
Stručný popis řídicího systému REX

Abstrakt

Příspěvek popisuje strukturu a možnosti nového průmyslového řídicího systému REX. Systém byl navržen s důrazem na možnost úplné simulace řídicích algoritmů v prostředí Matlab-Simulink. Po simulačním ověření lze řídicí algoritmy přeložit do binárních konfiguračních souborů, které lze pomocí diagnostického protokolu založeného na standardu TCP/IP poslat přímo do cílových zařízení a podle nich zahájit řízení bez nutnosti odstavení zařízení. Ekvivalentní chování simulace a řízení v reálném čase zaručuje rozsáhlá knihovna funkčních bloků ve verzích jak pro Simulink tak i každou cílovou platformu. Systém REX nevyužívá Real Time Workshop firmy The Mathworks.

Klíčová slova: průmyslový řídicí systém, vnořené řízení, Simulink, OPC, operační systém reálného času

1 Úvod

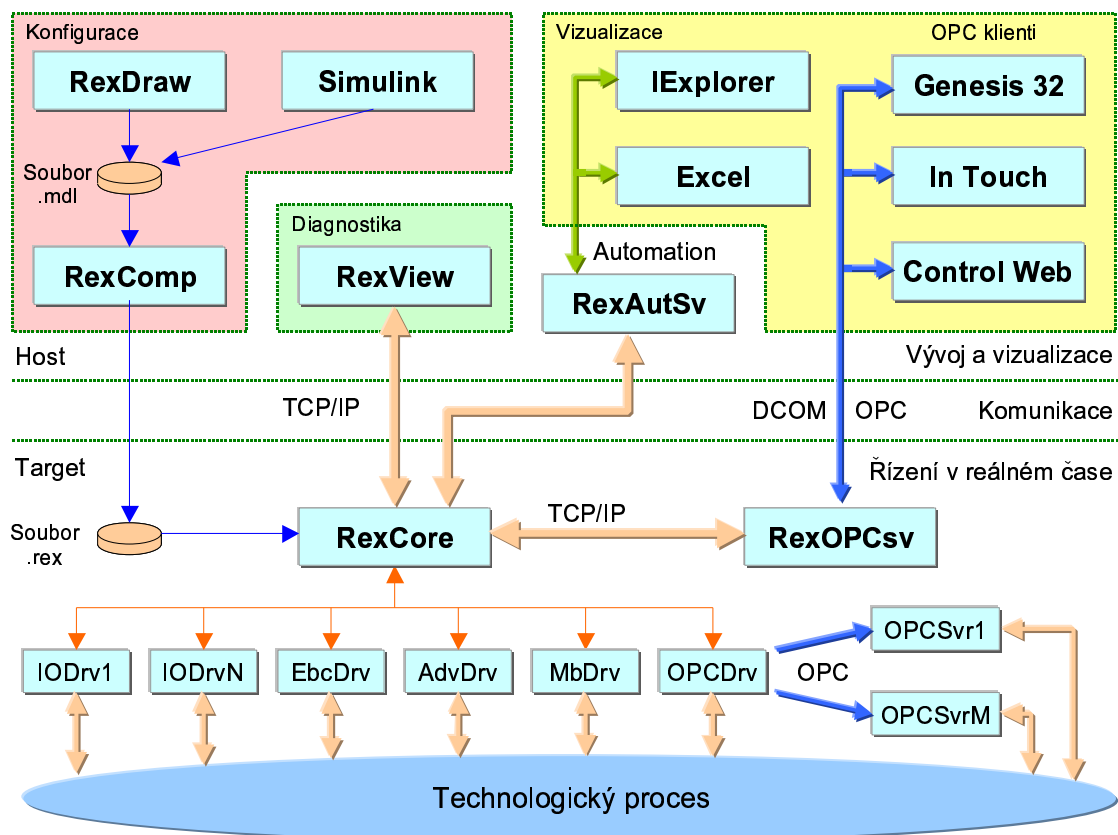
Již delší dobu lze pozorovat tlak ze strany zákazníků na neustálé zkracování doby určené pro instalaci a uvedení do provozu řídicích systémů technologických procesů. Přitom jsou nasazované řídicí systémy stále složitější a rozsáhlejší. Tato skutečnost nutí dodavatele aby nasazovali co možná nejlépe ověřená řešení v předchozích aplikacích. Pokud jde o inovační řešení řídicího systému, lze dobu nutnou k uvedení do provozu podstatně zkrátit předchozí detailní simulací. Pravděpodobně nejrozšířenějším nástrojem pro vývoj a testování nových algoritmů v oblasti řízení procesů je programový systém Matlab-Simulink, který postupně proniká z oblasti akademického výzkumu na univerzitách i do inženýrské praxe.

