MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-6378-MAN/012

ROTINAS COMPLEMENTARES DE CONTROLE DA INTERFACE PC-C (PC - SIMULADOR CONTRAVES) DESCRIÇÃO E UTILIZAÇÃO

Wilson Roberto Freire Rosa
Paulo G. Milani

INPE
São José dos Campos
1997

RESUMO

Neste trabalho apresentam-se cinco rotinas complementares de controle do simulador da Contraves. Essas rotinas permitem que se faça o controle de velocidade nos três eixos do simulador além de permitir que se realizem múlltiplas leituras de velocidade e posição dos eixos do simulador por meio de apenas uma chamada de rotina. Apresentam-se programas exemplo de utilização.

SUMÁRIO

LISTA DE FIGURAS

PÁG.

DESCRIÇÃO DAS ROTINAS ESPECIAIS

1. Introdução

Neste relatório são descritas 5 rotinas que foram desenvolvidas para realizar o controle da velocidade e da aceleração dos eixos do simulador da Contraves[1] através da interface PC-C[2]. As rotinas são:

a) RATE_CONTROL
b) PROG_R_RATE
c) PROG_R_POSITION
d) STOP_SIMULADOR
e) PROG_R_POSITION_TIME

2. Descrição das Rotinas

2.1 Rotina RATE CONTROL

2.1.1 Função: controlar a variação de velocidade nos eixos do simulador.

2.1.2 Sintaxe da rotina:

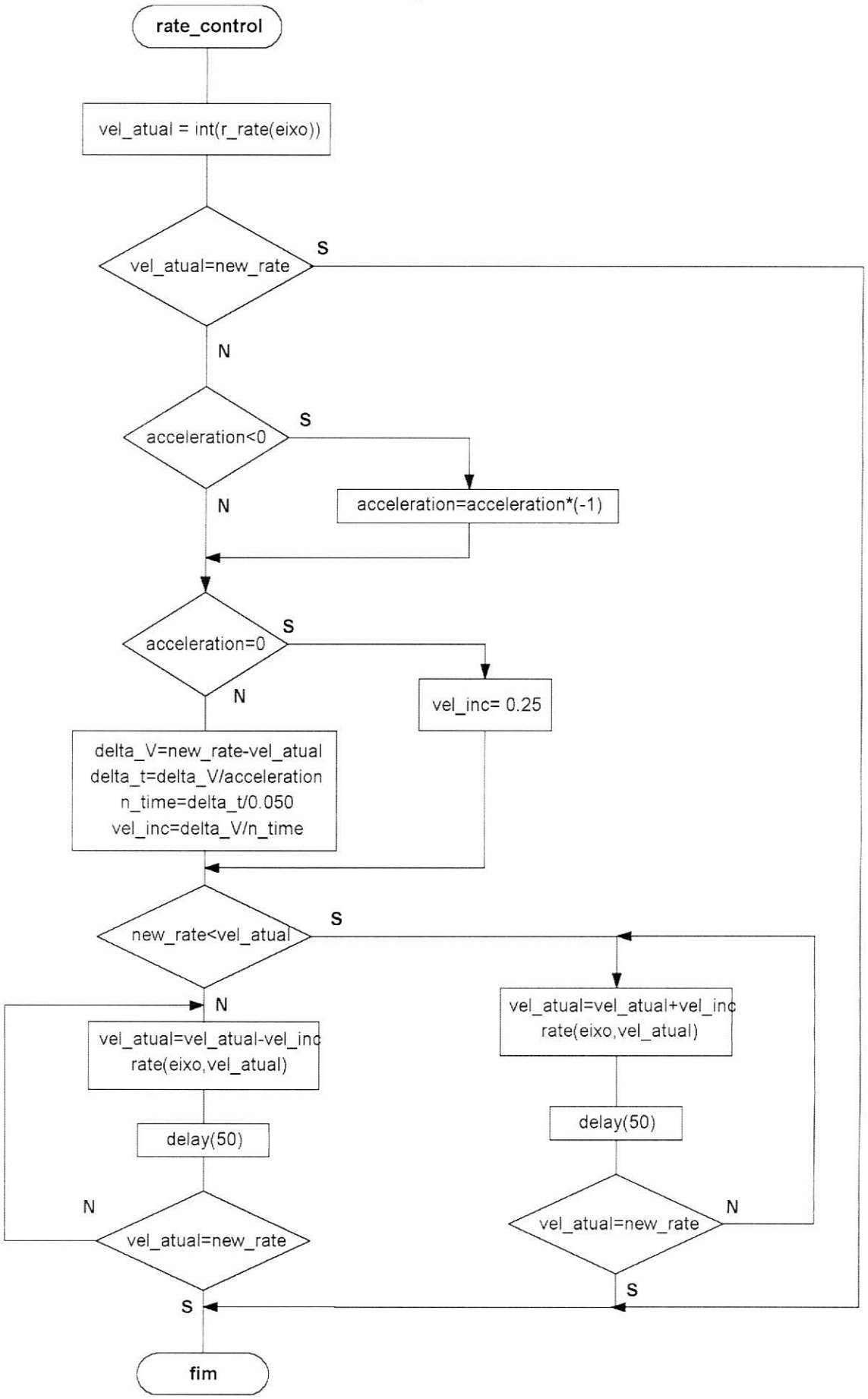
rate_control (int eixo, int new_rate, int acceleration)

onde:
eixo: eixo onde será comandada a velocidade.
new_rate: valor da nova velocidade a ser comandada no eixo selecionado.
acceleration: aceleração que será empregada durante a variação de velocidade.

