

KNIHOVNA C MEX BLOKŮ PRO PRŮMYSLOVOU REGULACI S APLIKAČNÍMI PŘÍKLADY

M. Schlegel, P. Balda, M. Štětina
Západočeská univerzita v Plzni, ZAT a. s. Příbram

Abstrakt

V příspěvku je stručně popsána nová knihovna funkčních bloků (C MEX) pro programový systém SIMULINK a jeho nadstavby ERTT, RTW a dSPACE určená pro realizaci vysoce kvalitních řídicích systémů regulačního typu. Základní záměr při vytváření této knihovny byla snaha skloubit různorodé inženýrské požadavky se současnými teoretickými přístupy. Použití bloků je ilustrováno na standardních regulačních úlohách (kroková, programová, kaskádní, selektorová, adaptivní regulace).

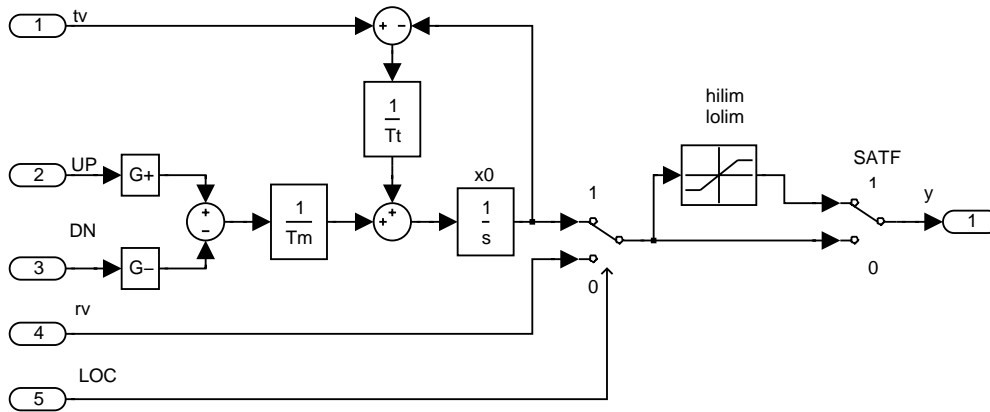
1 Úvod

Simulink je pravděpodobně nejrozšířenější nástroj pro vývoj a testování nových algoritmů v oblasti řízení procesů. Zcela výsadní postavení má v tomto směru na univerzitách. Díky jeho nadstavbám RTW (Real Time Workshop) a dSpace (dSpace GmbH), které umožňují automaticky generovat spustitelný kód na paměťově limitovaném hardwaru, postupně proniká i do reálných průmyslových aplikací. Naznačená technologie vývoje pokročilých řídicích systémů založená na Simulinku navíc snižuje propast mezi akademickým výzkumem a průmyslem. Teoretici, pracující na výzkumu nových algoritmů řízení procesů, se nemusí zabývat konkrétním cílovým hardwarem, ale stačí, když své algoritmy převedou do podoby knihovny funkčních bloků (Blockset) pro Simulink. Jejich další průmyslové využití je pak již téměř automatizováno.

Cílem tohoto příspěvku je stručně popsat nově vytvořenou knihovnu C MEX funkčních bloků pro průmyslovou regulaci. Hlavní záměr při jejím vývoji byla snaha skloubit různorodé inženýrské požadavky se současnými teoretickými přístupy [1]. Pomocí funkčních bloků z této knihovny je možné kvalitně realizovat nejen všechny standardní regulační obvody takové, jako dvouhodnotová, spojitá, kroková, kaskádní, selektorová, programová regulace, ale i pokročilé řídicí systémy s automatickým nastavováním parametrů regulátorů, s variabilní strukturou (přepínání typu a parametrů regulátoru), s fuzzy regulací a se zcela speciálními „jednoúčelovými“ algoritmy, realizovanými volně programovatelným blokem v jazyku C.