Cílem tohoto příspěvku je stručně popsat vytváření aplikací nového řídicího systému REX s důrazem na možnost téměř úplné předchozí simulace v systému Simulink. Řídicí systém REX je otevřený a škálovatelný systém vhodný pro vnořené (embedded) řízení, přenositelný na různé platformy s překladači jazyka C a C++ od jednoúčelových řídicích desek s jednoduchou exekutivou reálného času až po procesní stanice se standardními operačními systémy (Windows NT/2000/XP, VxWorks, apod.). Kompatibilita řídicího systému REX s programovým balíkem Simulink byla jednou ze základních myšlenek návrhu systému REX. Následující sekce popisují:

- 2 Strukturu systému REX
- 3 Běh systému REX a diagnostiku
- 4 Konfiguraci systému REX
- 5 Využití standardu OPC (OLE for Process Control)

2 Struktura řídicího systému REX

Struktura řídicího systému REX a jeho vazby na okolní prostředí jsou schematicky znázorněny na obr. 1.



Obrázek 1: Struktura řídicího systému REX

V horní části obrázku jsou obsaženy komponenty vývojového prostředí (Host), které je současně i prostředím pro vizualizaci a operátorské ovládání řídicího systému. Všechny současné vývojové nástroje systému REX jsou určeny pro operační systémy Windows 95/98/ME/NT/2000/XP. Pro vizualizaci se používají standardní vizualizační nástroje (některé jsou přímo uvedeny v obrázku). Vizualizační nástroje jsou napojeny na řídicí systém buď přes rozhraní OPC, využívající služby COM (Component Object Model) a DCOM (Distributed COM), nebo přes Automation (OLE Automation) a skripty (VBScript, JScript, Visual Basic).

Spodní část obrázku naznačuje strukturu cílového prostředí (Target), které realizuje vlastní řízení v reálném čase. Cílovým prostředím mohou být (stejně jako u vývojového prostředí) operační systémy Windows 95/98/ME/NT/2000/XP, ale i např. Windows CE, VxWorks nebo Real Time Linux. V případě operačních systémů Windows může být cílové prostředí totožné s vývojovým, dokonce na jednom počítači.

Spojovacím článkem mezi prostředími Host a Target je komunikační vrstva (na obr. 1 uprostřed). Nejčastěji používaným protokolem pro komunikaci je standard TCP/IP, nad nímž je vybudován vlastní diagnostický protokol systému REX.

3 Běh systému REX a diagnostika

Program RexCore, trvale provozovaný na cílové platformě, tvoří jádro řídicího systému REX. Aby bylo možno sledovat činnost řídicího systému na vývojové platformě (zejména v procesu

uvádění do provozu), jsou třeba ještě další nástroje. Částečně lze provoz systému sledovat i v již zmíněných vizualizačních nástrojích, detailní pohled však poskytuje program RexView.

3.1 RexCore – jádro řídicího systému

Jádro řídicího systému RexCore je dost složitý program provádějící paralelně různé činnosti obvyklé v řídicích systémech. Jednotlivé činnosti jsou vykonávány na základě priorit v režimu preemptivního multitaskingu pomocí jednotlivých subsystémů jádra. Jádro obsahuje subsystémy:

Subsystém reálného času – stará se o spuštění jednotlivých úloh a v nich vložených funkčních bloků, řídí spuštění ovladačů, získává a poskytuje diagnostické informace o časování úloh a ovladačů a vytížení systému

Vstupně-výstupní subsystém – poskytuje rozhraní pro ovladače technických prostředků pro získávání vstupů z procesu a nastavování výstupů

Algoritmický subsystém – obsahuje algoritmy funkčních bloků, které jsou volány z úloh subsystému reálného času

Diagnostický subsystém – poskytuje diagnostické informace o běhu řídicího systému, umožňuje download a ladění aplikací

Archivační subsystém – slouží pro archivaci událostí, alarmů a historických trendů veličin řídicího systému

3.2 RexView – diagnostický nástroj

Program RexView umožňuje sledovat, co se děje v jádře řídicího systému REX při jeho běhu, a proto je velmi důležitým nástrojem při uvádění řídicího systému do provozu i v případě vzniku nějakých problémů již během rutinního provozu. Program poskytuje detailní, hierarchicky uspořádané informace o všech subsystémech jádra. Komunikace pomocí protokolu TCP/IP umožňuje připojit se k běžícímu jádru na lokálním počítači, v lokální síti i ve vzdálené síti (např. přes Internet).

