MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-14637-NTC/330

DESCRIÇÃO DO SOFTWARE DA INTERFACE PC-C PARA LINUX

Suely Silva
Paulo Giácomo Milani

INPE
São José dos Campos
2007

Publicado por: **esta página é responsabilidade do SID**

Instituto Nacional de Pesquisas Espaciais (INPE)
Gabinete do Diretor – (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 – CEP 12.245-970
São José dos Campos – SP – Brasil
Tel.: (012) 3945-6911
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

Solicita-se intercâmbio
We ask for exchange

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-14637-NTC/330

DESCRIÇÃO DO SOFTWARE DA INTERFACE PC-C PARA LINUX

Suely Silva
Paulo Giácomo Milani

INPE
São José dos Campos
2007

RESUMO (ABSTRACT)

Este trabalho apresenta rotinas desenvolvidas para a realização da comunicação com o Simulador Contraves, bem como o controle do mesmo através de um computador padrão IBM-PC executando o sistema Operacional Linux. A comunicação com o Simulador é feita através da placa de interface PC-C desenvolvida no Laboratório da Divisão de Mecânica Espacial e Controle-DMC. Estas rotinas constituem-se numa adaptação das rotinas desenvolvidas anteriormente para comunicação e controle do Simulador em ambiente DOS. A estrutura básica das rotinas foi totalmente preservada, sofrendo apenas alterações para funcionamento no novo ambiente.

SUMÁRIO

LISTA DE FIGURAS III

1. INTRODUÇÃO 1

2. INSTALAÇÃO E CONFIGURAÇÃO DA PLACA PC-C 5

3. ROTINAS BÁSICAS E DE CONTROLE 7

3.1 UTILIZAÇÃO DO SOFTWARE 7

3.2 VARIÁVEIS GLOBAIS 7

3.3 DESCRIÇÃO DAS ROTINAS BÁSICAS 8

3.4 DESCRIÇÃO DAS ROTINAS DE CONTROLE 10

3.5 - DESCRIÇÃO DAS ROTINAS DE CONTROLE COMPLEMENTARES 17

4. PROGRAMA PARA TESTE DAS ROTINAS E UTILIZAÇÃO DO SIMULADOR 20

4.1 PROGRAMA DE TESTE DAS ROTINAS BÁSICAS E DE CONTROLE 20

4.2 UTILIZAÇÃO DO SIMULADOR 25

REFERÊNCIAS BIBLIOGRÁFICAS 28

LISTA DE FIGURAS

Figura 1.1 – Operação de escrita de dados no barramento do Simulador.....2

Figura 1.2 – Operação de leitura de dados do barramento do Simulador.....3

Figura 1.3 – Hirerarquia do Software.....4

Figura 2.1 - Esquema de ligação dos cabos entre o PC e o MPACS.....5

Figura 2.2 - Localização dos conectores e jumpers da placa de interface.....6

Figura 2.3 - Visualização dos jumpers da placa.....6

Figura 2.4 - Tabela de exemplos de configuração de endereços na placa de interface.....6

1. INTRODUÇÃO

O Laboratório de Simulação da Divisão de Mecânica Espacial e Controle-DMC está equipado com um Simulador de Três Eixos, modelo 53M2-30H da Contraves, que pode ser utilizado para a caracterização e testes de sensores de Sistemas de Controle de Atitude e Órbita (SCAO) de satélites, assim como para a simulação em tempo real, com hardware na malha, desses mesmos sistemas.

A comunicação e o controle do Simulador podem ser feitos em modo local ou remoto.

Para comunicação local existe um sistema de controle dedicado (MPACS) com um teclado que possibilita a execução de diversas funções de controle e comunicação. Em modo remoto o Simulador pode ser acessado via Interface GPIB, RS-232-C ou através de Interface CPU.

A comunicação via Interface CPU é especialmente interessante por permitir maior taxa de transmissão de dados que as outras opções. Esta interface possui 3 barramentos: o barramento de endereço, com 12 bits; o barramento de entrada de dados, com 16 bits; e o barramento de saída de dados, também com 16 bits.

Para a realização da comunicação e controle do Simulador através da Interface CPU utiliza-se um computador padrão IBM-PC, por sua vez equipado com placa de interface apropriada. Para este fim foi desenvolvida no Laboratório da DMC uma placa de interface denominada PC-C (Rosa ET ALII, 1996), que deve ser instalada em um slot ISA do PC a ser utilizado para comunicação com o Simulador.

Para fazer o gerenciamento desta interface e permitir a troca de informações entre o PC e o Simulador foi desenvolvido inicialmente um conjunto de rotinas de comunicação e controle escritas em liguagem C para execução em ambiente DOS. Este conjunto de rotinas está muito bem documentado em Sakuragui ET ALII, 1997.

Posteriormente com o objetivo de viabilizar o acesso ao Simulador de programas de aplicação desenvolvidos para o Sistema Operacional Linux as rotinas originais de gerenciamento da placa de interface foram adaptadas a partir do código fonte para execução em ambiente Linux.

O presente trabalho realiza a descrição do conjunto de rotinas adaptadas para o Sistema Operacional Linux com a finalidade de auxiliar o usuário no desenvolvimento de aplicações para controle do simulador neste ambiente.

As rotinas adaptadas, bem como as originais, se dividem em dois conjuntos: um conjunto com rotinas básicas de comunicação e um outro conjunto contendo rotinas de controle e rotinas complementares. As rotinas básicas interagem diretamente com o hardware da interface PC-C permitindo a leitura e escrita de dados no baramento do Simulador e se constituem de quatro rotinas: read_contraves, write_contraves, read_contraves_float e write_contraves_float. As duas primeiras realizam respectivamente a leitura e escrita de palavras de 16 bits no barramento do Simulador. As outras fazem uso das anteriores para o envio e recepção de dados formatados.

Os diagramas mostrados nas figuras 1.1 e 1.2 ilustram respectivamente como são realizadas as operações de escrita e leitura de dados no barramento da Interface CPU do Simulador através da placa PC-C conectada a um computador padrão IBM-PC.

O segundo conjunto de rotinas é formado por rotinas de controle, que implementam as funções existentes no painel do MPACS, e por rotinas complementares que implementam funções de controle não disponíveis no painel do MPACS. A implementação de todas as funções é realizada através da chamada das rotinas básicas.

Através da chamada das rotinas de controle as aplicações do usuário poderão realizar a manipulação das funções do Simulador, mantendo transparente para o usuário todo o processo de acesso ao hardware da interface e do Simulador. A figura 1.3 apresenta, dentro deste contexto, um diagrama que mostra a hierarquia do software de controle do Simulador no ambiente Linux.

