RegSISO

JEDNOTKA OBSAHUJÍCÍ JEDNOTLIVÉ IMPLEMENTACE REGULAČNÍCH OBJEKTŮ

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

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Ku		První vydání .
1.10	1.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000. Zrušený parametr tPID.Ts. Doplněné parametry tPID.KRp .. eLast. Doplněné metody tAddPID.Init a tAddPID.SISOInit. Doplněné metody tAddP2P.Init a tAddP2P.SISOInit. Doplněná kapitola o tPID2.

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky implementující jednotlivé regulační objekty.

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, popisujícím rozhraní svých potomků.

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

Programová jednotka RegSISO obsahuje objekty **tPID** a **tP2P**, jejichž instance vytváří implementaci regulačních metod PID regulátoru a jednoduchého dvoupolohového regulátoru s hysterezí. Regulační objekty obsahují metody **SetParam** a **GetParam**, umožňující zadávat a číst parametry regulátoru ve formě řetězce, hlavní funkce **ControlAlg** provádí vlastní výpočet akčního zásahu. Jednotka definuje objekty **tAddPID a tAddP2P**, které zajistí, aby dané regulační knihovny byly k aplikaci připojeny a byly v aplikaci k dispozici pro případné využití. Určení parametrů komunikace je voleno parametrem nastavovací metody.

Protože jsou objekty **tPID** a **tP2P** dědicem rodičovského objektu **tVirtSISO**, jsou v této příručce popsány jen odlišnosti a speciality pro daný druh regulační metody. Ostatní naleznete v příručce popisující objekt **tVirtSISO**.

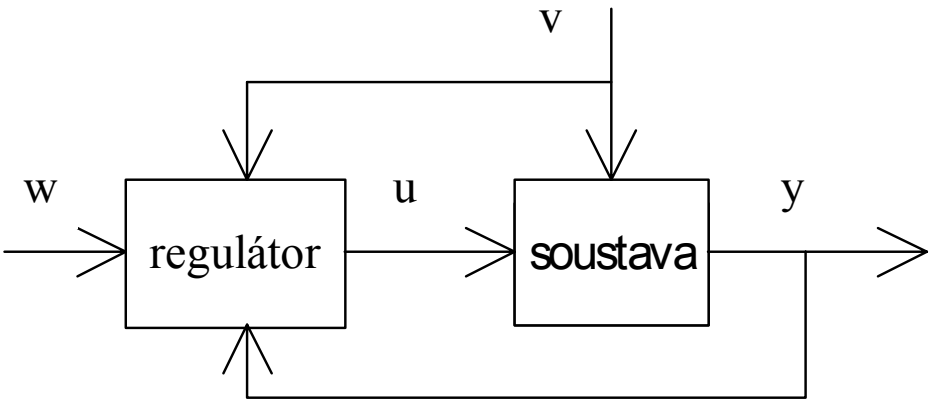
4. Teoretický rozbor regulátorů

4.1. Úvod

Do dnešního dne (21.11.1994) nebyl vymyšlen a odzkoušen univerzální algoritmus, který by mohl regulovat libovolný systém a mohl ho používat i ten, kdo s regulací nepřišel do styku. Z tohoto důvodu je nasazení jednotlivých regulátorů plně necháno na uživateli. Jeho schopnosti a zkušenosti se promítnou do kvality nastavení konstant regulátorů a kvality regulačního procesu. V následujících odstavcích jsou popsány implementované typy regulátorů.

4.2. Regulační systém

Schéma regulačního obvodu, které budeme používat je na obrázku:



Regulátor pracuje s pevně danou periodou vzorkování Ts a generuje posloupnost čísel {u(k); k=1,2,...}. Touto posloupností je ovládán akční orgán, který je chápán jako součást řízené soustavy, včetně D/A převodníku. Řízeným procesem z hlediska regulátoru je posloupnost čísel {y(k); k=1,2,...}, kterou

regulátor získává vzorkováním spojitého výstupu rovněž s periodou Ts. Většinou je nezbytné provádět před vzorkováním filtraci měřené výstupní hodnoty, přitom případný filtr spolu s A/D převodníkem se rovněž považují za část řízené soustavy. Úkolem regulátoru je pak zajistit, aby každý vzorek y(k) byl v určitém smyslu co nejblíže žádané hodnotě w(k), která se předpokládá rovněž jako posloupnost čísel s periodou Ts {w(k); k=1,2,...}.