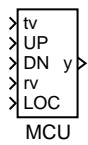
2 Knihovna bloků

V tomto oddílu se zmíníme o nejvýznamnějších blocích nově vytvořené knihovny. Většina těchto bloků je zařazena do podskupiny bloků pro průmyslovou regulaci `RegLib`. Mlčky se předpokládá, že při vytváření řídicího systému návrhář použije též bloky ze speciálních skupin `AnaLib`, `MathLib` a `LogicLib` a `SpecLib`, vytvořených též autory tohoto příspěvku, a nebo standardní diskrétní bloky Simulinku. Podrobněji popíšeme pouze blok PID regulátoru (PIDU), jeho varianty s přepínáním parametrů (PIDGS) a s reléovým autotunerem (PIDAT) a volně programovatelný blok (LANG) ze skupiny `SpecLib`. U ostatních bloků, použitých v dále uvedených příkladech, naznačíme pouze jejich základní funkci.



Obrázek 1: Podrobná funkce bloku MCU

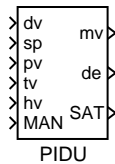
2.1 MCU – Jednotka pro ruční zadávání



V lokálním režimu ($LOC = 1$) je blok MCU určen k ručnímu zadávání výstupu y pomocí tlačítek „více“ (vstup UP) a „méně“ (vstup DN). Strmost najíždění z počáteční hodnoty y_0 na žádanou hodnotu je určena integrační časovou konstantou t_m a dobou stlačení ovládacích tlačítek. Po uplynutí každých t_a sekund je strmost vždy násobena faktorem q , až do vypršení doby t_f .

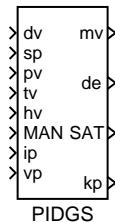
Rozsah výstupu y může být omezen ($SATF = 1$) saturačními mezemi $lolim$ a $hilim$. V případě, že žádné z tlačítek není stlačeno ($UP = 0$ a $DN = 0$), vysleduje výstup y vstupní hodnotu tv . Rychlost vysledování je dána integrační časovou konstantou tt . V případě $LOC = 0$ je vstup rv s případnými omezeními ($SATF = 1$) kopírován na výstup y . Podrobná funkce bloku je přímo patrná z obr. 1.

2.2 PIDU, PIDGS – PID regulátor, PID s přepínáním parametrů



Blok PIDU je základní blok pro vytvoření úplného regulátoru PID (P, I, PI, PD, PID, PI+S). V nejjednodušším případě může pracovat zcela samostatně a plnit standardní funkci PID regulátoru s dvěma stupni volnosti, v automatickém ($MAN = 0$) nebo manuálním režimu ($MAN = 1$). V automatickém režimu v lineární oblasti realizuje zákon řízení daný vztahem

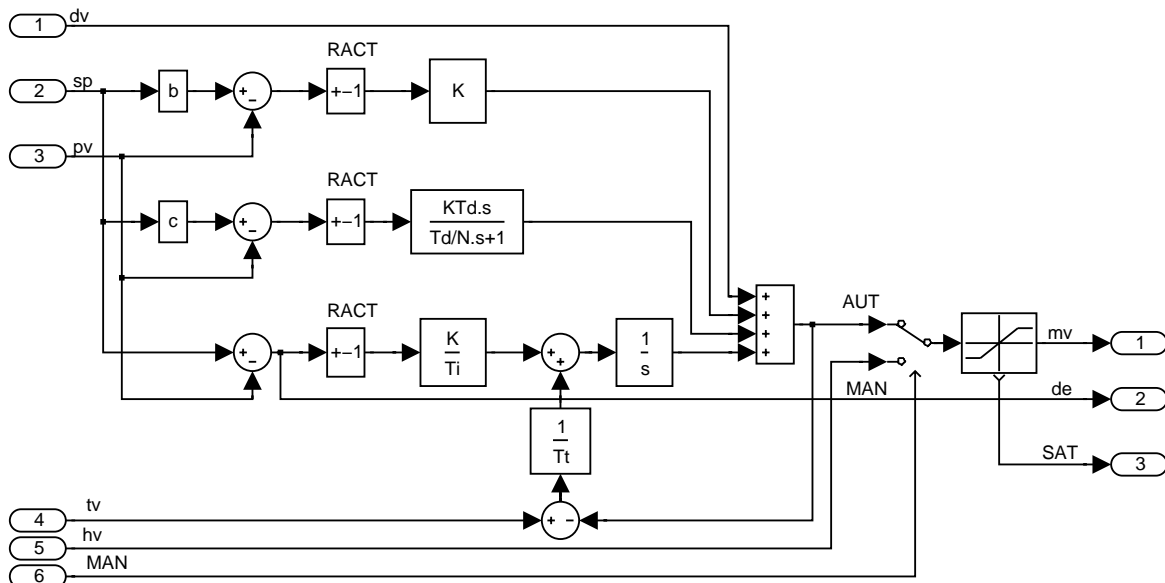
$$U = \pm K \left[bW - Y + \frac{1}{T_i s} E + \frac{T_d s}{T_d s / N + 1} (cW - Y) \right] + Z$$



