An Interactive MISO Regulator

Andersson, Leif; Åström, Karl Johan

1978

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT-7154)/1-034/(1978)

AN INTERACTIVE MISO REGULATOR

LEIF ANDERSSON
KARL JOHAN ÅSTRÖM

Department of Automatic Control
Lund Institute of Technology
October 1978

AN INTERACTIVE MISO REGULATOR

Leif Andersson
Karl Johan Åström

October 1978
Revised October 1980

Dokumentutgivare
Lund Institute of Technology
Handläggare Dept of Automatic Control
Åström Andersson
Författare
Åström Andersson
Karl Johan Åström

Dokumentnamn
REPORT
Dokumentbeteckning
LUTFD2/(TFRT-7154)/1-034)/1978
Utgivningsdatum
Okt 1978
Ärendebeteckning

Dokumenttitel och undertitel
AN INTERACTIVE MISO REGULATOR

Referat (sammandrag)
This report describes software for an interactive linear multiple-input single-output regulator implemented on LSI/11 under the operating system RT/11. The software is intended for educational and industrial experiments. The interactive approach allows good man-machine interaction. It makes the system easy to learn and easy to use. The software can also be used as case studies in courses on software for computer control. The program is written in PASCAL.

Referat skrivet av
authors

Förslag till ytterligare nyckelord

Klassifikationssystem och -klass(er)

Indextermer (ange källa)

Omfång
34 pages

Övriga bibliografiska uppgifter

Språk
English

Sekretessuppgifter

ISSN

ISBN

Dokumentet kan erhållas från
Department of Automatic Control
Lund Institute of Technology
P O Box 725, S-220 07 LUND 7, SWEDEN

Mottagarens uppgifter

Pris

DOKUMENTDATABLAD enligt SIS 62 10 12

SIS-DB 1

Blankett LU 11:25 1976–07

TABLE OF CONTENTS

1. INTRODUCTION

The program described in this paper was developed for three main reasons

- Need for a flexible system to allow students hands on experience with digital control for courses in sampled data and stochastic control.

- Need for a flexible system for digital process control experiments.

- Need for examples of process control software for courses on software for computer control.

The hardware facilities consist of four LSI-11 systems. Each system has a process interface, a simple alphanumeric display, and a dual diskette unit for program storage and program development. A detailed description of the hardware and software is given in [1].

Both for educational and industrial uses it is highly desirable to have a system which is easy to learn and use. It is also highly desirable to have a system which hides unnecessary and trivial details. This strongly points to an interactive system based on the use of an alphanumeric video display terminal with forms capability. The display is organized so that it is more or less self-explanatory. The configuration of the system and the parameters can also be changed easily using the display. When the regulator is running the terminal can be connected or disconnected without influencing the system.

During the development of the system there have been substantial discussions concerning suitable programming languages. Possible candidates that have been discussed are FORTRAN, PASCAL and BASIC. In connection with this work both PASCAL and FORTRAN were investigated. The control algorithms were coded in both languages. The programs and the compiled code were investigated. It was found that there were considerable advantages by using PASCAL. The compiled code was shorter. The PASCAL program was also easier to read mainly due to the availability of data structures. The advantages with PASCAL were even more pronounced when it came to operator communication. The features which were implemented would undoubtedly have required a much larger programming effort if they had been written in FORTRAN.

2. THE MISO REGULATOR

The MISO regulator and the associated algorithms will be described in this section. The equations are first described and a realization is then given. Various practical issues like reset windup, initialization and parameter changes are then discussed. A logging facility is also included in the regulator. This is also briefly discussed. Some features which were desirable for the special teaching situation are discussed together with some additional features that would be desirable to include.

REGULATOR ALGORITHM

The MISO regulator is charaterized by the following equations

$$\begin{aligned} & u(t) + r_1 u(t-1) + \ldots + r_n u(t-n) = \\ & t_0 y_r(t) + t_1 y_r(t-1) + \ldots + t_n y_r(t-n) \\ & - s_{10} y_1(t) - s_{11} y_1(t-1) - \ldots - s_{1n} y_1(t-n) \\ & - s_{20} y_2(t) - s_{21} y_2(t-1) - \ldots - s_{2n} y_2(t-n) \\ & \cdot \\ & \cdot \\ & \cdot \\ & - s_{m0} y_m(t) - s_{m1} y_m(t-1) - \ldots - s_{mn} y_m(t-n) \end{aligned} \tag{1}$$

where u is the control signal, i.e. the input signal to the process, y_r is the reference or the command signal, y_1 is the controlled process output, y_2, y_3, ... y_m are auxiliary measured process signals. The signals y_2, ... , y_m are typically measured disturbances used for feedforward. The regulator (1) can be written in the following shorthand notation

$$Ru = Ty_r - S_1 y_1 - S_2 y_2 - \ldots - S_n y_n \tag{2}$$

where R_1, S_1, S_2, ... S_m and T are polynomials in the backward shift operator.

The algorithm (1) can of course also represent a state feedback. In that case R, S_1, ... , S_m and T are simply constants.