Základní zpětnovazební obvod může být doplněn dopřednou vazbou od externí poruchy a pokud je přípustné měření, je její průběh vzorkován s periodou Ts a předáván regulátoru jako posloupnost čísel {v(k); k=1,2,...}.

Pro uvažovaný regulační obvod bývá ještě zaváděna tzv. odchylka e, pro kterou platí e(k)=w(k)-y(k).

Pro uvedené algoritmy je důležitý časový sled naměřených vzorků. Pro výpočet akčního zásahu u(k), budeme moci používat všechny vzorky s indexem (k-1) a nižším, tj. y(k-1),v(k-1),u(k-1) atd. Jednotlivé vzorky nemusí (ani nemohou) být odebírané synchronně. Mezi odebráním vzorku u(k) a y(k) uplyne doba εTs (kde ε je v intervalu (0,1)). Obecně lze říci, že nejvhodnější je získávat před vlastním výpočtem zásahu co nejčerstvější informaci, tj. aby použitý vzorek y(k-1) pro výpočet u(k) byl vzorkován těsně před vložením zásahu u(k). To je ale omezeno potřebnou dobou výpočtu Tc.

4.3. Regulátory

4.3.1. PID

Ve standardním přírůstkovém tvaru regulačního zákona PID je výstupní veličina regulátoru u(k) vyjádřena vztahem:

$$u(k) = Kc\left(e(k) + \frac{Ts}{Ti}\sum_{i=0}^{k} e(i) + \frac{Td}{Ts}\left(e(k) - e(k-1)\right)\right)$$

kde

e(k)	je regulační odchylka v normovaném tvaru
Kc	je zesílení [-]
Ts	je perioda vzorkování [s]
Td	je derivační konstanta [s]
Ti	je integrační konstanta [s]

Regulační odchylka se používá v normovaném tvaru:

$$e(k) = \frac{w(k) - y(k)}{y\max - y\min}.100 \qquad [\%]$$

V programu se k výpočtu akčního zásahu používá vztah:

$$u(k) = Kc.e(k) + Ki.\sum_{i=0}^{k} e(i) + Kd.\left(e(k) - e(k-1)\right)$$

kde

Kp	je zesílení proporcionální složky [-]
Ki	je zesílení integrační složky [-]

Kd je zesílení derivační složžky [-]

Dalším možným popisem regulačních konstant je tvar:

$$u(k) = \frac{100}{PBnd}\left(e(k) + \frac{Ts}{Ti}\sum_{i=0}^{k} e(i) + \frac{Td}{Ts}\left(e(k) - e(k-1)\right)\right)$$

kde

PBnd je šířka pásma proporcionality v [%]

Služba knihovny pro zadávání parametrů je napsána tak, aby bylo možné použít pro zadání konstant regulátoru jeden z uvedených tvarů konstant. Který z tvarů budeme používat určíme nastavením proměnné KMod na:

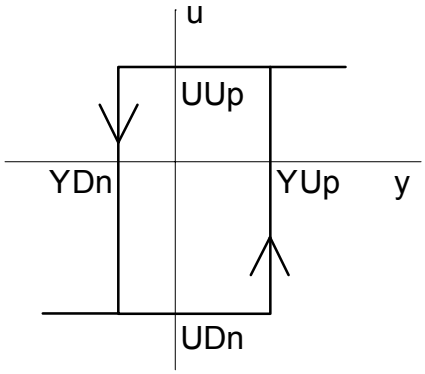
ISA	pro konstanty Kc, Ti, Td
SofConK	pro konstanty Kp, Ki, Kd
SofConP	pro konstanty PBnd, Ti, Td

Nelineární parametr "antireset window up" je uplatněn na integrační a derivační složku. Je-li odchylka větší než povolená hodnota konstant IGap, DGap v [%], pak je zastavena integrace respektive derivace. Dále je možné zadat pásmo necitlivosti EGap, kde se vypíná výpočet nových zásahů, je-li odchylka menší ne EGap [%]. Regulátor umožňuje pomocí konstanty RelK nastavit jiné pásmo proporcionality pro zápornou regulační odchylku a to tak, že Kminus=Kplus/RelK.

