# VirtSISO

## JEDNOTKA OBSAHUJÍCÍ RODIČOVSKÝ OBJEKT 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 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.Popis konstant 6
5.Popis typů 6
6.Proměnné 7
7.Objekty 7
7.1. tVirtSISO 7
7.1.1. Pole 7
7.1.2. Metody 8
7.1.2.1. Init constructor 8
7.1.2.2. Done destructor 8
7.1.2.3. ControlAlg 8
7.1.2.4. SetParam procedure 8
7.1.2.5. GetParam function 9
7.1.2.6. SetUMan procedure 9
7.1.2.7. SetAuto 9
7.2. tAddVirtSISO 9
7.2.1. Metody 9
7.2.1.1. Init constructor 9
7.2.1.2. SISOInit function 9
8.Příklad 9

# 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.<br>Doplněný popis parametru tVirtSISO.Ts.<br>Doplněné popisy metod tAddVirtSISO.SISOInit a tAddVIRTSISO.SISOInit. |
| | | | | |

## 1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky obsahující rodičovský objekt regulačních objektů.

## 1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

## 1.4. Související dokumenty

Pro čtení tohoto dokumentu není potřeba číst žádný další manuál, ale je potřeba orientovat se v používání programového vybavení SofCon.

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

Jednotka VirtSISO byla vytvořena s cílem sjednocení vzhledu všech regulačních jednotek. Byl vytvořen základní rodičovský objekt **tVirtSISO** pro tvorbu konkrétních regulačních jednotek. V tomto objektu jsou definovány metody, jejich názvy a význam pro konkrétní regulátory. Dědicové objektu vytvoří jen metody, které jsou specifické pro daný typ regulátoru. Tímto je vytvořeno jednotné rozhraní pro všechny typy regulátorů a uživatel může psát aplikační program bez ohledu na použitý regulační algoritmus. Knihovna rovněž definuje rodičovský objekt **tAddVirtSISO**, který zajistí, aby daná regulační knihovna byla k aplikaci připojena a byla v aplikaci k dispozici pro případné využití. Určení, která regulační knihovna se bude používat je voleno až parametrem nastavovací metody. Proto je v paměti udržován seznam všech objektů regulačních jednotek, z kterých po zavolání nastavovací metody vznikne daný regulátor. Instance požadovaných regulačních objektů jsou vytvářeny až za chodu aplikačního programu. 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. Popis konstant

```
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  = 'VSISO';
```

Konstanta definuje jméno komunikačního objektu tVirtSISO.

```
Res_Ok   = $0000;
Res_Err  = $FFFF;
```

Konstanty definující základní výsledky po provedení operací regulátoru. Výsledek **Res_Ok** znamená, že operace proběhla bez závad, výsledek **Res_Err** znamená, že operace skončila chybou.

```
ConstTs:RealP=18.18;
```

Konstanta definující počet tiků od přerušení v systému RETOS za sekundu. Jen pro vnitřní použití.

# 5. Popis typů

```
{$IFOPT N+}
  RealP = single;
{$ELSE}
  RealP = real;
{$ENDIF}
```

Je-li přítomen koprocesor, pracuje se s čísly typu single, jinak s čísly typu real.

```
  tNameStr  = String[10];
```

Pro zadaní jména regulačního objektu použit řetězec délky 10 znaků.

```
tRegMode   = (RegAut,RegMan);
```

Regulátor může být pouze v jednom z následujících stavů, regulace automatická nebo regulace manuální.

```
pVirtSISO = ^tVirtSISO;
```
Typ ukazatel na regulační objekt.

```
pAddVirtSISO = ^tAddVirtSISO;
```
Typ ukazatel na seznam objektů regulačních jednotek.

# 6. Proměnné

```
SISOCollection : PCollection=nil;
```
Ukazatel na seznam regulačních objektů. Do tohoto seznamu jsou vkládány určené regulační objekty a jeho pomocí jsou vytvářeny instance daného regulátoru.

# 7. Objekty

## 7.1. tVirtSISO

### 7.1.1. Pole

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

```
Ts      : RealP;
```
Proměnná **Ts** určuje periodu vzorkování regulátoru.

```
ww      : RealP;
```
Proměnná **ww** určuje žádanou hodnotu na výstupu regulované soustavy.

```
yy      : RealP;
```
Proměnná **yy** je naměřená hodnota na výstupu regulované soustavy.

```
vv      : RealP;
```
Proměnná **vv** je naměřená hodnota poruch působících na regulovanou soustavu.

```
uu      : RealP;
```
Proměnná **uu** určuje akční zásah do regulované soustavy.

```
AM      : tRegMode;
```
Proměnná **AM** určuje 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.

```
UMan    : RealP;
```

Proměnná **UMan** určuje hodnotu akčního zásahu do regulované soustavy pokud je proměnná AM nastavena na hodnotu RegMan .

```
Result  : Word;
```

Do proměnné **Result** je uložen výsledek naposledy volané operace regulačního objektu.

## 7.1.2. Metody

### 7.1.2.1. 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 nastavování parametrů regulátoru a metodu ControlAlg realizující výpočet akčního zásahu. Proměné objektu jsou inicializovány následovně :

```
Name   := cName;
yy     := 0;
vv     := 0;
uu     := 0;
ww     := 0;
UMan   := 0;
AM     := RegMan;
Result := Res_OK;
```

### 7.1.2.2. Done destructor

```
destructor  Done; virtual;
```

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

### 7.1.2.3. ControlAlg

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

Tato funkce provede výpočet akčního zásahu do regulované soustavy, a vrátí ho ve formě své funkční hodnoty. Parametr W je žádaná hodnota, parametr Y skutečná hodnota a parametr V naměřené poruchy.

### 7.1.2.4. 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**.

Jelikož je daný objekt pouze sjednocením názvů metod a proměnných a vlastní implementace je ponechána na dědice, odkazuji na příklad v popisu knihovny RegSISO, která je jedním z dědiců tohoto objektu.

#### 7.1.2.5. 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.

#### 7.1.2.6. SetUMan procedure

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

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

#### 7.1.2.7. 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.

## 7.2. tAddVirtSISO

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

### 7.2.1. Metody

#### 7.2.1.1. Init constructor

```
constructor Init
```

Konstruktor **Init** slouží k vytvoření instance seznamu regulačních objektů. Zároveň se provede inicializace:

```
  Name      := cNamePID;
```

#### 7.2.1.2. SISOInit function

```
function SISOInit: PVirtSISO;
```

Funkce vytvoří instanci daného regulačního objektu.

# 8. Příklad

Jelikož je daný objekt pouze sjednocením názvů metod a proměnných a vlastní implementace je ponechána na dědice, odkazuji na příklad v popisu knihovny RegSISO, která je jedním z dědiců tohoto objektu.