- Quando o usuário passar o valor **0(ZERO)** para a variável ACCELERATION, a função irá adotar como aceleração default o valor de $5°/s^2$.

Os três parâmetros que são passados para a função são escolhidos pelo usuário.

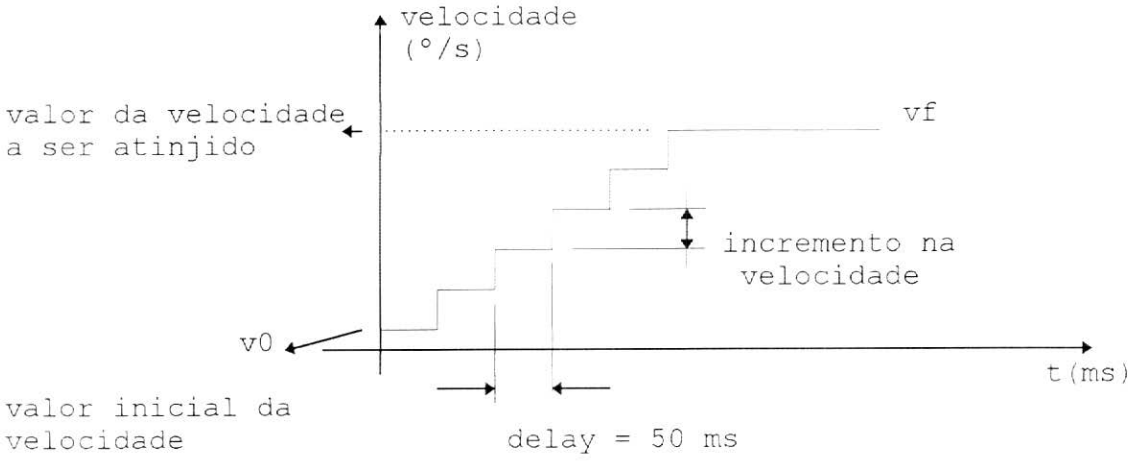
O fluxograma desta função é mostrado na figura 2.1.



2.1 - Fluxograma da rotina RATE_CONTROL

2.1.3 Funcionamento:

Esta rotina tem como objetivo alterar a velocidade angular do eixo selecionado do valor corrente para o valor comandado, de acordo com uma aceleração angular desejada, descrevendo uma curva de aceleração semelhante à que é apresentada na figura 2.2.



2.2 - Curva descrita pela velocidade na função RATE_CONTROL

Por questões de resposta em frequência, escolheu-se um tempo entre incrementos de velocidade fixo, a ser chamado de ***DELAY***, e com valor de 50 ms.

Tendo em vista que o valor do delay é fixo, é necessário que se calcule o valor do incremento a ser utilizado na mudança de velocidade. O valor do incremento irá variar conforme a diferença entre a velocidade final (novo valor a ser comandado) e a velocidade inicial do eixo(velocidade atual em que o eixo se encontra).

Para calcular o valor do incremento deve-se:

1°) Calcular o variação da velocidade do eixo(ΔV).
2°) Calcular o tempo total gasto para se atingir a nova velocidade(Δt).
3°) Calcular o número de degraus (n° de delays=n_time).
4°) Calcular o valor do incremento (vel_inc=Δv/n_time).

No programa a variável ***vel_inc*** terá o valor do incremento a ser utilizado na mudança de velocidade do eixo selecionado.

Dentro desta rotina são utilizadas outras 2 rotinas que são:
r_rate: lê a velocidade do eixo selecionado.
rate : comanda velocidade no eixo selecionado.

Essas rotinas estão descritas no documento ***DESCRIÇÃO DO SOFTWARE DA INTERFACE PC-C)*** [3].

2.1.4 Programa Exemplo

Este programa permite que o usuário altere a velocidade em um dos três eixos do simulador.

Durante a execução do programa principal é solicitado que o usuário entre com o número do eixo, o novo valor da velocidade e o valor da aceleração que será empregada na mudança de velocidade.

```
#include<stdio.h>
#include<conio.h>
#include<path\contrav.cpp>
#include<path\rotextra.cpp>

main()
{ /*início do programa principal*/

 // declaração das variáveis
   int eixo;
   int new_rate;  /* definição da variáveis que serão passadas
                     para a rotina*/
   int acceleration;

   // leitura dos dados que serão passados para a função
   printf("Eixo:");
   scanf("%d",&eixo);/* escolha do eixo*/
   printf("Novo valor da velocidade:");
    scanf("%d",new_rate);/* leitura do valor da nova velocidade
                            a ser comandada no eixo*/
   printf("Valor da Aceleração:");
   scanf("%d",&acceleration);// leitura do valor da aceleração

   // chamada da função
   rate_control(eixo, new_rate, acceleration);
   }
} /* fim do programa principal*/
```

2.2 Rotina Prog r rate

2.2.1 Função: usada para realizar ***n*** leituras de velocidade no eixo selecionado a cada chamada da rotina. O número ***n*** de leituras, selecionado pelo usuário, pode variar de 1 até a tamanho do buffer ***buf_rate***, que é utilizado para armazenar os dados lidos naquela chamada da rotina ***PROG_R_RATE***.

