Universidade de São Paulo

Escola Politécnica

Departamento de Engenharia Mecânica

Projeto Didático de Simulador de Vôo com base nos Princípios da Plataforma de Stewart

Elaborado por:
Danielle Morita
Fábio Fukunaga
Maurício Dias Menezes Mazza
Rafael Salla Paulilo

Orientador:
Tarcisio Hess Coelho

São Paulo
1999

Universidade de São Paulo
Escola Politécnica
Departamento de Engenharia Mecânica

Projeto Didático de Simulador de Vôo com base nos Princípios da Plataforma de Stewart

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção de título de Graduação em Engenharia.

Orientador:
Tarcisio Hess Coelho

Área de Concentração:
Engenharia Mecânica
Automação e Sistemas

São Paulo
1999

8,5 (oito e meio)

Aos nossos pais que sempre nos guiaram e incentivaram no sentido de nosso aprimoramento espiritual e intelectual.

AGRADECIMENTOS

Aos professores Jun Okamoto, Newton Maruyama e Marcelo Godoy que nos auxiliaram na concepção deste trabalho.

Ao Sr. Dimas da empresa Phoenix Contact que exerceu papel fundamental na doação de equipamentos de hardware para o Departamento de Engenharia Mecânica da Escola Politécnica, de modo a permitir a continuidade do trabalho aqui apresentado.

Ao Sr. Juliano da empresa Phoenix Contact que colaborou diretamente na execução deste trabalho.

E ao professor Tarcisio Coelho, orientador desse trabalho.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

h_a = deslocamento do atuador a

h_b = deslocamento do atuador b

h_c = deslocamento do atuador c

h_d = deslocamento do atuador d

(X_G, Y_G, Z_G) = coordenadas do centro de massa, ponto G

θ_y = posição angular em torno do eixo *y* fixo a base

θ_x = posição angular em torno do eixo *x* fixo a base

t_1 = tempo de translação

t_2 = tempo de rotação em torno do eixo *x*

t_3 = tempo de rotação em torno do eixo *y*

N_{max} = rotação máxima do motor

$V_{máx}$ = velocidade máxima linear do centro de massa da plataforma

$\omega_{x\_máx}$ = velocidade angular máxima em torno do eixo *x*

$\omega_{y\_máx}$ = velocidade angular máxima em torno do eixo *y*

t_{max} = maior valor entre t_1, t_2 e t_3

m = massa transportada pela plataforma

l_s = comprimento do deslocamento da castanha ao longo do fuso

l = passo do fuso

d_r = diâmetro do fuso

J_m = momento de inércia do motor

f = resistência da guia linear

J = momento de inércia da cada um dos atuadores

F = Número de graus de liberdade do mecanismo

λ = Graus de liberdade do espaço no qual o mecanismo opera

l = número de peças

j = número de articulações

f_i = Graus de liberdade da i-ésima articulação

I_d = Graus de liberdade redundante ou passivo

RESUMO

Mecanismos montados com base nos princípios da plataforma de Stewart têm sido importante objeto de estudo de grandes universidades de todo o mundo, dado amplo campo de aplicação que esse modelo proporciona.

O mecanismo abordado neste trabalho consiste de uma plataforma de Stewart simplicada resultante de trabalho de formatura elaborado por grupo de alunos graduados em 1998. Em cima desse modelo simplificado de plataforma de Stewart elaboramos estudos da cinemática inversa e construímos um protótipo didático do mecanismo.

Com o objetivo de se implementar um controle automático dos movimentos do protótipo, realizamos estudo de um sistema eletro-eletrônico correspondente, desde a seleção dos atuadores até a simulação do sistema controlado. Na prática, desenvolveu-se um sistema de aquisição e envio de dados (hardware+software) que simulasse o envio de comandos para os atuadores da plataforma.

Os resultados desse trabalho foram o protótipo mecânico didático supracitado e a implementação do conjunto software e hardware de aquisição, tratamento e envio de dados que simula exatamente o envio de comandos para os atuadores da Plataforma.

1 INTRODUÇÃO

O projeto desenvolvido visa a construção de um protótipo didático de um **simulador de vôo de cadeia cinemática fechada**, análogo aos braços robóticos. A topologia deste mecanismo tem atraído pesquisadores em todo mundo por apresentar vantagens operacionais importantes em relação aos tradicionais de cadeia cinemática aberta.

Algumas destas vantagens são: maior rigidez estrutural; estrutura mais leve e com massa uniformemente distribuída nos atuadores; menor erro de posicionamento e orientação; e maior simplicidade nas equações da cinemática inversa.

O protótipo é um sistema mecânico formado por uma plataforma sustentada por quatro colunas (atuadores) de comprimento variável. A alteração controlada de seus comprimentos proporciona a definição da posição e orientação da plataforma, com 4 (quatro) graus de mobilidade. A teoria a ser utilizada para a concretização desse projeto irá basear-se em resultados de estudos de mecanismos tri-dimensionais de cadeia cinemática fechada, mais conhecidos como ***Plataformas de Stewart.***

2 JUSTIFICATIVA E SÍNTESE BIBLIOGRÁFICA FUNDAMENTAL

2.1 O que é uma Plataforma de Stewart?

A Plataforma de Stewart consiste em um mecanismo de estrutura paralela, ou seja, uma estrutura cujos segmentos formam cadeias cinemáticas fechadas. No caso, este mecanismo é formado por seis colunas de comprimentos variáveis ligadas a uma base estacionária e a uma plataforma móvel.

As colunas podem ser ligadas à plataforma e à base por meio de juntas esféricas (3 graus de liberdade) ou/e por juntas universais (2 graus de liberdade). Em geral, tanto a base quanto a plataforma podem possuir qualquer formato contanto que a estrutura resultante seja estável. A variação do comprimento das colunas torna possível orientar e posicionar a plataforma em relação a base com contabilidade de seis graus de liberdade *(Figura 2.1)*.

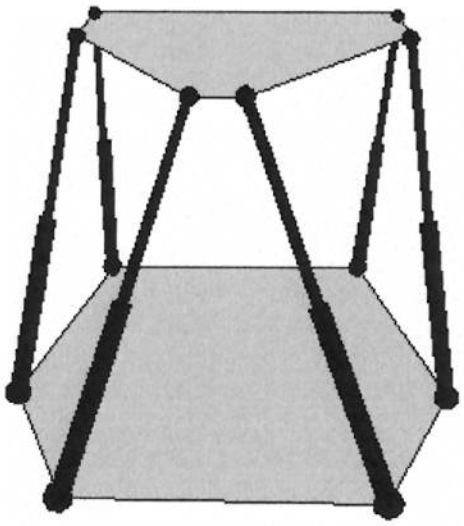


Figura 2.1: Esquema plataforma de Stewart

2.1.1 Vantagens e desvantagens do uso da Plataforma de Stewart

As principais **vantagens** deste mecanismo na aplicação como posicionador e simulador de vôo quando comparado com aquele de cadeia cinemática aberta são:

- **maior rigidez –** esta característica é devida a disposição dos atuadores, fazendo com que as forças aplicadas sobre a plataforma sejam distribuídas entre as seis colunas, tornando as forças necessárias para a movimentação menores uma vez que o carregamento é mais leve.

- **menor erro de posicionamento e orientação –** por ser um mecanismo de vínculos paralelos (cadeia cinemática fechada) sua precisão de posicionamento é melhor que a de mecanismos com vínculos seriais (cadeia cinemática aberta). Isto é melhor ilustrado pelo seguinte exemplo: em braços mecânicos, que são mecanismos de cadeia cinemática aberta, o fato dos atuadores estarem ligados em série torna o sistema pouco preciso pois os erros de posicionamento dos motores vão se acumulando. Isto já não ocorre em cadeia cinemática fechada.

- **maior simplicidade na solução das equações da cinemática inversa –** a cinemática inversa parte da posição da plataforma em relação a base, e a partir desta posição determina os comprimentos dos atuadores.

E as **desvantagens** inerentes ao mecanismo são:

- **maior complexidade na solução das equações de cinemática direta** – a cinemática direta pretende determinar as coordenadas cartesianas das articulações da plataforma por meio dos comprimentos dos atuadores. Usam-se métodos interativos para o cálculo por cinemática direta.

- **presença de posições singulares** – estas são as posições nas quais o mecanismo não pode ser mais controlado. Uma Plataforma de Stewart pode atingir estas posições de diversas formas diferentes, sendo as duas mais significativas: uma quando a plataforma torna-se coplanar a base e outra quando a plataforma gira 90º em torno do eixo perpendicular a ela.

- **menor volume de trabalho**

2.2 Aplicações das Plataformas de Stewart

As aplicações são inúmeras. Dentre elas poderiam ser citadas:

2.2.1 Máquinas Ferramenta

Alguns problemas podem ser associados às máquinas ferramentas convencionais, tais como:

- ✓ graus de mobilidade limitado;
- ✓ massa elevada;
- ✓ reduzida precisão no posicionamento;
- ✓ problemas de vibração;
- ✓ devido à limitação dos graus de liberdade, requer-se ferramentas extras para trabalhar (por exemplo, superfícies de difícil acesso) o que provoca aumento do tempo em que a peça ocupa a máquina e, portanto, seu custo.

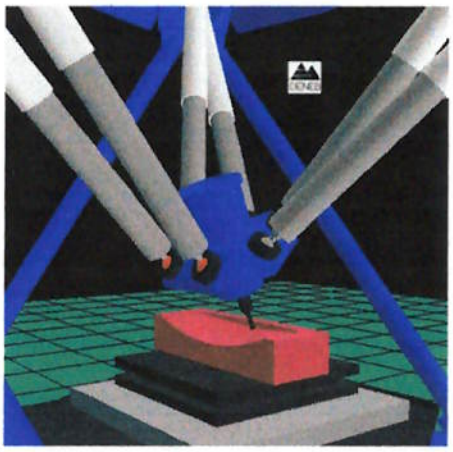


Figura 2.2: Máquina Ferramenta

Tendo em vista as dificuldades acima citadas, identifica-se na plataforma de Stewart uma solução para os problemas presentes nas máquinas ferramentas da atualidade.

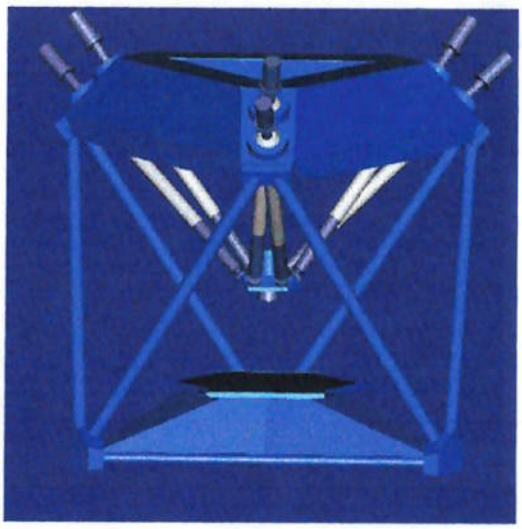


Figura 2.3: Máquina Ferramenta

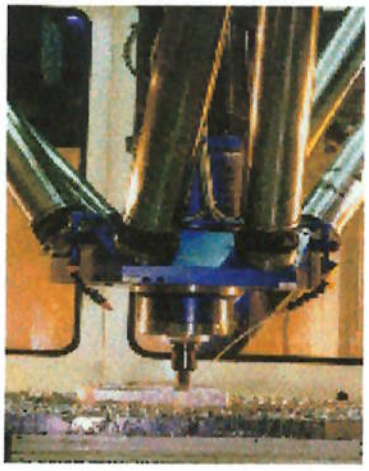


Figura 2.4: Máquina Ferramenta

2.2.2 Simuladores

Simuladores são equipamentos que tentam imprimir sensações ou comportamentos de uma situação real. Estes são utilizados de diversas formas diferentes e para várias aplicações, podendo ser divididos em simuladores de uso profissional e de entretenimento.

Os simuladores de uso profissional são amplamente difundidos no setor aeronáutico, onde o uso de um aparelho real exige grande experiência uma vez que o erro pode ser fatal. Normalmente, este tipo de simulador consiste em uma cabina de aeronave montada sobre uma plataforma que pode mover-se com três até seis graus de liberdade, dependendo do realismo com que se quer representar a situação verdadeira. A cabina é uma reprodução fiel da cabina de um avião existente, possuindo todos os instrumentos, controles, botões e até mesmo rádio.

Figura 2.5: Simulador de vôo

Na área de entretenimento, os simuladores que agregam movimentos vem aumentando a sua participação e se modernizando nos principais parques e centros de lazer. Este tipo de simulador é utilizado não só na simulação de aeronaves, mas também de carros de corrida, "jet sky", etc. No caso dos simuladores de vôo, por não necessitar de muito realismo, bastam três graus de liberdade para simular com razoável fidelidade os movimentos de um avião.

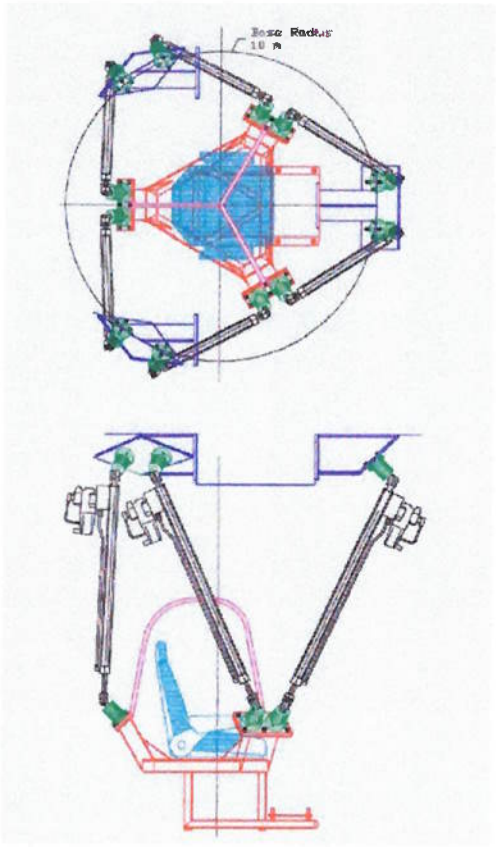


Figura 2.6: Esquema do simulador

Figura 2.7: Protótipo do simulador

2.2.3 Manipulador Nanométrico para Micro-Fabricação

Um manipulador nanométrico deve ser capaz de mover suas "mãos" ao longo de diâmetros atômicos, posicioná-las com precisão de fração-de-diâmetro atômico e então executar movimentos estritamente controlados.

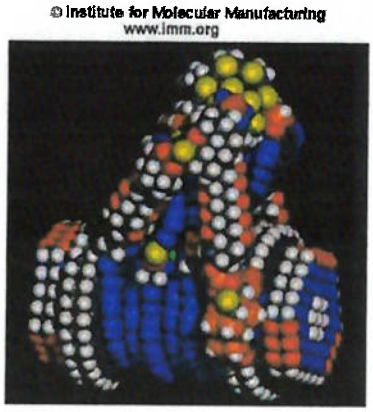


Figura 2.8: Manipulador Nanométrico

A ilustração acima mostra uma estrutura similar a uma plataforma de Stewart, resultante de um processo de modelagem e remodelagem o qual objetivava a obtenção de um controlador de movimentos em escala molecular com uma estrutura de dimensões atômicas.

2.2.4 Absorvedor/isolador de vibração (posicionadores de seis eixos)

Características:

- ✓ Seis graus de liberdade
- ✓ Trabalha em qualquer orientação
- ✓ Capacidade de carga de até 200kg (vertical) e de até 50kg (qualquer orientação)
- ✓ Tamanho extremamente reduzido se comparado aos posicionadores convencionais

- ✓ Controle real de direção
- ✓ Mais rígido do que os sistemas convencionais de multi-eixos : frequência de ressonância igual à 500Hz,
- ✓ carga de 10kg (supressão de vibração)]

Figura 2.9: Absorvedor de vibrações

O modelo M-800 desenvolvido pela Hexapod nada mais é do que um sistema posicionador de multi-eixos. Sua grande vantagem consiste na absorção de vibrações durante os movimentos, graças à sua grande rigidez e à acuracidade de seus componentes (juntas, rolamentos e parafusos). Além disso, o sistema automaticamente compensa os efeitos de movimentos indesejados, através de softwares que "corrigem" estes erros, mandando informações de velocidade e coordenadas para os motores que acionam os eixos.

2.2.5 Sensores de força

A sensação de força tem papel importante no reconhecimento de objetos virtuais. Neste sistema (figura 2.9) o usuário pode sentir a rigidez ou o peso de objetos virtuais

2.2.6 Simulador de movimentos de navios

Este sistema permite ao usuário simular fisicamente o movimento de navios de qualquer tamanho, sendo que é possível se selecionar as condições de navegação : desde mar calmo a tempestades intensas.

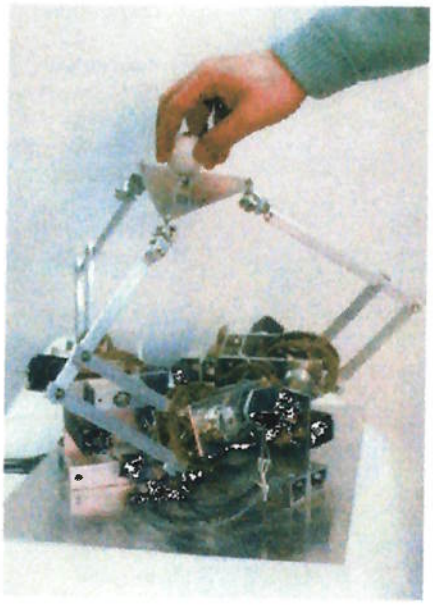


Figura 2.10: Sensor de força

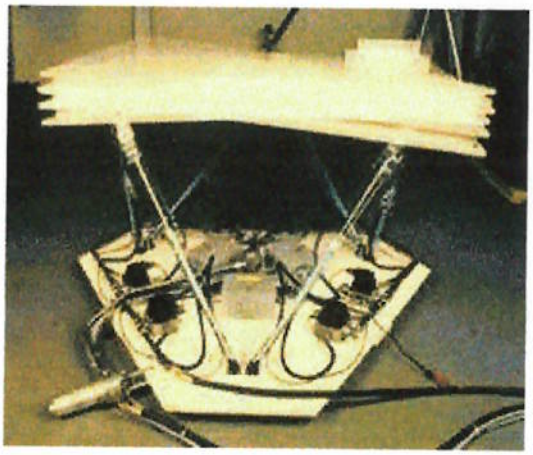


Figura 2.11: Simulador de Movimentos de um Navio

2.2.7 Manipulador Motorizado para Cirurgia Coronária de Ponte de Safena (CABG)

Figura 2.12: Manipulador

O objetivo da Cirurgia Coronária de Ponte de Safena (*CABG-coronary artery bypass graft*) é prover fluxo sanguíneo nas artérias coronárias através da incisão de uma veia de outra parte do corpo na artéria coronária obstruída, evitando a necrose da mesma.

Esse tipo de cirurgia torna-se bastante invasiva para o paciente na medida em que a cavidade peitoral deve ser aberta ("sternotomy") e toda circulação sanguínea deve ser substituída por uma *cardio-vasculary bypass machine*. Esse procedimento traz complicações pós-operatórias tais como destruição de glóbulos vermelhos e plaquetas, hemorragia e coágulos no cérebro.

A solução encontrada para substituir a máquina de by-pass foi a de manter o coração "trabalhando" durante a cirurgia. Essa meta seria significativamente simplificada se o cirurgião tivesse uma vista estável da superfície do coração. É justamente essa a idéia por trás da proposta feita para inovar a cirurgia de CABG: mover uma câmera através das vias coronárias de forma sincronizada com os batimentos cardíacos, proporcionando ao cirurgião uma imagem estacionária do coração. Essa câmera e a instrumentação necessária para a cirurgia estariam montadas sobre um manipulador robótico.}

A tese [13] descreve um manipulador que será utilizado para sustentar e mover tal câmera de vídeo e instrumentos cirúrgicos em elevadas velocidade e precisão de posicionamento suficiente para acompanhar os batimentos cardíacos.

Requisitos para o manipulador:

- ✓ 6 graus de mobilidade
- ✓ volume de trabalho de 50x50x50 mm
- ✓ precisão "submilimétrica"
- ✓ elevada velocidade e aceleração
- ✓ elevada rigidez para eliminar vibrações
- ✓ deve possibilitar acesso do cirurgião ao paciente
- ✓ deve ser facilmente removível.

Três opções foram investigadas: (1)mesa cartesiana, (2)braço robótico e (3)manipulador do tipo Plataforma de Stewart.

Opção	*Vantagens*	*Desvantagens*
Mesa Cartesiana	Conceitualmente mais simples	- Não consegue atuar com acelerações elevadas; - Possui apenas 3 graus de mobilidade.
Braço Robótico	- Possui 6 graus de mobilidade; - Pode posicionar-se sobre a cavidade peitoral remotamente; - mais robusto e estável; - grande volume de trabalho.	- A precisão de posicionamento e velocidade de atuação requeridas são proibitivas; - Um único motor deve sustentar todas as conexões
Plataforma de Stewart	- Possui 6 graus de mobilidade; - Elevada precisão de posicionamento; - Rigidez; - Estrutura mais leve e com peso distribuído sobre os atuadores/motores.	- reduzido espaço de trabalho; - Existência de posições de singularidade (não controláveis).

Tabela 2.1

2.3 Plataforma de Stewart com Atuadores Fixos

A Plataforma de Stewart pode sofrer algumas variâncias. Quando são necessárias altas velocidades e acelerações costuma-se utilizar um mecanismo chamado Plataforma de Stewart com Atuadores Fixos (**PSAF**). Esta trata-se de uma variação da Plataforma de Stewart tradicional, sendo seus atuadores fixados numa base e fazendo com que colunas verticais tenham seus comprimentos variados de modo a movimentar a plataforma.

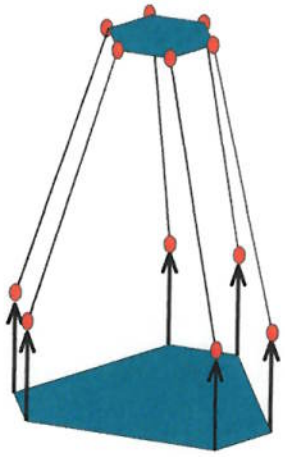


Figura 2.13: Plataforma de Stewart com atuadores fixos

A vantagem deste tipo de disposição é que a força não é usada para mover pesados atuadores mas somente as colunas. Além disso, os atuadores podem ser firmemente fixados de forma que a plataforma desenvolva movimentos em altas acelerações. Por outro lado, esta disposição faz com que para um mesmo deslocamento dos atuadores o volume de trabalho seja menor se comparado com a Plataforma de Stewart padrão. Outra desvantagem é o aparecimento de forças axiais nos atuadores.

3 SUBSISTEMA MECÂNICO

3.1 Mecanismo Proposto

Este trabalho visa o projeto de uma maquete de um posicionador tomando-se como base os princípios da Plataforma de Stewart. Trata-se, pois, de um sistema mecânico controlável, consistindo em uma plataforma sustentada por quatro colunas ligadas a atuadores lineares cujo comprimento pode ser variado (*Figura 2.1*).

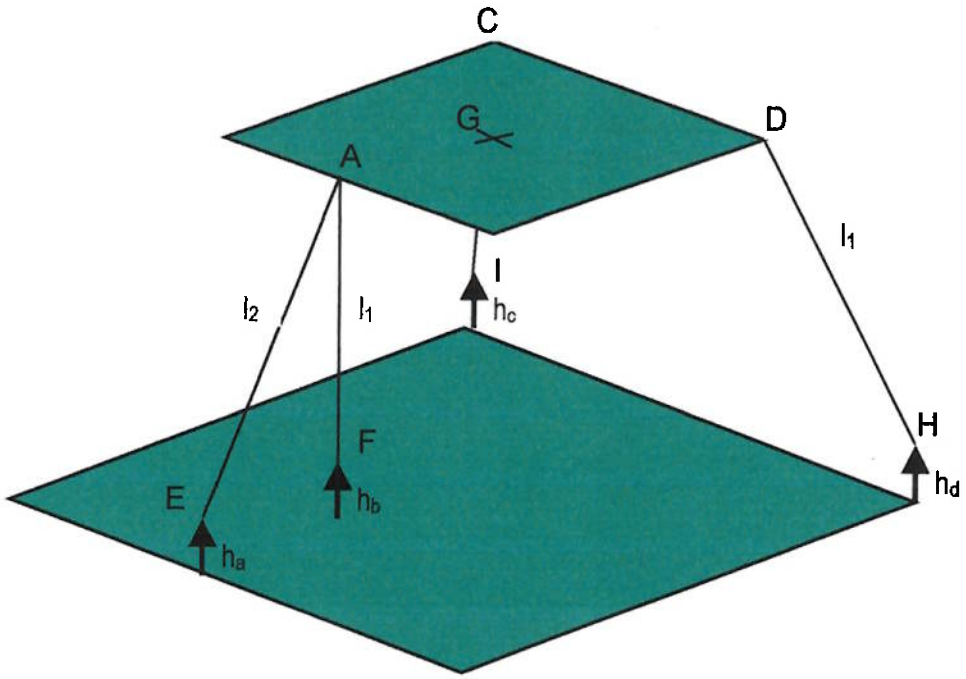


Figura 3.1: Mecanismo proposto

Onde:

- O plano que contém os pontos A, C e D consiste na plataforma móvel;
- h_a, h_b, h_c e h_d são os atuadores lineares e estão fixos a base;
- AF, CI e DH são as colunas de sustentação da plataforma e possuem comprimento igual a l_1;
- AE é uma coluna de sustentação da plataforma de comprimento l_2 maior que l_1;
- G é o centro de massa da plataforma.

A posição desejada da plataforma é obtida variando-se o comprimento dos atuadores, possibilitando movimentos com quatro graus de liberdade, sendo estes duas rotações e duas translações (*Figura 3.2*).

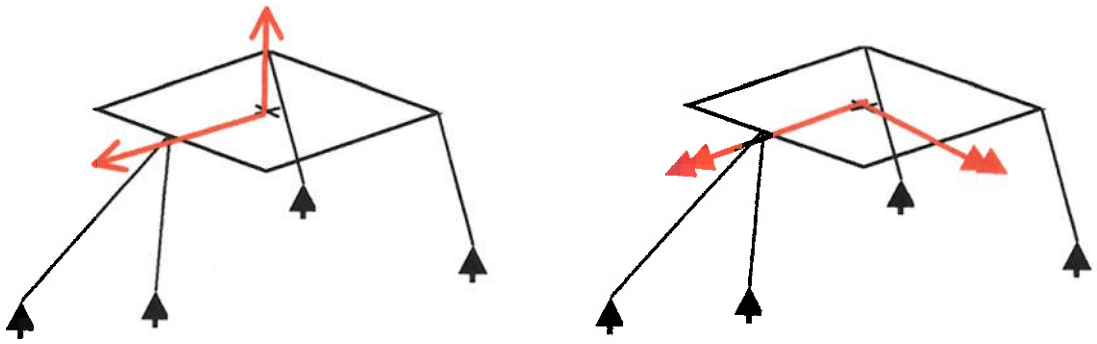


Figura 3.2: 4 graus de liberdade

3.2 Parâmetros de Projeto

Para que o dispositivo planejado cumpra o objetivo proposto são necessárias algumas características:

✓ **Alta rigidez**, uma vez que no caso real um posicicionador deste tipo pode estar submetido a esforços provenientes das forças de corte, quando usado como máquina ferramenta, ou a

esforços oriundos do peso de uma pessoa mais as forças necessárias para imprimir as acelerações no sistema, quando utilizado como simulador de vôo.

Simuladores de vôo convencionais, tal como o desenvolvido pela UBC, devem ser capaz de suportar forças superiores a 8000 N. Entretanto, no projeto proposto, como optou-se por construir apenas uma maquete, serão levados em conta valores bem menores, de forma a tornar o projeto financeiramente viável porém sem perder as características intrínsecas do mecanismo. Portanto, um valor conveniente utilizado para os cálculos é 100 N.

✓ **Mínimo de quatro graus de liberdade** permitindo, no caso do simulador, representar com verossimilhança os movimentos efetuados por uma aeronave; ou, no caso da máquina-ferramenta, oferecer uma vantagem com relação as máquinas convencionais que possuem apenas 3 (três) graus de liberdade (robô cartesiano).

✓ **Altas acelerações e velocidades**. Os dados numéricos utilizados para o teste e aferição do programa foram baseados em um simulador construído pelo departamento de engenharia da University of British Columbia (UBC). Estes valores foram: velocidade de 1 m/s, aceleração de +/- 1g, velocidade angular de +/- 30 graus/s e aceleração angular de +/- 400 graus/s^2. Esses valores serão utilizados no dimensionamento da maquete de forma a que se possa fazer um estudo fiel do comportamento de um simulador de vôo.

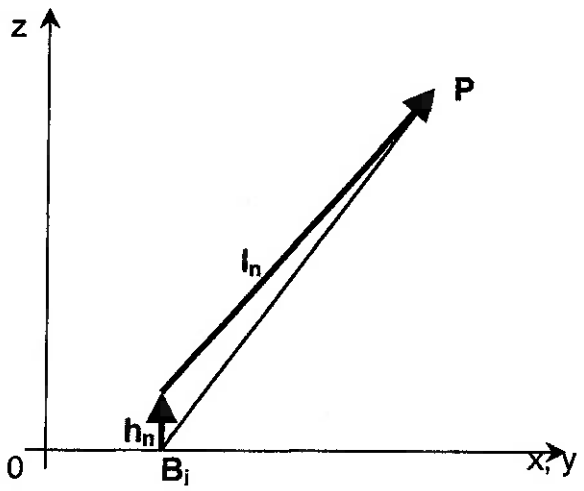
Pretende-se com esse valores analisar a dinâmica dos motores de forma a verificar a eficiência do controle utilizado.