A MINIMAL STATE REPRESENTATION

The control algorithm (1) can of course be coded according to the formula (1). This means implicitly that the state variables are chosen as past values of inputs and outputs. This is not a minimal state. It is of course not necessary to use a minimal realization of the dynamical system. It is, however, useful for pedagogical purposes to exhibit the regulator state explicitly and then we might just as well use a minimal state. The following observable canonical form is chosen

$$x(t+1) = \begin{bmatrix} -a_1 & 1 & 0 & \dots & 0 \\ -a_2 & 0 & 1 & \dots & 0 \\ \vdots & & & & \\ -a_{n-1} & 0 & 0 & \dots & 1 \\ -a_n & 0 & 0 & \dots & 0 \end{bmatrix} x(t) + \begin{bmatrix} b_{10} & b_{11} & \dots & b_{1m} \\ b_{20} & b_{21} & \dots & b_{2m} \\ \vdots & & & \\ b_{n-10} & b_{n-21} & \dots & b_{n-2m} \\ b_{n0} & b_{n1} & \dots & b_{nm} \end{bmatrix} y(t) \qquad (3)$$

$$u(t) = x_1(t) + d_0 y_0(t) + d_1 y_1(t) + \dots + d_m y_m(t) \qquad (4)$$

where

$$\begin{aligned} a_i &= r_i && i = 1, \dots, n \\ b_{i0} &= t_i - t_0 r_i && i = 1, \dots, m \\ b_{ij} &= -s_{ji} + t_0 r_i && i = 1, \dots, n;\ j = 1, \dots, m \qquad (5) \\ d_0 &= t_0 \\ d_i &= -s_{i0} && i = 1, \dots, m \end{aligned}$$

It is desirable to reduce the computational delays as much as possible. Using the realization (3) and (4) it follows that to determine the control signal $u(t)$ at time t, the state $x_1(t)$ can be precomputed based on measurements available at time $t - 1$. If this is done it is only necessary to do the scalar product in (4) when the measurements $y_0(t), y_1(t), \ldots, y_m(t)$ are obtained. The calculation of the control signal $u(t)$ from measurements $y_0(t), y_1(t), \ldots, y_m(t)$ and the state $x_1(t)$ is done in the procedure *output*. Updating the state according to (3) is carried out by the procedure *update*.

LIMITING OF REGULATOR OUTPUT

The control law given by (3) and (4) is a linear dynamical system. It has been found empirically that it is often useful to add some nonlinear functions. Typical examples are limits on the control signal and its rate of change. An actuator typically has both a limited range and a limited speed. Another feature is to introduce a dead zone so that small changes in the computed control will not generate any changes in the actuators. It was judged useful for pedagogical purposes to include some of these features. Only the saturation was included. The control signal is thus computed as

$$u(t) = \text{sat}\,[x_1(t) + dy(t)]$$

where

$$\text{sat } x = \begin{cases} \text{low} & \text{if } x \leq \text{low} \\ x & \text{if low} < x < \text{high} \\ \text{high} & \text{if } x \geq \text{high} \end{cases}$$

RESET WINDUP

In process control there is often a need to run a process under manual control. This means that the actual control variable may assume values that are different from those generated by the control algorithm. Since the regulator is a dynamical system care must be taken

to determine the state of the regulator which is compatible with the past sequence of inputs and outputs. The regulator is described by

$$x(t+1) = \begin{bmatrix} -a_1 & 1 & 0 & \dots & 0 \\ -a_2 & 0 & 1 & \dots & 0 \\ \vdots & & & & \\ -a_{n-1} & 0 & 0 & \dots & 1 \\ -a_n & 0 & 0 & \dots & 0 \end{bmatrix} x(t) + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix} y(t)$$

$$u(t) = x_1(t) + dy(t)$$

A simple dead-beat observer is obtained as follows

$$\begin{aligned} x_1(t) &= u(t) - dy(t) \\ x_n(t) &= -a_n x_1(t-1) + b_n y(t-1) \\ x_{n-1}(t) &= -a_{n-1} x_1(t-1) + x_n(t-1) + b_{n-1} y(t-1) \\ &\vdots \\ x_2(t) &= -a_2 x_1(t-1) + x_3(t-1) + b_{n-2} y(t-1) \end{aligned}$$

It is clearly required to use past values of the input and output to determine x(t).

The observer problem is simpified a little if it is assumed that the process is in steady state. Let the measured variable be y^o and the control variable u^o. The state is then given by

$$\begin{aligned} x_1^o &= u^o - dy^o \\ x_2^o &= (1+a_1)x_1^o - b_1^o y^o \\ x_3^o &= a_2 x_1^o + x_2^o - b_2^o y^o \\ &\vdots \\ x_n^o &= a_{n-1} x_1^o + x_{n-1}^o - b_{n-1}^o y^o \end{aligned}$$

to avoid that the state of the regulator drifts away. If this is not done large excursions may occur when the regulator is switched from manual to automatic mode. A similar situation may occur if the actuator saturates. The phenomena are called *reset windup* in the process control literature. Reset windup due to actuator saturation can be avoided if the regulator output is limited to bounds tighter than the actuator saturation and if the regulator state is reset when the regulator saturates.

