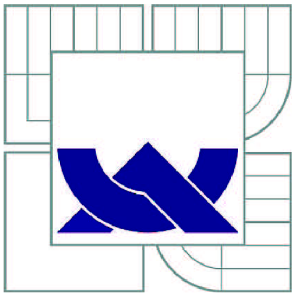
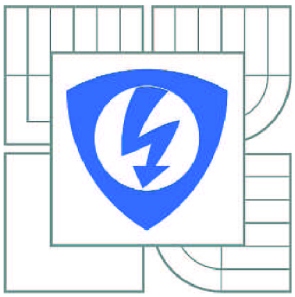


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

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

ZOBRAZOVÁNÍ ALARMOVÝCH HLÁŠENÍ SYSTÉMU SIMOTION PROSTŘEDNICTVÍM OPC AE

VISUALISATION OF ALARMS VIA OPC AE IN THE CONTROL SYSTEM SIMOTION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

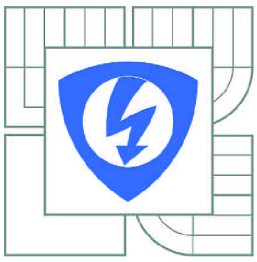
AUTOR PRÁCE — Bc. TOMÁŠ HRNČÍŘ

AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. FRANTIŠEK ZEZULKA, CSc.

SUPERVISOR

BRNO 2010



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. Tomáš Hrnčíř	***ID:***	83726
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Zobrazování alarmových hlášení systému SIMOTION prostřednictvím OPC AE

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte SW systém pro zobrazování alarmových hlášení řídicího systému SIMOTION pro řízení elektrických pohonů prostřednictvím systému WinCC. K disposici je programový systém OPC DA pro zobrazování procesních dat ve vizualizačním systému Win CC a dále system OPC AE pro zobrazování alarmových hlášení, avšak mimo WinCC. Úkolem je realizovat SW rozhraní - přechod mezi systémovým hlášením AE (Alarm Event) prostřednictvím OPC AE a vizualizačním systémem WinCC.
Proveďte funkční zkoušky realizovaného rozhraní a jejich vyhodnocení.

DOPORUČENÁ LITERATURA:

1. Siemens A.G .: SIMOTION. User manual.., 2006
2. Boček P.: Řídicí system Simotion. AUTOMA, č.7, 2008
3. Dočkal K.: Simotion. AUTOMA č. 4., 2004
4. Stianko M.: OPC v průmyslové komunikaci. AUTOMA č. 6., 2004

Termín zadání:	8.2.2010	***Termín odevzdání:***	24.5.2010

Vedoucí práce: prof. Ing. František Zezulka, CSc.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt:

V Diplomové práci je popsán řídicí systém SIMOTION, jehož výrobcem je firma Siemens. Je zde přiblížena koncepce tohoto systému a jeho vlastnosti. Dále se práce zabývá přenosem dat a alarmů mezi jednotlivými systémy, který je prováděn pomocí nejnovějších technologií za pomoci OPC (OLE for Proces Control), jehož aktualizací a specifikací se zabývá organizace OPC Foundation. Je zde uvedeno řešení problému s vizualizací alarmových hlášení ze systému SIMOTION do prostředí WinCC, které není klientem OPC Alarm & Events serveru.

Klíčová slova:

SIMOTION system, OPC Alarms & Events, OPC server, OPC klient, vizualizace alarmů.

Abstract:

In the Diploma Work is described the Simotion commanding system which has been produced by Siemens. Conception of the system and its features are explained. Further, the work deals with the data and alarm transfer among separate systems, an item which has been dealt with by the help of OPC (OLE for Process Control). Its update and specification have been covered with OPC Foundation organization. The alarm report visualization out of the Simotion system to WinnCC environment (which is not a client of OPC Alarms & Events server) is introduced here .

Key words:

Simotion system, OPC Alarms & Events, OPC server, OPC client, alarm visualization.

Bibliografická citace mé práce:

HRNČÍŘ, T. *Zobrazování alarmových hlášení systému SIMOTION prostřednictvím OPC AE.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 60 s. Vedoucí diplomové práce prof. Ing. František Zezulka, CSc.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma ***Zobrazování alarmových hlášení systému SIMOTION prostřednictvím OPC AE*** 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 § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne: **24. května 2010** .....
podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Prof. Ing. Františkovi Zezulkovi, CSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **24. května 2010** .....
podpis autora

Úvod

Diplomová práce je rozdělena do několika částí a je zaměřena na problém s vizualizací alarmů a událostí ve vizualizačním programu WinCC.

V první části se práce zabývá řídicím systémem SIMOTION, jenž je produktem firmy Siemens. Poukazuje na složení systému, možnosti použití a jeho vlastnosti.

Druhá část práce se věnuje problematice přenosu dat a alarmů prostřednictvím OPC specifikací, definovaných sdružením OPC Foundation. Zde je krátce zmíněna charakteristika specifikací a obecně popsána specifikace OPC Alarm & Events.

Další části se zabývají konkrétním řešením problému vizualizace alarmových hlášení systému SIMOTION, prostřednictvím OPC AE serveru a vytvořeného rozhraní do vizualizačního prostředí WinCC. Prvotně je uveden program vytvořený v prostředí SIMOTION Scout, který slouží jako prostředek k vyčítání dat a alarmů ze systému do OPC serveru. Krátce jsou zde uvedeny použité objekty potřebné ve vizualizaci a jejich konfigurace. Za tímto následuje výsledek práce v podobě rozhraní vytvořeného v aplikaci Visual Basic 6.

1. SYSTÉM SIMOTION

Systém SIMOTION nabízí optimalizovanou systémovou platformu pro automatizaci a řízení pohybu. Přitom je velký důraz kladen na aplikace řízení pohybu a technologické požadavky. Toho je dosaženo díky integrovanému systému řízení. SIMOTION je standardizovaný systém obsahující standardní komunikační rozhraní (Profibus, Ethernet) a jeho distribuovaná koncepce umožňuje téměř libovolně rozmístit jednotlivé součásti systému.

SIMOTION je především určen k řešení úloh typu Motion Control (MC) neboli k řízení pohybových os, jejich synchronizaci a realizaci vačkových funkcí.

Řídicí systém SIMOTION na jedné straně navazuje na dosavadní principy řešení úloh typu MC, na druhé straně však nabízí zcela novou koncepci úloh typu MC a logického řízení [1]. Současně je množina jeho funkcí nepřetržitě rozšiřována o specifické, technologicky orientované funkční bloky, které lze při použití v jednotlivých strojních úlohách jen vhodně parametrizovat a bez dalšího vývoje ihned použít [1].

Systém SIMOTION tedy slučuje tyto funkce:

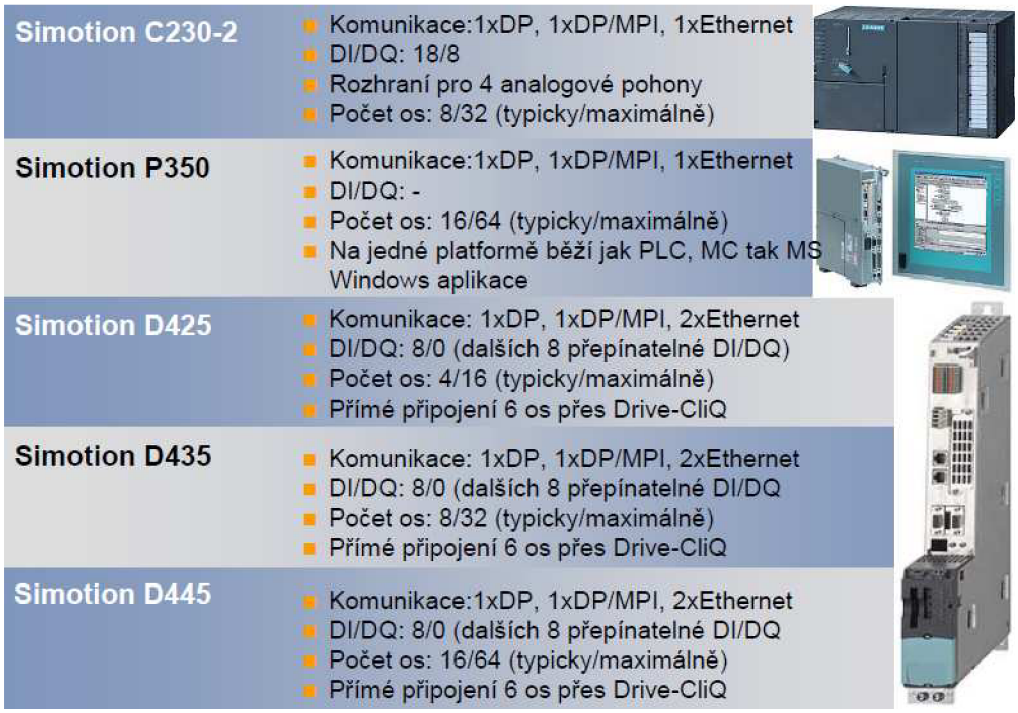
- PLC logické funkce
- Funkce řízení pohybu (např. synchronizace, polohování, …)
- Technologické funkce (např. regulace tlaku, teploty).

Hardware je odstupňován nejen podle výkonnosti, symbolicky měřené počtem řiditelných os, parametry procesorové jednotky nebo konfigurací hardwaru, kromě toho je dostupný v těchto hardwarových platformách:

1. **SIMOTION C** (Controller – based) – modulární varianta v designu SIMATIC S7-300
    - Pro místní a vzdálené řízení pohybu
    - Rozšiřitelné s pomocí S7-300 periferií
    - Univerzální pro číslicové a analogické pohony

2. **SIMOTION P** (PC – based) – otevřená varianta průmyslového PC
   - Pro místní řešení pohybu
   - Při požadavku otevřené architektury a PC řešení
3. **SIMOTION D** (Drive – based) – kompaktní varianta integrovaná do pohonu
   - Pro řešení související s pohonem nebo distribuovaným řízením pohybu
   - Pro jednotlivé osy
   - Pro komplexní řetězce os
   - Inteligentní číslicové pohony

Následující obrázek ukazuje různé platformy SIMOTIONu:



Obrázek č.1: Přehled HW platforem systému SIMOTION [5]

Oblast použití systému SIMOTION:

Řídicí systém SIMOTION je vhodné použít ve strojích s velkými požadavky na dynamiku pohybu jejich komponent [1]. Mnohdy je jeho použití motivováno také snahou snížit náklady a zrychlit vývoj stroje, a to i v případech, kdy z pouhého

hlediska výkonu by stačil např. jednoduchý řídicí systém v kombinaci s inteligentními frekvenčními měniči [1].

Účinný řídicí systém vyžaduje, aby dnešní požadavky a příští trendy na poli produkce strojů zahrnovaly použití otevřeného řídicího konceptu. SIMOTION je integrovaný Motion Control systém zaměřený na automatizaci výrobních strojů. Je integrovaný na poli strojírenství, programování, komunikaci, data managementu a tím tvoří jednotný systém.

Možnosti systému SIMOTION jsou zvýšeny spoluprací s řídicími systémy MASTERDRIVES, SIMODRIVE a SINAMICS [8].

Oblasti, kde se nový systém kontroly pohybu nejlépe osvědčuje [8]:

- balící stroje
- stroje pro výrobu plastů
- lisy
- textilní stroje
- tiskařské stroje
- stroje používané v dřevařském, sklářském nebo keramickém průmyslu atd.

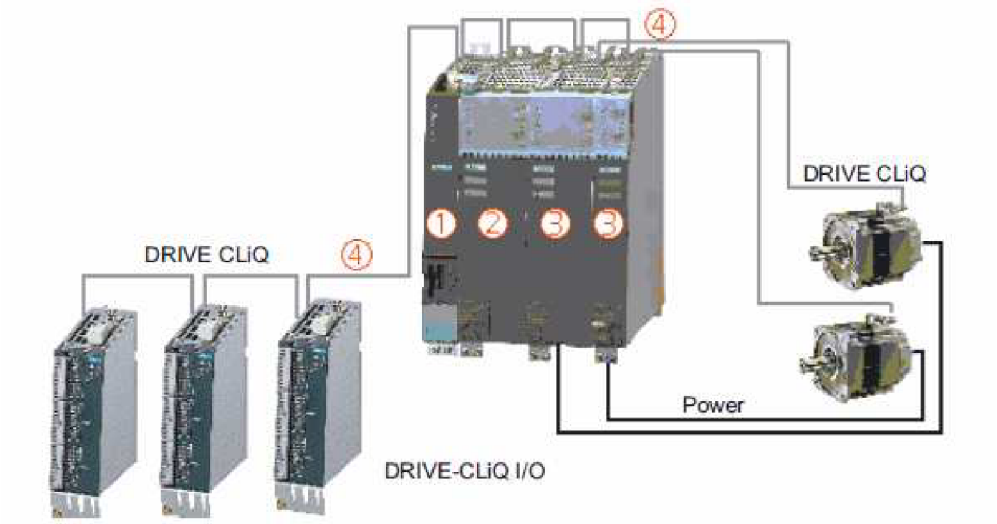
1.1 SIMOTION D4XX

Řídicí systém SIMOTION je koncipován tak, jak je zvykem výrobků společnosti Siemens, a to jako distribuovaný včetně napájecích jednotek i vlastních měničů, standardně vybavený rozhraním pro sítě Ethernet a Profibus, s možností připojení vizualizačních prostředků.

