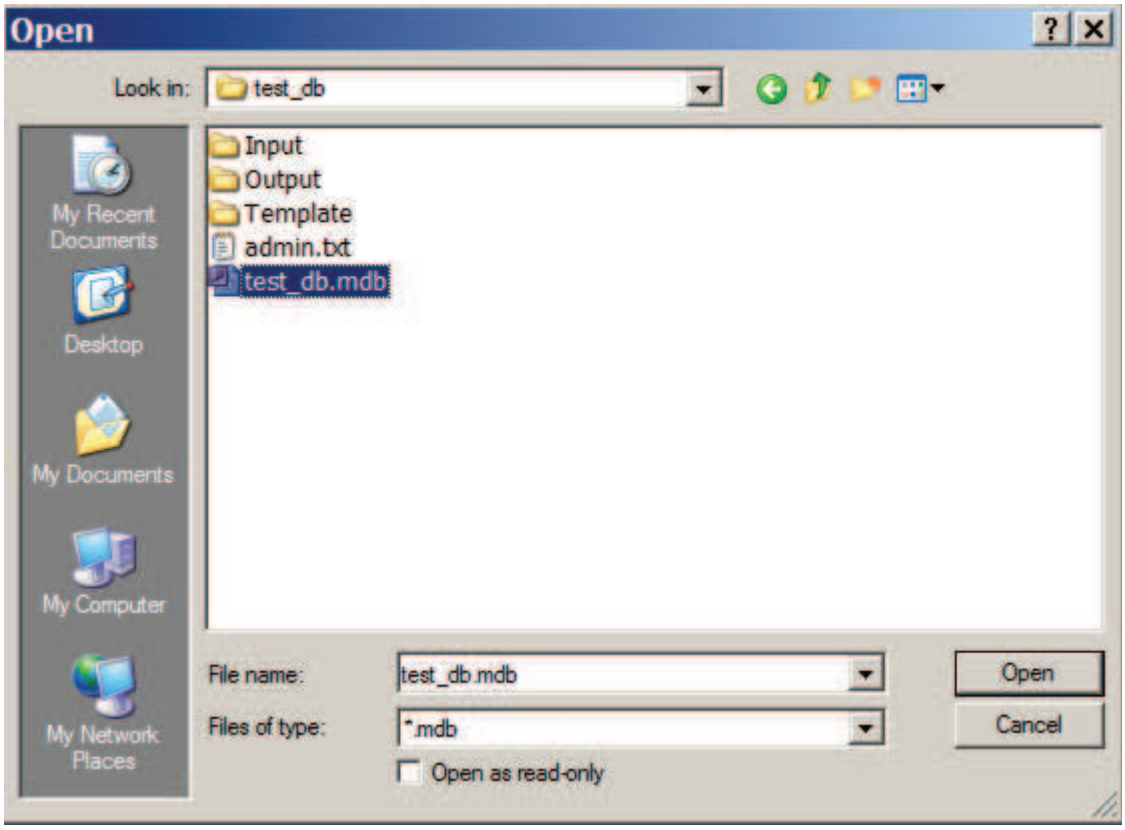


Obrázek 12: Formulář pro objekt měření

4.3.1 Volba pracovní databáze

Než začneme generovat všechny části objektu, je třeba se připojit na centrální databázi projektu. Na tlačítko *Choose DB* se objeví dialogové okno, ve kterém si můžeme vybrat databázi, na kterou si přejeme napojit náš grafický objekt. Tam pak budou naše data ukládána. Databáze musí být typu MS SQL (formát, který poskytuje MS ACCESS). Zároveň je připojení velice důležité z toho důvodu, že v centrální databázi se nachází doplňující informace ke generování (prefix, sufix, počet aktuálních prvků v alarmovém systému atd.).

Po výběru správné databáze se může v pořádku pokračovat. Bez výběru DB ale nebude dovoleno cokoliv generovat, protože daný objekt by vůbec nemohl být zaintegrován do aktuální procesní technologie, ve které má být použit.

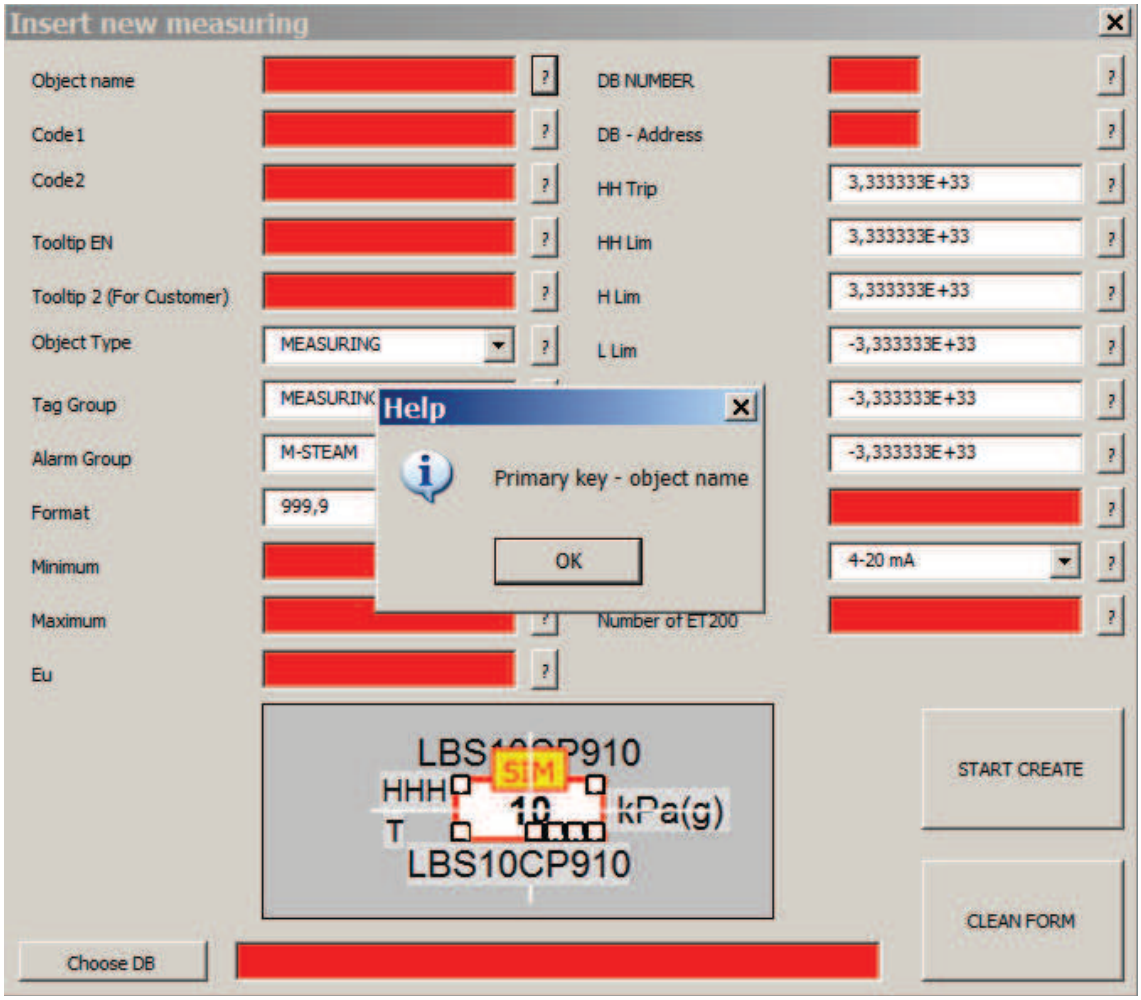


Obrázek 13: Výběr DB

4.3.2 Nápověda

Formulář má poměrně hodně položek a u některých nemusí být hned jasné, co se do nich má doplnit. Proto jsem vytvořil ještě nápovědu (tlačítka s textem otazníku), aby uživatel hned věděl, co má doplnit. Po kliknutí se zobrazí, co dané pole znamená, popřípadě co má obsahovat (Obrázek 14).

Deset polí se vyplňuje defaultními hodnotami, které uživatel může změnit, ale pokud nezmění, nic se nestane. Následně vygenerovaný objekt bude pracovat i tak správně. Defaultní hodnoty u některých polí jsou vyplňovány na základě znalosti programu v PLC. Protože objekt ve WinCC je silně svázán s programem v PLC, je vhodné se orientovat a znát oba systémy, jak fungují. Podle toho se pak dá lépe zautomatizovat celý proces vytváření objektů. Programátor by měl vědět, co doplňuje a s jakou technologií pracuje. Nápověda má pouze informační charakter.



Obrázek 14: Ukázka nápovědy

4.3.3 Vyčištění formuláře

Může se občas stát, že vyplníme všechny položky a pak zjistíme, že jsme vše vyplnili špatně. Budeme potřebovat vyčistit formulář. K tomu slouží tlačítko CLEAN FORM. Jeho jediným úkolem je vyčistit formulář. Vyčištění formuláře v tomto případě znamená inicializovat ho na původní načtený formát. Je samozřejmé, že během průběhu generování objektu je tlačítko neaktivní.

4.3.4 Export alarmů

Export alarmů pro objekt se podřizuje struktuře, která je nachystána ve WinCC. Podle toho je třeba navrhovat výsledný tvar na generování alarmů. Proto je důležité si nejdříve zjistit daný formát, následně programovat samotnou funkčnost. Samotný formát je složitý a obsahuje spoustu sloupců, které slouží k tomu, aby se vědělo, kam se objekt v alarmovém listu zařadí, jaká alarmová zpráva bude zobrazena při splnění podmínky pro alarm.

Alarmový list je poměrně rozsáhlý a mocný systém pro různá hlášení ze všech snímačů, spotřebičů, stejně tak slouží jako takový kontrolní prvek pro celou procesní technologii. Proto je vhodné generování alarmů pro jakýkoliv objekt nezanedbat a dbát na správnou formulaci do všech sloupců, které mají být naplněny. Je to důležité z toho důvodu, že když se stane v procesní technologii jakýkoliv problém, jde se vždy nejdřív do alarmových hlášení a hledá se tam důvod problému.