V regulátoru je též implementováno beznárazové přepínaní při přechodu man/auto.

4.3.2. P2P

Regulátor P2P (Position2Position) je jednoduchý dvoupolohový regulátor s hysterezí. Prahy pro přepínaní se zadávají pomocí poloh na ose vstupní veličiny. Přesná funkce regulátoru a význam jednotlivých parametrů je patrný z obrázku.



Implementace regulátoru obsahuje paměť, která zajišťuje zapamatování si stavu, který se podle hodnoty y mění. Všechny hodnoty se zadávají absolutně a regulátor neobsahuje žádná další omezení či filtrace vstupu y. Jelikož se jedná o nespojitý dvoupolohový regulátor, je přechod manuál-auto realizován následovně: při inicializaci je nastavena pozice do polohy Up, od té doby je regulátor spuštěn a provádí výpočet akčního zásahu vždy, pokud je nastaven na manuální provoz, je na

místo vypočtené hodnoty dosazena hodnota manuální, při přepnutí regulátoru do polohy auto se tedy pouze neprovádí koncový přepis vypočtené hodnoty a regulátor reguluje plně sám.

5. 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.

```
cNamePID  = 'PID';
```

Konstanta definuje jméno regulačního objektu tPID.

```
cNameP2P  = 'P2P';
```

Konstanta definuje jméno regulačního objektu tP2P.

```
pPID = ^tPID;
```

Typ ukazatel na regulační objekt PID.

```
pAddPID = ^tAddPID;
```

Typ ukazatel na seznam objektů regulačních jednotek PID.

```
tKonMode = (ISA,SofConK,SofConP);
```

Režimy zadávání konstant regulátoru.

```
pP2P = ^tP2P;
```

Typ ukazatel na regulační objekt P2P.

```
pAddP2P = ^tAddP2P;
```

Typ ukazatel na seznam objektů regulačních jednotek P2P.

```
tPozition = (Up,Dn);
```

Typ pozice P2P regulátoru.

6. Objekty

6.1. tPID

6.1.1. Pole

```
KMod    : tKonMode;
```

Proměnná **Kmod** určuje, zda regulátor nastavujeme pomocí konstant v normě ISA (Kc,Ti,Td), SofConK(Kp,Ki,Kd) nebo SofConP(PBnd,Ti,Td).

```
Kp    : RealP;
```

Proměnná **Kp** určuje konstantu proporcionální složky v normě SofConK.

```
Ki    : RealP;
```

Proměnná **Ki** určuje konstantu integrační složky v normě SofConK.

```
Kd    : RealP;
```

Proměnná **Kd** určuje konstantu derivační složky v normě SofConK.

```
Kc    : RealP;
```

Proměnná **Kc** určuje velikost Proporcionální složky v normě ISA.

```
Ti    : RealP;
```

Proměnná **Ti** určuje velikost Integrační časové konstanty v normě ISA a SofConP.

```
Td    : RealP;
```

Proměnná **Td** určuje velikost Derivační časové konstanty v normě ISA a SofConP.

```
PBnd  : RealP;
```

Proměnná **PBnd** určuje šířku pásma proporcionality v normě SofConP.

```
RelK  : RealP;
```

Proměnná **RelK** umožňuje nastavit pro zápornou regulační odchylku jiné konstanty než-li pro kladnou regulační odchylku.

```
MinU  : RealP;
```