If the control variable computed by the regulator is outside the interval (low, high) the state is reset to a value such that

$$x_1^+(t) + dy(t) = \text{high or low}$$

This is equivalent to the following input-output description of the regulator

$$u(t) = \text{sat}[Ty_r - S_1y_1 - \ldots - S_my_m - (R-1)u]$$

Notice that this is different from

$$u(t) = \text{sat}\ (Ty_r - S_1y_1 - \ldots - S_my_m)/R$$

This way of avoiding reset windup is easily coded. The code is given in the procedure *output*. Notice that the simplicity of the code depends on the chosen realization of the regulator.

If the state is reset in the way described it is also very easy to obtain a bumpless transfer between manual and automatic control.

BUMPLESS PARAMETER CHANGES

It is desirable to be able to change the parameters of the regulator when it is running. If the regulator parameters are changed without changing the state of the regulator there may be unnecessary changes in the process. When the parameters are changed it is therefore useful to reset the regulator state. This is an observer problem i.e.

If the
If the state is reset in this way it is easy to see that the control signal will remain constant after a parameter change. The code for resetting the state is given in the procedure *init*.

LOGGING AND ANALYSIS

It is useful when testing the system to have a running log of inputs and outputs. A simple algorithm which stores the 20 past values of inputs and outputs has therefore been included.

For educational applications it is useful to include performance analysis. There is therefore a simple algorithm to compute overshoot, rise time and settling time.

SPECIAL FEATURES

In an educational application it is very instructive to show regulator input and output on an oscilloscope. To obtain nice displays a trig signal is provided. The trig signal is synchronized to the sampling instants. It has the same sign as the reference signal.
It is useful to be able to take the inputs from other sources than the standard A/D converter. The following convention has therefore been adopted: Input channel numbers < = 15 correspond to analog input. Channel numbers > 15 correspond to digital input channel (number -16).

DESIRED ADDITIONAL FEATURES

It would be desirable to include a panel for manual control with corresponding changes in the software. It could also be useful to include a rate limitation of the control signal and a control bias. If a sufficiently long sampling period is used it would also be useful to have a logging mode which writes the measurements and the calculated control signal continuously.

3. OPERATOR COMMUNICATION

It was highly desirable to have a flexible operator communication. The system should be easy to learn and easy to use. There should be good facilities to connect the system to a process, to change the connections, the structure and the parameters of the regulator.

The alphanumeric display has function keys and the features "block send" and "protected areas". A form which contains all relevant information about the regulator is displayed in protected mode. This means that the form can neither be erased nor changed during the operation. The parameters of the regulator are displayed on the form in non-protected mode. The form is shown in Fig 1 with the changeable parameters underlined.

```
DISCRETE MULTI-INPUT-SINGLE-OUTPUT REGULATOR

     R(q) * u := T(q) * yr - S1(q) * y1 - S2(q) * y2 ...

STRUCTURE AND CONNECTIONS
     Number of measurements   ny = 2
     Degree of polynomials    nx = 1
     Variable/channel    u/ 0   yr/ 0   y1/ 1   y2/ 2

SAMPLING PERIOD  TS =  1.000

PARAMETERS
     lolim = -1.000
     hilim =  1.000
     Regulator transfer function:
     Coefficient          0          1
     polynomial
        R(q)            1.0      0.150
        T(q)          3.750      0.000
       S1(q)          3.900     -0.150
       S2(q)          1.350     -0.270

                                        RUN
```

Fig 1 - The operator communication form.

To change a number on the screen the operator simply places the cursor in the appropriate position and overwrites the old value. The new values are entered into the regulator algorithm by pushing a function key labeled ENTER (-parameters).

The other keys have the following functions:

SHOW (parameters)
The parameter values are written on the display

RUN/STOP

LOG
The last 20 values of measurement and control variables are written on the display
Show the last 20 values of inputs and outputs

ANALYSIS
Show risetime, overshoot, and settling time

EXIT
Leave the MISO Program

These functions are listed on a removable label which is placed around the function keys. The layout of the label is shown in Fig 2.

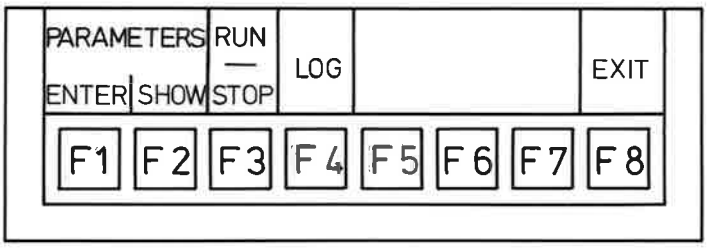

Fig 2 - Label for the function keys.

4. PROGRAMMING DETAILS

Some details of the program are given in this section. An overview is first given. The major programs are then desribed. The storage requirements are then given.

OVERVIEW

Most of the programs are written in PASCAL. A few procedures such as analog input, analog output, and the real-time scheduler are written in assembler. Some of the standard DEC software is also used.

The program is divided into two main tasks, a foreground process which contains the regulator functions, and a background process which allows operator communication.

PROCESS SCHEDULING

The scheduler is a simple "run-to-completion monitor" which allows the background program to be interrupted at the sampling instants. Once the foreground program is entered it runs to completion. This means that the two processes can operate on a common stack, which greatly simplifies context switching. The scheduler accepts the sampling period as a parameter, and by convention the foreground program is not run at all when the period is zero. The sampling period may, however, be changed any time, and then the foreground program is started immediately. The scheduler is described in [2].