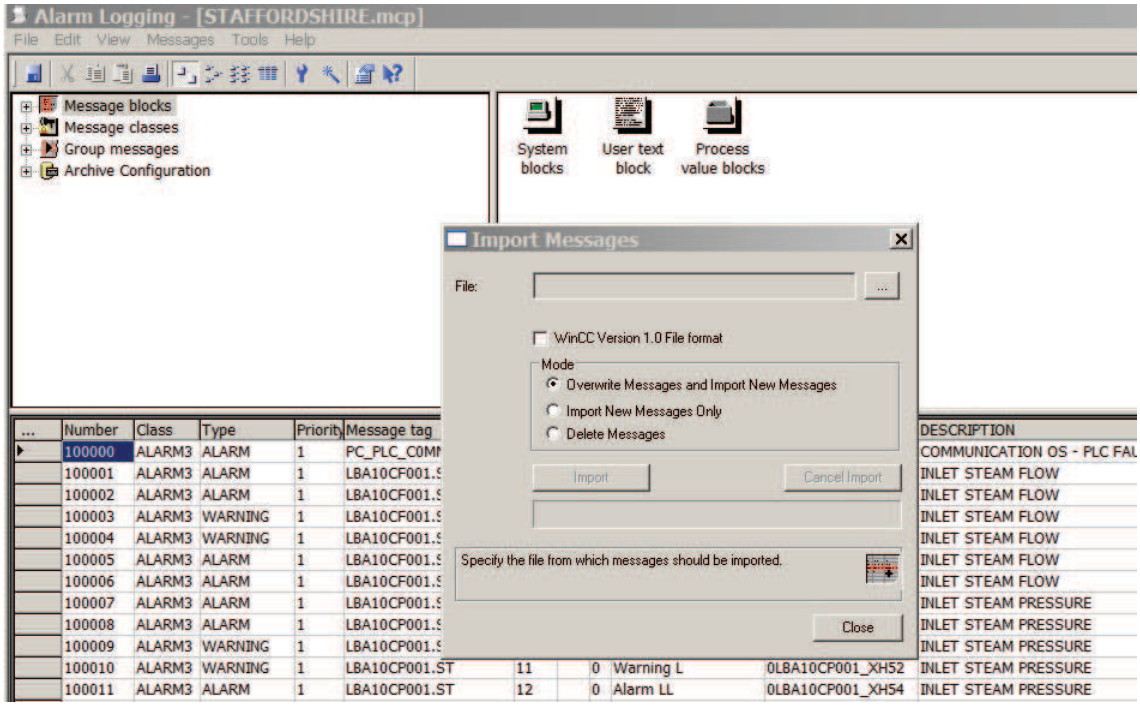
Když se zná formát pro generování, přistupuje se k naprogramování dané funkčnosti. Jak již bylo řečeno, očekává se na výstupu *.txt soubor s požadovanými daty. Funkce tedy musela umět vytvořit nejen *.txt soubor, ale naplnit ho a uložit na místo, kam si operátor zvolí. Celou tuto funkčnost zajišťuje funkce *CreateAlarms()*.

Ukázka kódu funkce **CreateAlarms()**:

```
'*********************************************************
'Načteme si vyplněné položky z formuláře
    strName = UserForm1.txtObjName.Text
    AlarmGroup = UserForm1.txtAlarm_group.Text
    Code1 = UserForm1.txtCode1.Text
    Code2 = UserForm1.txtCode2.Text
    TooltipEN = UserForm1.txtTooltipEN.Text

'*********************************************************
'Pokusíme se připojit
    Set dbProjectManager = OpenDatabase(strFilePath)

    'Nejdříve smažeme případný starý dotaz o stejném názvu
    Call DeleteMyQuery("Alarm", strFilePath)
    Set SqlQuery = dbProjectManager.CreateQueryDef("Alarm")

    'nastavime celou cestu
    strFilePathTxt = strFilePathTxt + "Output\"


'*********************************************************
    'Nastavíme možnost uložení
    'Otevřeme dialogové okno uložit jako
    UserForm1.dlgSaveAs.ShowSave
    strFile = UserForm1.dlgSaveAs.FileName
```

```
'*********************************************************
    'Vytvoříme txt soubor
    Set csvFile = fso.CreateTextFile(strFile +
"Alarm_.txt", True)
```

Po exportu alarmů se operátor přesune do WinCC, kde si spustí Alarm List. Tam je nástroj na import nových alarmových hlášení, popřípadě může stará hlášení přepsat.

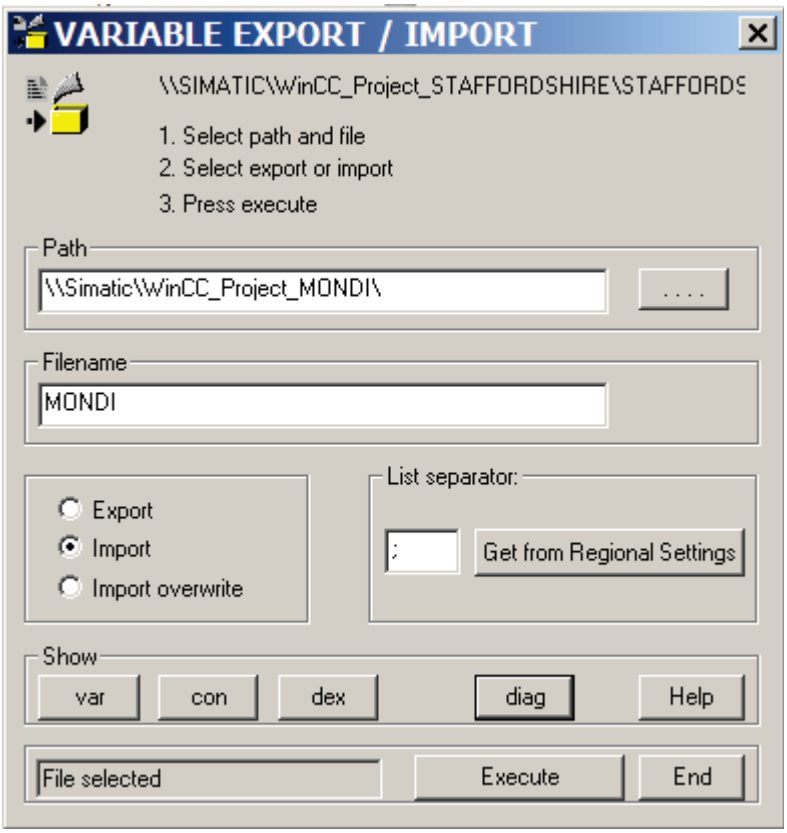


Obrázek 15: Import alarmů přes Alarm Logging ve WinCC

4.3.5 Vytvoření tagu

Programovací jazyk Visual Basic dovoluje přímé vytvoření tagu pomocí systémových funkcí. Pro naše účely se dané funkce nehodí. Nedovolují totiž vytvořit strukturovaný tag. Musí se tedy vytvořit soubor, který dovolí pak danou strukturu naimportovat do WinCC. K tomu slouží nástroj *VARIABLE Export Import* (Obrázek 16).

Nejdříve je potřeba si nastudovat, v jakém formátu má daný soubor být, jakou má mít hlavičku a co má obsahovat. Teprve později se může přistoupit k samotnému programování.



Obrázek 16: Nástroj pro export/import tagů

Pokud chceme importovat tagovou strukturu, musíme mít celkem tři soubory. Ty musí končit na *_vex.csv, *_dex.csv, *_cex.csv. Jak mají vypadat a co musí obsahovat, se nejlépe dozvíte tak, že si z vytvořeného projektu vyexportujete tyto soubory pomocí tohoto nástroje. Jedná se o složitou strukturu, která je uložena v každém měření. Proto zde budou popsány alespoň slovně.

- VEX – v tomto souboru jsou uloženy všechny informace potřebné k vytvoření tagu pro měření. Tento soubor je generován naší aplikací.
- DEX – obsahuje definici struktury objektu, na základě které se vytvoří tagy ve *_vex.csv souboru. Pokud tento soubor není detekován, aplikace ho vytvoří.
- CEX – ten určuje komunikaci a název připojení, do kterého se náš objekt naimportuje. Tento soubor není povinný.

Celý tento proces vytvoření požadovaných souborů zajišťuje mnou naprogramovaná funkce *CreateTagMeasuring()*.

4.3.6 Vytvoření grafického objektu

Další velice důležitou funkcí je samotné vygenerování grafického objektu. Zde využijeme šablonu objektu, která byla popsána v kapitole 2. Před samotným generováním objektu totiž musíme mít vzorový objekt, podle kterého budeme generovat nový objekt a naplníme požadovanými daty.

Daná šablona musí být umístěna na souboru *test.pdl*, ze kterého spouštíme skript na vytvoření grafického objektu. Pomocí skriptu pak přistupujeme nejdříve k samotnému objektu jako celku, a poté postupně vyplňujeme jednotlivé vlastnosti grafického objektu dle toho, co jsme vyplnili do formuláře.

Až je celý objekt vytvořen, zkopíruje se na nové prázdné *measure.pdl*. Odtud si ho můžeme následně vzít a zkopírovat do našeho hlavního projektu.

O správný chod celého procesu a následné vytvoření funkčního objektu se stará funkce *CreateMesObject()*. Pro ukázku zde dávám krátký výpis kódu z dané funkce, aby se ukázalo, jak se přistupuje k jednotlivým vlastnostem celého objektu.

```
'*******************************************************
'Zkopírujeme měření do našeho nového připraveného PDL

Application.Documents.Item(intWhereScripts).CopySelection
    ActiveDocument.PasteClipboard

'*******************************************************
'začneme načítat jednotlivá data do objektu měření
 ActiveDocument.Selection.Item(1).ObjectName.value = "AMB_"
& strName
     Set objMyObj = ActiveDocument.HMIObjects("AMB_" &
strName)

With objMyObj
'naplníme jednotlivé vlastnosti našeho objektu měření
     .Properties.Item("Index").value = Address_object
     .Properties.Item("Obj_Name").value = strName
     .Properties.Item("Code1").value = strCode1
     .Properties.Item("Code2").value = strCode2
     .Properties.Item("Format").value = Format
     .Properties.Item("min").value = R_Min
     .Properties.Item("max").value = R_Max
     .Properties.Item("EU").value = " " & Eu & " "

      With objMyObjTrigger
                .CycleType = hmiVariableCycleTypeUserCycle1
      End With
End With
```