Proměnná **MinU** určuje minimální velikost akčního zásahu.

```
MaxU  : RealP;
```

Proměnná **MaxU** určuje maximální velikost akčního zásahu.

```
MinY  : RealP;
```

Proměnná **MinY** určuje minimální velikost měřené veličiny.

```
MaxY  : RealP;
```

Proměnná **MaxY** určuje mmaximální velikost měřené veličiny.

```
EGap  : RealP;
```

Proměnná **EGap** určuje velikost pásma necitlivosti, pro které se regulační odchylka nuluje tj. je-li |ee| < EGap => ee=0.

```
IGap  : RealP;
```

Proměnná **IGap** určuje velikost omezení integrace tj. je-li |ee| > IGap je zastavena integrace.

```
DGap  : RealP;
```

Proměnná **DGap** určuje velikost omezení derivace tj. je-li |ee| > DGap je derivační složka nulová.

```
SumI0 : RealP;
```

Proměnná **SumIO** udává implicitní hodnota SumI při startu a při Ki=0.

```
KRp   : RealP;
```

Proměnná **KRp** udává P-složku regulátoru.

```
KRi   : RealP;
```

Proměnná **KRi** udává I-složku regulátoru.

```
KRd   : RealP;
```

Proměnná **KRd** udává D-složku regulátoru.

```
KRpm  : RealP;
```

Proměnná **KRpm** udává P-složku regulátoru pro ee<0.

```
KRim  : RealP;
```

Proměnná **KRim** udává I-složku regulátoru pro ee<0.

```
KRdm  : RealP;
```

Proměnná **KRdm** udává D-složku regulátoru pro ee<0.

```
SumI  : RealP;
```

Proměnná **SumI** udává součet regulačních odchylek*KRi, tj. integrační složku.

```
eLast : RealP;
```

Proměnná **eLast** udává poslední regulační odchylku.

6.1.2. Metody

6.1.2.1. Init constructor

```
constructor Init
```

Konstruktor **Init** slouží k vytvoření instance regulačního objektu. Po jeho volání jsou inicializovány proměnné objektu a je možné volat metody pro nastavení parametrů regulátoru a metodu ControlAlg realizující výpočet akčního zásahu. Nejdříve je zavolána metoda Init objektu tVirtSISI a poté jsou nově přidané proměnné objektu tPID inicializovány následovně :

```
  Name      := cNamePID;
  KMod      := SofConK;
  Kp        := 0;
  Ki        := 0;
  Kd        := 0;
  Kc        := 0;
  Ti        := 0;
  Td        := 0;
  PBnd      := 100;
  RelK      := 1;
  Ts        := 1;
  KRp       := 0;
  KRi       := 0;
  KRd       := 0;
  KRpm      := 0;
  KRim      := 0;
  KRdm      := 0;
  MinU      := -MaxInt;
  MaxU      := MaxInt;
  MinY      := -MaxInt;
  MaxY      := MaxInt;
  EGap      := 0;
  IGap      := 100;
  DGap      := 100;
  SumI0     := 0;
  SumI      := 0;
  eLast     := 0;
  Result    := Res_OK;
```

6.1.2.2. ControlAlg function

```
function ControlAlg(W : RealP; Y : RealP; V : RealP):RealP;virtual;
```

Funkce provede jeden krok regulátoru, tedy načte požadovanou hodnotu **W**, výstupní hodnotu **Y** a rušení **V**, spočítá odchylku **E** a na základě algoritmu PID regulátoru spočítá akční zásah **U**, který je zároveň výstupní hodnotou funkce.

6.1.2.3. SetParam procedure

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

Metoda **SetParam** slouží k novému nastavení parametrů regulačního algoritmu. Není třeba definovat pokaždé hodnoty všech parametrů, ale stačí uvést jen ty parametry, kterých se změna bude týkat. Jsou-li regulátory zřetězeny, jsou parametry jednotlivých knihoven odděleny znakem | a jednoznačně určeny jménem. Jsou-li parametry určeny i pro jiné zřetězené knihovny, jsou jim parametry předány voláním metod **SetParam**.