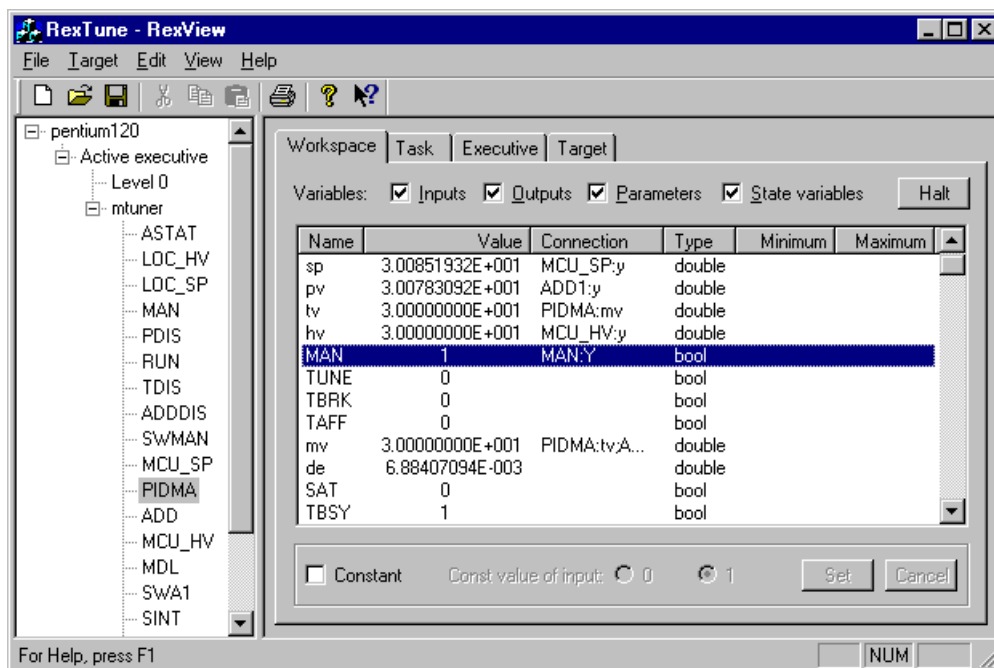
Na obrázku 2 je patrna ukázka jednoho typu snímku programu RexView, podrobný popis celého programu je uveden v příručce [2].

Levou část snímku tvoří hierarchie objektů v řídicím systému uspořádaná podle jednotlivých subsystémů. V této stromové struktuře je vybrán blok PIDMA, realizující PID regulátor s vestavěným autotunerem [6], který je v pořadí jedenáctým blokem úlohy *mtuner* subsystému reálného času.¹

V pravé části snímku je pro blok PIDMA vybrána záložka *Workspace* s proměnnými pracovního prostoru bloku. V horní části záložka obsahuje zaškrťovací políčka pro volby druhů proměnných pracovního prostoru, které se zobrazují (vstupy, výstupy, parametry a stavy). Ve střední části je vidět seznam vybraných proměnných bloku s jejich periodicky aktualizovanými hodnotami a připojením vstupů a výstupů k ostatním blokům. Spodní část umožňuje zadávat hodnoty parametrů bloků a simulovat hodnotu vybraných vstupů. V tomto příkladě je vybrán vstup *MAN* rovný 1 (tj. regulátor je v manuálním režimu); pokud bychom chtěli pro ladicí účely přepnout regulátor „natvrdo“ do automatického režimu, stačí zaškrtnout políčko *Constant*, v políčku *Const value of input* zvolit hodnotu 0 a stisknout tlačítko *Set*.

Ostatní záložky pravé části snímku odpovídají vlastnostem nadřazených objektů vybraného objektu (zde bloku PIDMA):

¹Bloky dané úlohy jsou ve stromové struktuře zobrazeny v tom pořadí, v jakém se v systému REX spouštějí.



Obrázek 2: Ukázka obrazovky programu RexView

Task zobrazuje diagnostiku úlohy *mtuner* (měřené okamžité, průměrné, minimální a maximální doby trvání jednoho cyklu úlohy, indikace výpočetních chyb, aj.)