4.3.7 Průvodce v aplikaci

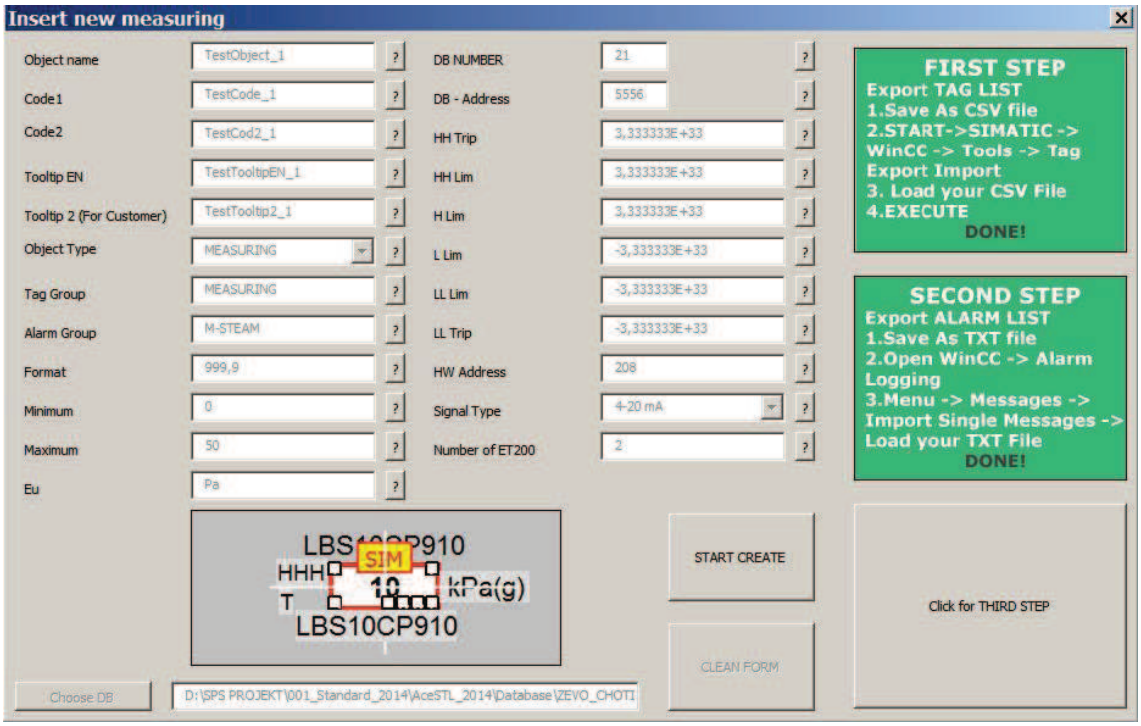
V minulých podkapitolách byl vysvětlen postupný popis tvorby všech částí, které stojí v pozadí při tvorbě objektu. Nyní bude představen průvodce, který programátora informuje, v které fázi se nachází tvorba objektu a je informovaný o tom, co se právě děje.

Význam aplikace se totiž bude uplatňovat až v závěru celého projektu, kdy přicházejí na řadu revize a možné přidávání dalších objektů. Pak se vyplatí mít nástroj, který bude navíc upozorňovat, co přesně se má udělat a jak postupovat. Na následujících obrázcích bude vše ukázáno.

Obrázek 17: První krok tvorby objektu měření

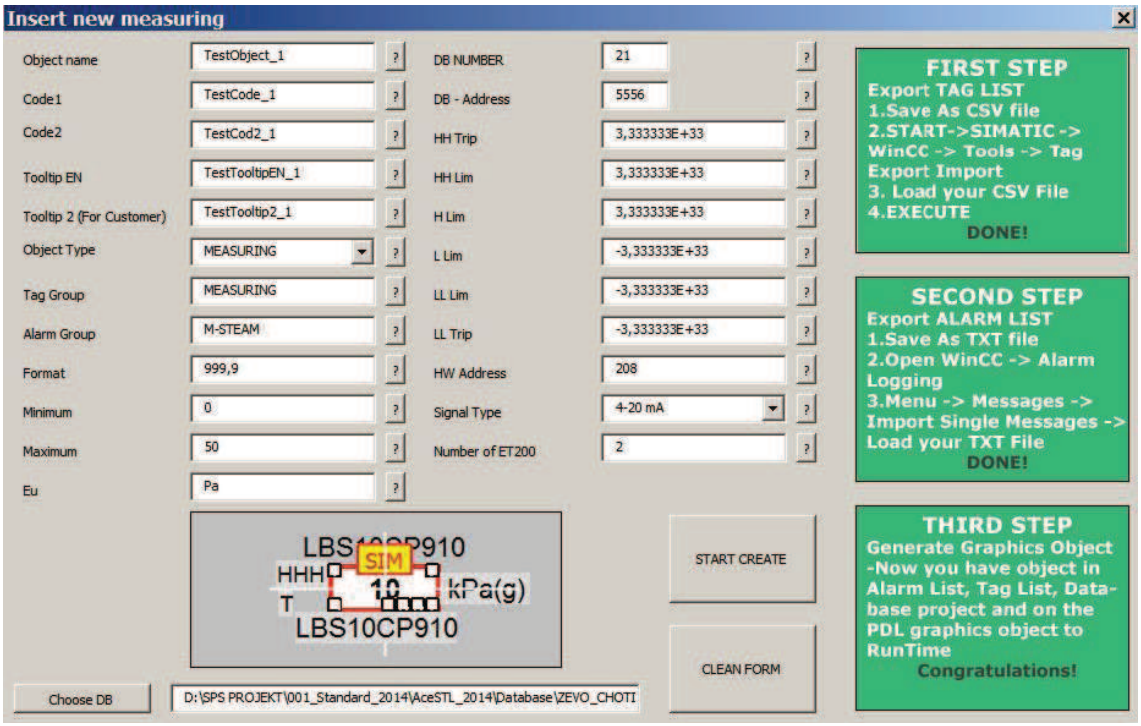
Všimněme si, že při prvním kroku (Obrázek 17) jsou všechny formulářové prvky zamčené a nelze je již měnit. Stejně tak nejde měnit cesta k databázi. Tento efekt má čistě bezpečnostní charakter, protože během samotného přidávání celého objektu dochází k průběžnému ukládání do databáze projektu a čtení dat z formuláře. V prvním kroku dochází k vytvoření tagové struktury pro objekt.

V druhém kroku (Obrázek 18) vytváříme alarm list. Zde je uživatel opět požádán prvně o uložení a v návodu požádán o import do WinCC. Až tak provede, může kliknout na třetí krok.



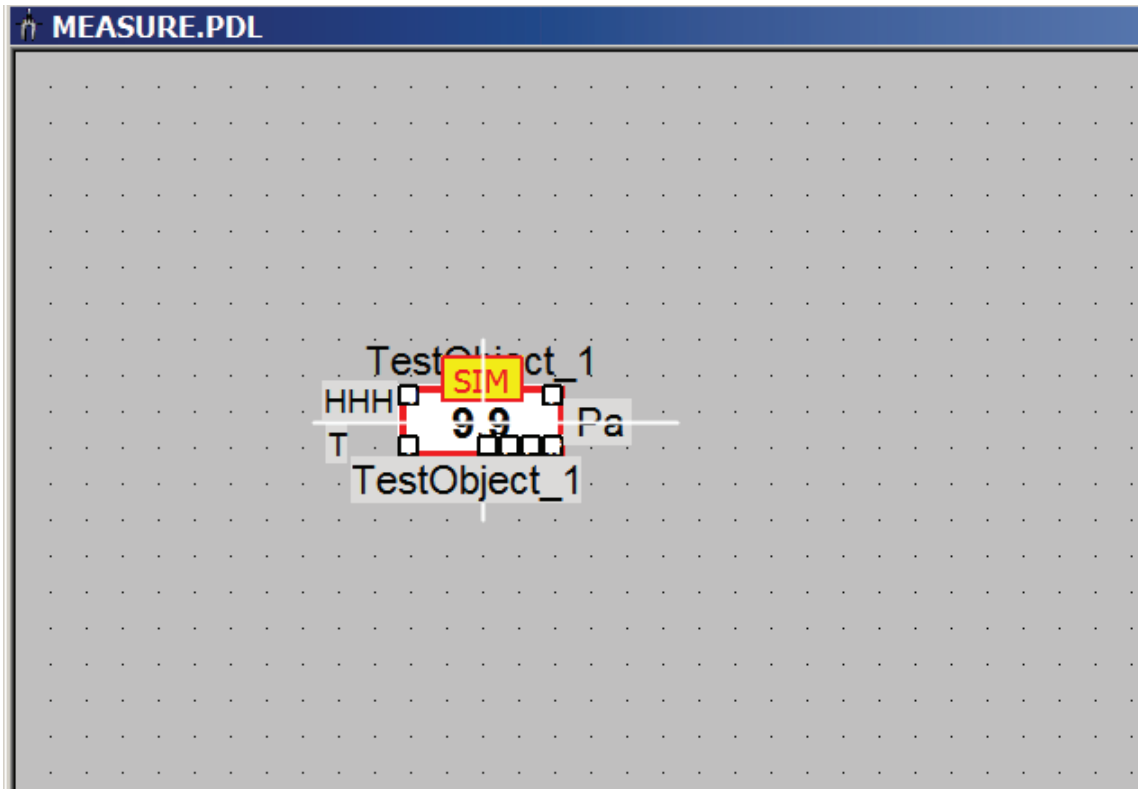
Obrázek 18: Druhý krok tvorby - alarm list

Ve třetím kroku (Obrázek 19) již máme vytvořený i grafický objekt a vše se úspěšně uložilo do databáze.



Obrázek 19: Třetí krok – dokončení

Kromě upozornění, že vše proběhlo v pořádku, se objeví i nový *measure.pdl* soubor. Na tom máme námi vytvořený objekt (Obrázek 20) se všemi vlastnostmi, které byly zadány ve formuláři. Následný objekt již stačí zkopírovat na příslušnou obrazovku WinCC, kde se nachází naše vizualizace procesní technologie.