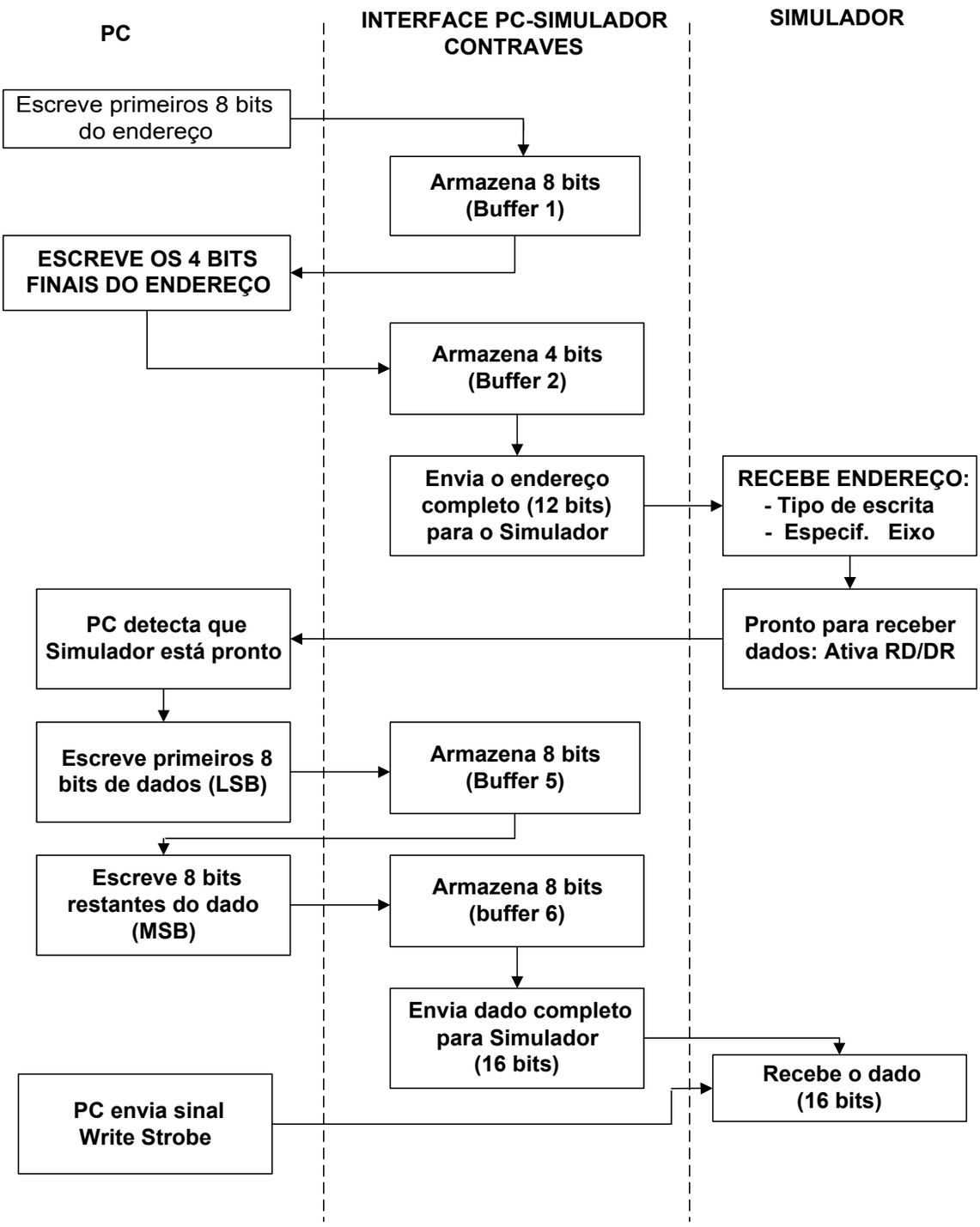

Figura 1.1 – Operação de escrita de dados no barramento do Simulador.