Executive zobrazuje identifikační údaje a diagnostiku běžící exekutivy reálného času *Active executive* (datum vytvoření, nahrátí a posledního spuštění konfigurace, obsazení paměti, aj.)

Target identifikuje cílové zařízení (zde *pentium120*), se kterým program *RexView* právě komunikuje (typ cílového zařízení, název a verzi operačního systému, verzi a datum sestavení systému *REX*, apod.)

Pro ostatní objekty jádra řídicího systému *REX* existují další záložky, které se zobrazí při vybrání objektu daného typu, např. pro moduly, ovladače, trendy a archivy.

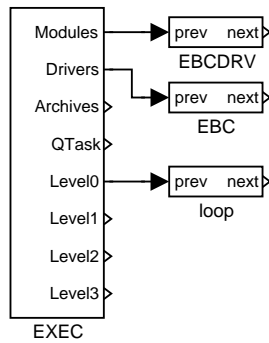
4 Konfigurace systému *REX*

Systém *REX* se konfiguruje formou vytváření funkčních schémat složených z funkčních bloků z rozsáhlé knihovny bloků [4]. Knihovna bloků existuje ve verzích pro systém *Simulink* i každou cílovou platformu (*Target*) řídicího systému *REX*. To umožňuje kreslit funkční schémata přímo v grafickém editoru zabudovaném do systému *Simulink*. Druhou možností je použít vlastní grafický editor *RexDraw*.

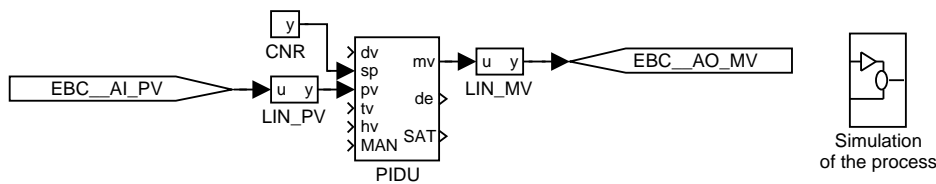
Simulink i *RexDraw* ukládají funkční schémata do souborů s příponou *.mdl* (model). Tyto textové soubory jsou příliš rozsáhlé pro přenos na některá cílová zařízení, např. pro paměťově omezená zařízení pro vnořené (embedded) řízení. Proto jsou soubory *.mdl* překládány do binárního formátu *.rex* překladačem *RexComp* (sekce 4.2).

Konfigurace systému *REX* bude demonstrována na příkladě jednoduché regulační smyčky, který je znázorněn na obrázcích 3, 4 a 5.

Vzájemné vazby všech tří obrázků a důvody právě takového zkonfigurování jsou vysvětleny ve zbytku této kapitoly.



Obrázek 3: Hlavní soubor příkladu jednoduché regulační smyčky `exec.mdl`



Obrázek 4: Vlastní konfigurace příkladu jednoduché regulační smyčky `loop.mdl`

4.1 RexDraw – grafický editor funkčních schémat

Program RexDraw umožňuje navrhovat funkční schémata řídicího systému REX velmi podobným způsobem, jako se konfiguruje ve vestavěném editoru systému Simulink. Oba programy generují soubory s příponou `.mdl`, v možnostech konfigurace však existují určité rozdíly.

Především editor RexDraw umožňuje vytvářet soubory `.mdl` složené jen z bloků z rozsáhlé knihovny systému REX [4]. Všechny bloky uvedené knihovny pracují v diskrétním čase, i když jejich velká část je diskretizována pro danou periodu vzorkování. Tato skutečnost odpovídá nastavení parametrů Solver options v Simulinku na Type: Fixed step a discrete (no continuous states).

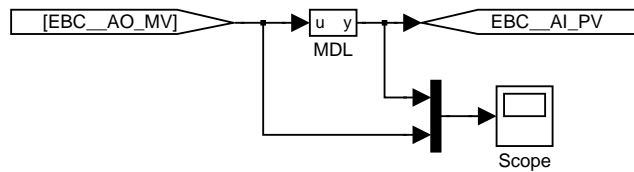
Zatímco v systému Simulink je celá konfigurace tvořena jediným souborem, který může obsahovat více subsystémů (v našem případě soubor `loop.mdl` z obr. 4 a subsystém Simulation of the process z obr. 5), v systému REX musí být konfigurace tvořena alespoň dvěma soubory, z nichž právě jeden je tzv. *hlavním souborem projektu* (v našem případě soubor `exec.mdl` z obr. 3).