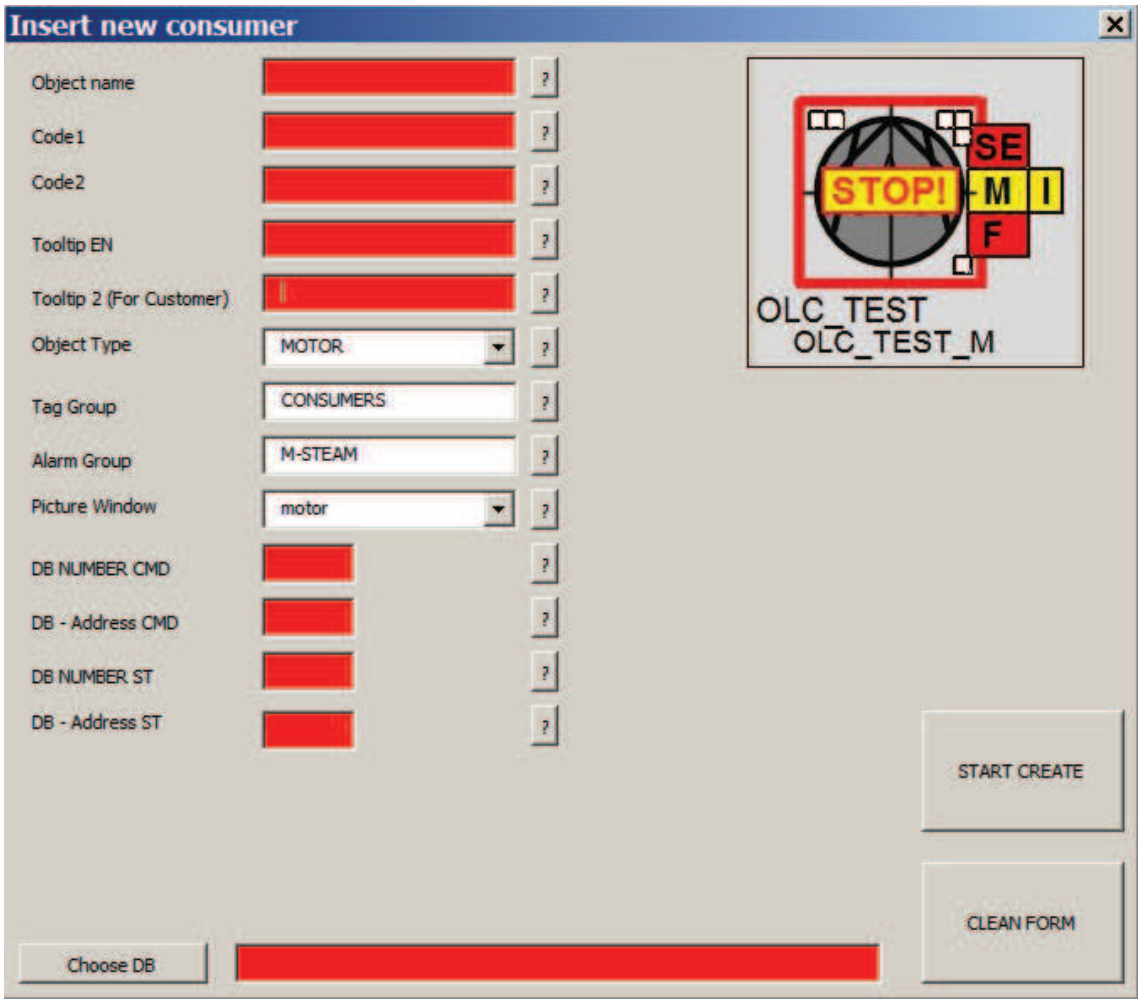
Obrázek 20: Vytvořený grafický objekt

4.4 Formulář pro objekt spotřebič

Formulář pro objekt spotřebič se vytvářel až po formuláři objektu měření. Už od počátku byl záměr, aby se daný formulář příliš neodlišoval a uživatel dané nadstavby se nemusel zbytečně učit s dalším nástrojem. Proto se opravdu dbalo, aby formulář pro generování objektu spotřebiče byl co nejvíce funkčně a designově podobný již vytvořenému formuláři pro objekt měření.

U tohoto formuláři se začalo volbou a rozmístěním prvků, do kterých se budou zadávat informace o novém objektu. Tyto prvky také měly vlastnost, kdy při nevyplnění budou textová pole svítit červeně. Povinností je samozřejmě vyplnit všechny položky. Splnění tohoto požadavku zajistí konzistentnost objektu v celém systému a samozřejmě bez vyplnění všech vlastností vás aplikace nepustí k vytvoření objektu. Každá vlastnost má totiž svůj význam, jinak by tam nebyla. Už v kapitole dva, kdy probíhal návrh objektu, se snažilo použit jen ty nejnutnější vlastnosti. Každá vlastnost navíc, která nemá svůj opodstatněný význam, je zbytečná a je třeba ji vyloučit z návrhu.

Naštěstí byla možnost si vyzkoušet si tvorbu či spolupráci na několika projektech, kde se s objekty pracovalo, dále možnost využít rady od zkušenějších spolupracovníků, kteří již absolvovali několik projektů. To vše značně ulehčilo rozhodování při určování, co je důležité a co naopak ne.



Obrázek 21: Formulář pro objekt spotřebič

Jak lze z obrázku vidět (Obrázek 21), máme zde mnohem méně vstupních polí. Další změnou je, že si můžeme vybrat, který objekt chceme generovat. Na základě volby v položce *Object Type* (jsou zde možnosti MOTOR a VENTIL) se vygeneruje grafický objekt ventilu nebo motoru. To, který objekt se bude generovat, nám zobrazí obrázek v pravém horním rohu formuláře. Rozmístění prvků zůstalo stejné. Každý prvek má svou nápovědu. Stejně jako u formuláře měření.

V tomto formuláři se při spuštění vyplňují 4 pole samostatně. Objekt spotřebič je velice svázaný s programem PLC, proto je zde snaha některé věci co nejvíce ulehčit a uživatel této aplikace mohl co nejdříve spustit samotné generování. Předem definovaná pole jdou změnit za vlastní hodnoty. Jejich původní hodnoty vychází ze zkušeností dané procesní technologie, kdy se zjistilo, že určité vlastnosti se opakují častěji, proto je výhodné je vyplnit předem.

Formulář má v sobě také tlačítka na zvolení databáze, dále vyčištění formuláře a v neposlední řadě tlačítko na spuštění celého procesu vytváření. Vyčištění formuláře má stejnou funkci jako u předchozího formuláře: má za úkol inicializovat formulář do původního stavu. V následujících podkapitolách si ukážeme proces celého vytváření. Už nebudeme ale popisovat všechny věci do hloubky, ale zaměříme se hlavně na odlišnosti oproti formuláři měření.

4.4.1 Export alarmů

Zde se export alarmů podřizuje již vytvořené struktuře ve WinCC. Hlavička i formát souboru jsou stejné jako u minulého formuláře. To značně ulehčilo další kroky. Vycházelo se tedy z toho, že na začátku *.txt (název si uživatel může zvolit sám) souboru se vytiskne hlavička dle struktury ve WinCC.

Co je ale jiné, jsou jednotlivé řádky v dané struktuře. Je jich mnohem více. Je to dáno tím, že objekt spotřebič má nejen vlastnosti stavové, ale také se může ovládat. To vše je třeba zapisovat do alarmového systému. Tím se vše značně zkomplikovalo. Na druhou stranu to bylo logické. Objekt spotřebič funguje v řídicím systému jako akční člen, je třeba ho sledovat více.

Bylo tedy zjištěno a následně naprogramováno, že u objektu spotřebič se musí do alarmového systému doplnit jak informace o stavu, tak i informace o povelech, jež byly zadány. O provedení celého procesu se stará funkce *CreateAlarmsOlc()*.

Po zavolání funkce se nejdříve uživateli položí dotaz na uložení souboru a následně se provede příkaz na vyplnění *.txt souboru daty vyplněnými ve formuláři.

4.4.2 Export tagu

Tagová struktura spotřebiče je samozřejmě složitější. Její požadavky byly ukázány v kapitole 2 (2.2.6). I tady byl vymyšlen způsob na import tagové struktury do WinCC, protože Visual Basic nepodporuje vytvoření tagové struktury přes své programové rozhraní, tj. nemá k tomu potřebné příkazy, které by to dovolily.

Zde se také pracuje se třemi soubory: *_cex.csv, *_dex.csv, *_vex.csv. Prvně jmenovaný soubor nehraje pro nás takovou roli. Druhý soubor už je důležitý. Obsahuje definici struktury tagu, který bude pak na základě *_vex.csv vytvořen. *Dex* soubor obsahuje jinou strukturu pro objekt spotřebič. Je ale možné, aby v sobě obsahoval struktury jak pro spotřebič, tak pro měření. Již bylo uvedeno, že pokud aplikace detekuje v místě ukládání, že daný soubor neexistuje, bude vytvořen. I tady to tak platí. Funkce po zavolání se nejprve zeptá, kam chceme ukládat naši tagovou strukturu. Následně vše zapíše do *_vex.csv souboru a pokud navíc detekuje, že *_dex.csv neexistuje, vytvoří ho. Daný soubor už bude obsahovat informace o strukturách jak pro spotřebič, tak pro měření.

Pro řešení celé této problematiky byla naprogramována funkce *CreateTagMeasuringOlc()*.

4.4.3 Vytvoření grafického objektu

Do objektu spotřebiče se vkládají jiné vlastnosti. A jak bylo v minulých kapitolách uvedeno, má i jinou strukturu než objekt měření. Na našem *test.pdl* souboru musíme mít předem vytvořenou šablonu objektu se všemi připravenými vlastnostmi, jejichž defaultní parametry přepíšeme pomocí našeho programu ve Visual Basicu. Stejně tak v tom objektu musí být importovány všechny skripty, které byly naprogramovány speciálně pro daný objekt. Pravidla generování zde jsou stejná jako u objektu měření. Po vygenerování se náš objekt objeví na novém prázdném *pdl* souboru, kde si ho můžeme zkopírovat do naší procesní technologie.

Zajímavostí je, že u objektu spotřebič se vyplňuje i název tzv. *picture window*, to je pro nás *faceplate*, na kterém se zobrazí například ovládací panel daného spotřebiče i s informací, v jakém stavu se nachází.

Dle názvu *picture window* se pak daný *faceplate* zobrazí. Můžeme tedy pro dva stejné spotřebiče mít rozdílné *faceplaty*. Záleží, co si připravíme a chceme využívat. V každém *faceplatu* můžeme mít integrované jakékoliv vstupy, výstupy, skriptovací funkce. *Faceplate* může být jednoduchý, stejně tak velice složitým ovládacím prvkem. Platí pro něho stejná pravidla návrhu, jako u tvorby samotného objektu. Je důležité si pečlivě rozvrhnout, co je na něm opravdu potřeba zobrazit, ať působí jednoduše a zároveň splní požadovaný účel. Zde jsem se celou touto problematikou již blíže nezabýval, byla řešena v minulé práci [4].

Celkově se může zdát, že programátor má vlastně neomezené možnosti. A je to vcelku správně. Je jen totiž na něm, jak moc si s objektem vyhraje, kolik je mu ochoten obětovat svého času. Problémem dnešního automatizačního průmyslu je, že na vše se tlačí a vše musí být rychle uděláno. Zde je tento problém samozřejmě řešen, nicméně podklad pro automatizované generování by se měl udělat kvalitní.

Celý proces generování zajišťuje naprogramovaná funkce *CreateMesObjectOlc()*. Jelikož máme na našem *test.pdl* více šablon objektů (měření, motor, ventil), je vhodné říct, že každá šablona má svůj unikátní název a v kódu se pak porovnávají názvy. Po nalezení správného podkladu, se teprve daný objekt vezme, a začne plnit daty. Samotný skript na plnění vlastností objektu je již jednoduchou záležitostí.