SIMOTION D je základní varianta pohonů SIMOTION řídicího systému pohybu, založeného na nové rodině pohonů SINAMICS S120. Obsahuje vlastní řídicí jednotku SIMOTION a regulační obvody pro několik měničů SINAMICS Integrated (až pro 6 os v servo režimu řízení), který přebírá funkci řídicí jednotky frekvenčního měniče SINAMICS S120. SIMOTION a SINAMICS Integrated jsou v modulu propojeny přes integrované rozhraní PROFIBUS. SIMOTION se stará o funkce PLC, ovládání a polohovou modulaci. SINAMICS provádí řízení motorů

pomocí otáčkové a proudové zpětné vazby regulace. Řídicí jednotka uvedeného systému a vlastní měniče SINAMICS S120 jsou připojeny na nově vyvinutou sběrnici Drive-CLiQ. Rovněž je možné připojit další měniče přes standardizované rozhraní Profibus-DP, či připojit systém do sítě Ethernet. Předností systému SIMOTION D je velmi snadný postup od uvádění do provozu, přes ladění regulátorů až po jednoduchou realizaci tahových, polohových a synchronizačních úloh. Systém SIMOTION může řešit běžné logické algoritmy a velké množství algoritmů pro řízení pohybu (Motion Control). S využitím standardního souboru funkcí je možné generovat požadované průběhy otáček, vytvářet vačkové průběhy rychlosti atd. Ke každé ose lze dále libovolně doplňovat licence pro různé složité technologické funkce, jako např. řízení polohy, synchronizaci, složité vačky atd. SIMOTION D se hlásí stejně jako SINAMICS S120 k Totally Integrated Automation (TIA) konceptu, tudíž poskytuje obsáhlý soubor automatizačních bloků. TIA – Plně integrovaná automatizace je strategie (filozofie/architektura) v automatizační technice vyvinutá v roce 1996 firmou Siemens. Tato strategie definuje interakce jednotlivých rozsáhlých komponent, SW nástrojů a služeb. Interakce provádí integraci na všech čtyřech úrovních automatizace: Management level, Operator's level, Controller's level, Field level.

Následující obrázek ukazuje typickou skladbu systému s pohonem:



Obrázek č.2: Připojení modulů a os pohonu [7]

Zapojení se skládá z těchto komponent:

1. *SIMOTION D4* (řídicí jednotka) (1):

   Tato jednotka obsahuje programovatelný runtime systém SIMOTION a software pro řízení pohonů SINAMICS S120. SIMOTION D je vždy schopen řídit více os/pohonů.

2. *SINAMICS napaječ* (line modul) (2):

   Tento modul generuje DC napájecí napětí pro ostatní moduly z rozvodné sítě.

3. *SINAMICS výkonová jednotka* (motor modul) (3):

   Tyto moduly se používají pro řízení motorů.

4. *DRIVE CLiQ* (4):

   Rozhraní společnosti Siemens pro automatickou detekci komponent. Poskytuje napájení 24 V/450 mA pro enkodéry a měřící zařízení.

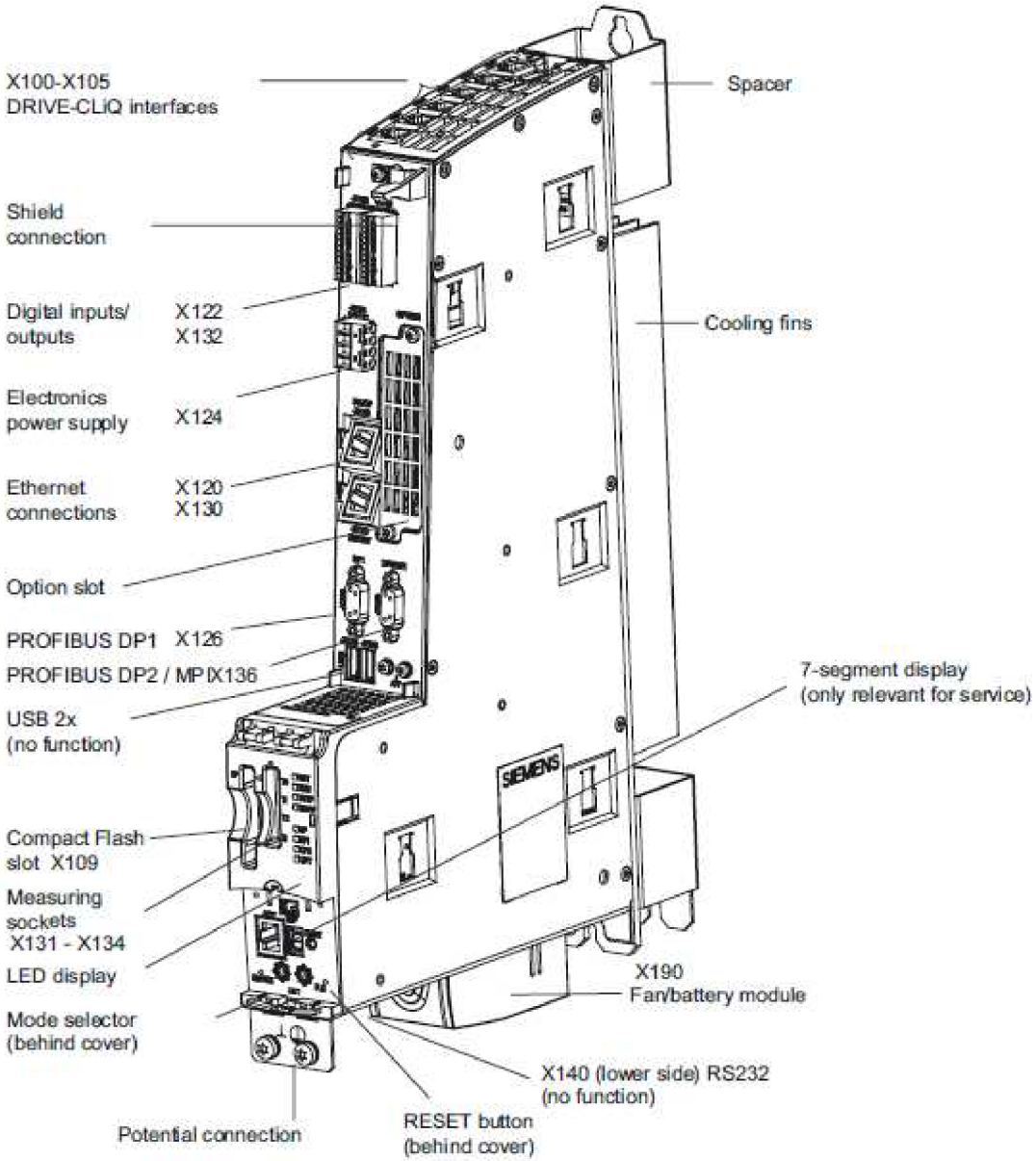
DRIVE CLiQ technologie

DRIVE CLiQ (DRIVE Component Link with IQ) je komunikační systém používaný k připojení SIMOTION D4xx s různými komponentami produktů od SINAMICS S120, jako line moduly, motor moduly, systémy enkodérů a speciálními DRIVE CLiQ I/O zařízeními.

DRIVE CLiQ nabízí tyto výhody [7]:

- Nezávislé rozšíření komponent
- Automatická detekce komponent řídící jednotky
- Standardizovaná rozhraní pro všechny komponenty
- Jednotná diagnostika komponent
- Kompletní servis komponent

Následující obrázek ukazuje řídicí jednotku s jejími rozhraními a komponentami čelního panelu (chybový a stavový display).



Obrázek č.3: Hardwarová platforma SIMOTION D445 [7]

Software pro SIMOTION

Základní funkčnost SIMOTION D je dodávána na Compact Flash kartě. Karta je vložena v zařízení a ukládá se na ni mimo jiné uživatelský program. Karta obsahuje následující:

SIMOTION runtime systém včetně těchto funkcí:

- Uživatelský programovatelný runtime systém (IEC 61131)
- Různé runtime úrovně
- Logické a aritmetické
- Komunikační funkce
- Funkce řízení pohybu

SINAMICS S120 řízení pohonu zahrnující následující funkce:

- Uzavřenou proudovou smyčku pro řízení momentu
- Uzavřenou smyčku řízení otáček
- Regulovaný napáječ

1.1.1 SIMOTION SCOUT

SIMOTION se programuje v uživatelsky přívětivém vývojovém prostředí SIMOTION SCOUT, v němž je integrováno rozhraní pro pohony (Drive ES) a STARTER pro práci s měniči SINAMICS [3]. Toto prostředí je začleněno do vývojového prostředí Step 7, z něhož se především využívají funkce HW-Config a NetPro. Hlavním nástrojem je tzv. Scout Workbench, ve kterém je možno prohlížet strukturu programu a modifikovat ji. Dále lze editovat jednotlivé funkce a vyčítat aktuální stavy proměnných. Nadstavbou SCOUTu může být např. konfigurační software pro programování vaček CamTool. SCOUT je určen pro všechny platformy systému SIMOTION.

Tento software umožňuje následující:

- Konfiguraci (Hardware, komunikace)
- Parametrizaci (včetně pohonů)
- Grafické programování (MMC, LAD, FBD, DCC)

- Textové programování (ST)
- Uvádění do provozu a diagnostiku

Řídicí systém SIMOTION je z hlediska programování založen na principu multitaskingového systému [3]. Na pozadí běží cyklicky zpracovávaný blok (tzv. background task) stejně jako hlavní cyklický blok OB1 v PLC SIMATIC. V němž je možné zpracovávat logiku a sekvence ne přímo závislé na pohonech a polohování. Další úlohy (tasky) jsou startovány, pozastavovány, znovu spouštěny a ukončovány nezávisle na tomto cyklu a právě ony jsou určeny pro řešení jednotlivých pohybů [3]. Tasky lze programovat pomocí grafických bloků podobných vývojovému diagramu nebo v programovacím jazyku Structured text (ST) dle IEC 61131-3 podobném např. Pascalu, popřípadě v grafickém programovacím jazyku Ladder.

Požadavky na jednoduché ovládání a optimální použití se stále zvyšují. Návrh SIMOTIONu vzal toto od počátku v úvahu. Prvotní výhoda je v celkovém řešení automatizační úlohy, ve které jsou interakce mezi rozdílnými komponenty optimalizovány.

Centrem tohoto přístupu je kombinace automatizačních a pohybových požadavků a použití grafických nástrojů (Motion Control Chart) k programování těchto úkolů. Jednotlivé požadavky pro výrobní stroje jsou formulovány v jednotném uživatelském rozhraní. Grafické programovací nástroje dovolují programovat logické a pohybové funkce. Také lze programovat logické povely vyhovující IEC 61131-3, pomocí jazyku Structured Text (ST) nebo graficky založenými jazyky Ladder Diagram/Function Block Diagram (LAD/FBD), které jsou též k dispozici.

Technologické požadavky jako nastavení polohy či synchronizace jsou umožněny jako funkce, které je snadné včlenit do programu, stejně jako logické povely.

Charakteristika provedení vývojového prostředí SCOUT:

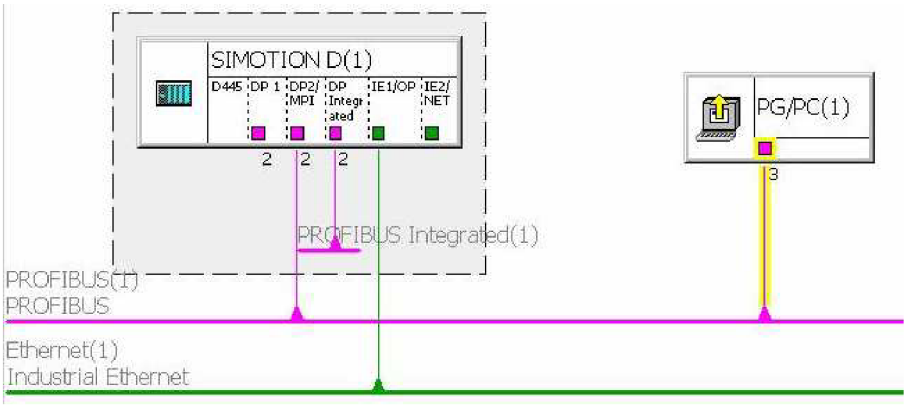
- integrovaný a intuitivní inženýrský systém
- místní údaje a správa programu
- na funkci orientovaná projektová struktura

- rychlý přístup do individuálních nástrojů jako je konfigurování, programování a uvádění do provozu.

Hardwarová konfigurace D4xx

Hardwarová konfigurace SIMOTION D4xx se provádí v PG/PC módu, kdy je v PC vložena karta pro komunikaci se SW, z balíčku od SIMATIC NET. Je možná i konfigurace rozhraním PROFIBUS nebo Ehernetovým rozhraním. Po připojení zařízení SIMOTION k PC se nakonfiguruje připojené spojení a žádaný protokol. Tímto je nastavena komunikace mezi PC a D4xx. Aby však PC mohlo komunikovat se SINAMICS Integrated je třeba vytvořit v nástroji NetPro další cestu z PG/PC (tzv. Routing). Po vytvoření tohoto spoje již SIMOTION a SINAMICS Integrated komunikuje mezi sebou uvnitř systému.

Následující obrázek ukazuje nastavení komunikace v PG/PC módu:



Obrázek č.4: Nastavení komunikace v NetPro

Konfigurace připojeného HW zařízení k SINAMICS přes sběrnici DRIVE-CLiQ je možná dvěma způsoby:

- Online konfigurace
- Offline konfigurace

Pro zařízení připojená např. přes rozhraní Ethernet nebo PROFIBUS je třeba nakonfigurovat ručně dle aktuální topologie.

Při volbě *Online* si zařízení v online módu vyhledá připojené komponenty na sběrnici DRIVE-CLiQ. Těmto komponentám vytvoří jejich topologii a přidělí defaultní nastavení parametrů. Defaultní parametry je samozřejmě nutno změnit dle aktuálního stavu a potřeby projektu (nastavení rámce zprávy, motor/enkodér data, atd.).

Při *Offline* konfiguraci se v offline módu nastaví topologie připojených komponent (tj. jaké zařízení je připojeno na jakém vstupu atd.) a dále stejně jako v Online konfiguraci se naparametrují jednotlivé komponenty.

Po těchto úkonech je nutné uložit projekt a překompilovat, uvést zařízení do online módu. Pokud je zařízení online je možné projekt stáhnout do zařízení na Compact Flash kartu.

1.1.2 Programovací jazyky

Programovací jazyky ve Scout poskytují všechny příkazy potřebné k implementaci určitých funkcí jednoduše a rychle. Běh programu SIMOTIONu také podporuje cyklické, sekvenční, časově řízené, událostně spouštěné programování [8].

Jestliže je preferován vysoce úrovňový jazyk pro úkol, je dostupný Structured Text (ST)-výkonný, vyhovuje normě IEC 61131-3.

Jestliže je preferován programovací jazyky graficky založené, Motion Control Chart (MCC) podporuje sekvenční programování formou vývojového diagramu.