MAIN PROGRAM

The main program looks as follows

```
begin
     period: = 0;
     tlog:  = 0;
     logpar:  = true;
     schedule (foreground, period);
     Opcom;
end
```

The variables period, tlog and logpar are initialized. When the scheduler is called with period = 0 it transfers control to Opcom, where the sampling period and the parameters are entered via the display. Once period is positive the scheduler interrupts the background program Opcom and runs foreground at times specified by period.

FOREGROUND PROGRAM

The Foreground program does analog input, calculates the control variable, transfers regulator parameters from the Opcom data area, and logs the inputs and outputs. A synopsis of the program is given below.

```
Procedure Foreground

  Procedure Regulate;
    Function Adin (chan: integer): real;
      {reads analog input}

    Procedure Daout (chan: integer; value: real);
      {sets analog output}

    Function Ushape
      {limits the control variable}

    Procedure Output;
      {computes the control variable from measurement and state}

    Procedure Update;
      {updates the regulator state}

  begin {Regulate}
     for i: = 1 to ny do y[i]: = Adin (i);
     Output;
     Daout (outch, u);
     Update
  end {Regulate};

  Procedure Trigosc;
    {generates a signal to trig the oscilloscope}

  Procedure Parset;
    {changes parameters and initializes regulator state}

  Procedure Loguy;
    {stores the control variable and the measurements in a cyclic file}
```

```
    Procedure Parchange;
      {Transforms the parameters from the polynomial representation
       to the state space form used in the regulator}

    Procedure Wlog;
      {Writes a back-log of the 20 past input and output values}

    Function Readfunc: integer;
      {Waits for the operator to push a function key and returns
       its number}

  begin {Opcom}

    Default; Writeform; Fillform;
    repeat

    case Readfunc of

     1:begin Readscreen; Parchange; Fillform end;

     2:Fillform;

     3:{Change the run/hold status of the regulator}

     4:begin
         repeat Writelog until Readfunc <>4;
         Writeform; Fillform
       end;

     5:begin
         analysisrequest: = true
         repeat until analysisdone
         {write the result}
       end;

     8:exit;

    forever

  end {Opcom};
```

THE DISPLAY DRIVER

The display is a Beehive B100-3 VDU terminal. Apart from sending and receiving text in a simple fashion it has some additional features (see [3]):

- Protected fields, i.e. parts of the screen may be set in a mode where they cannot be changed or erased from the keyboard.

```
    Procedure Analyse;
      {compute risetime, overshoot, and solution time}

  begin {Foreground}
     Regulate;
     Trigosc;
     if newpar then begin Parset; newpar: = false end;
     if logpar then Loguy
     if analysis request then Analyse;
  end  {Foreground}
```

This is largely self-explanatory. The variables newpar and logpar are Boolean variables which are set true by Opcom if new parameters are available or of a log is desired.

The regulator is run, the trig signal to the oscilloscope is set, parameters are changed if newpar is true and the logged data is displayed if logpar is true.

BACKGROUND PROGRAM OPCOM

The operator communication program, Opcom, runs in the background. It is thus always ready to accept the operator's commands. The program consists essentially of an initialization part which gives default values to all the variables, and a repeated case-statement which decodes the function keys and takes the appropriate action. A synopsis of the program is given below.

```
Procedure Opcom;

   Procedure Writeform;
     {Writes the protected areas on the screen. The form itself
      is contained in this procedure}

   Procedure Fillform;
     {Writes the actual parameters in the unprotected fields
      on the screen}

   Procedure Readscreen;
     {Reads the unprotected fields, i.e. the parameters from the
      screen and also performs some tests. All parameters are
      read whether they are changed or not}
```

- Block send, i.e. the terminal reads the unprotected fields off the screen and sends them. The block may be the entire screen or just one line.
- Cursor control, i.e. the computer may but the cursor in any position on the screen by sending a special code containing the desired position.
- Function keys. When one of these is pushed a special character sequence containing the number of the key is sent to the computer. The function keys send even if the the terminal is in block mode, where the keyboard normally works in a local mode.

These features are controlled with special character sequences containing the ASCII control characters ESC (ESCape), STX (Start of TeXt) and ETX (End of TeXt).

There exists no standard software to drive this terminal from a PDP-11, and after considering different alternatives it was decided to write a complete terminal driver.

The receiver part of this driver has two operating modes, "function keys" and "text".

In the function key mode the driver waits for the function key sequence, and decodes its key number. This number may be requested by a calling program. Any sequence which is not exactly a function key sequence is ignored by the driver. This means that if the operator pushes the SEND-key of the terminal the text will be sent by the terminal but it will never reach the program.

Text mode is entered with a special procedure call and the driver then gives the SEND command to the terminal. The receiver accepts the text coming in, and when the final ETX character is received the calling program is notified that text is available, and the driver reenters function key mode.

A set of procedures is available to control the terminal from a Pascal program. These procedures include facilities to return a function key number, to position the cursor, to clear the screen, to put the driver in text mode etc.