Hlavní soubor projektu specifikuje konfiguraci jednotlivých subsystémů jádra řídicího systému RexCore, popsanych v odst. 3.1. V uvedeném příkladě z obr. 3 je situace velmi jednoduchá, exekutivu reálného času RexCore vyjadřuje blok EXEC, ostatní bloky mají následující význam:

EBCDRV je jedním z modulů řídicího systému REX. Konkrétně je v tomto modulu implementován ovladač stanic s rozhraním EBC (Ethernet Base Controller)², to ovšem systému REX říká až následující blok.

EBC je blokem vstupně výstupního ovladače, jehož jedním parametrem je jméno modulu, implementujícího daný ovladač, v tomto případě je to EBCDRV). Dalším parametrem je jméno konfiguračního souboru ovladače, který kromě vlastních parametrů vstupně-výstupního zařízení obsahuje hlavně informace odkud se mají zjišťovat hodnoty vstupů a kam se mají nastavovat hodnoty výstupů řídicího systému. Z hlediska projektování řídicího systému je

²Rozhraním EBC mohou být vybaveny např. moduly řady Terminator I/O nebo automaty řady DL-205, jejichž distributorem je firma Automationdirect.com. Konkrétně moduly řady Terminator I/O jsou použity pro měření analogového vstupu regulované veličiny a nastavování analogového výstupu akční veličiny regulační smyčky.



Obrázek 5: Simulační subsystém `Simulation of the process` z konfigurace `loop.mdl`

nejdůležitější jméno bloku samotného (zde EBC), které určuje prefix názvů všech vstupů a výstupů které budou k tomuto ovladači připojeny, blíže viz odst. 4.2.

`loop` je po přidání přípony `.mdl` názvem souboru, ve kterém je zkonfigurována řídicí úloha, která je zde zařazena do řídicí úrovně 0 (výstup `Level0` bloku `EXEC`). Je to právě úloha z obr. 4.

Právě popsané tři bloky mají vstupy `prev` (předchozí) a `next` (následující), které umožňují do řady za sebou přidávat další bloky reprezentující v konfiguraci další moduly, ovladače i řídicí úlohy. Stejně pravidlo platí i pro v tomto příkladu nepoužité archivy, které se kreslí připojením na výstup `Archives` bloku `EXEC`, a úlohy z dalších řídicích úrovní (`Level1`, `Level2` a `Level3`). Jedinou výjimkou je (opět nepoužitý) blok `Qtask` pro velmi rychlou řídicí úlohu, který smí být v systému REX nejvýše jeden.

4.2 RexComp – překladač konfigurací

Na základě hlavního souboru projektu aplikace ve formátu `.mdl` generuje program `RexComp` binární konfigurační soubor `.rex` řídicího systému REX. Připomeňme, že příklad konfigurace hlavního souboru projektu `exec.mdl` je znázorněn na obr. 3. Překlad konfigurace se skládá zhruba řečeno z následujících kroků:

1. Nalezení bloku exekutivy reálného času `EXEC` v hlavním souboru projektu, kontrola parametrů exekutivy.
2. Nalezení a kontrola všech objektů řídicího systému, které jsou nakresleny v hlavním souboru projektu.
3. Přidání jednotlivých objektů do konfigurace: modulů, ovladačů, archivů, rychlé úlohy a úloh jednotlivých použitých výpočetních úrovní.
4. Alokace paměti a nastavení parametrů bloků z konfigurace.
5. Kontrola a propojení řídicích úloh.
6. Kontrola správnosti celé konfigurace.
7. Uložení přeloženého souboru s příponou `.rex` na disk.

Při svém spuštění vypisuje překladač informace o překládaných souborech a případně i výskyt chyb překladu. V každém z uvedených kroků může být detekována závažná chyba, která ukončí překlad konfigurace a zabrání vytvoření výsledného binárního souboru.