Známé programovací metody Ladder Diagram (LAD) nebo Function Block Diagram (FBD) jsou též ve SCOUTu přístupné.

V jednom projektu je možná kombinace několika různých programovacích jazyků, které systém podporuje.

1.1.2.1 Structured Text (ST)

ST je vyšší, na Pascal orientovaný programovací jazyk. Je založen na normě IEC61131-3, která standardizuje programovací jazyky pro programovatelné automaty (PLC) [8]. Základní rámec povelů realizuje vše, co se týká řízení dat, početních funkcí, kontrolní struktury a přístupu I/O.

Požadavky mohou být rozděleny do libovolného počtu sekcí. Takovou sekcí může být program přičleněný k nosnému programu, jako blok s vlastní pamětí, nebo bez ní. V takovém případě funkční bloky a funkce do nosné části programu, pracují jako vyvolávané programy [8].

Charakteristika použití [8]:

- kontrola pohybu, PLC a technologických funkcí v jednom jazyce
- dobře strukturované programy s možností úprav
- kvalitní funkce pro úpravy, např. Syntax coloring
- snadné funkce pro ONLINE testování a odstraňování virů, např. znázornění obsahu kódové sekvence, vytříděné v editoru.

1.1.2.2 Grafický diagram programování MCC (Motion Control Chart)

Smyslem MCC je formulovat procesní pořadí za použití jednoduchých a logických výrazů.

Výsledkem je jeden nebo více diagramů, ukazujících chronologické sekvence akce. Protože je s výrobními stroji spojen problém množství pohybových os, MCC podporuje jednoduchý popis těchto sekvencí.

K řízení stroje jsou přístupné povely pro splnění podmínek a pro formulaci výpočtů, taktéž pro vypracování různých kontrolních struktur jako je volba (IF), určení případu (CASE) a smyčky (FOR,WHILE,UNTIL) [8]. Některé MCC programy jsou vytvořeny tak, aby popsaly rozdílné procesní situace. Např. může být vytvořen jeden program, který definuje stroji počáteční stav při zapnutí, druhý program pro běžný provozní stav a třetí MCC program pro havarijní stav.

Všechny povely jsou v nabídce nástrojů. Kliknutím na povel se vše přesouvá do diagramu.

Dvojité kliknutí v diagramu otevírá dialogový box pro zadání přídavných parametrů.

Použití:

- snadné díky diagramům
- seřazená povelová nabídka pro motion control, PLC a technolog. Funkce
- kontrolní struktury (IF, WHILE, CASE atd.)
- funkce Zoom pro LAD a FBD
- strukturování založené na povelových modulech, to je na kombinaci příkazových sekvencí pro vytvoření modulů. Kliknutí na modul vyvolává korespondující pokynovou sekvenci.

1.1.2.3 Programovací jazyk Ladder Diagram/Function Block Diagram

Jsou grafické programovací jazyky. Příkazová syntaxe odpovídá obvodovému schématu. Umožňují jednoduché sledování toku signálu mezi proudovými lištami skrze vstupy, výstupy a operace. LAD/FBD sestává z částí a komponentů, graficky spojených do sítě. Operace LAD/FBD se řídí pravidel Booleanské logiky.

LAD/FBD tvoří mocný povelový celek. To zahrnuje rozličné základní operace a způsob jejich uplatnění. Provedení funkcí a funkčních bloků umožní vytvořit strukturu programu pomocí LAD/FBD jako názornou. LAD/FBD programový balíček je nedílnou částí základního softwaru SIMOTION [8].

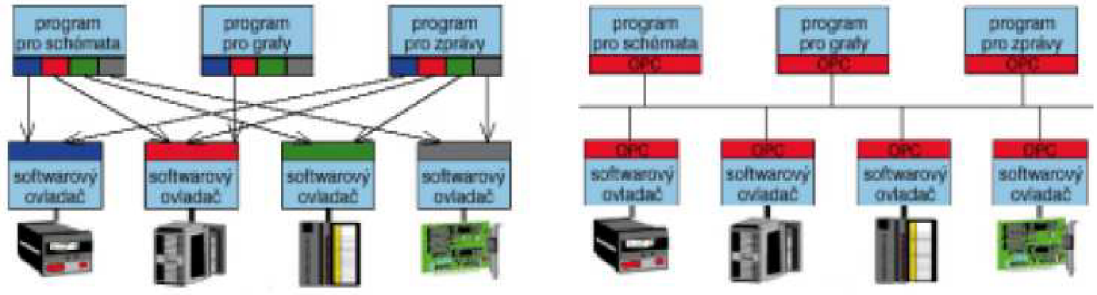
2. OPC

V současné době se pro přenos dat v průmyslových řídicích systémech stále častěji používá standard označený jako OPC. OPC (OLE for Process Control) je otevřený standard průmyslové komunikace. Je vytvořený ve spolupráci mnoha světových dodavatelů automatizačních prostředků, hardwaru a softwaru, v oblasti automatizace a společnosti Microsoft. Vytváří společné rozhraní pro vzájemnou komunikaci mezi různými zařízeními určenými pro monitorování a řízení technologických procesů. Jeho úkolem je zabránit závislosti daného monitorovacího nebo řídicího softwaru na výrobci hardwaru [9]. Díky tomu si koncový uživatel může vybrat libovolný software a hardware, podporující standard OPC. Nezajímá se tak o dostupnost komunikačních driverů pro jednotlivá zařízení. Standard OPC je založen na metodách OLE/COM/DCOM společnosti Microsoft.

2.1 HISTORIE VZNIKU OPC

Existuje mnoho klientských aplikačních programů, které byly vyvinuty s cílem získávat data z různých zdrojů. Přístup daného softwaru k datům je tradičně zajišťován prostřednictvím nezávisle vyvinutých ovladačů. Tato metoda vede k následujícím problémům (tzv. I/O driver problem – viz. obr. č.5) [9]:

- každá aplikace musí obsahovat ovladač pro konkrétní hardware, který využívá
- dochází k neshodám mezi ovladači od různých dodavatelů, když ne všichni dodavatelé (všechny ovladače) podporují všechny vlastnosti daného hardwaru
- změna vlastností hardwaru je příčinou nefunkčnosti některého ovladače
- vznikají konflikty v přístupu k hardwaru: dva různé programové balíky nemohou sdílet zařízení, pokud každý z nich neobsahuje nezávislý ovladač.



Obrázek č.5: I/O driver problem [9]

Mnozí výrobci hardwaru se snaží vyřešit tyto problémy vývojem dalších ovladačů, brání jim však odlišnosti v klientských protokolech [9]. Standard OPC vytváří mezi výrobci hardwaru a dodavateli softwaru dělicí čáru a poskytuje mechanismus umožňující získávat data z různých zdrojů a přenášet je do libovolného klientského programu nezávisle na dodavateli hardwaru [9]. Všichni dodavatelé pak mohou vyvinout výkonově optimalizovaný server pro komunikaci se zdrojem dat [9].

2.2 STANDARD OPC

Vypracováním standardu OPC, jeho udržováním, šířením a prezentací se zabývá mezinárodní dobrovolná organizace OPC Foundation. Tato organizace v současné době sdružuje několik stovek členů z řad nejvýznamnějších světových firem zabývajících se zejména monitorováním, vizualizací a dalšími aplikacemi z oblasti sledování a řízení technologických procesů. Členy OPC Foundation jsou také významní výrobci hardwaru.

Základem standardu OPC jsou technologie OLE a COM firmy Microsoft. Specifikace OPC je v podstatě souborem definic objektů OPC COM a jejich rozhraní, včetně popisu jejich implementace v konkrétní aplikaci OPC. Právě technologie COM umožňuje libovolné klientské aplikaci OPC (OPC klient) připojit se k různým OPC serverům různých výrobců, čímž je zaručena univerzálnost a nezávislost standardu na jediném výrobci [10]. K jednomu OPC serveru může být připojen libovolný počet OPC klientů.

Hlavním cílem standardizační iniciativy je umožnit klientským aplikacím konzistentní přístup k datům v technologických provozech. Předpokládané přínosy širokého přijetí standardu OPC jsou tyto [10]:

- výrobci hardwaru vystačí s jedním souborem programových komponent pro všechny zákazníky a jejich aplikace
- vývojáři softwaru nepotřebují psát stále nové ovladače kvůli změnám a novým vlastnostem hardwaru v jeho nových verzích
- zákazníkům se otvírá svoboda volby mezi dodavateli různých součástí a zařízení jak pro vývoj špičkových integrovaných technologických celků, tak i pro integraci sledování a řízení technologie na celozávodní i celopodnikové úrovni.
- rozhraní OPC se ve stále větší míře stává standardním rozhraním moderních programových produktů pro sledování a řízení technologických procesů, strojů a zařízení (SCADA/HMI), modulů programovatelných automatů a ostatních systémů (většinou již zahrnutých v ceně systému).

OPC servery jsou tedy konkrétní třídou objektů COM, vhodnou především pro oblast sledování a řízení technologických procesů. Představují zásadní pokrok díky robustnosti a skutečné modularitě objektové technologie [10].

Platformy pro použití OPC

Produkty odpovídající standardu OPC lze provozovat s operačními systémy, v nichž je implementována architektura COM a její síťová verze DCOM (s výjimkou specifikace OPC XML DA), tedy především s operačními systémy firmy Microsoft. A to Windows NT od verze 4.0, Windows 95/98/2000 a Windows CE. Do Windows 95/98 je nutné DCOM doinstalovat. V těchto systémech je tak možné přenášet data prostřednictvím OPC v rámci jednoho PC i v LAN.

Specifikace OPC

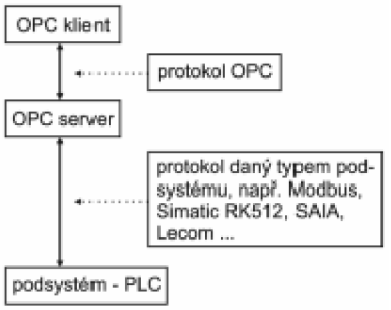
Standard OPC je vytvářen a udržován prostřednictvím tzv. specifikací OPC. Specifikace OPC je volně přístupná technická dokumentace, která stanovuje pravidla, jimiž se řídí chování a konfigurace standardního rozhraní OPC. Organizace OPC Foundation označuje každé vydání specifikace standardu OPC verzemi [10]. Jednotlivé verze se od sebe liší implementovanými rozhraními.

Dostupné jsou tyto OPC specifikace [9]:

- *OPC Overview* - obecný popis metody OPC, jejích výhod a způsobu použití.
- *OPC Common Definitions and Interfaces* - stanovuje případy použití většího počtu specifikací.
- *OPC Data Access (OPC DA)* - určuje přístup k datům v reálném čase, je nejčastěji používanou specifikací.
- *OPC Alarms and Events* - stanovuje poskytování informací o výskytech specifikovaných událostí a výstrah klientům OPC.
- *OPC Historical Data Access* - popisuje přístup k historickým datům uloženým v databázi.
- *OPC Batch* - podobně jako OPC DA, ale místo spojitých provozů je určena pro technologie s šaržovou výrobou. (potravinářství, farmacie apod.).
- *OPC Security* - určena k důkladnějšímu zabezpečení přístupu obsluhy při ovládání technologického zařízení z klientů OPC prostřednictvím serveru OPC s využitím zabezpečení systému Windows.
- *OPC XML DA* - vymezuje integrování OPC a XML do internetových aplikací.
- *OPC Data eXchange* - určena pro tzv. horizontální komunikaci mezi řídicími jednotkami s odlišnými komunikačními protokoly (např. EtherNet/IP, ProfiNet, High Speed Ethernet a Interbus) prostřednictvím sítě Ethernet.

2.3 ARCHITEKTURA OPC

Při výměně dat podle standardu OPC je využíváno schéma klient/server. Hovoří-li se o OPC, hovoří se primárně o dvou typech programů - o OPC Serveru a OPC Klientu. Jak OPC server, tak OPC klient jsou softwarové aplikace.

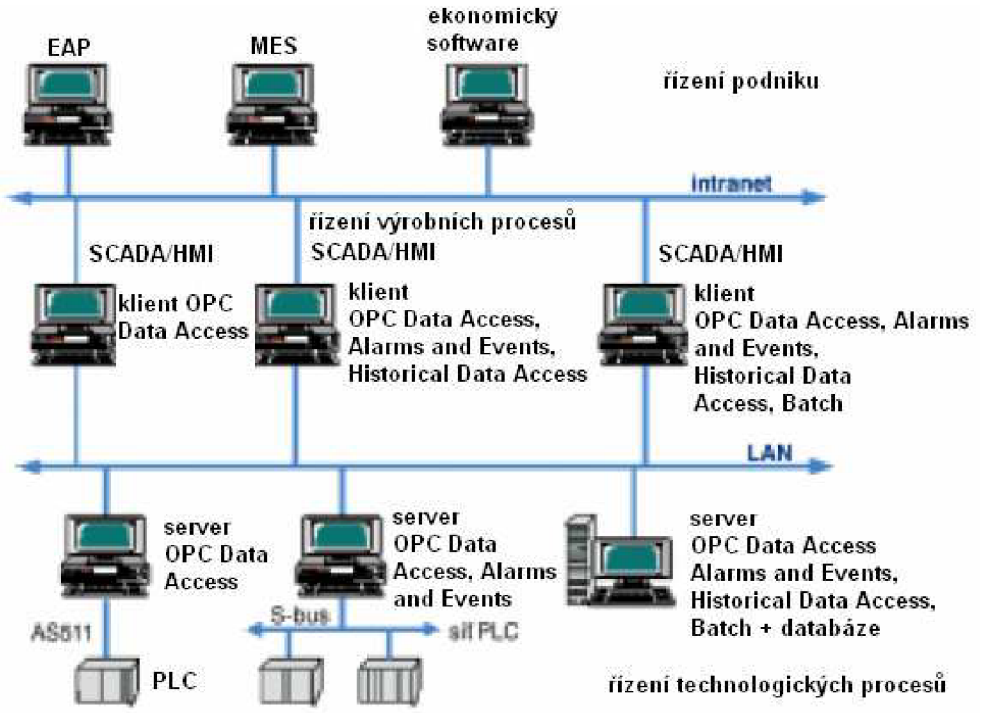


Obrázek č.6: Komunikace mezi OPC Klient – OPC Server – PLC [10]

OPC Server - je program, který komunikuje s připojeným zařízením jeho komunikačním protokolem (např. Modbus, MPI, PPI, atd.), získaná data převádí do formátu OPC a poskytuje je nadřazeným aplikacím ve formátu OPC. OPC Serverů existuje stovky druhů v závislosti na použitém zařízení/komunikačním protokolu, pro který jsou určeny.