DATA STRUCTURES AND SHARED VARIABLES

The global data in the MISO program is declared as follows:

```
Const nxmax=5; {max number of regulator state variables}
      nymax=4; {max number of measured variables}
      tmax=20; {length of log}
      dmax=6;  {nxmax+1}

Type yvector=array [0..nymax] of real;
     xvector=array [1..nxmax] of real;
     matrix=array [1..nxmax,0..nymax] of real;
     history=array [0..tmax,1..dmax] of real;
     structure=record
                 nx: integer; {number of regulator state variables}
                 ny: integer; {number of measured variables}
                 outch: integer; {output channel}
                 inch: array [0..nymax] of integer {input channels}
               end;
     parameter=record
     {Parameters of realization of regulator on observable canonical form}

                 hilim,lolim: real;
                 a: xvector;
                 b: matrix;
                 d: yvector;
               end;

Var  s1,s2: structure;
     p1,p2: parameter;
     h: history;
     x2: xvector;
     y: yvector;
     u: real;
     period,tlog: integer;
     newpar,logpar: boolean;
```

The operator communication program has the following internal data structures:

```
Type fivechar = array [1..5] of char;
     oppar = record
     {Image of parameters on operator communication screen
      ts            -sampling period
      hlim,lolim    -bounds on regulator output
      rpoly         -coefficients of polynomial R
      tspoly        -coefficients of polynomials T,S1,S2,S3 and S4}
               ts: real;
               hilim,lolim:real;
               rpoly:xvector;
               tspoly: array [0..nymax,0..nxmax] of real;
             end;

Var  name: array [0..nymax] of fivechar;
     rname: fivechar;
     op1,op2:oppar;
     ops1,ops2:structure;
     i,j: integer;
     errmess,newstruc: boolean;
```

In both cases the variables containing structure and parameters (the variables of type structure, parameter, and oppar) have been doubled. This has been done so that the parameters may be double-buffered. In the cases of Opcom the structure and the parameters are read off the screan into ops1 and op1. If all these numbers are within their legal ranges they are transferred to ops2 op2. They are then written on the screen. The old values are retained if an error occurs. When ops2 and op2 have been updated, the following statement sequence is performed:

```
newpar:=false;
S1:=ops2
P1:=op2
newpar:=true
```

The boolean variable newpar serves as a flag to notify the foreground that new parameters are available. S1 and P1 are then transferred to s2 and p2 by the foreground. s2 and p2 are the variables actually used by the regulator.

This arrangement solves the problem of shared variables. The foreground has strictly higher priority than the background. This means that the assignment statements s2:=p1, and p2:=p1 in the foreground program cannot be interrupted. The statements will on the other hand not be performed until the background has finished transferring its variables into s1 and p1.

The statement newpar:=false is necessary to protect against a situation where newpar has been set true at an earlier occasion, and the sampling interval is so long that the foreground has not yet transferred the new variables. In this case it would be possible for the foreground to access the variables s1 and p1 at the same time as the background.

STORAGE REQUIREMENT AND MANPOWER

The total storage requirements of the MISO package, including TTY handlers etc is 7370 words. The regulator proper takes only 770 words including the assembler-written routines for A/D and D/A conversion. The Pascal written part of the operator's communication compiles to 2400 words, and the support package for the operator's communication takes 2900 words. The part which may be labeled administration, which is shared between background and foreground is 300 words.

The maximum number of measured variables and the maximum number of states in the regulator are determined by the number of parameters which may be displayed on the screen, with 24 x 80 characters. At present these limits are set to four measurements and five states. They may, however, be changed by just changing one constant declaration in the program. The total data storage requirement of the program is about 600 words, where the log file takes 240 words.

The conclusion which may be drawn from these figures is that in a situation where memory is the limiting factor, it is probably the communication with humans rather than with the process which will suffer.

The total manpower to develop these programs is estimated at two man-months. Some parts of this work was on library routines, which may be utilised in the future.

APPENDIX - PROGRAM LISTINGS

```
Program MISO;

{Multi Input Single Output regulator for laboratory
 experiments in sampled data and stochastic control.

 Authors Karl Johan Astrom, Leif Andersson 1978-10-09}

{GLOBAL DATA DECLARATIONS}

Const nxmax=5; {max number of regulator state variables}
      nymax=4; {max number of measured variables}
      tmax=20; {length of log}
      dmax=6;  {nxmax+1}

Type yvector=array [0..nymax] of real;
     xvector=array [1..nxmax] of real;
     matrix=array [1..nxmax,0..nymax] of real;
     history=array [0..tmax,1..dmax] of real;
     structure=record
                 nx: integer; {number of regulator state variables}
                 ny: integer; {number of measured variables}
                 outch: integer; {output channel}
                 inch: array [0..nymax] of integer {input channels}
               end;
     parameter=record
     {Parameters of realization of regulator on observable canonical form}

                 hilim,lolim: real;
                 a: xvector;
                 b: matrix;
                 d: yvector;
               end;

Var  s1,s2: structure;
     p1,p2: parameter;
     h: history;
     x2: xvector;
     y: yvector;
     u: real;
     period,tlog: integer;
     newpar,logpar: boolean;
{------------------------------------------------------------------------------}

{EXTERNAL PROCEDURE DECLARATIONS}

Procedure Formon; external;
Procedure Formoff; external;
procedure Protect; external;
Procedure Unprotect; external;
Procedure Home; external;
Procedure Cursor(horiz,vert:integer); external;
Procedure Clear; external;
Function Readfunc:integer; external;
Procedure Getscreen; external;
```

5. REFERENCES

[1] Andersson, L (1978): A Process Interface for the LSI-11. Dept of Automatic Control, Lund Institute of Technology, Sweden. To appear.

