ProcSISO

JEDNOTKA OBSAHUJÍCÍ PROCES - REGULÁTOR

Příručka uživatele a programátora

***SofCon*®spol. s r.o.**
Střešovická 49
162 00 Praha 6
tel/fax: +420 220 180 454
E-mail: sofcon@sofcon.cz
www: http://www.sofcon.cz

Verze dokumentu 1.10

Informace v tomto dokumentu byly pečlivě zkontrolovány a SofCon věří, že jsou spolehlivé, přesto SofCon nenese odpovědnost za případné nepřesnosti nebo nesprávnosti zde uvedených informací.
SofCon negarantuje bezchybnost tohoto dokumentu ani programového vybavení, které je v tomto dokumentu popsané. Uživatel přebírá informace z tohoto dokumentu a odpovídající programové vybavení ve stavu, jak byly vytvořeny a sám je povinen provést validaci bezchybnosti produktu, který s použitím zde popsaného programového vybavení vytvořil.
SofCon si vyhrazuje právo změny obsahu tohoto dokumentu bez předchozího oznámení a nenese žádnou odpovědnost za důsledky, které z toho mohou vyplynout pro uživatele.

Datum vydání: 16.05.2003
Datum posledního uložení dokumentu: 16.05.2003
(Datum vydání a posledního uložení dokumentu musí být stejné)

Obsah :

1.O dokumentu 5
1.1. Revize dokumentu 5
1.2. Účel dokumentu 5
1.3. Rozsah platnosti 5
1.4. Související dokumenty 5
2.Termíny a definice 5
3.Úvod 6
4.Konstanty a jednoduché typy 6
5.Objekty 6
5.1. tRegSISO 6
5.1.1. Pole 6
5.1.2. Metody 7
5.1.2.1. Init constructor 7
5.1.2.2. Done destructor 8
5.1.2.3. SetParam procedure 8
5.1.2.4. GetParam function 8
5.1.2.5. uu 8
5.1.2.6. yy 9
5.1.2.7. ww 9
5.1.2.8. vv 9
5.1.2.9. MeasureW 9
5.1.2.10. MeasureY 9
5.1.2.11. MeasureV 9
5.1.2.12. Action 9
5.1.2.13. SetUMan 9
5.1.2.14. SetAuto 9
5.1.2.15. ProcRegSiso 9
5.2. tAddRegSISO 10
6.Příklad 10

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Ku		První vydání
1.10	2.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000. Doplněný popis parametrů TW, TT, TimeUntil, ProcExist, FlTw, TsOld. Doplněný popis fce SetUMan, SetAuto, ProcRegSISO.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky definující proces regulátoru.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu je potřeba seznámit se s manuálem VirtSISO a RegSISO.

Popis formátu verze knihovny a souvisejících funkcí je popsán v manuálu LibVer.

2. Termíny a definice

Používané termíny a definice jsou popsány v samostatném dokumentu Termíny a definice.

3. Úvod

ProcSISO je další regulační jednotkou, tentokrát umožňující použití regulačních metod implementovaných v jednotce RegSISO pod **o.s. ReTOS**. Je zde vytvořen základní objekt **tRegSISO** pro tvorbu konkrétních regulačních paralelních procesů. V tomto objektu je definována proměnná ukazující na zvolený regulační objekt, metody pro nastavení a inicializaci paralelního procesu a daného regulátoru a v neposlední řadě jsou zde předdefinovány metody pro měření vstupních údajů regulátoru resp. působení regulátoru na soustavu. Význam pro konkrétní regulátory je nutno dodefinovat přímo v aplikaci. Dědicové objektu vytvoří tedy metody, které jsou specifické pro danou aplikaci regulátoru. Tímto je vytvořeno jednotné rozhraní pro všechny typy regulačních procesů a uživatel může psát aplikační program bez ohledu na použitý regulační algoritmus. Knihovna rovněž definuje rodičovský objekt **tAddRegSISO**, který zajistí, aby daná knihovna byla k aplikaci připojena a byla v aplikaci k dispozici pro případné využití. Uživatel takto tvořených knihoven může až na poslední chvíli definovat, jaký typ regulačního algoritmu použije. Jsou-li dané regulační knihovny k aplikaci přisestaveny, není třeba provádět překlad zdrojového tvaru.

4. Konstanty a jednoduché typy

```
cVerNo = např. $0251; { BCD formát }
cVer   = např. '02.51,07.08.2003';
```

Číslo verze jednotky v BCD tvaru a v textové podobě včetně datumu změny.

```
cName  = 'RSISO';
```

Konstanta definuje implicitní jméno regulačního objektu tRegSISO.

```
pRegSISO = ^tRegSISO;
```

Typ ukazatel na objekt regulátoru.

```
pAddRegSISO = ^tAddRegSISO;
```

Typ ukazatel na seznam objektů regulačních jednotek.