OPC Klient - je program, který přijímá data z OPC Serveru ve formátu OPC a prezentuje tato data pro uživatele v podobě vizualizace, grafů, reportů apod. (zpravidla aplikace SCADA HMI, nebo-li programy pro vizualizaci, monitoring a řízení procesů průmyslové automatizace).

K jednomu serveru se může připojit několik klientů různých výrobců, stejně tak k jednomu klientu je možné připojit OPC servery různých výrobců. Komunikace mezi programy podle standardu OPC se používá k výměně dat zejména v průmyslových informačních systémech.



Obrázek č.7: Architektura informačního systému s OPC servery a klienty [9]

Typická architektura průmyslového informačního systému sestaveného za pomoci OPC lze rozdělit na tři úrovně řízení.

Nejnižší úroveň se označuje jako *řízení technologických operací* (field management). Obsahuje komunikační, popř. řídicí počítače připojené na jedné straně k řídicím jednotkám (PLC) a na straně druhé do místní podnikové sítě (LAN). Vedle těchto typicky serverových stanic je zde znázorněn komunikační a datový server - počítač, který může uchovávat technologická data v databázi (relační nebo real-time).

Střední úroveň se označuje jako *řízení výrobních procesů* (process management). Zde se nacházejí klientské počítače s vizualizačními a monitorovacími programy, prezentující technologické procesy operátorům v grafické a alfanumerické podobě (HMI).

Nejvyšší úroveň označená jako *řízení podniku* (business management). Zde se nacházejí nadřazené podnikové informační systémy s programy typu MES (Manufacturing Execution Systems), ERP (Enterprise Resources Planning) a různým

ekonomickým softwarem. Zde se OPC využívá k vytvoření společného komunikačního rozhraní pro přenos technologických dat mezi řídicími systémy, programy typu SCADA a databázemi [9]. Ty v reálném světě převážně pocházejí od různých výrobců a jejich propojování by bez společného komunikačního rozhraní bylo velmi složité a nesnadné [9].

2.4 OPC ALARM & EVENTS

V současnosti s úrovní automatizace, která je použita ve výrobě, se operátoři stýkají s vyšším a vyšším množstvím informací. Alarmující a událostní podsystémy byly použity pro označení oblastí procesu, které vyžadují bezprostřední pozornost. Oblasti zájmu zahrnují (ale nejsou na ně omezeny) bezpečnostní limity zařízení, detekci události, neobvyklé situace. Kromě operátorů mohou jiné klientské aplikace shromažďovat a zaznamenat alarm a informaci o události pro pozdější revidování nebo korelaci s dalšími historickými daty.

Alarmy a události v současnosti produkují přídavný proud informací, které musí být distribuovány uživatelům a softwarovým klientům, kteří se zajímají o tyto informace. V současné době většina alarm/událostních systémů používá svoje vlastní patentovaná rozhraní pro šíření a shromažďování dat. Neexistuje žádný způsob pro rozšíření existujících řešení alarmů s jinými způsoby v plug-n-play prostředí.

V souladu s přáním k integraci dat na všech úrovních podnikání, informace o alarmu může být považována za jiný typ dat než-li OPC Data. Tyto informace jsou cennou součástí informační architektury navrhnuté ve specifikaci OPC Data.

Specifikace OPC AE popisuje objekty a rozhraní, která jsou implementovaná do událostních OPC serverů, jenž poskytují mechanismus oznamování výskytu specifických podmínek (conditions) událostí a alarmů pro OPC klienty. Tato rozhraní také poskytují služby, které dovolují OPC klientům stanovit události a podmínky (condition) podporované událostním OPC serverem a získat jejich aktuální stav.

Entitami obvykle uvedených v procesu řídícího průmyslu jsou alarmy a události. Termíny alarm a události jsou často používány zaměnitelně a jejich významy nejsou odlišné.

V těchto specifikacích alarm je abnormální podmínka (condition), která vyžaduje zvláštní pozornost. Podmínka je pojmenování stavu událostního OPC serveru nebo jednoho z jeho obsaženého objektu, která je zajímavá pro jeho OPC klienty. Podmínky (condition) mohou být jedno-stavové, multi-stavové. Multi-stavová podmínka je ta, jejíž stav zahrnuje více rozsahů nebo sub-stavů, které jsou předmětem zájmu. Např. tag FIC101 může být spojen s podmínkami "LevelAlarm" nebo "DeviationAlarm". Kromě toho podmínka (condition) může být definovaná pro zahrnutí více dílčích sub-podmínek. Např. LevelAlarm podmínka může zahrnovat "HighAlarm", "HighHighAlarm", "LowAlarm" a "LowLowAlarm" sub-podmínku (condition). Jedno-stavová podmínka má jen jeden sub-stav, a tak má pouze jednu sub-podmínku s ním spojenou. Příkladem jedno-stavové podmínkou je podmínka poruchy HW, kdy HW zařízení je buď ve stavu selhal nebo neselhal.

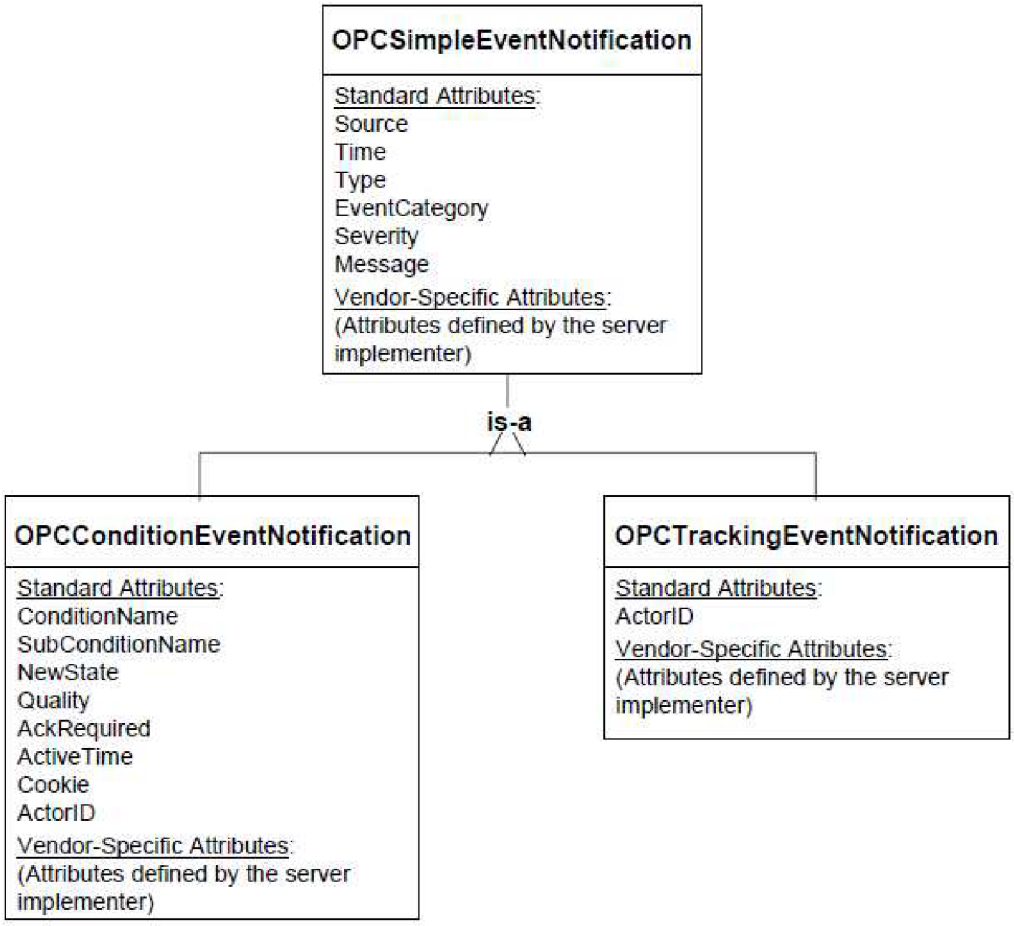
Na druhé straně událost je jev, který má důležitost pro OPC událostní server. Událost může a nemusí být spojená s podmínkou. Například změny úrovně podmínky alarmu a návrat do normálu jsou události, které jsou spojené s podmínkami. Nicméně, činnosti operátora, změny konfigurace systému a systémové chyby jsou příklady událostí, které nejsou spojeny s konkrétní podmínkou.

2.4.1 Typy událostí

Když se klientská aplikace přihlásí k serveru pro konkrétní nastavené události, server klientovi poskytuje oznámení o události. Existují tři typy událostí:

- Simple event
- Tracking event
- Condition event

Typy událostí, jak je definuje OPC AE specifikace jsou uvedeny na obrázku č.8. S podmínkou související události a tracking události sdílejí stejné atributy jako simple události, plus mají své vlastní jedinečné atributy, jak je uvedeno.



Obrázek č.8: Typy událostí v OPC AE [15]

2.4.1.1 Simple events

Simple události jsou typicky informační zprávy, které nevyžadují žádné zvláštní akce. Příklady simple událostí jsou systémová hlášení, jako je spuštění nebo vypnutí, přihlášení nebo odhlášení operátora, zprávy o selhání zařízení.

Simple události mají následující standardní atributy:

Source: Odkaz na objekt, který vygeneroval oznámení o události. Například by to může být zařízení pojmenované tagem (např. FIC101) v případě, že událost se týká tagu vstupujícího do úrovně alarm podmínky (condition-související události). Mohl by to být také název tagu pro tracking události, kdy operátor mění zadané hodnoty pro FIC101. Pro simple událost, jako je systémová chyba by hodnota Source mohla být "system".

Time: Čas, kdy k události došlo.

Type: Typ události, tj. zda souvisí s condition, tracking nebo simple událostí.

Event Category: Kategorie, do které tato událost patří. Event Category definuje seskupení událostí podporovaných OPC Event serverem. Spojení s událostí (event) lze filtrovat na základě Event Category.

Severity: Závažnost události. Může to být hodnota v rozmezí od 1 - 1000, kde 1 je nejnižší závažnost a 1000 je nejvyšší. Typicky závažnost 1 označuje událost, která má informační charakter, zatímco hodnota 1000 naznačuje událost katastrofického charakteru, což by mohlo mít za následek vážnou finanční ztrátu nebo ztrátu života. Dá se přepokládat, že málo implementovaných serverů bude podporovat 1000 různých úrovní závažnosti. Proto vývojáři serveru rozdělují úrovně závažnosti 1 až 1000 na části, takovým způsobem, aby klienti mohli předpokládat lineární rozdělení. Například klient, který si přeje pět úrovní závažnosti, lze rozdělit toto rozložení lineárně jak je znázorněno na obrázku č.9 níže.

Message: Text zprávy, který popisuje událost. Pro události s podmínkou související, obecně zahrnuje popis vlastnosti aktivní sub-podmínky.

2.4.1.2 Tracking Events

Tracking události jsou podobné simple událostem. Stejně jako simple události, tak i tracking události jsou často informační a nevyžadují žádné specifické akce. Příklady tracking událostí jsou operátorské akce procesu, jako jsou změny nastavení hodnot regulační smyčky nebo ladění parametrů, změny konfigurace systému, atd.

Tracking událost mají stejné standardní atributy jako simple události, kromě následujícího:

Actor ID: Identifikátor OPC klienta, který zahájil akci mající za následek tracking-související událost.

2.4.1.3 Condition Events

Podmínkové události jsou spojené s detekcí nějaké podmínky v systému, který obecně vyžaduje nějakou reakci nebo potvrzení ze strany uživatele nebo operátora. Proto tyto podmínky nějaké „stavy" informace, s nimi spojené. Dodatečné podmínky události jsou zasílány kdykoliv dojde ke změně informačního stavu spojeného s podmínkou.

Podmínkové události mají stejné standardní atributy jako single události, kromě následujících:

Condition Name: Pojmenování související s OPC Condition.

Sub-Condition Name: Pojmenování aktuálně aktivní sub-podmínky.

Change Mask: Indikuje klientovi, které vlastnosti se změnily v podmínce.

New State: Indikuje nový stav podmínky.

Condition Quality: Indikuje kvalitu datových položek, na nichž je založena tato podmínka.

Ack Required: Indikátor toho, zda je či není vyžadováno potvrzení.

Aktive Time: Čas přechodu do podmínky nebo sub-podmínky, která je spojená s oznámením této události.

Cookie: Server definuje cookie spojené s oznámením události. Tato hodnota se používá, když klient potvrzuje podmínku. Tato hodnota je matná pro klienta.

Actor ID: Identifikátor OPC klienta, který potvrdil podmínku, která je vedena jako Acknowledger ID vlastnost podmínky.