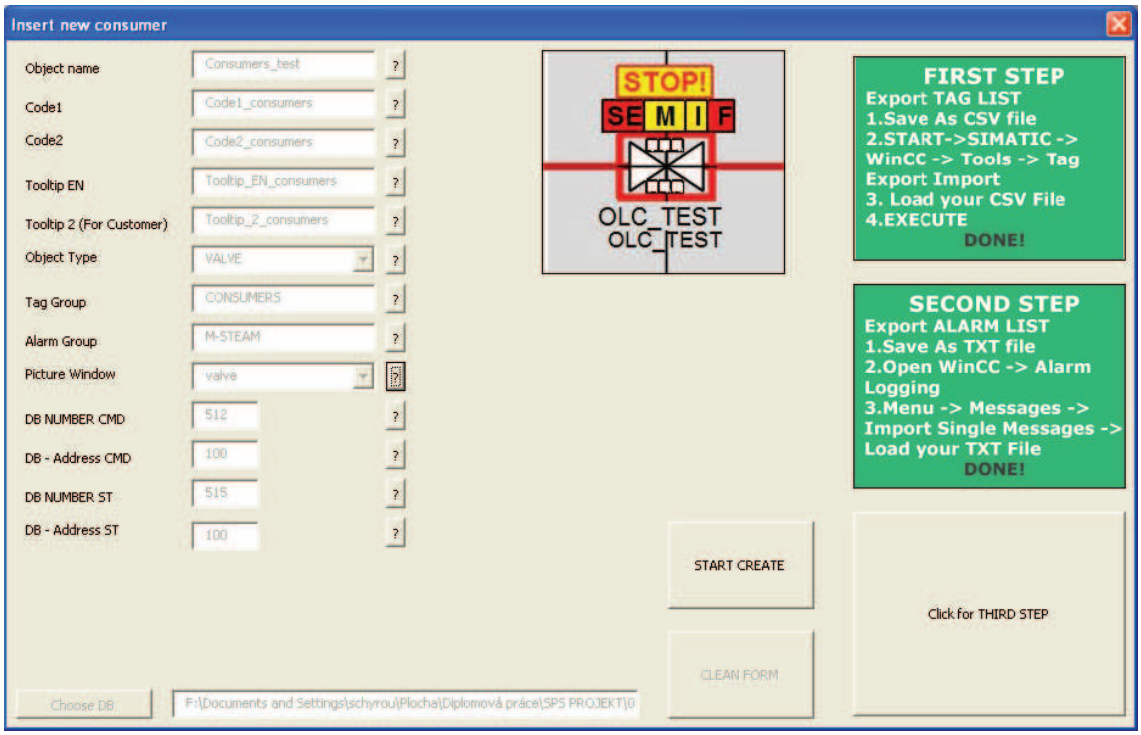
Ukázka kódu *CreateMesObjectOlc()*:

```
'*********************************************************
'zaktivujeme nový spotřebič a přitom se určí i typ
    If (ObjectType = 1) Then
        StrType = "MOTOR"
    Else
        StrType = "VENTIL"
    End If
    Set objMyObj =
Application.Documents.Item(intWhereScripts).HMIObjects(StrT
ype)

Application.Documents.Item(intWhereScripts).Selection.Desel
ectAll
objMyObj.Selected = True

'Zkopírujeme spotřebič do našeho nového připraveného PDL
Application.Documents.Item(intWhereScripts).CopySelection
    ActiveDocument.PasteClipboard

'začneme načítat jednotlivá data do objektu spotřebič
ActiveDocument.Selection.Item(1).ObjectName.value = "CONS_"
& ObjectName
Set objMyObj = ActiveDocument.HMIObjects("CONS_" &
ObjectName)
```

4.4.4 Průvodce v aplikaci

Objekt spotřebič stejně jako objekt měření zasahuje svými vlastnostmi do mnoha částí v celé procesní technologii řízení parních turbín, nelze tedy daný objekt vytvářet bez průvodce, který by uživatele aplikace vedl, co a jak dělat.

Průvodce je graficky totožný jako u objektu měření, proto si zde již nebudeme ukazovat všechny části. Na obrázku (Obrázek 22) je vidět náš průvodce v aplikaci. Princip kroků je také stejný. Nejdříve se musí vytvořit a integrovat tagová struktura, následně alarmový list (ve kterém je napojení na tagovou strukturu). Poté se vytvoří samotný grafický objekt a vše je uloženo do databáze k projektu. Potom je celý objekt hotový a připraven k nasazení do technologie.



Obrázek 22: Průvodce při přidání objektu spotřebič

5 WINCC

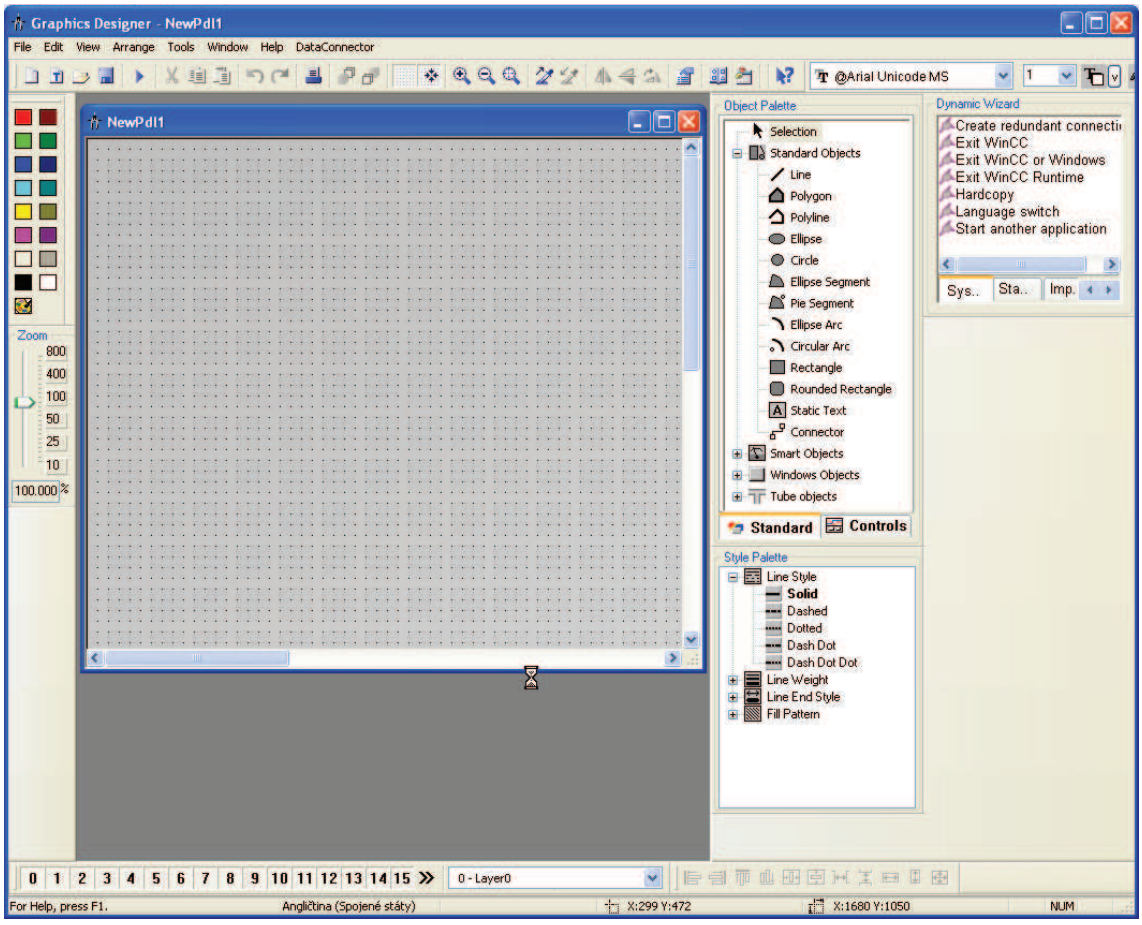
WinCC je vizualizační nástroj od firmy Siemens. Nástroj je to velice mocný a popsat to, co vše umí, by bylo na několik knih. Uvedu předem, že WinCC není nástroj inženýrský, ale spíše programátorský. Abyste plně využili všech jeho možností, hodí se umět programovat v jazyku Visual Basic nebo C.

Tohle vše se má ale časem změnit. Od verze 7.3 se plánuje skriptování více integrovat do uživatelského prostředí formou funkcí v menu. WinCC se tedy stane více nástrojem inženýrským. Už nebude třeba znát programovací jazyky. Na složitější věci se ale vždy bude hodit znalost programovacího jazyka.

5.1 Grafický nástroj

V minulých kapitolách jsme si ukázali vytvořené grafické objekty pro měření a spotřebič. Nyní bude ukázáno, v čem se dané objekty vytváří.

Na obrázku (Obrázek 23) je ukázáno grafické prostředí, ve kterém byly dané objekty vytvořeny.

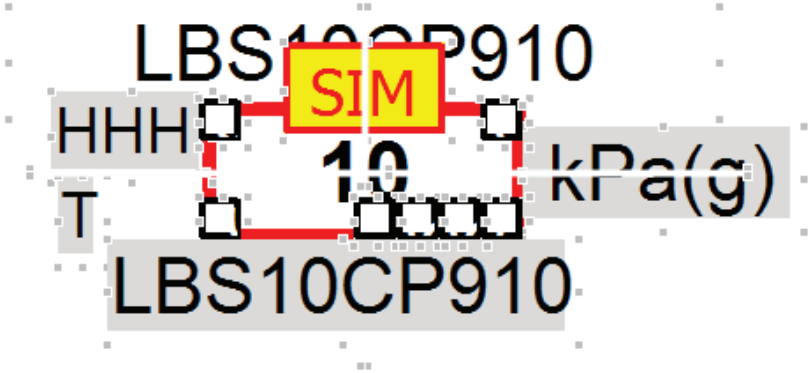


Obrázek 23: Grafické prostředí pro vytváření objektů

Dle ukázaných objektů v minulých kapitolách je vidět, že v daném prostředí toho jde namalovat poměrně dost. Je možnost vytvoření krásné grafické vizualizace kterékoliv procesní technologie. Je důležité mít na paměti, aby se ale vkládaly jen ty objekty, které jsou potřeba. V menu *Object Pallete* se nachází velice mnoho prvků, které lákají programátora (popřípadě grafika) použít ve vizualizaci. Z praxe a ze zkušeností se zákazníky se ví, že to ale není nejlepší přístup. Méně často znamená více. A opravdu je lepší mít střídmější vizualizaci, ale pohodlně funkční a propojenou se všemi komponentami, které WinCC nabízí – například alarmová hlášení.