✓ **Precisão de posicionamento.** Para que seja possível tanto usinar uma peça quanto simular com realismo as situações de vôo.

4 MÉTODOS

4.1 Cinemática Inversa

Se entende por Cinemática Inversa para uma Plataforma de Stewart com Atuadores Fixos (**PSAF**), a obtenção dos comprimentos dos atuadores dado a posição da plataforma com relação a base de coordenadas. Na **PSAF**, os atuadores agem linearmente na direção vertical, paralelamente ao eixo **z**.



A esquema acima representa a situação de um atuador onde o ponto **P** é a articulação na plataforma móvel, $\mathbf{B_i}$ é a posição inicial da articulação que está fixa ao atuador, $\mathbf{l_n}$ é o comprimento da coluna *n* e $\mathbf{h_n}$ é o deslocamento do atuador *n*.

Sendo $B_i = (X_{Bi}, Y_{Bi}, Z_{Bi})$ e $P = (X_P, Y_P, Z_P)$, obtemos por meio da equação de Pitágoras a seguinte relação:

$$h_n = Z_P - Z_{B_i} - \sqrt{l_n^2 - (X_P - X_{B_i})^2 + (Y_P - Y_{B_i})^2} \quad (1)$$

Para o mecanismo escolhido para o projeto representado pela figura abaixo:

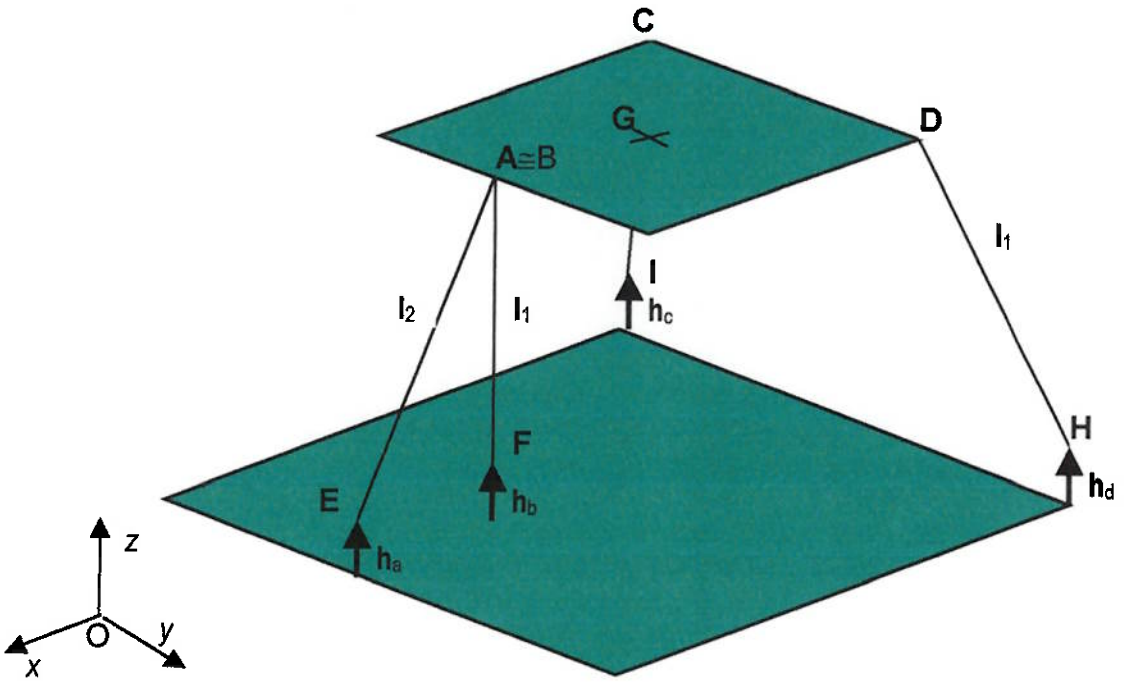


Figura 4.2: Esquema da plataforma utilizada

Temos as seguintes equações:

$$h_a = Z_A - Z_{E_i} - \sqrt{l_2^2 - (X_A - X_{E_i})^2 - (Y_A - Y_{E_i})^2} \quad (2)$$

$$h_b = Z_A - Z_{F_i} - \sqrt{l_1^2 - (X_A - X_{F_i})^2 - (Y_A - Y_{F_i})^2} \quad (3)$$

$$h_c = Z_C - Z_{I_i} - \sqrt{l_1^2 - (X_C - X_{I_i})^2 - (Y_C - Y_{I_i})^2} \quad (4)$$

$$h_d = Z_D - Z_{H_i} - \sqrt{l_1^2 - (X_D - X_{H_i})^2 - (Y_D - Y_{H_i})^2} \quad (5)$$

Onde:

- l_1 é o comprimento das barras menores: AF, CI e DH
- l_2 é o comprimento da barra maior: AE.

- As coordenadas com índice *i* são conhecidas

4.2 Orientação e Posição da Plataforma

A posição e orientação da plataforma será obtida dando-se como entrada as coordenadas do centro de massa, ponto G, e duas posições angulares, em torno dos eixos *y* (θ_y) e *x* (θ_x) fixos a base. De posse desses valores pode-se conseguir as coordenadas dos pontos A, C e D. Para isso, utilizaremos as matrizes de rotação descritas abaixo.

Matriz de rotação em torno do eixo *x*:

$$R_{x,\theta_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\operatorname{sen}\theta_x \\ 0 & \operatorname{sen}\theta_x & \cos\theta_x \end{bmatrix}$$

Matriz de rotação em torno do eixo *y*:

$$R_{y,\theta_y} = \begin{bmatrix} \cos\theta_y & 0 & \operatorname{sen}\theta_y \\ 0 & 1 & 0 \\ -\operatorname{sen}\theta_y & 0 & \cos\theta_y \end{bmatrix}$$

A matriz de rotação representa uma transformação de coordenadas relacionando as coordenadas do ponto G em dois sistemas, um fixo e outro solidário a plataforma, com origens coincidentes em um mesmo ponto. Além disso, a matriz de rotação fornece a orientação do sistema de coordenadas transformado (ou rotacionado) em relação ao sistema fixo.

De posse das matrizes de rotação e das coordenadas no ponto G, pode-se obter as coordenadas das articulações da plataforma por meio das seguintes equações:

$$(A-O)=(G-O)+\begin{bmatrix}\cos\theta_y & 0 & \operatorname{sen}\theta_y\\ 0 & 1 & 0\\ -\operatorname{sen}\theta_y & 0 & \cos\theta_y\end{bmatrix}.\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta_x & -\operatorname{sen}\theta_x\\ 0 & \operatorname{sen}\theta_x & \cos\theta_x\end{bmatrix}.\begin{bmatrix}b\\ 0\\ 0\end{bmatrix}$$

$$(C-O)=(G-O)+\begin{bmatrix}\cos\theta_y & 0 & \operatorname{sen}\theta_y\\ 0 & 1 & 0\\ -\operatorname{sen}\theta_y & 0 & \cos\theta_y\end{bmatrix}.\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta_x & -\operatorname{sen}\theta_x\\ 0 & \operatorname{sen}\theta_x & \cos\theta_x\end{bmatrix}.\begin{bmatrix}-b\\ -a\\ 0\end{bmatrix}$$

$$(D-O)=(G-O)+\begin{bmatrix}\cos\theta_y & 0 & \operatorname{sen}\theta_y\\ 0 & 1 & 0\\ -\operatorname{sen}\theta_y & 0 & \cos\theta_y\end{bmatrix}.\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta_x & -\operatorname{sen}\theta_x\\ 0 & \operatorname{sen}\theta_x & \cos\theta_x\end{bmatrix}.\begin{bmatrix}-b\\ a\\ 0\end{bmatrix}$$

Vale a pena salientar que a ordem das matrizes de rotação influenciam a posição final da plataforma. Por isso, no caso deste mecanismo, é necessário realizar primeiro a rotação em torno do eixo *x* e depois em torno de *y* (tomando-se como referência o sistema de coordenadas da base).

Desenvolvendo as equações acima tem-se:

$$(A-O)=(G-O)+b.\cos\theta_y\vec{i}-b.\operatorname{sen}\theta_y\vec{j} \quad (6)$$

$$(C-O)=(G-O)-(b.\cos\theta_y+a.\operatorname{sen}\theta_x.\operatorname{sen}\theta_y)\vec{i}-a\cos\theta_x\vec{j}+(-a\operatorname{sen}\theta_x.\cos\theta_y+b\operatorname{sen}\theta_y)\vec{k} \quad (7)$$

$$(D-O)=(G-O)+(a.\operatorname{sen}\theta_x.\operatorname{sen}\theta_y-b.\cos\theta_y)\vec{i}+a\cos\theta_x\vec{j}+(a\operatorname{sen}\theta_x\cos\theta_y+b\operatorname{sen}\theta_y)\vec{k} \quad (8)$$

Rescrevendo estas equações obtém-se:

$$(X_A, Y_A, Z_A) = (X_G + b.\cos\theta_y ; Y_G ; Z_G - b\,\mathrm{sen}\,\theta_y) \quad (9)$$

$$(X_C, Y_C, Z_C) = (X_G - b.\cos\theta_y - a.\mathrm{sen}\,\theta_x.\mathrm{sen}\,\theta_y ; Y_G - a.\cos\theta_x ; Z_G - a\,\mathrm{sen}\,\theta_x.\cos\theta_y + b.\mathrm{sen}\,\theta_y) \quad (10)$$

$$(X_D, Y_D, Z_D) = (X_G - b.\cos\theta_y + a.\mathrm{sen}\,\theta_x.\mathrm{sen}\,\theta_y ; Y_G + a.\cos\theta_x ; Z_G + a\,\mathrm{sen}\,\theta_x.\cos\theta_y + b.\mathrm{sen}\,\theta_y) \quad (11)$$

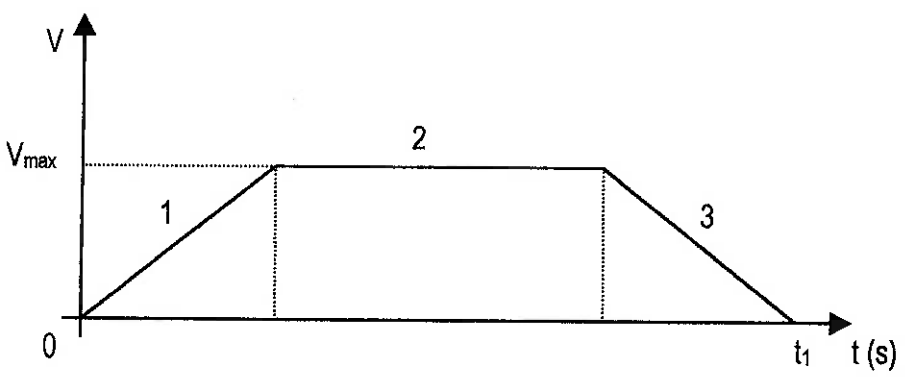
Para encontrar os deslocamentos nos atuadores devemos substituir os valores obtidos nas equações (9), (10) e (11) nas equações (2), (3), (4) e (5).

4.3 Movimento da Plataforma

Para que a maquete cumpra o seu objetivo de representar o mais fidedignamente possível tanto um simulador de vôo quanto uma máquina-ferramenta, algumas medidas devem ser tomadas, a saber:

4.3.1 Curva de velocidade da plataforma

Os movimentos da plataforma, tanto de translação quanto de rotação, devem obedecer a curva de velocidade descrita abaixo.



Onde:

1. Aceleração constante até que a velocidade máxima seja atingida;
2. Velocidade constante;
3. Desaceleração constante até o término do movimento.

Além disso, o caminho percorrido pelo centro de massa deve ser uma reta, cujo comprimento é dado pela equação a seguir:

$$d = \sqrt{(X_G - X_{G_i})^2 + (Z_G - Z_{G_i})^2} \qquad (12)$$

4.3.2 Ajustes nas curvas de velocidade

O movimento final da plataforma deve apresentar-se de forma contínua e suave, ou seja, devem ser evitados todos e quaisquer trancos. Para isso, a translação e as rotações devem terminar no mesmo instante. Isto pode ser feito por meio do seguinte procedimento:

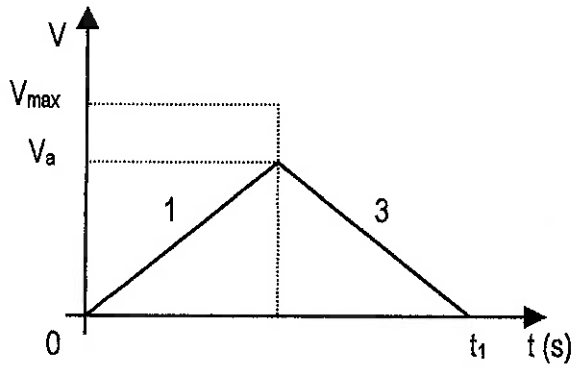
1) Calcula-se o tempo de deslocamento da plataforma para cada movimento separadamente considerando acelerações e velocidades máximas permitidas pelo software através das seguintes equações:

$$t = \frac{d}{V_{\max}} + \frac{V_{\max}}{a_{\max}} \qquad (13)$$

$$t = \frac{\theta}{\omega_{\max}} + \frac{\omega_{\max}}{\alpha_{\max}} \qquad (14)$$

Onde a equação (13) é usada para translação e a equação (14) para as rotações em torno dos eixos *x* e *y*.

Porém, pode ocorrer que para ser atingido o deslocamento desejado seja necessário que a desaceleração comece antes da velocidade máxima seja atingida:



Neste caso as seguintes equações serão utilizadas:

$$t = \sqrt{\frac{4.d}{a_{max}}} \qquad (15)$$

$$t = \sqrt{\frac{4.\theta}{\alpha_{max}}} \qquad (16)$$

Sendo t_1 o tempo de translação, t_2 o tempo de rotação no eixo *x* e t_3 o tempo de rotação no eixo *y*.

2) As correções nas velocidades e acelerações serão feitas conforme o seguinte regra: multiplicaremos $V_{máx}$, $\omega_{x\_máx}$ e $\omega_{y\_máx}$ por, respectivamente, t_1/t_{max}, t_2/t_{max} e t_3/t_{max}; e, $a_{máx}$, $\alpha_{x\_máx}$ e $\alpha_{y\_máx}$ por, $(t_1/t_{max})^2$, $(t_2/t_{max})^2$ e $(t_3/t_{max})^2$. Onde t_{max} é o maior valor entre t_1, t_2 e t_3.

4.3.3 Trajetória da Plataforma

Muitas vezes é conveniente poder especificar que a plataforma deve movimentar-se em linha reta, e para que isso ocorra será necessário especificar pontos intermediários de interpolação com uma pequena distância entre eles, sendo todos pertencentes a uma mesma linha reta. Isso fará com que a plataforma se mova aparentemente segundo uma linha reta, independente da função de interpolação escolhida para conectar esses pontos.

Em conjunto com as restrições espaciais, o usuário deve especificar também atributos temporais para o movimento, ou seja, o tempo de percurso entre dois pontos consecutivos. Isso é necessário para que o software calcule as posições dos atuadores de forma a realizar o movimento com as devidas velocidades e acelerações, sendo que o algoritmo básico é o seguinte:

$t = t_0$;

Para $t \leq t_f$ *faça;*

$t = t + \Delta t$;

Calcular a posição e orientação da plataforma no tempo t;

Calcular as posições correspondentes de todos os atuadores da plataforma no tempo t;

Fim;

5 RESULTADOS

5.1 Seleção dos Atuadores

A Plataforma de Stewart com Atuadores Fixos necessita de atuadores lineares. Conforme a aplicação, os atuadores lineares podem ser implementados por pistões hidráulicos ou por fusos de esferas recirculantes ou por pinhão-e-cremalherira ou por motores elétricos lineares.

O uso de pistões hidráulicos se faz necessário quando se quer alta força e rigidez mas baixas velocidades.

O uso do fuso de esfera recirculante possibilita grande precisão de posicionamento e altas velocidades mas menor rigidez.

A transmissão por pinhão-e-cremalheira possui melhor eficiência e é melhor adaptado quando se requer altas velocidades.

Tanto o fuso de esfera recirculante quanto pinhão-e-cremalheira exigem o uso de motores elétricos. Estes são relativamente baratos e de fácil obtenção, mas sua desvantagens são baixa eficiência de transmissão (aproximadamente 60%).

Motores elétricos lineares são rápidos e sua energia de alimentação e diretamente transformada em movimento linear, contudo possuem baixa potência.

Para o mecanismo em estudo os meios mais apropriados são os fusos de esferas e o pinhão-e-cremalheira. O ideal seria a escolha do fuso de esferas, no entanto devido ao seu alto custo

optou-se por uma alternativa similar porém com menor precisão e maior atrito, que é uma rosca-sem-fim associada a um motor elétrico.

5.1.1 Parâmetros para seleção

- Massa transportada: m = 10Kg
- Comprimento do deslocamento: l_s = 400mm
- Velocidade máxima: V_{max} = 1000 mm/s
- Aceleração máxima: a_{max} = 9,8 m/s²
- Tempo de aceleração: t = V_{max}/a_{max} = 0,102s
- Passo do fuso: l = 20 mm
- Diâmetro do fuso: d_r = 14 mm
- Momento de inércia do motor (assumida): J_m = 0,001 Kg.m²
- Resistência da guia linear: f = 20 N

5.1.2 Rotação máxima do motor

$$N_{max} = \frac{V_{max}}{l} = 50Hz = 3000rpm$$

5.1.3 Seleção do método de montagem

Para uso com um comprimento de deslocamento de 400mm e velocidade máxima de 1000mm/s é aconselhável o usos do método de montagem fixed-supported conforme figura abaixo.

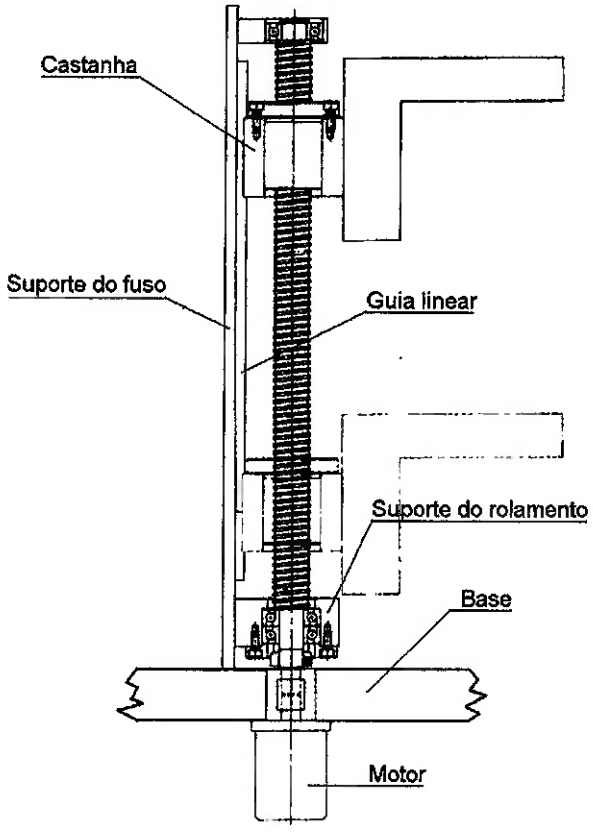


Figura 5.1: Esquema de Montagem dos atuadores

5.1.4 Considerações sobre a carga axial

Para um movimento de ascensão constante temos: $F_{a1} = f + g.m = 118N$

Para ascensão acelerada temos: $F_{a2} = a_{max}.m + F_{a1} = 236N$

Para a ascensão desacelerada temos: $F_{a3} = F_{a1} - a_{max}.m = 0N$

5.1.5 Considerações sobre o torque de rotação devido a carga axial

Dada a equação: $T_i = \frac{F_a.l}{2\pi.\eta}.10^{-3}$

E considerando as forças do item 2.2.4, temos:

Para ascensão a velocidade constante: ***T_1 = 0,42 N.m***

Para ascensão acelerada: ***T_2 = 0,84 N.m***

Para ascensão desacelerada: ***T_3 = 0 N.m***

5.1.6 Consideração do torque requerido na aceleração devido a inércia do sistema

Momento de inércia:

$$J = m.\left(\frac{l}{2\pi}\right)^2.10^{-6} + J_s$$

onde J_s é o momento de inércia do fuso e vale 0,000025 Kg.m².

$$J = 0{,}00006 Kg.m^2$$

Aceleração angular:

$$\dot{\omega} = \frac{2\pi.N}{60.t} = 3079 \frac{rad}{s^2}$$

Utilizando os parâmetros dados acima, o torque devido a rotação requerido para o movimento de aceleração é dado por:

$$T_7 = (J + J_m).\dot{\omega} = 3{,}16 N.m$$

5.1.7 Cálculo do torque máximo

O torque máximo se dá no movimento de ascensão acelerado e pode ser obtido pela equação:

$$T_{k1} = T_2 + T_7 = 4 N.m$$

5.2 Seleção da Juntas

A Plataforma selecionada deve possuir quatro graus de liberdade, que serão controlados variando-se os comprimentos dos quatro atuadores fixos. O mecanismo deve ser projetado de tal forma que as juntas que ligam atuadores, colunas e plataforma possua a quantidade de graus de liberdade suficiente para permitir o movimento desejado. Da mesma forma, o mecanismo deverá ter zero graus de liberdade quando estiver fixo em uma posição.

Pode-se obter os graus de liberdade de um mecanismo no espaço pela seguinte equação:

$$F = \lambda(l - j - 1) + \sum_{i=1}^{j} f_i - I_d \quad (12)$$

Onde:

F = Número de graus de liberdade do mecanismo

λ = Graus de liberdade do espaço no qual o mecanismo opera (para movimento espacial $\lambda = 6$, e para movimento num plano ou numa superfície $\lambda = 3$)

l = número de peças

j = número de articulações

f_i = Graus de liberdade da i-ésima articulação

I_d = Graus de liberdade redundante ou passivo

Para que o mecanismo possuísse zero graus de liberdade quando os atuadores estivessem parados foram escolhidas as seguintes articulações:

- Para as articulações A, B, C e D foram selecionados juntas esféricas que possuem três graus de liberdade;

Figura 5.2: Juntas esféricas

- Para as articulações I e H foram escolhidas juntas universais que possuem dois graus de liberdade;

Figura 5.3: Juntas universais

- Para as articulações E e F optou-se por juntas simples que possuem apenas um grau de liberdade.

Para a Plataforma utilizada nota-se que l = 10, j = 8, $f_A = f_B = f_C = f_D = 3$, $f_H = f_I = 2$, $f_E = f_F = 1$ e $I_d = 0$. Substituindo estes valores na equação (12) temos:

$$F = 6(6 - 8 - 1) + 4.3 + 2.2 + 2.1 - 0 = 0$$

O mecanismo possui zero graus de liberdade quando os atuadores estão imóveis logo o sistema é estável.

5.3 Seleção dos demais Componentes

5.3.1 Plataforma móvel

Para a construção da plataforma móvel optou-se pele uso de alumínio devido ao seu baixo peso específico. Esta solução permite uma diminuição na inércia do mecanismo e um aumento da potência transferida ao posicionador.

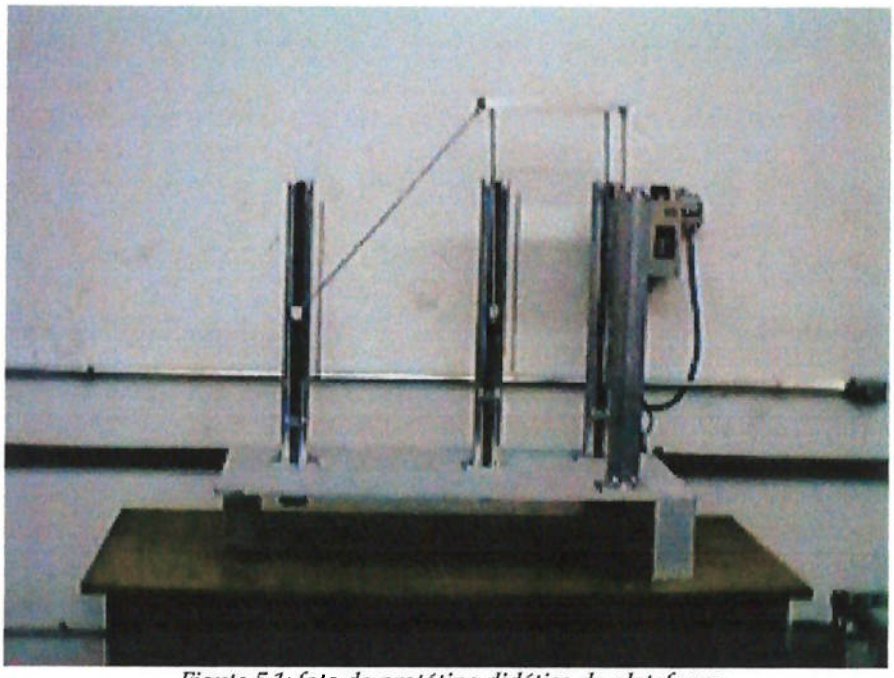
As dimensões tidas como razoáveis para a maquete de um simulador de vôo foram um retângulo de 297 por 210 mm e de espessura 2 mm.

5.3.2 Castanha

A castanha é a peça onde estão fixadas as articulações. Estas foram fabricadas especialmente para este projeto uma vez que possuem diversas particularidades.

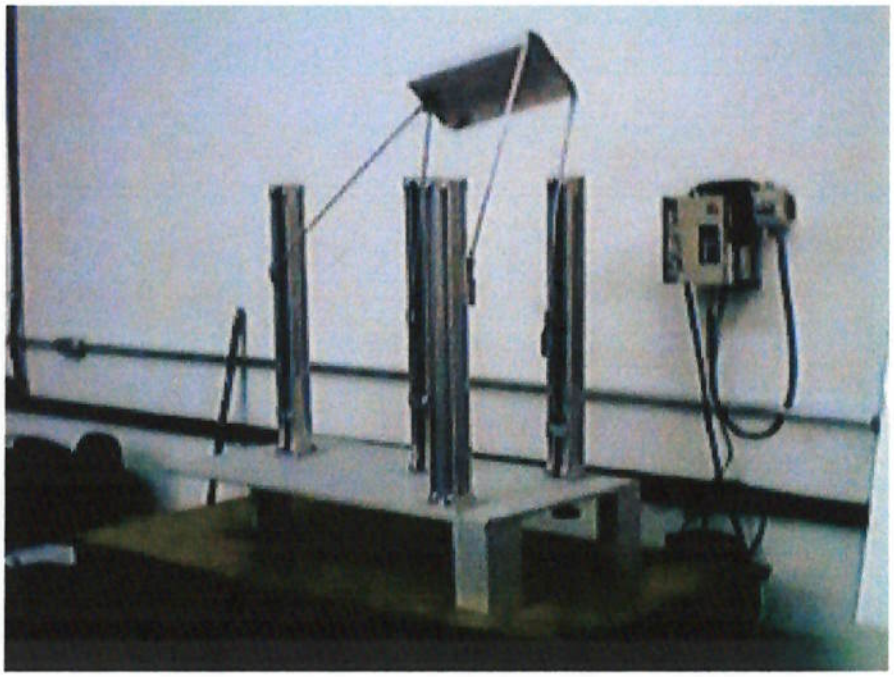
5.4 Protótipo

Como resultado dos estudos realizados para o desenvolvimento deste trabalho foi construído um protótipo do posicionador proposto para assim poder verificar a eficiência do software desenvolvido bem como analisar as posições de singularidade da Plataforma de Stewart, comprovando na prática a teoria desenvolvida.

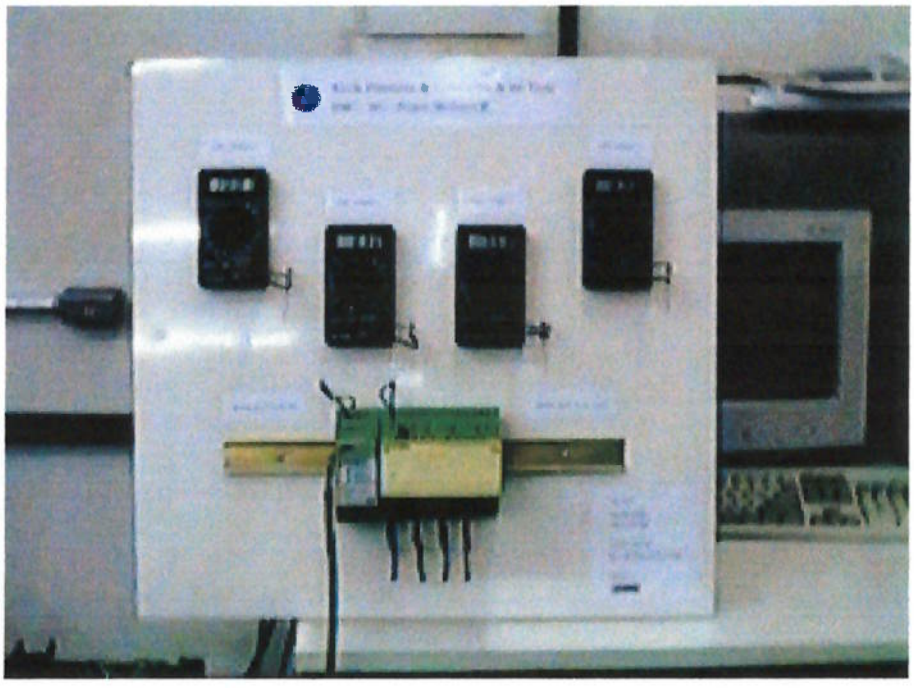


Figuta 5.1: foto do protótipo didático da plataforma

Figuta 5.2: foto do protótipo didático da plataforma



Figuta 5.3: foto do protótipo didático da plataforma



Figuta 5.4: foto do sistema de monitoração dos sinais de saída da placa controladora

5.5 Software de Movimentação da Plataforma

O Software constitui a parte inicial do trabalho, representando a aquisição e tratamento dos dados relativos ao movimento da plataforma móvel. Os dados gerados são então convenientemente convertidos para que seja feito o acionamento dos atuadores da plataforma.