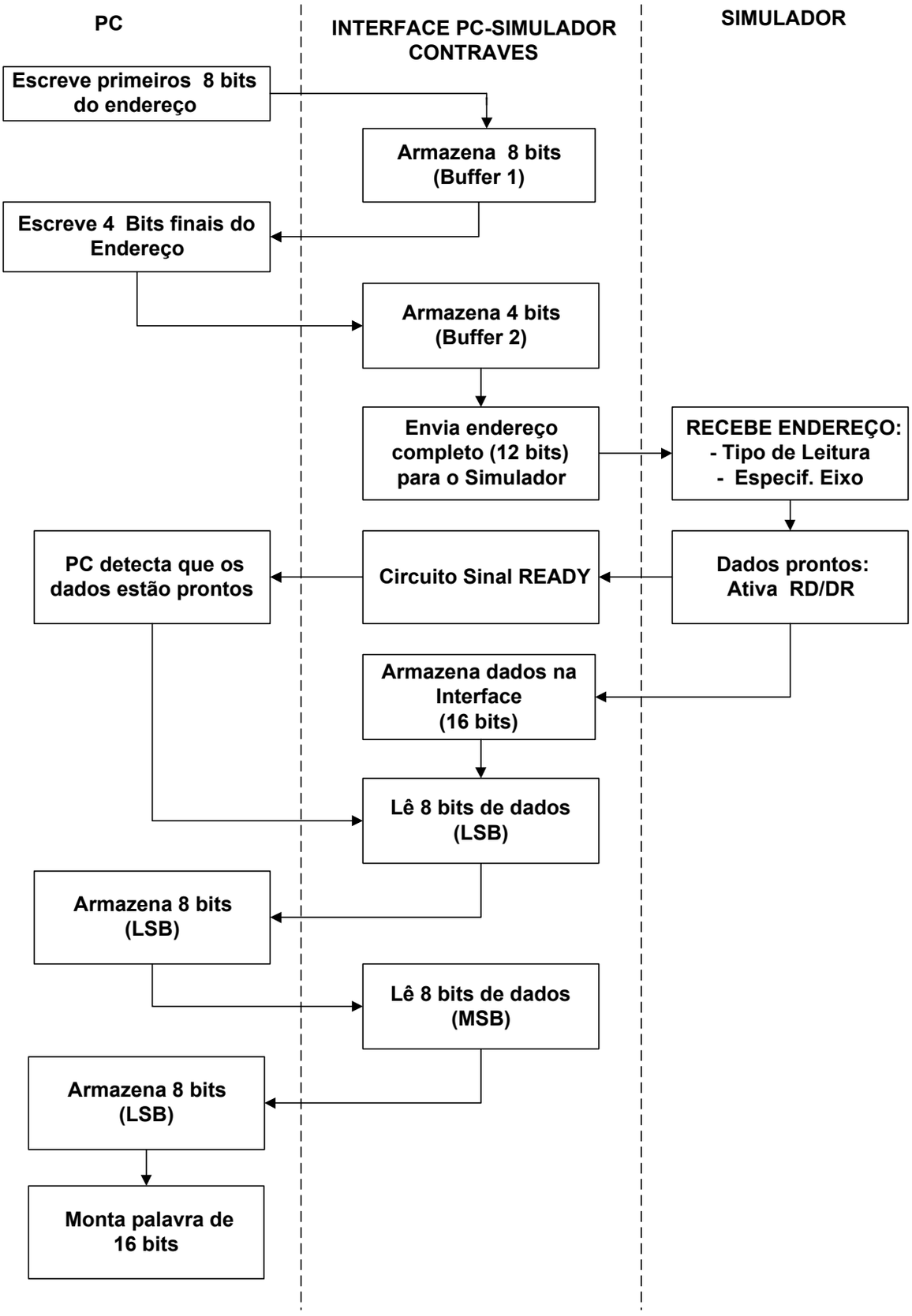

Figura 1.2 – Operação de leitura de dados do barramento do Simulador.

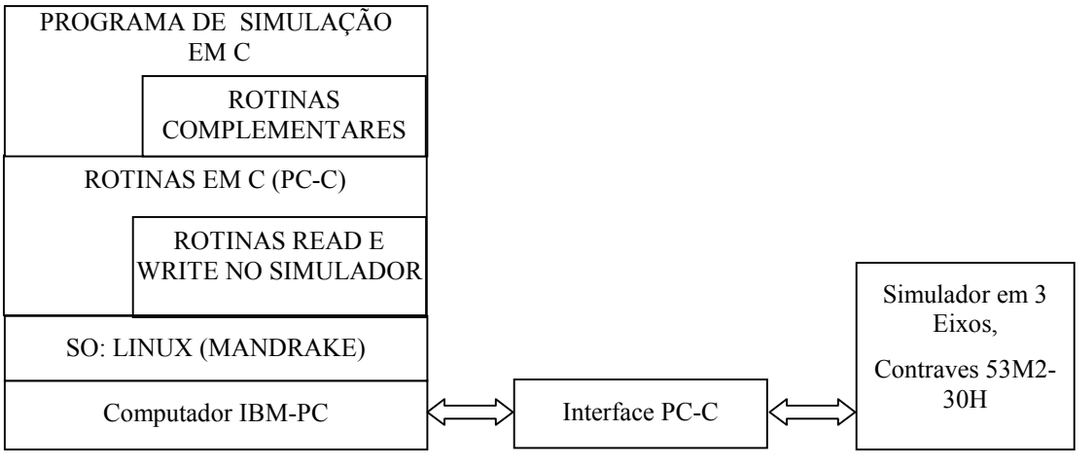

Figura 1.3 – Hirerarquia do Software.

Com a finalidade de propiciar ao leitor maior facilidade de consulta a tópicos relacionados na descrição do software de comunicação e controle, o capítulo 2 deste documento apresenta a transcrição da exposição feita em Sakuragui et alii, 1997, sobre a instalação e configuração da placa de interface PC-C. Detalhes sobre o funcionamento da placa de interface e a descrição do hardware podem ser encontrados em Rosa et alii, 1996.

O capítulo 3 apresenta os procedimentos que o usuário deve realizar para utilizar as rotinas de comunicação e controle em seu programa de aplicação em ambiente Linux e faz a descrição de todas as rotinas, mostrando as variáveis globais definidas e os argumentos de chamada de cada uma delas.

O capítulo 4 apresenta um programa de teste das rotinas de comunicação e controle, incluindo as rotinas complementares. Ainda neste capítulo são apresentados os modos de operação do Simulador e os passos que o usuário deverá seguir para habilitação do mesmo nos diferentes modos de utilização.

2. INSTALAÇÃO E CONFIGURAÇÃO DA PLACA PC-C

Instalação da placa

A instalação da placa de interface PC-C deve ser feita em um computador padrão tipo IBM-PC, para isso deve-se abrir o gabinete do micro e inserir a placa em qualquer slot ISA livre. Os 3 cabos (flat) devem sair pela parte traseira do computador e ser conectados à placa CPU do Simulador, na parte traseira do MPACS.

A interface possui 3 conectores para a conexão física com o MPACS. O esquema de ligação dos cabos é mostrado na figura 2.1.

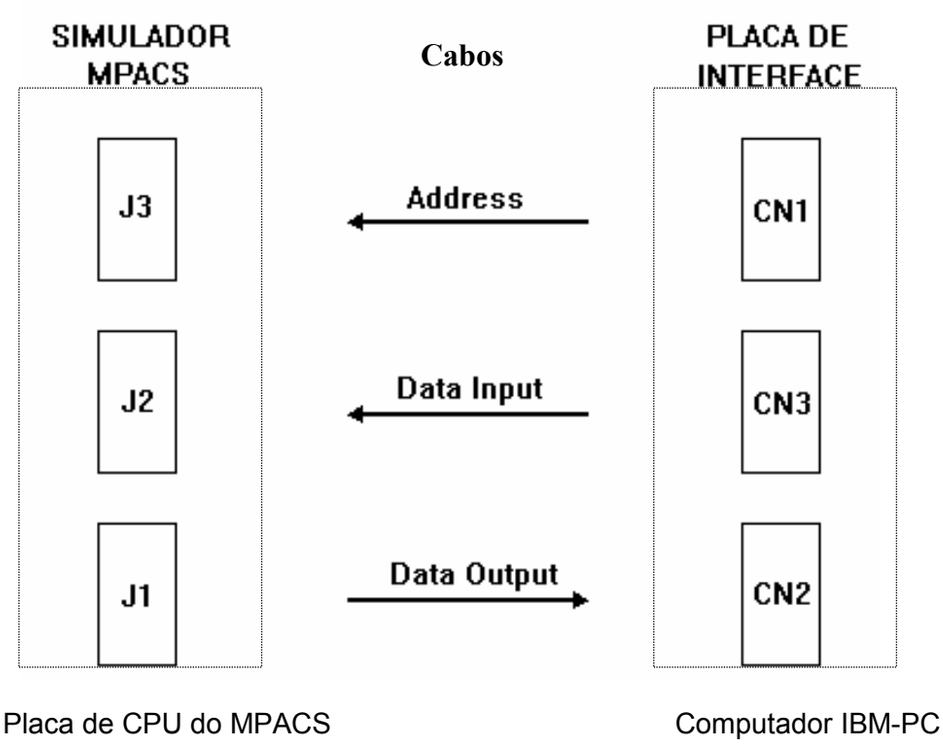

Figura 2.1 - Esquema de ligação dos cabos entre o PC e o MPACS.