kde $U(s)$ je obraz akční veličiny mv , $W(s)$ je obraz požadované hodnoty sp , $Y(s)$ je obraz regulované veličiny pv , $Z(s)$ je obraz dopředné vazby dv a K , T_i , T_d , N , b , c jsou parametry regulátoru.

Znaménko pravé strany je definováno parametrem $RACT$, určujícím směr působení akční veličiny mv na regulovanou veličinu pv ($RACT = 0$: větší $mv \rightarrow$ větší pv ; $RACT = 1$: větší $mv \rightarrow$ menší pv). Rozsah výstupu mv je omezen saturačními mezemi $lolim$ a $hilim$. Propojením výstupu mv se vstupem tv a vhodnou volbou parametru tt dosáhneme žádaného chování regulátoru při dosažení saturačních hodnot mv . Odstraníme tak nežádoucí unášení integrační složky (*wind up effect*) a současně s tím zajistíme bezrázové přepínání (*bumpless transfer*) automatického a manuálního režimu. V manuálním režimu je vstup hv (po případném omezení) kopírován na výstup mv . Signál připojený na vstup tv zajišťuje v tomto režimu příslušné vysledování vnitřního stavu regulátoru pro následné bezrázové přepnutí do automatického režimu. Funkce bloku je dobře patrná z vnitřního schématu, zobrazeného na obr. 2.

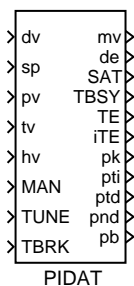
Blok PIDGS se liší od bloku PIDU pouze tím, že má 6 kompletních sad parametrů, které je



Obrázek 2: Vnitřní schéma bloku PIDU

možné přepínat buď celočíselným vstupem *ip* nebo analogovým vstupem *vp* podle zadaných přepínacích mezí.

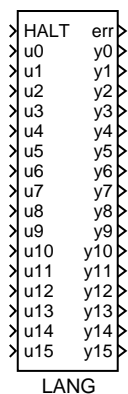
2.3 PIDAT – PID regulátor s autotunerem



Blok PIDAT má zcela stejné regulační funkce jako blok PIDU. Navíc je vybaven funkcí automatického nastavování parametrů regulátoru. Pro využití této funkce je nutné převést řízený systém do přibližně ustáleného stavu (ve vhodném pracovním bodě), zvolit požadovaný typ regulátoru (PI nebo PID) a aktivovat vstup TUNE hodnotou 1 (start identifikačního experimentu). V následném identifikačním experimentu je řízený proces regulován pomocí speciálního adaptivního reléového regulátoru a ze získaného záznamu vstupu a výstupu procesu je odhadnut vhodný bod jeho frekvenční charakteristiky. Na základě toho jsou poté určeny parametry regulátoru.

Amplitudu reléového regulátoru (úroveň vybuzení systému) je možné nastavit parametrem *amp* a jeho hysterezi parametrem *hys*. Zvolíme-li $hys = 0$, potom se hystereze relé určí automaticky na základě odhadu úrovně šumu měření regulované veličiny. Během identifikačního experimentu je $TBSY = 1$. Po řádném skončení experimentu je $TE = 0$ a vypočítané parametry se objeví na výstupech *pk*, *pti*, *ptd*, *pnd*, *pb*. Skončil-li experiment s chybou je $TE = 1$ a *iTE* blíže specifikuje důvod chyby. Při výskytu chyby se doporučuje zvětšit parametr *amp*. Jeho volbu usnadňuje zabudovaná funkce, která parametr *amp* automaticky zmenšuje při hrozbě překročení maximální dovolené odchylky *maxdev* regulované veličiny od jejího počátečního ustáleného stavu. Identifikační experiment je možné předčasně ukončit aktivací vstupu TBRK.

