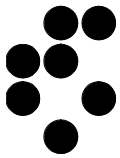
Univerza v Ljubljani

Institut "Jožef Stefan", Ljubljana, Slovenija



IJS Delovno poročilo
DP-6812

PROGRAMSKI MODUL CONTROL

Damir Vrančić
Janko Petrovčič
Đani Juričić

KAZALO

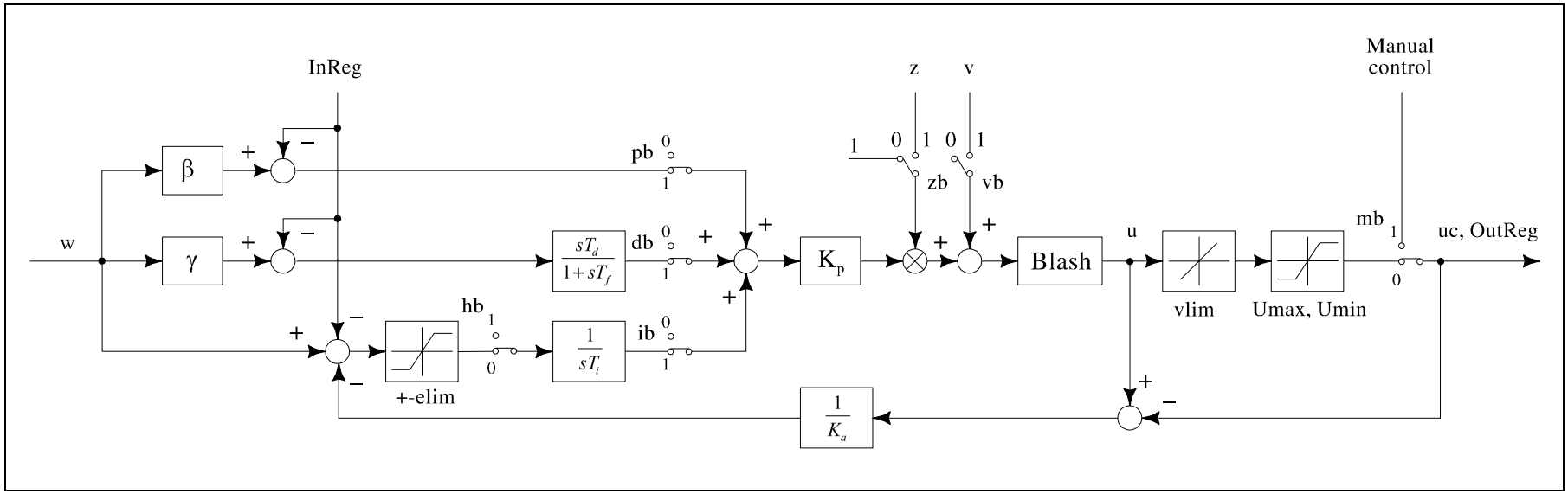
1. UVOD

Pri raziskovalnem delu na področju vodenja sistemov pogosto uporabljamo simulacijska orodja (programske pakete) za preverjanje delovanja regulacijskih algoritmov. Simulacijska orodja nam lahko precej pomagajo pri razumevanju sistema ter določanju regulacijske sheme z okvirnimi vrednostmi konstant, vendar pogosto nastopijo težave pri implementaciji algoritma na realen proces zaradi pojavov na procesu, ki jih je težko simulirati. Težave nastanejo zaradi časovne spremenljivosti procesa (ireverzibilne spremembe), odvisnosti od drugih fizikalnih veličin (temperatura, vlaga), ter pogostih motenj znotraj sistema. Pri dejanski realizaciji vodenja pa prihaja do dodatnih težav tudi zaradi implementacije industrijskih regulatorjev, ki se po delovanju pogosto razlikujejo od simuliranih znotraj programskih paketov. Algoritmi, ki tečejo znotraj realnih regulatorjev, uporabljajo pogosto dodatne funkcije, hkrati pa delujejo v realnem času.
Zaradi težav, ki nastopajo pri prenosu znanja iz teorije v prakso, smo na našem odseku (E-2) razvili multivariabilno in nestabilno modelno napravo, ki služita študiji problemov, ki se v praksi pogosto pojavljajo. Procesa smo vodili z, na odseku razvitim, večzančnim mikroračunalniškim regulatorjem MMC-90 in s simulacijskim programskim paketom Simcos z merilno-regulacijsko opremo Burr Brown.
Pri obeh uporabljenih regulatorjih smo naleteli na določene težave. MMC-90 se je izkazal za premalo prilagodljivega s stališča študija regulacijskih algoritmov in glede hitrosti pri vodenju nestabilnega procesa. Programski paket Simcos je prav tako imel relativno omejeno hitrost delovanja, hkrati pa je vsaka sprememba v programu zahtevala relativno dolgotrajno prevajanje (fortran) s precej kompleksno izhodno kodo.
Glede na težave, na katere smo naleteli pri uporabi omenjenih regulatorjev, smo se odločili, da napišemo program (modul), ki bi dokaj reprezentativno predstavljal (pokrival) precejšnje število v industriji uporabljenih regulatorjev, hkrati pa naj bi bil imel dokaj hiter prevajalnik. Program (modul) naj bi omogočal tudi relativno lahko dodajanje novih funkcij (adaptivna kontrola, komunikacija z višjim, nadzornim sistemom, ...) in enostavno uporabo za sodelavce odseka in študente.
Glede na podane zahteve smo se odločili za programski jezik Turbo Pascal, ki je dokaj uporabljan med študenti elektrotehnike, hkrati pa omogoča hitrejše prevajanje in izvajanje prevedene kode od programskega paketa Simcos. Izdelali smo modul *Control*, ki nam poleg regulacijske funkcije splošnega PID regulatorja omogoča tudi selektiven vklop vhodnega filtra in popolnoma brezudaren preklop iz ročnega v avtomatski način delovanja. Z uporabo modula dosežemo tudi enostaven vnos parametrov regulatorja tako preko tipkovnice, kot tudi preko datoteke.

2. REGULACIJSKA SHEMA

Programski modul Control (procedura Controller) izvaja regulacijsko funkcijo, ki je shematično prikazana na sliki 1. Vhod v regulator je označen z InReg, referenca s črko w, izhod iz regulatorja pa z OutReg. S konstantama β (beta) in γ (gamma) določamo *regulacijsko strukturo* regulatorja (vezanost proporcionalnega in diferencialnega dela regulatorja na signal reference), signal z (v primeru, če je stikalo zb=1), nam služi kot *multiplikativni vhod* v regulator, signal v (pri stikalu vb=1) pa nam služi kot *aditivni vhod* v regulator. S stikalom mb preklapljamo med *ročnim in avtomatskim* delovanjem, povratni signal s hitrostnega (vlim) in amplitudnega (Umax in Umin) omejevalnika pa vodimo preko konstante 1/Ka nazaj na integracijski člen, kar je v bistvu algoritem za *preprečevanje integracijskega pobega*. Tip regulatorja (PID, PI, PD, P ali I) izbiramo s položajem stikal pb, db in ib. Pred vhodom v integrator (1/sTi) imamo omejevalnik amplitude (elim), ki nam služi za *zmanjševanje prevzpona* na izhodu iz reguliranega procesa (tako kot konstanti beta in gamma), za njim pa stoji stikalo hb, ki predstavlja *hold* funkcijo integratorja (zaustavitev integracije). Pred izhodnimi limiterji regulatorja je še funkcija *Blash*, ki nam pomaga voditi procese ki vsebujejo *histerezo* (slabi ležaji, obrabljeni zobniki ali slabo priviti vijaki loput in ventilov).

Spremenljivke označene na sliki 1 nastopajo z enakimi imeni tudi v programskem modulu Control. Bolj detajlna razlaga pomena posameznih spremenljivk sledi v naslednjem poglavju.



Slika 1: Shematičen prikaz implementiranega regulatorja

3. OPIS PROGRAMSKEGA MODULA CONTROL

V tem poglavju bomo najprej pojasnili pomen posameznih spremenljivk, ki nastopajo v modulu Control, nato bo sledil kratek opis delovanja procedur in funkcij modula, na koncu pa še prikaz uporabe modula v glavnem programu, kot tudi rezultati vodenja na modelni napravi.

3.1. POMEN SPREMENLJIVK

Spremenljivke, ki so uporabljene v modulu control, lahko razdelimo v dve skupini. V prvi skupini so *splošne spremenljivke in konstante regulatorjev*, ki veljajo za vse regulatorje. Ta tip spremenljivk ni direktno vezan na posamezen regulator (npr. Ts pomeni čas vzorčenja in je enak za vse regulatorje). V drugi skupini pa so *posamezne regulatorske spremenljivke*, ki so (v večini primerov) v obliki vektorja. Prvi člen vektorja se tako nanaša na prvi regulator, zadnji člen pa na zadnjega (npr. Kp[1] pomeni proporcionalno ojačenje prvega regulatorja, Td[3] pa diferencialno časovno konstanto tretjega regulatorja).

3.1.1. SPLOŠNE SPREMENLJIVKE IN KONSTANTE REGULATORJEV

Ime spremenljivke	Tip	Pomen	Predef. vrednost
Max_Reg	integer	Maksimalno število regulatorjev	9
n_reg	integer	Dejansko število implementiranih regulatorjev	1
Active_reg	integer	Indeks aktivnega regulatorja (na katerega lahko vplivamo preko tipkovnice)	1
All_regb	boolean	TRUE - Vsi regulatorji imajo enake parametre FALSE - Regulatorji imajo različne parametre	TRUE
Init_Constb	boolean	TRUE - Regulator ima predefinirane vrednosti konstant FALSE - Regulatorju vnašamo vrednosti konstant	FALSE
Ts	real	Čas vzorčenja	0.1
Dim_array	integer	Velikost polja vhodnega filtra	11

3.1.2. POSAMEZNE REGULATORSKE SPREMENLJIVKE

Posamezne regulatorske spremenljivke lahko po njihovih funkcijah razdelimo na več podskupin:

- osnovne konstante regulatorjev
- limite
- določanje modela regulatorjev in zmanjševanje prevzpona
- preprečevanje integralskega pobega

- dodatni vhodi regulatorjev
- histereza
- vhodni filtri
- brezudarni avtomatičen preklop ročno/avtomatsko
- ostale nastavitve regulatorjev

3.1.2.1. Osnovne konstante regulatorjev

Ime spremenljivke	Tip	Pomen	Predef. vrednost
pb	bool_array	TRUE/FALSE - Vklop/izklop proporcionalnega člena	TRUE
ib	bool_array	TRUE/FALSE - Vklop/izklop integracijskega člena	TRUE
db	bool_array	TRUE/FALSE - Vklop/izklop diferencialnega člena	TRUE
Kp	real_array	Proporcionalno ojačenje	1
Ti	real_array	Integracijska časovna konstanta	1e6
Td	real_array	Diferencialna časovna konstanta	0
Tf	real_array	Časovna konstanta filtra diferencialnega člena: $D(s) = \frac{sT_d}{1+sT_f}$	Td/10
K_Tf	real_array	Uporaba v algoritmu računanja filtra: $K\_Tf = e^{-\frac{T_s}{T_f}}$	
w	real_array	Velikost signala reference	1

3.1.2.2. Limite

Ime spremenljivke	Tip	Pomen	Predef. vrednost
Umax	real_array	Maksimalna vrednost izhodne napetosti	10
Umin	real_array	Minimalna vrednost izhodne napetosti	0
Vlim	real_array	Hitrostna limita na izhodu iz regulatorja. Odvisna je od časa vzorčenja: Vlim = Max. hitrost (V/s) * Ts	1000*Ts
Incr_Manual	real_array	Sprememba izhodne napetosti ob pritisku na puščici gor/dol.	1*Ts

3.1.2.3. Določanje modela regulatorjev in zmanjševanje prevzpona

Ime spremenljivke	Tip	Pomen	Predef. vrednost

beta	real_array	Določanje povezave med referenco in proporcionalnim členom regulatorja (beta*w): beta = 0: P-člen je vezan samo na vhod v regulator (-InReg) beta = 1: P-člen je vezan na signal regulacijskega pogreška (w-InReg). Namen je tudi zmanjševanju prevzpona na izhodu iz procesa pri spremembi signala reference.	1
gamma	real_array	Določanje povezave med referenco in diferencialnim členom regulatorja (gamma*w): gamma = 0: D-člen je vezan samo na vhod v regulator (-InReg) gamma = 1: D-člen je vezan na signal regulacijskega pogreška (w-InReg). Prav tako, kakor beta, služi zmanjševanju prevzpona na izhodu iz procesa pri spremembi signala reference.	0
elim	real_array	Vrednost omejevalnika na vhodu v integracijski del regulatorja. Če je vhodni signal v omejevalnik e_{in} in izhodni signal e_{out}, potem omejevalnik deluje tako: $e_{out} = \begin{cases} e_{in} & ; -e_{\lim} < e_{in} < +e_{\lim} \\ -e_{\lim} & ; e_{in} < -e_{\lim} \\ +e_{\lim} & ; e_{in} > +e_{\lim} \end{cases}$	1e3

3.1.2.4. Preprečevanje integracijskega pobega

Ime spremenljivke	Tip	Pomen	Predef. vrednost
Ka	real_array	Pri metodi prilagajanja stanj je Ka=Kp*beta. V splošnem poljubna konstanta.	Kp*beta

3.1.2.5. Dodatni vhodi regulatorjev

Ime spremenljivke	Tip	Pomen	Predef. vrednost
vb	bool_array	TRUE/FALSE - Vklopljen/izklopljen adicijski vhod (v)	FALSE

zb	bool_array	TRUE/FALSE - Vklopljen/izklopljen multiplikativni vhod (z)	FALSE
v	real_array	Velikost adicijskega vhoda	
z	real_array	Velikost multiplikativnega vhoda	
hb	bool_array	TRUE/FALSE - Zaustavitev/normalno delovanje integratorja (Hold funkcija)	FALSE
mb	bool_array	TRUE/FALSE - Vklop/izklop ročnega režima delovanja	FALSE

3.1.2.6. Histereza

Ime spremenljivke	Tip	Pomen	Predef. vrednost
Blhyst	real_array	Velikost histereze	0
Blmin	real_array	Minimalna velikost izhodnega signala v nasprotno smer, ki povzroči dodatek histerezne napetosti (Blhyst)	1e3
Bl_Catchb	bool_array	TRUE/FALSE - Vklop/izklop iskanja temenskih vrednosti izhodne napetosti	TRUE
Bl_Ampl	real_array	Velikost temenske vrednosti izhodne napetosti	0
Bl_Direct	integ_array	Temenska vrednost pomeni: 1 = dolino -1 = hrib	1

3.1.2.7. Vhodni filtri

Ime spremenljivke	Tip	Pomen	Predef. vrednost
Filt_Automb	bool_array	TRUE/FALSE - Vklop/izklop avtomatskega vklapljanja in izklapljanja vhodnega filtra	TRUE
Filtb	bool_array	TRUE/FALSE - Vklopljen/izklopljen filter	FALSE
Filt_t	real_array	Časovna konstanta filtra (filter 1. reda)	Td/2 ali Td/10*
Filt_k	real_array	Uporaba pri računanju filtra: $Filt\_k = e^{-\frac{T_s}{Filt\_t}}$	
Filt_Diff	real_array	Služi za avtomatičen vklop/izklop vhodnega filtra. V primeru, če je razlika med minimalno in maksimalno vrednostjo polja InRegf_array večja od Filt_Diff, se vhodni filter izklopi	4.88e-3 (2 LSB)
Filt_mean_ Diff	real_array	Služi za avtomatičen vklop/izklop vhodnega filtra. Če je razlika med InReg_mean in InRegf_mean večja od Filt_mean_Diff, se vhodni filter izklopi	2.44e-3 (1 LSB)
InRegold	real_array	Stara vrednost vhodnega signala v regulator	0