5.2 Vytvoření objektu

Naše objekty, které byly ukázány v kapitole 2 (2), jsou složeny z mnoha menších tvarů. Na obrázku (Obrázek 24) je vidět objekt měření s šedými body. Značí ohraničení jednotlivých elementárních objektů, které jsou následně vloženy do jednoho velkého objektu.

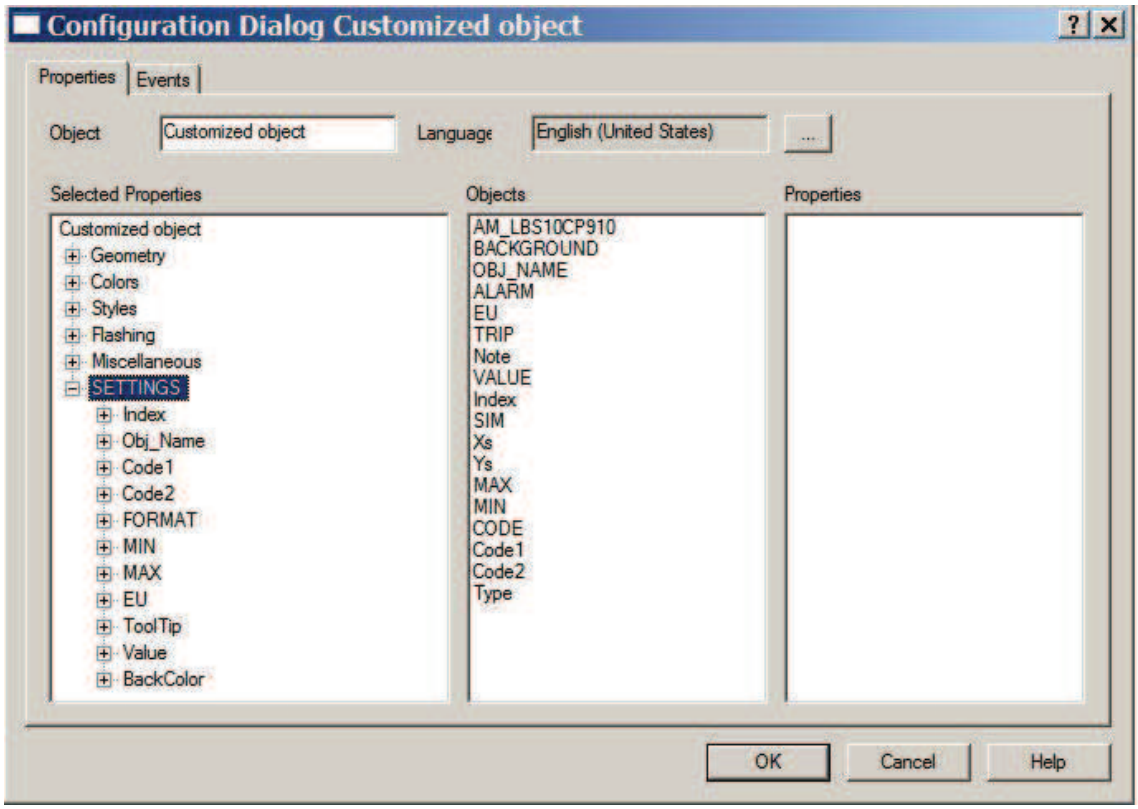


Obrázek 24: Objekt měření se všemi označenými částmi

Než se začne jakýkoliv objekt vytvářet, vyplatí si rozvrhnout, co vše musí obsahovat a co bude zobrazovat za hlášky. Následně podle toho vytvářet celý grafický objekt z menších objektů. Na závěr se všechny menší objekty dají do jedné skupiny s vybranými vlastnostmi (tzv. *Customized object*) a tím se z mnoha menších objektů stane jeden větší.

Customized object má tu výhodu, že po spojení elementárních objektů do jednoho, si můžeme vytáhnout z elementárních objektů jen takové vlastnosti, které se nám budou hodit v našem složitém objektu v globálním měřítku. Například u textového pole hlavní hodnoty si vytáhneme vlastnost barvy pozadí, u jiného textového pole zase naopak

pouze textovou část atd. Na obrázku (Obrázek 25) je ukázán konfigurační dialog, ve kterém si můžu vytáhnout požadované vlastnosti. Jde o to, že systémem definované vlastnosti mají název podle odpovídajících vlastností. V závislosti na objektu tedy názvy můžou být jiné. Ale uživatelsky definované vlastnosti, jejichž jména si můžeme zvolit, jsou stále stejná, ať už se jedná o objekt měření nebo objekt spotřebič. Celkové nastavení má pak v našem případě název SETTINGS.

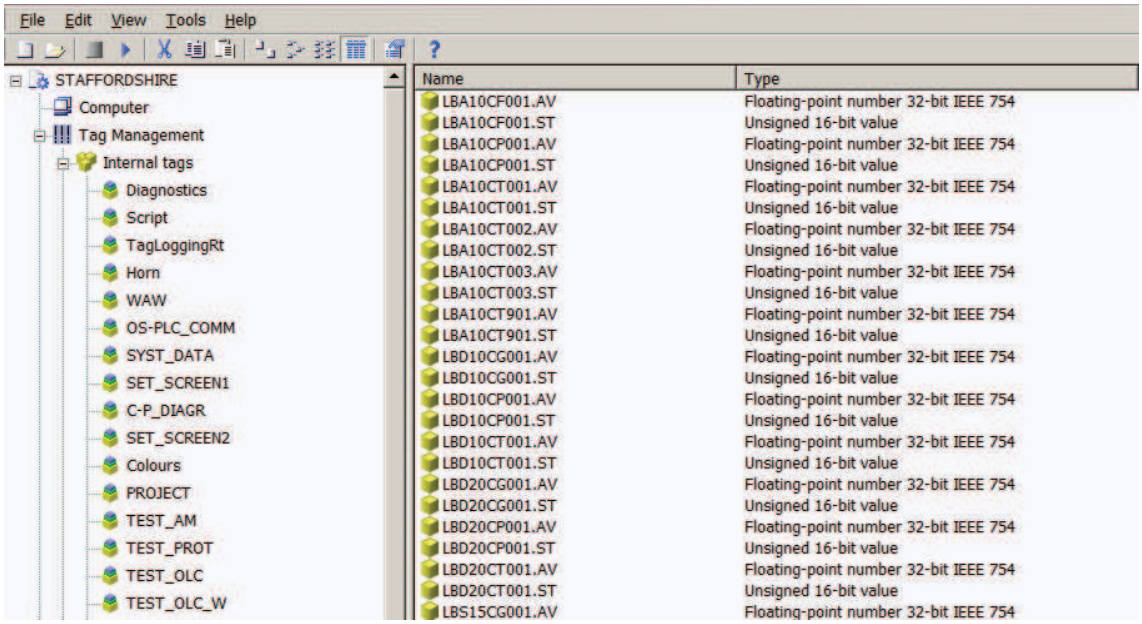


Obrázek 25: Konfigurační dialog pro vytažení vlastností pro Customized object

Každý objekt měření, popřípadě objekt spotřebič, by měl obsahovat tzv. *faceplate*. *Faceplate* je dialogové okno, které obsahuje bližší informace k našemu objektu. U objektu spotřebič byl uveden *faceplate* jako povinnost. Jistě by se náš objekt bez něj obešel, ale pro úplnost a možnost bližších zásahů se vždy doporučuje ho k objektu přiřadit.

5.3 Systém proměnných

Každý objekt potřebuje mít datové rozhraní. K tomu slouží tagy, které již byly zmíněny v minulých kapitolách. V podstatě se jedná o rozhraní mezi vizualizací a PLC. Objekt má svou tagovou strukturu. A podle ní se musí vytvořit tagy pro ukládání požadovaných dat. K tomu slouží tzv. Tag Management (Obrázek 26). Komponenta programu WinCC, která dovoluje pracovat se všemi proměnnými a nastavit jakého bude daná proměnná typu a hlavně na jaké místo v paměti PLC bude odkazovat.



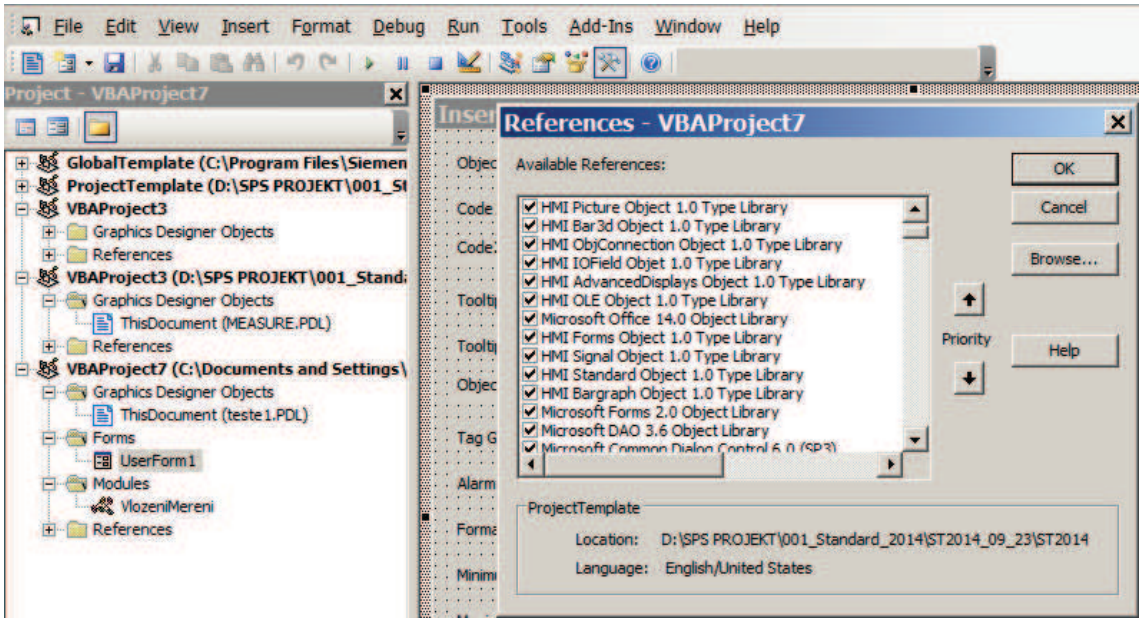
Obrázek 26: Ukázka prostředí Tag Managementu

Tagy můžeme vkládat ručně nebo pomocí nástroje *Variable Export/Import*, jak bylo uvedeno dříve. Samozřejmě mnohem rychlejší je mít nachystaný *.csv soubor, kde již máme všechny požadované struktury tagů pro náš projekt vygenerovány z databázového nástroje. A nástrojem *Variable Export/Import* pouze tagy naimportovat.