2.4 LANG – Volně programovatelný blok



V některých případech se může stát, že je do řídicího algoritmu nutné implementovat funkci, kterou nelze z dostupné množiny bloků vytvořit (popř. to sice lze, ale komplikovaně a neefektivně). Proto byl vyvinut blok LANG, který implementuje kód naprogramovaný uživatelem v jazyku, syntakticky velmi podobném jazyku C (nebo JAVA). Napojení na vstupy u0 až u15 a výstupy y0 až y15 umožňují nová klíčová slova `input` a `output`. Vstup `HALT` umožňuje zastavit periodické spouštění algoritmu bloku, výstup `err` indikuje kód chyby výpočtu. Zdrojový soubor algoritmu lze vytvořit v libovolném textovém editoru a jeho jméno uvést v parametru `srcname`.

Všechny běžné konstrukce použitého jazyka jako jsou výrazy, přiřazovací příkazy, příkazy `if`, `switch`, `for`, `while`, apod. mají syntaxi totožnou se syntaxí jazyka C. Ve výrazech lze pracovat s jednorozměrnými poli hodnot (vektory) a používat všechny běžné matematické funkce (`abs`, `sqrt`, `sin`, `exp`, `pow`, atd.), další speciální funkce `poly` (výpočet hodnoty polynomu), `conv` (výpočet konvoluce), `scal` (skalární součin), `min`, `max`, `summ`, `avg` (minimální a maximální hodnota, součet a průměr z libovolného počtu hodnot).

Jsou zde však i některá omezení oproti jazyku C: není implementován preprocesor, jsou podpořeny jen typy `double` a `long` (lze použít i `int`, `short`, které se interně zpracovávají jako `long`, a `float`, který se interně zpracovává jako `double`), není implementován `typedef`, ukazatele (pointery) a struktury.

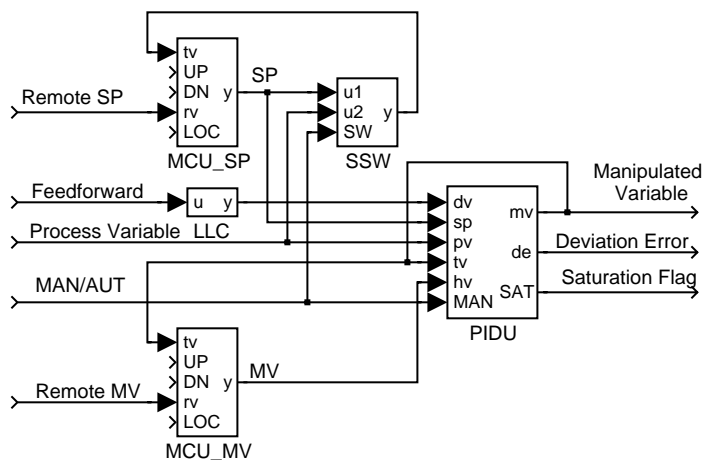
2.5 Ostatní bloky použité v příkladech

Následující tabulka uvádí název a stručný popis ostatních bloků, použitých v příkladech z od-
dílu 3.

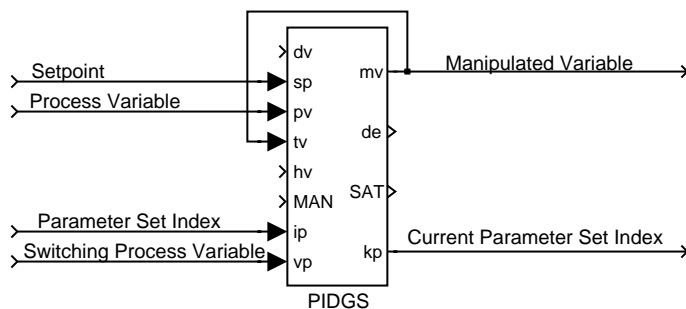
PRGM	Programátor požadované hodnoty
SELU	Selektor pro realizaci selektorové regulace
SWU	Selektor pro výběr sledované veličiny
SCU	Krokový regulátor ventilu
SCUM	Krokový regulátor ventilu bez polohové zpětné vazby
FLCU	Fuzzy regulátor
MVD	Model ventilu s pohonem
SSW	Jednoduchý přepínač
SAT	Saturace s řízenými mezemi
RLY	Relé s hysterézí
ARLY	Relé s předstihem
LC	Derivační kompenzátor
LLC	Integračně-derivační kompenzátor
MDL	Model řízeného systému
INTE	Integrátor
SPIKE	Nelineární filtr pro odstranění krátkých špiček ze signálu
DER	Derivace, predikce, filtrace signálu
LPF	Dolnopropust
BPF	Pásmová propust
AVG	Vlečný průměr
RLIM	Omezovač strmosti
DEL, DELM	Modely dopravního zpoždění
CMP	Komparátor