InReg_array	fi_array	Polje vhodnih, nefiltriranih vrednosti. InReg_array[n,1] ... trenutna velikost vhodnega signala n-tega regulatorja. InReg_array[n,Dim_Array] ... najstarejša velikost vhod. signala n-tega regulatorja.	celo polje = 0
InRegf_array	fi_array	Podobno, kot pri InReg_array, le da velja za filtriran vhodni signal	celo polje = 0
InReg_mean	real_array	Srednja vrednost polja InReg_array (brez zadnjega člena InReg_array[Dim_array])	0
InRegf_mean	real_array	Podobno, kot pri InReg_mean, le da velja za filtriran signal	0

3.1.2.8. Brezudarni avtomatičen preklop ročno/avtomatsko

Ime spremenljivke	Tip	Pomen	Predef. vrednost
Auto_Catchb	bool_array	TRUE/FALSE - Vklop/izklop lovljenja avtomatskega režima delovanja. Če je vklopljen, čaka, da bo reguliran signal prečkal referenčno vrednost. V tem trenutku bo preklopil regulator iz ročnega v avtomatični režim	FALSE
Auto_Catch_ Direct	integ_array	1 ... InReg > w ob vklopu Auto_Catchb -1 ... InReg < w ob vklopu Auto_Catchb	-1

3.1.2.9. Ostale nastavitve regulatorjev

Ime spremenljivke	Tip	Pomen	Predef. vrednost
uold	real_array	Stara vrednost nelimitiranega izhodnega signala	0
wold	real_array	Stara vrednost signala reference	0
u	real_array	Vrednost nelimitiranega izhodnega signala	uold
uc	real_array	Vrednost limitiranega izhodnega signala	uold
x1old	real_array	Stara vrednost interne spremenljivke regulatorja (uporabljena pri filtru D-člena)	0
x2old	real_array	Stara vrednost interne spremenljivke regulatorja (uporabljena pri P-členu)	0
vold	real_array	Stara vrednost adicijskega signala	0
OutReg	real_array	Izhodna vrednost iz regulatorja	0

3.2. KRATEK OPIS DELOVANJA PROCEDUR IN FUNKCIJ

*Funkcija **Limiter*** limitira vhodno vrednost:

$$Limiter = \begin{cases} X_{in} & ; X_{\min} \le X_{in} \le X_{\max} \\ X_{\min} & ; X_{in} < X_{\min} \\ X_{\max} & ; X_{in} > X_{\max} \end{cases}$$

*Funkcija **Sgn*** je podobna matematični funkciji signum:

$$Sgn = \begin{cases} 1 & ; X > 0 \\ 0 & ; X = 0 \\ -1 & ; X < 0 \end{cases}$$

*Procedura **Shift*** premakne v desno vrednosti v vektorju Xarray (Xarray[i+1] := Xarray[i]), na prvo mesto (Xarray[1]) pa postavi vrednost Xin. Spremenljivka n podaja dolžino vektorja Xarray.

*Funkcija **Mean*** izračunava sredno vrednost vektorja Xarray, pri tem pa ne upošteva zadnjega člena (Xarray[n]). Izračunavanje se vrši po rekurzivnem postopku. Spremenljivka Xmean pomeni staro srednjo vrednost vektorja Xarray.

*Funkcija **Max*** izračuna maksimalni element vektorja Xarray.

*Funkcija **Min*** izračuna minimalni element vektorja Xarray.

*Procedura **Filter*** vhese vhodno vrednost (y) na prvo mesto vektorja InReg_array, nato vhodno spremenljivko y filtrira s filtrom 1. reda (časovna konstanta Filt_t), ter filtrirano vrednost vnese na prvo mesto vektorja InRegf_array. Procedura še izračuna nove srednje vrednosti vektorjev InReg_array in InRegf_array.

*Funkcija **Switch_filter*** je namenjena detekciji prihoda v delovno točko. Njena vrednost postane TRUE, če je razlika med srednjo vrednostjo filtriranega in nefiltriranega signala manjša od Filt_mean_Diff in, če je razlika med maksimalnim in minimalnim členom vektorja filtriranih vrednosti manjša od Filt_Diff. Če kateri od obeh pogojev ni izpolnjen, postane rezultat funkcije FALSE.

*Procedura **Status*** na osnovi kode pritisnjene tipke (Code) vklaplja oz. izklaplja ročni način delovanja (mb), določa trenutno aktivni regulator (Active_reg), ter vklaplja "lovljenje" avtomatičnega načina delovanja (Auto_Catchb).

*Funkcija **Readcode*** da na svoj izhod kodo pritisnjene tipke na tipkovnici. Razlika nastane pri pritisku smernih tipk (gor, dol, levo, desno), katerim kodam doda vrednost 200, da ne prihaja do mešanja z običajnimi ASCII znaki.

Funkcija ***Simcos*** je namenjena kompatibilnosti formata izpisa realnih veličin s programskim paketom SIMCOS. Vhod v funkcijo je realno število, ki ga funkcija priredi za SIMCOS-ov format.

Procedura ***Manual_out*** spreminja izhodno napetost regulatorja (OutReg) na osnovi pritisnjene smerne tipke (Code = 272, 280, 277 ali 275). Hkrati nastavlja tudi vrednosti Bl_Direct in Bl_Catchb, ki so pomembne pri regulaciji procesov s histerezo. Tipki levo/desno (Code = 275/277) spreminjata izhodno velikost iz regulatorja 10 krat manj kakor tipki gor/dol.

Procedura ***Out_Screen*** izriše osnovni videz zaslona regulatorja. Kliče se pred glavno regulacijsko zanko.

Procedura ***Values_Screen*** izpisuje vrednosti izbranih regulacijskih veličin (InReg, u, OutReg) na zaslon ter signalizira vklopljenost filtra in stanje ročno/avtomatsko. Kličemo jo znotraj regulacijske zanke.
Procedura ***In_Integ*** izpiše besedilo (st) na zaslon, ter zapiše naš odgovor v spremenljivko inp. V primeru, če namesto odgovora pritisnemo tipko <ENTER>, postane vrednost inp enaka vrednosti default.

Procedura ***In_Real*** je podobna proceduri In_Integ, le da sedaj vpisujemo realna števila namesto celoštevilčnih.

Procedura ***In_Boolean*** je prav tako podobna proceduri In_Integ. Namesto celoštevilčnih vrednosti vpisujemo logične spremenljivke. Dovoljeni odgovori so: 'TRUE', 'T', '1', 'Y', 'FALSE', 'F', '0' in 'N' (velja tudi za male črke).

Procedura ***In_Initial*** vnaša željeno število regulatorjev (n_reg) in informacijo, če želimo enake parametre za vse regulatorje (All_Regb).

Procedura ***In_Basic*** vnaša osnovne parametre regulatorja (pb, ib, db, Kp, Ti, Td, Tf, Ts, w). K_Tf se izračunava iz Tf.

Procedura ***In_Limits*** vnaša limite regulatorja (Umax, Umin, Vlim, Incr_manual).

Procedura ***In_Overshoot_AW*** vnaša konstante za določanje modela regulatorja, zmanjševanje prevzpona in konstanto za preprečevanje integracijskega pobega (beta, gamma, elim, Incr_manual).

Procedura ***In_Additional_Inputs*** vnaša nastavitve za dodatne vhode in kontrolo integratorja ter ročno kontrolo (vb, zb, hb in mb).

Procedura ***In_Hysteresis*** vnaša velikost histereze ter minimalno vrednost spremembe za vklop histereze (Blhyst, Blmin), postavi pa predefinirane velikosti ostalih spremenljivk (Bl_Catchb, Bl_Direct, Bl_Ampl).

Procedura ***In_Filter*** vnaša informacijo o vklopu filtra, časovni konstanti ter stari velikosti vhodnega signala (Filt_Automb, Filtb, Filt_t, InRegold). Konstanta Filt_k se izračuna iz Filt_t. Vrednosti InReg_array, InRegf_array, InReg_mean in InRegf_mean pa se nastavijo na vrednost InRegold.

Procedura ***In_Manual_Automatic*** nastavi predefinirane vredosti spremenljivk (Auto_Catchb, Auto_Catch_Direct).

Procedura ***In_Other*** nastavi predefinirane vrednosti spremenljivk (wold, uold, u, uc, x1old, x2old, vold, OutReg).

Procedura ***Extract*** nam iz niza (st) izlušči podniz (stparam), ki vsebuje ime parametra in podniz (stvalue), ki vsebuje vrednost parametra. Uporabljamo jo pri branju datoteke z vrednostmi parametrov.

Procedura ***Valb*** nam v spremenljivko result vpiše vrednost stringa st. Če je st = 'TRUE', 'T'', '1' ali 'Y', postane result = TRUE, v primeru, če je st = 'FALSE', 'F', '0' ali 'N', pa postane result = FALSE (enak rezultat tudi za male črke). V primeru napake (st ne ustreza nobeni od navedenih konstant), postane ierr = 1.

Procedura ***Get_Param*** nem s pomočjo vhodnih konstant (stparam, stvalue), dobljenih v proceduri Extract, spravi vrednost stvalue v spremenljivko, ki jo označuje stparam.

Procedura ***Read_File*** prebere datoteko (stfile) ter zapisane parametre z njihovomi vrednostmi vpiše v dejanske, programske parametre. Nekaterim parametrom, ki niso zapisani v datoteki, poda predefinirane vrednosti ali pa jih izračuna na osnovi podanih parametrov.

Procedura ***Write_File*** zapiše nekatere parametre regulatorjev v datoteko. Najprej zapiše splošne konstante, ki veljajo za vse regulatorje (n_reg, Ts), nato pa še vrednosti, ki se nanašajo na posamezen regulator.

Procedura ***Param_Copy*** kopira konstante prvega regulatorja v preostale regulatorje.

Procedura ***In_Param*** vnaša parametre regulatorja direktno preko tipkovnice ali pa preko že kreirane datoteke z vrednostmi parametrov. Če vnašamo vrednosti preko tipkovnice, nam procedura po končanem vnosu omogoča shranjevanje parametrov v datoteko.

Procedura ***Filtering*** izvede proceduro Filter, ter, če je omogočeno avtomatsko vključevanje filtra (Filt_Automb = TRUE), izvede funkcijo Switch_Filter. Če le-ta vklopi vhodni filter (Filtb = TRUE), se na vhod v regulator (InReg) postavi filtrirana vhodna vrednost, v nasprotnem postane vhod v regulator nefiltrirana vhodna vrednost (y).

Procedura ***Blash*** izvaja funkcijo, ki zmanjšuje vpliv elementov s histerezo na vhodu v reguliran proces (slabi ventili, lopute, ...). Na izhodu iz procedure se, če je potrebno, spremeni vrednost uold.

Procedura ***Controller*** vsebuje celoten regulacijski algoritem enega regulatorja. Na osnovi vhodnih veličin in notranjih spremenljivk, izračuna izhod iz regulatorja (OutReg). Regulacijski algoritem vsebuje PID regulator z možnostjo spreminjanja strukture (beta, gamma), določanja amplitudnih in hitrostnih limit, algoritma proti integracijskemu pobegu, algoritma za zmanjševanje vpliva histerez na vhodu v voden proces, algoritma za vklop vhodnega filtra, možnost ročnega vodenja, spreminjanja regulacijskih parametrov med samo regulacijo, ...

Procedura ***Regulator*** izračunava, na osnovi vhodnih veličin in notranjih spremenljivk, izhode (OutReg) vseh (n_reg) regulatorjev.

Procedure in funkcije, ki se nanašajo na komunikacijo in nastavitve Burr-Brownovega sistema (***Isys, Dsys, Sinhi, Sinh, AD, DA***), so opisane v literaturi [1]. Vsebuje jih modul Pci.

3.3. PRIKAZ UPORABE MODULA

Modul control (procedure in funkcije, ki jih vsebuje) uporabljamo znotraj glavnega programa tako, da za stavkom Program uporabimo stavek Uses Control, Pci, Crt. Modul Crt je že vgrajen pascalski modul in ga uporabljamo za kontrolo izpisa na zaslonu.

3.3.1. IZGLED GLAVNEGA PROGRAMA

Za boljše razumevanje delovanja naštetih procedur in funkcij si oglejmo dva primera glavnih programov:

```
Program Prvi;

Uses Crt,Pci,Control;


Var Kp, Ka, Ti, Td, Tf, K_Tf, beta, gamma, w, elim, Vlim,
    wold, InReg, InRegold, x1old, x3old, u, uold, uc, Umax, Umin,
    Bl_Ampl, Blmin, Blhyst, Filt_k,
    Filty_mean, Filtyf_mean, Filt_t, OutReg,
    v, vold, z, yvhod, y, Incr_manual, Filt_Diff, Filt_mean_Diff : real_array;
    Ts, t : real;
    pb, ib, db, mb, vb, zb, hb,
    Bl_Catchb, Filtb, Auto_catchb, Filt_automb : bool_array;

    Bl_Direct, Auto_catch_direct : integ_array;
    n_reg, Active_reg, i, j, k, Code : integer;
    All_Regb, Init_Constb : boolean;
    ierr : word;
    ch1, ch2 : char;
    datoteka : text;
    Filty_array, Filtyf_array : fi_array;
    st : string;


begin

  Clrscr;

            { Vnos parametrov }

  In_Param (Dim_Array, All_regb, n_reg, Active_reg, Ts, Init_constb, Incr_manual,
            Kp, Ti, Td, K_Tf, w, Umax, Umin, Vlim, beta, gamma, elim, Ka,
            Bl_Ampl, Blmin, Blhyst, Filt_k, InRegold,
            Filty_mean, Filtyf_mean, wold, uold, u, uc, x1old,
            x3old, vold, Tf, Filt_t, OutReg, Filt_Diff, Filt_mean_Diff,
            Bl_direct, Auto_Catch_Direct,
            pb, ib, db, vb, zb, hb, Bl_Catchb, Filtb, Filt_automb,
            Auto_catchb, mb,Filty_array, Filtyf_array);
```

```
  Isys;      { Inicializacija sistema }
  Clrscr;
  Out_Screen (n_reg);   { Osnovni izgled regulatorja }
  Sinhi (Ts);           { Cas vzorcenja sistema }
  Repeat Until Keypressed;  { Cakanje na pritisk tipke }
  t := 0;
  Repeat                    { Glavna regulacijska zanka }
    Sinh (ierr);            { Sinhronizacija s casom vzorcenja (real-time) }
    for i:= 1 to n_reg do
      y[i] := AD (i-1,1);   { Branje prvih n_reg vhodov iz A/D pretvornika }

              { Filtriranje vhodnih podatkov (vhodov) }

    Filtering (n_reg, Dim_Array, y, Filt_k, Filty_mean, Filtyf_mean,
               InReg, InRegold, Filt_Diff, Filt_mean_Diff, Filty_array,
               Filtyf_array, Filt_automb, Filtb);

    Code := Readcode;    { Branje tipkovnice }

           { Spreminjanje statusa regulatorja glede na pritisnjeno tipko }

    Status (Active_reg, Code, n_reg, Auto_catch_direct, mb,
            Auto_catchb, InReg, w);

              { Glavni regulacijski algoritem (vseh n_reg regulatorjev) }

    Regulator (Ts, n_reg, Active_reg, Code, Incr_manual, Kp, Ka, Ti, Td,
               K_Tf, beta, gamma, w, elim, Vlim, InReg,
               v, z, Blhyst, Blmin, Umin, Umax, pb, ib,
               db, vb, zb, mb, hb, Bl_Catchb, Bl_Direct,
               uc, InRegold, x1old, x3old,
               uold, u, vold, OutReg, Bl_Ampl, wold);

    For i := 1 to n_reg do
      DA (i-1, OutReg[i]);   { Izhod iz regulatorjev na D/A pretvornike }

    t := t+Ts;

                  { Zapis regulirnih in reguliranih velicin na zaslon }

    Values_Screen (Active_reg, t, InReg[i], u[i], OutReg[i], Filtb[i], mb[i]);

  Until (Code=27);  { Pritisnjena tipka <ESC> ? }
  Dsys;             { Deinicializacija sistema }

end.
```