[3] Beehive B-100 Operator Manual. Beehive International 1977.

[2] Mattsson, S E ~~(1978)~~: A Simple Real-time Scheduler. Dept of Automatic Control, Lund Institute of Technology, Sweden, ~~To appear.~~ 1978. Report TFRT-7156

```
Function Screenok:boolean; external;
Function Rform(r:real; size:integer):integer; external;
{-------------------------------------------------------------------------------}


Procedure Opcom;       (1978-10-09)

(Operator's communication for the MISO package.
 Subprogram structure:
   Opcom
     Writeform
       Spaces
       Field
     Fillform
       Rwrite
       Delay
     Setstruc
     Readscreen
       Iget
       Rget
     Partrans
     Writelog
       Wlog1
   and all the external procedures listed above.
)
Type fivechar = array [1..5] of char;
     oppar = record
     (Image of parameters on operator communication screen
      ts            -sampling period
      hlim,lolim    -bounds on regulator output
      rpoly         -coefficients of polynomial R
      tspoly        -coefficients of polynomials T,S1,S2,S3 and S4)
                ts: real;
                hilim,lolim:real;
                rpoly:xvector;
                tspoly: array [0..nymax,0..nxmax] of real;
              end;

Var  name: array [0..nymax] of fivechar;
     rname: fivechar;
     op1,op2:oppar;
     ops1,ops2:structure;
     i,j: integer;
     errmess,newstruc: boolean;


Procedure Writeform;
Var i,j,nx,ny: integer;

Procedure Spaces(i:integer);
begin
if i>0 then   write(' ':i)
end;
```

```
Procedure Field(i: integer);
begin
  Unprotect; Spaces(i); Protect
end;

begin {Writeform}
  Formoff; Clear; Protect;
  write('DISCRETE MULTI-INPUT-SINGLE-OUTPUT REGULATOR'); Spaces(36);
  Spaces(80);
  write(' ':5,'R(q) * u := T(q) * yr - S1(q) * y1 - S2(q) * y2 ...');
  Spaces(24); Spaces(80);
  write('STRUCTURE AND CONNECTIONS'); Spaces(55);
  nx:=ops2.nx; ny:=ops2.ny;
  write(' ':5, 'Number of measurements  ny = '); Field(1); Spaces(45);
  write(' ':5, 'Degree of polynomials   nx = '); Field(1); Spaces(45);
  write(' ':5, 'Variable/channel   u/ '); Field(1);
  write('   yr/ '); Field(2);
  for i:=1 to ny do begin
    write('   y',i:1,'/ '); Field(2)
  end; Spaces(43 - 9*ny);
  Spaces(80);
  write('SAMPLING PERIOD  TS = '); Field(6); Spaces(52);
  Spaces(80);
  write('PARAMETERS'); Spaces(70);
  write(' ':5,'lolim = '); Field(6); Spaces(61);
  write(' ':5,'hilim = '); Field(6); Spaces(61);
  write(' ':5,'Regulator transfer function:'); Spaces(47);
  write(' ':5,'Coefficient');
  for j:=0 to nx do write(j:8,'  ');
  Spaces(54 - 10*nx);
  write(' ':5,'polynomial');Spaces(65);
  write(' ':8,'R(q)',' ':10,'1.0 ');
  for j:=1 to nx do begin
    Spaces(4); Field(6)
  end;
  Spaces(54 - 10*nx);
  for i:=0 to ny do begin
    write(' ':5,name[i],chr(108),'(q)    ');
    for j:=0 to nx do begin
      Spaces(4); Field(6)
    end;
    Spaces(54 - 10*nx)
  end;
  Spaces(80*(4-ny));
  Spaces(75);
  Field(4);
  Formon
end {Writeform};

Procedure Fillform;
Var i,j: integer;

Procedure Rwrite(r:real);
begin
```

```
  write(r:6:Rform(r,6))
end {Rwrite};

Procedure Delay;
begin
  write(chr(177B),chr(177B),chr(177B))
end;

begin {Fillform}
  Delay; Home; Delay;
  with ops2,op2 do begin
    write(ny:1,nx:1,outch:1);
    for i:=0 to ny do write(inch[i]:2);
    Rwrite(ts); Rwrite(lolim); Rwrite(hilim);
    Delay;
    for j:=1 to nx do Rwrite(rpoly[j]);
    for i:=0 to ny do
      for j:=0 to nx do Rwrite(tspoly[i,j]);
  end {with ops2,op2 do};
  Delay;
  if period>0 then write('RUN ') else write('STOP')
end;

Procedure Setstruc(ny,nx:integer);
var i,j: integer;
begin
  period:=0; newstruc:=true;
  ops1.nx:=nx; ops1.ny:=ny;
  with op1 do begin
    for j:=1 to nx do rpoly[j]:=0.0;
    for i:=0 to ny do
      for j:=0 to nx do tspoly[i,j]:=0.0
  end {with op1 do}
end;

Procedure Readscreen;
Label 999;
Const lo=-9999.0; hi=9999.0;
Var i,j: integer;

Procedure Iget(var i:integer; imin,imax:integer;txt:fivechar);
var j:integer;
begin
  readln(j);
  if screenok then begin
    if (j>=imin) and (j<=imax) then i:=j
    else begin
      Formoff; Clear;
      write('VALUE OF ',txt,' OUT OF RANGE.  ');