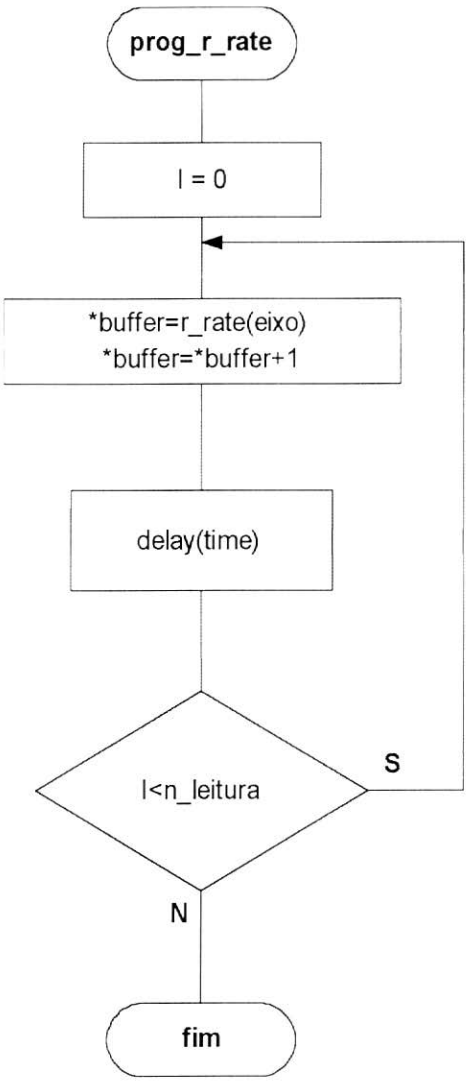
2.2.2 Sintaxe da rotina:

prog_r_rate(int eixo, int t_ms, int n_leitura, float *buffer)

onde:

eixo: eixo onde serão feitas as leituras de velocidade.
t_ms: intervalo de tempo, em milisegundos, entre as leituras, maior que 200 ms.
n_leitura: número de leituras a serem realizadas.
***buffer:** buffer onde serão armazenados os dados lidos.

A figura 2.3 mostra o fluxograma da rotina prog_r_rate



2.3 - Fluxograma da rotina PROG_R_RATE

2.2.3 Funcionamento:

O objetivo desta rotina é realizar ***n*** leituras no eixo selecionado. Cada leitura é feita num intervalo mínimo de 200 ms uma da outra, intervalo esse necessário para que haja a atualização do ***RATE READOUT do MPACS***.

A variável buffer é um vetor de ***n*** posicões que é definido pelo programador no programa principal e passado para a função, Baseportanto seado nesse valor, a rotina incrementa os índices desse vetor para que as leituras não sejam sobrepostas.

2.2.4 Uso da Rotina PROG_R_RATE

Para que se possa chamar a rotina PROG_R_RATE é necessário que no programa principal tenha-se declarado, no mínimo, o tamanho do buffer que irá armazenar os dados(buffer), a variável de leitura do eixo(eixo), a variável de leitura do intervalo de tempo(t_ms) e a variável para armazenar o número de leituras a serem realizadas(n_leitura), como mostra o programa exemplo a seguir.

2.2.5 Programa exemplo

```
/* Este programa realiza 10 leituras de velocidade no eixo
selecionado*/
#include<stdio.h>
#include<conio.h>
#include<contrav.cpp>
#include<rotextra.cpp>

main()
// declaração das variáveis
   float buf_rate[10]; /*  tamanho do buffer que irá armazenar
                       os dados lidos nesta rotina. Este buffer
                       é definido pelo programador*/
  int eixo;
  int t_ms;
  int n_leituras;
   printf("Eixo:");
   scanf("%d",&eixo);/* escolha do eixo*/
   printf("Intervalo de tempo entre as leituras:");
   scanf("%d",&t_ms);/* leitura do intervalo de tempo*/
   printf("Numero de leituras:");
   scanf("%d",&n_leitura); /*
  prog_r_rate(eixo, t_ms, n_leitura,buf_rate);
  }
```

2.3 Rotina Prog r position

2.3.1 Função: realizar ***n*** leituras de posição no eixo selecionado a cada chamada da rotina.

2.3.2 Sintaxe da rotina:

prog_r_position(int eixo, int n_leitura, float *buffer)

onde:

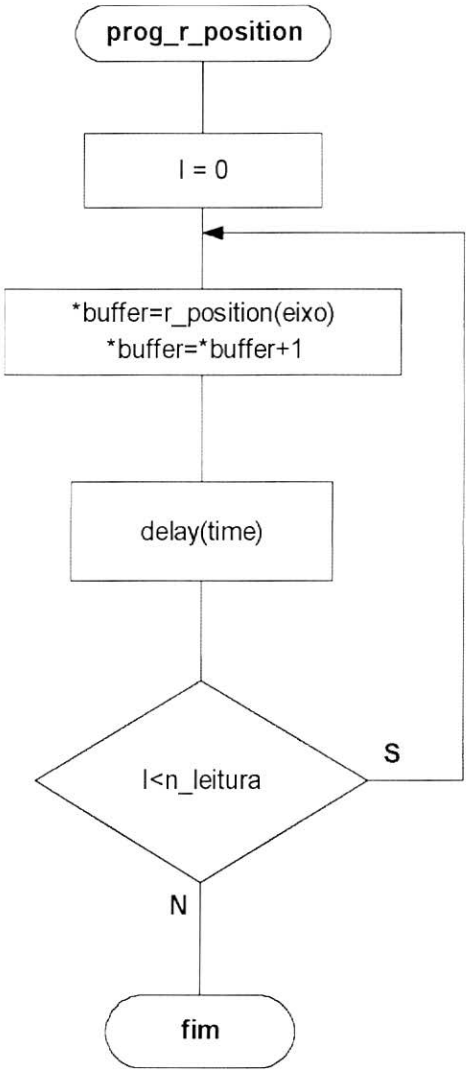
eixo: eixo onde serão feitas as leituras de velocidade.

t_ms: intervalo de tempo, em milisegundos, entre as leituras.

n_leitura: número de leituras a serem realizadas.

***buffer:** buffer onde serão armazenados os dados lidos.

A figura 2.4 mostra o fluxograma da função prog_r_position



2.4 - Fluxograma da rotina PROG_R_POSITION.

2.3.3 Funcionamento:

A lógica utilizada nesta função é a mesma utilizada na função ***prog_r_rate***, lembrando que nesta função são relizadas leituras de posição. De forma contrária ao da leitura de velocidade, nesta rotina não existe tempo mínimo entre 2 leituras de posição consecutivas.

2.3.4 Uso da Rotina PROG_R_POSITION

Nesta rotina também é necessário que se definam as variáveis que serão lidas no programa principal(*ver programa exemplo*) e passadas para a rotina como parâmetro.

2.3.5 Programa exemplo

```
/*Este programa realiza 10 leituras de posição no eixo
selecionado*/
#include<stdio.h>
#include<conio.h>
#include<path\contrav.cpp>
#include<path\rotextra.cpp>

main()
{ /*início do programa principal*/
// declaração das variáveis
   float buf_posit[10]; /*  tamanho do buffer que é definido
                            pelo programador*/
   int eixo;
   int t_ms;
   int n_leituras;

   // leitura dos dados que serão passados para a função
   printf("Eixo:");
   scanf("%d",&eixo);/* escolha do eixo*/
   printf("Intervalo de tempo entre as leituras:");
   scanf("%d",&t_ms);/* leitura do intervalo de tempo*/
   printf("Numero de leituras:");
   scanf("%d",n_leitura);

  // chamada da função
   prog_r_position(eixo, t_ms, n_leitura,buf_posit);
   }
} /* fim do programa principal*/
```