O lay-out de placa com a localização dos conectores e dos jumpers de configuração de endereço são mostrados a seguir, na figura 2.2.

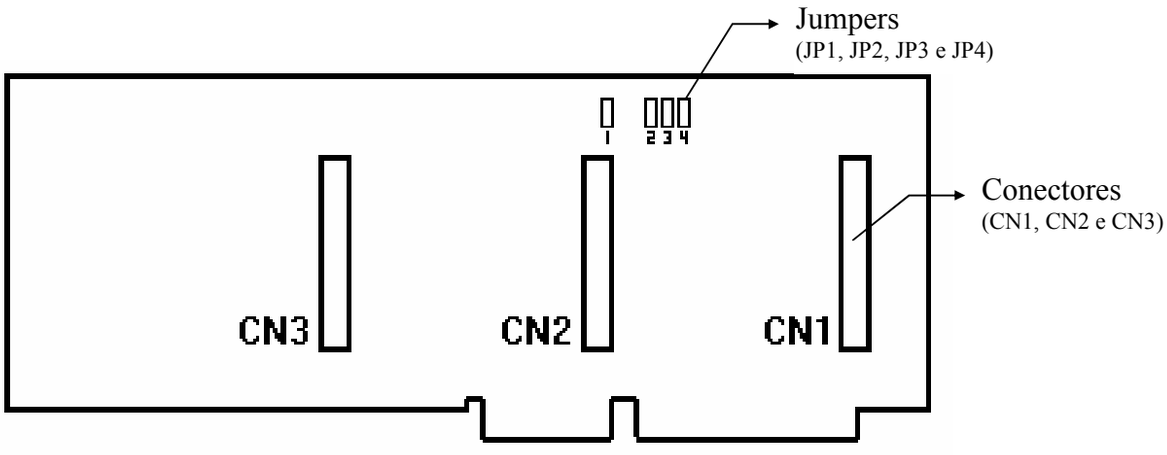

Figura 2.2 - Localização dos conectores e jumpers da placa de interface.

Os jumpers da placa são configurados da seguinte forma (visualizando a placa como na figura 2.3):

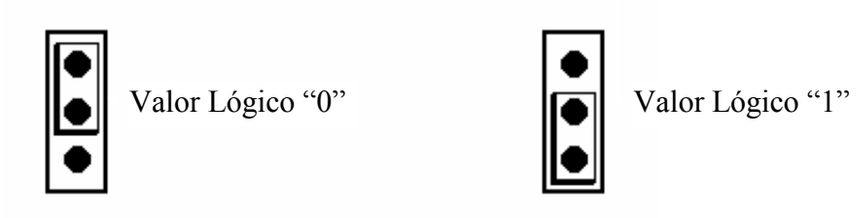

Figura 2.3 - Visualização dos jumpers da placa.

JP1=Define a faixa de endereços, sendo "0" para faixa 200h e "1" para 300h.

JP2, JP3 e JP4 = Variam o endereço dentro da faixa estabelecida pelo JP1. O valor binário colocado nesses jumpers deve ser multiplicado por 10 e somados à faixa de 200h ou 300h.

Exemplos:

Endereço	JP1	JP2	JP3	JP4
200h	“0”	“0”	“0”	“0”
220h	“0”	“0”	“1”	“0”
300h	“1”	“0”	“0”	“0”
310h	“1”	“0”	“0”	“1”
340h	“1”	“1”	“0”	“0”
360h	“1”	“1”	“1”	“0”

Figura 2.4 - Tabela de exemplos de configuração de endereços na placa de interface.

3. ROTINAS BÁSICAS E DE CONTROLE

3.1 UTILIZAÇÃO DO SOFTWARE

Como mencionado no capítulo 1, o software de comunicação e controle do Simulador para Sistema Operacional Linux foi adaptado do software preexistente executável em DOS. A adaptação foi realizada de forma a preservar a estrutura funcional de todas as rotinas. Foram efetuadas alterações no código fonte apenas quando não era possível manter a mesma sintaxe ou mesmas funções originais ou quando o próprio sistema operacional exigia um tratamento diferente na execução de determinadas tarefas. O compilador utilizado para gerar o executável das rotinas em Linux foi o gcc versão 3.4.1, o qual segue o padrão ANSI C.

O código fonte das rotinas básicas de comunicação e das rotinas de controle estão no arquivo rbasica_linux.c e as rotinas complementares de controle no arquivo extra2_linux.c. Para utilizar estas rotinas em um programa de aplicação o usuário deverá linkar as mesmas com o seu programa. O usuário deverá também linkar as bibliotecas especiais ncurses e a biblioteca de rotinas matemáticas como mostra o exemplo abaixo:

```
$gcc -o prog_usu_ex prog_usu.c rbasica_linux.c extra2_linux.c -lncurses -lm
```

sendo, "prog_usu_ex" o nome do executável a ser gerado e prog_usu.c o arquivo contendo o código fonte em C do programa do usuário.

Para executar qualquer programa que utilize direta ou indiretamente as rotinas read_contraves e write_contraves o usuário deverá se logar com prioridade de super usuário pois o Sistema Operacional Linux exige esta prioridade para permitir acesso ao hardware para escrita e leitura de dados.

3.2 VARIÁVEIS GLOBAIS

As variáveis globais das rotinas e suas funções são mostradas a seguir:

- **unsigned int ADDRESS_PC**

  Valor default: 0x200 (200h)

  Função: Indica para as rotinas básicas em qual endereço a placa de interface PC-C se encontra no PC. Esse endereço é configurado na placa por meio de jumpers (straps), como explicado anteriormente, e deve ser atualizado no software. Para isto basta atribuir, em qualquer parte do programa, o novo valor para ADDRESS_PC.

- **unsigned char TIME_OUT_VALUE_MAX**

  Valor default: 3

  Função: Indica para as rotinas básicas o valor máximo de Time-Out para caso de falha na comunicação com o Simulador. Após esse tempo as rotinas terminarão sua execução, enviando o código de erro igual a "1". A rotina também envia uma mensagem ao vídeo indicando ter excedido o tempo de espera. Essa mensagem pode ser habilitada ou desabilitada de acordo com o valor da variável

"WARNING_ON" que será explicada posteriormente. Uma das causas mais comuns do erro de Time-Out é quando ocorre uma tentativa de comunicação com o Simulador, estando este, em modo LOCAL. Diferentemente das rotinas originais a variável TIME_OUT_VALUE_MAX aqui representa o tempo máximo em segundos.