Přehled jednotlivých parametrů:

```
NAM=PID
```

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

`AM=aaa`

Parametr určující zda je regulátor ve stavu regulace automatická tj. na soustavu působí regulačním algoritmem vypočtený akční zásah, nebo ve stavu regulace manuální-ruční tj. na soustavu působí akční zásah daný hodnotou proměnné UMan.

`W=bbb`

Parametr určující velikost požadované hodnoty.

`KMD=ccc`

Parametr určující která norma konstant se bude akceptovat při regulaci.

`KP=ddd`

Parametr určující velikost proporcionální složky.

`KI=eee`

Parametr určující velikost integrační složky.

`KD=fff`

Parametr určující velikost derivační složky.

`KC=ggg`

Parametr určující velikost zesílení regulátoru.

`TI=hhh`

Parametr určující velikost integrační časové konstanty.

`TD=iii`

Parametr určující velikost derivační časové konstanty.

`PBD=jjj`

Parametr určující šířku pásma proporcionality.

`REK=kkk`

Parametr určující relativní zesílení pro zápornou regulační odchylku.

`TS=lll`

Parametr určující velikost periody vzorkování.

`MNU=mmm`

Parametr určující minimální velikost akčního zásahu.

`MXU=nnn`

Parametr určující maximální velikost akčního zásahu.

`MNY=ooo`

Parametr určující minimální velikost měřené veličiny.

`MXY=ppp`

Parametr určující maximální velikost měrené veličiny.

`UMN=qqq`

Parametr určující velikost manuálního akčního zásahu.

`EGP=rrr`

Parametr určující velikost pásma necitlivosti regulátoru.

`IGP=sss`

Parametr určující velikost pásma necitlivosti integrační složky.

`DGP=ttt`

Parametr určující velikost pásma necitlivosti derivační složky.

Příklad:

Daný příklad ukazuje jak nastavit regulátor do regulace automatické, s normou konstant SofConK, Kp=10, Ki=2,Kd=0, maximální resp. minimální hodnoty Y a U na 100 resp. -100, manuální výstup na 10 a požadovanou hodnotu na 20 .

```
SetParam('NAM=PID AM=RegAut W=20 KMd=SofConK Kp=10, Ki=2, Kd=0,
MxY=100, MnY=-100, MxU=100, MnU=-100, uMn=10');
```

6.1.2.4. GetParam function

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

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

6.1.2.5. SetUMan procedure

```
procedure tPID.SetUMan(Val:RealP);
```

Procedura SetUMan nastaví velikost **UMan** na honodtu **Val**.

6.1.2.6. SetAuto

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

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

6.2. tAddPID

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

6.2.1. Metody

6.2.1.1. Init constructor

```
constructor Init
```

Konstruktor **Init** slouží k vytvoření instance seznamu regulačních objektů. Nejdříve je zavolána metoda Init objektu tAddVirtSISO a poté je nově přidaný objekt tAddPID inicializován:

```
  Name      := cNamePID;
```

6.2.1.2. SISOInit function

```
function SISOInit: PVirtSISO;
```

Funkce vytvoří instanci daného regulačního objektu tPID.

6.3. tP2P

6.3.1. Pole

```
UUp   : RealP;
```

Proměnná **UUp** určuje velikost akčního zásahu při Y>=YUp.

```
UDn   : RealP;
```

Proměnná **UDn** určuje velikost akčního zásahu při Y<=YDn.

```
YUp   : RealP;
```

Proměnná **YUp** určuje horní mez přepnutí regulátoru.

```
YDn   : RealP;
```

Proměnná **YDn** určuje dolní mez přepnutí regulátoru.