Como citado anteriormente, o movimento é descrito pelas coordenadas iniciais e finais do Centro de Massa da Placa, seguida de duas rotações em torno do mesmo.

Não foi escopo deste trabalho descrever um controle de trajetória para o movimento. Foi admitido que os deslocamentos seriam linearmente acelerados até uma velocidade máxima e depois desacelerados com o mesmo valor de aceleração.

Cabe ressaltar que o comportamento do centro de massa (citado acima), não se estende aos atuadores, cujos movimentos apresentam um perfil essencialmente não-linear, garantindo porem a linearidade do deslocamento do Centro de Massa.

O sinal de controle, entretanto, atua diretamente sobre os atuadores. Isso aliado a não-linearidade do movimento implica na necessidade de um valor pequeno para o intervalo de interpolação, para que o acionamento dos atuadores seja feito da forma mais suave possível, evitando trancos ou até mesmo danos ao equipamento.

Portanto, movimentos mais complexos exigem um menor tempo de amostragem, porém geram um banco de dados significativamente maior e um maior tempo de processamento.

5.5.1 Seleção da Linguagem

Seguindo as atuais tendências de programação, o software foi desenvolvido em linguagem orientada a objeto. Foram selecionados três compiladores para análise: o C++ Builder, o Delphi e o Visual Basic.

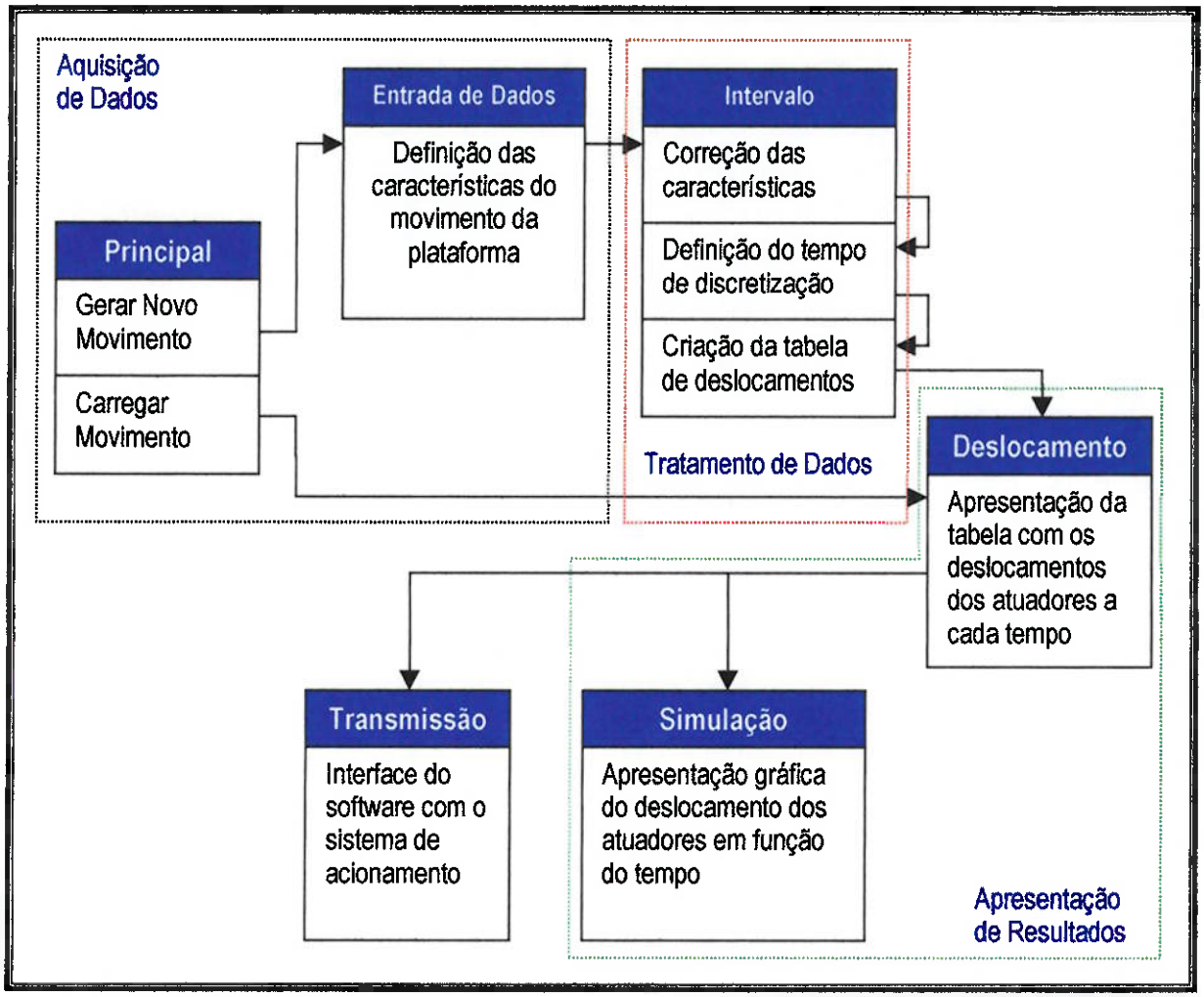
Os critérios adotados para a seleção foram a facilidade de manipulação de banco de dados, uma vez que este recurso é extremamente utilizado durante o processamento, e a facilidade do desenvolvimento de uma interface amigável com o usuário (GUI).

Tendo em vista que a interação do software com o Hardware foi fornecida com o equipamento da Phoenix, a necessidade de um compilador que trabalhasse bem esta interface foi eliminada. Dessa forma, eliminamos também a opção do C++ Builder, já que sua vantagem com relação as outras linguagens é exatamente esta.
O Delphi apresenta, dentre todos, maior facilidade de manipulação de banco de dados, porém não apresenta uma facilidade tão grande como a do Visual Basic no desenvolvimento de interfaces com o usuário.

De acordo com os critérios acima, selecionamos o Visual Basic como base de programação. Recursos ActiveX e integração OLE com os demais softwares comumente utilizados também contribuíram para a escolha do compilador, já prevendo um futuro desenvolvimento do mesmo, que com certeza exigirá estes recursos.

5.5.2 Estrutura do Software

O Software é constituído de 6 formulários, que obedecem a seguinte estrutura:



Do diagrama acima, podemos identificar quatro classes de operações que compõem a estrutura do programa:

- Aquisição de Dados
- Tratamento de Dados
- Apresentação de Resultados

- Transmissão de Dados

Essas classes são definidas a partir do tipo de atividade que é realizada em cada formulário, seja pelo tipo de entrada do usuário, seja pelo código e pelas operações matemáticas incluídas.

Cada classe será discutida e detalhada de uma forma mais completa nos itens que seguem:

5.5.3 Aquisição de Dados

Esta classe é composta por dois formulários: o formulário **Principal** e o formulário **Ent_Dados**.

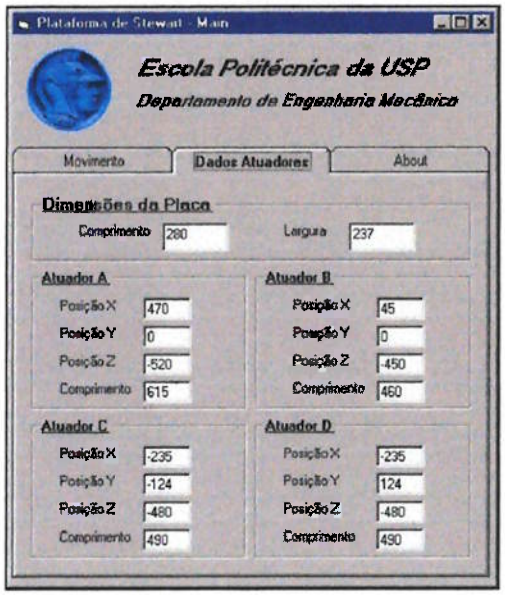
Formulário Principal

O Formulário Principal é o formulário inicial do software. Nesta etapa é feita a aquisição dos dados referentes à plataforma.

No primeiro item deste formulário ("Movimento") é permitido ao usuário optar por gerar um novo movimento ou carregar uma tabela do disco, quando o movimento já foi configurado anteriormente.

Caso o usuário opte por utilizar um movimento pré-definido, não há a necessidade de o programa tratar os dados. Nesse caso, o usuário é levado diretamente ao formulário de apresentação de Resultados.

Se o usuário optar por gerar um novo movimento, os dados a respeito das dimensões da plataforma (largura e comprimento) e o posicionamento dos atuadores podem ser definidos no segundo item deste formulário, chamado "Dados Atuadores".



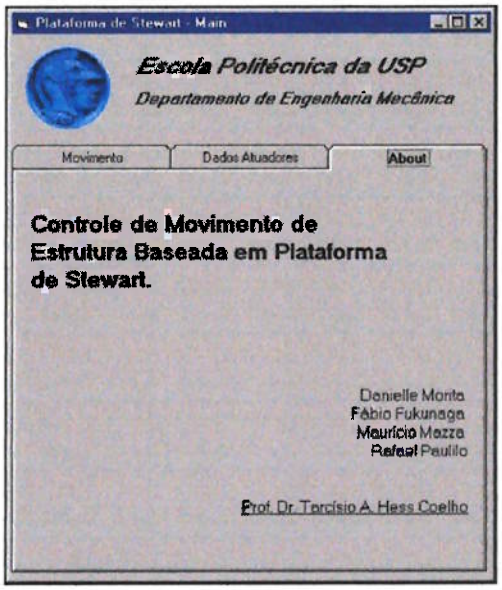
A nomenclatura dos atuadores e o seu sistema de orientação segue o que foi exposto na apresentação teórica do trabalho.

Os dados introduzidos nesta etapa estão sujeitos a alterações geométricas na plataforma, sendo portanto, praticamente constantes.

Cabe ressaltar que no caso de alguma alteração geométrica ser feita, todos os movimentos pré-definidos, perdem sua validade, uma vez que estão atrelados a esses valores de posicionamento dos atuadores. Nesse caso, novos movimentos com as mesmas características devem ser gerados, de forma a contemplar essas modificações.

O último item deste formulário contem os dados referentes ao desenvolvimento do software, como versão, grupo de trabalho e professor orientador.

Quaisquer alterações de versão do programa, componentes do grupo ou do professor orientador estão sujeitas a aprovação do professor e só podem ser feitas através de alteração do código fonte.



5.5.4 Formulário Ent_dados

Este formulário só é utilizado caso o usuário tenha optado por gerar um novo movimento no formulário principal.

Nesta etapa são introduzidos os valores dos deslocamentos lineares e/ou angulares da plataforma, e suas características.

Valor	Características
Deslocamento Linear ao longo do Eixo X (x)	Velocidade Linear / Aceleração Linear
Deslocamento Linear ao longo do Eixo Z (z)	Velocidade Linear / Aceleração Linear
Rotação em torno do eixo X (θx)	Velocidade Angular / Aceleração Angular
Rotação em torno do eixo Y (θy)	Velocidade Angular / Aceleração Angular

Deslocamentos Lineares: mm Velocidade Linear: mm/s Aceleração Linear: mm/s²

Deslocamentos Angulares: rad Velocidade Angular: rad/s Aceleração Angulra: rad/s²

Cabe ressaltar que os valores dessas características são valores máximos, sendo necessária uma correção durante o processamento.

Os valores máximos para as características acima são sempre aplicados aos pares correspondentes. Por exemplo, a máxima aceleração angular é a mesma para os dois deslocamentos angulares. Esses valores só se alteram após feita a correção.

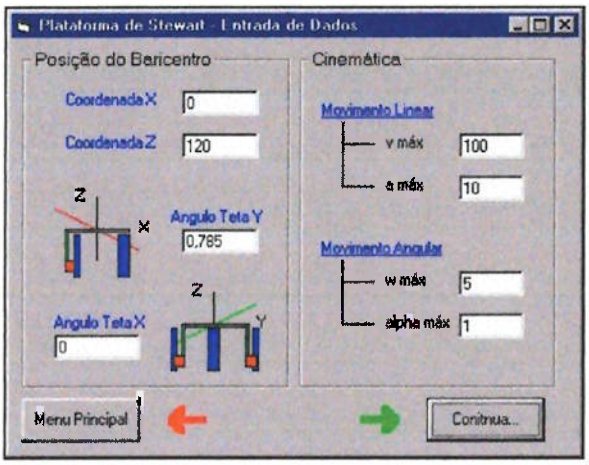
Os deslocamentos lineares são condensados em um só, representado pela diagonal formada entre os vetores (x,Xg) e (z,Zg), definidos pelas posições final (x,z) e inicial (Xg,Zg) do Baricentro. Esse deslocamento será daqui por diante tratado tratado por "d".

Nota-se, que como não há movimento linear em Y nem rotação em torno de Z, esses valores não são parâmetros de entrada do programa, sendo admitidos como zero quando necessário.

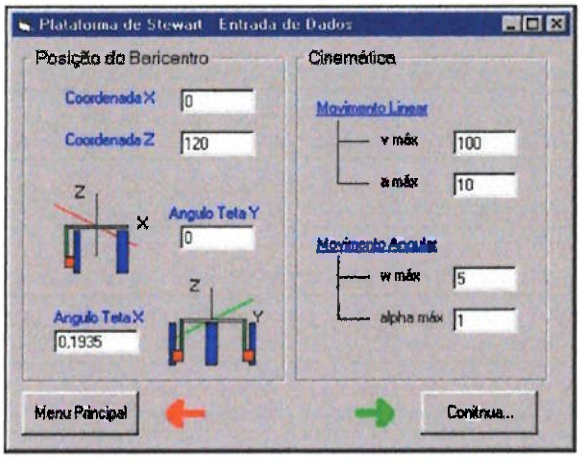
Durante todo o processamento, o valor da posição inicial do Baricentro também é considerada nula. Mais considerações sobre esse fato serão discutidas posteriormente.

A seguir ilustramos o formulário de entrada para dois casos de estudo:

- Deslocamento Z : 120,0 mm
- Deslocamento X : 0,0 mm
- Rotação θx:: 0,0 rad
- Rotação θy:: 0,785 rad (45°)



- Deslocamento Z : 120,0 mm
- Deslocamento X : 0,0 mm
- Rotação θx:: 0,0 rad
- Rotação θy:: 0,193 rad (11,25°)



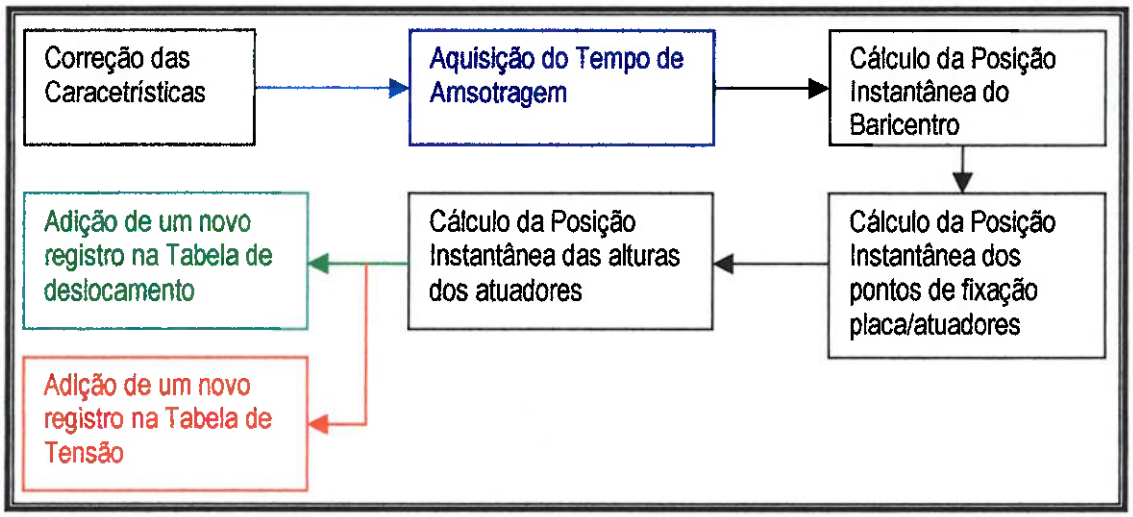
Para efeito dos cálculos, as velocidades e acelerações não podem ser nulas. Caso isto ocorra, o programa acusará erro durante o processamento.

5.5.5 Tratamento de Dados

Essa classe é representada pelo formulário **Intervalo.**

Embora neste formulário haja aquisição do tempo de amostragem/discretização, é nele que se encontram a maioria das operações relativas ao cálculo dos deslocamentos.

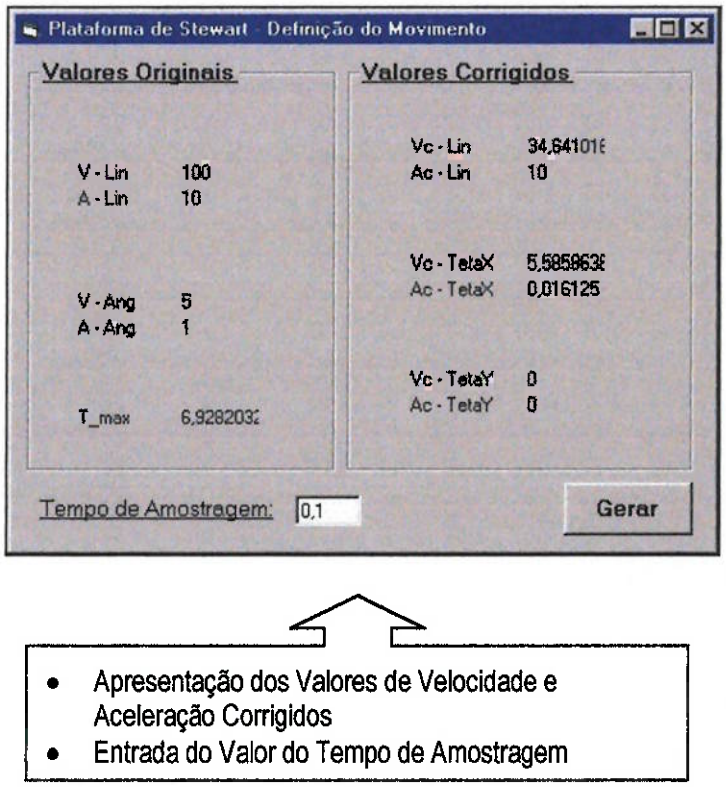
Os cálculos seguem a seguinte ordem:



A partir dos valores das características dos movimentos lineares/angulares, o programa calcula o tempo gasto para cada movimento.

O maior tempo (de cada deslocamento) é considerado como o tempo total do movimento. O tempo de cada deslocamento é então alterado de forma que todas as componentes do movimento terminem simultaneamente.

Em função do novo tempo total, são também corrigidas as características de cada deslocamento.



O usuário pode, a partir do tempo total, estimar um tempo de amostragem ou de distcretização para o movimento. É recomendado que este tempo tenha uma ordem de grandeza entre duas ou três vezes menor que o tempo total, nunca esquecendo a relação entre tempo de amostragem, suavidade do movimento e tempo de processamento.

Conforme explicitado na introdução teórica do trabalho, o programa calcula, em função do tempo, os valores instantâneos para o deslocamento dos atuadores.

São então criadas duas tabelas de dados, sempre indexadas pelo tempo, onde cada registro contem os deslocamentos e os correspondentes valores de tensão para os atuadores.

5.5.6 Apresentação dos Resultados

Esta classe é composta por dois formulários: o formulário **Deslocamentos** e o formulário **Simula.**

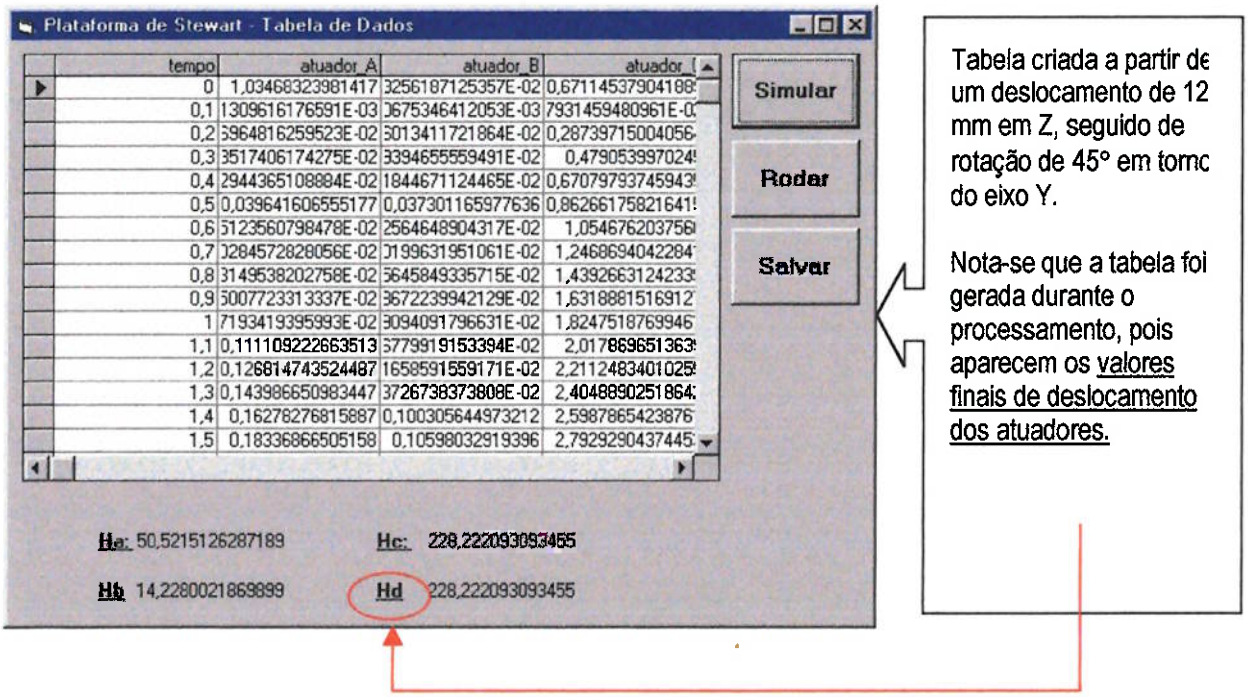
Em ambos os casos podem ser vistas as variações dos valores calculados anteriormente em função do tempo.

Deslocamentos

Neste formulário é apresentada a tabela de deslocamentos em função do tempo para ambas as opções do Formulário Principal.

Para um movimento carregado diretamente do arquivo, entretanto, não é possível ver os valores finais das alturas dos atuadores. Essa opção só está disponível neste formulário se o movimento tiver sido gerado durante o processamento.

A seguir vemos um exemplo do Formulário Deslocamentos, para o caso da rotação em tono do eixo Y de 45°.



Simula

Este formulário é responsável pela simulação e apresentação gráfica dos cálculos efetuados pelo programa.

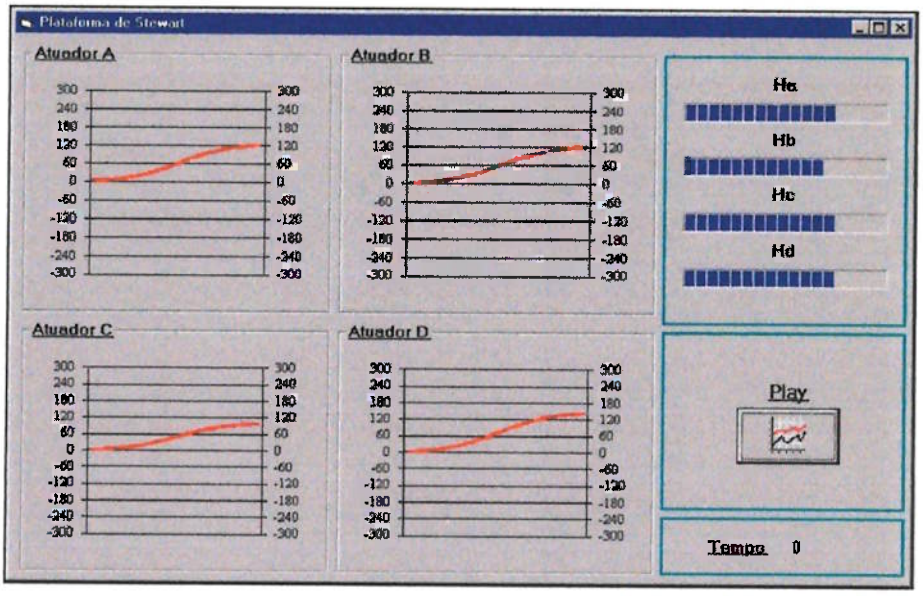
A partir deste formulário, o usuário pode analisar a curva de deslocamento dos atuadores em função do tempo e verificar suas posições finais, para ambos os casos (movimentos gerados ou carregados).

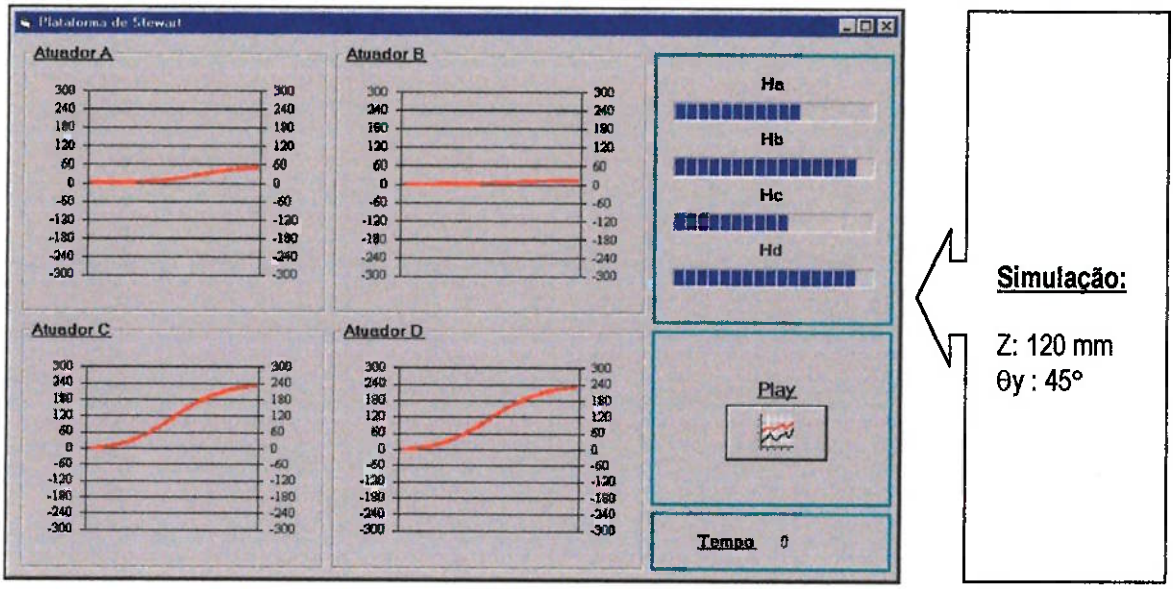
A apresentação da curva tem sua importância na não-linearidade do movimento dos atuadores.

Para exemplificar o que foi citado acima, podemos ver os dois exemplos de simulação para os mesmos casos do item "Ent_Dados".

Simulação:

Z: 120 mm
θx : 11,25°





5.5.7 Transmissão de Dados

A simulação da transmissão dos dados gerados para os atuadores está descrita mo capítulo 10.

5.6 Observações

Existem alguns pontos que devem ser esclarecidos, antes de dar continuação ao trabalho. Os valores que possuem casas decimais devem ser separados por VÍRGULAS, e não por pontos, devido ao tipo de conversão efetuada pelo programa.

Outro ponto que merece uma certa discussão é a possibilidade de se gerar movimentos mais complexos utilizando este software.

Uma forma de criar um movimento complexo é gerar um novo movimento, utilizando como posições iniciais dos atuadores as posições finais do movimento anterior. Assim, gera-se uma nova tabela, que leva a plataforma a uma posição determinada em relação a situação anterior.

A nova tabela deve então ser copiada e colada no arquivo inicial *.mdb utilizando um software de manipulação de banco de dados, como o Microsoft Access. A nova tabela, quando carregada, será interpretada como um apenas um movimento.

6 SUBSISTEMA ELETRO-ELETRÔNICO

Dentro desta descrição, o desenvolvimento do sub-sistema eletro-eletrônico consiste na determinação dos parâmetros dos controladores e na implementação de um software de controle.