- **unsigned char WARNING_ON**

  Valor default: 1

  Função: Variável que tem a função de habilitar/desabilitar o envio de mensagens de erro (Time-Out) para a tela (do computador onde este software esteja sendo executado), enviadas pelas rotinas básicas de comunicação. Esta variável pode ser mudada de acordo com a aplicação do usuário, sendo:

  0 ⇨ Não exibir mensagem
  1 ⇨ Exibir mensagem

3.3 DESCRIÇÃO DAS ROTINAS BÁSICAS

- int write_contraves(int data, unsigned int address)

  Rotina que escreve uma palavra de 16 bit's no simulador (MPACS)

  Entrada: "data" - Dado a ser escrito no MPACS
  "address" - Endereço onde será escrito o dado

  Saída: Retorna o código de erro:
  0 - sem erro
  1 - time out

- int read_contraves (unsigned int *data, unsigned int address)

  Rotina que lê 16 bit's no simulador (MPACS)

  Entrada: "*data" - Ponteiro do dado a ser lido
  "address" - Endereço de onde será lido o dado no MPACS

  Saída: Retorna o código de erro:
  0 - sem erro
  1 - time out

  Retorna em "*data" o valor lido (16 bit's)

- int write_contraves_float(float rdado,unsigned int address)

Rotina que envia ***rdado***, que é um número float (real), para os endereços ***address*** e ***address*** + 4 do simulador Contraves. Transforma rdado em duas palavras bcd de 16 bits e as envia para o simulador.

Entrada:	"rdado"	- Dado (número real) a ser escrito no MPACS
	"address"	- Endereço onde será escrito o dado no MPACS

Saída: Retorna o código de erro:
0 - sem erro
1 - time out

O valor de rdado (+/- d1 d2 d3 . d4 d5 d6 d7) é transformado em duas palavras bcd da seguinte forma:

palavra 1: d1 d2 d3 d4

palavra 2: st d5 d6 d7

st=0001 para números positivos e st=0000 para números negativos.

Em ***rdado*** são permitidos no máximo quatro dígitos após o ponto decimal (os demais são ignorados)

- int read_contraves_float(float *rdado,int *idado,unsigned int address)

Rotina que lê um número float (real) a partir dos endereços ***address*** e ***address*** + 4 do simulador Contraves. Transforma as duas palavras lidas em um número real e devolve em ***rdado***.

Entrada:	"*rdado"	- Ponteiro do dado (número real) a ser lido do MPACS.
	"*idado"	- Ponteiro do dado onde será colocado uma palavra de status que indica a posição do ponto flutuante.
	"address"	- Endereço de onde será lido o dado do MPACS

Saída: Retorna o código de erro:
0 - sem erro
1 - time out

As duas palavras lidas em ***address*** e ***address***+4 contém dígitos bcd como representado abaixo:

palavra 1: d1 d2 d3 d4

palavra 2: st d5 d6 d7

d1 a d7 são os dígitos bcd que vão compor ***rdado***. ***st*** contém informação de status e a posição do ponto flutuante conforme a medida lida.

As palavras 1 e 2 são transformadas em ***rdado*** da seguinte forma:
rdado = d1 d2 d3 . d4 d5 d6 d7

A palavra 2 é devolvida em ***idado*** para análise do status ou da posição do ponto flutuante.

3.4 DESCRIÇÃO DAS ROTINAS DE CONTROLE

- void axis_off()

  Rotina que coloca todos os 3 eixos do simulador em modo "off"

- int mode(int eixo,int nmode)

  Rotina que comanda o modo dos eixos 1 a 3 do Simulador Contraves.

  Entrada: "eixo" - Seleciona qual o eixo a ser comandado
  "nmode" - Seleciona o modo de operação do Simulador conforme a tabela abaixo:

  nmode = 0 : Off
  nmode = 1 : Position
  nmode = 2 : Rate
  nmode = 3 : Acceleration Control
  nmode = 4 : Simulate
  nmode = 5 : Tach Rate
  nmode = 6 : Gyro
  nmode = 7 : Scorsby

  Saída: Retorna o código de erro:
  0 - sem erro
  1 - erro nos valores de entrada

- int position(int eixo,float posit)

  Rotina que coloca o eixo especificado pelo parâmetro ***eixo*** na posição em graus especificada pelo parâmetro ***posit*** para comandar posição. O Simulador deve estar em modo Remote e o servo em modo Position ou em modo Simulate.

  Entrada: "eixo" - Seleciona qual o eixo a ser comandado
  "posit" - Posição, em graus, em que o eixo deve se posicionar.

  Saída: Retorna o código de erro:

0 - sem erro
1 - erro nos valores de entrada

- int rate(int eixo,float rat)

Rotina que comanda a velocidade especificada pelo parâmetro ***rat***, em graus por segundo, no eixo especificado pelo parâmetro ***eixo*.** Para comandar a velocidade o sistema deve estar em modo Remote e o servo deve estar em modo Digital Rate ou em modo Simulate.

Entrada: "eixo" - Seleciona qual o eixo a ser comandado
"rat" - Velocidade, em graus por segundo, em que o eixo deve girar. A velocidade máxima depende do eixo utilizado, conforme a tabela abaixo:

eixo 1: -999.9999 <= rat <= 999.9999
eixo 2: -749.9999 <= rat <= 749.9999
eixo 3: -499.9999 <= rat <= 499.9999

Saída: Retorna o código de erro:
0 - sem erro
1 - erro nos valores de entrada

Para comandar velocidades acima de 200 graus/segundo, o módulo referente do MPACS deve estar no modo "High Speed".

Observação: a mudança do modo "low speed" para o modo "high speed", ou vice-versa, com o simulador em operação pode provocar uma transição brusca no sistema e a desconexão do eixo comandado (pelo circuito de proteção). dessa forma, esse procedimento deve ser evitado.

- int acceleration_control(int eixo,float acc)

Rotina que tem a função de limitar a aceleração do eixo especificado pelo parâmetro ***eixo***, no valor especificado pelo parâmetro ***acc*** (em graus/seg^2) em comandos de velocidade. Para utilização desta subrotina, verificar se o modulo "acceleration control" está instalado.

Entrada: "eixo" - Seleciona qual o eixo a ser comandado
"acc" - Aceleração, em graus/seg^2, em que o eixo deve girar.

Saída: Retorna o código de erro:
0 - sem erro
1 - erro nos valores de entrada

- int w_control_status(int status)

  Rotina que tem a função de escrever a palavra de status do controle do mpacs. os valores de ***status*** e suas respectivas funções estão descritos na tabela a seguir:

  status = 1 : desabilita freeze
  status = 2 : desabilita high speed mode
  status = 3 : desabilita freeze e high speed mode
  status = -32767: habilita freeze
  status = -32766: habilita high speed mode

  Entrada: "status" - Palavra de status conforme explicação acima..

  Saída: Nenhuma.