3 Příklady použití bloků

V tomto oddílu jsou uvedeny obrázky příkladů použití bloků z knihovny v typických regulačních obvodech (obr. 3 až 11). Tmavě (modře) vyplněné bloky v některých příkladech jsou zde jen pro ilustraci, ve skutečnosti představují model řízeného systému. Na posledním obrázku 12 je ukázka zdrojového textu bloku LANG.

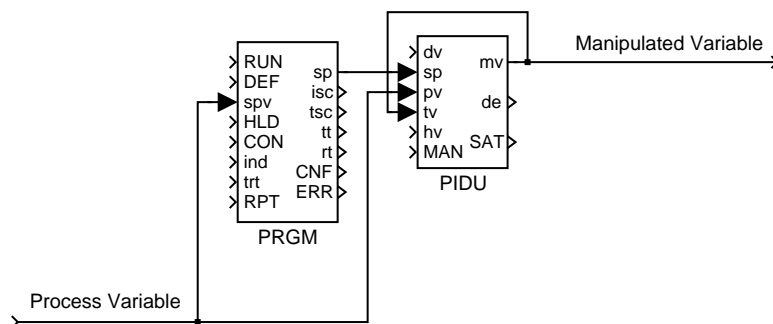


Obrázek 3: Úplné zapojení PID regulátoru se zadávacími jednotkami

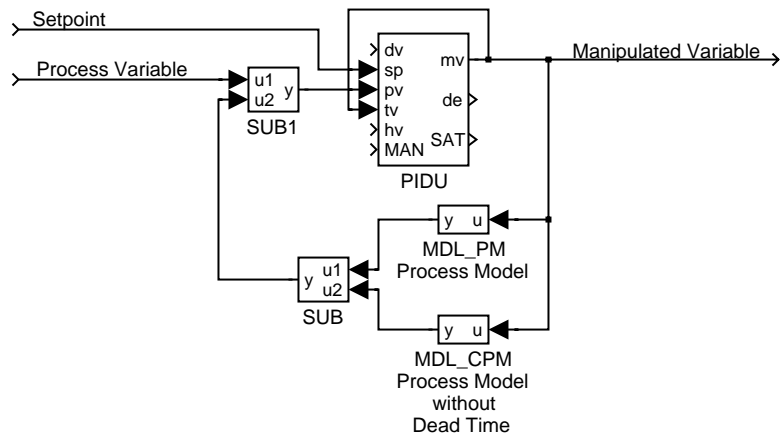


If GSCF=0 then Parameter Sets are switched by the value of Parameter Set Index else by Switching Process Variable

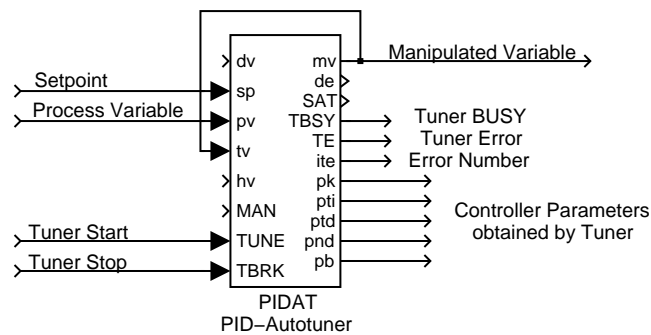
Obrázek 4: Použití regulátoru s přepínáním sad parametrů