Client Severity	OPC Severity
HIGH	801 – 1000
MEDIUM HIGH	601 – 800
MEDIUM	401 – 600
MEDIUM LOW	201 – 400
LOW	1 – 200

Obrázek č.9: Příklad rozdělení severity

3. SIMOTION – VYTVOŘENÍ PROGRAMU

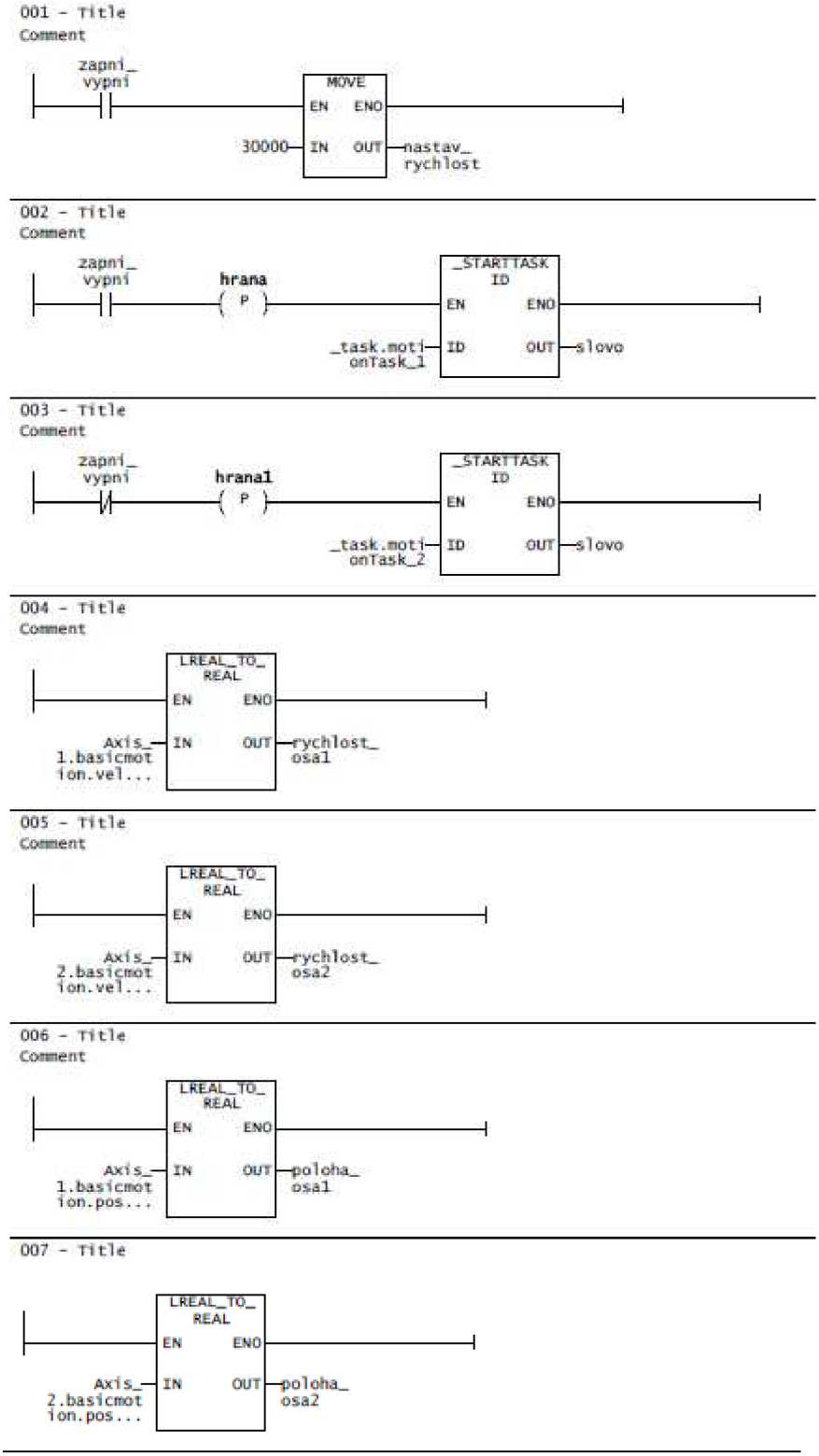
Po HW konfiguraci jak je popsáno v části 1.1.1 SIMOTION SCOUT a nastavení IP adresy Ethernetové sítě stanice (např. 10.3.3.50) v programu NetPro, se může začít vytvářet program v SIMOTION SCOUT.

V následující části bude uveden program, který po zapnutí vypínače *zapni_vypni* provede polohovou synchronizaci dvou vytvořených os (*axis_1* a *axis_2*). Tyto dvě virtuální osy mají vůči sobě nastaven převodový poměr 1:2, kde osa *axis_2* je nastavena jako master osa. Osa *axis_1* je synchronizovaná s touto master osou. Master osa *axis_2* se pohybuje rychlostí definovanou v proměnné *nastav_rychlost*, v jednotkách [°/s]. Při vypnutí vypínače *zapni_vypni* dojde k desynchronizaci os a jejich zastavení.

3.1 LADDER DIAGRAM

Tento Ladder diagram je cyklicky vyvoláván v *Backgroundtasku* a vykonává následující:

1. Network: Zapnutím vypínače *zapni_vypni* se nastaví proměnná *nastav_rychlost* na hodnotu jenž má uvedenou na vstupu.
2. Network: Zapnutím vypínače *zapni_vypni* se aktivuje blok *_STARTTASK ID*, který aktivuje danou sekvenci úloh zadaných v *motionTask_1*.
3. Network: Vypnutím vypínače *zapni_vypni* se aktivuje blok *_STARTTASK ID*, který aktivuje danou sekvenci úloh zadaných v *motionTask_2*.
4. Network: Slouží pro převod rychlosti osy *axis_1* z formátu *LongReal* do formátu *Real* do proměnné *rychlost_osa1*.
5. Network: Slouží pro převod rychlosti osy *axis_2* z formátu *LongReal* do formátu *Real* do proměnné *rychlost_osa2*.
6. Network: Slouží pro převod polohy osy *axis_1* z formátu *LongReal* do formátu *Real* do proměnné *poloha_osa1*.
7. Network: Slouží pro převod polohy osy *axis_1* z formátu *LongReal* do formátu *Real* do proměnné *poloha_osa1*.

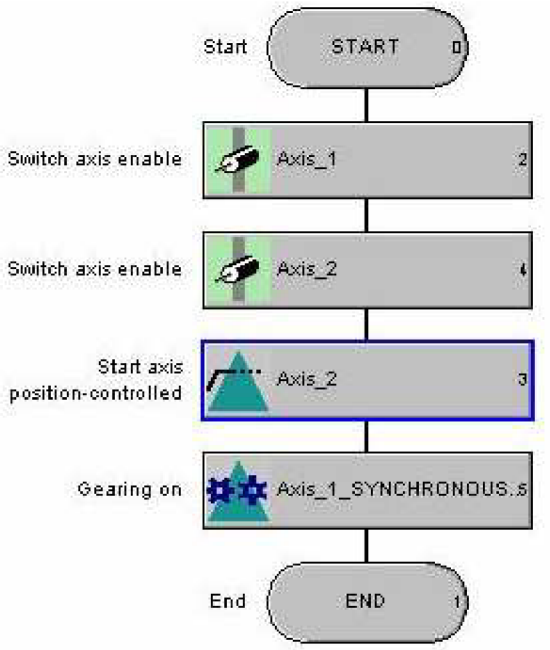


Obrázek č.10: Ladder diagram programu

Proměnné, které jsou použity ve vizualizaci WinCC jsou uloženy pro přehlednost v samostatné *Ladder_Unit* a přeneseny do *Watch table*, z níž jsou následně exportovány soubory, které využívá OPC server při komunikaci se SIMOTION D445.

3.2 MOTION TASK 1

V MotionTask_1, který je vyvoláván v Network 1 po zapnutí vypínače *zapni_vypni*, se spustí následující sekvence úloh zobrazená na obrázku č.11.



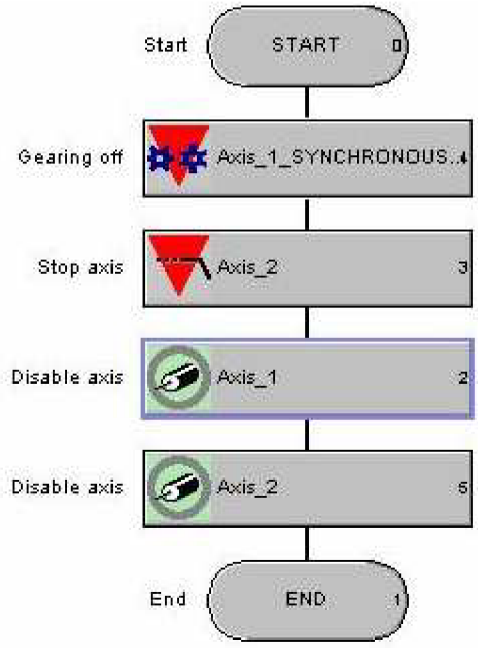
Obrázek č.11: Motion_Task_1

- Bloky *Switch axis enable* slouží pro uvolnění os *axis_1* a *axis_2*.
- Blok *Start axis position-controlled* je polohová regulace. Určuje jakým způsobem se má master osa *axis_2* pohybovat (rychlost, zrychlení, směr a další dynamické vlastnosti).
- Blok *Gearing on* aktivuje polohovou synchronizaci osy *axis_1* podle osy *axis_2*. V tomto bloku je nastavena *axis_2* jako leader (master) osa s převodovým poměrem 1:2.

Sekvencí těchto bloků je tedy aktivován pohyb obou os a jejich polohová synchronizace.

3.3 MOTION TASK 2

V *MotionTask_2*, který je vyvoláván v *Network 2* po vypnutí vypínače *zapni_vypni*, se spustí následující sekvence úloh zobrazená na obrázku č.12.



Obrázek č.12: Motion_Task_2

- Blok *Gearing off* ukončuje synchronizaci osy *axis_1* podle osy *axis_2*.
- Blok *Stop axis* zastavuje master osu *axis_2* a tím i osu *axis_1*.
- Bloky *Disable axis* blokují obě osy proti pohybu.

Sekvencí těchto bloků je tedy zastaven pohyb obou os a jejich polohová synchronizace.

Průběhy rychlostí obou os i průběhy jejich poloh jsou zobrazeny v grafech ve vizualizačním prostředí WinCC na obrázku č.16.

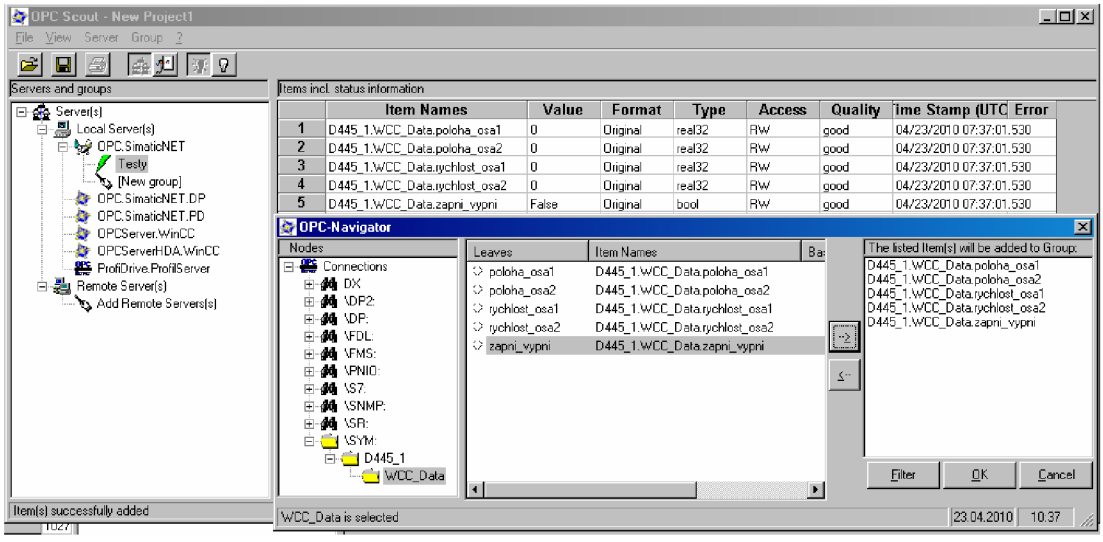
4. NASTAVENÍ OPC SERVERU

Po vytvoření programu pro systém SIMOTION je nutné z programovacího prostředí SIMOTION SCOUT následně exportovat potřebné soubory pro OPC server (soubory OPC_AE.xml, OPC_Data.idl, OPC_Data.sti). Tyto soubory slouží pro přenos dat (Data Access, Alarm & Events) mezi OPC serverem a SIMOTION D445. V souboru OPC_AE.xml jsou definovaná data pro přenos alarmů, v souboru OPC_Data.sti jsou definovány proměnné z programu, jejichž data chceme znát. Exportované soubory musí být uloženy na místě, které je určeno v nastavení OPC serveru. Defaultní nastavení bývá:

c:\ProgramFiles\SIEMENS\SIMATIC.NET\opc2\binS7\simotion\xml

Spoj mezi OPC serverem a SIMOTION D445 je uskutečněn pomocí Ethernetového rozhraní. Server se podle konfigurace spouští automaticky.

Po připojení SIMOTION D445 k PC, kde je nainstalován SIMATIC NET s OPC serverem, je třeba nastavit IP adresy této sítě pro komunikaci. Funkčnost spoje si lze ověřit pomocí utility s názvem OPC Scout. Jak vypadá takové ověření spoje je ukázáno na následujícím obrázku.



Obrázek č.13: Ověření spoje OPC server – SIMOTION D445

V obrázku je vidět proměnné, které OPC DA server přenáší s kvalitou *good*, což značí správnou funkci. Pokud je hodnota *Quality – bad* neprobíhá žádná komunikace mezi serverem a stanicí.

Jestliže je spoj OPC server – SIMOTION D445 funkční, může se přejít k nastavení programu WinCC, popsaného v následující části.