  Observação: a mudança do modo "low speed" para o modo "High Speed" no módulo referente do MPACS, ou vice-versa, com o simulador em operação pode provocar uma transição brusca no sistema e a desconexão do eixo comandado (pelo circuito de proteção), dessa forma, esse procedimento deve ser evitado.

- void pos_encoder_inhibit_set()

  Rotina que tem a função de setar o "inhibit" dos 3 eixos para leitura de posição. Isso e' feito para que a posição dos 3 eixos não seja atualizada enquanto e' feita a leitura de posição de cada eixo. Durante a leitura o simulador continua se movimentando, apenas o "display" não e' atualizado.

  parâmetros: nenhum

- void pos_encoder_inhibit_clear()

  parâmetros: nenhum

  Rotina que tem a função de limpar o "inhibit" dos 3 eixos, de modo a permitir a atualização de posição pelo MPACS.

- int r_rate_range_sel(int eixo,int sel)

  Rotina que tem a função de selecionar o período de amostragem para leitura de velocidade no eixo especificado pelo parâmetro ***eixo***. A tabela abaixo mostra os períodos de amostragem selecionados conforme o valor do parâmetro ***sel***:

  sel = 1 : 0.1 seg
  sel = 2 : 1 seg
  sel = 4 : 10 seg

sel = 8 : 100 seg

Entrada: "eixo" - Seleciona qual o eixo a ser comandado
"sel" - Período de amostragem conforme tabela acima.

Saída: Retorna o código de erro:
0 - sem erro
1 - erro nos valores de entrada

- int r_mode(int eixo)

Rotina que tem a função de ler o modo de operação corrente dos eixos 1 a 3 do Simulador Contraves.

Entrada: "eixo" - Seleciona qual o eixo a ser lido.

Saída: Retorna o valor do modo de operação do eixo selecionado , conforme tabela abaixo:

mode = 0 : Off
mode = 1 : Position
mode = 2 : Rate
mode = 3 : Acceleration Control
mode = 4 : Simulate
mode = 5 : Tach Rate
mode = 6 : Gyro
mode = 7 : Scorsby

- float r_position(int eixo)

Rotina que tem a função de ler a posição em graus do eixo especificado pelo parâmetro ***eixo***. o parâmetro ***posit*** devolve o valor lido.

Entrada: "eixo" - Seleciona qual o eixo a ser lido.

Saída: Posição do eixo em graus.

- float r_rate(int eixo)

Rotina que tem a função de ler a velocidade em graus por segundo do eixo especificado pelo parâmetro ***eixo***. O parâmetro ***rat*** devolve o valor lido.

Entrada: "eixo" - Seleciona qual o eixo a ser lido.

Saída: Velocidade do eixo selecionado em graus/seg^2

Observação: A rotina permite a leitura de velocidade nos três eixos do simulador, mas para isso o módulo Encoder do eixo 3 no MPACS deve estar com suas chaves em mpx e cpu.

- int r_control_status()

Rotina que tem a função de ler o status do controle do mpacs. os bits da palavra de status são numerados de 1 a 16 da esquerda (mais significativo) para a direita (menos significativo). A tabela abaixo descreve os bits significativos da palavra de status lida.

Entrada: Nenhuma.

Saída: Status, conforme tabela abaixo.

bit 14=1 : modo remote (indica que o simulador esta' sendo controlado por computador remoto, e não pelo teclado do MPACS).
bit 15=1 : modo high speed (indica que a eletrônica esta' em modo alta velocidade).
bit 16=1 : freeze (indica que todos os codificadores estão inibidos, porque a linha de freeze no barramento do MPACS esta' ativada)

- int r_encoder_status(int eixo)

Rotina que tem a função de ler o status do módulo Encoder para cada eixo do Simulador Contraves. Os bits da palavra de status são numerados de 1 a 16 da esquerda (mais significativo) para a direita (menos significativo).

Entrada: "eixo" - Seleciona qual o eixo a ser lido.

Saída: Status, conforme explicação abaixo.

bits 9 a 12 de status: "coarse encoder information"

bits 13 a 16 de status: "fine encoder information"

A tabela abaixo descreve os bits significativos da palavra destatus lida:

bit 8 'cpu-mpx'
=1 as chaves do painel frontal estão setadas para mpx e cpu. isso permite ao computador selecionar o eixo a ser codificado.

bit 9 ou 13 'inhibit'
=1 indica que o codificador de posição esta' inibido

bit 10 ou 14 'phase lock loss'
=1 indica perda de 'lock' no(s) 'phase locked loop(s)"

bits 11 e 12 ou 15 e 16 'axis fbk sel #2 e axis fbk sel #1'
=indicam o eixo selecionado. É possível selecionar qualquer eixo de 1 a 3 para qualquer um dos três módulos Encoder através de chave no painel frontal, mesmo que o Encoder seja dedicado a um eixo específico (1, 2 ou 3).

- int r_servo_status(int eixo)

Rotina que tem função de ler o status do modulo servo para cada eixo do Simulador Contraves. os bits da palavra de status são numerados de 1 a 16 da esquerda (mais significativo) para a direita (menos significativo). a tabela abaixo descreve os bits significativos da palavra de status lida:

bit 11 'acceleration'
=1 indica que o eixo vai operar sob aceleração controlada sempre que estiver no modo Digital Rate.

bit 12 'on rate'
=1 indica que o eixo atingiu a velocidade comandada.

bit 13 'hi res. switch'
=1 indica que o eixo atingiu a posição comandada no modo Position ou a velocidade comandada no modo Digital Rate.

bit 14 'low res. switch'
=1 indica que o eixo atingiu a velocidade comandada no modo Tach ou que nenhum erro significativo de posição ou velocidade existe no modo Position e Digital Rate. Este bit e' ativado (=1) antes do bit 13.

bit 15 'rate trip'
=1 indica que a velocidade do eixo excedeu o limite setado no controle do painel frontal do módulo. Isto faz com que o servo seja desligado e uma interrupção seja ativada.

bit 16 'servo off'
=1 indica que o servo foi desligado.

- void keyboard_display(float rdado)

Escreve no display miniatura o valor ***rdado***.

Entrada: Valor a ser escrito no display

Saída: Nenhuma.

- int mode_display(int eixo,int imode)

  Atualiza o modo de operação no keyboard para cada eixo do simulador segundo a tabela abaixo:

  imode = 0 : Off
  imode = 1 : Position
  imode = 2 : Rate
  imode = 3 : Acceleration Control
  imode = 4 : Simulate
  imode = 5 : Tach Rate
  imode = 6 : Gyro
  imode = 7 : Scorsby

  Entrada: "eixo" - Seleciona o eixo referente.
  "imode" - Seleciona o modo.