Esquema do sistema em estudo:

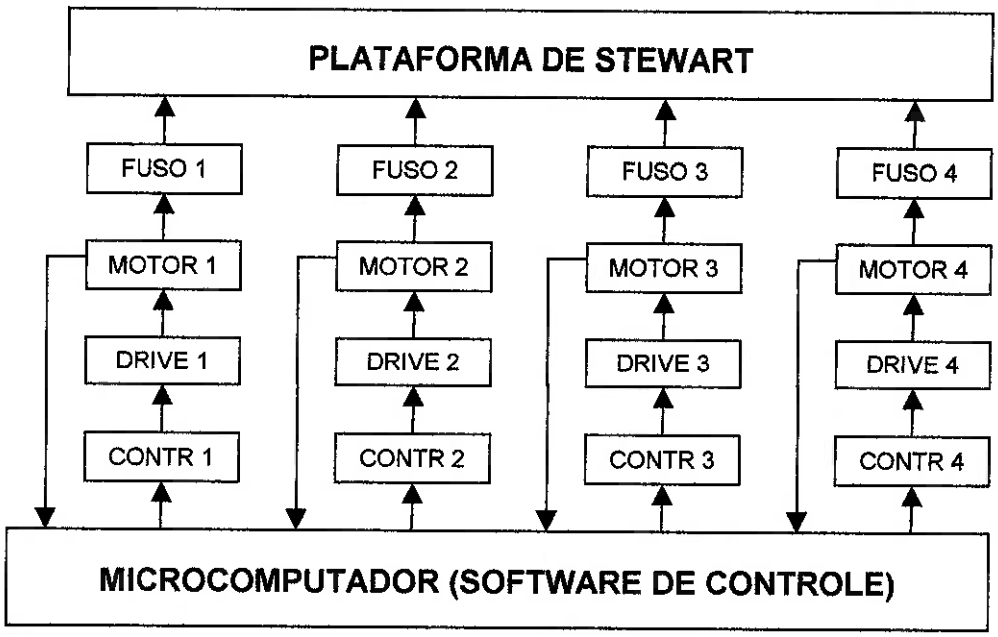


Figura 6.1: Descrição do sub-sistema eletro-eletrônico

✓ ***Plataforma de Stewart***: a plataforma a ser construída faz parte do projeto do sub-sistema mecânico desenvolvido em paralelo com este trabalho. Conforme já descrito anteriormente, este mecanismo é dotado de 4 colunas de comprimento variável, conferindo 4 graus de mobilidade.

✓ ***Atuadores***: são os elementos responsáveis pela variação do comprimento das colunas da plataforma, proporcionando-lhes o deslocamento necessário em cada movimento. Podem ser, por exemplo, cilindros pneumáticos ou hidráulicos, motores elétricos com fusos de esferas ou com pinhão e cremalheira.

✓ ***Controladores***: dispositivos de controle do movimento da plataforma segundo um sinal de referência.

✓ ***Computador***: dispositivo de realização do controle do movimento da plataforma, envia o sinal de controle para os controladores através de um software de controle.

✓ ***Fusos***: integrantes do sistema de atuação, são responsáveis pelo movimento das colunas da plataforma

✓ ***Motores***: também fazem parte do dispositivo de atuação e proporcionam o movimento de rotação dos fusos. Sensores acoplados ao eixo desses motores gerarão sinais de realimentação ao controlador.

✓ ***Drivers dos Motores:*** interface entre os motores e seus controladores. São responsáveis por fornecer alimentação de tensão aos motores.

6.1 Seleção de Componentes

Nesta fase serão analisados os valores atribuídos aos parâmetros mecânicos (vide tabela 3.1) da planta com o objetivo de se selecionar os motores adequados que servirão como atuadores da plataforma e de determinar a melhor forma de controle para o simulador de vôo.

6.1.1 Escolha do Tipo de Motor

A plataforma posicionadora é atuada por 4 barras rígidas que são movimentadas por um "carro" guiado através de um fuso de esferas. Este fuso é rotacionado por um motor, cuja seleção será feita nessa seção.

Para a escolha do melhor tipo de motor para essa aplicação foram levantadas 3 possibilidades de acionamento. Os fatores considerados na escolha do melhor acionamento foram:

- ✓ Método, complexidade e qualidade do controle de velocidade e de posição;
- ✓ Preço;
- ✓ Possibilidade de aplicação futura em plantas mais complexas e robustas.

A primeira a ser considerada foi a de se utilizar *motores de passo.*

MOTORES DE PASSO

Características:

- os motores de passo caracterizam-se pela boa atuação em sistemas de *malha aberta* proporcionando precisão de posicionamento. O fato de atuar em malha aberta faz de seu controle bastante *simples.*

Desvantagens:

- podem ocasionar, em determinadas faixas de velocidade, *instabilidade* na rotação (oscilação) cuja eliminação é problemática. Uma solução seria fechar a malha de controle;

- são mais caros e robustos que os motores CC ou AC equivalentes.

As outras 2 possibilidades referem-se aos de motores de corrente alternada (AC) síncronos e aos de corrente contínua (CC):

MOTORES SÍNCRONOS

Características:

- são motores de velocidade *rigorosamente* constante com a frequência de excitação da rede. Os pólos do rotor seguem o campo girante imposto ao estator pela rede de alimentação. Assim, a velocidade do motor é a do campo girante (velocidade síncrona).
- o controle de posicionamento com esses motores é feito utilizando-se *encoders* ou *resolvers.*
- de maneira geral são mais baratos que os motores CC equivalentes.

Desvantagens:

- o método de controle é de complexidade bastante elevada.

MOTORES CC

Características:

- são motores de velocidade ajustável. A velocidade de um motor de corrente contínua pode ser ajustada de acordo com a corrente de armadura, tensão de armadura ou pela variação do fluxo no entreferro.
- o controle de posicionamento com esses motores é feito utilizando-se *encoders, resolver* ou *tacômetros.*
- a implementação de um controle para motores CC é bem menos complexa que para os motores síncronos.

Desvantagens:

- são mais caros que os motores síncronos equivalentes.
- a velocidade do rotor depende não só da corrente de armadura, tensão de armadura ou do fluxo no entreferro, mas também da carga nele submetida;

- necessidade de frequente manutenção.

A possibilidade de aplicação futura em plantas mais complexas e robustas para motores de passo é limitada, pois é recomendável restringir sua utilização em aplicações com cargas reduzidas (tornam-se muito caros e grandes para aplicações mais "pesadas"). Os motores AC e CC são flexíveis no sentido de poderem ser aplicados em situações de cargas reduzidas ou elevadas, em níveis de velocidade diversos.

Optou-se então pelo **motor CC**, já que o mesmo oferece um controle de velocidade e posição de qualidade e mais simples que os dos motores AC, podendo ser largamente utilizado em aplicações futuras em plantas de maiores complexidade e robustez, além de ser mais barato que o motor de passo equivalente, apesar de mais caro que um AC equivalente.

Resumindo, serão utilizados **4 motores CC** (cada um com um driver e um controlador);

6.1.2 Vantagens e Desvantagens dos Motores CC

a) Vantagens

- ✓ Alta flexibilidade através de seus vários tipos de excitações;
- ✓ Relativa simplicidade dos modernos conversores de corrente contínua.

b) Desvantagens

- ✓ Preço (mais caros que os motores de indução compatíveis);

- ✓ presença do comutador exige maior necessidade de manutenção;
- ✓ Comutação de corrente por meio mecânico implica em arcos e faíscas, impedindo sua utilização em ambientes perigosos.
- ✓ Limite de tensão de alimentação para uma boa comutação

6.1.3 Modelagem do motor DC

Equações

As seguintes equações descrevem as grandezas envolvidas no sistema a ser controlado:

- ✓ Momento de Inércia total do mecanismo (J):

$$J = \frac{m.l^2}{4.\pi^2} + J_s + J_r \qquad (1)$$

onde:

- ✓ Velocidade angular (ϖ):

$$\varpi = \frac{T}{J} \qquad (2)$$

onde: T = Torque resultante;

J = Momento de Inércia total do mecanismo

- ✓ Corrente gerada pelo motor (i_a):

$$i_a = \frac{V - K_e.\varpi}{L_a.s + R_a} \qquad (3)$$

✓ Torque elétrico gerado pelo motor (T_e):

$$T_e = K_t . i_a = K_t . \frac{V - K_e . \varpi}{L_a . s + R_a} \quad (4)$$

✓ Torque resistivo por atrito (T_a):

$$T_a = K_a . \varpi \quad (5)$$

✓ Torque resistivo por pré-carga (T_d):

$$T_d = \text{constante} \quad (6)$$

✓ Torque resistivo por inércia (T_i):

$$T_i = \frac{m.g..l}{2.\pi.\mu} \quad (7)$$

6.2 Seleção do motor CC para a Aplicação

O processo de seleção do motor CC a ser utilizado teve um caráter iterativo e foi feito em conjunto a equipe do subsistema mecânico.

Os primeiros parâmetros mecânicos utilizados foram:

Parâmetros	***Valor***
Rotação do fuso de esferas máxima (rpm)	3 000
Passo do fuso de esferas (mm)	20
Potência fornecida pelo motor (kW)	3,3
Torque máximo (Nm)	10,5

Tabela 4.1

O motor selecionado com base nesses parâmetros possuí as seguintes características:

Parâmetros	***Valor***
Rotação nominal (rpm)	3320
Potência nominal (kW)	3,8
Toque nominal (Nm)	10,9
Rotação mámima (rpm)	4250
Momento de Inércia ($kg.m^2$)	0,016
Corrente nominal (A)	14,8
Eficiência (%)	78
Resistência de armadura (ohms)	1,39
Indutância de armadura (mH)	5,30
Indutância série (mH)	34

Tabela 4.2

Uma vez selecionado o motor, realizou-se uma simulação no MatLab para análise de comportamento dos motores em conjunto com a planta.

Ao final da simulação concluiu-se que o motor selecionado não suportava os níveis de corrente e de torque necessários para movimentar a carga segundo os parâmetros de velocidades e acelerações máximas. Conforme selecionávamos um motor capaz de fornecer maior torque, o seu momento de inércia elevado sempre inviabilizava sua utilização. Este processo foi repetido diversas vezes e não se obteve sucesso.

A solução encontrada foi modificar alguns parâmetros mecânicos, conforme ilustra a tabela a eguir:

Parâmetros	***Valor Antigo***	***Novo Valor***
Rotação do fuso de esferas máxima (rpm)	3 000	**1 000**
Passo do fuso de esferas (mm)	20	**60**
Potência fornecida pelo motor (kW)	3,3	**9,8**
Torque máximo (Nm)	10,5	**93,24**

Tabela 4.3

A manipulação dos parâmetros foi feita visando-se a utilização de um redutor acoplado ao motor, o qual proporcionaria maior torque mediante diminuição de velocidade de rotação dos fusos. O motor selecionado com base nos novos parâmetros possui as seguintes características:

Parâmetros	***Valor***
Rotação nominal (rpm)	3 080
Potência nominal (kW)	10,7
Toque nominal (Nm)	33,3
Rotação máxima (rpm)	4 000
Momento de Inércia (kg.m^2)	0,042
Corrente nominal (A)	40,0
Eficiência (%)	84
Resistência de armadura (ohms)	0,397
Indutância de armadura (mH)	3,2
Indutância série (mH)	15

Tabela 4.4

Será utilizada uma redução de 3:1, o que reduz a rotação máxima do fuso para aproximadamente 1 027 rpm e aumenta seu torque máximo para 100 Nm. Mais uma vez realizou-se uma simulação no MatLab para análise de comportamento do novo motor em

conjunto com a planta. Os resultados obtidos estão descritos no item 5 (Simulação) desse trabalho.

6.3 Simulação do Sistema

Conforme descrito anteriormente, a planta do sistema mecânico e eletro-eletrônico é formado pelos seguintes elementos:

- ✓ Plataforma móvel (mesa+colunas);
- ✓ Motores DC;

O sistema a ser controlado pode ser dividido em quatro partes iguais, sendo cada uma constituída pelo seguinte conjunto:

- ✓ Coluna de deslocamento da mesa;
- ✓ Mecanismo de atuação (motor+fuso);
- ✓ Carga a ser movimentada (parte da massa da mesa que atua sobre a determinada coluna).

Com relação ao último item, fez-se aqui uma simplificação de projeto: partiu-se da hipótese de massa fixa atuando em cada coluna. É claro que na realidade a carga distribuída em cada base de sustentação da plataforma varia conforme o seu movimento, porém uma modelagem considerando esta situação traria ao projeto os seguintes inconvenientes:

- ✓ A parametrização da relação entre carga atuante em cada haste e seu movimento é demasiadamente complexa, envolvendo inclusive dependência entre o movimento de cada coluna, estando este tipo de estudo completamente fora do escopo de um projeto de graduação;
- ✓ A simplificação para massa fixa não acarreta mudança significativa no comportamento do sistema, nem invalida o projeto a ser executado;

Vale lembrar também que a planta possui controle do tipo realimentação unitária negativa. Assim o modelo de controle da planta é descrito da seguinte forma:

- ✓ O sistema recebe como entrada o deslocamento linear (h) da coluna desejado;
- ✓ Este deslocamento é convertido para o equivalente em tensão (V) para a entrada do motor;
- ✓ O motor então oferece o torque necessário para o movimento (T_e), que é transmitido para a plataforma através de uma redução;
- ✓ Soma-se à entrada da plataforma os torques resistivos de pré-carga (T_d), de atrito (T_a) e de inércia do sistema (Ti)
- ✓ O torque resultante é convertido para o equivalente em velocidade angular (ϖ);
- ✓ A velocidade angular resultante realimenta a entrada de tensão do motor
- ✓ A velocidade angular também é convertida pelo seu equivalente em deslocamento linear, que realimenta a entrada do sistema.

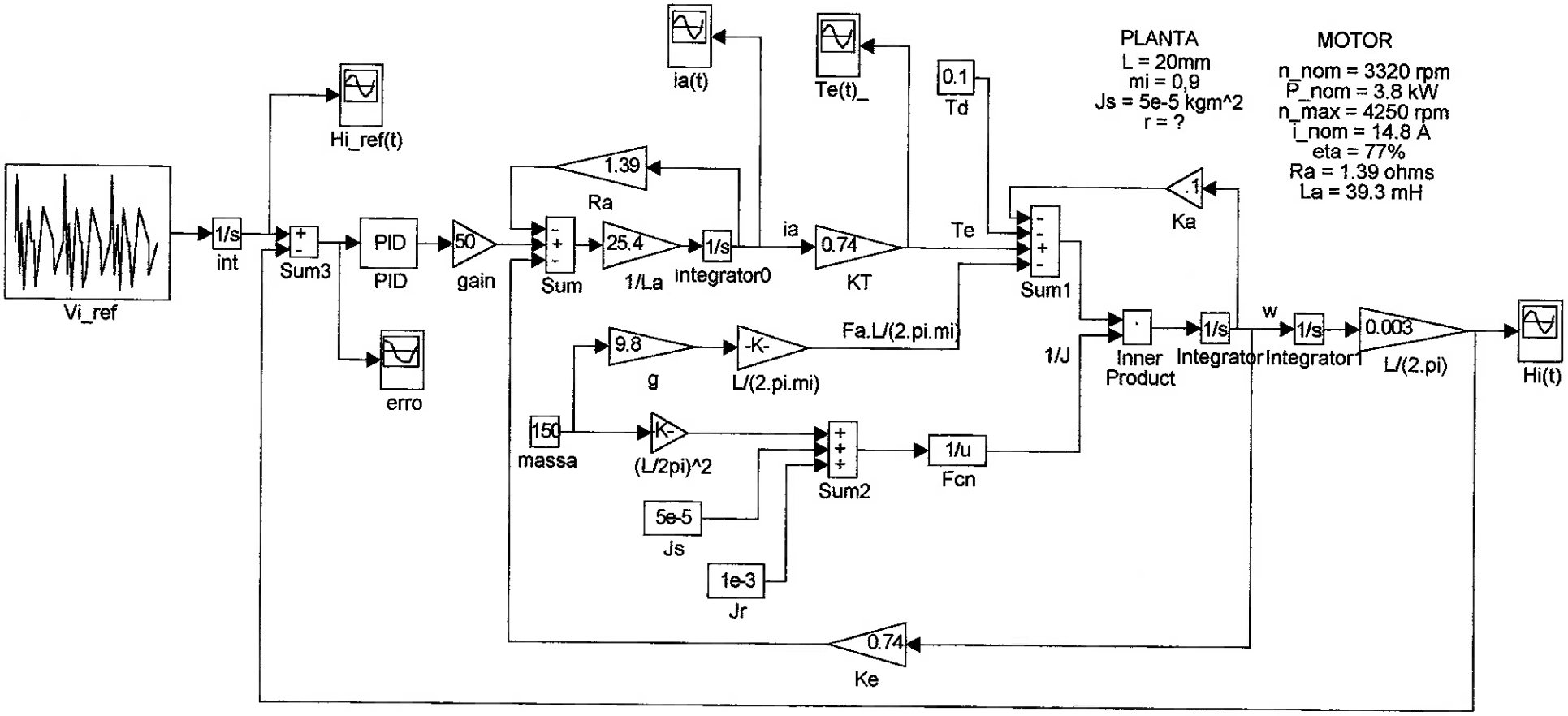
A análise do comportamento do sistema é feita através do relacionamento entre a tensão de entrada no motor e a velocidade de angular gerada. A velocidade angular do sistema pode ser descrito por:

$$\varpi = \frac{1}{J.s}[K_t.(\frac{V - K_e.\varpi}{L_a.s + R_a}) - \frac{m.g.l}{2.\pi.\mu} - T_d - K_a.\varpi] \quad (9)$$

Nota-se em (9) a presença de dois termos constantes. Por simplificação, desprezou-se estes termos na análise do comportamento transitório do sistema (onde a função de transferência será utilizada, conforme descrito adiante), ficando a planta descrita então pela seguinte função de transferência:

$$\frac{\varpi}{V} = \frac{K_t}{J.L_a.s^2 + (J.R_a + K_a.L_a).s + K_t.K_e + K_a.L_a} \quad (10)$$

6.3.1 Diagrama de Blocos da Planta Controlada



6.3.2 Modelagem do controlador do sistema

A partir das definições anteriores, o projeto do controlador do sistema deve levar em conta a análise do seu comportamento em transitório e em regime permanente. Através do MatLab Simulink, simulou-se o sistema para uma entrada degrau, a fim de se obter um compensador que atende-se as seguintes especificações:

- ✓ Máximo sobresinal : $M_p = 0.2$;
- ✓ Tempo de subida : $t_r = 0,5$ s;
- ✓ Tempo de assentamento : $t_{s2\%} = 2,5$ s;
- ✓ Erro em regime : $e_{ss} = 0$;

6.3.3 Análise da Resposta Transitória

A análise da resposta transitória da planta não-controlada foi feita observando-se o comportamento do sistema descrito por (10) para uma entrada degrau através do Matlab.

6.3.4 Modelagem do Controlador

Será utilizado um controlador PI para controlar o sistema plataforma+motor.

Controladores PID

Os compensadores da classe dos PID possuem as seguintes relações:

- ✓ Compensador PD

$$U(s) = K(1 + T_d)E(s)$$

- ✓ Compensador PI

$$U(s) = K\left(1 + \frac{1}{T_i s}\right)E(s)$$

- ✓ Compensador PID

$$U(s) = K\left(1 + T_d s + \frac{1}{T_i s}\right)E(s)$$

onde :

U(s) =Entrada

E(s) = Erro

K = Ganho do Controlador

T_d = Tempo de Derivação

T_i = Tempo de Integração

Com base nos parâmetros de máximo sobressinal, tempo de assentamento, tempo de subida e erro em regime optou-se por adorar um controlador PI, cujos parâmetros são:

- ✓ Ganho proporcional : $K_c = 7,5$;
- ✓ Ganho integral : $K_I = 4$;

7 AQUISIÇÃO DOS EQUIPAMENTOS

A idéia inicial do trabalho era implementar o protótipo de simulador de vôo utilizando motores DC. No entanto, devido à questão de custos, foram contatadas algumas empresas com o objetivo de se obter empréstimos ou doações de equipamentos para a montagem do sistema de controle.

O resultado disso foi uma importante parceria montada junto à empresa Phoenix Contact, que disponibilizou para fins didáticos equipamentos de primeira linha. A Phoenix Contact forneceu os equipamentos para a montagem de um sistema de aquisição e envio de dados que simulasse o envio de comandos para os atuadores da plataforma.

Perante a oportunidade de se trabalhar com equipamentos de alta tecnologia e de se montar um protótipo utilizando-se destes componentes, o grupo decidiu alterar a meta do projeto em termos de construção do sistema eletro-eletrônico: ao invés de se montar um sistema de controle e acionamento dos motores DC, optou-se pela construção de um sistema de simulação dos envios de comando para os atuadores, composto pelos seguintes elementos: placa de aquisição e envio de dados e módulo de 4 saídas analógicas, este último representando simbolicamente as saídas de tensão para os motores e monitorados através de multimetros.

Em outras palavras, pode-se fazer o seguinte comparativo:

- No sistema inicialmente proposto, um dispositivo de aquisição e envio de dados (uma placa slotada ao barramento de um PC por exemplo) receberia os valores de deslocamento exigidos às colunas da plataforma e os converteria em um sinal analógico correspondente.

Este sinal seria enviado através de um módulo de I/O para um dispositivo auxiliar de chaveamento de potência, que por sua vez acionaria os motores;

- No novo sistema modelado, uma placa de aquisição e envio de dados recebe os valores de deslocamento correspondentes e os converte em um sinal analógico que é enviado pelo módulo de 4 saídas analógicas. Estes valores de tensão representam simbolicamente os níveis necessários para o acionamento dos atuadores, numa escala arbitrária entre o range de deslocamento da coluna da plataforma (positivo ou negativo) e o range de tensão fornecido pelo módulo de I/O.

A seguir apresenta-se uma descrição teórica da tecnologia utilizada nos equipamentos da Phoenix Contact, incluindo uma descrição do Protocolo de Comunicação Interbus, utilizado por estes equipamentos.

8 TECNOLOGIA DA REDE INTERBUS UTILIZADA NO PROJETO

A tecnologia de barramento serial está se defrontando com uma crescente aceitação em aplicações de todos os setores industriais tais como o automotivo, manipulação de materiais, logística, transportes, alimentício, etc. Em seguida serão apresentadas algumas características do Interbus como uma tecnologia de barramento sensor/atuador.

8.1 Características do Serial Networking

Tradicionalmente, sensores e atuadores são conectados a uma máquina por meio de cabeamento em paralelo, o que torna o sistema pouco flexível, mais caro e menos eficiente. A tecnologia de interligar os componentes do sistema através de uma rede serial construída em um nível de automação hierarquicamente mais baixo surge como uma alternativa de redução de custos.

Ao mesmo tempo, a intensificação da globalização do mercado internacional está originando novas demandas de sistemas o que impulsiona os engenheiros mecânicos a estarem considerando novas soluções em tecnologia de controle. Ao contrário do que se imagina, a tecnologia de automação mundial não é homogênea, isto é, os líderes de mercado determinam localmente a linha a ser adotada. Por outro lado, ser competitivo nos mercados internacionais implica possuir soluções de automação padronizadas e capazes de serem aplicadas no mundo sem maiores alterações. O resultado disso é a necessidade de arquiteturas de controles abertas que possam ser aceitas mundialmente ou, com pequenas adaptações, serem empregadas nos mercados locais.

Interbus é um sistema de barramento sensor/atuador que, devido a suas características técnicas, satisfaz aplicações no campo de sensores e atuadores industriais e dispõe de um sistema de rede consistente desde o nível de controle até a última chave de fim de curso. Além disso permite o desenvolvimento de soluções tecnológicas universais e versáteis.

8.2 Tipos de Dados Transmitidos

8.2.1 Dados de Processamento (Dados de I/O)

Caracterizam-se por possuírem o caráter de informação cíclica que precisa ser periodicamente atualizada pela rede. Ademais, para que as tarefas de controle automático sejam realizadas, é necessário que haja intervalos constantes e calculáveis de varredura. Em outras palavras, os dados de processamento relacionam-se com a demanda de transmissão periódica de dados determinísticos.

8.2.2 Parâmetros

São utilizados para monitorar e programar dispositivos inteligentes. Ao contrário dos dados de processamento, os parâmetros têm um caráter acíclico. Isto significa que a informação é transmitida somente mediante solicitação (chamada).

Ambos os tipos de dados (processamento e parâmetro) podem ser usados em qualquer dispositivo sensor/atuador. Para os mais simples, normalmente bastam os dados de

processamento; para os mais complexos, tais como acionamentos, requer ambos os tipos (inicialização e parametrização é realizada via parâmetros e valores de rotação ou frequência que são tipicamente determinados por dados de processamento.

8.3 Interbus, o Barramento Sensor/Atuador

Além de o perfil sensor/atuador Interbus ser aceito universalmente, ele também satisfaz as demandas do cliente por soluções 'user-friendly', flexíveis e duradouras.

Com relação à topologia da rede Interbus, esta é na forma de sistema em anel, ou seja, todos os dispositivos são interligados por um caminho cujo início coincide com o seu fim. A partir do anel principal, subanéis podem ser formados para estruturar todo o sistema. Estas conexões são levadas a usar componentes especiais chamados módulos terminais do barramento ('bus terminal modules'). Um subsistema pode ser de caráter local, denominado barramento local, que é usado para criar os cachos de I/O locais dentro de um gabinete de chaveamento. Por outro lado, o subsistema pode ter o papel de unir dispositivos remotos separados por grandes distâncias.

Uma característica exclusiva do sistema Interbus em comparação com os outros sistemas em anel é o fato de, ambas as linhas de transmissão e de recebimento de dados estão contidas em um mesmo cabo que liga cada dispositivo do sistema. Isto dá a nítida impressão de estarmos tratando de um estrutura em árvore ou linear.

O Interbus permite uma comunicação entre dispositivos através de uma distância superior a 13 km e o número de dispositivos de uma estrutura deve limitar-se a 512.

8.4 Topologia Interbus

A estrutura ponto-a-ponto do sistema Interbus e sua divisão em anel principal e subanéis são ideais para implementação de tecnologias de transmissão físicas diversas e, em particular, de tecnologia de transmissão com fibra ótica. O sistema de barramento é equipado de maneira tal que se torna fácil converter um sistema que utiliza fios de cobre para transmissão em um que se vale de tecnologia de fibra ótica ou de transmissão de dados por infra-vermelho.

A estrutura em anel do Interbus, provê duas vantagens principais: em primeiro lugar, permite que dados sejam enviados e recebidos simultaneamente ('full duplex'); em segundo, a função de auto-diagnóstico é consideravelmente melhorada com a estrutura em anel.

8.5 Detecção de problemas

No caso de sistemas de barramento (linha tronco) com conexão entre dispositivos do tipo multidrop, todos os dispositivos são quase passivamente ligados ao barramento. O caráter passivo dos dispositivos faz com que estes estejam livres apenas dos erros de operação e daqueles oriundos da desconexão dos mesmos. Se, por exemplo, a falha de um dispositivo causa um curto-circuito no barramento ou se a linha estiver curto-circuitada em um ponto fora do dispositivo, nenhum tipo de comunicação ocorre no sistema. Se um tipo de falha ocorre em um sistema de barramento, não é possível detectar a localidade em que ocorreu o erro com as funções de diagnóstico de que dispõem. Por outro lado, um sistema em anel conectado a dispositivos ativos, permite a segmentação de todo o complexo em subsistemas eletricamente

independentes. Por conta disso, em caso de falha em um dispositivo (curto-circuito ou desconexão do barramento), a comunicação só é interrompida localmente.

O fato de os subanéis serem configurados na rede Interbus permite conectar ou desconectar dispositivos sem provocar interferência. Os elementos acopladores entre os segmentos do barramento podem ser controlados pelo barramento principal ou mestre e permite que subsistemas sejam habilitados ou desabilitados. Portanto, é possível manipular localmente um subsistema sem afetar o resto do sistema.

8.5.1 Vantagens do sistema em anel para diagnóstico de problemas

Os dados não são alocados a determinados dispositivos através do endereçamento desses dispositivos no barramento como ocorre nos outros sistemas, mas através da posição física dos mesmos ao longo da estrutura em anel (os dispositivos são ativos). Isso contribui consideravelmente para facilitar a manutenção do sistema.

8.6 O Protocolo Interbus

O protocolo Interbus é estruturado em 3 níveis. O primeiro nível é o físico, onde são determinados as condições de tempo e o formato de codificação dos cabos. O segundo nível é denominado 'Data Link' que tem como função garantir a integridade dos dados; deve suportar os dois tipo de dados utilizados na tecnologia sensor/atuador: os dados de processamento cíclicos e os parâmetros não cíclicos. Uma característica importante do Data Link do Interbus é seu

caráter determinístico, ou seja, a garantia de tempo com a qual os dados são transportados de um dispositivo para outro que se relacionam remotamente.

Interbus baseia-se no procedimento de summation-frame que significa que para cada slot é dado um tempo de processamento associado à sua função, o que torna simples de se calcular o tempo de transmissão através da soma dos tempos de cada slot envolvido.

O protocolo e a topologia do sistema Interbus asseguram pela primeira vez uma integração inovadora da demanda por transporte de dados cíclicos e transporte acíclico de mensagens em um único sistema e, ao mesmo tempo, levam em consideração as demandas de todo o campo por por sensores/atuadores industriais.

O usuário do sistema Interbus é também de fundamental importância. Por conta disso, uma interface para aplicação Interbus foi desenvolvida dividida de modo a dividir-se de acordo com os dois tipos de dados. Uma rotina cíclica do barramento principal atualiza os dados e faz com que estes estejam disponível para o sistema de controle do usuário em forma de imagem de processos de I/O. Isto significa que os dados podem ser acessados com comandos e lógicas da sintaxe do PLC.

Quando do acesso aos dados, o usuário do sistema Interbus não irá notar diferença entre um sistema com cabeamento serial ou em paralelo. Os usuários não precisam estar familiarizados com as complexas estruturas de comunicação do sistema para que possam programar em protocolo Interbus.

Serviços do PMS (Peripherals Message Specification): permite, entre outras coisas, estabelecer e monitorar links de comunicação, escrita e leitura de parâmetros e iniciar programas

8.7 Integração com o Programa de Controle

Com os controladores lógicos programáveis, o acesso aos serviços de comunicação PMS a partir dos programas não é feito via I/O paralelo, mas através de blocos. Ademais, os serviços de comunicação podem ser predefinidos no planejamento do projeto do sistema e ativados como blocos de função quando do funcionamento do sistema. Esta interface isenta o usuário de estar manipulando rotinas complicadas de troca de dados e reduz a utilização dos serviços de comunicação para manipulação de operações lógicas.