2.4. Rotina STOP SIMULADOR

2.4.1 Função: parar os três eixos do simulador (velocidade = 0) à partir de velocidades quaisquer em seus três eixos.

2.4.2 Sintaxe da rotina:

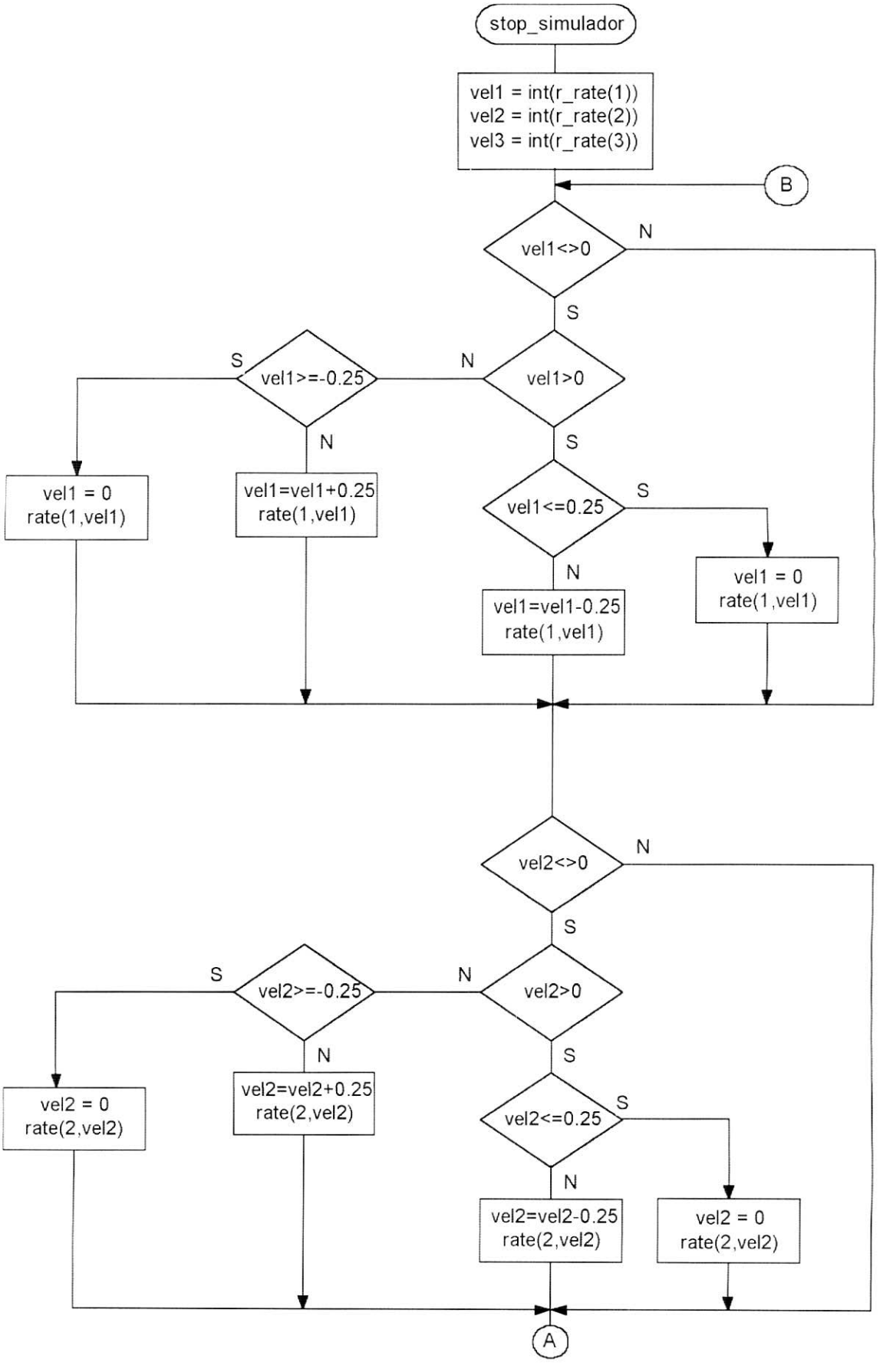
stop_simulador()

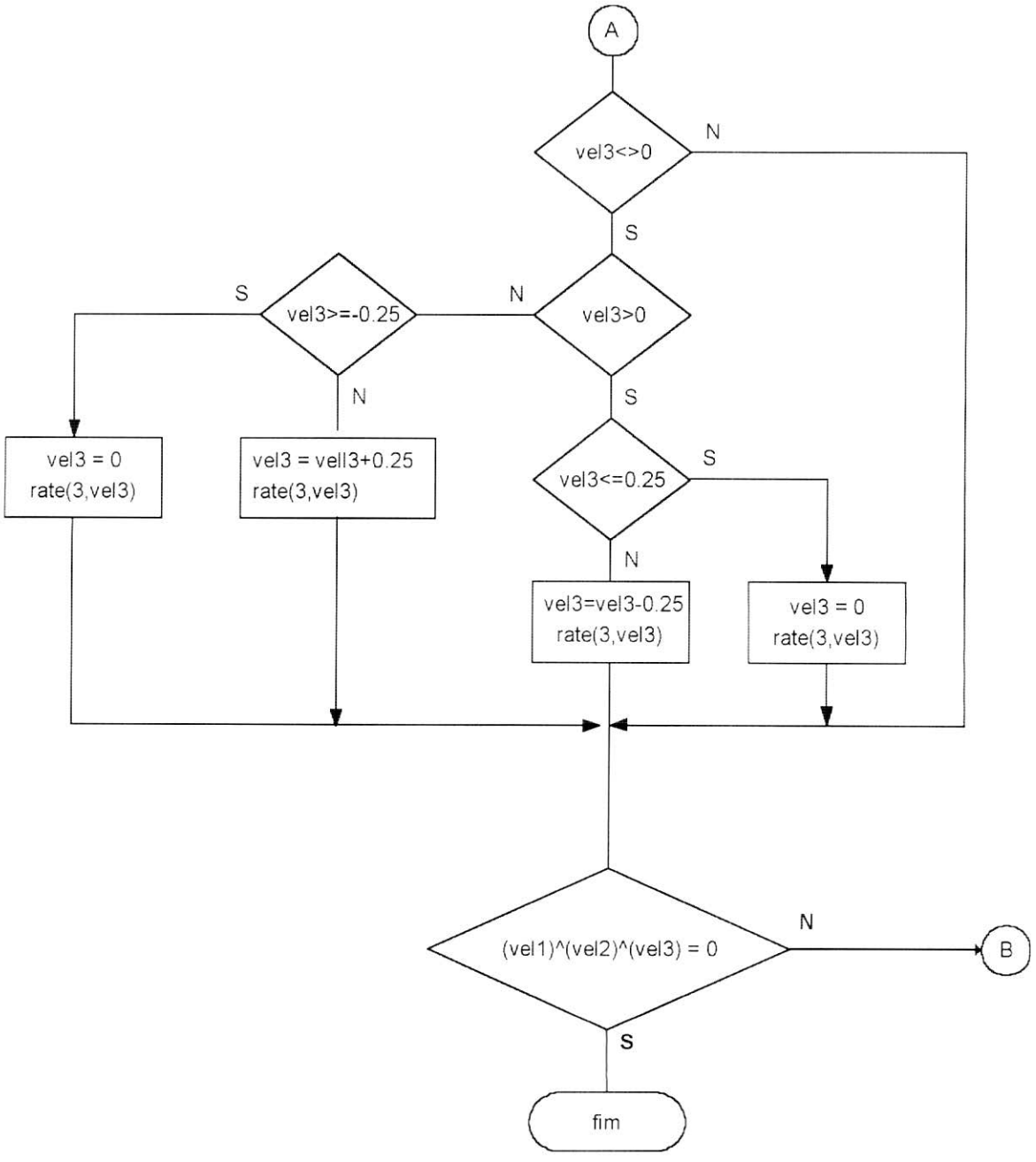
- não há passagem de parâmetros para a função.