Na začetku programa zbri{emo zaslon (*clrscr*) in vnesemo parametre regulatorja (*In_Param*) ročno ali preko datoteke. Nato inicializiramo sistem (*Isys* - Modul Pci), ponovno zbri{emo zaslon in izpi{emo osnovni izgled zaslona za n_reg regulatorjev (*Out_Screen*). S proceduro *Sinhi* (Modul Pci) nastavimo čas vzorčenja sistema, nato pa čakamo na pritisk tipke za začetek glavne regulacijske zanke. Ob pritisku na tipko, program nadaljuje izvajanje v glavni regulacijski zanki. Najprej (*Sinh* - Modul Pci) čaka na sinhronizacijski impulz (čas vzorčenja, ki je že prej nastavljen), nato pa preberemo prvih n_reg vhodov iz sistema (z A/D pretvorbo: *AD* - Modul Pci), ter filtriramo (če je potrebno) vhodno napetost (y) in jo prenesemo v vhodno regulacijsko spremenljivko (InReg). Za tem preberemo tipkovnico (*ReadCode*) in na osnovi pritisnjene tipke (*Status*) spreminjamo parametre regulatorja (npr. ročna kontrola, velikost regulirnega signala,...). Nato poženemo glavni regulacijski algoritem (*Regulator*) in rezultate pretvorbe po{ljemo na izhod sistema (D/A pretvorba: *DA* - Modul Pci). Regulirane in regulirne veličine za tem izpi{emo na zaslon (*Values_Screen*).
Glavna regulacijska zanka teče dokler ne pritisnemo tipke <ESC>. Takrat deinicializiramo sistem (*Dsys* - Modul Pci) in končamo program.
Oglejmo si {e primer programa pri katerem uporabimo {e nekatere dodatne možnosti:

```
Program Drugi;

Uses Crt,Pci,Control;


Var Kp, Ka, Ti, Td, Tf, K_Tf, beta, gamma, w, elim, Vlim,
    wold, InReg, InRegold, x1old, x3old, u, uold, uc, Umax, Umin,
    Bl_Ampl, Blmin, Blhyst, Filt_k,
    Filty_mean, Filtyf_mean, Filt_t, OutReg,
    v, vold, z, yvhod, y, Incr_manual, Filt_Diff, Filt_mean_Diff : real_array;
    Ts, t, Tkon : real;
    pb, ib, db, mb, vb, zb, hb,
    Bl_Catchb, Filtb, Auto_catchb, Filt_automb : bool_array;

    Bl_Direct, Auto_catch_direct : integ_array;
    n_reg, Active_reg, i, j, k, Code : integer;
    All_Regb, Init_Constb : boolean;
    ierr : word;
    ch1, ch2 : char;
    datoteka : text;
    Filty_array, Filtyf_array : fi_array;
    st : string;


begin

  Clrscr;
  In_Param (Dim_Array, All_regb, n_reg, Active_reg, Ts, Init_constb, Incr_manual,
            Kp, Ti, Td, K_Tf, w, Umax, Umin, Vlim, beta, gamma, elim, Ka,
            Bl_Ampl, Blmin, Blhyst, Filt_k, InRegold,
            Filty_mean, Filtyf_mean, wold, uold, u, uc, x1old,
            x3old, vold, Tf, Filt_t, OutReg, Filt_Diff, Filt_mean_Diff,
            Bl_direct, Auto_Catch_Direct,
            pb, ib, db, vb, zb, hb, Bl_Catchb, Filtb, Filt_automb,
            Auto_catchb, mb,Filty_array, Filtyf_array);

  Writeln;
  Write ('Vnesite ime izhodne datoteke rezultatov vodenja : ');
  Readln (st);
  Write ('Vnesite cas zapisovanja v sekundah : ');
  Readln (Tkon);
  assign (datoteka, st);
  rewrite (datoteka);
  Writeln (Datoteka,'Regulator E-2');
  Writeln (Datoteka,'   Cas          w            InReg        u            OutReg');

  Isys;
  Clrscr;
  Out_Screen (n_reg);
  Sinhi (Ts);

  Repeat Until Keypressed;
  t := 0;
  Repeat
    Sinh (ierr);
    for i:= 1 to n_reg do
      y[i] := AD (i-1,1);
    If (odd(trunc(t/60))) then
    begin
      for i:= 1 to (n_reg-1) do
        w[i] := 2;
    end
    else
    begin
      for i:= 1 to (n_reg-1) do
        w[i] := 1;
    end;

    w[n_reg] := OutReg[n_reg-1];
```

```
    Filtering (n_reg, Dim_Array, y, Filt_k, Filty_mean, Filtyf_mean,
               InReg, InRegold, Filt_Diff, Filt_mean_Diff, Filty_array,
               Filtyf_array, Filt_automb, Filtb);

    Code := Readcode;

    Status (Active_reg, Code, n_reg, Auto_catch_direct, mb,
            Auto_catchb, InReg, w);

    If ((abs(InReg[1]-w[1]) < 0.1) then Kp[1] := 0.5
                                   else Kp[1] := 1;

    Regulator (Ts, n_reg, Active_reg, Code, Incr_manual, Kp, Ka, Ti, Td,
               K_Tf, beta, gamma, w, elim, Vlim, InReg,
               v, z, Blhyst, Blmin, Umin, Umax, pb, ib,
               db, vb, zb, mb, hb, Bl_Catchb, Bl_Direct,
               uc, InRegold, x1old, x3old,
               uold, u, vold, OutReg, Bl_Ampl, wold);

    For i := 1 to n_reg do
      DA (i-1, OutReg[i]);

    t := t+Ts;

    Values_Screen (Active_reg, t, InReg[i], u[i], OutReg[i], Filtb[i], mb[i]);

    If (t < Tkon) then Writeln (datoteka, Simcos (t), Simcos (w[1]),
                                Simcos(InReg[1]), Simcos (u[1]), Simcos (OutReg[1]));
  Until (Code=27);
  Dsys;
  Close (datoteka);

end.
```

Drugi program nam kaže postopek *pisanja rezultatov v izbrano datoteko* (Simcosov format), *spreminjanje signala referenc* ter izbrane *konstante regulatorja* (w[i], Kp[1]) med izvajanjem glavne regulacijske zanke, ter način implementacije *kaskadne regulacije* (w[n_reg] = OutReg[n_reg-1]), kjer je signal reference zadnjega regulatorja izhod iz predzadnjega regulatorja.
Seveda je poseganja v program lahko {e dosti več (kontrola vseh spremenljivk regulatorja, poljubna vezava vhodov in izhodov regulacijskega algoritma, možnost dodatnega preoblikovanja vhodnega signala (kvadriranje, korenjenje,...), kot tudi izhodnega signala, dodatek ročnih funkcij, ...). Prikazana primera programov prikazujeta le uporabo najosnovnej{ih regulacijskih funkcij. Uporabimo lahko {e druge funkcije, lahko pa tudi napi{emo nove in tako prilagodimo program konkretnim zahtevam vodenja.
Oglejmo si {e izgled zaslona pri vna{anju parametrov regulatorja preko tipkovnice:

```
Fill in parameters directly ..... 1
Parameters are on the file ..... 2

Number of controllers (1) :

***** CONTROLLER 1 *****

Do you want predefined values of the parameters (N) :
Do you want predefined basic settings (Y) ?  n
Fill in pb (TRUE) :
Fill in ib (TRUE) :
Fill in db (TRUE) :
Proportional gain Kp (1) : 5
Integration time constant Ti (1e6) : 5
Diferential time constant Td (0) : 1
Time constant of the dif. filter Tf (Td/10) :
Fill in the sampling time Ts (0.1) :
Fill in the reference signal w (1) :
```

```
Do you want predefined Limits (Y) ?  n
Fill in maximum output Umax (10) : 8
Fill in minimum output Umin (0) :
Velocity limit of the output in V/s (1000) :
Velocity of the manual control in V/s (1) :

Do you want predefined values for overshoot and AW control (Y) ?

Do you want predefined values for other inputs and control (Y) ?

Do you want predefined values for hysteresis (Y) ?

Do you want predefined values for the input filter (Y) ?  n
Automatic switching of the input filter Filt_automb (TRUE) : f
Initial state of the input filter Filtb (FALSE) :
Fill in the time constant of the input filter Filt_t (Td/10) :
Initial value of the input signal InRegold (0) :
Enter the Filt_Diff (4.88e-3) :
Enter the Filt_mean_Diff (2.44e-3) :

Write new parameters into the file ? (Y)  y
Fill in the name of the file : test.dat
```

Program shrani parametre v datoteko, katero smo navedli na koncu (test.dat). Izgled datoteke s parametri je naslednji:

```
N_REG = 1
TS =  1.0000000000E-01

************************************************
                CONTROLLER = 1
************************************************

***************** BASIC SETTINGS ******************

PB = TRUE  ; Proportional part
IB = TRUE  ; Integral part
DB = TRUE  ; Diferential part
KP =  5.0000000000E+00  ; Proportional gain
TI =  5.0000000000E+00  ; Integral time constant
TD =  1.0000000000E+00  ; Diferential time constant
TF =  1.0000000000E-01  ; Time constant of the dif. filter
W =  1.0000000000E+00  ; Reference

******************** LIMITS ***********************

UMAX =  8.0000000000E+00  ; Max. output of the controller
UMIN =  0.0000000000E+00  ; Min. output of the controller
VLIM =  1.0000000000E+02  ; Velocity limit at the output (V/sample)
INCR_MANUAL =  1.0000000000E-01  ; Increment at the output for man. contr. (V/sample)

*********** OVERSHOOT & ANTI-WINDUP ***************

BETA =  1.0000000000E+00  ; Beta factor for the P-part
GAMMA =  0.0000000000E+00  ; Gamma factor for the D-part
ELIM =  1.0000000000E+03  ; Limiter at the input of the I-part
KA =  5.0000000000E+00  ; Anti-windup constant

************ OTHER INPUTS & CONTROL ***************

VB = FALSE  ; Aditive input (feed forward)
ZB = FALSE  ; Multiplicative input
HB = FALSE  ; Hold function (stop integrator)
MB = FALSE  ; Manual control

****************** HYSTERESIS *********************

BLHYST =  0.0000000000E+00  ; Amplitude of the hysteresis
BLMIN =  1.0000000000E+03  ; Change of the output for adding Blhyst

****************** INPUT FILTER *******************

FILT_AUTOMB = FALSE  ; Automatic switching on/off the input filter
FILTB = FALSE  ; Initial state of the input filter
```

```
FILT_T =  1.0000000000E-01  ; Time constant of the input filter
INREGOLD =  0.0000000000E+00  ; Initial value of the input signal
FILT_DIFF =  4.8800000000E-03  ; Value for autom. switching on/off the input filter
FILT_MEAN_DIFF =  2.4400000000E-03  ; Value for autom. switching on/off the input filter
```

Program nato nadaljuje z izvajanjem glavne regulacijske zanke. Med delovanjem programa je izgled zaslona naslednji:

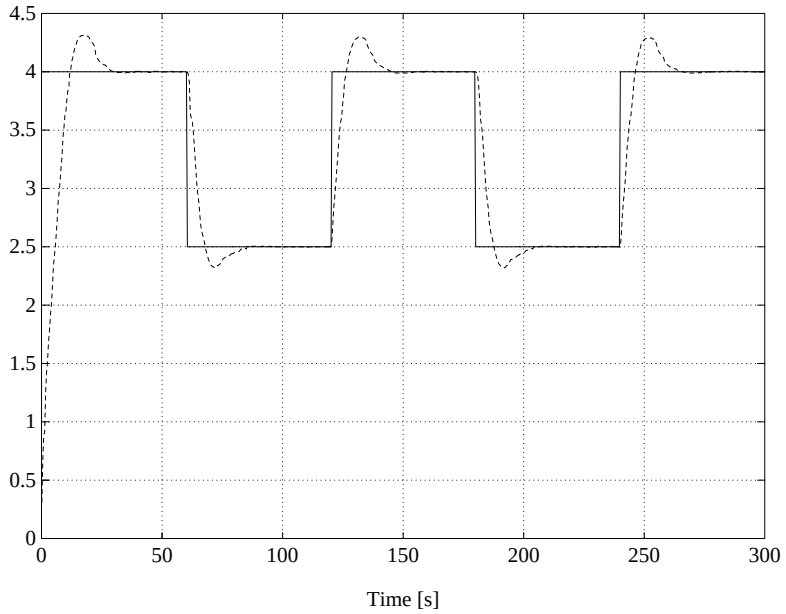
```
Time :       71.20            InReg         u        OutReg    Filter    A/M

Controller 1 :               2.2461      2.4394      2.4394               A


Esc = Exit
```

V zgornjem levem kotu je prikazan pretečen čas od začetka regulacije (v sekundah), Nato pa so prikazane vrednosti reguliranih (InReg) in regulirnih (u, OutReg) veličin. V primeru, če bi bil vklopljen vhodni filter, bi se v polju pod njim pokazala zvezdica, A/M pa nam kaže stanje regulatorja (avtomatsko/ročno).

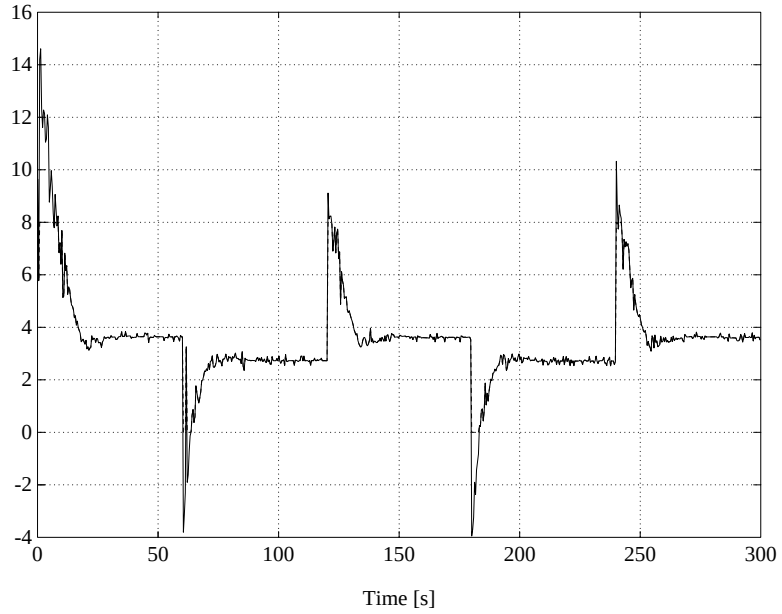
3.3.2. PRIMERI REZULTATOV VODENJA

Tu so predstavljeni rezultati vodenja (regulacije) na modelni napravi (hidravlična modelna naprava). Uporabili smo glavni program, ki je bil podoben drugemu programu, predstavljenemu v prej{njem podpoglavju (na strani 14), le za razliko, da nismo uporabljali kaskadne regulacije in spreminjanja parametra Kp[1]. Pri vnosu parametrov smo uporabili predefinirane vrednosti konstant in spremenljivk, razen Kp=5, Ti=5s, Td=1s, Umax=8V, Filt_Automb = FALSE (izključen vhodni filter). Odziv izhoda procesa prikazuje slika 2, odziv regulatorja pa slika 3.
Opazimo lahko, da je signal na izhodu iz regulatorja precej {umen (slika 2), kar smo posku{ali zmanj{ati z vključitvijo avtomatskega vklopa vhodnega filtra (Filt_Automb = TRUE), ter z vnosom konstant Filt_t = 1s, Filt_Diff = 0.00976 (4 LSB) in Filt_mean_Diff = 0.00488 (2 LSB). Dobili smo rezultate prikazane na slikah 4 in 5. Opazimo, da postane regulirni signal (slika 5) precej manj {umen (po prihodu sistema v delovno točko). Uporabnost vhodnega filtra pride do izraza takrat, ko ločljivost A/D pretvorbe ne zagotavlja dobre regulacije (npr. pri močnem ojačenju regulatorja (Kp) in (ali) relativno visoki diferencialni časovni konstanti (Td)), filter pa nam lahko posredno služi tudi za detekcijo prehoda v delovno točko.
Nato smo spremenili diferencialno časovno konstanto Td=0.2s in dobili rezultate prikazane na slikah 6 in 7. Opazimo, da ima sedaj sistem rahlo manj{e prevzpone, regulirni signal (slika 7) pa je dosti manj {umen v območju, ko se proces {e ne nahaja v delovni točki.
@eleli smo zmanj{ati prevzpon na izhodu iz procesa, zato smo uporabili omejevalnik na vhodu v integracijski člen: elim=0.2. Rezultate odzivov procesa in regulatorja kažeta sliki 8 in 9. Sedaj so prevzponi občutno zmanj{ani.
Poskusili smo {e ročno vodenje ter preklop med ročnim in avtomatskim načinom delovanja. Izklopili smo omejevalnik na vhodu v integracijski člen (elim=1000) in pognali regulacijski program. 30