5. NASTAVENÍ WINCC

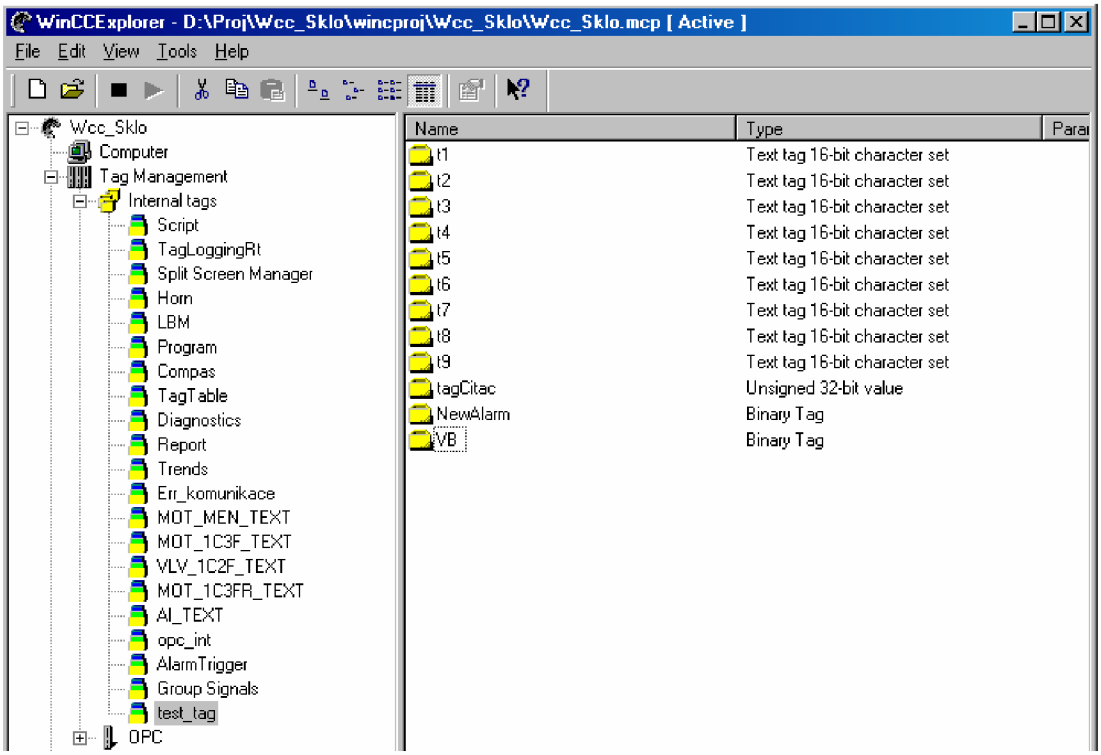
WinCC je modulový systém, který je používán pro vizualizaci procesu a konfiguraci grafickým uživatelským rozhraním. Na obrázku č.16 je vidět rozložení objektů použitých ve vizualizaci. V horní části obrazovky se nachází objekt *AlarmOneLine* – červené pole s bílím textem v němž jsou zobrazovány alarmy a další informace o běhu aplikace. Pod tímto objektem jsou umístěna textová pole v nichž jsou pro kontrolu zobrazovány jednotlivé části alarmů pomocí tagů. v těchto polích zadaných. Ve středu obrazovky je umístěno tlačítko s názvem *Start/Stop* pro ovládání běhu programu v systému SIMOTION za pomoci aplikace WinCC a OPC DA serveru. Ve spodní části obrazovky jsou umístěny grafy pro zobrazení dat o rychlosti a poloze jednotlivých os.

5.1 TAGY

WinCC dokáže z OPC serveru vyčítat data za pomoci tzv. tagů. Tag je pojmenování proměnné ve WinCC, která je vyčítaná z OPC serveru nebo použita interně ve WinCC. Při definici tagu se určí typ dat, adresa odkud se má vyčítat, update time, formát dat a jeho název. Tato definice se provádí ve *WinCC_Explorer* → *Tag_Management* → *OPC* → *OPC_Groups* → *wcc*, což je vytvořený channel mezi WinCC a OPC DA serverem. Takto vytvořeným tagem lze nyní sledovat aktuální hodnoty dané proměnné v některém z objektů (např. textové pole), jenž nám nabízí vizualizační prostředí. Pokud je zapotřebí vytvářet například tabulky nebo grafy, musí se tagy archivovat v modulu WinCC_Tag_Logging, aby bylo možné používat jejich dřívější hodnoty.

Úkolem této práce je zobrazovat v programu WinCC přicházející alarmy ze stanice SIMOTION pomocí OPC AE. Jelikož WinCC je klientem OPC serveru pouze pro přenos dat (OPC DA), bude zde ukázáno vytvoření tagů, pomocí nichž bude WinCC přijímat alarmová hlášení od OPC AE serveru přes vytvořené komunikační rozhraní v programu Visual Basic 6 (dále jen VB6). Tento program bude popsán dále v textu.

V následujícím obrázku č. 14 jsou uvedeny tagy, ve skupině interních tagů v nabídce WinCC Explorer: *Tag Management* → *Internal tags* → *test_tag*. Tyto tagy jsou používány pro komunikaci mezi *WinCC* a *VB6*.



Obrázek č.14: Tagy ve WinCC

Význam jednotlivých tagů z obrázku č. 14 , pro použití ve WinCC a Visual Basic 6 je následující:

Tag „*t1*“ je typu text tag 16-bit, udává údaje o datu a čase vzniklého alarmu.

Tag „*t2*“ je typu text tag 16-bit, udává jméno podmínky vzniklého alarmu.

Tag „*t3*“ je typu text tag 16-bit, udává zprávu o alarmu.

Tag „*t4*“ je typu text tag 16-bit, udává zprávu o zdroji alarmu.

Tag „*t5*“ je typu text tag 16-bit, udává zprávu, zda je podmínka aktivní.

Tag „*t6*“ je typu text tag 16-bit, udává zprávu, zda je podmínka potvrzena.

Tag „*t7*“ je typu text tag 16-bit, udává zprávu, zda je vyžadováno potvrzení.

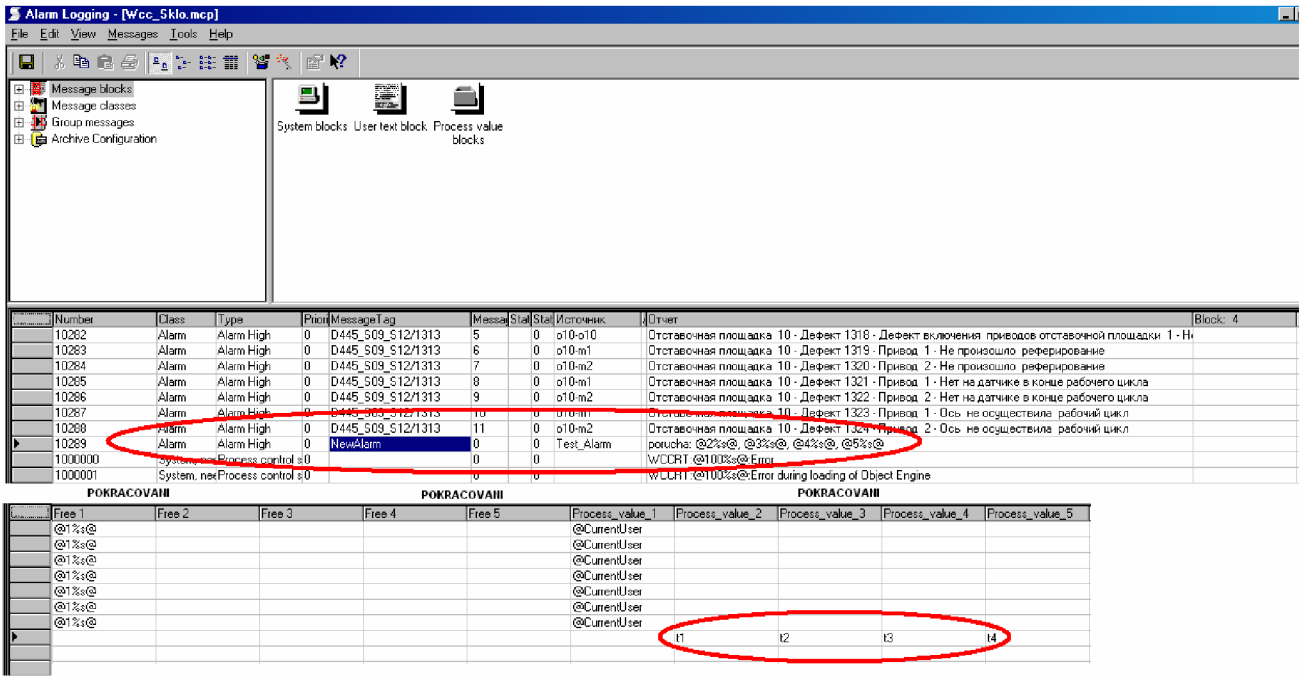
Tag „*t8*“ je typu text tag 16-bit, udává informaci o závažnosti alarmu.

Tag *„tagCitac“* je typu Unsigned 32-bit, udává počet alarmů, které mají být ještě přijaty do WinCC.
Tag *„NewAlarm“* je typu Boolean, slouží pro ukládání alarmů a zobrazení v objektu AlarmOneLine ve WinCC. Jeho inicializační hodnota je nastavena na „0“.
Tag *„VB“* je typu boolean, slouží pro ovládání výměny dat alarmů mezi WinCC a VB6. Jeho inicializační hodnota je nastavena na „0“.

Jednotlivé tagy, které jsou zde uvedeny jsou přiřazeny k objektům ve WinCC, které na jejich základě zobrazují daný alarm nebo konají definovanou funkci.

5.2 ALARM LOGGING

Alarm Logging je modul WinCC, který slouží k ukládání údajů do souboru. Tyto údaje jsou využívány k pozdějšímu vyčítání. Následující části bude ukázáno jak nastavit *Alarm Logging* pro ukládání alarmových hlášení přijímaných z programu *VB6*.



Obrázek č.15: Alarm Logging

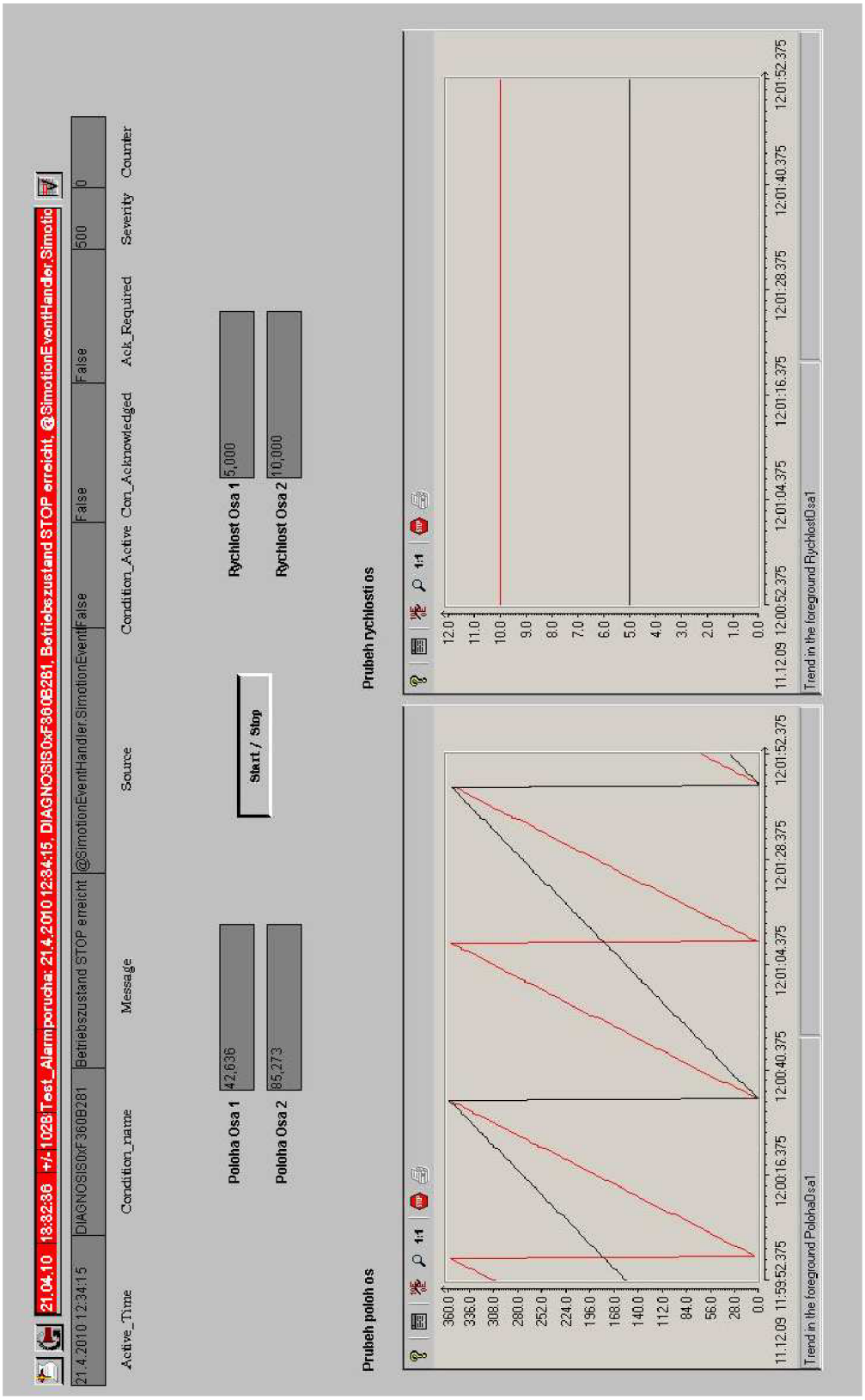
Nastavení modulu *Alarm Loggingu* se provádí ve *WinCC Explorer* → *Alarm Logging*. Po spuštění *Alarm Loggingu* se otevře konfigurační tabulka jako je na obrázku č. 15 . V tabulce se nastaví *Class* → *Alarm* a v této třídě se nastaví *Message Tag*. *Message Tag* určuje, kdy bude *Alarm Logging* reagovat. Alarm se zapíše v okamžiku změny *Message Tagu* (typu Boolean) z „log 0“ na „log 1“, tedy při náběžné hraně. V tomto konkrétním případě reaguje *Alarm Logging* na hodnotu tagu „*NewAlarm*“.

Dále se nastaví *Source* – název události jako „*Test_Alarm*“. V poli *Event* se nakonfiguruje, jak bude vypadat zobrazovaná zpráva alarmu (z čeho se bude skládat). Např. „Porucha: @2%s@, @3%s@, @4%s@, @5%s@ “. Tímto bude uložen do *Alarm Loggingu* text „Porucha: a následné řetězce znaků, které vyčítají procesní hodnoty tagů v nich nastavených“. Tedy sekvence znaků @2%s@ zajistí vyčtení hodnoty tagu „*t1*“, jenž je nastaven jako *Process_value_2*.

5.2.1 Alarm One Line

Je objekt WinCC určený pro zobrazování hlášení z *Alarm Loggingu*, se kterým je ve WinCC spojen.