O uso do PMS universal nas redes Interbus não só permite acesso direto aos dados, mas também provê a comunicação entre sensores/atuadores e os dispositivos inteligentes ou sistemas de controle. Além disso, permite que estruturas de programas utilizadas para um sistema de cabeamento paralelo sejam transferidas diretamente para redes seriais.

O Interbus pode ser incorporado ao conceito de rede universal, desde o nível de chão de fábrica, passando pelos níveis de processos e células até o de sensores/atuadores, assim servindo, por exemplo, como suplemento ideal para os padrões Ethernet inerentes aos níveis superiores de comunicação, muito comuns hoje em dia.

9 FUNDAMENTOS DE PROGRAMAÇÃO DO SISTEMA INTERBUS

O software do driver acessa o Multi-Port-Memory (MPM) da placa controladora através de uma área de 4 Kbytes mapeada na memória do PC, no range de 640 KB e 1 MB. Para essa região, as seguintes funções são disponibilizadas:

- funções para abertura e fechamento de canais;
- funções para comando de escrita e leitura de mensagens (interface de mailbox, opera com sinais de handshaking e controle de interrupções);
- funções para leitura e escrita de dados de I/O (interface de dados, opera sem mensagens de handshaking);
- funções de diagnóstico para controle do estado de operação da placa controladora.

9.1 Controle do Interbus

O aplicativo controla a placa controladora através de comandos tais como, 'inicie o sistema de barramento', 'leia na configuração do bus'.

Após inicializar a placa controladora e iniciar a transmissão de dados, a placa controladora opera o barramento de modo independente e retorna as mensagens correspondentes.

Se um erro grave afeta o sistema (por exemplo, 'cabo do barramento defeituoso'), todos os dispositivos conectados são automaticamente resetados e suas saídas são levadas a zero. Logo

em seguida a placa controladora investiga o erro e descreve a sua causa e localização em detalhes.

9.2 Interfaces entre hardware e software

Esta seção descreve as interfaces ente a placa controladora e o software de seu driver.

9.2.1 Multi-Port_Memory (MPM)

O MPM é a interface central das placas controladoras. O MPM é uma memória localizada na placa controladora que pode ser acessada por qualquer MPM accessors (PC e Interbus Master). Os MPM accessors armazenam todos os dados para uso compartilhado no MPM. O MPM é a única conexão entre os MPM accessors.

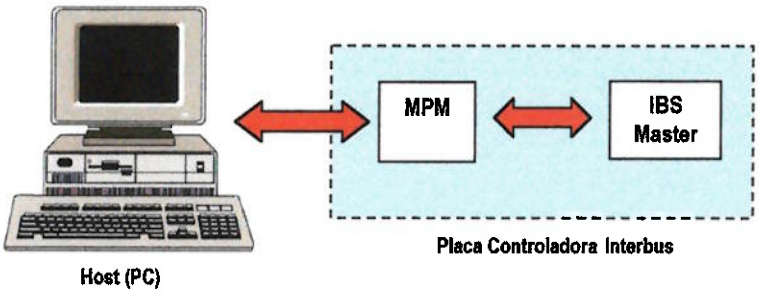


Figura 9.1: Arquitetura PC / Multi-Port-Memory

9.2.2 Estrutura Fundamental do Software do Driver

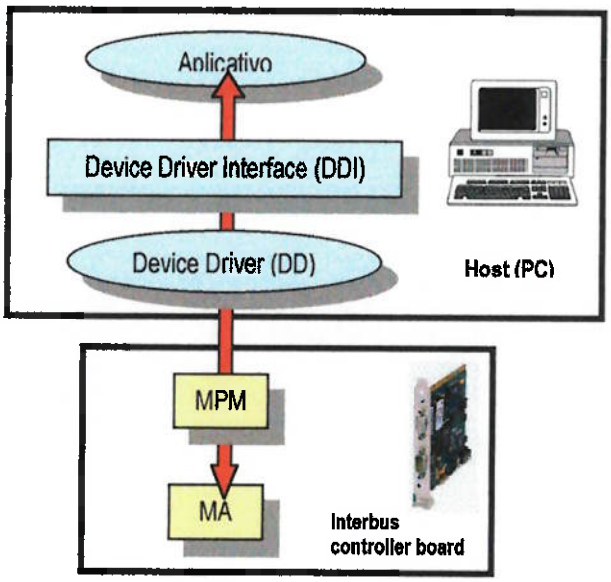


Figura 9.2: Estrutura Fundamental do Software do Driver

O driver consiste de duas partes:

1) O Device Driver Interface (DDI) é uma interface para compilação do aplicativo;
2) O Device Driver (DD) que é específico com relação ao sistema operacional utilizado. O Device Driver conecta o Host (PC) ao Interbus Master (MA), via a MPM.

Até 8 placas controladoras podem ser instaladas em um host. Para cada placa deve-se ter um Device Driver. A Device Driver Interface é responsável por gerenciar e controlar todos os Device Drivers.

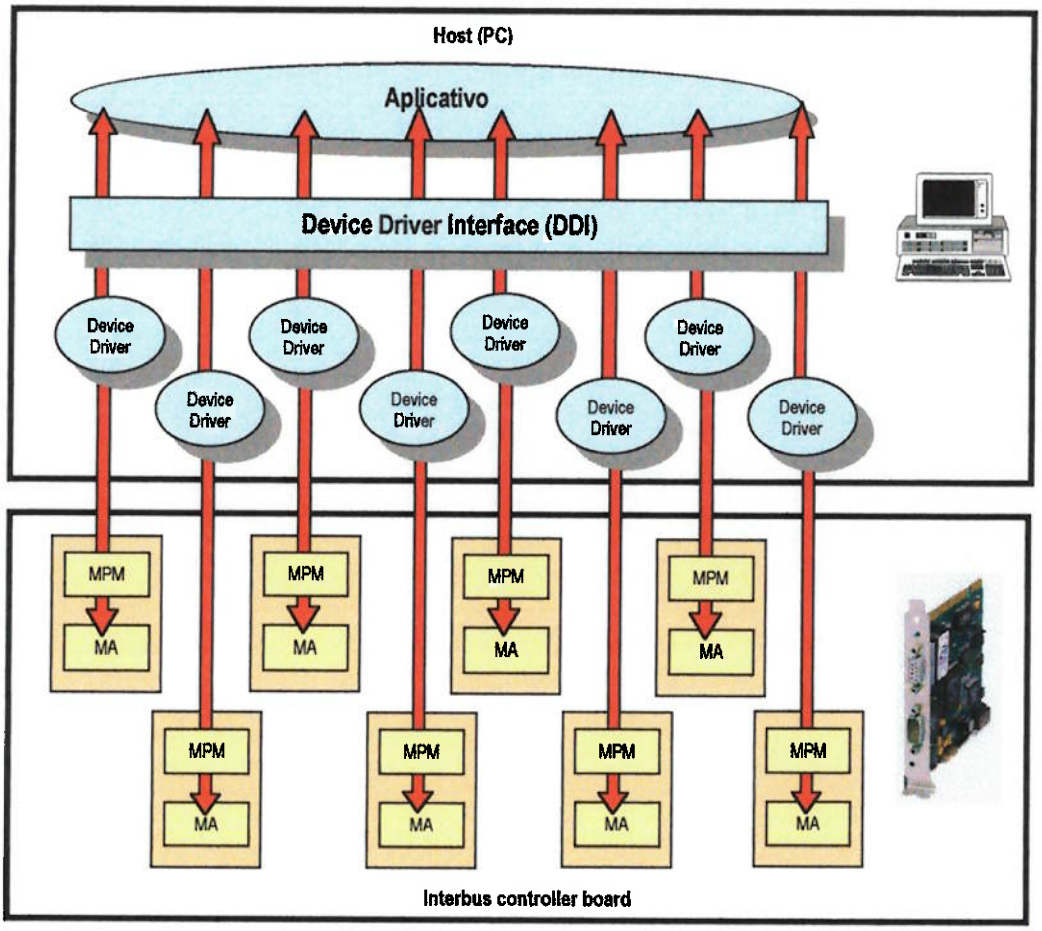


Figura 9.3: Estrutura do Software do Driver com múltiplas placas controladoras

9.2.3 Implementação da DDI e do DD

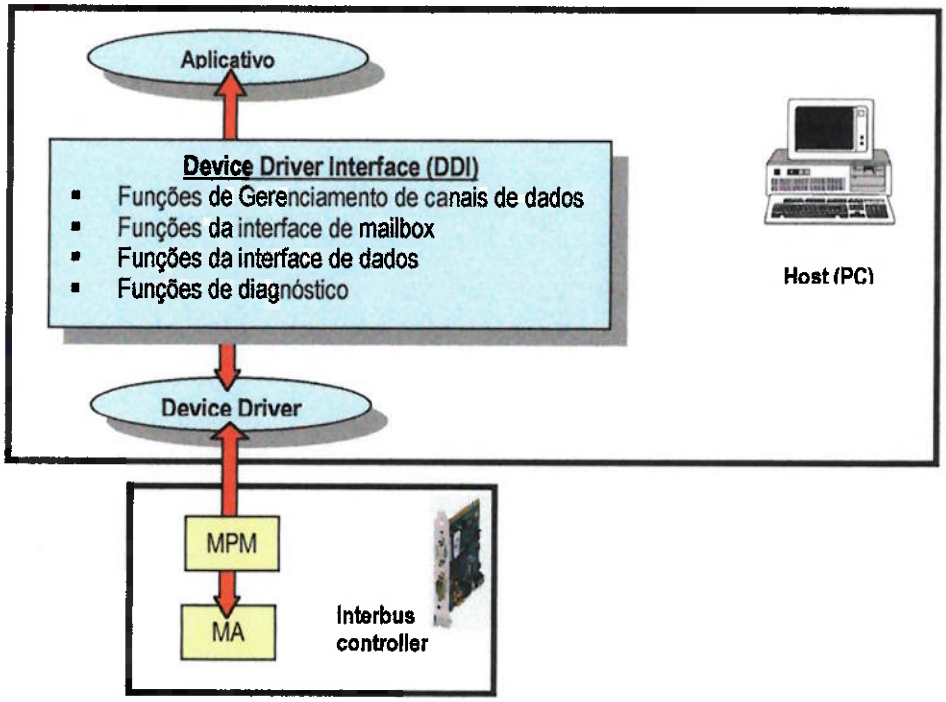


Figura 9.4: Funções da Device Driver Interface

9.3 Monitoração por Watchdogs

- ***Watchdog da placa mestre do Interbus***: se o watchdog da placa mestre torna-se ativo, o sistema Interbus e todas as suas saídas são resetados.
- ***Watchdog para monitoração do Host***: o circuito watchdog para monitoração do PC localiza-se na placa mãe da placa controladora. Se o watchdog torna-se ativo, o sistema Interbus é estabelecido em uma determinada condição (todas as saídas são resetadas).

9.4 Software do Driver para Windows 95

9.4.1 Estrutura do software no Host

O software do driver para MS Windows 95 é desenhado como uma *Dynamic Link Library* (DLL) com um Device Driver virtual. Esta DLL chama o device driver virtual em tempo de execução, ocasionando sua gravação em memória.

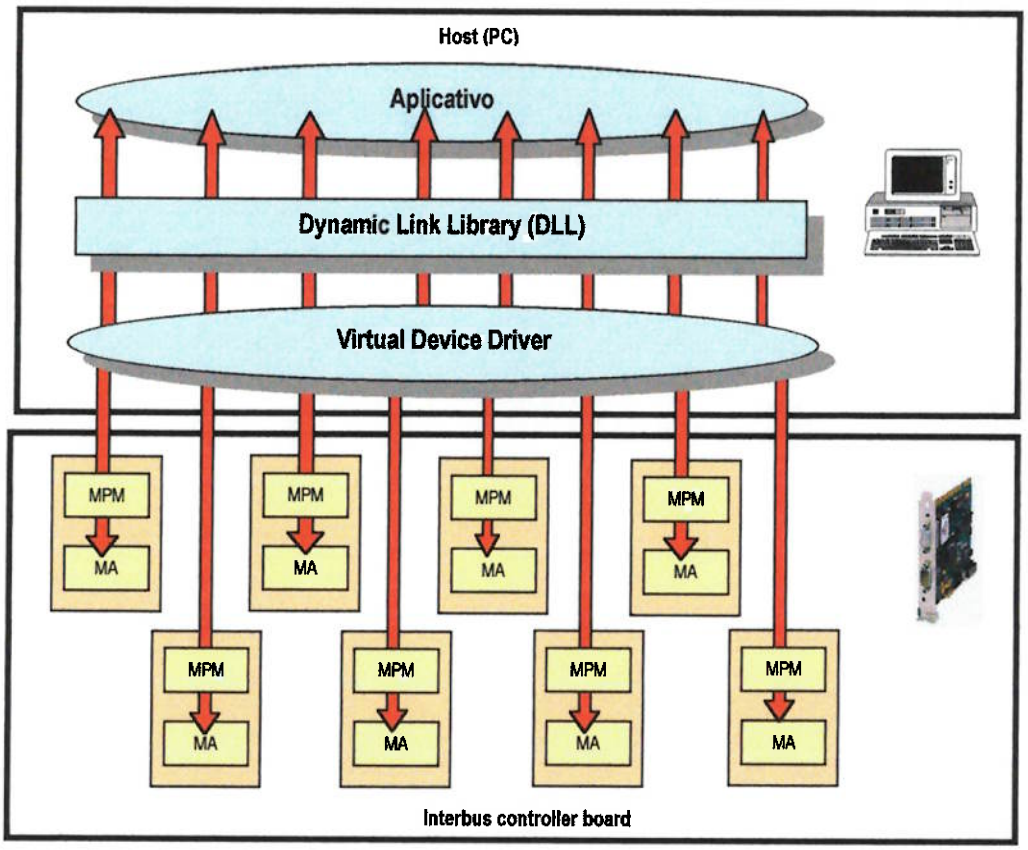


Figura 9.5: Estrutura do Software do Driver para Windows 95/NT

10 ESTRUTURA E FUNCIONALIDADES DA INTERFACE HLI

HLI é uma interface com usuário inteligente responsável por proporcionar as funcionalidades das funções do Interbus. A HLI converte chamadas de funções do aplicativo em procedimentos complexos de comunicação com a placa controladora Interbus. É uma aplicação passiva, ou seja, apenas será ativada quando associada à chamada de uma função.

Se o aplicativo roda em um PC, até 4 placas controladoras podem ser operadas ao mesmo tempo. Ademais, o HLI suporta modo de operação controlado por aplicativo. Este modo permite que dados sejam transmitidos em ciclos, sincronizados com o aplicativo.

Dentre as funcionalidades contempladas pela HLI têm-se:

- Inicialização da placa controladora e startup do Interbus;
- Gerenciamento dos dados de processo;
- Gerenciamento de erros e eventos;
- Controle do Interbus;
- Gerenciamento de serviços PCP (*peripherals communication protocols*);
- Serviços de informação.

10.1 Inicialização da Placa Controladora

Existem dois modos de inicialização da placa: padrão e personalizada. A inicialização padrão utiliza configurações padrão do barramento. Na personalizada, o aplicativo transmite a

configuração feita pelo usuário dos dispositivos do barramento. A placa controladora compara o arranjo físico do sistema com a configuração transmitida. Se estiverem idênticas, será feita a inicialização.

10.2 Gerenciamento de Dados de Processamento

A interface HLI permite mapear dados de processamento do Interbus em variáveis declaradas no programa do aplicativo (*process data objects, PDO*). Nesse sentido, a programação das funções de controle é bastante simples e fácil.

10.3 Gerenciamento de Eventos

Atividades do barramento, mensagens e erros são eventos que a HLI codifica em registro de dados de evento e armazena em uma fila FIFO interna. Os eventos são classificados da seguinte maneira:

- Erros internos (por exemplo, falta de memória interna);
- Erros na DDI;
- Erros na placa controladora;
- Erros que provocam parada do barramento;
- Mensagens de dispositivos;
- Mudanças no estado do Interbus;
- Erros de comunicação PCP;
- Erros do aplicativo;

O aplicativo pode requerer os registros desses dados em ordem cronológica e ler a descrição de evento do registro de dado associado em forma texto.

Com base em elemento ou variável que contém o status da aplicação, a HLI informa da ocorrência de eventos.

10.4 Controle do Interbus

Para controlar o sistema Interbus, os seguintes serviços estão disponíveis:

- Início, parada e alarme da transferência de dados pelo barramento;
- Comutação entre os segmentos de barramentos remoto e local;
- Comutação entre grupo de dispositivos e grupos alternativos.

10.5 Gerenciamento e Serviços do PCP

A HLI permite utilizar os serviços para dispositivos PCP através da indicação do número lógico do dispositivo. A HLI controla e monitora toda relação de comunicação com o dispositivo e a execução do serviço.

Existem duas maneiras de se executar um serviço PCP:

- Como um 'pedido': A HLI executa o pedido de serviço e retorna para o aplicativo. Enquanto a placa controladora estiver executando o serviço, o aplicativo pode processar outras tarefas.
- Como um 'serviço': A HLI executa o pedido de serviço e aguarda até que este seja concluído. Nesse entremeio, outras tarefas não podem ser processadas.

Os seguintes serviços são contemplados:

- Identificação de serviço;
- Serviço de leitura;
- Serviço de escrita;
- Serviço de escrita de um dispositivo;
- Serviço de início de programa;
- Serviço de parada de programa;
- Serviço de finalização de programa;
- Serviço de reset de programa.

10.6 Serviços de Informação

A HLI permite o pedido das seguintes informações sobre o sistema Interbus:

- sobre a placa controladora (versão);
- diagnóstico/estatísticas;
- configuração do barramento (número de dispositivos, bits de I/O, dispositivos PCP, tempo do ciclo, etc.);
- informações sobre os dispositivos;
- informações sobre os dispositivos PCP;

10.7 Máquina de Estado da Interface HLI

O gerenciamento interno da HLI é produzido pela máquina de estado interna. O processamento dessa máquina de estado é ligada com a chamada de determinadas funções HLI.

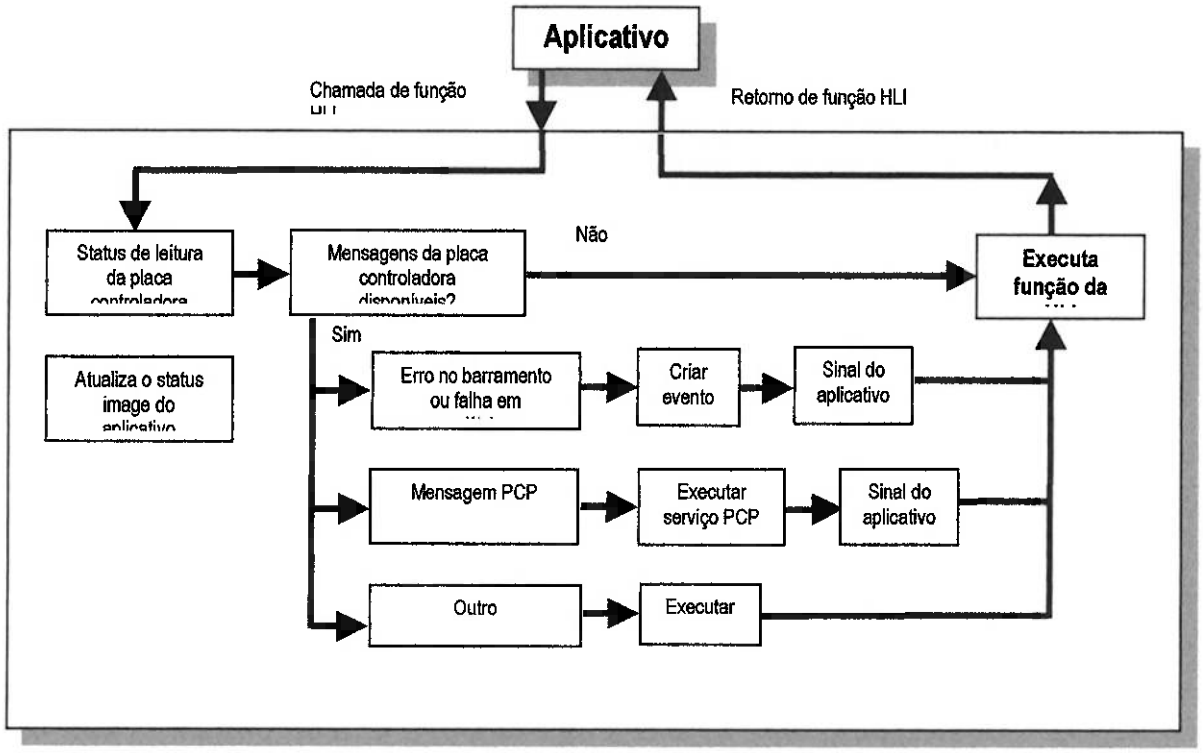


Figura 9.6: Máquina de Estado da Interface HLI

11 ESTRUTURA DA REDE INTERBUS UTILIZADA NO PROJETO

11.1 Descrição do hardware de aquisição de dados

Conforme descrito anteriormente, o sistema desenvolvido consiste no conjunto placa de aquisição e envio de dados e de um módulo de 4 saídas analógicas que simboliza os níveis de tensão enviados para os motores. A arquitetura do hardware da Phoenix para aplicações como o deste projeto pode ser definida conforme a Figura 11.1:

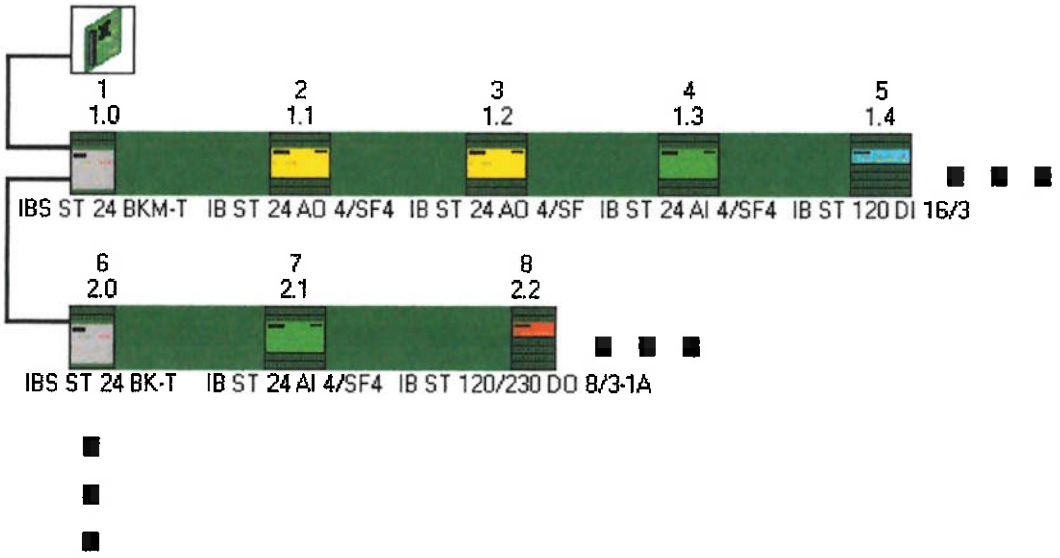


Figura 11.1 – Estrutura Genérica da Rede Interbus Utilizada no Projeto

Conforme mostra a Figura 11.1, uma rede Interbus é composta por três elementos básicos:

- Placa Controladora de aquisição e envio de dados
- Módulos BK de interface entre a placa controladora e os módulos de I/O e de execução de controle independente sobre os módulos de I/O nele conectados

- Módulos de I/O

Nota-se que a disposição dos módulos de BK e I/O tem a forma de matriz, onde:

- Para cada linha da matriz, o elemento da primeira coluna é um módulo BK, que faz o interfaceamento e controle independente sobre os módulos de I/O que se seguem imediatamente à sua direita. Cada módulo de BK suporta uma certa quantidade limitada de módulos de I/O, conforme o tipo.
- Na necessidade de introdução de mais módulos de I/O na rede do que um módulo BK suporta, pode-se adicionar um novo módulo BK na primeira coluna da linha posterior à ocupada pelo módulo de BK "saturado".
- Portanto a primeira coluna desta matriz é sempre formada por módulos BK, que se comunicam com seus respectivos módulos de I/O e com a placa controladora. Uma placa controladora pode suportar um certo limite de módulos BK.

Partindo desta definição, a arquitetura do hardware da Phoenix foi montada conforme o esquema a seguir:

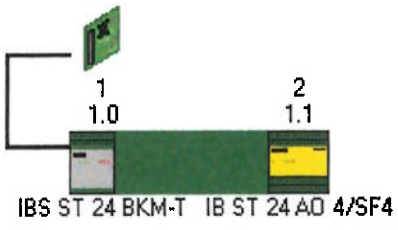


Figura 11.2: Estrutura da Rede Intebus utilizada no projeto.

Os seguintes componentes foram utilizados para a rede do projeto:

11.1.1 Placa controladora IBS PC ISA SC/I-T

A placa controladora IBS PC ISA SC/I-T é utilizada em aplicações onde se deseja programar o software de controle em linguagens de alto nível (Borland C, Borland Delphi, MS Visual Basic, etc.). Esta placa de controle pode ser slotada no barramento da placa-mãe de um PC, utilizando-se uma de suas portas seriais para a comunicação com os módulos de BK, via uma rede serial RS-232 ou RS-485 para longas distâncias, por exemplo. Maiores informações sobre este equipamento estão contidas em seu Data-Sheet, presente no Apêndice III.

11.1.2 Módulo IBS ST 24 BKM-T

O módulo IBS ST 24 BKM-T foi selecionado já que para esta aplicação estaremos utilizando módulos de I/O da família ST. Maiores informações estão contidas no seu Data-Sheet, presente no Apêndice IV.

11.1.3 Módulo IB ST 24 AO 4/SF4

O módulo IB ST 24 AO 4/SF4 fornece quatro saídas analógicas em tensão 0 a 10 V ou em corrente 4 a 20 mA. Para a aplicação, utiliza-se as saídas em tensão, que vão exatamente representar simbolicamente os níveis de tensão que seriam enviados para o motor. Portanto, os valores de deslocamento das colunas da plataforma terão seus intervalos (desde o valor negativo de deslocamento máximo no caso de descida da coluna ao valor máximo positivo em caso de subida) de deslocamento proporcionais ao range de 0-10V das saídas em tensão do módulo de I/O.

Maiores informações sobre este módulo podem ser encontradas em seu Data-Sheet, presente no Apêndice V.

11.2 Descrição do software de aquisição de dados e da Interface visual

11.2.1 Software de aquisição de dados

Conforme citado anteriormente, a placa IBS PC ISA SC-I-T permite programação em linguagens de alto nível. Neste caso, foi escolhido o MS Visual Basic como ferramenta de programação, conforme explicação contida no item 5.5.1.

Seguindo a topologia da estrutura de rede explanada no item 11.1, os seguintes passos foram realizados para a programação do software de aquisição de dados:

- Identificação do endereçamento da placa controladora no PC: através do driver específico obtido no site da Phoenix Contact, configurou-se o seu endereço de I/O, o seu endereço de comunicação e o seu IRQ (no caso utilizou-se o IRQ #7 que estava disponível no PC). Vale lembrar que este endereçamento foi realizado baseando-se na posição dos switches determinada na placa controladora. Além disso, o driver instalado foi o referente ao sistema operacional utilizado pelo host (no caso o PC), que é o Windows NT. Assim, o driver DD correspondente ao sistema operacional adotado também foi instalado, assim como o driver DDI necessário para a programação desta placa.
- Após este passo, foi utilizado o software de programação da rede IBS CMD G4 Versão 4.41, com o qual foi possível se configurar a placa controladora e os módulos periféricos utilizados na rede de comunicação do projeto.

- Configurada a rede, pode-se exportá-la para um módulo de programação (*.bas) do MS Visual Basic, através da interface HLI, citada e detalhada no item 7 deste trabalho.
- Outro módulo *.bas foi introduzido no projeto do software no MS Visual Basic, contendo as bibliotecas disponíveis para a programação da placa controladora através desta ferramenta de programação, via interface HLI.

O código da aplicação desenvolvida no MS Visual Basic encontra-se nos Apêndices I e II, juntamente com o código da aplicação de coleta de parâmetros da plataforma e geração dos respectivos deslocamentos necessários, detalhada no item 4.3.3.

11.2.2 Interface Visual

A interface visual desenvolvida visa exatamente disponibilizar ao usuário as funções de iniciar/finalizar a comunicação com o módulo de I/O (ativando/desativando o barramento correspondente), além de processar os dados gerados pelo host. A Figura 11.3 mostra a Interface desenvolvida.