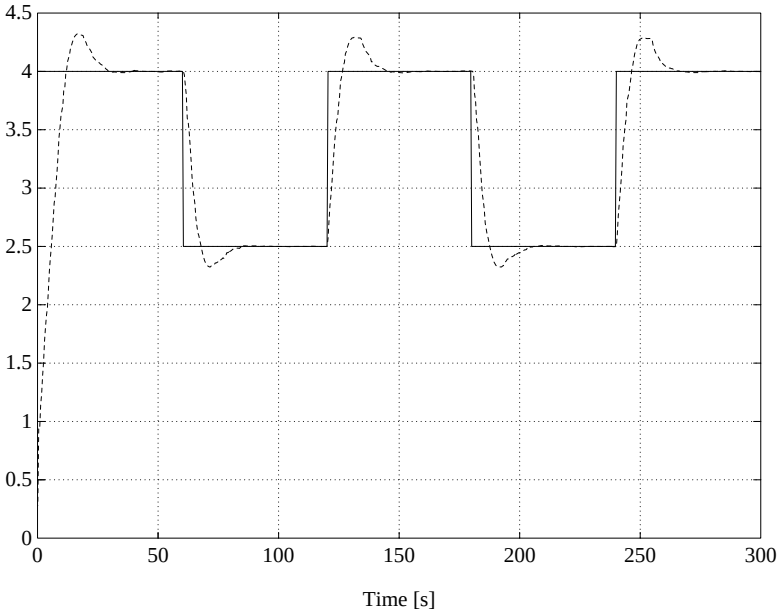
sekund po vsaki spremembi signala reference smo za par sekund vklopili ročno vodenje (pritisnili smo tipko 'm'), dvignili napetost na izhodu iz regulatorja za 0.5V, ter nato spet vklopili avtomatsko vodenje (s pritiskom tipke 'a'). Odzive prikazujeta sliki 10 in 11. Iz slike 11 (signali regulatorja) vidimo, da pri preklopu iz ročnega v avtomatski način delovanja ne prihaja do velikih skokov regulirne veličine. Seveda bi bili skoki veliko bolj neopazni, če bi vklopili "lovljenje" avtomatskega načina delovanja (s tipko 'c').



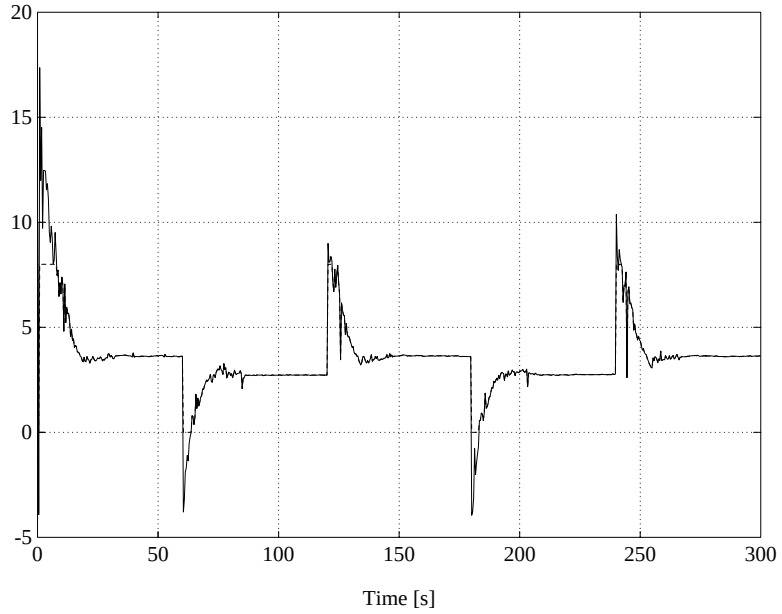
Slika 2: Odziv hidravlične modelne naprave pri Kp=5, Ti=5s, Td=1s, Umax=8V in Filt_Automb=FALSE; __ Signal reference, -- Izhod iz procesa



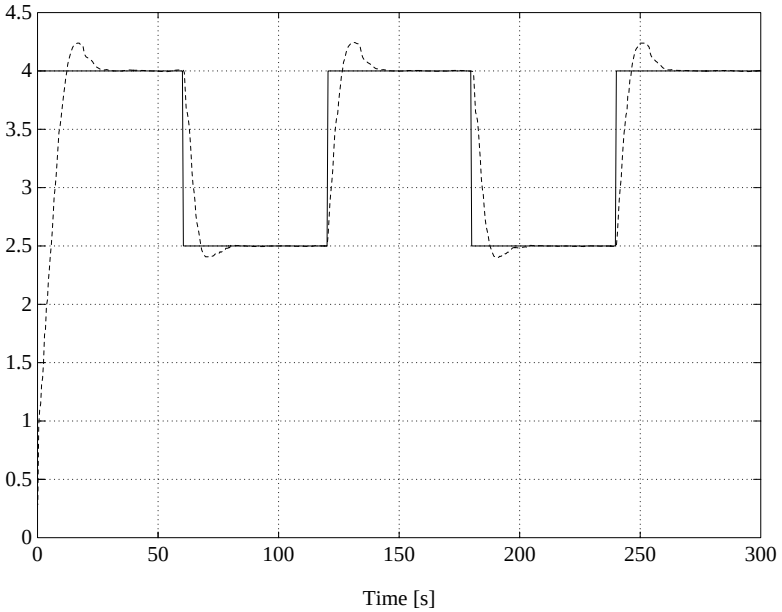
Slika 3: __ Nelimitiran signal regulatorja, -- Dejanski signal iz regulatorja



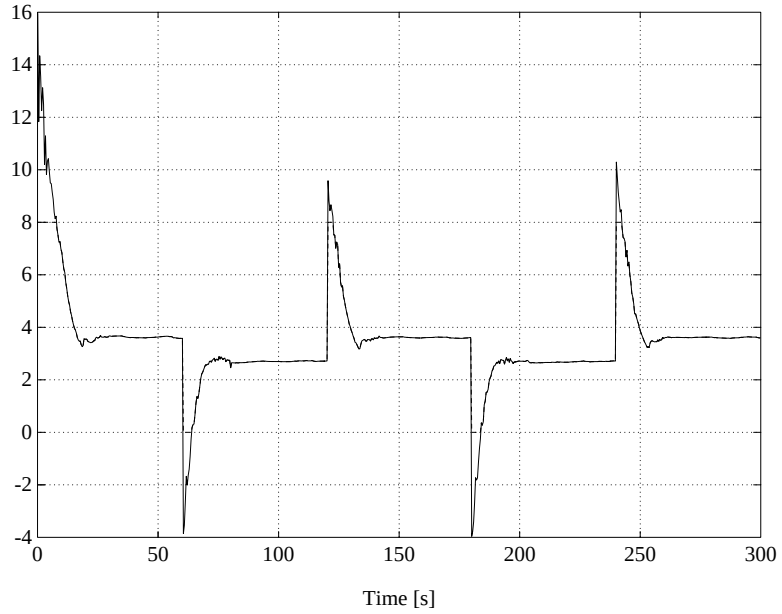
Slika 4: Filt_Automb=TRUE, Filt_t=1s, Filt_Diff=4LSB, Filt_mean_Diff=2LSB; __ Signal reference, -- Izhod iz procesa



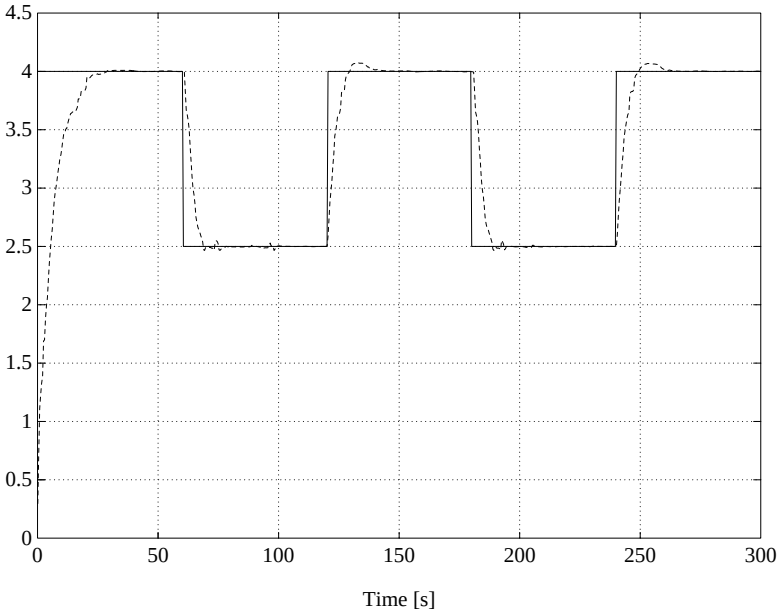
Slika 5: __ Nelimitiran signal regulatorja, -- Dejanski signal iz regulatorja



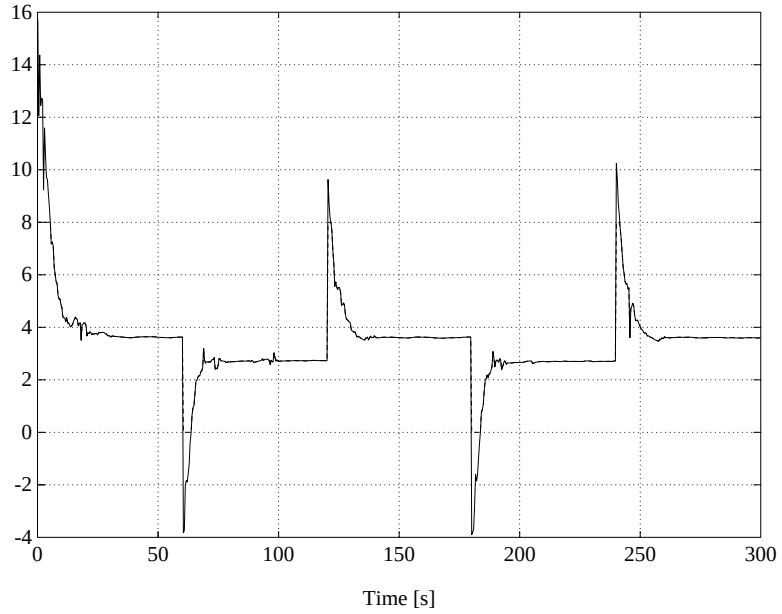
Slika 6: Td=0.2s; __ Signal reference, -- Izhod iz procesa



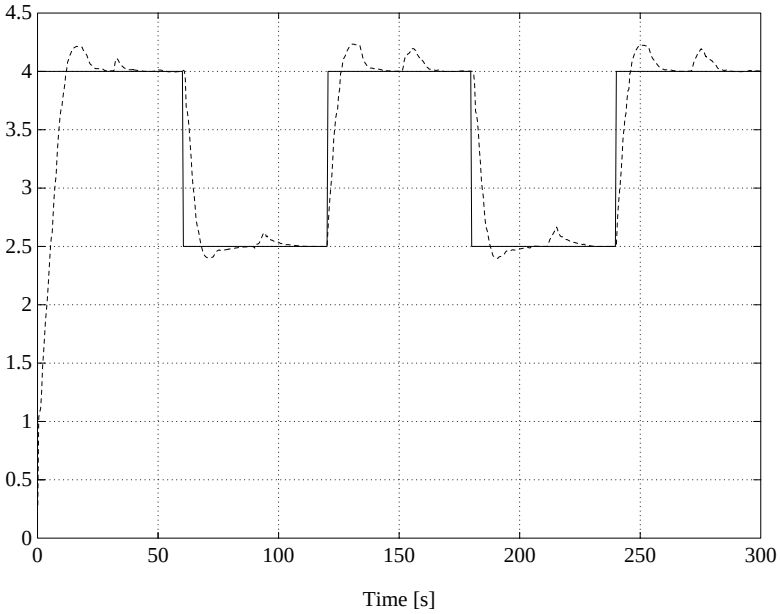
Slika 7: __ Nelimitiran signal regulatorja, -- Dejanski signal iz regulatorja



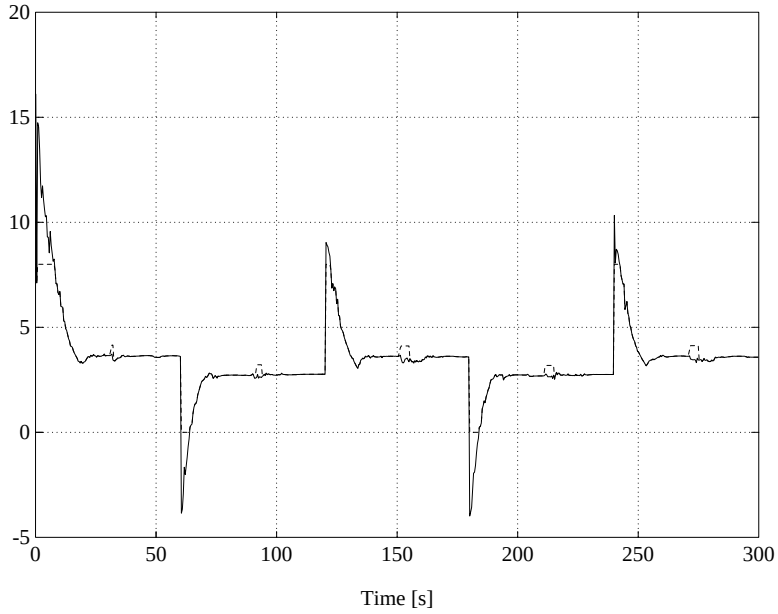
Slika 8: elim=0.2; __ Signal reference, -- Izhod iz procesa



Slika 9: __ Nelimitiran signal regulatorja, -- Dejanski signal iz regulatorja



Slika 10: Poskus ročnega vodenja, elim=1000;
__ Signal reference, -- Izhod iz procesa



Slika 11: __ Nelimitiran signal regulatorja, -- Dejanski signal iz regulatorja

4. ZAKLJUČEK

Rezultati testiranja modula so pokazali večje hitrosti izvajanja regulacijske zanke od Simcosovega (8 regulacijskih zank je delalo istočasno pri času vzorčenja 10 ms na AT486/33 MHz), hkrati pa je bila hitrost prevajanja programa neprimerno večja od tiste v programskem paketu Simcos. Vhodni filter smo za tem uporabili tudi na nestabilni modelni napravi kot signal za detekcijo prihoda v delovno točko in na osnovi tega menjali regulacijske parametre, kar se je zelo dobro obneslo. Modul je prav tako pripravljen na spremembe in nove regulacijske algoritme, ki bodo dodani po potrebi.