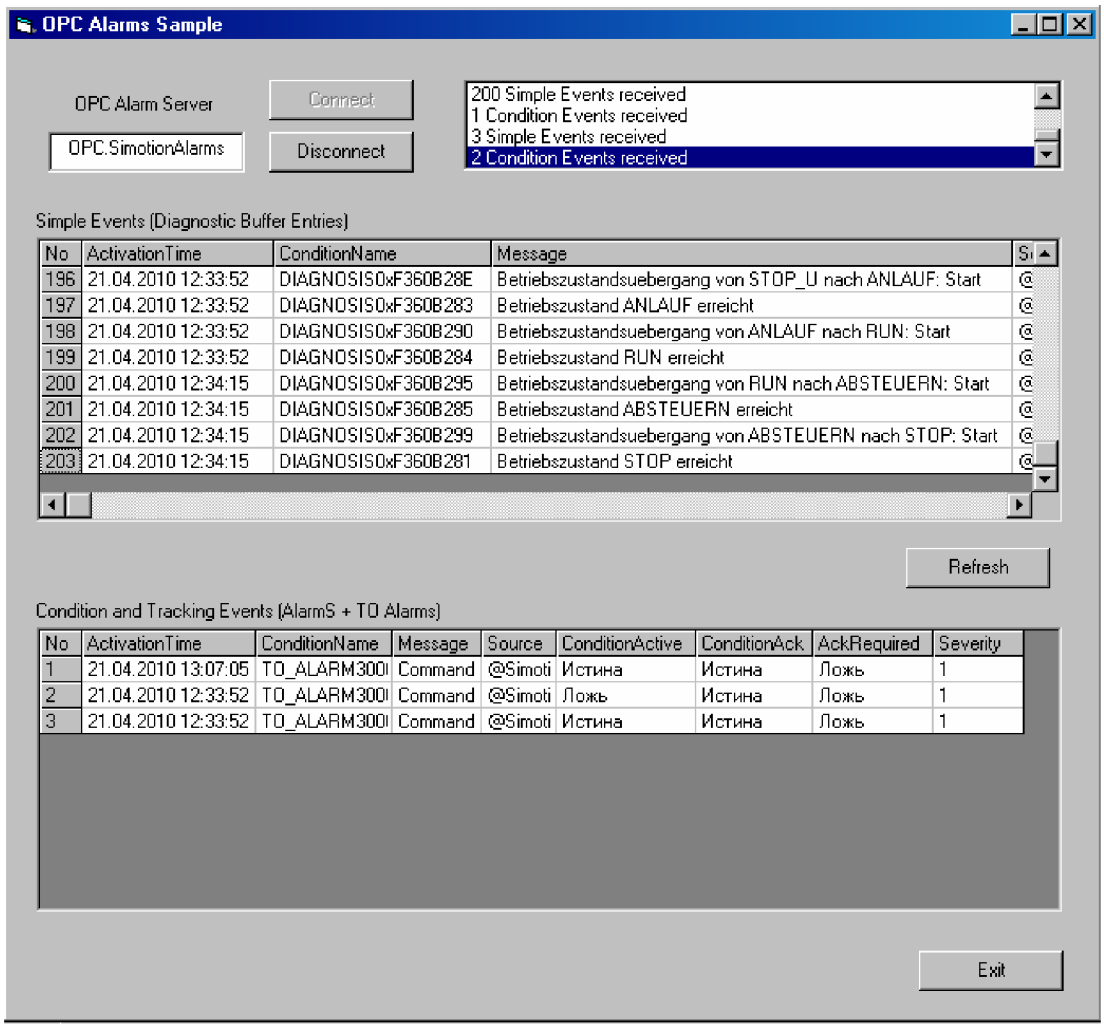
Lze v něm nakonfigurovat jaké části uložených alarmů má zobrazovat, zobrazení aktuálního času, zda byla poruch potvrzena/nepotvrzena, jazyk aktuálního národního prostředí, atd.



Obrázek č.16: Rozložení objektů ve WinCC

6. ZOBRAZENÍ ALARMŮ VE WINCC POMOCÍ APLIKACE VISUAL BASIC 6

V následující pasáži práce bude ukázán program pro vyčítání alarmů a událostí z OPC AE serveru a následnému přenosu těchto hlášení do vizualizačního prostředí WinCC. Program je vytvořen v aplikaci VB6, jenž může být klientem OPC AE serveru. Tento program z OPC AE serveru vyčte alarmová a událostní hlášení a vypíše je do tabulky. Současně při vypisování do tabulek alarmy a události uloží do globálních proměnných. Tyto proměnné budou následovně využity programem WinCC pro zobrazení a uložení do Alarm Loggingu.



Obrázek č.17: Grafické rozčlenění programu

6.1 POPIS PROGRAMU A JEHO PROCEDUR

Celý program vytvořený v aplikaci VB6 se skládá z jednoho Form objektu a jednoho Modulu.

Ve Form objektu jsou nastaveny procedury jednotlivých tlačítek v tomto Form objetu uložených. Dále jsou v tomto Form objektu dvě tabulky (Grid), jedno textové zobrazovací pole (ListBox) a jedno editační textové pole (TextBox) pro zadání názvu OPC serveru.

V Modulu jsou zadány public string, pro ukládání částí alarmových hlášení.

6.1.1 *Module1* - public řetězcová pole

Do polí v Module1 se ukládají části alarmů, které se využívají při komunikaci s vizualizačním prostředím WinCC. Jednotlivé části alarmů jsou popsány níže:

```
Public Alarms_Time(10000) As String      - čas vzniku alarmu
Public Alarms_Name(10000) As String      - název alarmu
Public Alarms_Mess(10000) As String      - hlášení alarmu
Public Alarms_Sour(10000) As String      - zdroj alarmu
Public Alarms_ConAct(10000) As String    - trvání alarmu
Public Alarms_ConAck(10000) As String    - potvrzení alarmu
Public Alarms_AckReq(10000) As String    - žádost potvrzení alarmu
Public Alarms_Seve(10000) As String      - závažnost alarmu
```

6.1.2 Deklarace objektů rozhraní a proměnných ve Form objektu:

```
Option Explicit
Option Base 1          'pole v OPC vždy začínají indexem 1
Public WithEvents MyAServerObj As OPCEventServer
Public WithEvents MyEventSubscriptionObj As OPCEventSubscription
Public Citac As Integer        'čitac počtu alarmových hlášení
Public s As String     'řetězec pro nadpisy sloupců objektů Grid1 a Grid2
```

6.1.3 Definice načtení formuláře

Procedura *Form_Load()* slouží k počátečnímu nastavení tabulek, jako jsou jejich rozměry a popis jednotlivých sloupců pomocí řetězce *s*. Je zde definováno, že po načtení Form budou aktivní pouze tlačítka *Connect* a *Exit*. Ostatní jsou neaktivní.

```
Private Sub Form_Load()
        With MSFlexGrid1
           .ColWidth(0) = 200
           .ColWidth(1) = 1700
           .ColWidth(2) = 2000
           .ColWidth(3) = 4000
           .ColWidth(4) = 5000
        End With

        With MSFlexGrid2
           .ColWidth(0) = 200
           .ColWidth(1) = 1700
           .ColWidth(2) = 2000
           .ColWidth(3) = 4000
           .ColWidth(4) = 5000
        End With
s = "< No |< ActivationTime |< ConditionName |< Message |< Source |< ConditionActive |< ConditionAck|< AckRequired |< Severity  "

        MSFlexGrid1.FormatString = s
        MSFlexGrid2.FormatString = s
        CmdDisconnect.Enabled = False
        CmdRefresh.Enabled = False
        CmdConnect.Enabled = True
        CmdExit.Enabled = True
End Sub
```

6.1.4 Připojení k OPC serveru

Procedura *CmdConnect_Click()* je určena pro připojení po stisknutí tlačítka *Connect* k OPC serveru, jehož název je zadán v TextBox (OPC.SimotinAlarms). Během této procedury se vypisuje do ListBox hlášení o jednotlivých stavech připojování k serveru. Při správně zadaném názvu OPC serveru, se k němu pokusí připojit pomocí nastaveného objektu *MyAServerObj*. Jestliže je server aktivní připojí se k němu. Po připojení se k serveru se nastaví objekt *MyEventSubscriptionObj* reprezentující hlášení alarmů. Znepřístupní se tlačítko *Connect* a povolí tlačítka *Refresh*, *Disconnect*. Při chybně zadaném názvu OPC serveru nebo neaktivním OPC serveru vygeneruje chybové hlášení zadané v ErrorHandler a opětovně zpřístupní tlačítko *Connect*.

```
Private Sub CmdConnect_Click()
        Dim OutText As String
        On Error GoTo ErrorHandler

        CmdConnect.Enabled = False
        OutText = "Connecting to OPC Alarm Server"
        Report.AddItem (OutText)
        Set MyAServerObj = New OPCEventServer
        MyAServerObj.Connect (EdtOPCServer.Text)
Report.AddItem "Connection to " & MyAServerObj.ServerName & " established"
Report.AddItem "State of " & MyAServerObj.ServerName & " is " &
MyAServerObj.ServerState
        Report.ListIndex = Report.ListCount - 1
        OutText = "Adding a Subscription to OPC-AlarmServer"
        Report.AddItem OutText
Set MyEventSubscriptionObj =
MyAServerObj.OPCEventSubscriptions.Add("MyAlarmSubscription")
        OutText = "Activating Subscription"
        Report.AddItem OutText
```

```
        MyEventSubscriptionObj.IsActive = True
        Report.AddItem "Subscription is active"
        Report.ListIndex = Report.ListCount - 1
        CmdDisconnect.Enabled = True
        CmdConnect.Enabled = False
        CmdRefresh.Enabled = True

    Exit Sub
    ErrorHandler:
        MsgBox Err.Description + Chr(13) + OutText, vbCritical, "ERROR"
        CmdConnect.Enabled = True
End Sub
```

6.1.5 Přijmutí hlášení o alarmu

Procedura *MyEventSubscriptionObj_SimpleEvent()* zajišťuje příjem alarmového hlášení *Simple Event* z OPC serveru při jejich vzniku. V proceduře je definovaná proměnná *myEvent* jako OPCEvent, přes kterou jsou vzniklé alarmy v proceduře dále zpracovávány po jejich jednotlivých částech (ActiveTime, ConditionName, atd.) a cyklicky ukládány do *Grid1*. Procedura vypisuje do ListBox počet přijatých alarmů. V prvním cyklu procedury jsou alarmy současně s výpisem do tabulky ukládány prostřednictvím procedury *SaveAlarms ()*, jenž je popsána dále v textu (kapitola 6.1.9). Druhý cyklus v proceduře je pouze prostředkem pro přečíslování indexů v tabulce (*Grid1*).

Další dvě procedury *MyEventSubscriptionObj_ConditionEvent* a *MyEventSubscriptionObj_TrackingEvent* zajištují příjem hlášení Condition Event a hlášení Tracking Event. Mají obdobnou strukturu kódu jako procedura *MyEventSubscriptionObj_SimpleEvent()*. Tyto procedury jsou uvedeny na přiloženém CD-ROM. Procedura pro Tracking Event má jako vstupní parametry (ByVal myEvents As OPCSiemensAlarmEventAutomation.OPCEvents). Hlášení těchto procedur jsou ukládány do tabulky *(Grid2)*, viz přiložené CD.

```
Private Sub MyEventSubscriptionObj_SimpleEvent(ByVal myEvents As
OPCEvents)
        Dim myEvent As OPCEvent
        Dim index As Long
        Report.AddItem myEvents.Count & " Simple Events received"
        Report.ListIndex = Report.ListCount – 1

        For index = 1 To myEvents.Count
            Set myEvent = myEvents.Item(index)
            With myEvent

MSFlexGrid1.AddItem index & vbTab & .ActiveTime & vbTab & _
.ConditionName & vbTab & .Message & vbTab & .Source & vbTab & _
.ConditionActive & vbTab & .ConditionAcknowledged & vbTab & _
.AckRequired & vbTab & .Severity, 1

            SaveAlarms myEvents, index
            End With
        Next

        'precislovani indexu v tabulce
        For index = 1 To MSFlexGrid1.Rows - 1
           MSFlexGrid1.Row = index
           MSFlexGrid1.Col = 0
           MSFlexGrid1.Text = index
        Next
End Sub
```

6.1.6 Refresh formuláře

Procedura *CmdRefresh_Click()* provede po stisknutí tlačítka *Refresh* zápis do ListBox, že provádí tuto operaci zablokuje tlačítko pro další kliknutí, smaže obsah obou tabulek ve Form. Nakonec provede refresh objektu *MyEventSubscriptionObj*. Při vyskytnutí chyby vypíše hlášení (v ErrorHandler) o chybě. Po dokončení procedury *MyEventSubscriptionObj_RefreshComplete()* je tlačítko *Refresh* opět aktivní a provede se zápis do ListBox o ukončení akce refresh.

```
Private Sub CmdRefresh_Click()
    Dim OutText As String

     On Error GoTo ErrorHandler

     OutText = "Refresh activation"
     Report.AddItem OutText
     Report.ListIndex = Report.ListCount - 1
     CmdRefresh.Enabled = False
     MSFlexGrid1.Clear
     MSFlexGrid2.Clear
     MSFlexGrid1.FormatString = s
     MSFlexGrid2.FormatString = s
     MSFlexGrid1.Rows = 1
     MSFlexGrid2.Rows = 1
     MyEventSubscriptionObj.Refresh

    Exit Sub
     ErrorHandler:
      MsgBox Err.Description + Chr(13) + OutText, vbCritical, "ERROR"
End Sub
```

```
Private Sub MyEventSubscriptionObj_RefreshComplete()
        CmdRefresh.Enabled = True
        Report.AddItem "Refresh completed"
        Report.ListIndex = Report.ListCount – 1
End Sub
```

6.1.7 Odpojení OPC serveru

Procedura *CmdDisconnect_Click()* je určena pro odpojení aplikace od aktuálně připojeného OPC serveru po stisknutí tlačítka *Disconnect*. Během této procedury se vypisuje do ListBox hlášení o deaktivaci OPC serveru, aktivní hlášení o alarmech se deaktivuje a nastaví se jako Nothing. Po deaktivaci alarmů se odpojí OPC server a nastaví se jako Nothing. Mohou se opětovně zablokovat tlačítka *Disconnect*, *Refresh* a povolit tlačítko *Connect*. Tak je ukončeno spojení s OPC. Při chybě během deaktivace OPC serveru se vygeneruje chybové hlášení zadané v ErrorHandler.

```
Private Sub CmdDisconnect_Click()
        Dim OutText As String
        On Error GoTo ErrorHandler

          OutText = "Deactivating Subscription"
          Report.AddItem OutText
          Report.ListIndex = Report.ListCount - 1
          MyEventSubscriptionObj.IsActive = False
          Set MyEventSubscriptionObj = Nothing
          OutText = "Disconnecting from Server"
          Report.AddItem OutText
          Report.ListIndex = Report.ListCount - 1
          MyAServerObj.Disconnect
          Set MyAServerObj = Nothing
          OutText = "Disconnected from Server"
```

```
        Report.AddItem OutText
        Report.ListIndex = Report.ListCount - 1
        CmdDisconnect.Enabled = False
        CmdRefresh.Enabled = False
        CmdConnect.Enabled = True

        Exit Sub
        ErrorHandler:
        MsgBox Err.Description + Chr(13) + OutText, vbCritical, "ERROR"

End Sub
```

6.1.8 Vypnutí aplikace

Procedura *Form_QueryUnload()* slouží pro správné odpojení serveru při zavření Form. Pokud je připojen OPC server, procedura ho odpojí příkazem kliknutí na tlačítko *Disconnect*.