2.4.3 Funcionamento:

Esta rotina tem como objetivo parar os três eixos do simulador, ou seja, obter velocidade final igual a zero. Para tanto há a verificação inícial dos valores das velocidades dos três eixos para constatar se os valores são positivos ou negativos. Caso sejam positivos, a velocidade, em módulo, é decrementada a cada 50 milisegundos em 0,25°, fazendo com que a velocidade do eixo diminua 5° por segundo.

A figura 2.5 mostra o fluxograma da função.


stop_simulador
vel1 = int(r_rate(1))
vel2 = int(r_rate(2))
vel3 = int(r_rate(3))
B
vel1<>0
N
S
vel1>0
N
S
vel1>=-0.25
S
N
vel1 = 0
rate(1,vel1)
vel1=vel1+0.25
rate(1,vel1)
vel1<=0.25
S
N
vel1=vel1-0.25
rate(1,vel1)
vel1 = 0
rate(1,vel1)
vel2<>0
N
S
vel2>0
N
S
vel2>=-0.25
S
N
vel2 = 0
rate(2,vel2)
vel2=vel2+0.25
rate(2,vel2)
vel2<=0.25
S
N
vel2=vel2-0.25
rate(2,vel2)
vel2 = 0
rate(2,vel2)
A



2.5 - Fluxograma da rotina STOP_SIMULADOR

2.4.4 Uso da Rotina STOP_SIMULADOR

Para utilizar a rotina ***STOP_SIMULADOR*** basta fazer a chamada da mesma no programa principal.

2.4.5 Programa Exemplo

Este programa para os três eixos do simulador.

```
#include<path\contrav.cpp>
#include<path\rotextra.cpp>
main()
{//início do programa principal
// chamada da função
   stop_simulador();
}// fim do programa principal
```

2.5 ROTINA PROG R POSITION TIME

2.5.1 Função: esta rotina tem a mesma função da rotina ***PROG_R_POSITION*** descrita no ítem 2.3 deste manual. Só que esta rotina além de realizar leituras de posição no eixo selecionado, realiza também a leitura da hora em que a leituras foram realizadas.

2.5.2 Sintaxe da função

prog_r_position(int eixo, int n_leitura, float *buffer, float *buftime)

onde:

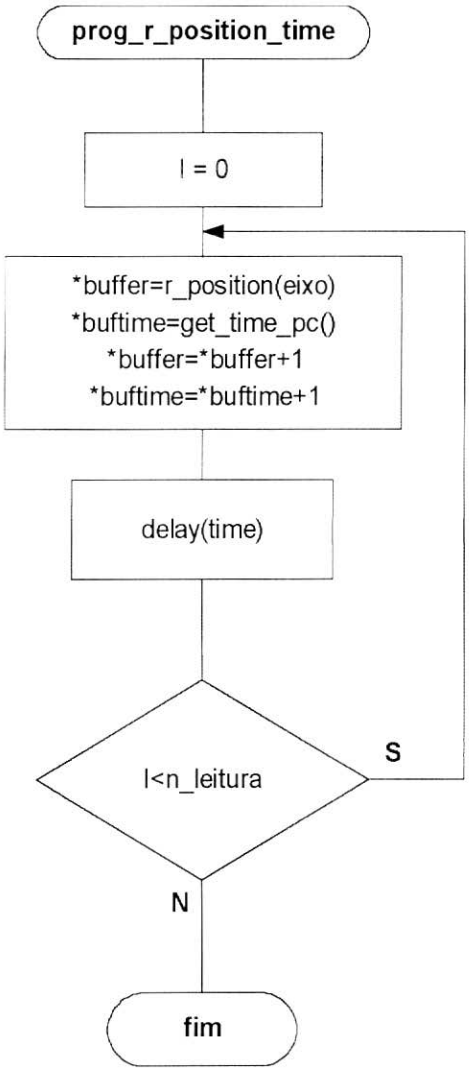
eixo: eixo onde será realizada a leitura de posição.

n_leitura: número de leituras a serem realizadas.

***buffer :** buffer onde serão armazenadas os valores de posição.

***buftime:**buffer onde será armazenada a hora em que cada leitura foi realizada.

O fluxograma da rotina ***PROG_R_POSITION_TIME*** é apresentado na figura 2.6.