Slabosti modula so te, da ni pisan objektno (v vsakem osnovnem programu potrebujemo deklarirati precej{nje {tevilo spremenljivk) in je dokaj slabo strukturiran. Te slabosti bomo sku{ali odpraviti v nadaljnjem delu.

5. LITERATURA

[1] K.J.Åström, B.Wittenmark: Computer Controlled Systems: Theory and design, Prentice-Hall, 1984.

[2] K.J.Åström, C.C.Hang, P.Persson, W.K.Ho: Towards Intelligent PID Control, Automatica, Vol. 28, No. 1, pp. 1-9, 1992.

[3] E.H.Bristol: Designing and Programming Control Algorithms for DDC Systems, Control Engineering, January 1997.

[4] C.C.Hang, K.J.Åström, W.K.Ho: Refinements of the Ziegler-Nichols tuning formula, IEE proceedings-D, Vol. 138, No. 2, March 1991.

[5] R.Hanus: A New Technique for Preventing Control Windup, Journal A, Vol. 21, No. 1, 1980.

[6] R.Hanus: Anti windup and Bumpless Transfer: A survey, Computing and Computers for Control Systems, IMACS 1989.

[7] T.B.Kinney: Tuning process controllers, Chemical engineering, September 19, 1983.

[8] J.Petrovčič: Doktorska disertacija, FER, Ljubljana, maj 1992.

[9] J.Petrovčič, A.Bitenc: Dopolnitve regulacijskih algoritmov za večzančni mikroračunalni{ki regulator MMC-90, IJS, DP-6334.

[10] L.Rozsa: Design and implementation of Practical Digital PID Controllers, IFAC - Low Cost Automation 1989.

[11] L.Rundqwist: Anti-reset Windup for PID Controllers, 11th IFAC World Congress, Vol. 8, 1990.

[12] F.G.Shinskey: Process Control Systems, 3rd edition, McGraw Hill, 1988.

[13] D.Vrančić, J.Petrovčič, Đ.Juričić: Analiza metod za preprečevanje integralskega pobega in prevelikega prevzpona, IJS, DP-6769.

[14] D.Vrančić, J.Petrovčič, J.Grom, M.Štrubelj: Izdelava nestabilne in multivariabilne modelne naprave, IJS, DP-6729.

[15] D.Vrančić, J.Petrovčič, Đ.Juričić: Nastavitve parametrov PID regulatorjev za procese 2. reda, IJS, DP-6782.