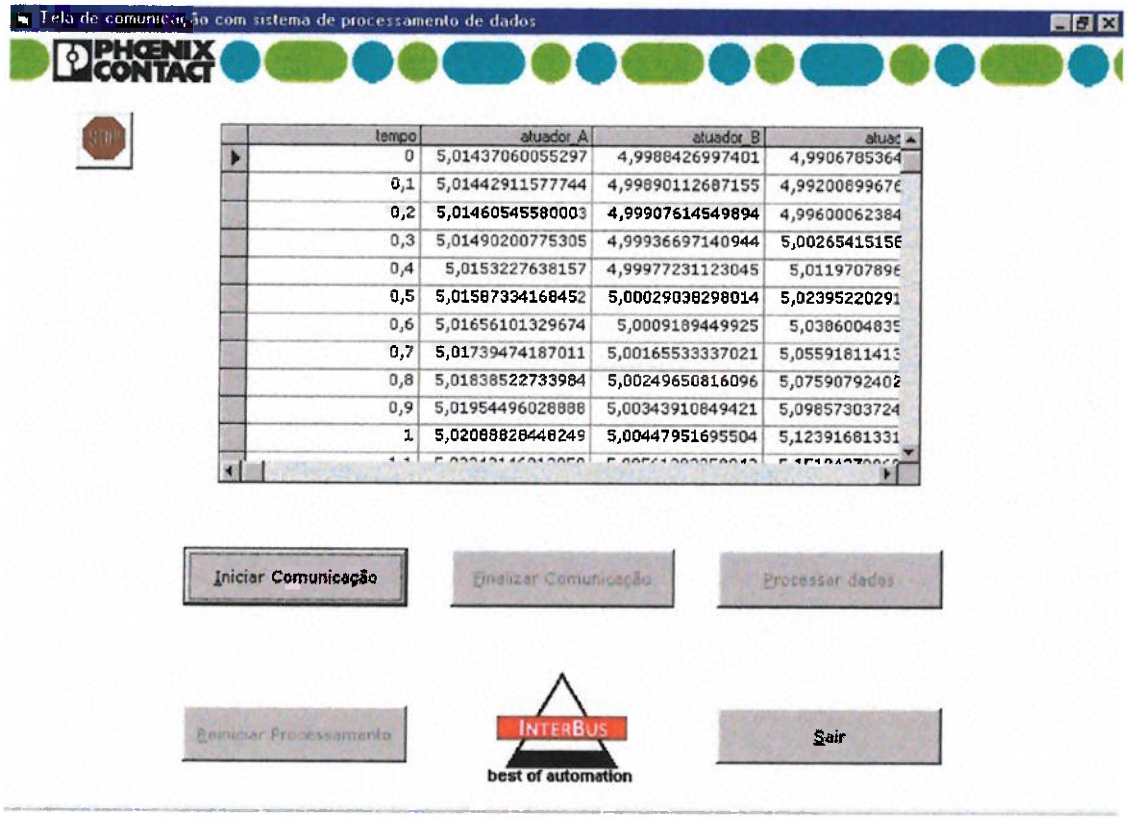


Figura 11.3: Interface desenvolvida no MS Visual Basic.

Esta tela de interface também oferece a opção de se reinicializar o processamento dos dados de tensão gerados pelo host (clicando-se a tecla reiniciar processamento, o programa volta a apontar para a primeira posição da tabela de tensões), além de uma tecla de emergência (botão stop) que, se acionada, interrompe imediatamente o processo, parando o processamento de dados, desativando a comunicação com os módulos e zerando as tensões de saída do módulo de I/O.

12 DISCUSSÃO E CONCLUSÃO

Com este projeto pôde-se implementar os conceitos de cinemática inversa bastante utilizados em robôs industriais, onde o sucesso do experimento foi verificado por meio da construção do protótipo e dos teste nele realizados.

O único problema verificado foi o surgimento de um aparente grau de liberdade a mais (rotação em torno do eixo z), porém foi verificado que este não foi proveniente de erros de cálculo mas do fato de as articulações simples e as colunas não serem totalmente rígidas. Isto faz com que ocorra uma translação das juntas esféricas localizadas na plataforma gerando um grau de liberdade a mais. No entanto, este tipo de problema é muito difícil de ser verificado teoricamente. Caberá a quem for continuar este projeto verificar experimentalmente a melhor disposição das juntas que minimize esta falha.

Com relação ao volume de trabalho, foi observado que apesar de não ser uma das vantagens da Plataforma de Stewart, este é mais do que suficiente quando o princípio é aplicado a máquinas-ferramenta e simuladores de vôo. Porém, por se tratar de um mecanismo de cinemática fechada, o que torna complicado o estudo analítico do volume de trabalho, ainda é necessário desenvolver uma análise mais aprofundada a esse respeito de forma a verificar os limites e todas as posições singulares desta plataforma.

Como sugestão para a continuação e melhoria deste projeto é sugerido alguns trabalhos:

- desenvolvimento da interface entre a placa controladora e os motores elétricos;
- melhoria na precisão de posicionamento dos fusos através da aquisição de fusos de esferas recirculantes e da melhor fabricação das juntas e colunas;
- desenvolvimento de um programa que torne possível controlar a plataforma por meio de um *joystick*.

13 REFERÊNCIAS BIBLIOGRÁFICAS

P.A. Drexel, *"A Six Degree-of-Freedom, Hydraulic, One Person Motion Simulator"*, Tese de Mestrado, University of British Columbia, 1992.

Zorah Lazarevic, *"Feasibility of a Stewart Platform with Fixed Actuators as a Platform for CABG Surgery Device"*, Tese de Mestrado, Columbia University – Department of Bioengineering.

E. L. L. Cabral, *"Notas de Aula de PMC 591 – Elementos de Robótica"*, Escola Politécnica da Universidade de São Paulo – Departamento de Engenharia Mecânica.

A.M. Oura & R. S. Yamamoto, *"Projeto de uma Plataforma Móvel para Uso em Simuladores"*, Trabalho de Formatura, Universidade de São Paulo, 1998.

THK LM System – Ball Screws, *Catálogo No. 200-1 BE*, Tokyo, Japão

13.1 Referências Internet

- http://www.cs.columbia.edu/~laza/stewart/
- http://www.ee.ubc.ca/staff/faculty/tims/etc/www/plat_com.html
- http://www.me.mtu.edu/~asbanka/stewart.html
- http://www.mech.nwu.edu/MFG/AML/SPAEA/Model.html
- http://wwwrobot.gmc.ulaval.ca/ANG/rob_parall_an.html
- http://ssb~www.larc.nasa.gov/facility/ga.html
- www.phoenixcontact.com
- www.interbus.com