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
        If Not (MyAServerObj Is Nothing) Then CmdDisconnect_Click
End Sub
```

Procedura *CmdExit_Click()* ukončuje program kliknutím na tlačítko *Exit*.

```
Private Sub CmdExit_Click()
        End
End Sub
```

6.1.9 Uložení hlášení o alarmech

Procedura *SaveAlarms()* při každém uloženém alarmu inkrementuje public proměnnou *Citac* o jedničku (ukládá počet alarmových hlášení). Zapisuje právě přijímané části alarmů od OPC serveru z procedur:

MyEventSubscriptionObj_SimpleEvent(),

MyEventSubscriptionObj_ConditionEvent(),

MyEventSubscriptionObj_TrackingEvent().

do public řetězcových polí v Module1.

```
Private Sub SaveAlarms(ByVal mojeEvents As OPCEvents, index As Long)
    Dim myEvent As OPCEvent
            Citac = Citac + 1

            Set myEvent = mojeEvents.Item(index)
            With myEvent        'zapis alarmu do poli
            Alarms_Time(Citac) = .ActiveTime
            Alarms_Name(Citac) = .ConditionName
            Alarms_Mess(Citac) = .Message
            Alarms_Sour(Citac) = .Source
            Alarms_ConAct(Citac) = .ConditionActive
            Alarms_ConAck(Citac) = .ConditionAcknowledged
            Alarms_AckReq(Citac) = .AckRequired
            Alarms_Seve(Citac) = .Severity
            End With
End Sub
```

6.2 PŘENOS ALARMOVÝCH HLÁŠENÍ DO WINCC

Přenos alarmových hlášení z programu VB6 do vizualizačního prostředí WinCC je uskutečněn dvěmi cyklicky běžícími procedurami. Jedna procedura pojmenovaná *Timer1_Timer()*, jenž běží ve VB6 a druhá procedura běžící v prostředí WinCC. V obou vyvolávaných procedurách je společná proměnná „VB“ (typu Boolean), jejíž pomocí je určeno, zda může pracovat s daty uložených alarmů v Module1 aplikace VB6 nebo WinCC. Jde tedy o přepínač, který udává, jaká z aplikací pracuje s řetězcovými poli alarmů, aby nedošlo k současnému přístupu k datům.

Procedura Timer1_Timer() cyklicky vyvolává zdrojový kód v ní zapsaný každých 500 ms. Přenos dat a hodnot tagů se uskutečňuje prostřednictvím vytvořeného objektu mcp, který definuje WinCC-Runtime. Funkce *mcp.SetValue()* je určena pro nastavení hodnoty tagu, v ní zadaného. Funkce *mcp.GetValue()* je naopak určena pro vyčtení hodnoty tagu, v ní zadaného. Procedura zde uvedená pracuje následovně. Inicializační hodnota tagu *„VB“* je ve WinCC nastavena na hodnotu „log 0“. Jestliže je splněna podmínka *„VB“* = 0 a současně splněna podmínka Citac > 0 zapíše se do tagů *„t1“* až *„t8“* hlášení prvního uloženého alarmu. Následuje cyklus, ve kterém se zmenší řetězcová pole alarmů a posunou se o jednu pozici v poli dopředu. Po provední těchto úkonů se sníží počítadlo alarmů *Citac* o jedničku a hodnota tagu *„VB“* se nastaví na „log 1“.

```
Private Sub Timer1_Timer()
    Dim mcp   As Object
    Dim bRet  As Variant
    Dim i As Integer

'objekt pro komunikaci mezi VB a WinCC
    Set mcp = CreateObject("WinCC-Runtime-Project")
    bRet = mcp.SetValue("tagCitac", Citac)
```

```
If (mcp.GetValue("VB") = 0) Then
    If Citac > 0 Then
    'vypis alarmu do tagu WinCC
    bRet = mcp.SetValue("t1", Alarms_Time(1))
    bRet = mcp.SetValue("t2", Alarms_Name(1))
    bRet = mcp.SetValue("t3", Alarms_Mess(1))
    bRet = mcp.SetValue("t4", Alarms_Sour(1))
    bRet = mcp.SetValue("t5", Alarms_ConAct(1))
    bRet = mcp.SetValue("t6", Alarms_ConAck(1))
    bRet = mcp.SetValue("t7", Alarms_AckReq(1))
    bRet = mcp.SetValue("t8", Alarms_Seve(1))

    'cyklus zmenseni retezce alarmu vlevo o -1 (posun)
    For i = 1 To Citac - 1
    Alarms_Time(i) = Alarms_Time(i + 1)
    Alarms_Name(i) = Alarms_Name(i + 1)
    Alarms_Mess(i) = Alarms_Mess(i + 1)
    Alarms_Sour(i) = Alarms_Sour(i + 1)
    Alarms_ConAct(i) = Alarms_ConAct(i + 1)
    Alarms_ConAck(i) = Alarms_ConAck(i + 1)
    Alarms_AckReq(i) = Alarms_AckReq(i + 1)
    Alarms_Seve(i) = Alarms_Seve(i + 1)
    Next i

    Citac = Citac - 1              'snizeni citace
    bRet = mcp.SetValue("VB", 1)

  End If
 End If
End Sub
```

Druhá cyklicky vyvolávaná procedura běží na pozadí WinCC, ve *WinCC Explorer → Global Skript C → Global Actions*. Je vyvolávána každých 250 ms a je zapsaná v jazyce C. Jsou v ní definovány tagy *„VB“* (pro řízení přístupu k datům ve VB6) a tag *„NewAlarm“* (pro ukládání a zobrazování alarmů v Alarm Loggingu). Jestliže je splněna podmínka, kdy je tag *„VB“* v „log 1“, nastaví se tag *„NewAlarm“* na „log 1“, čímž je vytvořena skoková změna, na kterou Alarm Logging reaguje zápisem nastavených částí alarmových hlášení (viz. kapitoly 5.1, 5.2). Aby toto fungovalo i pro další alarmová hlášení je tag *„NewAlarm“* opětovně nastaven na „log 0“. Jestliže se do podmínky nevstupuje je tag *„VB“* nulován a značí tak, že může pracovat aplikace VB6.

```
#include "apdefap.h"
int gscAction( void )
{
#define NewAlarm "NewAlarm"
#define VB "VB"
        if (GetTagBitWait(VB) == 1)
                        {
                                SetTagBitWait(NewAlarm,1);
                                SetTagBitWait(NewAlarm,0);
                        }
        SetTagBitWait(VB,0);
return 0;
}
```

7. ZÁVĚR

Diplomová práce se věnuje problému zobrazování alarmových hlášení ze SIMOTION systému firmy Siemens do vizualizačního prostředí WinCC. Systém SIMOTION generuje při výskytu abnormálních stavů systému hlášení o výskytu tohoto stavu. Tato hlášení dokáže OPC server definovaný sdružením OPC Foundation vyvolat a uchovat je pro další poskytování. Zde se objevil problém s poskytnutím těchto dat pro vizualizaci WinCC. Tato totiž není klientem OPC serveru a bylo nutno vytvořit funkční rozhraní mezi OPC serverem a vizualizací.

Bylo tedy nutné se seznámit s více systémy (SIMOTION a WinCC) a specifikací OPC. První část práce je věnována právě systému SIMOTION a podrobněji je popsáno zařízení SIMOTION D445. Na D445 byl vytvořen jednoduchý program, jenž sloužil pro přenos dat pomocí OPC DA (Data Access) a současně sloužil pro testovaní generovaných alarmů a událostí pomocí OPC AE (Alarm & Events). Specifikacím OPC byla věnována druhá část práce.

V dalších částech jsou uvedena nastavení jednotlivých programů a prostředí, což bylo nezbytné pro správnou funkci vytvořeného rozhraní mezi OPC AE a WinCC. Po vytvoření programu v SIMOTION SCOUT, jenž je uveden v třetí části, se práce krátce věnuje nastavení OPC serveru a testování přenosu dat mezi OPC a SIMOTION. Tato fáze sloužila pro ověření komunikace mezi OPC DA a WinCC, pro které je klientem.

V šesté části práce jsou uvedeny podstatné části vytvořeného komunikačního rozhraní mezi OPC AE serverem a WinCC. Je zde uvedeno, jak se aplikace připojuje k OPC AE serveru pomocí definovaných objektů, jak přijímá hlášení od serveru, jak se odpojuje a jakým způsobem předává dané informace vizualizačnímu prostředí. Server OPC, vizualizace WinCC i vytvořené rozhraní byly umístěny na stejném PC, na němž bylo rozhraní testováno. Server OPC byl nastaven tak, aby se spustil při startu operačního systému PC.

Při testování rozhraní byly porovnávány údaje o alarmech generované SIMOTION systémem s údaji ve WinCC, které spolu korespondovaly. Tímto lze konstatovat, že rozhraní funguje správně. Při selhání WinCC a nevyčtení všech

hlášení z rozhraní není problém se ztrátou hlášení o alarmech. Opětovným spuštěním WinCC se nevyčtená hlášení začnou znovu ukládat do Alarm Logginu. Problém s nevyčtenými hlášeními nastane v okamžiku, kdy selže aplikace s rozhraním. Tím se ztratí hlášení nevyčtených alarmů. Nevýhodou navrženého rozhraní je řešení přenosu mezi Visual Basic 6 a WinCC pomocí cyklických vyvolávání procedur, což se zdá být pomalé, ale dostačující řešení.

8. SEZNAM POUŽITÝCH ZKRATEK:

COM – Common Object Model

DCOM – Distributed COM

DRIVE CLiQ - DRIVE Component Link with IQ

FBD – Function Block Diagram

HMI – Human Machine Interface

HW – Hardware

I/O – Input/Output

LAD – Ladder Diagram

MCC – Motion Control Chart

MC – Motion Control

OPC – OLE for Process Control

OLE – Object Linking and Embedding

PC – Personal Computer

PLC – Programmable Logic Controller

SCADA – Supervisory Control and Data Acquisition

ST – Structured Text

SW – Software

TIA – Totally Integrated Automation

VB6 – Visual Basic 6

Přílohy viz.: přiložené CD

9. SEZNAM POUŽITÝCH ZDROJŮ:

[1] Boček, P.: Automa: Řídicí systém Simotion [online]. 2008, Číslo 07. URL: <http://www.odbornecasopisy.cz/index.php?id_document=37543>

[2] Dočkal, K.: Automa: Simotion [online], 2004, číslo 03. URL: <http://www.odbornecasopisy.cz/index.php?id_document=32223>

[3] Siemens: Řídicí systém SIMOTION, [online], 2009. URL: <http://www1.siemens.cz/ad/current/index.php?ctxnh=d86d92be87&ctxp=home&PHPSESSID=9efedf2d216311781f43751de5d1e264#>

[4] Siemens: Řídicí systém SIMOTION D [online], 2009. URL: <http://www1.siemens.cz/ad/current/index.php?ctxnh=2a0e9cc007&ctxp=home&PHPSESSID=9efedf2d216311781f43751de5d1e264>

[5] SIEMENS: Simotion Přehled, 16384_Simotion_$_prehled [online], 23.6.2005. URL: <http://www.siemens.cz/siemjetstorage/files/16384_Simotion_$_prehled.pdf>

[6] SIEMENS: SIMOTION D4xx – Commissioning and Installation Manual. D4xx_Commissioning.pdf [CD-ROM], 03.2006, 171 s.

[7] SIEMENS: SIMOTION D4x5 – Operating Manual. D4x5_operating.pdf [CD-ROM], 03.2006, 82 s.

[8] SIEMENS: SIMOTION SCOUT – Configuration Manual, Simotion SCOUT manual.pdf [CD-ROM], 01.2004 Edition, 562 s.

[9] Stianko, M.: Automa: OPC v průmyslové komunikaci, [online], 2004, číslo 06. URL: <http://www.odbornecasopisy.cz/index.php?id_document=32378>

[10] Stianko, M.: Automa: OPC- nový průmyslový standard pro informační technologie, [online], 2000, číslo 06. URL: <http://www.odbornecasopisy.cz/index.php?id_document=27739>

[11] FOXON: Co je OPC?, [online], URL: <http://www.foxon.cz/index.php?main_page=faq_info&fcPath=28&faqs_id=91>

[12] Morkes, D.: Visual Basic 6 pro střední školy, Praha 2000, vydání první, 167 s.

[13] SIEMENS: WinCC V6.2 Getting started.pdf, [CD-ROM], 11/2006, 330 s.

[14] OPC Foundation.: Alarms and Events Custom Interface, [online], October 2, 2002, version 1.10, 135 s., URL: <http://www.opcfoundation.org/>

[15] DeltaV.: OPC Alarms and Events Overview, [online], March 2007, 13 s., URL: <http://www2.emersonprocess.com/siteadmincenter/PM%20DeltaV%20Documents/Whitepapers/WP_OPC_Alarms_Events.pdf>