2.6 - Fluxograma da rotina ***PROG_R_POSITION_TIME***

2.5.3 Observação: A função ***GET TIME PC*** utilazada nesta rotina, utiliza a rotina ***GETTIME***, que está incluida na biblioteca da linguagem C[4], para ler a hora no PC. A única coisa que a rotina ***GET TIME PC*** faz é pegar a hora lida através da rotina ***GETTIME*** no formato **HH:MM:SS** e convertê-la para segundos.

3. PROGRAMA DE DEMONSTRAÇÃO

3.1 Observação: para que as rotinas descritas anteriormente possam ser executadas é necessário que os arquivos ***CONTRAV.CPP*** e ***ROTEXTRA.CPP*** estejam incluidos no programa que será executado. Para isso basta inserir as sequintes linhas no início do programa.

#include<path\contrav.cpp>

#include<path\rotextra.cpp>

O arquivo ***contrav.cpp*** é necessario porque é nele que estão as rotinas básicas de controle do simulador.

O arquivo ***rotextra.cpp*** é necessário porque é nele que se encontram as 5 rotinas descritas acima.

A descrição das rotinas básicas estão no manual ***DESCRIÇÃO DO SOFTWARE DA INTERFACE PC-C*** [3].

A seguir é apresentado um programa exemplo utlizando as 5 rotinas. Esse programa, o ***EXAMPLE2.CPP***, é o mesmo que foi utilizado para testar as cinco rotinas desenvolvidas.