APÊNDICES

APÊNDICE I - CÓDIGO DOS FORMULÁRIOS DO SOFTWARE

```
'*****************************************************************************************
'*
'*      ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
'*      DEPARTAMENTO DE ENGENHARIA MECÂNICA
'*
'*****************************************************************************************
'*
'*      PROJETO DE PLATAFORMA MÓVEL COM 4 GRAUS DE LIBERDADE
'*      BASEADA EM PLATAORMA DE STEWART
'*
'*      Autores:
'*          Danielle Morita
'*          Fábio Fukunaga
'*          Maurício Dias Menezes Mazza
'*          Rafael Salla Paulilo
'*
'*      Professor Orientador:
'*          Tarcísio A. Hess Coelho
'*
'*****************************************************************************************
'*
'*      FORMULÁRIO DE APRESENTAÇÃO DA TABELA DE DADOS DE POSIÇÃO
'*
'*      Formulário: Deslocamento.frm
'*
'*****************************************************************************************

'*****************************************************************************************
'*
'*      Função Bt_Rodar_Click : Inicia Formulário de Comunicação
'*
'*****************************************************************************************

Private Sub Bt_Rodar_Click()

    Rodar.Show

End Sub

'*****************************************************************************************
'*
'*      Função Bt_Save_Click   : Salva o Banco de Dados
'*
'*****************************************************************************************

Private Sub Bt_Save_Click()

    CommonDialog1.Filter = "Banco de Dados (*.mdb) |*.mdb| All Files (*.*)|*.*|"
    CommonDialog1.ShowSave
    If CommonDialog1.Filename <> "" Then Intervalo.movimento.Name =
CommonDialog1.Filename

End Sub

'*****************************************************************************************
'*
'*      Função Bt_Sim_Click   : Inicia Formulário de Simulação Gráfica
'*
'*****************************************************************************************

Private Sub Bt_Sim_Click()

    Simula.Show

End Sub
```

```
'*************************************************************************************
'*
'*       Função Form_Load    : Carrega o Formulário Deslocamentos
'*                             Define a origem da tabela de dados
'*
'*************************************************************************************

Private Sub Form_Load()

    If Principal.Tipo_db = 0 Then

        Data1.DatabaseName = Intervalo.movimento.Name
        Data1.RecordSource = "deslocamento"

        Label1 = Intervalo.Ha
        Label2 = Intervalo.Hb
        Label3 = Intervalo.Hc
        Label4 = Intervalo.Hd

    Else
        Data1.DatabaseName = Principal.banco_de_dados
        Data1.RecordSource = "deslocamento"
    End If

End Sub

'*************************************************************************************
'*
'*       Função Form_Unload : Apaga tabela temporária, para Tipo_db = 0
'*
'*************************************************************************************

Private Sub Form_Unload(Cancel As Integer)

    If Principal.Tipo_db = 0 Then

        Intervalo.movimento.Close
        Intervalo.teste.Close

    End If

End Sub
```

```
'*****************************************************************************************
*
'*
'*       ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
'*       DEPARTAMENTO DE ENGENHARIA MECÂNICA
'*
'*****************************************************************************************
*
'*
'*       PROJETO DE PLATAFORMA MÓVEL COM 4 GRAUS DE LIBERDADE
'*       BASEADA EM PLATAORMA DE STEWART
'*
'*       Autores:
'*           Danielle Morita
'*           Fábio Fukunaga
'*           Maurício Dias Menezes Mazza
'*           Rafael Salla Paulilo
'*
'*       Professor Orientador:
'*           Tarcísio A. Hess Coelho
'*
'*****************************************************************************************
'*
'*       FORMULÁRIO DE ENTRADA DE DADOS DO MOVIMENTO
'*
'*       Formulário: Ent_Dados.frm
'*
'*****************************************************************************************

'Distância linear total raiz(x^2+Z^2)
Dim d_xz As Double

'Ângulo formado pelas posições final e inicial do CG
Public sin_gama As Double
Public cos_gama As Double

'Tempos dos movimentos linear/angular
Public t_xz As Double
Public t_tetax As Double
Public t_tetay As Double
Public t_max As Double

'Velocidade real dos movimentos linear/angular
Public v_lin As Double
Public v_tetax As Double
Public v_tetay As Double

'*****************************************************************************************
'*
'*       Função Bt_Principal_Click : Inicia Formulário Principal
'*                                   Descarrega Formulário Atual
'*
'*****************************************************************************************

Private Sub Bt_Principal_Click()

    Unload Me
    Principal.Show

End Sub

'*****************************************************************************************
'*
'*       Função Bt_Próximo_Click : Inicia Formulário de Tempo de Amostragem/Discretização
'*                                 Efetua os cálculos dos tempos e velocidades reais de
'*                                 cada movimento
'*
'*****************************************************************************************

Private Sub Bt_Proximo_Click()

'Considerndo as coordenadas iniciais do Baricentro Xg = Zg = 0

    d_xz = Sqr(CDbl(TB_X) ^ 2 + CDbl(TB_Z) ^ 2)
```

```
    sin_gama = (CDbl(TB_Z) - 0) / d_xz
    cos_gama = (CDbl(TB_X) - 0) / d_xz

    t_xz = Calcula_Tempo(d_xz, CDbl(TB_VLMax), CDbl(TB_ALMax))
    t_tetax = Calcula_Tempo(CDbl(TB_TetaX), CDbl(TB_VAMax), CDbl(TB_AAMax))
    t_tetay = Calcula_Tempo(CDbl(TB_TetaY), CDbl(TB_VAMax), CDbl(TB_AAMax))

    t_max = Calcula_tmax(t_xz, t_tetax, t_tetay)

    v_lin = Calcula_v(d_xz, CDbl(TB_VLMax), CDbl(TB_ALMax), t_xz)
    v_tetax = Calcula_v(CDbl(TB_TetaX), CDbl(TB_VAMax), CDbl(TB_AAMax), t_tetax)
    v_tetay = Calcula_v(CDbl(TB_TetaY), CDbl(TB_VAMax), CDbl(TB_AAMax), t_tetay)

    Intervalo.Show

End Sub


'*****************************************************************************
'**  IBS G4 HLI       : INTERBUS Controller Initialization Module            **
'**  Application type : Microsoft Visual Basic                               **
'**  Controller-ID    : ISASC1                                               **
'*****************************************************************************
'**  Produced by:                                                           **
'**  HLI-Export-Filter Version 1.04 (02/99)                                 **
'**  (c) Phoenix Contact 01/1998                                            **
'*****************************************************************************


' This array will be set to the bus configuration data
Private IBS_DeviceList(0 To 1) As T_HLI_DEVICE_DATA


' Instance of Controller State Object
Public SC1 As T_HLI_STATE

' private variable to substitute Null reference
Private hliNull As Integer

' Global Process Data Objects
Public AO1                                  As Integer
Public AO2                                  As Integer
Public AO3                                  As Integer
Public AO4                                  As Integer


' function to initialize the device configuration array
Function SC1_SetConfigList(I As Integer, Seg As Integer, Pos As Integer, ID As Integer,
Lg As Integer, Lev As Integer, G As Integer, alt As Integer, deac As Integer, st As
String) As Integer
Dim z As Integer
IBS_DeviceList(I).Segment = Seg
IBS_DeviceList(I).Position = Pos
IBS_DeviceList(I).ID_Code = ID
IBS_DeviceList(I).LengthCode = Lg
IBS_DeviceList(I).Level = Lev
IBS_DeviceList(I).Group = G
IBS_DeviceList(I).Group = G
IBS_DeviceList(I).Alternative = alt
IBS_DeviceList(I).Deactivated = deac
For z = 1 To Len(st)
 IBS_DeviceList(I).Station(z - 1) = Asc(Mid(st, z, 1))
Next
IBS_DeviceList(I).Station(z) = 0
End Function


'#############################################################################
'#                                                                           #
'#      Call this function for startup of the INTERBUS system                #
'#                                                                           #
'#############################################################################

Public Function IBS_HLI_Init_SC1() As Integer
Dim ret As Integer
```

```
 ' initialize configuration array
 Call SC1_SetConfigList(0, 1, 0, 8, 0, 0, 255, 255, &H0, "Hall 2 Cabinet 1")
 Call SC1_SetConfigList(1, 1, 1, 125, 4, 1, 255, 255, &H0, "Hall 2 Cabinet 1")

 ' Create the event-logging window
 Call IBS_HLI_CreateLogWindow(ISASC1, HLI_ENGLISH, True)


 ' call init function
 ret = IBS_HLI_Init_CFG(ISASC1, SC1, IBS_STANDARD, 2, IBS_DeviceList(0))
 If ret = HLI_OKAY Then
  ' call PDO registration functions
  Call IBS_HLI_RegisterPDObject(ISASC1, IBS_PDO_OUTPUT, 1, 1, IBS_PDO_WORD, 0, 0, 16,
AO1, hliNull)
  Call IBS_HLI_RegisterPDObject(ISASC1, IBS_PDO_OUTPUT, 1, 1, IBS_PDO_WORD, 2, 0, 16,
AO2, hliNull)
  Call IBS_HLI_RegisterPDObject(ISASC1, IBS_PDO_OUTPUT, 1, 1, IBS_PDO_WORD, 4, 0, 16,
AO3, hliNull)
  Call IBS_HLI_RegisterPDObject(ISASC1, IBS_PDO_OUTPUT, 1, 1, IBS_PDO_WORD, 6, 0, 16,
AO4, hliNull)
  ' Reset all outputs
  Call IBS_HLI_ResetAllOutputs(ISASC1)
  ' Start bus now
  ret = IBS_HLI_StartBus(ISASC1)
 End If
 IBS_HLI_Init_SC1 = ret

End Function

'#########################################################################
'#                                                                       #
'#                  Shut-down of the INTERBUS system -                   #
'#     Call this function before your application terminates !           #
'#                                                                       #
'#########################################################################

Public Function IBS_HLI_Exit_SC1() As Integer
 IBS_HLI_Exit_SC1 = IBS_HLI_Exit(ISASC1)
End Function

'#########################################################################
'#                                                                       #
'#             Cyclic processing of the INTERBUS system -                #
'#     This is a PROPOSED FUNCTION for cyclic processing of the          #
'#     HLI state machine and the INTERBUS process data objects           #
'#                                                                       #
'#########################################################################

Public Sub IBS_HLI_Process_SC1()
 ' HLI State machine
 Call IBS_HLI_Process(ISASC1)
 ' Look at the Controller-State-object for INTERBUS-State ...

 If SC1.BusState = HLI_IBS_RUN Then

  ' Update of Interbus-S In-data
  Call IBS_HLI_PD_In(ISASC1)

  ' Here you can place your application-specific code
  ' for process data control */

  ' Update of Interbus-S Out-data
  Call IBS_HLI_PD_Out(ISASC1)
 End If
End Sub


' ------------------------------ END OF FILE ------------------------------
```

```
'*****************************************************************************************
'*
'*
'*      ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
'*      DEPARTAMENTO DE ENGENHARIA MECÂNICA
'*
'*****************************************************************************************
'*
'*      PROJETO DE PLATAFORMA MÓVEL COM 4 GRAUS DE LIBERDADE
'*      BASEADA EM PLATAORMA DE STEWART
'*
'*      Autores:
'*          Danielle Morita
'*          Fábio Fukunaga
'*          Maurício Dias Menezes Mazza
'*          Rafael Salla Paulilo
'*
'*      Professor Orientador:
'*          Tarcísio A. Hess Coelho
'*
'*****************************************************************************************
'*
'*      FORMULÁRIO DE APRESENTAÇÃO DAS CORREÇÕES NO MOVIMENTO
'*      E DETERMINAÇÃO DO INTERVALO DE AMOSTRAGEM/DISCRETIZAÇÃO
'*
'*      Formulário: Intervalo.frm
'*
'*****************************************************************************************

'Variáveis para os valores instantâneos dos movimentos linear/angular
Dim d As Double
Dim tetax As Double
Dim tetay As Double

'Variáveis de tempo: Tempo de Amostragem e Tempo Atual
Public amostragem As Integer
Dim t_atual As Double

'Características dos movimentos linear/angular CORRIGIDOS
Dim vc_lin As Double
Dim vc_tetax As Double
Dim vc_tetay As Double
Dim ac_lin As Double
Dim ac_tetax As Double
Dim ac_tetay As Double

'Variáveis para as posições instantânea de cada atuador
Public Ha As Double
Public Hb As Double
Public Hc As Double
Public Hd As Double

'Variáveis do Banco de Dados de Deslocamentos
Public teste As Workspace
Public movimento As Database
Dim tab_deslocamento As TableDef
Dim tab_tensao As TableDef
Public rec_deslocamento As Recordset
Public rec_tensao As Recordset

'*****************************************************************************************
'*
'*      Função Bt_Rodar_Click : Inicia Formulário de Comunicação
'*
'*****************************************************************************************

Private Sub Bt_Gera_Click()

'Coordenadas do Baricentro
Dim Xg As Double
Dim Zg As Double

'Variável auxiliar para o delta dos atuadores
Dim h_temp As Double
```

```
'Coordenadas dos pontos de fixação das barras na placa
Dim Xa As Double
Dim Ya As Double
Dim Za As Double
Dim Xc As Double
Dim Yc As Double
Dim Zc As Double
Dim Xd As Double
Dim Yd As Double
Dim Zd As Double

'Definição das Tabelas/Espaço de Trabalho do Software

    Set teste = CreateWorkspace("", "admin", "")
    Set movimento = teste.CreateDatabase("movimento.mdb", dbLangGeneral)

    Set tab_deslocamento = movimento.CreateTableDef("deslocamento")
    With tab_deslocamento

        .Fields.Append .CreateField("tempo", dbDouble)
        .Fields.Append .CreateField("atuador_A", dbDouble)
        .Fields.Append .CreateField("atuador_B", dbDouble)
        .Fields.Append .CreateField("atuador_C", dbDouble)
        .Fields.Append .CreateField("atuador_D", dbDouble)
        .Fields.Append .CreateField("d", dbDouble)
        .Fields.Append .CreateField("tetax", dbDouble)
        .Fields.Append .CreateField("tetay", dbDouble)

    End With

    movimento.TableDefs.Append tab_deslocamento

    Set tab_tensao = movimento.CreateTableDef("tensao")
    With tab_tensao

        .Fields.Append .CreateField("tempo", dbDouble)
        .Fields.Append .CreateField("atuador_A", dbDouble)
        .Fields.Append .CreateField("atuador_B", dbDouble)
        .Fields.Append .CreateField("atuador_C", dbDouble)
        .Fields.Append .CreateField("atuador_D", dbDouble)

    End With

    movimento.TableDefs.Append tab_tensao

'Preparação das Tabelas para Manipulação de dados

    Set rec_deslocamento = movimento.OpenRecordset("deslocamento")
    Set rec_tensao = movimento.OpenRecordset("tensao")

'Edição de Registros - Cálculo do Deslocamento instantâneo

    For amostragem = 0 To Int(Ent_Dados.t_max / CDbl(TB_Tempo))

        rec_deslocamento.AddNew
        rec_tensao.AddNew

        t_atual = amostragem * CDbl(TB_Tempo)

        rec_deslocamento!tempo = t_atual
        rec_tensao!tempo = t_atual

        If ac_lin <> 0 Then d = deslocamento(t_atual, Ent_Dados.t_max, vc_lin, ac_lin)
        If ac_tetax <> 0 Then tetax = deslocamento(t_atual, Ent_Dados.t_max, vc_tetax,
ac_tetax)
        If ac_tetay <> 0 Then tetay = deslocamento(t_atual, Ent_Dados.t_max, vc_tetay,
ac_tetay)

        rec_deslocamento!d = d
        rec_deslocamento!tetax = tetax
        rec_deslocamento!tetay = tetay

        Xg = d * Ent_Dados.cos_gama
        Zg = d * Ent_Dados.sin_gama
```

```
'Cálculo das coordenadas de fixação

        Xa = Xg + (Principal.TB_2b) / 2 * Cos(tetay)
        Ya = 0
        Za = Zg - (Principal.TB_2b) / 2 * Sin(tetay)

        Xc = Xg - (Principal.TB_2b) / 2 * Cos(tetay) - (Principal.TB_2a) / 2 *
Sin(tetax) * Sin(tetay)
        Yc = 0 - (Principal.TB_2a) / 2 * Cos(tetax)
        Zc = Zg + (Principal.TB_2b) / 2 * Sin(tetay) - (Principal.TB_2a) / 2 *
Sin(tetax) * Cos(tetay)

        Xd = Xg - (Principal.TB_2b) / 2 * Cos(tetay) + (Principal.TB_2a) / 2 *
Sin(tetax) * Sin(tetay)
        Yd = 0 + (Principal.TB_2a) / 2 * Cos(tetax)
        Zd = Zg + (Principal.TB_2b) / 2 * Sin(tetay) + (Principal.TB_2a) / 2 *
Sin(tetax) * Cos(tetay)

'Cálculo dos deltas -> H(t+1)-H(t)
'Preenchimento dos Registros

        h_temp = Ha
        Ha = Atuador(Xa, Ya, Za, CDbl(Principal.TB_AX), CDbl(Principal.TB_AY),
CDbl(Principal.TB_AZ), CDbl(Principal.TB_CompA))
        rec_deslocamento!atuador_a = Ha - h_temp
        rec_tensao!atuador_a = ((Ha - htemp) + 360) / 72

        h_temp = Hb
        Hb = Atuador(Xa, Ya, Za, CDbl(Principal.TB_BX), CDbl(Principal.TB_BY),
CDbl(Principal.TB_BZ), CDbl(Principal.TB_CompB))
        rec_deslocamento!atuador_b = Hb - h_temp
        rec_tensao!atuador_b = ((Hb - htemp) + 360) / 72

        h_temp = Hc
        Hc = Atuador(Xc, Yc, Zc, CDbl(Principal.TB_CX), CDbl(Principal.TB_CY),
CDbl(Principal.TB_CZ), CDbl(Principal.TB_CompC))
        rec_deslocamento!atuador_c = Hc - h_temp
        rec_tensao!atuador_c = ((Hc - htemp) + 360) / 72

        h_temp = Hd
        Hd = Atuador(Xd, Yd, Zd, CDbl(Principal.TB_DX), CDbl(Principal.TB_DY),
CDbl(Principal.TB_DZ), CDbl(Principal.TB_CompD))
        rec_deslocamento!atuador_d = Hd - h_temp
        rec_tensao!atuador_d = ((Hd - htemp) + 360) / 72

        rec_deslocamento.Update
        rec_tensao.Update

    Next amostragem

'Linha adicional na tabela de tensão para parar motores

    rec_tensao.AddNew
        rec_tensao!tempo = t_atual + CDbl(TB_Tempo)
        rec_tensao!atuador_a = 0
        rec_tensao!atuador_b = 0
        rec_tensao!atuador_c = 0
        rec_tensao!atuador_d = 0
    rec_tensao.Update

    Deslocamentos.Show

End Sub

'*****************************************************************************************
'*
'*      Função Form_Load : Calcula Velocidades/Acelerações Corrigidas
'*                         Apresentação gráfica - Antiga/Nova Situações
'*
'*****************************************************************************************

Private Sub Form_Load()

    Lb_tmax = Ent_Dados.t_max
```

```
    Lb_Vlin = Ent_Dados.TB_VLMax
    Lb_Alin = Ent_Dados.TB_ALMax
    LB_Vang = Ent_Dados.TB_VAMax
    LB_Aang = Ent_Dados.TB_AAMax

'Corrige e Exibe dados de Velocidade linear/angular

    vc_lin = corrige(Ent_Dados.t_xz, Ent_Dados.t_max, 1, CDbl(Ent_Dados.v_lin))
    Lb_Vclin(0).Caption = vc_lin
    vc_tetax = corrige(Ent_Dados.t_tetax, Ent_Dados.t_max, 1, CDbl(Ent_Dados.v_tetax))
    Lb_Vctetax(1).Caption = vc_tetax
    vc_tetay = corrige(Ent_Dados.t_tetay, Ent_Dados.t_max, 1, CDbl(Ent_Dados.v_tetay))
    Lb_Vctetay(2).Caption = vc_tetay

'Corrige e Exibe dados de Aceleração linear/angular

    ac_lin = corrige(Ent_Dados.t_xz, Ent_Dados.t_max, 2, CDbl(Ent_Dados.TB_ALMax))
    Lb_Aclin(1).Caption = ac_lin
    ac_tetax = corrige(Ent_Dados.t_tetax, Ent_Dados.t_max, 2, CDbl(Ent_Dados.TB_AAMax))
    Lb_Actetax(0).Caption = ac_tetax
    ac_tetay = corrige(Ent_Dados.t_tetay, Ent_Dados.t_max, 2, CDbl(Ent_Dados.TB_AAMax))
    Lb_Actetay(2).Caption = ac_tetay

End Sub
```

```
'*************************************************************************
'*                                                                       **
'*              PHOENIX CONTACT GmbH & Co., D-32819 Blomberg             **
'*              (c) Phoenix Contact 01/1998                              **
'*                                                                       **
'*************************************************************************
'*                                                                       **
'* Project      : HIGH LEVEL LANGUAGE INTERFACE                          **
'*                for PC-based INTERBUS G4 standard controllers (SC series) **
'* Sub-Project  : Application Interface for Visual Basic 4 or higher     **
'*                                                                       **
'*************************************************************************

'*
'* File         : IBSG4HLI.BAS
'* Component(s) : -
'* Device(s)    : independent
'* Terminology  : -
'*              :
'* Definition   : S. Gallmann
'* Author       : S. Gallmann
'*              :
'* Version      : 1.04
'* Status       : Released
'*              :
'* Description  : Definition/Declaration of Constants, Types, Functions and Class
'*                of IBS PC SC HLI for 16-Bit and 32-Bit-Version of Windows
'*************************************************************************
'* Change Notes :
'*
'* Date      Version     Author     Description
'* ----------------------------------------------------------------------
'* 04/98     1.00        SG         Released
'* 04/98     1.02        SG         Change in Definition of T_HLI_PCP_DEVICE_INFO;
'*                                  Add. Event Code HLI_EVENT_CANNOT_ACTIVATE_CFG
'* 09/98     1.03        SG         No changes in this module
'* 10/98     1.04        SG         Correction of return value
IBS_HLI_GetDiagnosticInfo()


' ---------------------------------------------------------------------
'                      I.  CONSTANTS
' ---------------------------------------------------------------------


' Controller board access constants ;
' use these constants for all functions that have the <CID> parameter

' IBS PC ISA SC/I-T  Board 1 ... 4
Public Const ISASC1 = &H1
Public Const ISASC2 = &H2
Public Const ISASC3 = &H3
Public Const ISASC4 = &H4

' IBS ETH DSC/I-T  Controller 1 ... 4
Public Const ETHDSC1 = &H11
Public Const ETHDSC2 = &H12
Public Const ETHDSC3 = &H13
Public Const ETHDSC4 = &H14

'  INTERBUS operating modes **

 '  default operating mode of Controller (free running cycles): **
Public Const IBS_NORMAL = 0
Public Const IBS_STANDARD = 0

 '  application triggers INTERBUS-cycles : **
Public Const IBS_CONTROLLED = 1


'  classification of Process-data-object types used for
'  PDO registration : **

Public Const IBS_PDO_INPUT = 0                    '  input type **
```

```
Public Const IBS_PDO_OUTPUT = 1                    '  output type **

Public Const IBS_PDO_BOOL = 1                      '  bit object **
Public Const IBS_PDO_BITSTRING = 2                 '  bit object **
Public Const IBS_PDO_BYTE = 3                      '  byte object **
Public Const IBS_PDO_WORD = 4                      '  Integer object **
Public Const IBS_PDO_DWORD = 5                     '  double Integer object **


' ********* General error codes returned by HLI functions: ***** **

Public Const HLI_OKAY = &H0
Public Const HLI_OK = &H0

Public Const HLI_ACCESS_DENIED = &HFF01
Public Const HLI_INVALID_CID = &HFF02
Public Const HLI_NO_VALID_CFG = &HFF03
Public Const HLI_NO_LOG_WINDOW = &HFF04
Public Const HLI_LOGFILE_OPEN_ERROR = &HFF05

Public Const HLI_MSG_TOO_LONG = &HFF31
Public Const HLI_INVALID_HOST_DATA_ACCESS = &HFF32
Public Const HLI_NO_MESSAGE_AVAILABLE = &HFF33
Public Const HLI_SEND_MESSAGE_TIMEOUT = &HFF34

Public Const HLI_NO_MORE_EVENT = &HFF21

Public Const HLI_PCP_SET_NOT_FOUND = &HFF41
Public Const HLI_PCP_SET_INVALID = &HFF42

' Constants of the <BusState> member of the <T_HLI_STATE> type

Public Const HLI_IBS_READY = &H1                   '  dto.            **
Public Const HLI_IBS_ACTIVE = &H2                  '  dto.            **
Public Const HLI_IBS_RUN = &H3                     '  running cycle(s)**

 ' Add. states for the application-controlled mode
Public Const HLI_IBS_IDLE = &H4                    '  cycle finished **
Public Const HLI_IBS_PREPARING_CYCLE = &H5         '  processing I/O **
Public Const HLI_IBS_CAN_RUN = &H6                 '  ready for cycle**

 '  stop states
Public Const HLI_IBS_STOPPED = &HF001 '  Bus in stop     **
Public Const HLI_IBS_DETECT = &HF002 '  detecting error **
Public Const HLI_IBS_BUS_FAIL = &HF003 '  bus fail->stop  **
Public Const HLI_IBS_CONTROLLER_FAIL = &HF004 '  Controller error **

' constants for PCP device states

Public Const HLI_PCP_DEVICE_READY = 0
Public Const HLI_PCP_DEVICE_NOT_INITIATED = &HF012
Public Const HLI_PCP_DEVICE_BUSY = &HF013
Public Const HLI_PCP_DEVICE_NOT_READY = &HF014

' <State> member constants for PCP service processing

Public Const HLI_PCP_SERVICE_OKAY = &H0
Public Const HLI_PCP_SERVICE_PENDING = &H1

' States for PCP device programs (<PRG_State> member

Public Const HLI_PCP_PRG_RESET = &H0
Public Const HLI_PCP_PRG_RUNNING = &H1
Public Const HLI_PCP_PRG_STOPPED = &H2

' PCP object types used with PCP-object translation functions

Public Const HLI_POT_BYTE = 0
Public Const HLI_POT_INT16 = 1
Public Const HLI_POT_INT32 = 2
Public Const HLI_POT_FLOAT = 3
Public Const HLI_POT_STRING = 4

' PCP service types received from event parameters
```

```
Public Const HLI_EVENT_LB_SEGMENT_ON = &H2007
Public Const HLI_EVENT_GROUP_OFF = &H2008
Public Const HLI_EVENT_GROUP_ON = &H2009
Public Const HLI_EVENT_BUS_WARNING = &H200A
Public Const HLI_EVENT_DEVICE_ENABLED = &H200B
Public Const HLI_EVENT_DEVICE_DISABLED = &H200C

'  Event group: PCP events

Public Const HLI_EG_PCP = &H40

Public Const HLI_EVENT_PCP_LOCAL_ABORT = &H4001
Public Const HLI_EVENT_PCP_REMOTE_ABORT = &H4002
Public Const HLI_EVENT_PCP_RECONNECTED = &H4003
Public Const HLI_EVENT_PCP_NOT_READY = &H4004

Public Const HLI_EVENT_PCP_SERVICE_FAILED = &H4005
Public Const HLI_EVENT_PCP_SERVICE_REJECTED = &H4006
Public Const HLI_EVENT_PCP_SERVICE_TIMEOUT = &H4007

'  Event group: application faults

Public Const HLI_EG_APP = &H80
Public Const HLI_EVENT_NO_VALID_CFG = &H8001
Public Const HLI_EVENT_NULL_POINTER = &H8002
Public Const HLI_EVENT_INVALID_PARA = &H8003

Public Const HLI_EVENT_UNKNOWN_DEVNO = &H8004
Public Const HLI_EVENT_INVALID_IDCODE = &H8005
Public Const HLI_EVENT_NO_PCP_DEVICE = &H8006
Public Const HLI_EVENT_WRONG_NR_OF_DEVICES = &H8007
Public Const HLI_EVENT_UNKNOWN_GROUP = &H8008
Public Const HLI_EVENT_NO_INPUT_DEVICE = &H8009
Public Const HLI_EVENT_NO_OUTPUT_DEVICE = &H800A

Public Const HLI_EVENT_TOO_MANY_PDOS = &H800B
Public Const HLI_EVENT_INVALID_PDO_TYPE = &H800C
Public Const HLI_EVENT_INVALID_PDO_SIZE = &H800D
Public Const HLI_EVENT_INVALID_PDO_POINTER = &H800E
Public Const HLI_EVENT_INVALID_SYNCH_OP = &H800F
Public Const HLI_EVENT_IBS_NOT_RUNNING = &H8010
Public Const HLI_EVENT_IBS_DETECTING = &H8011
Public Const HLI_EVENT_DEVICE_NOT_INITIATED = &H8012
Public Const HLI_EVENT_DEVICE_BUSY = &H8013
Public Const HLI_EVENT_DEVICE_NOT_READY = &H8014

Public Const HLI_EVENT_CFG_INVALID_DEVNO = &H8021
Public Const HLI_EVENT_CFG_INVALID_IDCODE = &H8022
Public Const HLI_EVENT_CFG_INVALID_LEVEL = &H8023
Public Const HLI_EVENT_CFG_INVALID_GROUP = &H8024

Public Const HLI_EVENT_ACTION_NOT_ALLOWED = &H8031
Public Const HLI_EVENT_CANNOT_ACTIVATE_CFG = &H8032

' **** Constants used by INTERBUS service functions *********

 ' IBS_HLI_Switchxxx() functions */
Public Const HLI_SEGMENT_OFF = &H0
Public Const HLI_SEGMENT_ON = &H1
Public Const HLI_GROUP_OFF = &H0
Public Const HLI_GROUP_ON = &H1
Public Const HLI_DEVICE_DISABLED = &H0
Public Const HLI_DEVICE_ENABLED = &H1

 '  Constants for watchdog timeout values used by the
 '  IBS_HLI_ActivateWatchdog() function
Public Const HLI_WDT_8 = &H0                    '  Timeout = 8.2 ms     **
Public Const HLI_WDT_16 = &H4                   '  Timeout = 16.4 ms    **
Public Const HLI_WDT_32 = &H8                   '  Timeout = 32.8 ms    **
Public Const HLI_WDT_65 = &HC                   '  Timeout = 65.5 ms    **
Public Const HLI_WDT_131 = &H10                 '  Timeout = 131.1 ms   **
Public Const HLI_WDT_262 = &H14                 '  Timeout = 262.1 ms   **
Public Const HLI_WDT_524 = &H18                 '  Timeout = 524.3 ms   **
Public Const HLI_WDT_1048 = &H1C                '  Timeout = 1048.6 ms **
```

```
Public Const HLI_PCP_SERVICE_READ = 1
Public Const HLI_PCP_SERVICE_WRITE = 2
Public Const HLI_PCP_SERVICE_IDENTIFY = 3
Public Const HLI_PCP_SERVICE_START = 4
Public Const HLI_PCP_SERVICE_STOP = 5
Public Const HLI_PCP_SERVICE_RESUME = 6
Public Const HLI_PCP_SERVICE_RESET = 7


'  ******************* HLI Event type definitions *****************

'  HLI_EG_xxx constants define event groups to use with the
'  IBS_HLI_EnableEvents(), IBS_HLI_DisableEvents(),
'  IBS_HLI_EnableLogEvents() and IBS_HLI_DisableLogEvents() functions

Public Const HLI_EG_ALL = &HFF

'  Event group: internal faults

Public Const HLI_EG_INTERNAL = &H1

Public Const HLI_EVENT_MEMORY_FAULT = &H101
Public Const HLI_EVENT_OS_FAULT = &H102
Public Const HLI_EVENT_NOTE_NO_PF_EV = &H103
Public Const HLI_EVENT_NOTE_NO_PF_RESET_EV = &H104

'  Event group: DDI errors

Public Const HLI_EG_DDI = &H2

Public Const HLI_EVENT_DDI_NOT_AVAILABLE = &H201
Public Const HLI_EVENT_DDI_ERROR = &H202
Public Const HLI_EVENT_NO_RIGHTS = &H203
Public Const HLI_EVENT_FW_CNF_TIMEOUT = &H204
Public Const HLI_EVENT_FW_CNF_ERROR = &H205

'  Event group: controller faults

Public Const HLI_EG_CTRL = &H4

Public Const HLI_EVENT_CONTROLLER_FAIL = &H401
Public Const HLI_EVENT_SYSTEM_FAIL = &H402
Public Const HLI_EVENT_WATCHDOG_ACTIVE = &H403
Public Const HLI_EVENT_WATCHDOG_SHUT = &H404

'  Event group: bus faults

Public Const HLI_EG_BUSFAULT = &H8

Public Const HLI_EVENT_LOCATED_BUS_ERROR = &H801
Public Const HLI_EVENT_UNLOCATED_BUS_ERROR = &H802
Public Const HLI_EVENT_INVALID_DATA_CYCLE = &H803

'  Event group: device state messages

Public Const HLI_EG_DEVICEMSG = &H10

Public Const HLI_EVENT_DEVICE_ERROR = &H1001
Public Const HLI_EVENT_DEVICE_MAU = &H1002
Public Const HLI_EVENT_DEVICE_RECONFIG = &H1003
Public Const HLI_EVENT_DEVICE_ERROR_RESET = &H1004
Public Const HLI_EVENT_DEVICE_STATE = &H1005

'  Event group: bus state changes

Public Const HLI_EG_BUSSTATE = &H20

Public Const HLI_EVENT_BUS_START = &H2001
Public Const HLI_EVENT_BUS_STOP = &H2002
Public Const HLI_EVENT_ALARM_STOP = &H2003
Public Const HLI_EVENT_RB_SEGMENT_OFF = &H2004
Public Const HLI_EVENT_RB_SEGMENT_ON = &H2005
Public Const HLI_EVENT_LB_SEGMENT_OFF = &H2006
```

```
'  *************** Event text & window constants ***************** **

Public Const HLI_ENGLISH = &H0
Public Const HLI_DEUTSCH = &H1
Public Const HLI_EW_FIXED = &H0
Public Const HLI_EW_FILTERCHANGE = &H1

' HOST-COP-communication

Public Const DATA_AREA_SIZE = 5120


' ------------------------------------------------------------------**
'                    II.  TYPE DEFINITIONS                          **
' ------------------------------------------------------------------**

'  struct to hold HLI state information **
'*************************************************************

' record to hold HLI state information

  Type T_HLI_STATE

        BusState        As Integer
        DiagState       As Integer
        EventIndication As Integer
      End Type


'  record to hold HLI date/time information

   Type T_HLI_TIME

        d               As Integer
        mo              As Integer
        y               As Integer
        h               As Integer
        m               As Integer
        S               As Integer
    End Type

' record to hold HLI event information

  Type T_HLI_EVENT

        Timestamp       As T_HLI_TIME
        CID             As Integer
        Code            As Integer
        Param(0 To 4)   As Integer
      End Type

' record to controller information

  Type T_HLI_SC_INFO

        reserved1  As Byte
        Ident      As String * 32
        reserved2  As Byte
        FW_Version As String * 8
        reserved3  As Byte
        HW_Version As String * 5
    End Type

' record to hold Bus information

  Type T_HLI_BUS_INFO

        Devices         As Integer
        IO_Points       As Integer
        PCP_Devices     As Integer
        PD_Length       As Integer
        Cycletime       As Long
      End Type

' record to hold bus device data
```

```
  Type T_HLI_DEVICE_DATA

        Segment         As Integer
        Position        As Integer
        ID_Code         As Integer
        LengthCode      As Integer
        Level           As Integer
        Group           As Integer
        Alternative     As Integer
        Deactivated     As Integer
        Station(0 To 31) As Byte
      End Type

' record to hold bus device information

  Type T_HLI_DEVICE_INFO

        Segment         As Integer
        Position        As Integer
        ID_Code         As Integer
        LengthCode      As Integer
        Level           As Integer
        Group           As Integer
        Alternative     As Integer
        InBitLength     As Integer
        OutBitLength    As Integer
        PCP_CR          As Integer
        Active          As Integer
      End Type

' record to hold PCP device information

  Type T_HLI_PCP_DEVICE_INFO

        Segment         As Integer
        Position        As Integer
        State           As Integer
        ID_Code         As Integer
        LengthCode      As Integer
        reserved1       As Byte
        Manufacturer    As String * 255
        reserved2       As Byte
        Name            As String * 255
        reserved3       As Byte
        Revision        As String * 255
      End Type

' record to hold pcp data object

  Type T_HLI_PCP_RW

        Segment         As Integer
        Position        As Integer
        Index           As Integer
        Subindex        As Integer
        length          As Integer
        State           As Integer
        Data(0 To 255)  As Byte
      End Type

' record to hold pcp program object

  Type T_HLI_PCP_PRG

        Segment         As Integer
        Position        As Integer
        Index           As Integer
        PRG_State       As Integer
        State           As Integer
      End Type


' record to diagnostic information
```

```
Type T_HLI_SC_DIAG

      Cycle_Count            As Long
      Cycle_Error_Count      As Long
      ID_Cycle_Count         As Long
      ID_Cycle_Error_Count   As Long
      Data_Cycle_Count       As Long
      Data_Cycle_Error_Count As Long
    End Type


' record to hold message block for COP communication

    Type T_HLI_MSG

       Code            As Integer
       nParms          As Integer
       Parms(0 To 511) As Integer
    End Type




' ------------------------------------------------------------------**
'                        H L I  FUNCTIONS                           **
' ------------------------------------------------------------------**


' ###################### Windows 16-Bit-Version ####################


#If Win16 Then

Public Declare Function IBS_HLI_GetVersion Lib "G4HLIW16.DLL" (ByVal destination As
String) As Integer

' HLI-Init-functions

Public Declare Function IBS_HLI_Init Lib "G4HLIW16.DLL" (ByVal CID As Integer, ByRef SC
As T_HLI_STATE, ByVal ibs_mode As Integer) As Integer
Public Declare Function IBS_HLI_Init_CFG Lib "G4HLIW16.DLL" (ByVal CID As Integer, ByRef
SC As T_HLI_STATE, ByVal ibs_mode As Integer, ByVal nDevices As Integer, ByRef
DeviceList As T_HLI_DEVICE_DATA) As Integer
Public Declare Function IBS_HLI_Exit Lib "G4HLIW16.DLL" (ByVal CID As Integer) As
Integer

' HLI-Event management

Public Declare Function IBS_HLI_EnableEvents Lib "G4HLIW16.DLL" (ByVal CID As Integer,
ByVal eventgroup As Integer) As Integer
Public Declare Function IBS_HLI_DisableEvents Lib "G4HLIW16.DLL" (ByVal CID As Integer,
ByVal eventgroup As Integer) As Integer
Public Declare Function IBS_HLI_GetEventInfo Lib "G4HLIW16.DLL" (ByVal CID As Integer,
ByRef Ev As T_HLI_EVENT) As Integer
Public Declare Function IBS_HLI_GetEventText Lib "G4HLIW16.DLL" (ByRef Ev As
T_HLI_EVENT, ByVal Language As Integer, ByVal destination As String) As Integer
Public Declare Function IBS_HLI_EnableLogEvents Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal eventgroup As Integer) As Integer
Public Declare Function IBS_HLI_DisableLogEvents Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal eventgroup As Integer) As Integer
Public Declare Function IBS_HLI_CreateLogWindow Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal Langauge As Integer, ByVal FilterChange As Integer) As Integer
Public Declare Function IBS_HLI_PopupLogWindow Lib "G4HLIW16.DLL" (ByVal CID As Integer)
As Integer
Public Declare Function IBS_HLI_CloseLogWindow Lib "G4HLIW16.DLL" (ByVal CID As Integer)
As Integer
Public Declare Function IBS_HLI_LogUserEvent Lib "G4HLIW16.DLL" (ByVal CID As Integer,
ByVal UserText As String, ByVal WithTimeStamp As Boolean) As Integer
Public Declare Function IBS_HLI_LogAutoSave Lib "G4HLIW16.DLL" (ByVal CID As Integer,
ByVal Filename As String, ByVal Append As Boolean) As Integer

' HLI-INTERBUS-Information functions
```

```
Public Declare Function IBS_HLI_GetControllerInfo Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByRef CI As T_HLI_SC_INFO) As Integer
Public Declare Function IBS_HLI_GetBusInfo Lib "G4HLIW16.DLL" (ByVal CID As Integer,
ByRef BI As T_HLI_BUS_INFO) As Integer
Public Declare Function IBS_HLI_GetDeviceInfo Lib "G4HLIW16.DLL" (ByVal CID As Integer,
ByVal Index As Integer, ByRef DI As T_HLI_DEVICE_INFO) As Integer
Public Declare Function IBS_HLI_GetDiagnosticInfo Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByRef DI As T_HLI_SC_DIAG) As Integer

' HLI-INTERBUS-Management-functions

 ' HLI-State-Machine-function
Public Declare Function IBS_HLI_Process Lib "G4HLIW16.DLL" (ByVal CID As Integer) As
Integer

 ' HLI-process data management
Public Declare Function IBS_HLI_PD_In Lib "G4HLIW16.DLL" (ByVal CID As Integer) As
Integer
Public Declare Function IBS_HLI_PD_Out Lib "G4HLIW16.DLL" (ByVal CID As Integer) As
Integer
Public Declare Function IBS_HLI_PD_Input Lib "G4HLIW16.DLL" (ByVal CID As Integer, ByRef
lpAppVar As Any) As Integer
Public Declare Function IBS_HLI_PD_Output Lib "G4HLIW16.DLL" (ByVal CID As Integer,
ByRef lpAppVar As Any) As Integer
Public Declare Function IBS_HLI_PD_DeviceIn Lib "G4HLIW16.DLL" (ByVal CID As Integer,
ByVal segment_nr As Integer, ByVal segment_pos As Integer) As Integer
Public Declare Function IBS_HLI_PD_DeviceOut Lib "G4HLIW16.DLL" (ByVal CID As Integer,
ByVal segment_nr As Integer, ByVal segment_pos As Integer) As Integer
Public Declare Function IBS_HLI_ResetAllOutputs Lib "G4HLIW16.DLL" (ByVal CID As
Integer) As Integer

 ' for application controlled mode only
Public Declare Function IBS_HLI_GetCyclePhase Lib "G4HLIW16.DLL" () As Integer
Public Declare Function IBS_HLI_RunCycle Lib "G4HLIW16.DLL" () As Integer
Public Declare Function IBS_HLI_RunCompleteCycle Lib "G4HLIW16.DLL" () As Integer

' INTERBUS-PD-Object registration function

   ' master function
Public Declare Function IBS_HLI_RegisterPDObject Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal PD_Type As Integer, ByVal segment_nr As Integer, ByVal segment_pos As
Integer, ByVal PDOType As Integer, ByVal ByteOffset As Integer, ByVal BitPosition As
Integer, ByVal length As Integer, ByRef lpAppVar As Any, ByRef lpChanged As Any) As
Integer
   ' Bool objects
Public Declare Function IBS_HLI_RegisterPDO_BOOL Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal PD_Type As Integer, ByVal segment_nr As Integer, ByVal segment_pos As
Integer, ByVal ByteOffset As Integer, ByVal BitPosition As Integer, ByRef lpAppVar As
Integer, ByRef lpChanged As Any) As Integer

   ' Bitstring objects
Public Declare Function IBS_HLI_RegisterPDO_BITSTRING Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal PD_Type As Integer, ByVal segment_nr As Integer, ByVal segment_pos As
Integer, ByVal ByteOffset As Integer, ByVal BitPosition As Integer, ByVal BitLength As
Integer, ByRef lpAppVar As Integer, ByRef lpChanged As Any) As Integer

   ' Integer objects
Public Declare Function IBS_HLI_RegisterPDO_BYTE Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal PD_Type As Integer, ByVal segment_nr As Integer, ByVal segment_pos As
Integer, ByVal ByteOffset As Integer, ByRef lpAppVar As Byte, ByRef lpChanged As Any) As
Integer

   ' Word objects
Public Declare Function IBS_HLI_RegisterPDO_WORD Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal PD_Type As Integer, ByVal segment_nr As Integer, ByVal segment_pos As
Integer, ByVal ByteOffset As Integer, ByRef lpAppVar As Integer, ByRef lpChanged As Any)
As Integer

   ' DWord objects
Public Declare Function IBS_HLI_RegisterPDO_DWORD Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal PD_Type As Integer, ByVal segment_nr As Integer, ByVal segment_pos As
Integer, ByVal ByteOffset As Integer, ByRef lpAppVar As Long, ByRef lpChanged As Any) As
Integer
```

```
    ' unregistering objects
Public Declare Function IBS_HLI_UnregisterPDObject Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByRef lpAppVar As Any) As Integer

' INTERBUS-PCP functions

Public Declare Function IBS_HLI_PCP_SetConnectTimeout Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal timeout As Integer) As Integer
Public Declare Function IBS_HLI_PCP_DeviceState Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal Segment As Integer, ByVal Position As Integer) As Integer
Public Declare Function IBS_HLI_PCP_ReconnectDevice Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal Segment As Integer, ByVal Position As Integer) As Integer
Public Declare Function IBS_HLI_PCP_ReadRequest Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpRRQ As T_HLI_PCP_RW) As Integer
Public Declare Function IBS_HLI_PCP_ReadService Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpRRQ As T_HLI_PCP_RW, timeout_ms As Integer) As Integer
Public Declare Function IBS_HLI_PCP_WriteRequest Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpWRQ As T_HLI_PCP_RW) As Integer
Public Declare Function IBS_HLI_PCP_WriteService Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpWRQ As T_HLI_PCP_RW, timeout_ms As Integer) As Integer
Public Declare Function IBS_HLI_PCP_IdentifyRequest Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpDI As T_HLI_PCP_DEVICE_INFO) As Integer
Public Declare Function IBS_HLI_PCP_IdentifyService Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpDI As T_HLI_PCP_DEVICE_INFO, timeout_ms As Integer) As Integer
Public Declare Function IBS_HLI_PCP_RegisterClientObject Lib "G4HLIW16.DLL" (ByVal CID
As Integer, lpWRQ As T_HLI_PCP_RW) As Integer
Public Declare Function IBS_HLI_PCP_StartRequest Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG) As Integer
Public Declare Function IBS_HLI_PCP_StartService Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG, timeout_ms As Integer) As Integer
Public Declare Function IBS_HLI_PCP_StopRequest Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG) As Integer
Public Declare Function IBS_HLI_PCP_StopService Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG, timeout_ms As Integer) As Integer
Public Declare Function IBS_HLI_PCP_ResumeRequest Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG) As Integer
Public Declare Function IBS_HLI_PCP_ResumeService Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG, timeout_ms As Integer) As Integer
Public Declare Function IBS_HLI_PCP_ResetRequest Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG) As Integer
Public Declare Function IBS_HLI_PCP_ResetService Lib "G4HLIW16.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG, timeout_ms As Integer) As Integer

' INTERBUS-PCP-object translation functions

Public Declare Function IBS_HLI_PCP_LoadObjectByName Lib "G4HLIW16.DLL" (ByVal Filename
As String, ByVal VarName As String, ByRef lpPCPObj As T_HLI_PCP_RW) As Integer
Public Declare Function IBS_HLI_PCP_LoadObjectByIndex Lib "G4HLIW16.DLL" (ByVal Filename
As String, ByVal Index As Integer, ByRef lpPCPObj As T_HLI_PCP_RW) As Integer
Public Declare Function IBS_HLI_PCP_LoadObjectByEntry Lib "G4HLIW16.DLL" (ByVal Filename
As String, ByVal Entry As Integer, ByRef lpPCPObj As T_HLI_PCP_RW) As Integer
Public Declare Function IBS_HLI_PCP_InitObject Lib "G4HLIW16.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Segment As Integer, ByVal Position As Integer, ByVal Index As
Integer, ByVal Subindex As Integer, ByVal POT As Integer, ByVal nElements As Integer,
ByRef lpData As Any) As Integer
Public Declare Function IBS_HLI_PCPObject_SetInt16 Lib "G4HLIW16.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Offset As Integer, ByVal VInt16 As Integer) As Integer
Public Declare Function IBS_HLI_PCPObject_SetInt32 Lib "G4HLIW16.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Offset As Integer, ByVal VInt32 As Long) As Integer
Public Declare Function IBS_HLI_PCPObject_SetFloat Lib "G4HLIW16.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Offset As Integer, ByVal VFloat As Single) As Integer
Public Declare Function IBS_HLI_PCPObject_SetString Lib "G4HLIW16.DLL" (ByRef lpPCPObj
As T_HLI_PCP_RW, ByVal Offset As Integer, ByVal lpS As String) As Integer

Public Declare Function IBS_HLI_PCPObject_GetInt16 Lib "G4HLIW16.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Offset As Integer, ByRef lpInt16 As Integer) As Integer
Public Declare Function IBS_HLI_PCPObject_GetInt32 Lib "G4HLIW16.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Offset As Integer, ByRef lpInt32 As Long) As Integer
Public Declare Function IBS_HLI_PCPObject_GetFloat Lib "G4HLIW16.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Offset As Integer, ByRef lpFloat As Single) As Integer
Public Declare Function IBS_HLI_PCPObject_GetString Lib "G4HLIW16.DLL" (ByRef lpPCPObj
As T_HLI_PCP_RW, ByVal Offset As Integer, ByVal lpS As String) As Integer

' Controller-Services
```

```
Public Declare Function IBS_HLI_ActivateWatchdog Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal WDT As Integer) As Integer

' INTERBUS-Services

Public Declare Function IBS_HLI_StartBus Lib "G4HLIW16.DLL" (ByVal CID As Integer) As
Integer
Public Declare Function IBS_HLI_StopBus Lib "G4HLIW16.DLL" (ByVal CID As Integer) As
Integer
Public Declare Function IBS_HLI_AlarmStop Lib "G4HLIW16.DLL" (ByVal CID As Integer) As
Integer
Public Declare Function IBS_HLI_SwitchRemoteBus Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal segment_nr As Integer, ByVal switch_code As Integer) As Integer
Public Declare Function IBS_HLI_SwitchLocalBus Lib "G4HLIW16.DLL" (ByVal CID As Integer,
ByVal segment_nr As Integer, ByVal switch_code As Integer) As Integer
Public Declare Function IBS_HLI_SwitchGroup Lib "G4HLIW16.DLL" (ByVal CID As Integer,
ByVal group_nr As Integer, ByVal Alternative As Integer, ByVal switch_code As Integer)
As Integer
Public Declare Function IBS_HLI_ChangeDeviceState Lib "G4HLIW16.DLL" (ByVal CID As
Integer, ByVal Segment As Integer, ByVal Position As Integer, ByVal new_state As
Integer) As Integer

#End If



' ###################### Windows 32-Bit-Version ###################



#If Win32 Then

Public Declare Function IBS_HLI_GetVersion Lib "G4HLIW32.DLL" (ByVal destination As
String) As Integer

' HLI-Init-functions

Public Declare Function IBS_HLI_Init Lib "G4HLIW32.DLL" (ByVal CID As Integer, ByRef SC
As T_HLI_STATE, ByVal ibs_mode As Integer) As Integer
Public Declare Function IBS_HLI_Init_CFG Lib "G4HLIW32.DLL" (ByVal CID As Integer, ByRef
SC As T_HLI_STATE, ByVal ibs_mode As Integer, ByVal nDevices As Integer, ByRef
DeviceList As T_HLI_DEVICE_DATA) As Integer
Public Declare Function IBS_HLI_Exit Lib "G4HLIW32.DLL" (ByVal CID As Integer) As
Integer

' HLI-Event management

Public Declare Function IBS_HLI_EnableEvents Lib "G4HLIW32.DLL" (ByVal CID As Integer,
ByVal eventgroup As Integer) As Integer
Public Declare Function IBS_HLI_DisableEvents Lib "G4HLIW32.DLL" (ByVal CID As Integer,
ByVal eventgroup As Integer) As Integer
Public Declare Function IBS_HLI_GetEventInfo Lib "G4HLIW32.DLL" (ByVal CID As Integer,
ByRef Ev As T_HLI_EVENT) As Integer
Public Declare Function IBS_HLI_GetEventText Lib "G4HLIW32.DLL" (ByRef Ev As
T_HLI_EVENT, ByVal Language As Integer, ByVal destination As String) As Integer
Public Declare Function IBS_HLI_EnableLogEvents Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal eventgroup As Integer) As Integer
Public Declare Function IBS_HLI_DisableLogEvents Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal eventgroup As Integer) As Integer
Public Declare Function IBS_HLI_CreateLogWindow Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal Langauge As Integer, ByVal FilterChange As Integer) As Integer
Public Declare Function IBS_HLI_PopupLogWindow Lib "G4HLIW32.DLL" (ByVal CID As Integer)
As Integer
Public Declare Function IBS_HLI_CloseLogWindow Lib "G4HLIW32.DLL" (ByVal CID As Integer)
As Integer
Public Declare Function IBS_HLI_LogUserEvent Lib "G4HLIW32.DLL" (ByVal CID As Integer,
ByVal UserText As String, ByVal WithTimeStamp As Boolean) As Integer
Public Declare Function IBS_HLI_LogAutoSave Lib "G4HLIW32.DLL" (ByVal CID As Integer,
ByVal Filename As String, ByVal Append As Boolean) As Integer

' HLI-INTERBUS-Information functions
```

```
Public Declare Function IBS_HLI_GetControllerInfo Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByRef CI As T_HLI_SC_INFO) As Integer
Public Declare Function IBS_HLI_GetBusInfo Lib "G4HLIW32.DLL" (ByVal CID As Integer,
ByRef BI As T_HLI_BUS_INFO) As Integer
Public Declare Function IBS_HLI_GetDeviceInfo Lib "G4HLIW32.DLL" (ByVal CID As Integer,
ByVal Index As Integer, ByRef DI As T_HLI_DEVICE_INFO) As Integer
Public Declare Function IBS_HLI_GetDiagnosticInfo Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByRef DI As T_HLI_SC_DIAG) As Integer

' HLI-INTERBUS-Management-functions
 ' HLI-State-Machine-function
Public Declare Function IBS_HLI_Process Lib "G4HLIW32.DLL" (ByVal CID As Integer) As
Integer

 ' HLI-process data management
Public Declare Function IBS_HLI_PD_In Lib "G4HLIW32.DLL" (ByVal CID As Integer) As
Integer
Public Declare Function IBS_HLI_PD_Out Lib "G4HLIW32.DLL" (ByVal CID As Integer) As
Integer
Public Declare Function IBS_HLI_PD_Input Lib "G4HLIW32.DLL" (ByVal CID As Integer, ByRef
lpAppVar As Any) As Integer
Public Declare Function IBS_HLI_PD_Output Lib "G4HLIW32.DLL" (ByVal CID As Integer,
ByRef lpAppVar As Any) As Integer
Public Declare Function IBS_HLI_PD_DeviceIn Lib "G4HLIW32.DLL" (ByVal CID As Integer,
ByVal segment_nr As Integer, ByVal segment_pos As Integer) As Integer
Public Declare Function IBS_HLI_PD_DeviceOut Lib "G4HLIW32.DLL" (ByVal CID As Integer,
ByVal segment_nr As Integer, ByVal segment_pos As Integer) As Integer
Public Declare Function IBS_HLI_ResetAllOutputs Lib "G4HLIW32.DLL" (ByVal CID As
Integer) As Integer

 ' for application controlled mode only
Public Declare Function IBS_HLI_GetCyclePhase Lib "G4HLIW32.DLL" () As Integer
Public Declare Function IBS_HLI_RunCycle Lib "G4HLIW32.DLL" () As Integer
Public Declare Function IBS_HLI_RunCompleteCycle Lib "G4HLIW32.DLL" () As Integer

' INTERBUS-PD-Object registration function

   ' master function
Public Declare Function IBS_HLI_RegisterPDObject Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal PD_Type As Integer, ByVal segment_nr As Integer, ByVal segment_pos As
Integer, ByVal PDOType As Integer, ByVal ByteOffset As Integer, ByVal BitPosition As
Integer, ByVal length As Integer, ByRef lpAppVar As Any, ByRef lpChanged As Any) As
Integer

   ' Bool objects
Public Declare Function IBS_HLI_RegisterPDO_BOOL Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal PD_Type As Integer, ByVal segment_nr As Integer, ByVal segment_pos As
Integer, ByVal ByteOffset As Integer, ByVal BitPosition As Integer, ByRef lpAppVar As
Integer, ByRef lpChanged As Any) As Integer

   ' Bitstring objects
Public Declare Function IBS_HLI_RegisterPDO_BITSTRING Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal PD_Type As Integer, ByVal segment_nr As Integer, ByVal segment_pos As
Integer, ByVal ByteOffset As Integer, ByVal BitPosition As Integer, ByVal BitLength As
Integer, ByRef lpAppVar As Integer, ByRef lpChanged As Any) As Integer

   ' Integer objects
Public Declare Function IBS_HLI_RegisterPDO_BYTE Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal PD_Type As Integer, ByVal segment_nr As Integer, ByVal segment_pos As
Integer, ByVal ByteOffset As Integer, ByRef lpAppVar As Byte, ByRef lpChanged As Any) As
Integer

   ' Word objects
Public Declare Function IBS_HLI_RegisterPDO_WORD Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal PD_Type As Integer, ByVal segment_nr As Integer, ByVal segment_pos As
Integer, ByVal ByteOffset As Integer, ByRef lpAppVar As Integer, ByRef lpChanged As Any)
As Integer

   ' DWord objects
Public Declare Function IBS_HLI_RegisterPDO_DWORD Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal PD_Type As Integer, ByVal segment_nr As Integer, ByVal segment_pos As
Integer, ByVal ByteOffset As Integer, ByRef lpAppVar As Long, ByRef lpChanged As Any) As
Integer

   ' unregistering objects
```

```
Public Declare Function IBS_HLI_UnregisterPDObject Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByRef lpAppVar As Any) As Integer

' INTERBUS-PCP functions

Public Declare Function IBS_HLI_PCP_SetConnectTimeout Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal timeout As Integer) As Integer
Public Declare Function IBS_HLI_PCP_DeviceState Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal Segment As Integer, ByVal Position As Integer) As Integer
Public Declare Function IBS_HLI_PCP_ReconnectDevice Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal Segment As Integer, ByVal Position As Integer) As Integer
Public Declare Function IBS_HLI_PCP_ReadRequest Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpRRQ As T_HLI_PCP_RW) As Integer
Public Declare Function IBS_HLI_PCP_ReadService Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpRRQ As T_HLI_PCP_RW, timeout_ms As Integer) As Integer
Public Declare Function IBS_HLI_PCP_WriteRequest Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpWRQ As T_HLI_PCP_RW) As Integer
Public Declare Function IBS_HLI_PCP_WriteService Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpWRQ As T_HLI_PCP_RW, timeout_ms As Integer) As Integer
Public Declare Function IBS_HLI_PCP_IdentifyRequest Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpDI As T_HLI_PCP_DEVICE_INFO) As Integer
Public Declare Function IBS_HLI_PCP_IdentifyService Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpDI As T_HLI_PCP_DEVICE_INFO, timeout_ms As Integer) As Integer
Public Declare Function IBS_HLI_PCP_RegisterClientObject Lib "G4HLIW32.DLL" (ByVal CID
As Integer, lpWRQ As T_HLI_PCP_RW) As Integer
Public Declare Function IBS_HLI_PCP_StartRequest Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG) As Integer
Public Declare Function IBS_HLI_PCP_StartService Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG, timeout_ms As Integer) As Integer
Public Declare Function IBS_HLI_PCP_StopRequest Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG) As Integer
Public Declare Function IBS_HLI_PCP_StopService Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG, timeout_ms As Integer) As Integer
Public Declare Function IBS_HLI_PCP_ResumeRequest Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG) As Integer
Public Declare Function IBS_HLI_PCP_ResumeService Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG, timeout_ms As Integer) As Integer
Public Declare Function IBS_HLI_PCP_ResetRequest Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG) As Integer
Public Declare Function IBS_HLI_PCP_ResetService Lib "G4HLIW32.DLL" (ByVal CID As
Integer, lpPRG As T_HLI_PCP_PRG, timeout_ms As Integer) As Integer

' INTERBUS-PCP-object translation functions

Public Declare Function IBS_HLI_PCP_LoadObjectByName Lib "G4HLIW32.DLL" (ByVal Filename
As String, ByVal VarName As String, ByRef lpPCPObj As T_HLI_PCP_RW) As Integer
Public Declare Function IBS_HLI_PCP_LoadObjectByIndex Lib "G4HLIW32.DLL" (ByVal Filename
As String, ByVal Index As Integer, ByRef lpPCPObj As T_HLI_PCP_RW) As Integer
Public Declare Function IBS_HLI_PCP_LoadObjectByEntry Lib "G4HLIW32.DLL" (ByVal Filename
As String, ByVal Entry As Integer, ByRef lpPCPObj As T_HLI_PCP_RW) As Integer
Public Declare Function IBS_HLI_PCP_InitObject Lib "G4HLIW32.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Segment As Integer, ByVal Position As Integer, ByVal Index As
Integer, ByVal Subindex As Integer, ByVal POT As Integer, ByVal nElements As Integer,
ByRef lpData As Any) As Integer
Public Declare Function IBS_HLI_PCPObject_SetInt16 Lib "G4HLIW32.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Offset As Integer, ByVal VInt16 As Integer) As Integer
Public Declare Function IBS_HLI_PCPObject_SetInt32 Lib "G4HLIW32.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Offset As Integer, ByVal VInt32 As Long) As Integer
Public Declare Function IBS_HLI_PCPObject_SetFloat Lib "G4HLIW32.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Offset As Integer, ByVal VFloat As Single) As Integer
Public Declare Function IBS_HLI_PCPObject_SetString Lib "G4HLIW32.DLL" (ByRef lpPCPObj
As T_HLI_PCP_RW, ByVal Offset As Integer, ByVal lpS As String) As Integer

Public Declare Function IBS_HLI_PCPObject_GetInt16 Lib "G4HLIW32.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Offset As Integer, ByRef lpInt16 As Integer) As Integer
Public Declare Function IBS_HLI_PCPObject_GetInt32 Lib "G4HLIW32.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Offset As Integer, ByRef lpInt32 As Long) As Integer
Public Declare Function IBS_HLI_PCPObject_GetFloat Lib "G4HLIW32.DLL" (ByRef lpPCPObj As
T_HLI_PCP_RW, ByVal Offset As Integer, ByRef lpFloat As Single) As Integer
Public Declare Function IBS_HLI_PCPObject_GetString Lib "G4HLIW32.DLL" (ByRef lpPCPObj
As T_HLI_PCP_RW, ByVal Offset As Integer, ByVal lpS As String) As Integer

' Controller-Services
```

```
Public Declare Function IBS_HLI_ActivateWatchdog Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal WDT As Integer) As Integer

' INTERBUS-Services

Public Declare Function IBS_HLI_StartBus Lib "G4HLIW32.DLL" (ByVal CID As Integer) As
Integer
Public Declare Function IBS_HLI_StopBus Lib "G4HLIW32.DLL" (ByVal CID As Integer) As
Integer
Public Declare Function IBS_HLI_AlarmStop Lib "G4HLIW32.DLL" (ByVal CID As Integer) As
Integer
Public Declare Function IBS_HLI_SwitchRemoteBus Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal segment_nr As Integer, ByVal switch_code As Integer) As Integer
Public Declare Function IBS_HLI_SwitchLocalBus Lib "G4HLIW32.DLL" (ByVal CID As Integer,
ByVal segment_nr As Integer, ByVal switch_code As Integer) As Integer
Public Declare Function IBS_HLI_SwitchGroup Lib "G4HLIW32.DLL" (ByVal CID As Integer,
ByVal group_nr As Integer, ByVal Alternative As Integer, ByVal switch_code As Integer)
As Integer
Public Declare Function IBS_HLI_ChangeDeviceState Lib "G4HLIW32.DLL" (ByVal CID As
Integer, ByVal Segment As Integer, ByVal Position As Integer, ByVal new_state As
Integer) As Integer

' COP communication functions

Public Declare Function IBS_HLI_SendMessageToCop Lib "G4HLIW32.DLL" (ByVal COPID As
Integer, ByRef Msg As T_HLI_MSG, timeout As Integer) As Integer
Public Declare Function IBS_HLI_ReceiveMessageFromCop Lib "G4HLIW32.DLL" (ByVal COPID As
Integer, ByRef Msg As T_HLI_MSG) As Integer
Public Declare Function IBS_HLI_WaitForMessageFromCop Lib "G4HLIW32.DLL" (ByVal COPID As
Integer, ByRef Msg As T_HLI_MSG, ByVal timeout As Integer) As Integer

Public Declare Function IBS_HLI_ReadDataFromCop Lib "G4HLIW32.DLL" (ByVal COPID As
Integer, ByVal from_offset As Integer, ByVal nbytes As Integer, ByRef destination As
Any) As Integer
Public Declare Function IBS_HLI_WriteDataToCop Lib "G4HLIW32.DLL" (ByVal COPID As
Integer, ByVal to_offset As Integer, ByVal nbytes As Integer, ByRef source As Any) As
Integer

Public Declare Function IBS_HLI_GetCopDiagnostic Lib "G4HLIW32.DLL" (ByVal COPID As
Integer, ByRef StateReg As Integer, ByRef ParameterReg As Integer) As Integer

#End If


' ------------------------------------------------------------------**
'                     END OF FILE                                    **
' ------------------------------------------------------------------**
```

```
'*****************************************************************************************
'*
'*       ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
'*       DEPARTAMENTO DE ENGENHARIA MECÂNICA
'*
'*****************************************************************************************
'*
'*       PROJETO DE PLATAFORMA MÓVEL COM 4 GRAUS DE LIBERDADE
'*       BASEADA EM PLATAORMA DE STEWART
'*
'*       Autores:
'*           Danielle Morita
'*           Fábio Fukunaga
'*           Maurício Dias Menezes Mazza
'*           Rafael Salla Paulilo
'*
'*       Professor Orientador:
'*           Tarcísio A. Hess Coelho
'*
'*****************************************************************************************
'*
'*       FORMULÁRIO INICIAL
'*       ENTRADA DE DADOS GEOMÉTRICOS DA PLATAFORMA
'*
'*       Formulário: Principal.frm
'*
'*****************************************************************************************

Public banco_de_dados As String
Public Tipo_db As Integer

'*****************************************************************************************
'*
'*       Função Bt_Gera_Click : Inicia Formulário de Entrada de Dados
'*
'*****************************************************************************************

Private Sub Bt_Gera_Click()

    Tipo_db = 0
    Ent_Dados.Show

End Sub

'*****************************************************************************************
'*
'*       Função Bt_Load_Click : Carrega um banco de dados do disco
'*
'*****************************************************************************************

Private Sub Bt_Load_Click()

    Tipo_db = 1

    CommonDialog1.Filter = "Banco de Dados (*.mdb) |*.mdb| All Files (*.*)|*.*|"
    CommonDialog1.ShowOpen

    If CommonDialog1.Filename <> "" Then

        banco_de_dados = CommonDialog1.Filename
        Deslocamentos.Show

    End If

End Sub
```

```
'*****************************************************************************************
'*
'*      ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
'*      DEPARTAMENTO DE ENGENHARIA MECÂNICA
'*
'*****************************************************************************************
'*
'*      PROJETO DE PLATAFORMA MÓVEL COM 4 GRAUS DE LIBERDADE
'*      BASEADA EM PLATAORMA DE STEWART
'*
'*      Autores:
'*          Danielle Morita
'*          Fábio Fukunaga
'*          Maurício Dias Menezes Mazza
'*          Rafael Salla Paulilo
'*
'*      Professor Orientador:
'*          Tarcísio A. Hess Coelho
'*
'*****************************************************************************************
'*
'*      FORMULÁRIO DE COMUNICAÇÃO DE DADOS COM A PLACA DE CONTROLE
'*
'*      Formulário: Rodar.frm
'*
'*****************************************************************************************

Dim Tabela(1 To 100) As Integer
Dim I As Integer

Private Sub Command1_Click()

    Var1 = IBS_HLI_Init_SC1()
    Command2.Enabled = True
    Command1.Enabled = False
    Command4.Enabled = True

End Sub

Private Sub Command2_Click()

    Command1.Enabled = True
    Command2.Enabled = False
    Command4.Enabled = False
    Var2 = IBS_HLI_Exit_SC1()

End Sub

Private Sub Command3_Click()

    Var4 = MsgBox("Deseja sair da aplicação?", vbYesNo, "Confirmação")
    If Var4 = vbYes Then
    End
    End If

End Sub

Private Sub Command4_Click()

    Timer1.Enabled = True
    Command4.Enabled = False
    Command2.Enabled = False
    Command5.Enabled = True

End Sub

Private Sub Command5_Click()

    Data1.Recordset.MoveFirst

End Sub

Private Sub Command6_Click()

    Timer1.Enabled = False
```

```
    Var6 = IBS_HLI_Init_SC1()
    Var7 = IBS_HLI_Exit_SC1()
    Command1.Enabled = True
    Command2.Enabled = False
    Command4.Enabled = False

End Sub

Private Sub Form_Load()

    If Principal.Tipo_db = 0 Then
        Data1.DatabaseName = Intervalo.movimento.Name
        Data1.RecordSource = "tensao"
        Data1.Refresh
    Else
        Data1.DatabaseName = Principal.banco_de_dados
        Data1.RecordSource = "tensao"
        Data1.Refresh
    End If

    Timer1.Interval = Intervalo.TB_Tempo * 1000
    Command2.Enabled = False
    Command4.Enabled = False
    Command5.Enabled = False

End Sub

Private Sub Timer1_Timer()

    IBS_HLI_Process_SC1
    If Data1.Recordset.EOF Then
        Var5 = MsgBox("Processamento executado com sucesso!", vbOKOnly, "Aviso")
        Command2.Enabled = True
        Command4.Enabled = True
        Timer1.Enabled = False
    Else
        AO1 = Data1.Recordset(1) * 3276.7
        AO2 = Data1.Recordset(2) * 3276.7
        AO3 = Data1.Recordset(3) * 3276.7
        AO4 = Data1.Recordset(4) * 3276.7
        Data1.Recordset.MoveNext
    End If

End Sub
```

```
'*****************************************************************************************
'*
'*      ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
'*      DEPARTAMENTO DE ENGENHARIA MECÂNICA
'*
'*****************************************************************************************
'*
'*      PROJETO DE PLATAFORMA MÓVEL COM 4 GRAUS DE LIBERDADE
'*      BASEADA EM PLATAORMA DE STEWART
'*
'*      Autores:
'*          Danielle Morita
'*          Fábio Fukunaga
'*          Maurício Dias Menezes Mazza
'*          Rafael Salla Paulilo
'*
'*      Professor Orientador:
'*          Tarcísio A. Hess Coelho
'*
'*****************************************************************************************
'*
'*      FORMULÁRIO DE APRESENTAÇÃO GRÁFICA E SIMULAÇÃO DO MOVIMENTO
'*
'*      Formulário: Simula.frm
'*
'*****************************************************************************************

'Tabela auxiliar de delocamentos
Dim espaco As Workspace
Dim banco As Database
Dim tabela As Recordset

'Variáveis dos tempos máximo e de amostragem
Dim tempo As Double
Dim maximo As Double

'Variáveis da posição instantâneas dos atuadores
Dim Ha As Double
Dim Hb As Double
Dim Hc As Double
Dim Hd As Double

'*****************************************************************************************
'*
'*      Função Bt_Play_Click : Inicia simulação dos atuadores
'*
'*****************************************************************************************

Private Sub Bt_Play_Click()

Dim index As Integer

    Load Simula
    Timer1.Enabled = True

End Sub

'*****************************************************************************************
'*
'*      Função Form_load : "Reseta" os Charts do Formulário
'*                         Define Intervalo do Timer
'*                         Define número de colunas dos Charts
'*                         Define o valor inicial das Progress Bars
'*
'*****************************************************************************************

Private Sub Form_Load()

    Set espaco = CreateWorkspace("espaco", "admin", "")

    If Principal.Tipo_db = 0 Then

        Set banco = espaco.OpenDatabase(Intervalo.movimento.Name, vbReadOnly)
        Set tabela = banco.OpenRecordset("deslocamento")
```

```
    Else:

        Set banco = espaco.OpenDatabase(Principal.banco_de_dados, vbReadOnly)
        Set tabela = banco.OpenRecordset("deslocamento")

    End If

    tabela.MoveFirst

    tempo = acho_intervalo(tabela)
    maximo = acho_maximo(tabela)

    Timer1.Interval = tempo * 1000

    MSChart1.RowCount = maximo / tempo
    MSChart2.RowCount = maximo / tempo
    MSChart3.RowCount = maximo / tempo
    MSChart4.RowCount = maximo / tempo

    Pb_Ha.Value = 300
    Pb_Hb.Value = 300
    Pb_Hc.Value = 300
    Pb_Hd.Value = 300

End Sub

'*****************************************************************************************
'*
'*      Função Timer1_Timer : Preenche os Charts e as P.Bars com
'*                            os dados da tabela de deslocamentos
'*
'*****************************************************************************************

Private Sub Timer1_Timer()

'Variável que detecta fim do preenchimento
Dim action As Integer

    If MSChart1.Row < MSChart1.RowCount Then

        MSChart1.Data = Ha + tabela!atuador_a
        MSChart2.Data = Hb + tabela!atuador_b
        MSChart3.Data = Hc + tabela!atuador_c
        MSChart4.Data = Hd + tabela!atuador_d

        Ha = MSChart1.Data
        Hb = MSChart2.Data
        Hc = MSChart3.Data
        Hd = MSChart4.Data

        Pb_Ha.Value = Pb_Ha.Value + tabela!atuador_a
        Pb_Hb.Value = Pb_Hb.Value + tabela!atuador_b
        Pb_Hc.Value = Pb_Hc.Value + tabela!atuador_c
        Pb_Hd.Value = Pb_Hd.Value + tabela!atuador_d

        MSChart1.Row = MSChart1.Row + 1
        MSChart2.Row = MSChart2.Row + 1
        MSChart3.Row = MSChart3.Row + 1
        MSChart4.Row = MSChart4.Row + 1

        Label6.Caption = tabela!tempo

        action = 0

    Else:

        Timer1.Enabled = False
        Bt_Play.Enabled = True

        Label6.Caption = 0

        action = 1

    End If
```

```
    If action = 0 Then tabela.MoveNext
    If action = 1 Then
        tabela.MoveFirst
        aceion = 0
    End If

End Sub
```