5.4 Prostředí pro programování

WinCC má zvlášť prostředí pro programování samotných funkcí do objektů a jiné prostředí pro programování samotných *pdl* souborů. V nich můžou tak vznikat další aplikace. Navíc tím, že je tu podpora spousta knihoven od Microsoftu, je tu možnost propojení s databází, tvorba formulářů, propojení s knihovnami WinCC, generování objektů a spoustu dalších praktických věcí.

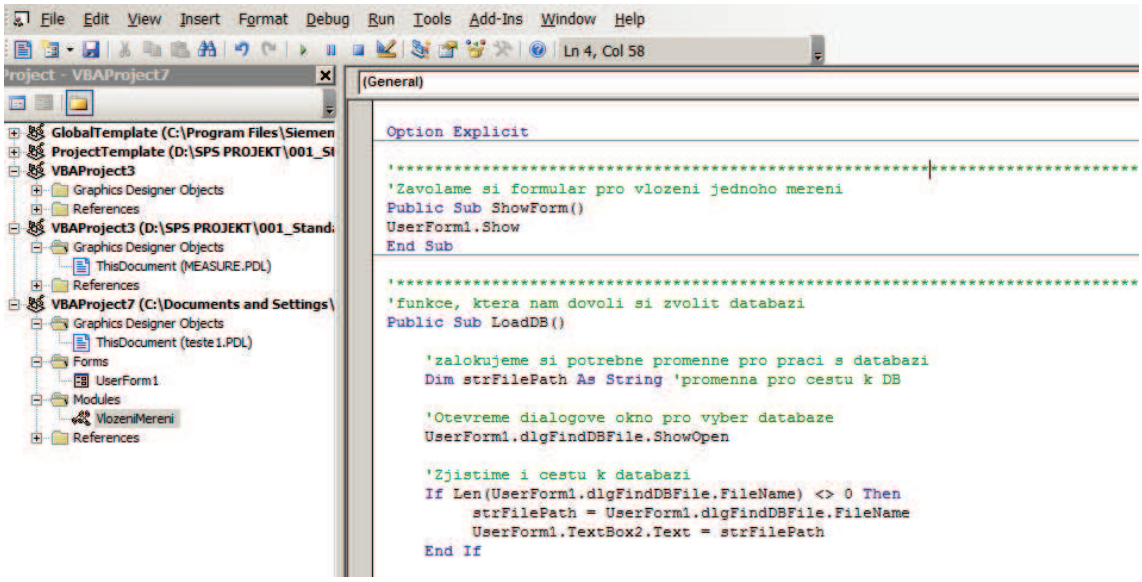
Prostředí, kde se programuje ve Visual Basicu, je totožné jako prostředí pro programování maker v MS EXCEL. Fungují tu tedy velice podobná pravidla. Je důležité si ale naimportovat správné knihovny. To se dělá pomocí tzv. *References* (Obrázek 27), kdy máme seznam všech dostupných knihoven a my si vybereme ty, které přesně potřebujeme. To je velice důležité, protože pak se nestane, že budete volat systémové objekty (například objekt sloužící k připojení databáze), při kterých vám program bude hlásit chybu.



Obrázek 27: Možnost načtení knihoven

Dále se podíváme na samotné prostředí Visual Basicu. Z obrázku (Obrázek 28) je patrné, že je opravdu totožné s prostředím, které můžeme znát z programování maker u MS EXCEL.

Platí tu stejná syntaxe a pravidla pro psaní komentářů, podmínek atd. Samozřejmě nabízí se srovnání jazyka C a Visual Basicu, protože WinCC se dovoluje programovat v obou jazycích. Zde bych uvedl, že jazyk C se hodí spíše pro samotné programování funkcí pro objekt, ale Visual Basic má výbornou základnu knihoven pro samotný soubor *pdl* (v něm může být uloženo více objektů dle libosti), ze kterého se pak volá samotná aplikace. Programování pak bylo o to pohodlnější.

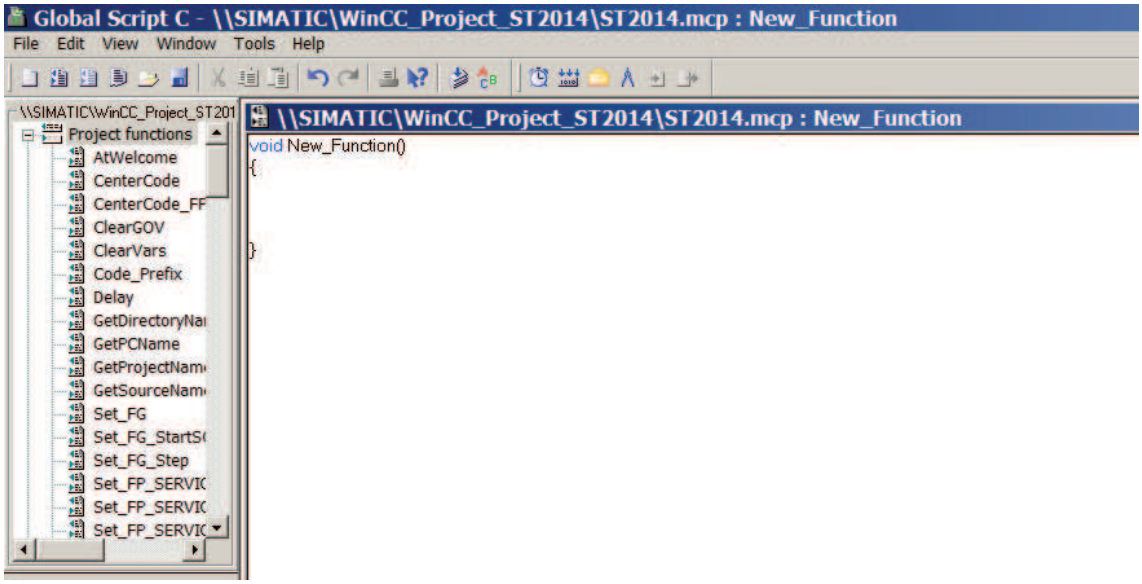


Obrázek 28: Programování ve Visual Basicu

Poslední částí programovacího prostředí je samotné programování funkcí do objektu. To probíhá samozřejmě v jiném programovacím prostředí.WinCC na to má i svůj vlastní programovací modul. Nazývá se *Global Script* (Obrázek 29).

Obrázek 29: Ukázka rozdělení položek v modulu GlobalScript

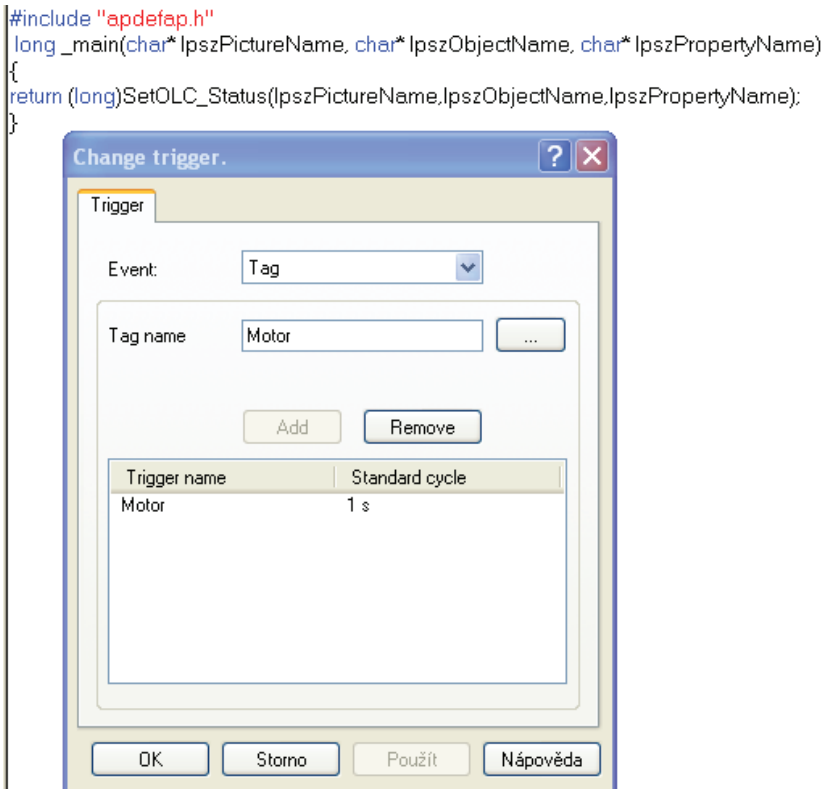
Už z obrázku je vidět, že je možné využít dvou editorů: programování v C-Editoru anebo ve VBS-Editoru. Na obrázku (Obrázek 30) je vidět C-Editor.



Obrázek 30: C-Editor v modulu Global Script

Je zde možnost si naimportovat stejné knihovny, jako například v MS VISUAL STUDIU. Syntaxe odpovídá klasické syntaxi jazyka C. Díky tomuto nástroji se WinCC stává nesmírně mocným nástrojem, protože si můžeme napsat C-funkci napojenou na vlastnost objektu, která může měnit vizualizační podobu celého objektu, ale zároveň i podávat hlášení o stavu objektu, zapisovat informace do souboru a další užitečné věci. Zároveň je důležité přistupovat k této možnosti rozumně. Příliš složité funkce na mnoha

objektech mohou zpomalit celý systém WinCC. Je vhodné už v samotném návrhu objektů a celé vizualizace najít kompromis. Často funkce reagují na událost z PLC, kdy dojde k zápisu do datového bloku, ten se překlopí do tagu a na tag zareaguje naše zmíněná funkce na objektu, protože má nastavený *trigger* (časovač) na 1 vteřinu.