      write(imin:3,'<= ',txt,' <=',imax:3);
      errmess:=true;
      goto 999
    end end
  else goto 999
```

```
end {Iget};

Procedure Rget(var r:real; rmin,rmax:real; txt:fivechar);
var s:real;
begin
  readln(s);
  if Screenok then begin
    if (s)=rmin) and (s(=rmax) then r:=s
    else begin
      Formoff; Clear;
      write('VALUE OF ',txt,' OUT OF RANGE.   ');
      write(rmin:6:Rform(rmin,6), '(= ',txt,' (=',rmax:6:Rform(rmax,6));
      errmess:=true;
      goto 999
    end end
  else goto 999
end {Rget};

begin {Readscreen}
  with ops1,op1 do begin
    Getscreen;
    Iget(i,1,nymax,'  NY '); Iget(j,0,nxmax,'  NX ');
    if (i()ny) or (j()nx) then Setstruc(i,j)
    else begin
      Iget(outch,0,1,'OUTCH');
      for i:=0 to ny do Iget(inch[i],0,20,' INCH');
      Rget(ts,0.02,650.0,'  TS ');
      i:=round(ts*50.0);
      ts:=i/50.0;
      Rget(lolim,lo,hi,'LOLIM');
      Rget(hilim,lo,hi,'HILIM');
      for j:=1 to nx do begin
        rname[5]:=chr(60B+j);
        Rget(rpoly[j],lo,hi,rname)
      end;
      for i:=0 to ny do begin
        for j:=0 to nx do begin
          name[i][5]:=chr(j+60B);
          Rget(tspoly[i,j],lo,hi,name[i])
        end
      end
    end
  end {with ops1,op1 do};
999:
  if errmess or not Screenok then begin
    errmess:=true;
    i:=Readfunc;
    Writeform
  end
end {Readscreeen};

procedure Partrans;
var i,j,nx,ny: integer;
r: real;
```

```
begin
  nx:=ops2.nx; ny:=ops2.ny;
  with p1,op2 do begin
    for j:=1 to nx do a[j]:=rpoly[j];
    r:=tspoly[0,0];
    d[0]:=r;
    for j:=1 to nx do b[j,0]:=tspoly[0,j]-r*a[j];
    for i:=1 to ny do begin
      r:=tspoly[i,0];
      d[i]:=-r;
      for j:=1 to nx do b[j,i]:=-tspoly[i,j]+r*a[j];
    end;
  end {with p1,op2 do};
  p1.lolim:=op2.lolim;
  p1.hilim:=op2.hilim;
  s1:=ops2
end {Partrans};

Procedure Writelog;
var i,n,ny: integer;

Procedure Wlog1(k,l:integer);
var i,j:integer; r:real;
begin
  for i:=k to l do begin
    write(n:5,' ');
    for j:=1 to ny+2 do begin
      r:=h[i,j];
      write(' ':4,r:6:Rform(r,6))
    end;
    writeln;
    n:=n+1
  end
end {Wlog1};

begin {Writelog}
  ny:=s2.ny;
  write(' Time        u          yr' );
  for i:=1 to ny do write('         y',i:1);
  writeln; writeln;
  n:=-tmax;
  Wlog1(tlog+1,tmax);
  Wlog1(0,tlog);
end {Writelog};

begin {OPCOM}
  Setstruc(3,0);
  rname:='    R ';
  name[0]:='    T ';
  for i:=1 to nymax do begin
    name[i]:='   S  ';
    name[i][4]:=chr(i+60B)
  end;
  for i:=0 to nymax do ops1.inch[i]:=i;
```

```
ops1.outch:=0;
with op1 do begin
  ts:=1.0;
  hilim:=1.0;
  lolim:=-1.0;
end {with op1 do};
op2:=op1;
ops2:=ops1;
Partrans;
Clear; writeln('Set switch in block mode and strike a function key' );
i:=Readfunc;
Writeform; Fillform;
repeat
  case Readfunc of
  1: begin
       errmess:=false;
       newstruc:=false;
       Readscreen;
       if not errmess then begin
         ops2:=ops1;
         op2:=op1;
         Partrans;
         newpar:=true;
         if period >0 then period:=round(op2.ts/0.02)
         else begin
           p2:=p1; s2:=s1;
           newpar:=false
         end;
         if newstruc then writeform
       end;
       Fillform;
     end;
  2: Fillform;
  3: begin
       Cursor(23,76);
       if period=0 then begin
         for i:=1 to s2.nx do x2[i]:=0.0;
         period:=round(op2.ts/0.02);
         write('RUN ' ) end
       else begin
         period:=0;
         write('STOP' )
       end
     end;
  4: begin
      repeat
        logpar:=false;
       Formoff; Clear;
       Writelog;
       logpar:=true;
      until Readfunc <> 4;
      Writeform; Fillform;
     end;
  8: exit;
```

```
Program MISO;

{Multi Input Single Output regulator for laboratory
 experiments in sampled data and stochastic control.

 Authors Karl Johan Astrom, Leif Andersson 1978-10-09}

{GLOBAL DATA DECLARATIONS}

Const nxmax=5; {max number of regulator state variables}
      nymax=4; {max number of measured variables}
      tmax=20; {length of log}
      dmax=6;  {nxmax+1}

Type yvector=array [0..nymax] of real;
     xvector=array [1..nxmax] of real;
     matrix=array [1..nxmax,0..nymax] of real;