DODATEK

Programski modul Control:

```
Unit control;

Interface

Uses Pci, Crt;

Const Dim_Array = 11;
      Max_Reg = 9;

Type filt_i = 1..Dim_Array;
     reg_i = 1..Max_Reg;
     filt = array [filt_i] of real;
     real_array = array [reg_i] of real;
     integ_array = array [reg_i] of integer;
     bool_array = array [reg_i] of boolean;
     fi_array = array [reg_i] of filt;

{ Meaning of the constants and variables - in () is a type of the constant
  or variable, in [] is predefined value

a)  Settings valid for all the controllers:

  Max_Reg (integer) : Max. number of the controllers [9];
  n_reg (integer) : Actual number of the controllers [1];
  Active_reg (integer) : Active controller - on which we can influence
                         by the keyboard [1];
  All_regb (boolean) : TRUE - all the controllers have the same parameters
                       [TRUE];
  Init_Constb (boolean) : TRUE - controller have the predefined values [TRUE];
  Ts (real) : Sampling time [0.1];
  Dim_array (integer) : length of the input filter's field [11];


b) Settings valid for each controller:


  Basic settings of the controller:

  pb (bool_array) : TRUE/FALSE - On/off proportional part [TRUE];
  ib (bool_array) : TRUE/FALSE - On/off the integral part [TRUE];
  db (bool_array) : TRUE/FALSE - On/off the diferential part [TRUE];
  Kp (real_array) : Proportional gain [1];
  Ti (real_array) : Integral time constant [1e6];
  Td (real_array) : Diferential time constant [0];
  Tf (real_array) : Time constant of the diferential's filter [Td/10];
  K_Tf (real_array) : Constant used for calculating the filter [exp(-Ts/Tf)];
  w (real_array) : Reference [1];


  Limits:

  Umax (real_array) : Max. value of the output signal [10];
  Umin (real_array) : Min. value of the output signal [0];
  Vlim (real_array) : Velocity limit at the output of the controller =
                      max. change of the output in one sample (Ts) [1000*Ts];
  Incr_manual (real_array) : Change of the output signal in one sample when
                             pressing the up/down arrows on manual control
                             [1*Ts];
```

```
Constants for controlling the overshoot and anti-windup:

beta (real_array) : Represents connection between the reference and P-part
                    of the controller - beta*w [1];
gamma (real_array) : Represents connection between the reference and D-part
                     of the controller - gamma*w [0];
elim (real_array) : Amplitude of the limiter at the input of the integrator
                    [1000];
Ka (real_array) : Anti-windup constant. For conditioning technique : Ka=Kp.
                  [Kp];


Aditional inputs and control:

vb (bool_array) : TRUE/FALSE - On/off the aditive input [FALSE];
zb (bool_array) : TRUE/FALSE - On/off the multiplicative input [FALSE];
v (real_array) : aditive input [-];
z (real_array) : multiplicative input [-];
hb (bool_array) : TRUE/FALSE - Stop/normal integration [FALSE];
mb (bool_array) : TRUE/FALSE - On/off the manual control [FALSE];


Hysteresis:

Blhyst (real_array) : Amplitude of the hysteresis [0];
Blmin (real_array) : Minimum value of the output change to the opposite
                     side for adding the Blhyst signal [1000];
Bl_Catchb (bool_array) : TRUE/FALSE - On/off searching of the peak value
                         of the output signal [TRUE];
Bl_Ampl (real_array) : Amplitude of the output peak [0];
Bl_Direct (integ_array) : The peak means 1 = minimum
                                        -1 = maximum [1];


Input filter:

Filt_Automb (bool_array) : TRUE/FALSE - On/off the automatic switching of
                           the input filter [TRUE];
Filtb (bool_array) : TRUE/FALSE - On/off the input filter [FALSE];
Filt_t (real_array) : Time constant of the input filter - 1. order
                      [Td/2 if Filt_Automb = TRUE,
                       Td/10 if Filt_Automb = FALSE];
Filt_k (real_array) : Constant used for calculating the input filter
                      [exp(-Ts/Filt_t)];
Filt_Diff (real_array) : Used for the automatic switching of the input
                         filter. If the difference between the minimum and
                         maximum value of the vector
                         InRegf_array > Filt_Diff, then input filter is
                         switched off [2*LSB = 4.88e-3];
Filt_mean_Diff (real_array) : Used for the automatic switching of the input
                              filter. If the difference between the mean
                              values of the vectors Inreg_array and
                              InRegf_array are bigger then Filt_mean_Diff,
                              then input filter is switched off
                              [1*LSB = 2.44e-3];
InRegold (real_array) : The old value of the input of the controller [0];
InReg_array (fi_array) : The array of the input, no-filtered values
                         [vector = InRegold];
InRegf_array (fi_array) : The array of the input, filtered values
                          [vector = InRegold];
InReg_mean (real_array) : The mean value of the vector InReg_array
                          [InRegold];
InRegf_mean (real_array) : The mean value of the vector InRegf_array
                           [InRegold];


Bumpless manual/automatic switching:

Auto_Catchb (bool_array) : TRUE/FALSE - On/off "catching" the automatic mode
                           If it's on, is waiting when input signal crosses
                           the reference value. Then switches from manual
```

```
                              to automatic mode (mb = FALSE) [FALSE];
  Auto_Catch_Direct (integ_array) : Information about the "position" of the
                                    input signal:
                                    1 ... InReg > w when Auto_Catchb
                                    is switched on
                                   -1 ... InReg < w when Auto_Catchb
                                    is switched on  [-1];

  Other variables:

  uold (real_array) : Old value of the no-limited output signal [0];
  wold (real_array) : Old value of the reference [0];
  u (real_array) : No-limited output signal [uold];
  uc (real_array) : Limited output signal [uold];
  x1old (real_array) : Old value of the internal variable (D-part) [0];
  x2old (real_array) : Old value of the internal variable (P-part) [0];
  vold (real_array) : Old value of the aditive input [0];
  OutReg (real_array) : Initial value for the output of the controller [0];

}



Procedure In_Param (Dim_Array : integer;
                    Var All_regb : boolean;
                    Var n_reg, Active_Reg : integer;
                    Var Ts : real;
                    Var Init_Constb : boolean;
                    Var Incr_manual, Kp, Ti, Td, K_Tf, w, Umax, Umin, Vlim,
                    beta, gamma, elim, Ka, Bl_Ampl, Blmin, Blhyst, Filt_k,
                    InRegold, InReg_Mean, InRegf_Mean, wold, uold, u, uc,
                    x1old, x2old, vold, Tf, Filt_t, OutReg, Filt_Diff,
                    Filt_mean_Diff : real_array;

                    Var Bl_direct, Auto_Catch_Direct : integ_array;

                    Var pb, ib, db, vb, zb, hb, Bl_Catchb, Filtb,
                    Filt_automb, Auto_catchb, mb : bool_array;

                    Var InReg_Array, InRegf_Array : fi_array);

Procedure Out_Screen (n_reg : integer);

Procedure Filtering (n_reg, Dim_Array : integer;
                     y, Filt_k : real_array;
                     Var InReg_Mean, InRegf_Mean, InReg, InRegold,
                     Filt_Diff, Filt_mean_Diff : real_array;
                     Var InReg_Array, InRegf_Array : fi_array;
                     Filt_automb : bool_array;
                     Var Filtb : bool_array);

Function Readcode : integer;

Procedure Status (Var Active_reg : integer;
                  Code, n_reg : integer;
                  var Auto_catch_direct : integ_array;
                  var mb, Auto_catchb : bool_array;
                  InReg, w : real_array);

Procedure Controller (Kp, Ka, Ti, Td, K_Tf, Ts, beta, gamma, w,
                      elim, Vlim, InReg, v, z, Blhyst, Blmin,
                      Umin, Umax : real;
                      pb, ib, db, vb, zb, mb, hb : boolean;
                      Var Bl_Catchb : boolean;
                      Var Bl_Direct : integer;
                      var uc, InRegold, x1old, x2old, uold, u, vold,
                      OutReg, Bl_Ampl, wold : real);

Procedure Regulator ( Ts : real;
                      n_reg, Active_reg, Code : integer;
                      Incr_manual, Kp, Ka, Ti, Td, K_Tf, beta, gamma, w,
                      elim, Vlim, InReg, v, z, Blhyst, Blmin,
```

```
                    Umin, Umax : real_array;
                    pb, ib, db, vb, zb, mb, hb : bool_array;
                    Var Bl_Catchb : bool_array;
                    Var Bl_Direct : integ_array;
                    var uc, InRegold, x1old, x2old, uold, u, vold,
                    OutReg, Bl_Ampl, wold : real_array);

Procedure Values_Screen (Active_reg, n_reg : integer;
                         t : real;
                         InReg, u, OutReg : real_array;
                         Filtb, mb : bool_array);

Procedure Manual_out (Code : integer;
                      Var Bl_Direct : integer;
                      Var Bl_Catchb : boolean;
                      mb : boolean;
                      Active_Reg, i : integer;
                      Umax, Umin, Incr_manual : real;
                      Var OutReg : real);

Procedure Write_file (stfile : string;
                      n_reg : integer;
                      Ts : real;
                      Incr_manual, Kp, Ti, Td, Tf, w, Umax, Umin, Vlim,
                      beta, gamma, elim, Ka, Blmin, Blhyst, Filt_k,
                      InRegold, Filt_t, Filt_Diff, Filt_mean_Diff : real_array;

                      pb, ib, db, vb, zb, hb, Filtb, Filt_automb,
                      mb : bool_array);

Procedure Read_file (stfile : string;
                     Dim_Array : integer;
                     Var n_reg : integer;
                     Var Ts : real;
                     Var Incr_manual, Kp, Ti, Td, K_Tf, w, Umax, Umin, Vlim,
                     beta, gamma, elim, Ka, Bl_Ampl, Blmin, Blhyst, Filt_k,
                     InRegold, InReg_Mean, InRegf_Mean, wold, uold, u, uc,
                     x1old, x2old, vold, Tf, Filt_t, OutReg, Filt_Diff,
                     Filt_mean_Diff : real_array;

                     Var Bl_direct, Auto_Catch_Direct : integ_array;

                     Var pb, ib, db, vb, zb, hb, Bl_Catchb, Filtb,
                     Filt_automb, Auto_catchb, mb : bool_array;

                     Var InReg_Array, InRegf_Array : fi_array);

Procedure Filter (Var InReg_Mean, InRegf_Mean : real;
                  Var InReg_Array, InRegf_Array : filt;
                  y, Filt_k : real;
                  Dim_Array : integer);

Function Switch_filter (InReg_Mean, InRegf_Mean,
                        Filt_Diff, Filt_mean_Diff : real;
                        Dim_Array : integer;
                        InRegf_Array : filt) : boolean;

Function Simcos (Xinp : real) : string;


Implementation


Function Limiter (Xin : real;
                  Xmax, Xmin : real) : real;

{ Function limits the input signal Xin }

var temp : real;

begin
```

```
  if (Xin > Xmax) then temp := Xmax
  else if ( Xin < Xmin) then temp := Xmin
  else temp := Xin;
  Limiter := temp;
end;   { Procedure Limiter }



Function Sgn (x : real) : integer;

{ Function gives result 1 if x > 0, 0 if x = 0 and -1 if x < 0 }

begin
  if (x > 0) then Sgn := 1
  else if (x = 0) then Sgn := 0
  else Sgn := -1;
end; { Function Sgn }



Procedure Shift (Xin : real;
                 Var Xarray : filt;
                 n : integer);

{ Procedure shifts to the right values in the vector Xarray. On the first
  place it puts Xin, n is the length of the vector Xarray }

Var i : integer;

begin
  for i := n-1 downto 1 do
    Xarray [i+1] := Xarray [i];
  Xarray [1] := Xin;
end;  { Procedure Shift }


Function Mean (Xmean : real;
               Xarray : filt;
               n : integer) : real;

{ Function calculates the mean value of the vector Xarray. Xmean is old
  mean value }

begin
  Mean := Xmean + (Xarray [1] - Xarray [n])/(n-1);
end; { Procedure Mean }



Function Max (Xarray : filt;
              n : integer) : real;

{ Function calculates the highest element of the vector Xarray }

Var i : integer;
    temp : real;

begin
  temp := Xarray [1];
  for i := 2 to n do
    if (Xarray [i] > temp) then temp := Xarray [i];
  Max := temp;
end;  { Function max }



Function Min (Xarray : filt;
              n : integer) : real;

{ Function calculates the lowest element of the vector Xarray }

Var i : integer;
    temp : real;
```

```
begin
  temp := Xarray [1];
  for i := 2 to n do
    if (Xarray [i] < temp) then temp := Xarray [i];
  Min := temp;
end;  { Function min }



Procedure Filter (Var InReg_Mean, InRegf_Mean : real;
                  Var InReg_Array, InRegf_Array : filt;
                  y, Filt_k : real;
                  Dim_Array : integer);

{ Procedure shifts the vector InReg_Array and puts the input value (y) on the
  first place. Then filters the input value and puts the filtered value to the
  first place of the shifted vector InRegf_Array. After that procedure
  calculates the new mean values of vectors InReg_Array and InRegf_Array }

Var yfilt : real;

begin
  Shift (y, InReg_Array, Dim_Array);
  yfilt := Filt_k*InRegf_Array[1] + (1-Filt_k)*y;
  Shift (yfilt, InRegf_Array, Dim_Array);
  InReg_Mean := Mean (InReg_Mean, InReg_Array, Dim_Array);
  InRegf_Mean := Mean (InRegf_Mean, InRegf_Array, Dim_Array);

end;  { Procedure Filter }



Function Switch_filter (InReg_Mean, InRegf_Mean,
                        Filt_Diff, Filt_mean_Diff : real;
                        Dim_Array : integer;
                        InRegf_Array : filt) : boolean;

{ Function detects process came into the working point }

begin
  if ((abs (InReg_Mean-InRegf_Mean)) < Filt_mean_Diff) and
     ((max (InRegf_Array, Dim_Array) - min (InRegf_Array, Dim_Array))
     < Filt_Diff)

   then Switch_filter := TRUE
   else Switch_filter := FALSE;

end; { Function Switch_filter }



Procedure Status (Var Active_reg : integer;
                  Code, n_reg : integer;
                  Var Auto_catch_direct : integ_array;
                  Var mb, Auto_catchb : bool_array;
                  InReg, w : real_array);

{ Procedure switches between automatic and manual mode (mb), defines the active
  controller (Active_reg) and switches "catching" of the automatic mode
  (Auto_Catchb) according to the pressed character (Code) }

begin
  if (Code > 48) and (Code < 58) then
  begin
    if ((Code-48) <= n_reg) then Active_reg := Code-48;
  end
           { Character 'a' or 'A' }
  else if (Code = 97) or (Code = 65) then mb[Active_reg] := FALSE
           { Character 'm' or 'M' }
  else if (Code = 109) or (Code = 77) then mb[Active_reg] := TRUE
  else if (Code = 99) or (Code = 67) then       { Character 'c' or 'C' }
  begin
```

```
    Auto_catchb[Active_reg] := TRUE;
    Auto_catch_direct[Active_reg] := Sgn (InReg[Active_reg]-w[Active_reg]);
  end;
  if (Auto_catchb[Active_reg]) then
  begin
    if (Auto_catch_direct[Active_reg] <>
       (Sgn (InReg[Active_reg]-w[Active_reg]))) then
    begin
      mb[Active_reg] := FALSE;
      Auto_catchb[Active_reg] := FALSE;
    end;
  end;

end;  { Procedure Status }



Function Readcode : integer;

{ Function gives a code of the pressed character (sign). 200 (decimal) is
  added to the codes of arrows, due to the reason not to be mixed with
  common characters }

Var temp, temp1, temp2, temp3 : integer;

begin

  temp1 := Iscan;
  temp := Iscan;
  repeat
    temp3 := temp1;
    temp2 := temp;
    temp1 := Iscan;
    temp := Iscan;
  until (temp = 0);
  if (temp1 = 0) and (temp2 = 0) then Readcode := temp3
  else if (temp1 = 0) and (temp3 <> 0) then Readcode := temp2
  else if (temp1 = 0) and (temp3 = 0) then Readcode := temp2+200
  else if (temp2 <> 0) then Readcode := temp1
  else Readcode := temp1+200;

end;  { Function Readcode }



Function Simcos (Xinp : real) : string;

{ Function is used for writing to the file real values when they have to be
  written in SIMCOS format }

Var temp1, temp2 : string;
begin
  Str (Xinp,temp1);
  temp2 := temp1;
  temp1 := Copy (temp1, 1, 8);
  temp2 := Copy (temp2, 14,4);
  Simcos := ' ' + temp1 + temp2;

end;  { Function Simcos }



Procedure Manual_out (Code : integer;
                      Var Bl_Direct : integer;
                      Var Bl_Catchb : boolean;
                      mb : boolean;
                      Active_Reg, i : integer;
                      Umax, Umin, Incr_manual : real;
                      Var OutReg : real);

{ Procedure changes the output value at the controller (OutReg) for the
  active, manual controlled controller, according to the pressed arrow
  (Code = 272, 280, 277 or 275). Bl_Direct and Bl_Catchb (for hysteresis
```

```
  control) are updated, as well. }

begin
  if (mb and (Active_Reg = i)) then
  begin
    if (Code = 272) then
    begin
      OutReg := OutReg + Incr_manual;
      Bl_Direct := 1;
      Bl_Catchb := TRUE;
    end
    else if (Code = 280) then
    begin
      OutReg := OutReg - Incr_manual;
      Bl_Direct := -1;
      Bl_Catchb := TRUE;
    end
    else if (Code = 277) then
    begin
      OutReg := OutReg + Incr_manual/10;
      Bl_Direct := 1;
      Bl_Catchb := TRUE;
    end
    else if (Code = 275) then
    begin
      OutReg := OutReg - Incr_manual/10;
      Bl_Direct := -1;
      Bl_Catchb := TRUE;
    end;
    OutReg := Limiter (OutReg,Umax,Umin);
  end;

end;  { Function Manual_out }



Procedure Out_Screen (n_reg : integer);

{ Procedure writes the basic text (without values) to the screen. It is
  used before the main controller loop }

Var i : integer;

begin
  Clrscr;
  gotoxy (1,4);
  Write ('Time :');
  gotoxy (21,4);
  Write ('          InReg         u        OutReg    Filter    A/M');