Pro efektivní přechod od simulace v systému Simulink k řízení v reálném čase jsou v překladači `RexComp` implementovány následující dva klíčové rysy:

1. Všechny subsystémy (v souborech `.mdl`), jejichž název začíná slovem `Simulation`, jsou vypuštěny (tj. nepřekládají se)

2. Všechny bloky `From` a `Goto` jejichž značka má tvar `<prefix>__<name>` jsou nahrazeny standardními vstupními a výstupními bloky systému REX, které se odkazují na blok ovladače s názvem `<prefix>`, v němž hledají symbolický název vstupu nebo výstupu `<name>`. Dva znaky `_` (podtržítka) za sebou slouží jako oddělovač specifikující, že jde o vstup nebo výstup ovladače.

Vrátíme-li se k našemu příkladu, je podle pravidla 1 z obr. 4 vypuštěn subsystém `Simulation of the process`. Podle pravidla 2 je blok `From` se značkou `EBC__AI_PV` nahrazen standardním vstupem systému REX, který se odkazuje na signál `AI_PV` ovladače, zkonfigurovaného v hlavním souboru projektu (obr. 3) blokem s názvem `EBC`. Podobně blok `Goto` se značkou `EBC__AO_MV` odkazuje na signál `AO_MV` téhož ovladače.

Při dodržení uvedených pravidel lze právě popsaným postupem přejít od simulace řídicího systému v systému Simulink ke konfiguraci systému reálného času REX, bez jakékoliv změny konfiguračních souborů.

5 Využití standardu OPC (OLE for Process Control)

OPC je relativně nový, ale dnes už velmi rozšířený standard pro výměnu dat mezi prostředky pro řízení procesů, zejména mezi procesními stanicemi a nadřazenou vrstvou vizualizace. Současná verze systému REX podporuje specifikaci OPC Data Access [5] dvěma způsoby, popsanými v následujících podkapitolách.

5.1 RexOPCsv – OPC server systému REX

Pro komunikaci zejména s nadřazeným systémem byl vytvořen OPC Data Access server `RexOPCsv`, znázorněný ve spodní části (`Target`) obrázku 1. Nadřazeným systémem může být jakýkoliv systém, který je klientem OPC Data Access verze 2.0. Takových systémů jsou dnes na trhu desítky, několik z nich je přímo naznačeno na obr. 1.

Uvedená struktura je použitelná tehdy, podporuje-li cílové prostředí standard DCOM, tj. zejména pro operační systémy Windows. Pro platformy podporující pouze TCP/IP lze server spouštět až v prostředí `Host`. Tuto strukturu lze výhodně použít i v rozsáhlých sítích (Internet), kde vstupní brány do lokálních sítí jsou zabezpečeny proti neoprávněnému přístupu (Firewall) a tím vzniká problém s průchodností paketů DCOM.

5.2 OPCDrv – ovladač pro komunikaci s OPC servery

Druhý způsob využití standardu OPC představuje ovladač `OPCDrv`. Na rozdíl od serveru `RexOPCsv` je tento ovladač klientem OPC Data Access 2.0, a tak umožňuje systému REX číst (nastavovat) vstupy (výstupy) libovolných zařízení, pro která existují OPC Data Access servery. Takových zařízení s příslušnými servery jsou dnes na trhu stovky od mnoha firem.

6 Závěr

Článek se snažil představit nový řídicí systém REX pro čtenáře seznámené s používáním rozsáhlého simulačního systému Matlab-Simulink. Při návrhu řídicího systému REX byl kladen důraz na možnost věrné simulace aplikace řídicího systému v prostředí Matlab-Simulink ještě před jejím nasazením. Z tohoto důvodu je konfigurace systému REX do značné míry kompatibilní s konfigurací systému Simulink (sekce 4). Za určitých podmínek, popsaných v sekci 4.2 lze dokonce přejít (či během procesu uvádění do provozu i opakovaně přecházet) od simulace v systému Simulink k řízení v reálném čase v systému REX bez jediné změny konfigurace.

Reference

- [1] The Mathworks 2001. Using Simulink, Version 4.1
- [2] Řídicí systém REX – Uživatelská příručka. REX Controls s.r.o., Plzeň, 2002.
- [3] Funkční bloky systému REX. REX Controls s.r.o., Plzeň, 2002.
- [4] Schlegel M., Balda P., Štětina M.: Knihovna C MEX bloků pro průmyslovou regulaci s aplikačními příklady. Konference Matlab 2001, Humusoft s.r.o.
- [5] OPC Foundation 2000. OLE for Process Control, Data Access Custom Interface Standard, Version 2.04
- [6] Schlegel M., Balda P., Štětina M.: PID autotuner pro průmyslové použití. Automatizace energetických procesů '02, Zlín, Česká republika