     history=array [0..tmax,1..dmax] of real;
     structure=record
                 nx: integer; {number of regulator state variables}
                 ny: integer; {number of measured variables}
                 outch: integer; {output channel}
                 inch: array [0..nymax] of integer {input channels}
               end;
     parameter=record
     {Parameters of realization of regulator on observable canonical form}

                 hilim,lolim: real;
                 a: xvector;
                 b: matrix;
                 d: yvector;
               end;

Var  s1,s2: structure;
     p1,p2: parameter;
     h: history;
     x2: xvector;
     y: yvector;
     u: real;
     period,tlog: integer;
     newpar,logpar: boolean;
{------------------------------------------------------------------------------}

{1978-10-09}

{EXTERNAL PROCEDURE DECLARATIONS}

Function Adin(chan: integer):real; external;
Function Rdigin(chan:integer):real; external;
Procedure Daout(chan: integer; value: real); external;
Procedure Schedule(procedure fore; var period:integer);external;
Procedure Opcom; external;
{------------------------------------------------------------------------------}
```

```
{PROCEDURE DECLARATIONS}
{The procedures are used in the following way
     Foreground
       Regulate
         Adin (external)
         Output
         Ushape
         Daout (external)
         Update
       Trigosc
       Parset
         Init
       Loguy
     Opcom (external)    }

procedure Foreground;

procedure Regulate(var x:xvector;var y:yvector; p:parameter;
                   s:structure;var u:real);
{Calculate and limit control signal and update regulator state}

var i,j:integer; r: real;

function Ushape(u,lolim,hilim:real):real;
begin
  ushape:=u;
  if  u < lolim   then ushape:=lolim;
  if  u > hilim   then ushape:=hilim;
end;{Ushape}

procedure Output;
{Compute control signal from measurement and state}
var w1,w:real;i:integer;
begin
  with p,s do
    begin
      w1:=d[0]*y[0];
      for i:=1 to ny do
      w1:=w1+d[i]*y[i];
      if nx>0 then w:=x[1]+w1 else w:=w1;
      u:=ushape(w,lolim,hilim);
      if nx>0 then x[1]:=u-w1;
    end;
end;{Output}

procedure Update;
{update regulator state}
var x1,w:real;i,j:integer;
begin
  x1:=x[1];
  with p,s do
  begin
    for i:=1 to nx do
```

```
    begin
      w:=-a[i]*x1;
      for j:=0 to ny do
      w:=w+b[i,j]*y[j];
      if i<nx then x[i]:=w+x[i+1] else x[i]:=w;
    end;
  end;
end;{Update}

{code Regulate}
begin
  with p,s do
  begin
    for i:=0 to ny do begin
      j:=inch[i];
      if j<=15 then y[i]:=Adin(j)
      else y[i]:=Rdigin(j-16)
    end;
    Output;
    Daout(outch,u);
    if nx>0 then Update;
  end;
end;{Regulate}

procedure Trigosc;
{generate signal to trig oscilloscope}
var r:real;
begin
  if y[0]>0.0 then r:=0.5 else r:=-0.5;
  Daout((s2.outch+1) mod 2,r);
end;{trig}


procedure Parset;
{Change regulator parameters and initialize regulator state}

procedure Init(u:real;y:yvector;p:parameter;s:structure;var x:xvector);
{Initialize regulator state under assumption that y and u have been
 constant}
var w:real;i,j:integer;
begin
  with p,s do
  begin
    w:=u-d[0]*y[0];
    for j:=1 to ny do
    w:=w-d[j]*y[j];
    x[1]:=w;
    for i:=2 to nx do
    begin
      w:=x[i-1];
      for j:=1 to ny do
      begin
        w:=w-b[i-1,j]*y[j];
        x[i]:=w+a[i-1]*x[1];
```

```
    end
  end
 end
end{Init};


begin{Parset}
  s2:=s1;
  p2:=p1;
  Init(u,y,p2,s2,x2);
  newpar:=false;
end;{Parset}

procedure Loguy;
{Store control variable u and measurements y in cyclic file called h[i,j]
 current time is at i=tlog previous sampling at i=tlog-1}
var i:integer;
begin
  tlog:=(tlog+1) mod (tmax+1);
  h[tlog,1]:=u;
  for i:=0 to s2.ny do
  h[tlog,i+2]:=y[i];
end;{Loguy}
{-------------------------------------------------------------------------------}


{CODE FOREGROUND}
begin
  Regulate(x2,y,p2,s2,u);
  Trigosc;
  if newpar then Parset;
  if logpar then Loguy;
end; {Foreground}

{CODE MAIN PROGRAM}
begin
  period:=0;
  tlog:=0;
  logpar:=true;
  Schedule(foreground,period);
  Opcom;
end.{MISO}
```