5. Objekty

5.1. tRegSISO

5.1.1. Pole

```
Name        : tNameStr;
```

Proměnná **Name** uchovává jméno použitého regulačního algoritmu. Většinou bude shodné se jménem definovaným pro daný regulátor.

```
Lstack      : Word;
```

Proměnná **LStack** určuje žádanou hodnotu velikosti zásobníku vyhrazeného pro potřeby procesu regulátoru.

```
Sprio       : Integer;
```
Proměnná **SPrio** určuje statickou prioritu procesu.

```
Dprio       : Byte;
```
Proměnná **DPrio** určuje dynamickou prioritu procesu.

```
Tw          : TimeType;
```
Proměnná **Tw** určuje počet tiků na periodu.

```
Ts          : RealP;
```
Proměnná **Ts** určuje periodu opakování procesu.

```
TT          : Byte;
```
Proměnná **TT** určuje počet ms jednoho časového intervalu wait.

```
TimeUntil   : TimeTipe;
```
Pomocné proměnná **TimeUntil** určuje čas čekání. Jen pro vnitřní použití.

```
SISO        : pVirtSISO;
```
Proměnná **SISO** je ukazatel na vlastní regulační algoritmus. Jen pro vnitřní použití.

```
Result      : Word;
```
Do proměnné **Result** je uložen výsledek naposledy volané operace regulačního procesu.

```
ProcExist   : Boolean;
```
Proměnná **ProcExist** je nastavena na true pokud proces existuje.

```
FlTw        : Boolean;
```
Proměnná **FlTw** je nastavena na true pokud byl zadán parametr **Tw**.

```
TsOld       : RealP;
```
V proměnná **TsOld** je uložen poslední hodnota periody vzorkování.

5.1.2. Metody

5.1.2.1. Init constructor

```
constructor Init
```
Konstruktor **Init** slouží k vytvoření instance regulačního procesu. Po jeho volání jsou inicializovány proměnné procesu a je možné volat metody pro nastavení parametrů regulačního procesu. Proměnné objektu jsou inicializovány následovně :

```
Name            := cName;
LStack          := 8000;
SPrio           := 200;
DPrio           := 255;
Ts              := 1;
SISO            := nil;
Result          := Res_OK;
```

5.1.2.2. Done destructor

```
destructor  Done; virtual;
```

Destruktor **Done** slouží k uzavření a zrušení regulačního objektu.

5.1.2.3. SetParam procedure

```
procedure SetParam(Param: TParamStr); virtual;
```

Metoda **SetParam** slouží k novému nastavení parametrů celého regulačního algoritmu. Není třeba definovat pokaždé hodnoty všech parametrů, ale stačí uvést jen ty parametry, kterých se změna bude týkat. Vzhledem k tomu, že regulační proces obsahuje ukazatel na použitou regulační metodu, je nutní parametry pro proces zadávat v první části řetězce, parametry pro regulační metodu za oddělovacím znakem |. Na začátku řetězce parametrů musí být vždy uvedeno jméno určující komu jsou parametry určeny. Přehled jednotlivých parametrů objektu tRegSISO:

NAM=aaa

Parametr určující jméno regulačního procesu, kterému jsou parametry určeny, parametr musí být určen vždy a musí být uveden jako první.

LST= bbb

Parametr určující velikost zásobníku pro proces.

SPR=ccc

Parametr určující velikost statické priority procesu.

DPR=ddd

Parametr určující velikost dynamické priority procesu.

TS=eee

Parametr určující periodu volání procesu.

Příklad:

Nastavíme jméno regulačního procesu na REG1, velikost zásobníku 8000Byte, statickou prioritu na 200, dynamickou prioritu na 254 a periodu volání regulačního procesu na 1 sekundu :

```
SetParam('NAM=REG1,LST=8000,SPR=200,DPR=255,Ts=1');
```

5.1.2.4. GetParam function

```
function  CHGetParam(S: TParamStr): TParamStr; virtual;
```

Metoda **CHGetParam** navrací nastavené hodnoty parametrů daného regulační jednotky. Parametr **S** určuje, z které jednotky a které parametry bude metoda navracet.

5.1.2.5. uu

```
function  uu:RealP;
```

Tato funkce vrátí jako funkční hodnotu velikost naposledy vypočteného akčního zásahu do soustavy.