  for i := 1 to n_reg do
  begin
    Gotoxy (1,5+i);
    Write ('Controller ',i,' :  ');
  end;
  Gotoxy (1,8+n_reg);
  Writeln ('Esc = Exit');
end;  { Procedure Out_Screen }



Procedure Values_Screen (Active_reg, n_reg : integer;
                         t : real;
                         InReg, u, OutReg : real_array;
                         Filtb, mb : bool_array);

{ Procedure writes the values of the controller (input value, not-limited
  output value, output value, state of the filter and manual/automatic
  control). It's used in the main control loop of the program }

Var ch1, ch2 : char;
    i : integer;

begin
```

```
  for i := 1 to n_reg do
  begin
    if (Filtb[i]) then ch1 := '*'
                  else ch1 := ' ';
    if (mb[i]) then ch2 := 'M'
               else ch2 := 'A';
    if (i = 1) then
    begin
      Gotoxy (9,4);
      Write (t:10:2);
    end;
    Gotoxy (24,5+i);
    if (Active_reg = i) then Highvideo
                        else Normvideo;
    Write (InReg[i]:12:4, u[i]:12:4, OutReg[i]:12:4,
           '      ', ch1,'        ',ch2);
    Normvideo;
  end;

end;  { Procedure Values_Screen }



Procedure In_Integ (st : string;
                    default : integer;
                    Var inp : integer);

{ Procedure writes the text (st) and then puts the answer into the value
  inp. If only <ENTER> is pressed, then inp = default. }

Var st1 : string;
    ierr : integer;

Begin
  Repeat
    Write (st);
    Readln (st1);
    val (st1,inp,ierr);
    If (ierr <> 0) then inp := default;
  Until (ierr = 0) or (length (st1) = 0);

end;  { Procedure In_Integ }



Procedure In_Real (st : string;
                   default : real;
                   Var inp : real);

{ Procedure writes the text (st) and then puts the answer into the value
  inp. If only <ENTER> is pressed, then inp = default. }

Var st1 : string;
    ierr : integer;

Begin
  Repeat
    Write (st);
    Readln (st1);
    val (st1,inp,ierr);
    If (ierr <> 0) then inp := default
  Until (ierr = 0) or (length (st1) = 0);

end;  { Procedure In_Real }



Procedure In_Boolean (st : string;
                      default : boolean;
                      Var inp : boolean);

{ Procedure writes the text (st) and then puts the answer into the value
  inp. If only <ENTER> is pressed, then inp = default. Right answers are
```

```
  (valid for big and small characters): 'true', 't', '1', 'y' and
  'false', 'f', '0', 'n' }

Var st1 : string;
    i,ierr : integer;

Begin
  Repeat
    ierr := 0;
    Write (st);
    Readln (st1);
    For i := 1 to length (st1) do
      st1[i] := upcase (st1[i]);
    If ((st1 = 'TRUE') or (st1 = 'T') or (st1 = '1') or (st1 = 'Y')) then
      inp := TRUE
    else if ((st1 = 'FALSE') or (st1 = 'F') or (st1 = '0') or (st1 = 'N')) then
      inp := FALSE
    else if (length (st1) = 0) then inp := default
    else ierr := 1;
  Until (ierr = 0);

end;  { Procedure In_Boolean }



Procedure In_Initial (Var n_reg : integer;
                      Var All_Regb : boolean);

{ Procedure inputs the initial values for the controllers }

Begin
  All_Regb := FALSE;
  In_Integ ('Number of controllers (1) : ',1,n_reg);
  If (n_reg > 1) then
  begin
    Writeln;
    In_Boolean ('The same parameters for each controller (Y) : ',
                TRUE,All_Regb);
  end;
end;  { Procedure In_Initial }



Procedure In_Basic (Init_Constb : boolean;
                    Var pb, ib, db : boolean;
                    Var Kp, Ti, Td, Tf, Ts, K_Tf, w : real);

{ Procedure inputs the basic values for the controller }

Var ch : char;
    tmpb : boolean;

Begin
  If (not Init_Constb) then
    In_Boolean ('Do you want predefined basic settings (Y) ?  ',TRUE,tmpb);
  if ((tmpb = FALSE) and (not Init_Constb)) then
  begin
    In_Boolean ('Fill in pb (TRUE) : ',TRUE,pb);    { ON / OFF P-part }
    In_Boolean ('Fill in ib (TRUE) : ',TRUE,ib);    { ON / OFF I-part }
    In_Boolean ('Fill in db (TRUE) : ',TRUE,db);    { ON / OFF D-part }
    In_Real ('Proportional gain Kp (1) : ',1,Kp);
    In_Real ('Integration time constant Ti (1e6) : ',1e6,Ti);
    In_Real ('Diferential time constant Td (0) : ',0,Td);
    In_Real ('Time constant of the dif. filter Tf (Td/10) : ',Td/10,Tf);
    In_Real ('Fill in the sampling time Ts (0.1) : ',0.1,Ts);
    If (Tf > Ts/50) then K_Tf := exp (-Ts/Tf)
                    else K_Tf := 0;
    In_Real ('Fill in the reference signal w (1) : ',1,w);
  end
  else
  begin
    pb := TRUE;
    ib := TRUE;
```

```
    db := TRUE;
    Kp := 1;
    Ti := 1e6;
    Td := 0;
    Tf := Td/10;
    Ts := 0.1;
    K_Tf := 0;
    w := 1;
  end;

end;  { Procedure In_Basic }


Procedure In_Limits (Init_Constb : boolean;
                     Ts : real;
                     Var Umax, Umin, Vlim, Incr_manual : real);

{ Procedure inputs the Limits for the controller }

Var ch : char;
    temp : real;
    tmpb : boolean;

Begin
  If (not Init_Constb) then
    In_Boolean ('Do you want predefined Limits (Y) ?  ',TRUE,tmpb);
  if ((tmpb = FALSE) and (not Init_Constb)) then
  begin
              { Maximum value at the output of the controller }
    In_Real ('Fill in maximum output Umax (10) : ',10,Umax);
              { Minimum value at the output of the controller }
    In_Real ('Fill in minimum output Umin (0) : ',0,Umin);
    In_Real ('Velocity limit of the output in V/s (1000) : ',1000,temp);
    Vlim := temp*Ts;     { Vlim = Velocity [V/s] * Ts }
    In_Real ('Velocity of the manual control in V/s (1) : ',1,temp);
    Incr_manual := temp*Ts;   { Velocity limit in manual control on using
                                arrows : Incr_manual = Velocity [V/s] * Ts }
  end
  else
  begin
    Umax := 10;
    Umin := 0;
    Vlim := 1000*Ts;
    Incr_manual := 1*Ts;
  end;

end;  { Procedure In_Limits }


Procedure In_Overshoot_AW (Init_Constb : boolean;
                           Kp : real;
                           Var beta, gamma, elim, Ka : real);

{ Procedure inputs the constants for the overshoot rejection and
  anti-windup for the controller }

Var ch : char;
    temp : real;
    tmpb : boolean;

Begin
  If (not Init_Constb) then
    In_Boolean ('Do you want predefined values for overshoot and AW control (Y) ?  ',
                TRUE,tmpb);
  if ((tmpb = FALSE) and (not Init_Constb)) then
  begin
              { Beta factor for P-part of the PID controller }
    In_Real ('Fill in beta factor (1) : ',1,beta);
              { Gamma faktor for D-part of the PID controller }
    In_Real ('Fill in gamma factor (0) : ',0,gamma);
              { Limiter at the input of the integrator }
```

```
    In_Real ('Fill in the value of the limiter elim (1000) : ',1000,elim);
    temp := Kp*beta;
    In_Real ('Fill in the anti-windup constant Ka (Kp*beta) : ',temp,Ka);
  end
  else
  begin
    beta := 1;
    gamma := 0;
    elim := 1000;
    Ka := Kp*beta;
  end;

end;  { Procedure In_Overshoot_AW }


Procedure In_Aditional_Inputs (Init_Constb : boolean;
                               Var vb, zb, hb, mb : boolean);

{ Procedure inputs the settings for the aditional inputs }

Var ch : char;
    temp : real;
    tmpb : boolean;

Begin
  If (not Init_Constb) then
    In_Boolean ('Do you want predefined values for other inputs and control (Y) ?  ',
                TRUE,tmpb);
  if ((tmpb = FALSE) and (not Init_Constb)) then
  begin
                { OFF/ON aditive input }
    In_Boolean ('Fill in vb - aditive input (FALSE) : ',FALSE,vb);
                { OFF/ON multiplicative input }
    In_Boolean ('Fill in zb - multiplicative input (FALSE) : ',FALSE,zb);
                { Hold function for integrator }
    In_Boolean ('Fill in hb - stop integrator (FALSE) : ',FALSE, hb);
                { OFF/ON manual control }
    In_Boolean ('Fill in mb - manual control (FALSE) : ',FALSE,mb);
  end
  else
  begin
    vb := FALSE;
    zb := FALSE;
    hb := FALSE;
    mb := FALSE;
  end;

end;  { Procedure In_Aditional_Inputs }



Procedure In_Hysteresis (Init_Constb : boolean;
                         Var Bl_Catchb : boolean;
                         Var Bl_Direct : integer;
                         Var Bl_Ampl, Blmin, Blhyst : real);

{ Procedure inputs the settings for controlling the hysteresis }

Var ch : char;
    temp : real;
    tmpb: boolean;

Begin
  If (not Init_Constb) then
    In_Boolean ('Do you want predefined values for hysteresis (Y) ?  ',
                TRUE,tmpb);
  if ((tmpb = FALSE) and (not Init_Constb)) then
  begin
    Bl_Catchb := TRUE;   { Switching on catching of the peak value }
              { 1=up, -1=down }
    Bl_Direct := 1;   { Direction of the peak value at the output }
```

```
    Bl_Ampl := 0;  { The last peak value at the output of the controller }
    In_Real ('Fill in minimum change of the output for hysteresis Blmin (1000) : ',
             1000,Blmin);
    In_Real ('Fill in amplitude of the hysteresis Blhyst (0) : ',0,Blhyst);
  end
  else
  begin
    Bl_Catchb := TRUE;
    Bl_Direct := 1;
    Bl_Ampl := 0;
    Blmin := 1000;
    Blhyst := 0;
  end;

end;  { Procedure In_Hysteresis }



Procedure In_Filter (Init_Constb : boolean;
                     Ts, Td : real;
                     Var Filt_automb, Filtb : boolean;
                     Var Filt_t, Filt_k, InRegold,
                     InReg_Mean, InRegf_Mean, Filt_Diff,
                     Filt_mean_Diff : real;
                     InReg_Array, InRegf_Array : filt);

{ Procedure sets the input filter of the controller }

Var ch : char;
    temp1, temp2 : real;
    i : integer;
    tmpb : boolean;

Begin
  If (not Init_Constb) then
    In_Boolean ('Do you want predefined values for the input filter (Y) ?  ',
                TRUE,tmpb);
  if ((tmpb = FALSE) and (not Init_Constb)) then
  begin
    In_Boolean ('Automatic switching of the input filter Filt_automb (TRUE) : ',
                TRUE,Filt_automb);
                { OFF/ON filter }
    In_Boolean ('Initial state of the input filter Filtb (FALSE) : ',
                FALSE,Filtb);
    temp1 := Td/2;
    temp2 := Td/10;
    if (Filt_automb) then
      In_Real ('Fill in the time constant of the input filter Filt_t (Td/2) : ',
               temp1,Filt_t)
      else In_Real ('Fill in the time constant of the input filter Filt_t (Td/10) : ',
                    temp2,Filt_t);
    If (Filt_t > Ts/50) then Filt_k := exp (-Ts/Filt_t)
                        else Filt_k := 0;
             { The old input value of the controller }
    In_Real ('Initial value of the input signal InRegold (0) : ',0,InRegold);
    for i := 1 to 10 do   { Initialisation of the input arrays of filtered and
                            no-filtered input signals }
    begin
      InReg_Array [i] := InRegold;
      InRegf_Array [i] := InRegold;
    end;
    InReg_Mean := InRegold;    { Mean value of the no-filtered input signals }
    InRegf_Mean := InRegold;   { Mean value of the filtered input signals }
    In_Real ('Enter the Filt_Diff (4.88e-3) : ',4.88e-3,Filt_Diff);
    In_Real ('Enter the Filt_mean_Diff (2.44e-3) : ',2.44e-3,Filt_mean_Diff);
  end
  else
  begin
    Filt_automb := TRUE;
    Filtb := FALSE;
    if (Filt_automb) then
      Filt_t := Td/2
      else Filt_t := Td/10;
```

```
    If (Filt_t > Ts/50) then Filt_k := exp (-Ts/Filt_t)
                        else Filt_k := 0;
    InRegold := 0;
    for i := 1 to 10 do
    begin
      InReg_Array [i] := InRegold;
      InRegf_Array [i] := InRegold;
    end;
    InReg_Mean := InRegold;
    InRegf_Mean := InRegold;
    Filt_Diff := 4.88e-3;
    Filt_mean_Diff := 2.44e-3;
  end;

end;  { Procedure In_Filter }


Procedure In_Manual_Automatic (Init_Constb : boolean;
                               Var Auto_Catchb : boolean;
                               Var Auto_Catch_Direct : integer);

{ Procedure sets the predefined values for the automatic man/auto switching }

Begin

  Auto_catchb := FALSE;   { OFF/ON catching the automatic mode }
  Auto_catch_direct := -1;   { InReg < w in catching the automatic mode }

end;  { Procedure In_Manual_Automatic }


Procedure In_Other (Var wold, uold, u, uc, X1old, x2old, vold, OutReg : real);

{ Procedure sets the predefined values for other constants }

Begin
  wold := 0;  { Old value of the reference }
  uold := 0;  { Old value of the no-limited output }
  u := uold;  { Initial value of the no-limited output }
  uc := uold;  { Initial value of the limited output }
  x1old := 0;  { Old value of the internal variable }
  x2old := 0;  { Old value of the internal variable }
  vold := 0;   { Old value of the aditive input }
  OutReg := 0;    { Initial value of the output }

end;  { Procedure In_Other }


Procedure Extract (st : string;
                   Var stparam, stvalue : string);

{ Procedure extracts from the string (st) name of the variable (stparam)
  and it's value (stvalue) }

Var i,j,n : integer;

begin
  n := length (st);
  i := 0;
  repeat
    i := i+1;
  until ((st[i] <> ' ') or (i >= n));
  j := i;
  repeat
    j := j+1;
  until ((st[j] = ' ') or (st[j] = '=') or (j >= n));
  stparam := copy (st,i,j-i);
  repeat
    j := j+1;
  until (((st[j] <> ' ') and (st[j] <> '=')) or (j >= n));
```

```
  i := j;
  repeat
    i := i+1;
  until ((st[i] = ' ') or (st[i] = ';') or (st[i] = '{')
                       or (st[i] = ',') or (i >= n));
  if (i < n) then stvalue := copy (st,j,(i-j))
             else stvalue := copy (st,j,(n-j+1));
  for i := 1 to length (stparam) do
    stparam[i] := upcase (stparam[i]);

end;  { Procedure Extract }



Procedure Valb (st : string;
                Var result : boolean;
                ierr : integer);

{ Procedure writes in result the logical value of the string st. Allowed
  values are (the same for small and big characters): 'true', 't', '1', 'y'
  and 'false', 'f', '0', 'n' }

Var i : integer;

Begin
  ierr := 0;
  for i := 1 to length (st) do
    st[i] := upcase (st[i]);
  if ((st = 'TRUE') or (st = 'T') or (st = '1') or (st = 'Y')) then
    result := TRUE
  else if ((st = 'FALSE') or (st = 'F') or (st = '0') or (st = 'N')) then
    result := FALSE
  else ierr := 1;

end;  { Procedure Valb }



Procedure Get_param (stparam, stvalue : string;
                     Var i, n_reg : integer;
                     Var Ts : real;
                     Var Incr_manual, Kp, Ti, Td, Tf, w, Umax, Umin,
                     Vlim, beta, gamma, elim, Ka, Blmin, Blhyst,
                     InRegold, Filt_t, Filt_Diff, Filt_mean_Diff : real_array;

                     Var pb, ib, db, vb, zb, hb, Filtb,
                     Filt_automb, mb : bool_array);


{ Procedure puts the value which is saved in stvalue into the variable
  named stparam }

Var ierr : integer;

begin
  if stparam='CONTROLLER' then val (stvalue,i,ierr)
  else if stparam='N_REG' then val (stvalue,n_reg,ierr)
  else if stparam='TS' then val (stvalue,Ts,ierr)
  else if stparam='INCR_MANUAL' then val (stvalue,Incr_manual[i],ierr)
  else if stparam='KP' then val (stvalue,Kp[i],ierr)
  else if stparam='TI' then val (stvalue,Ti[i],ierr)
  else if stparam='TD' then val (stvalue,Td[i],ierr)
  else if stparam='W' then val (stvalue,w[i],ierr)
  else if stparam='UMAX' then val (stvalue,Umax[i],ierr)
  else if stparam='UMIN' then val (stvalue,Umin[i],ierr)
  else if stparam='VLIM' then val (stvalue,Vlim[i],ierr)
  else if stparam='BETA' then val (stvalue,beta[i],ierr)
  else if stparam='GAMMA' then val (stvalue,gamma[i],ierr)
  else if stparam='ELIM' then val (stvalue,elim[i],ierr)
  else if stparam='KA' then val (stvalue,Ka[i],ierr)
  else if stparam='BLMIN' then val (stvalue,Blmin[i],ierr)
  else if stparam='BLHYST' then val (stvalue,Blhyst[i],ierr)
  else if stparam='INREGOLD' then val (stvalue,Inregold[i],ierr)
```

```
  else if stparam='TF' then val (stvalue,Tf[i],ierr)
  else if stparam='FILT_T' then val (stvalue,Filt_t[i],ierr)
  else if stparam='FILT_DIFF' then val (stvalue,Filt_Diff[i],ierr)
  else if stparam='FILT_MEAN_DIFF' then val (stvalue,Filt_mean_Diff[i],ierr)
  else if stparam='PB' then valb (stvalue,pb[i],ierr)
  else if stparam='IB' then valb (stvalue,ib[i],ierr)
  else if stparam='DB' then valb (stvalue,db[i],ierr)
  else if stparam='VB' then valb (stvalue,vb[i],ierr)
  else if stparam='ZB' then valb (stvalue,zb[i],ierr)
  else if stparam='HB' then valb (stvalue,hb[i],ierr)
  else if stparam='FILTB' then valb (stvalue,Filtb[i],ierr)
  else if stparam='FILT_AUTOMB' then valb (stvalue,Filt_Automb[i],ierr)
  else if stparam='MB' then valb (stvalue,mb[i],ierr);

end;  { Procedure Get_Param }


Procedure Read_file (stfile : string;
                     Dim_Array : integer;
                     Var n_reg : integer;
                     Var Ts : real;
                     Var Incr_manual, Kp, Ti, Td, K_Tf, w, Umax, Umin, Vlim,
                     beta, gamma, elim, Ka, Bl_Ampl, Blmin, Blhyst, Filt_k,
                     InRegold, InReg_Mean, InRegf_Mean, wold, uold, u, uc,
                     x1old, x2old, vold, Tf, Filt_t, OutReg, Filt_Diff,
                     Filt_mean_Diff : real_array;

                     Var Bl_direct, Auto_Catch_Direct : integ_array;

                     Var pb, ib, db, vb, zb, hb, Bl_Catchb, Filtb,
                     Filt_automb, Auto_catchb, mb : bool_array;

                     Var InReg_Array, InRegf_Array : fi_array);

{ Procedure reads the values of the constants and variables from the file.
  Values whic are not read from the file are got predefined values }

Var txt : text;
    st, stparam, stvalue : string;
    i,j : integer;

Begin
   assign (txt,stfile);
   reset (txt);
   repeat
     readln (txt,st);
     Extract (st,stparam,stvalue);
     Get_param (stparam, stvalue, i, n_reg, Ts, Incr_manual, Kp, Ti, Td, Tf,
                w, Umax, Umin, Vlim, beta, gamma, elim, Ka, Blmin,
                Blhyst, InRegold, Filt_t, Filt_Diff, Filt_mean_Diff,
                pb, ib, db, vb, zb, hb, Filtb, Filt_automb, mb);

   until (Eof (txt));
   close (txt);
   for j := 1 to n_reg do
   begin
     If (Filt_t[j] > Ts/50) then Filt_k[j] := exp (-Ts/Filt_t[j])
                            else Filt_k[j] := 0;
     InReg_Mean[j] := InRegold[j];
     InRegf_Mean[j] := InRegold[j];