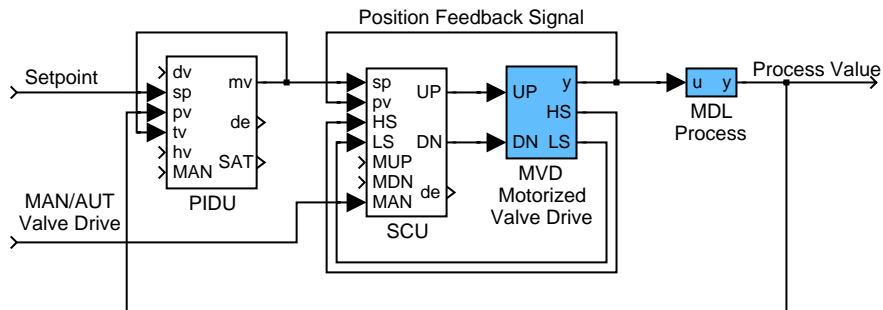
Obrázek 5: Programová regulace



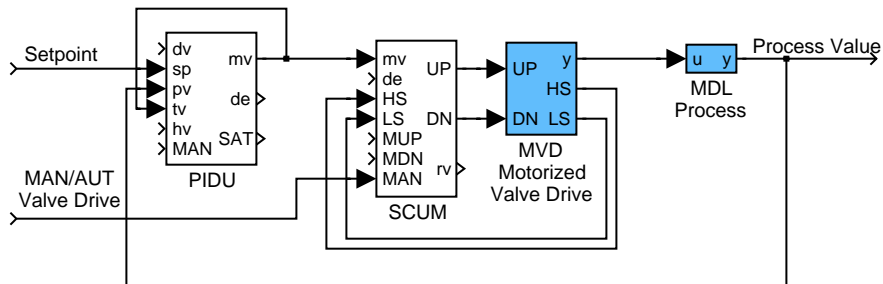
Obrázek 6: Regulátor se Smithovým prediktorem



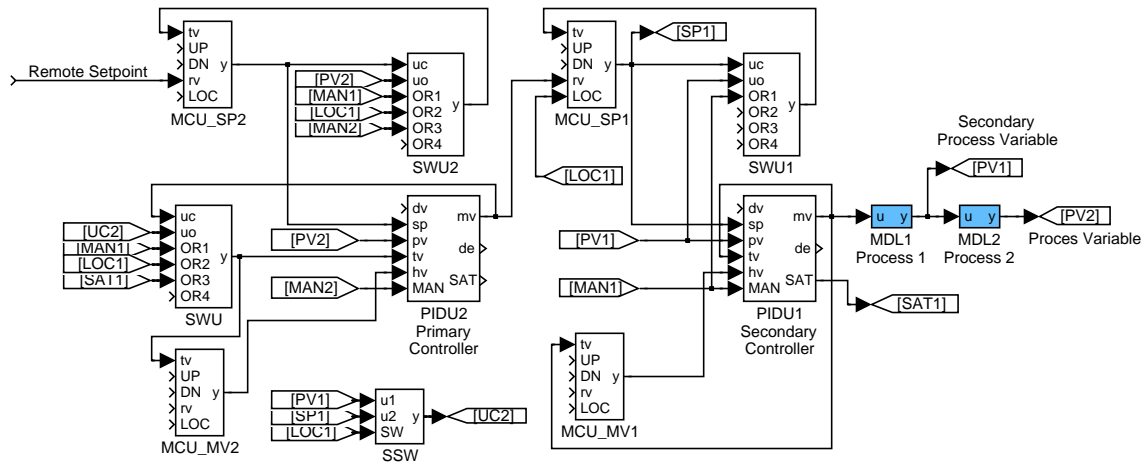
Obrázek 7: Použití PID regulátoru s vestavěným autotunerem