5.1.2.6. yy

```
function  yy:RealP;
```

Tato funkce vrátí jako funkční hodnotu velikost naposledy změřené výstupní hodnoty regulované soustavy.

5.1.2.7. ww

```
function  ww:RealP;
```

Tato funkce vrátí jako funkční hodnotu velikost naposledy zadané žádané hodnoty.

5.1.2.8. vv

```
function  vv:RealP;
```

Tato funkce vrátí jako funkční hodnotu velikost naposledy naměřené poruchy.

5.1.2.9. MeasureW

```
function  MeasureW: RealP; virtual;
```

Tato funkce provede měření požadované hodnoty a vrátí ji jako svoji funkční hodnotu.

5.1.2.10. MeasureY

```
function  MeasureY: RealP; virtual;
```

Tato funkce provede měření výstupní hodnoty soustavy a vrátí ji jako svoji funkční hodnotu.

5.1.2.11. MeasureV

```
function  MeasureV: RealP; virtual;
```

Tato funkce provede měření poruchy a vrátí ji jako svoji funkční hodnotu.

5.1.2.12. Action

```
procedure Action(U: RealP); virtual;
```

Tato procedura provede působení hodnotou akční veličiny U na vstup regulované soustavy.

5.1.2.13. SetUMan

```
procedure SetUMan(Val:RealP); virtual;
```

Tato procedura nastavuje hodnota **uMan** SISO regulátoru na velikost **Val**.

5.1.2.14. SetAuto

```
procedure SetAuto(Fl:Boolean); virtual;
```

Tato procedura nastavuje stav SISO regulátoru do stavu automatické regulace (Auto = True, Manuál = False) pokud je nastaven příznak **Fl**. V opačném případě nastaví regulátor do stavu manuální regulace.

5.1.2.15. ProcRegSiso

```
procedure ProcRegSISO(RegSISO: pRegSISO); virtual;
```

Procedura spustí samotný procese regulátoru, ve kterém se periodicky počítá akční zásah **U** na základě hodnot **V**, **W** a **Y**.

5.2. tAddRegSISO

Objekt **tAddRegSISO** slouží k vytvoření seznamu objektů regulačních jednotek, které můžeme v dané aplikaci využívat. Je rovněž schopen vygenerovat instanci daného regulačního objektu.

6. Příklad

```
uses
  Kernel,
  ProcSISO,
  VirtSISO,
  RegSISO;

const
  Reg1Ini:string='NAM=REG1,LST=8000,SPR=200,DPR=255,Ts=1';
  Reg1Par1:string='|NAM=PID AM=RegAut w:20 KMd=SofConK
                   Kp=10,Ki=2,Kd=0,ReK=2';
  Reg1Par2='|NAM=PID MxY=100,MnY=-100,MxU=100,MnU=-
            100,uMn=10,EGp=0,IGp=10,DGp=10';

const
  FlEnd : Boolean=false;
  y1    : RealP=0;
  u1    : RealP=0;
  w1    : RealP=0;
  v1    : RealP=0;

type
  pReg1 = ^tReg1;
  tReg1 = object(tRegSISO)
            function  MeasureW: realp; virtual;
            function  MeasureY: realp; virtual;
            function  MeasureV: realp; virtual;
            procedure Action(U: realp); virtual;
  end;

function tReg1.MeasureW: realp;
begin
  w1:=ww;
  MeasureW:=w1;
end;

function tReg1.MeasureY: realp;
begin
  y1:=0.8*y1 + 0.2*uu;
  MeasureY:=y1;
end;

function tReg1.MeasureV: realp;
begin
  v1:=0;
  MeasureV:=v1;
end;

procedure tReg1.Action(U: Realp);
begin
  u1:=uu;
end;

var
  Reg1 : pReg1;
```

```
begin{Main}
  ...
  AddVirt;              {přilinkování potřebných objektů}
  AddPID;
  ...
  StartMain(200,255);               { init Kernelu }
  InitInterruptStack(1,254);
  StartTimeSlicing(8);
  ...
  Reg1:=New(pReg1,Init(Reg1Ini));   { inicializace a spuštěni }
  Reg1^.SetParam(Reg1Par1);         { regulátoru }
  Reg1^.SetParam(Reg1Par2);
  ...
  SetPriority(10,255);
  ...
{zde se vedle regulátoru spusti další procesy zajišťující
komunikaci s obsluhou a osetření případných chyb resp. zajišťující
konec programu nastavením FlEnd na True }
  ...
  repeat
  until FlEnd;
  ...
  Abort('*');
  TerminateMain;
  ...
end{Main}.
```