     If (Tf[j] > Ts/50) then K_Tf[j] := exp (-Ts/Tf[j])
                        else K_Tf[j] := 0;
     Auto_Catchb[j] := FALSE;
     Auto_Catch_Direct[j] := -1;
     Bl_Ampl[j] := 0;
     wold[j] := 0;
     uold[j] := 0;
     u[j] := 0;
     uc[j] := 0;
     x1old[j] := 0;
     x2old[j] := 0;
     vold[j] := 0;
```

```
      Bl_Direct[j] := 1;
      Bl_Catchb[j] := TRUE;
      OutReg[j] := 0;
      for i := 1 to Dim_Array do
      begin
        InReg_Array [j,i] := InRegold [j];
        InRegf_Array [j,i] := InRegold [j];
      end;
    end;

end;  { Procedure Read_file }


Procedure Write_file (stfile : string;
                      n_reg : integer;
                      Ts : real;
                      Incr_manual, Kp, Ti, Td, Tf, w, Umax, Umin, Vlim,
                      beta, gamma, elim, Ka, Blmin, Blhyst, Filt_k,
                      InRegold, Filt_t, Filt_Diff, Filt_mean_Diff : real_array;

                      pb, ib, db, vb, zb, hb, Filtb, Filt_automb,
                      mb : bool_array);

{ Procedure writes some variables to the file }

Var txt : text;
    i : integer;

Begin
  Writeln;
  assign (txt,stfile);
  rewrite (txt);
  writeln (txt,'N_REG = ',n_reg);
  writeln (txt,'TS = ',Ts);
  for i := 1 to n_reg do
    begin
      writeln (txt);
      writeln (txt,'*****************************************************');
      writeln (txt,'                   CONTROLLER = ',i);
      writeln (txt,'*****************************************************');
      writeln (txt);
      writeln (txt,'***************** BASIC SETTINGS ******************');
      writeln (txt);
      writeln (txt,'PB = ',Pb[i],'  ; Proportional part');
      writeln (txt,'IB = ',Ib[i],'  ; Integral part');
      writeln (txt,'DB = ',Db[i],'  ; Diferential part');
      writeln (txt,'KP = ',Kp[i],'  ; Proportional gain');
      writeln (txt,'TI = ',Ti[i],'  ; Integral time constant');
      writeln (txt,'TD = ',Td[i],'  ; Diferential time constant');
      writeln (txt,'TF = ',Tf[i],'  ; Time constant of the dif. filter');
      writeln (txt,'W = ',w[i],'  ; Reference');
      writeln (txt);
      writeln (txt,'******************** LIMITS ***********************');
      writeln (txt);
      writeln (txt,'UMAX = ',Umax[i],'  ; Max. output of the controller');
      writeln (txt,'UMIN = ',Umin[i],'  ; Min. output of the controller');
      writeln (txt,'VLIM = ',Vlim[i],
                   '  ; Velocity limit at the output (V/sample)');
      writeln (txt,'INCR_MANUAL = ',Incr_manual[i],
                   '  ; Increment at the output for man. contr. (V/sample)');
      writeln (txt);
      writeln (txt,'************ OVERSHOOT & ANTI-WINDUP ***************');
      writeln (txt);
      writeln (txt,'BETA = ',beta[i],'  ; Beta factor for the P-part');
      writeln (txt,'GAMMA = ',gamma[i],'  ; Gamma factor for the D-part');
      writeln (txt,'ELIM = ',elim[i],'  ; Limiter at the input of the I-part');
      writeln (txt,'KA = ',Ka[i],'  ; Anti-windup constant');
      writeln (txt);
      writeln (txt,'************* OTHER INPUTS & CONTROL ***************');
      writeln (txt);
      writeln (txt,'VB = ',Vb[i],'  ; Aditive input (feed forward)');
      writeln (txt,'ZB = ',Zb[i],'  ; Multiplicative input');
```

```
      writeln (txt,'HB = ',Hb[i],'  ; Hold function (stop integrator)');
      writeln (txt,'MB = ',Mb[i],'  ; Manual control');
      writeln (txt);
      writeln (txt,'****************** HYSTERESIS ******************');
      writeln (txt);
      writeln (txt,'BLHYST = ',Blhyst[i],'  ; Amplitude of the hysteresis');
      writeln (txt,'BLMIN = ',Blmin[i],
                   '  ; Change of the output for adding Blhyst');
      writeln (txt);
      writeln (txt,'***************** INPUT FILTER *****************');
      writeln (txt);
      writeln (txt,'FILT_AUTOMB = ',Filt_Automb[i],
                   '  ; Automatic switching on/off the input filter');
      writeln (txt,'FILTB = ',Filtb[i],
                   '  ; Initial state of the input filter');
      writeln (txt,'FILT_T = ',Filt_t[i],
                   '  ; Time constant of the input filter');
      writeln (txt,'INREGOLD = ',Inregold[i],
                   '  ; Initial value of the input signal');
      writeln (txt,'FILT_DIFF = ',Filt_Diff[i],
                   '  ; Value for autom. switching on/off the input filter');
      writeln (txt,'FILT_MEAN_DIFF = ',Filt_mean_Diff[i],
                   '  ; Value for autom. switching on/off the input filter');
      writeln (txt);
      writeln (txt);
    end;
  close (txt);

end;  { Procedure Write_File }


Procedure Param_Copy (n_reg : integer;
                      Var Kp, Ti, Td, K_Tf, w, Umax, Umin, Vlim,
                      beta, gamma, elim, Ka, Bl_Ampl, Blmin, Blhyst,
                      Filt_k, InRegold, InReg_Mean, InRegf_Mean, wold, uold,
                      u, uc, x1old, x2old, vold, Tf, Filt_t, OutReg,
                      Incr_Manual, Filt_Diff, Filt_mean_Diff : real_array;

                      Var Bl_direct, Auto_Catch_Direct : integ_array;

                      Var pb, ib, db, vb, zb, hb, Bl_Catchb, Filtb,
                      Filt_automb, Auto_catchb, mb : bool_array;

                      Var InReg_Array, InRegf_Array : fi_array);

{ Procedure is used for copying the variables from the first controller to the
  other controllers if All_Regb = TRUE }

Var i,j : integer;

Begin

For i := 2 to n_reg do
begin

{ Basic PID settings }

  pb[i] := pb[1];
  ib[i] := ib[1];
  db[i] := db[1];
  Kp[i] := Kp[1];
  Ti[i] := Ti[1];
  Td[i] := Td[1];
  Tf[i] := Tf[1];
  K_Tf[i] := K_Tf[1];
  w[i] := w[1];

{ Limits }

  Umax[i] := Umax[1];
  Umin[i] := Umin[1];
  Vlim[i] := Vlim[1];
  Incr_Manual[i] := Incr_Manual[1];
```

```
{ Overshoot rejection }

  beta[i] := beta[1];
  gamma[i] := gamma[1];
  elim[i] := elim[1];

{ Anti windup }

  Ka[i] := Ka[1];

{ Aditional inputs }

  vb[i] := vb[1];
  zb[i] := zb[1];
  hb[i] := hb[1];
  mb[i] := mb[1];

{ Hysteresis - (Back-lash) }

  Bl_Catchb[i] := Bl_Catchb[1];
  Bl_Direct[i] := Bl_Direct[1];
  Bl_Ampl[i] := Bl_Ampl[1];
  Blmin[i] := Blmin[1];
  Blhyst[i] := Blhyst[1];

{ Input filter }

  Filtb[i] := Filtb[1];
  Filt_automb[i] := Filt_automb[1];
  Filt_t[i] := Filt_t[1];
  Filt_k[i] := Filt_k[1];
  InRegold[i] := InRegold[1];
  for j := 1 to 10 do
  begin
    InReg_Array [i,j] := InReg_Array [1,j];
    InRegf_Array [i,j] := InRegf_Array [1,j];
  end;
  InReg_Mean[i] := InReg_Mean[1];
  InRegf_Mean[i] := InRegf_Mean[1];
  Filt_Diff[i] := Filt_Diff[1];
  Filt_mean_Diff[i] := Filt_mean_Diff[1];

{ Automatic Manual-Automatic bumpless switching }

  Auto_catchb[i] := Auto_Catchb[1];
  Auto_catch_direct[i] := Auto_catch_direct[1];

{ Other settings }

  wold[i] := wold[1];
  uold[i] := uold[1];
  u[i] := u[1];
  uc[i] := uc[1];
  x1old[i] := x1old[1];
  x2old[i] := x2old[1];
  vold[i] := vold[1];
  OutReg[i] := OutReg[1];

end;

end;  { Procedure Param_Copy }



Procedure In_Param (Dim_Array : integer;
                    Var All_regb : boolean;
                    Var n_reg, Active_Reg : integer;
                    Var Ts : real;
                    Var Init_Constb : boolean;
                    Var Incr_manual, Kp, Ti, Td, K_Tf, w, Umax, Umin, Vlim,
                    beta, gamma, elim, Ka, Bl_Ampl, Blmin, Blhyst, Filt_k,
```

```
                    InRegold, InReg_Mean, InRegf_Mean, wold, uold, u, uc,
                    x1old, x2old, vold, Tf, Filt_t, OutReg, Filt_Diff,
                    Filt_mean_Diff : real_array;

                    Var Bl_direct, Auto_Catch_Direct : integ_array;

                    Var pb, ib, db, vb, zb, hb, Bl_Catchb, Filtb,
                    Filt_automb, Auto_catchb, mb : bool_array;

                    Var InReg_Array, InRegf_Array : fi_array);

{ Procedure inputs the parameters of the controllers - by keyboard or by
  file }

Var i,k : integer;
    In_File, Out_file : string;
    ch1 : char;
    tmpb : boolean;

begin
  Active_Reg := 1;
  Writeln;
  Writeln ('Fill in parameters directly ..... 1');
  Writeln ('Parameters are on the file ..... 2');
  Writeln;
  Repeat
    k := Iscan;
  Until ((k = 49) or (k = 50));
  If (k = 49) then
  begin
    In_Initial (n_reg, All_Regb);
    i := 1;
    repeat
      Writeln;
      If (All_Regb and (n_reg > 1)) then
        Writeln ('***** CONTROLLERS 1..',n_reg,' *****')
      else Writeln ('***** CONTROLLER ',i,' *****');
      Writeln;
      In_Boolean ('Do you want predefined values of the parameters (N) : ',
                  FALSE,Init_Constb);
      In_Basic (Init_Constb, pb[i], ib[i], db[i], Kp[i], Ti[i], Td[i], Tf[i],
                   Ts, K_Tf[i], w[i]);
      Writeln;
      In_Limits (Init_Constb, Ts, Umax[i], Umin[i], Vlim[i], Incr_manual[i]);
      Writeln;
      In_Overshoot_AW (Init_Constb, Kp[i], beta[i], gamma[i],
                           elim[i], Ka[i]);
      Writeln;
      In_Aditional_Inputs (Init_Constb, vb[i], zb[i], hb[i], mb[i]);
      Writeln;
      In_Hysteresis (Init_Constb, Bl_Catchb[i], Bl_Direct[i], Bl_Ampl[i],
                       Blmin[i], Blhyst[i]);
      Writeln;
      In_Filter (Init_Constb, Ts, Td[i], Filt_automb[i], Filtb[i], Filt_t[i],
                 Filt_k[i], InRegold[i], InReg_Mean[i], InRegf_Mean[i],
                 Filt_Diff[i], Filt_mean_Diff[i], InReg_Array[i],
                 InRegf_Array[i]);
      Writeln;
      In_Manual_Automatic (Init_Constb, Auto_Catchb[i], Auto_Catch_Direct[i]);
      Writeln;
      In_Other (wold[i], uold[i], u[i], uc[i],
                  X1old[i], x2old[i], vold[i], OutReg[i]);
      if (All_Regb) then
        Param_Copy (n_reg, Kp, Ti, Td, K_Tf, w, Umax,
             Umin, Vlim, beta, gamma, elim, Ka, Bl_Ampl,
             Blmin, Blhyst, Filt_k, InRegold, InReg_Mean,
             InRegf_Mean, wold, uold, u, uc, x1old, x2old,
             vold, Tf, Filt_t, OutReg, Incr_Manual, Filt_Diff,
             Filt_mean_Diff, Bl_direct, Auto_Catch_Direct,
             pb, ib, db, vb, zb, hb, Bl_Catchb,
             Filtb, Filt_automb, Auto_catchb, mb,
             InReg_Array, InRegf_Array);
```

```
      i := i+1;
    Until ((i > n_reg) or All_Regb);
    Writeln;
    Writeln;
    In_Boolean ('Write new parameters into the file ? (Y)  ',TRUE,tmpb);
    If (tmpb) then
    begin
      Write ('Fill in the name of the file : ');
      Readln (Out_file);
      Write_file (Out_file, n_reg, Ts, Incr_manual, Kp, Ti, Td, Tf, w, Umax,
                  Umin, Vlim, beta, gamma, elim, Ka, Blmin, Blhyst,
                  Filt_k, InRegold, Filt_t, Filt_Diff, Filt_mean_Diff,
                  pb, ib, db, vb, zb, hb, Filtb, Filt_automb, mb);
    end;

  end
  else
  begin
    Writeln;
    Write ('Enter the name of the file : ');
    Readln (In_File);
    Read_file (In_File, Dim_Array, n_reg, Ts, Incr_manual, Kp, Ti, Td, K_Tf,
               w, Umax, Umin, Vlim, beta, gamma, elim, Ka, Bl_Ampl, Blmin,
               Blhyst, Filt_k, InRegold, InReg_Mean, InRegf_Mean, wold, uold,
               u, uc, x1old, x2old, vold, Tf, Filt_t, OutReg, Filt_Diff,
               Filt_mean_Diff, Bl_direct, Auto_Catch_Direct,
               pb, ib, db, vb, zb, hb, Bl_Catchb, Filtb,
               Filt_automb, Auto_catchb, mb, InReg_Array, InRegf_Array);
  end;

end;  { Procedure In_Param }



Procedure Filtering (n_reg, Dim_Array : integer;
                     y, Filt_k : real_array;
                     Var InReg_Mean, InRegf_Mean, InReg, InRegold,
                     Filt_Diff, Filt_mean_Diff : real_array;
                     Var InReg_Array, InRegf_Array : fi_array;
                     Filt_automb : bool_array;
                     Var Filtb : bool_array);

{ Procedure executes procedure Filter and function Switch_Filter. If the
  input filter switches on, input value to the controller (InReg) becomes
  the filtered input value (filtered y) }

Var i : integer;

Begin
  for i:= 1 to n_reg do
  begin
    Filter (InReg_Mean[i], InRegf_Mean[i], InReg_Array[i], InRegf_Array[i],
            y[i], Filt_k[i], Dim_Array);
    if (Filt_automb[i]) then
      Filtb[i] := Switch_filter (InReg_Mean[i], InRegf_Mean[i],
                                 Filt_Diff[i], Filt_mean_Diff[i],
                                 Dim_Array, InRegf_Array[i]);
    If (Filtb[i]) then
    begin
      InReg[i] := InRegf_Array[i,1];
      InRegold[i] := InRegf_Array[i,2];  { Because of the D-part }
    end
    else
    begin
      InReg[i] := InReg_Array[i,1];
      InRegold[i] := InReg_Array[i,2];
    end;
  end;

end;  { Procedure Filtering }
```

```
Procedure Blash (Var Bl_Ampl, uold : real;
                 Var Bl_Catchb : boolean;
                 Var Bl_Direct : integer;
                 du, Umax, Umin, Blhyst, Blmin : real;
                 mb : boolean);

{ Procedure is used for controlling the hysteresis in the processes. According
  to the output value of the controller, adds or substracts the value Blhyst
  from the variable uold }

Var Added_Value, temp : real;

begin
  If (not mb) then
  begin
    Added_Value := 0;
    temp := uold+du;
    if (Limiter (temp,Umax,Umin) = temp) then
    begin
      if (Bl_Catchb) and (Sgn(du) <> Bl_Direct) then
      begin
        Bl_Ampl := uold;
        Bl_Catchb := FALSE;
        Bl_Direct := Sgn (du);
      end;
      if (not Bl_Catchb) then
      begin
        if (Bl_Direct = -1) then
        begin
          if (Bl_Ampl - (uold+du)) > Blmin then
          begin
            Added_Value := -Blhyst;
            if (uold+du+Added_Value) < Umin then Added_Value := Umin-uold-du;
            Bl_Catchb := TRUE;
          end
          else if (Bl_Ampl - (uold+du)) < 0 then Bl_Catchb := TRUE;
        end
        else if (Bl_Direct = 1) then
        begin
          if (Bl_Ampl - (uold+du)) < Blmin then
          begin
            Added_Value := Blhyst;
            if (uold+du+Added_Value) > Umax then Added_Value := Umax-uold-du;
            Bl_Catchb := TRUE;
          end
          else if (Bl_Ampl - (uold+du)) > 0 then Bl_Catchb := TRUE;
        end;
      end;
    end;
    uold := uold + Added_Value;
  end;

end; { Procedure Blash }



Procedure Controller (Kp, Ka, Ti, Td, K_Tf, Ts, beta, gamma, w,
                      elim, Vlim, InReg, v, z, Blhyst, Blmin,
                      Umin, Umax : real;
                      pb, ib, db, vb, zb, mb, hb : boolean;
                      Var Bl_Catchb : boolean;
                      Var Bl_Direct : integer;
                      Var uc, InRegold, x1old, x2old, uold, u, vold,
                      OutReg, Bl_Ampl, wold : real);

{ Procedure contains the control algorithm of the one controller. According
  to the input values and internal states, calculates the output value of the
  controller (OutReg) }

Var dy, du, duc, dv, corr, x1, x2, e, ei, el : real;
```

```
begin
  If (db) then     { If D-part is switched on }
  begin
    dy := gamma*(w-wold)-InReg+InRegold;
    x1 := K_Tf*x1old + Td*(1-K_Tf)*dy/Ts;   { D-part's filter }
  end
  else x1 := 0;
  x1old := x1;
  corr := (1-beta)*(wold-w);   { Correction factor for beta }
  x2 := x1+w-InReg;            { D and P part togeather }
  du := x2-x2old;              { Due to incremental algorithm }
  x2old := x2;
  if (not pb) then du := 0;        { If there is only integrator }
  e := (w+wold-InReg-InRegold)/2;  { Bilinear integration }
  wold := w;
  InRegold := InReg;

  If (pb) then     { PID, PI, PD and P controller }
  begin
    If (not zb) then el := e-(u-uc)/Ka        { Anti-windup }
                else el := e-(u-uc)/(Ka*z);   { If multiplicative input }
  end
  else el := e;                       { I controller }
  el := Limiter (el,elim,-elim);    { Limiter at the input of the integrator }

  If (ib and (not hb)) then ei := el*Ts/Ti  { Integration }
                       else ei := 0;        { Hold on or P, PD controller }
  If (pb) then du := (du+ei+corr)*Kp        { Adding D, P and I part }
          else du := ei;                    { I controller }
  If (zb) then du := du*z;                  { Multiplicative input }
  If (vb) then                              { Aditive input }
  begin
    If (mb) then dv := 0                    { If manual control }
            else dv := v - vold;
    vold := v;
    du := du + dv;
  end;

  Blash (Bl_Ampl, uold, Bl_Catchb, Bl_Direct,
         du, Umax, Umin, Blhyst, Blmin, mb);  { Blash function for hysteresis
                                                compensation }

  u := uold + du;     { Not-limited output }
  uold := u;

  duc := u-uc;
  duc := Limiter (duc,Vlim,-Vlim);   { Velocity limit }
  uc := uc+duc;
  uc := Limiter (uc,Umax,Umin);      { Amplitude limit - limited variable }

  If (not pb) then uold := uc;  { Anti windup for only I-part }

  If (mb) then uc := OutReg
          else OutReg := uc;    { Output of the controller }

end;  { Procedure controller }


Procedure Regulator ( Ts : real;
                      n_reg, Active_reg, Code : integer;
                      Incr_manual, Kp, Ka, Ti, Td, K_Tf, beta, gamma, w,
                      elim, Vlim, InReg, v, z, Blhyst, Blmin,
                      Umin, Umax : real_array;
                      pb, ib, db, vb, zb, mb, hb : bool_array;
                      Var Bl_Catchb : bool_array;
                      Var Bl_Direct : integ_array;
                      Var uc, InRegold, x1old, x2old, uold, u, vold,
                      OutReg, Bl_Ampl, wold : real_array);

{ Procedure regulator calculates all the outputs of the n_reg controllers }

Var i : integer;
```

```
Begin
  for i := 1 to n_reg do
  begin
    Manual_Out (Code, Bl_Direct[i], Bl_Catchb[i], mb[i], Active_reg,
                i, Umax[i], Umin[i], Incr_manual[i], OutReg[i]);
    controller (Kp[i], Ka[i], Ti[i], Td[i], K_Tf[i], Ts, beta[i], gamma[i],
                w[i], elim[i], Vlim[i], InReg[i], v[i], z[i], Blhyst[i],
                Blmin[i], Umin[i], Umax[i], pb[i], ib[i], db[i], vb[i],
                zb[i], mb[i], hb[i], Bl_Catchb[i], Bl_Direct[i], uc[i],
                InRegold[i], x1old[i], x2old[i], uold[i], u[i], vold[i],
                OutReg[i], Bl_Ampl[i], wold[i]);

  end;

end;  { Procedure Regulator }


end.
```