  Saída: Retorna o código de erro:
  0 - sem erro
  1 - erro nos valores de entrada

- int tach_rate(int eixo,float rat)

  Comanda a velocidade dos eixos 1 a 3 do Simulador Contraves.
  Velocidade valida: 0.00<=rat<999 em graus por segundo, apenas os três dígitos mais significativos, diferentes de zero, são usados. Por exemplo, um comando de velocidade de 456.78 vai resultar numa velocidade de 456 graus por segundo. O sistema deve estar em modo Remote. o servo deve estar em modo Tach Rate.

  Entrada: "eixo" - Seleciona qual o eixo a ser comandado.
  "rat" - Velocidade em graus por segundo.

  Saída: Retorna o código de erro:
  0 - sem erro
  1 - erro nos valores de entrada

- float r_position_passive(int eixo)

  Lê a posição dos eixos 1 a 3 do Simulador Contraves (leitura passiva.)

  Entrada: "eixo" - Seleciona qual o eixo a ser lido.

  Saída: Retorna a posição do eixo escolhido.

- float r_rate_passive(int eixo)

  Lê a velocidade dos eixos 1 a 3 do Simulador Contraves (leitura passiva)
  Para que se possa ler a velocidade dos três eixos do Simulador, o módulo Encoder do eixo 3 deve estar com suas chaves em mpx e cpu.

  Entrada: "eixo" - Seleciona qual o eixo a ser lido.

  Saída: Retorna a velocidade do eixo escolhido.

- int w_encoder_sel(int eixo,int isel)

  Função: caso os controles do módulo Encoder do terceiro eixo estejam em mpx e cpu, com o módulo Rate Readout instalado nesse eixo, e' possível comandar este módulo para leitura de velocidade (e posição) dos 3 eixos, um de cada vez.

  Entrada: "eixo" - Seleciona um dos 3 eixos.

  Saída: "sel" - Seta valores de acordo com a tabela abaixo:

  isel:

  0 ⇨ programmed axis feedback
  1 ⇨ inner axis feedback
  2 ⇨ middle axis feedback
  3 ⇨ outer axis feedback
  4 ⇨ system test axis
  5 ⇨ system test inner axis
  6 ⇨ system test middle axis
  7 ⇨ system test outer axis
  8 ⇨ reference test

- void Timeout_text()

  Rotina chamada por **write_contraves** e **read_contraves** para apresentar na tela do computador mensagem indicando tempo excedido durante tentativa de comunicação com o Simulador. Como já mencionado esta mensagem pode ser habilitada ou desabilitada de acordo com o valor da variável "WARNING_ON".

  parâmetros: nenhum

3.5 - DESCRIÇÃO DAS ROTINAS DE CONTROLE COMPLEMENTARES

- void rate_control (int eixo, float new_rate, int acceleration)

Rotina que controla a variação de velocidade nos eixos do simulador. - Quando o usuário passar o valor **0(ZERO)** para a variável ACCELERATION, a função irá adotar como aceleração default o valor de $5º/s^2$.

Entrada: "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.

Saída: Nenhuma.

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

Esta rotina realiza ***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 o tamanho do buffer que é utilizado para armazenar os dados lidos naquela chamada da rotina ***PROG_R_RATE***.
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.

Entrada: "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.

Saída: Nenhuma.

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

Esta rotina realiza ***n*** leituras de posição no eixo selecionado a cada chamada da rotina.

Entrada: "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.

Saída: Nenhuma.

- int stop_simulador()

Tem a função de parar os três eixos do simulador (velocidade = 0) a partir de velocidades quaisquer em seus três eixos.

Entrada: Nenhuma.

Saída: Nenhuma.

- int prog_r_position_time (int eixo, int n_leitura, float *buffer, float *buftime)

Esta rotina tem a mesma função da rotina ***PROG_R_POSITION***, no entanto além de realizar leituras de posição no eixo selecionado, realiza também o registro da hora em que a leituras foram realizadas.

Entrada:
- "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.

Saída: Nenhuma.

4. PROGRAMA PARA TESTE DAS ROTINAS E UTILIZAÇÃO DO SIMULADOR

4.1 PROGRAMA DE TESTE DAS ROTINAS BÁSICAS E DE CONTROLE

Esta seção apresenta a listagem do programa de teste para todas as rotinas adaptadas para o ambiente Linux. O programa fonte encontra-se no arquivo pcc_linux.c. O usuário poderá gerar um executável com o mesmo nome (pcc_linux) utilizando o makefile que se encontra no mesmo pacote e também será listado nesta seção. Para executar o pcc_linux basta abrir um shell como super usuário e executar no mesmo diretório onde se encontra o executável do programa a seguinte linha de comando: #./pcc_linux. O programa apresentará ao usuário um menu auto explicativo para escolha da função a ser executada. A seguir são apresentadas as listagens do makefile e do programa fonte pcc_linux.c.

Listagem do makefile

```
# Makefile para compilar o programa de teste das rotinas básicas e de controle do Simulador de Três Eixos da Contraves
#
All::
	gcc -o pcc_linux pcc_linux.c rbasica_linux.c extra2_linux.c -lncurses -lm
```