6.3.2. Metody

6.3.2.1. Init constructor

```
constructor Init
```

Konstruktor **Init** slouží k vytvoření instance regulačního objektu. Po jeho volání jsou inicializovány proměnné objektu a je možné volat metody pro nastavení parametrů regulátoru a metodu ControlAlg realizující výpočet akčního zásahu. Nejdříve je zavolána metoda Init objektu tVirtSISI a poté jsou nově přidané proměnné objektu tPID inicializovány následovně :

```
  Name      := cNamePID;
  UUp       := 0;
  UDn       := 0;
  YUp       := 0;
  YDn       := 0;
```

```
  Result    := Res_OK;
```

6.3.2.2. SetParam procedure

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

Metoda **SetParam** slouží k novému nastavení parametrů regulačního algoritmu. Není třeba definovat pokaždé hodnoty všech parametrů, ale stačí uvést jen ty parametry, kterých se změna bude týkat. Jsou-li regulátory zřetězeny, jsou parametry jednotlivých knihoven odděleny znakem | a jednoznačně určeny jménem. Jsou-li parametry určeny i pro jiné zřetězené knihovny, jsou jim parametry předány voláním metod **SetParam**.

Přehled jednotlivých parametrů:

`UUp : RealP;`
Proměnná **UUp** určuje velikost akčního zásahu při Y>=YUp.

`UDn : RealP;`
Proměnná **UDn** určuje velikost akčního zásahu při Y<=YDn.

`YUp : RealP;`
Proměnná **YUp** určuje horní mez přepnutí regulátoru.

`YDn : RealP;`
Proměnná **YDn** určuje dolní mez přepnutí regulátoru.

`NAM=P2P`
Parametr určující jméno regulátoru kterému jsou parametry určeny, parametr musí být určen vždy a musí být uveden jako první.

`AM=aaa`
Parametr určující zda je regulátor ve stavu regulace automatická tj. na soustavu působí regulačním algoritmem vypočtený akční zásah, nebo ve stavu regulace manuální-ruční tj. na soustavu působí akční zásah daný hodnotou proměnné UMan.

`YUP=bbb`
Parametr určující horní mez přepnutí regulátoru.

`YDN=ccc`
Parametr určující dolní mez přepnutí regulátoru.

`UUP=ddd`
Parametr určující velikost akčního zásahu při Y>=YUp.

`UDN=eee`
Parametr určující velikost akčního zásahu při Y<=YDn.

`UMN=fff`
Parametr určující velikost manuálního akčního zásahu.

Příklad:
Následující příklad ukazuje jak nastavit regulátor do regulace automatické, manuální výstup na 10, mez přepínání Y na -10 resp. 10 a tomu odpovídající hodnotu výstupu U na 20 resp. -20.

```
SetParam('NAM=P2P AM=RegAut UMn=10 UUp=-20 UDn=20 YUp=10 YDn=-10');
```

6.3.2.3. GetParam function

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

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

6.4. tAddP2P

Objekt **tAddP2P** slouží k vytvoření seznamu objektů regulačních jednotek P2P, které můžeme v dané aplikaci využívat. Je rovněž schopen vygenerovat instanci daného regulačního objektu.

6.4.1. Metody

6.4.1.1. Init constructor

```
constructor Init
```

Konstruktor **Init** slouží k vytvoření instance seznamu regulačních objektů. Nejdříve je zavolána metoda Init objektu tAddVirtSISO a poté je nově přidaný objekt tAddPID inicializován:

```
  Name      := cNameP2P;
```

6.4.1.2. SISOInit function

```
function SISOInit: PVirtSISO;
```

Funkce vytvoří instanci daného regulačního objektu tP2P.

6.5. tPID2

Regulátor PID2 je klasicky PID regulátor se dvěma bankami konstant, který je navíc doplněny řadou nelineárních omezeni. Objekt regulátoru tPID2 obsahuje stejné metody jako objekt regulátoru tPID. Tento regulátor se sice nachází v interface sekci jednotky, ale ještě není zařazen v podporovaných (a zdokumentovaných) typech regulátorů.

7. Příklad

Vzhledem k tomu, že knihovna RegSISO obsahuje objekty určené k použití s objektem ProcSISO je konkrétní příklad uveden v popisu knihovny tohoto objektu, v manuálu ProcSISO.