3.2 Listagem do programa de demonstração

```
#include <d:\tc\bin\contrav.cpp>
#include <d:\tc\bin\rotextra.cpp>

void main()
{
float buf_posit[100];
float buf_rate[100];
float buf_time[100];
char op;
int i,acceleration,n_leituras,t_ms;
float new_rate;
char opcao;
int eixo,modo;
int a,b; //
float c; //auxiliares
float velocidade,posicao;
do
{
clrscr();
printf(" - TESTE DAS ROTINAS DE CONTROLE DO CONTRAVES - \n\n");
```

```
printf("   1 - Leitura do modo\n");
printf("   2 - Leitura de posicao\n");
printf("   3 - Leitura de velocidade\n");
printf("   4 - Escrita do modo\n");
printf("   5 - Escrita de posicao\n");
printf("   6 - Escrita de velocidade\n");

printf("   7 - Teste de modo dos 3 eixos\n");
printf("   8 - Seta inhibt dos 3 eixos\n");
printf("   9 - Reseta inhibit dos 3 eixos\n");
printf("  10 - Mudar taxa de amostragem da velocidade\n");
printf("  11 - Status do modo de operacao do Mpacs \n");
printf("  12 - Status das chaves o encoder\n");
printf("  13 - Status do servo\n");
printf("  14 - Escrever no display do contraves\n");
printf("  15 - Teste da rotina MODE_DISPLAY \n");
printf("  16 - Escrita da velocidade controlada por
tacometro\n");
printf("  17 - Leitura passiva da posicao\n");
printf("  18 - Leitura passiva da velocidade\n");
printf("  19 - Teste das Rotinas Extras\n\n");
printf("");

printf("   0 - Sair\n");
printf("\n Entre com sua escolha:");

scanf("%d",&opcao);
if(opcao==0)
{axis_off();
 return;}
switch(opcao)
     {
case 1:printf("\n\n Eixo:");eixo=getche()-0x30;
            printf("\n Modo=>%d",r_mode(eixo));
            break;
case 2:printf("\n\n Eixo:");eixo=getche()-0x30;
            printf("\n Posicao=>%f",r_position(eixo));
            break;
case 3:printf("\n\n Eixo:");eixo=getche()-0x30;
            printf("\n Velocidade=>%f",r_rate(eixo));
            break;
case 4:printf("\n\n Eixo:");eixo=getche()-0x30;
            printf("\n Modo:");modo=getche()-0x30;
            mode(eixo,modo);
            break;
case 5:printf("\n\n Eixo:");eixo=getche()-0x30;
            printf("\n Posicao:");scanf("%f",&posicao);
            position(eixo,posicao);
            break;
case 6:printf("\n\n Eixo:");eixo=getche()-0x30;
            printf("\n Velocidade:");scanf("%f",&velocidade);
            rate(eixo,velocidade);
            break;
```

```
case 7:printf("\n\nTeste rodando...Tecle algo para terminar");
            do{
             for (a=3;a>=1;a--) for (b=0;b<=7;b++)
                    {mode(a,b);delay(30);}
             for (a=1;a<=3;a++) for (b=7;b>=0;b--)
                    {mode(a,b);delay(30);}
            }while(!kbhit());
            break;
case 8:printf("\n\nInhibit setado");
            pos_encoder_inhibit_set();
            break;
case 9:printf("\n\nInhibit resetado");
            pos_encoder_inhibit_clear();
            break;
case 10:printf("\n\n Eixo:");eixo=getche()-0x30;
            printf("\n\nValores validos:");
            printf("\n    1 ==> 0.1 seg");
            printf("\n    2 ==>   1 seg");
            printf("\n    4 ==>  10 seg");
            printf("\n    8 ==> 100 seg");
            printf("\n\n Entre com a nova taxa:");
            r_rate_range_sel(eixo,getche()-0x30);
            break;
case 11:printf("\n\n Control status=>%X",r_control_status());
            break;
case 12:printf("\n\n Eixo:");eixo=getche()-0x30;
      printf("\n Encoder status=>%X",r_encoder_status(eixo));
            break;
case 13:printf("\n\n Eixo:");eixo=getche()-0x30;
            printf("\n
Servo_status=>%X",r_servo_status(eixo));
            break;
case 14:printf("\n\n Entre com um numero real:");
            scanf("%f",&c);
            keyboard_display(c);
            printf("\n\n Dado gravado ");
            break;

case 15:printf("\n\n Eixo:");eixo=getche()-0x30;
            printf("\n Modo:");modo=getche()-0x30;
            mode_display(eixo,modo);
            break;
case 16:printf("\n\n Eixo:");eixo=getche()-0x30;
      printf("\n\n Entre com a velocidade:");scanf("%f",&c);
            tach_rate(eixo,c);
            break;
case 17:printf("\n\n Eixo:");eixo=getche()-0x30;
       printf("\n posicao=>%f",r_position_passive(eixo));
            break;
case 18:printf("\n\n Eixo:");eixo=getche()-0x30;
            printf("\n velocidade=>%f",r_rate_passive(eixo));
            break;
```

```
case 19:do
      {
       clrscr();
       printf("  ***    TESTE DAS ROTINAS ESPECIAIS  ***");
       printf("\n\n");
       printf("1. Controle de velocidade\n");
       printf("2. Leitura de posicao programada\n");
       printf("3. Leitura de velocidade programada\n");
       printf("4. Parar eixos do SIMULADOR\n");
       printf("5. Controle de posicao do eixo do SIMULADOR\n");
       printf("6. Impressao de dados (POSICAO)\n");
       printf("7. Impressao de dados (VELOCIDADE)\n");
       printf("8. Impressao de dados (CONTROLE DE POSICAO)\n");
       printf("Opcao:");
       op=getche();

 switch (op){
 case '1': {
   printf("\n\n Eixo:");eixo=getche()-0x30;
   printf("\n Velocidade:");scanf("%f",&new_rate);
   printf("\n Aceleracao:");scanf("%d",&acceleration);
   rate_control(eixo,new_rate,acceleration); break; }
 case '2': {
   printf("\n\n Eixo:");eixo=getche()-0x30;
   printf("\n Intervalo (ms):");scanf("%d",&t_ms);
   printf("\n N§ de Leituras:");scanf("%d",&n_leituras);
   prog_r_position(eixo,t_ms,n_leituras,buf_posit);
   break;}
 case '3': {
   printf("\n\n Eixo:");eixo=getche()-0x30;
   printf("\n Intervalo (ms):");scanf("%d",&t_ms);
   printf("\n N° de Leituras:");scanf("%d",&n_leituras);
   prog_r_rate(eixo,t_ms,n_leituras,buf_rate); break;
        }
 case '4':{
   do{
        }
   while( (stop_simulador()!=0)&&(!getche()) );
   printf("ROTINA EXECUTADA");
   delay(2000); break;
             }
  case '5':{
    printf("\n\n Eixo:");eixo=getche()-0x30;
    printf("\n Intervalo (ms):");scanf("%d",&t_ms);
    printf("\n N§ de Leituras:");scanf("%d",&n_leituras);

prog_r_position_time(eixo,t_ms,n_leituras,buf_posit,buf_time);
    break; }
  case '6':{
    do{
    imprime_dados1(buf_posit,n_leituras);
    printf("\n tecle algo p/ continuar...");
    }
    while(!getch()); break; }
```

```
  case '7':{
   do{
    imprime_dados1(buf_rate,n_leituras);
    printf("\n tecle algo p/ continuar...");
     }
   while(!getch()); break;
    }

  case '8':{
    do{
    imprime_dados2(buf_posit,buf_time,n_leituras);
    printf("\n tecle algo p/ continuar...");
    }
    while(!getch()); break;
                }
  default: break;
 }
}
while (op!='0');
}//switch
printf("\n\nTecle algo para continuar...");getch();
}while(1);
}
```

4. CONCLUSÕES:

Este conjunto de rotinas complementares permitirá uma utilização mais suave e precisa do Simulador (Contraves) do que as funções já disponíveis em seu sistema de controle, MPACS.

Com estas rotinas fica disponível para o usuário um controle mais eficiente do Simulador do que aquele fornecido pelo próprio fabricante do equipamento.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Contraves. *Instruction Manual for a Standard MPACS*. USA, ContravesGoes Corporation, 1983.P.155-214

[2] Rosa, W.R.F; Sakuragui, R.R.M; Milani, P.G, *Descrição do hardware da Interface PC-C*. INPE, 1997. no prelo.

[3] Sakuragui R.R.M.; Rosa, W.R.F, Milani, P.G; *Descrição do Software da Interface PC-C*. INPE, 1997. no prelo.

[4] Silveira, C.G; Linguagem C - *Guia do Operador*; ed. McGrawHill, 1989.