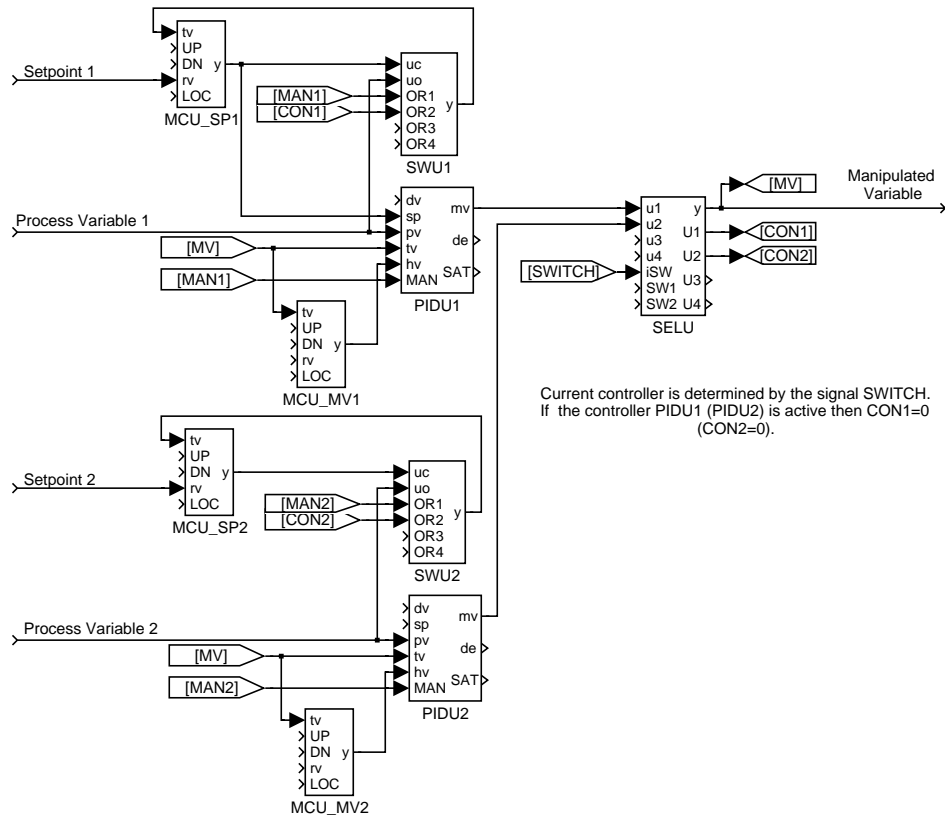
Obrázek 8: Kroková regulace ventilu s polohovou zpětnou vazbou



Obrázek 9: Kroková regulace ventilu bez polohové zpětné vazby



Obrázek 10: Kaskádní regulace



Obrázek 11: Selektorová regulace

```

/* Simulace FIR filtru */
input(0) double vstup; //promenna 'vstup' predstavuje hodnotu u0
output(0) double vystup; //promenna 'vystup' predstavuje prirazeni do y0
double x[5]; //stavove pole
double h[5]; //koeficienty vahove posloupnosti
const long max=5;

long init()
{
    //prirazeni koeficientu
    h[0]=0.5; h[1]=1.0; h[2]= 3.0; h[3]=1.0; h[4]=0.5;
    return 0;
}

long main()
{
    long i;
    double soucet=0;
    for(i=0;i<max-1;i++)
    {
        stav[i]=stav[i+1];
    }
    if(fabs(vstup)>1)
        stav[max-1]=(vstup>0)? 1 : -1;
    else
        stav[max-1]=vstup;

    for(i=0;i<max;i++)
        soucet+=stav[i]*param[max-1-i];
    vystup=soucet;
    return 0;
}

long exit(){ return 0; }

```

Obrázek 12: Příklad zdrojového souboru bloku LANG

4 Závěr

V příspěvku byly stručně popsány některé vybrané bloky z knihovny bloků pro průmyslovou regulaci. V současné době knihovna obsahuje více než 100 různých druhů funkčních bloků, které umožňují realizovat nejen algoritmy regulačního typu, ale i podpůrné algoritmy logického typu včetně sekvenčních automatů a fuzzy logiky. Koncepce knihovny umožňuje postupné rozšiřování a modifikaci řídicích algoritmů tak, jak to požaduje inženýrská praxe. Funkční vlastnosti knihovny byly úspěšně ověřeny na praktické průmyslové aplikaci – řízení vodní turbíny.

Reference

- [1] Åström, K. J. – Hägglund, T.: PID Controllers: Theory, Design and Tuning. Instruments Society of America, Research Triangle Park, NC, second edition, 1995.

Miloš Schlegel, Západočeská univerzita v Plzni, Univerzitní 8, 306 14 Plzeň,
schlegel@kky.zcu.cz

Pavel Balda, ZAT a. s. Příbram, pracoviště Plzeň, Vejprnická 56, 318 02 Plzeň,
pavel.balda@easy-ctrl.cz

Milan Štětina, ZAT a. s. Příbram, pracoviště Plzeň, Vejprnická 56, 318 02 Plzeň,
milan.stetina@easy-ctrl.cz