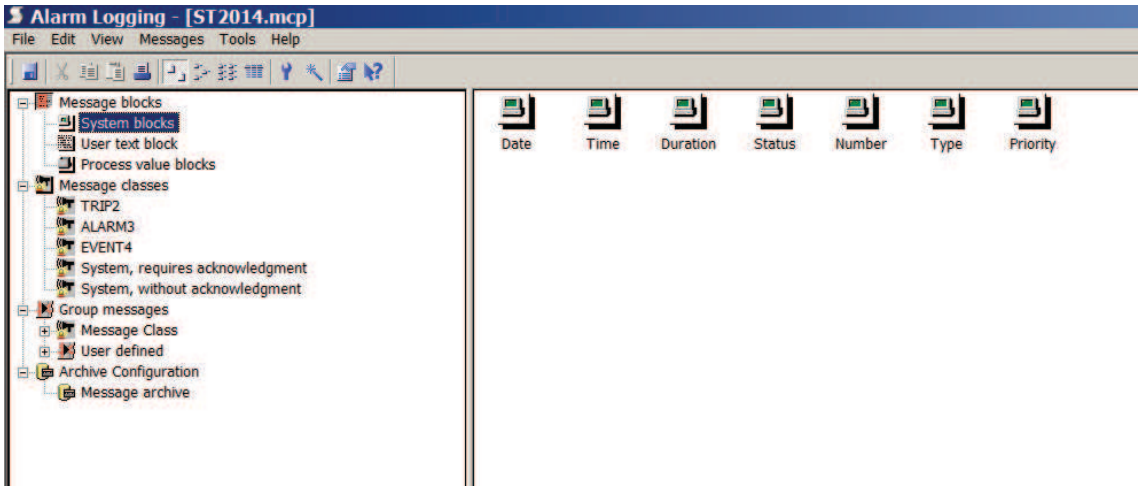


Obrázek 31: Ukázka funkce propojené s triggerem

5.5 Seznam alarmových hlášení

U našich objektů jsme automaticky počítali s alarmovým hlášením, ale je třeba si říct, kde dochází k nastavení celého alarmového hlášení, stejně tak vytvoření struktury hlášek pro naše potřeby.

Na tohle vše má WinCC také speciální modul. Nazývá se *Alarm Logging*. Zde je možné vytvářet bloky zpráv, řadit je do tříd dle důležitosti a samozřejmě mít celý seznam hlášení pro každý objekt zvlášť.



Obrázek 32: Alarm Logging a jeho menu

Tento systém může působit na první pohled trošku chaoticky, ale je to tím, že pro procesní technologii je třeba velmi často mnoho úrovní typů alarmových hlášení a tak je důležité mít možnost vytvořit složitou strukturu pro alarmy. Tento modul to umožňuje bez problémů. A když už máme jednou takovou strukturu vytvořenou, je dalším krokem možnost alarmová hlášení importovat do dané struktury. Pak dochází k významnému ušetření času. Můžeme sice samotná alarmová hlášení objektu do dané struktury vkládat ručně, ale hrozí zde, že uděláme spoustu chyb, zapomeneme na něco a je to zdlouhavé. Mít možnost daný proces zautomatizovat je významným krokem vpřed.

6 DALŠÍ MOŽNOSTI ROZŠÍŘENÍ APLIKACE

Touto prací vznikly nejen složité objekty do automatizace se všemi náležitostmi, ale i aplikace, která dané objekty umí vytvořit a provede uživatele rychlou implementací do celé procesní technologie. Nabízí se tedy otázka, co s tím dál. Každý objekt má své zastoupení také v PLC, proto první co napadne je to, že by bylo vhodné, kdyby aplikace generovala kód do této technologie. Ale zde je velice důležitá znalost rozvržení kódu v PLC, uživatel musí přesně vědět, kam se vygenerovaný kód má vložit, to za něj totiž nikdo neudělá. Řešením by mohl být průvodce, který by uživatele navedl, do kterých míst v PLC programu vygenerovaný kód implementovat.

Jinou možností je, že daná aplikace může sloužit pro procházení již vzniklých objektů uložených v databázi. Mohla by se tedy udržovat skrz operátorskou stanici, na které je nainstalováno pouze WinCC. Stejně by se tak mohl jakýkoliv objekt smazat z databáze. S takovou možností se musí zacházet opatrně. Proto při vytvoření této funkce by se hodil tzv. *logovací* systém, který by zaznamenával jakékoliv zásahy do databáze, aby se vědělo, kdy například daný objekt zmizel, i když tam měl být. Snáze by šlo vše pak spravit, popřípadě obnovit.

Poslední možností je, že by aplikace po napojení na databázi mohla vygenerovat všechny dostupné objekty, a uživatel by si zaškrtl jen ty, které chce vygenerovat. Daná funkčnost by našla velké uplatnění tehdy, kdy by došlo k rozsáhlejším změnám na stávajících objektech. Pak je ale nutné všechny nové revize implementovat do již existujících objektů anebo je jednoduše přepsat novým vygenerováním. Daná funkcionalita má více úskalí. Například tu, zda vygenerovat kromě grafického objektu i nový alarm list, tagovou strukturu. Musela by vzniknout i možnost implementovat jen část z celého celku.

Jak je vidět, možností na rozšíření je mnoho. Bude záležet na požadavcích firmy, stejně tak na dalších podnětů od zákazníků, zda by se daná rozšíření uplatnila a vyplatilo se je doprogramovat do aplikace. Objekty se budou časem měnit stále. Minimálně jejich vzhled, ale tím, že pro každý objekt máme vytvořenou grafickou šablonu, nebude v tomto ohledu žádný problém reagovat rychle na změny.

7 ZÁVĚR

Diplomová práce měla více cílů. Jedním z prvních cílů bylo nastudování procesní technologie parních turbín a jejich řízení. Ačkoliv se již tato problematika vzpomínala v mé bakalářské práci, zde jsem před samotným vypracováním aplikační nadstavby musel jít hloub v dané problematice. Bylo třeba pochopit, co objekt v automatizaci vše potřebuje ke svému chodu. Dále rozlišit, co všechno svými vstupy a výstupy ovlivňuje v celém systému. V úvodních kapitolách je tedy popisováno, jaké vlastnosti má daný objekt mít a kam všude svými vlastnostmi zasahuje.

Poté přišlo na řadu samotné grafické zobrazení objektů spotřebič a měření. Zde se samozřejmě vycházelo z norem a interních požadavků firmy. Dále se muselo dbát, aby každý použitý prvek měl svůj funkční význam a důvod použití. Samotné vytváření daných objektů vycházelo z letitých zkušeností spolupracovníků. Po vytvoření grafické stránky objektů, byl objekt upraven o různá nastavení vlastností doplněné skripty.

V další fázi se bylo potřeba zaměřit na samotné uložení všech vlastností. Objekty již byly vytvořené, tak se mohlo přejít k návrhu aplikace, která proces jejich vytváření celý zautomatizuje. Pro danou aplikaci jsme použili databází MS SQL a to z důvodu použití v průmyslu, kde danou databázi již používají. Byly tedy navrženy a popsány tabulky, odkud se dané vlastnosti budou čerpat. Nutno zdůraznit, že se opět vycházelo ze zkušeností z průmyslu, co je a co není třeba a daná struktura byla utvářena za pochodu, aby její využití bylo co nejefektivnější.

Pak přišlo na řadu studium alarmového a tagového systému pro dané objekty použité v procesní technologii parních turbín. Zde bylo úkolem zjistit, jak se celý systém používá a najít možnost rychlého začlenění do něj. Povedlo se to pomocí speciálních souborů (*.csv a *.txt), které dovolují obsahovat informace pro následný import do WinCC. Celé studium zabralo poměrně hodně času a bylo nutné sledovat, jak se objekty chovají za různých situací a zda jejich případná stavová hlášení opravdu přicházejí do alarmového systému, stejně tak, zda dochází k uložení hodnot.

Po analýze nejen objektů, ale i alarmového a tagového systému se mohlo přejít k dalšímu cílu diplomové práce. Tím bylo vytvoření aplikace, která dokáže všechny tyto prvky spojit a jednoduchým postupem vše vygenerovat a následně zaintegrovat do již vytvořeného a běžícího systému. Zde se dostáváme k systému řízení změn na probíhající stavbě (místo, kde již procesní technologie běží). Celý proces dotváření objektů se musel zkrátit na minimum času a hlavně aby fungoval automaticky.

Požadavkem bylo, aby aplikace běžela na systému WinCC, který slouží jako SCADA systém na všech zakázkách. Tedy byla vytvořena aplikace, které dovolovala volat aplikační nadstavbu z prostředí WinCC a dovolovala integrovat naše příkazy pro generování do menu a do formulářů. Menu se zobrazilo po otevření souboru test.pdl, ve kterém byla celá naše nadstavba uložen. Po otevření tohoto grafického souboru se v menu WinCC zobrazilo menu pro vložení objektu měření anebo objektu spotřebič. Uživatel pak mohl jednoduše doplnit vlastnosti objektu a celá aplikace ho jednoduše vedla při přidávání objektu do celého systému. Tím byla zajištěna použitelnost objektu,

soudržnost procesní technologie a zároveň se vše zaznamenalo do databáze k projektu. To má nesporné výhody pro zákazníka, protože na základě databáze všech spotřebičů a měření se generuje tzv. Signal transfer list, ve kterém je ukázáno, na jakých vstupech (popřípadě výstupech) jsou dané objekty připojeny, jejich názvy, vlastnosti a jiné údaje. Každá následující oprava pak vychází z toho, že se nejdříve podívá, na jakých svorkách je objekt připojen a zda z něj vůbec jde nějaký signál.

V závěru celé práce se ještě popisuje prostředí WinCC z pohledu nástrojů, jež byly využívány k tvorbě aplikace pro generování objektů se všemi vlastnostmi.

Práce mi ukázala, jak náročná může být integrace hotových objektů do již běžící procesní technologie, ale zároveň jak důležité je být důsledný v jejich vypracování. Z pohledu WinCC a i napojení na program v PLC jsem se dozvěděl velice užitečné informace, které se jinak získávají dlouholetou praxí. Díky ochotným spolupracovníkům z firmy Siemens mi bylo celé toto učení značně ulehčeno a jsem jim za to velice vděčný.

Literatura

[1] ISA-5.5-1985. *Graphic Symbols for Process Displays*. North Carolina: Instrument Society of America, 1985.

[2] DIN 2481. *Thermal Power Plants: Graphical Symbol*. Německo: Deutsche Institut für Normung, 1976.

[3] Database definition. *The Linux Information Project* [online]. 2006 [cit. 2014-11-25]. Dostupné z: http://www.linfo.org/database.html

[4] SYSEL, Radek. *Nadstavba programu WinCC - parametrizace PLC aplikace*. Brno, 2013. Bakalářská práce. VUT Brno. Vedoucí práce Ing. Radek Štohl, Ph.D.

[5] SIEMENS AG. *WinCC V7.0 SP1 MDM - WinCC: Working with WinCC: System Manual*. 2008, 1996 s.

Seznam obrázků

Seznam symbolů a zkratek

CS	Režim návrhu ve WinCC
Faceplate	Windows okno ve vizualizaci
MS SQL	Databáze od firmy Microsoft
Pdl	Koncovka souboru grafických souborů ve WinCC
PLC	Programovatelný automat
RT	Runtime režim ve WinCC
Tag	Proměnná ve WinCC
WinCC	Vizualizační nástroj používaný v průmyslu

Seznam příloh