Listagem do programa de teste

```
// pcc_linux.c
// Programa de teste das rotinas básicas e de controle, incluindos as rotinas
// complementares.
//

int r_mode(int eixo);
float r_position(int eixo);
float r_rate(int eixo);
int mode(int eixo,int nmode);
int position(int eixo,float posit);
int rate(int eixo,float rat);
void pos_encoder_inhibit_set();
void pos_encoder_inhibit_clear();
int r_rate_range_sel(int eixo,int sel);
int r_control_status();
int r_encoder_status(int eixo);
int r_servo_status(int eixo);
void keyboard_display(float rdado);
int mode_display(int eixo,int imode);
int tach_rate(int eixo,float rat);
float  r_position_passive(int eixo);
float r_rate_passive(int eixo);
void rate_control(int eixo, float new_rate, int acceleration);
```

```
int prog_r_position(int eixo,int t_ms,int n_leitura, float *buffer);
int prog_r_rate(int eixo,int t_ms,int n_leitura,float *buffer);
int stop_simulador(void);
int prog_r_position_time(int eixo,int t_ms,int n_leitura, float *buffer,float *buftime);
void imprime_dados1(float *buffer, int n_leitura);
void imprime_dados2(float *buffer, float *buftime, int n_leitura);


int main()
{
float buf_posit[100];
float buf_rate[100];
float buf_time[100];
int i,acceleration,n_leituras,t_ms;
float new_rate;
int eixo,modo,opcao,op,fcont,sel,tcont;

int a,b; //
float c; //auxiliares

float velocidade,posicao;
do
{
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(0);}

switch(opcao)
        {
        case 1:printf("\n\n Eixo:");scanf("%d",&eixo);
               printf("\n Modo=>%d",r_mode(eixo));
               break;
        case 2:printf("\n\n Eixo:");scanf("%d",&eixo);
               printf("\n Posicao=>%f",r_position(eixo));
               break;
        case 3:printf("\n\n Eixo:");scanf("%d",&eixo);
               printf("\n Velocidade=>%f",r_rate(eixo));
               break;
        case 4:printf("\n\n Eixo:");scanf("%d",&eixo);
               printf("\n Modo:");scanf("%d",&modo);
               mode(eixo,modo);
               break;
        case 5:printf("\n\n Eixo:");scanf("%d",&eixo);
               printf("\n Posicao:");scanf("%f",&posicao);
               position(eixo,posicao);
               break;
        case 6:printf("\n\n Eixo:");scanf("%d",&eixo);
               printf("\n Velocidade:");scanf("%f",&velocidade);
               rate(eixo,velocidade);
               break;
        case 7:printf("\n\nTeste rodando...");
                    for (a=3;a>=1;a--) for (b=0;b<=7;b++)
                                    {mode(a,b);usleep(30000);}
                    for (a=1;a<=3;a++) for (b=7;b>=0;b--)
                                    {mode(a,b);usleep(30000);}
                   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:");scanf("%d",&eixo);
               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:");scanf("%d",&sel);
            r_rate_range_sel(eixo,sel);
            break;
case 11:printf("\n\n Control status=>%X",r_control_status());
            break;
case 12:printf("\n\n Eixo:");scanf("%d",&eixo);
            printf("\n Encoder status=>%X",r_encoder_status(eixo));
            break;
case 13:printf("\n\n Eixo:");scanf("%d",&eixo);
            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:");scanf("%d",&eixo);
            printf("\n Modo:");scanf("%d",&modo);
            mode_display(eixo,modo);
            break;
case 16:printf("\n\n Eixo:");scanf("%d",&eixo);
            printf("\n\n Entre com a velocidade:");scanf("%f",&c);
            tach_rate(eixo,c);
            break;
case 17:printf("\n\n Eixo:");scanf("%d",&eixo);
            printf("\n posicao=>%f",r_position_passive(eixo));
            break;
case 18:printf("\n\n Eixo:");scanf("%d",&eixo);
            printf("\n velocidade=>%f",r_rate_passive(eixo));
            break;

case 19:do
                {
                printf("       ***       TESTE DAS ROTINAS EXTRAS       ***");
                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("");
```

```
            printf("   0 - Sair\n");
            printf("Opcao:");
            scanf("%d",&op);

switch (op){
 case 1: {
             printf("\n\n Eixo:");scanf("%d",&eixo);
             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:");scanf("%d",&eixo);
              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:");scanf("%d",&eixo);
             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 );

                     printf("\nROTINA EXECUTADA\n\n");
                     sleep(2);
                     break;
                    }

case 5:{
                     printf("\n\n Eixo:");scanf("%d",&eixo);
                     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 1 p/ continuar...");
                    }
                    while(!scanf("%s",&tcont));
                    break;
                    }

        case 7:{
                    do{
                    imprime_dados1(buf_rate,n_leituras);
                    printf("\n tecle 1 continuar...");
                }
                while(!scanf("%s",&tcont));
                break;
                }

         case 8:{
                do{
                imprime_dados2(buf_posit,buf_time,n_leituras);
                printf("\n tecle 1 para continuar...");
                }
                while(!scanf("%s",&tcont));
                break;
                }

        default: break;
 }
}
while (op!=0);


}//switch

printf("\n\nTecle 0 para sair ou 1 para continuar...");scanf("%d",&fcont);
}while(fcont!=0);
}
```

4.2 UTILIZAÇÃO DO SIMULADOR

Esta seção apresenta os modos de operação do Simulador e os passos para utilização do mesmo nos dois modos.

Modo Simulado

Neste modo de execução os eixos do Simulador Contraves não se movem, mas o display se apresenta como se eles estivessem se movendo, atualizando o display e

respondendo a comandos de forma similar à execução normal. Neste modo não é necessário ligar a parte de potência do Simulador.

Para utilizar o modo Simulado (SYS) do Simulador Contraves deve-se seguir os seguintes passos:

1. Ligar a chave POWER no AC POWER PANEL;
2. Colocar a chave dos ENCODERS dos eixos 1 a 3 em SYS;
3. Colocar a chave REM-LOC do keyboard em REM.

Para desligar o Simulador após o uso:

1. Desfazer os passos 1,2 e 3 anteriores.

Modo Real

Neste modo de execução , os eixos do Simulador se movem, atendendo aos comandos enviados, e o display é atualizado de acordo com o movimento real dos eixos.

Os passos a seguir para a execução em modo real são:

1. Ligar a chave POWER no AC POWER PANEL;
2. Ligar a chave AC POWER no DC POWER PANEL;
3. Pressionar os botões HOT / NORM dos eixos 1 a 3 no AC POWER PANEL para limpar a condição de MOTOR OVERTEMP;
4. Pressionar AC ON para os eixos 1 a 3 no SWITCHING POWER AMPLIFIER;
5. Pressionar OUTPUT ENABLE para os eixos 1 a 3 no SWITCHING POWER AMPLIFIER;
6. Destravar os eixos do simulador se estiverem travados (no próprio Simulador);
7. Se a chave MOUNT DISABLE estiver ativa, desliga-la (puxar o botão para fora);
8. Resetar os High Resolution Switch nos módulos Servo dos 3 eixos;
9. Pressionar TORQUER ON para os eixos 1 a 3 no AC POWER PANEL;
10. Colocar a chave dos ENCODERS dos eixos 1 a 3 em MPX;
11. Colocar a chave REM-LOC do keyboard em REM.

Para desligar o Simulador após o uso:

1. Pressionar TORQUER OFF para os eixos 1 a 3 no AC POWER PANEL;
2. Desligar OUTPUT ENABLE dos eixos 1 a 3 no SWITCHING POWER SUPPLY;

3. Desligar AC ON dos eixos 1 a 3 no SWITCHING POWER SUPPLY;
4. Desligar a chave AC POWER no DC POWER SUPPLY;
5. Desligar a chave POWER no AC POWER PANEL.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Sakuragui R.R.M.; Rosa, W.R.F, Milani, P.G; *Descrição do Software da Interface PC-C.* INPE, 1997.

[2] Rosa, W.R.F; Sakuragui, R.R.M; Milani, P.G, *Descrição do hardware da Interface PC-C.* INPE, 1997.

[1] Rosa, W.R.F, Milani, P.G; *Rotinas Complementares de Controle da Interface PC-C.* INPE, 1997.

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programa de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. São aceitos tanto programas fonte quanto executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.