APÊNDICE II - CÓDIGO DOS MÓDULOS DO SOFTWARE

```
'*****************************************************************************************
'*
'*       ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
'*       DEPARTAMENTO DE ENGENHARIA MECÂNICA
'*
'*****************************************************************************************
'*
'*       PROJETO DE PLATAFORMA MÓVEL COM 4 GRAUS DE LIBERDADE
'*       BASEADA EM PLATAORMA DE STEWART
'*
'*       Autores:
'*           Danielle Morita
'*           Fábio Fukunaga
'*           Maurício Dias Menezes Mazza
'*           Rafael Salla Paulilo
'*
'*       Professor Orientador:
'*           Tarcísio A. Hess Coelho
'*
'*****************************************************************************************
'*
'*       MÓDULO DE APRESENTAÇÃO GRÁFICA DOS RESULTADOS
'*
'*       Módulo: Grafico.bas
'*
'*****************************************************************************************

'*****************************************************************************************
'*
'*       Função Reset_Chart    : Zera todos os valores do objeto Chart
'*
'*       Parâmetros: - Grafico : Objeto Chart a ser "resetado"
'*                   - relogio : Objeto Timer
'*
'*****************************************************************************************

Public Function Reset_Chart(Grafico As MSChart, relogio As Timer) As Integer

Dim index As Integer

    Grafico.RowCount = relogio.Interval

    For index = 0 To Grafico.RowCount

        Grafico.Data = 0
        If Grafico.Row < Grafico.RowCount Then Grafico.Row = Grafico.Row + 1

    Next index

    Grafico.Row = 1

    Reset_Chart = 0

End Function

'*****************************************************************************************
'*
'*       Função acho_intervalo : Busca na tabela o tempo de amostragem
'*
'*       Parâmetros: - tabela  : Tabela dos deslocamentos
'*
'*****************************************************************************************

Public Function acho_intervalo(tabela As Recordset) As Double

    tabela.MoveNext
    acho_intervalo = tabela!tempo
    tabela.MoveFirst

End Function
```

```
'*****************************************************************************************
'*
'*       Função acho_maximo     : Busca na tabela o tempo maximo do movimento
'*
'*       Parâmetros: - tabela   : Tabela dos deslocamentos
'*
'*****************************************************************************************

Public Function acho_maximo(tabela As Recordset) As Double

    tabela.MoveLast
    acho_maximo = tabela!tempo
    tabela.MoveFirst

End Function
```

```
'*****************************************************************************************
'*
'*      ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
'*      DEPARTAMENTO DE ENGENHARIA MECÂNICA
'*
'*****************************************************************************************
'*
'*      PROJETO DE PLATAFORMA MÓVEL COM 4 GRAUS DE LIBERDADE
'*      BASEADA EM PLATAORMA DE STEWART
'*
'*      Autores:
'*          Danielle Morita
'*          Fábio Fukunaga
'*          Mauricio Dias Menezes Mazza
'*          Rafael Salla Paulilo
'*
'*      Professor Orientador:
'*          Tarcísio A. Hess Coelho
'*
'*****************************************************************************************
'*
'*      MÓDULO DE OPERAÇÕES E FUNÇÕES MATEMÁTICAS UTILIZADAS
'*      DURANTE O PROCESSAMENTO DOS DADOS DE ENTRADA
'*
'*      Módulo: Operações.bas
'*
'*****************************************************************************************

'*****************************************************************************************
'*
'*      Função Calcula_Tempo : Calcula o tempo gasto em cada movimento
'*
'*      Parâmetros: - desloc : Valor do Deslocamento linear/angular
'*                  - vmax   : Valor da Velocidade Máxima linear/angular
'*                  - amax   : Valor da Aceleração Máxima linear/angular
'*
'*****************************************************************************************


Public Function Calcula_Tempo(desloc As Double, vmax As Double, amax As Double) As
Double

    If (vmax ^ 2 / amax) > desloc Then
        Calcula_Tempo = Sqr(4 * desloc / amax)
    Else
        Calcula_Tempo = desloc / vmax + vmax / amax
    End If

End Function

'*****************************************************************************************
'*
'*      Função Calcula_v     : Calcula o valo rda velocidade máxima real para
'*                             cada movimento
'*
'*      Parâmetros: - desloc : Valor do Deslocamento linear/angular
'*                  - vmax   : Valor da Velocidade Máxima linear/angular
'*                  - amax   : Valor da Aceleração Máxima linear/angular
'*                  - t      : Tempo total do movimento
'*
'*****************************************************************************************

Public Function Calcula_v(desloc As Double, vmax As Double, amax As Double, t As Double)
As Double

    If (vmax ^ 2 / amax) > desloc Then
        Calcula_v = amax * t / 2
    Else
        Calcula_v = vmax
    End If

End Function
```

```
'*****************************************************************************************
'*
'*      Função Calcula_tmax : Calcula o máximo dos três tempos abaixo
'*
'*      Parâmetros: - t1    : Tempo gasto para completar o deslocamento linear "d"
'*                  - t2    : Tempo gasto para completar o deslocamento angular "teta x"
'*                  - t3    : Tempo gasto para completar o deslocamento angular "teta y"
'*
'*****************************************************************************************

Public Function Calcula_tmax(t1 As Double, t2 As Double, t3 As Double) As Double

    Calcula_tmax = t1
    If Calcula_tmax < t2 Then Calcula_tmax = t2
    If Calcula_tmax < t3 Then Calcula_tmax = t3

End Function

'*****************************************************************************************
'*
'*      Função corrige    : Efetua a correção da velocidade/aceleração dos movimentos
'*                          mais rápidos em relação ao mais lento
'*
'*      Parâmetros: - t1  : Tempo gasto pelo movimento atual
'*                  - t2  : Tempo gasto pelo movimento mais lento
'*                  - exp : Expoente da relação "t1/t2" (1-velocidade/2-aceleração)
'*
'*****************************************************************************************

Public Function corrige(t1 As Double, t2 As Double, exp As Integer, orig As Double) As
Double

    corrige = orig * (t1 / t2) ^ exp

End Function

'*****************************************************************************************
'*
'*      Função deslocamento   : Calcula deslocamento linear/angular no instante atual
'*
'*      Parâmetros: - t_atual : Tempo no instante atual
'*                  - t_max   : Tempo máximo (movimento mais lento)
'*                  - v       : Velocidade Corrigida
'*                  - a       : Aceleração Corrigida
'*
'*****************************************************************************************

Public Function deslocamento(t_atual As Double, t_max As Double, v As Double, a As
Double)

Dim tb As Double
Dim ta As Double

    ta = v / a
    tb = t_max - ta
    If t_atual <= ta Then deslocamento = a * t_atual ^ 2 / 2
    If t_atual >= tb Then deslocamento = v * (t_atual - ta / 2) - a * (t_atual - tb) ^ 2
/ 2
    If t_atual > ta And t_atual < tb Then deslocamento = v * (t_atual - ta / 2)

End Function

'*****************************************************************************************
'*
'*      Função Atuador     : Calcula o deslocamento linear de um determinado atuador
'*
'*      Parâmetros: - Xp   : Coordenada "X" do ponto de ligação placa/barra do atuador
'*                  - Yp   : Coordenada "Y" do ponto de ligação placa/barra do atuador
'*                  - Zp   : Coordenada "Z" do ponto de ligação placa/barra do atuador
'*                  - Xi   : Coordenada "X" da posição inicial da atuador
'*                  - Yi   : Coordenada "Y" da posição inicial da atuador
'*                  - Zi   : Coordenada "Z" da posição inicial da atuador
'*                  - comp : Comprimento da barra que liga o atuador à placa
'*
'*****************************************************************************************
```

```
Public Function Atuador(Xp As Double, Yp As Double, Zp As Double, Xi As Double, Yi As
Double, Zi As Double, comp As Double) As Double

    Atuador = Zp - Zi - Sqr(comp ^ 2 - (Xp - Xi) ^ 2 - (Yp - Yi) ^ 2)

End Function
```

APÊNDICE III

IBS PC ISA SC/I-T

Data Sheet

INTERBUS

Controller Board for PC Systems

Data Sheet Revision B

03/1997

Product Description

INTERBUS Generation 4 controller board for IBM-compatible PCs with integrated ISA bus (8/16-bit slot).

Features

- INTERBUS protocol (DIN E 19258)
- Complete Generation 4 functionality:
  - Up to 255 bus segments
  - Up to 16 device levels
  - Up to 512 devices per configuration
  - Up to 4096 I/O points per configuration
  - Up to 62 devices with parameter channel
  - INTERBUS Loop support
  - CMD G4 support
- Logical addressing
- Data preprocessing on the controller board
- Firmware bootable via RS232
- PCP 2.x
- Electrical isolation between remote bus connector and host PC
- Driver software for DOS, Windows 95, Windows NT

Applications

Direct connection of simple sensors/actuators up to intelligent field devices to the PC via INTERBUS.

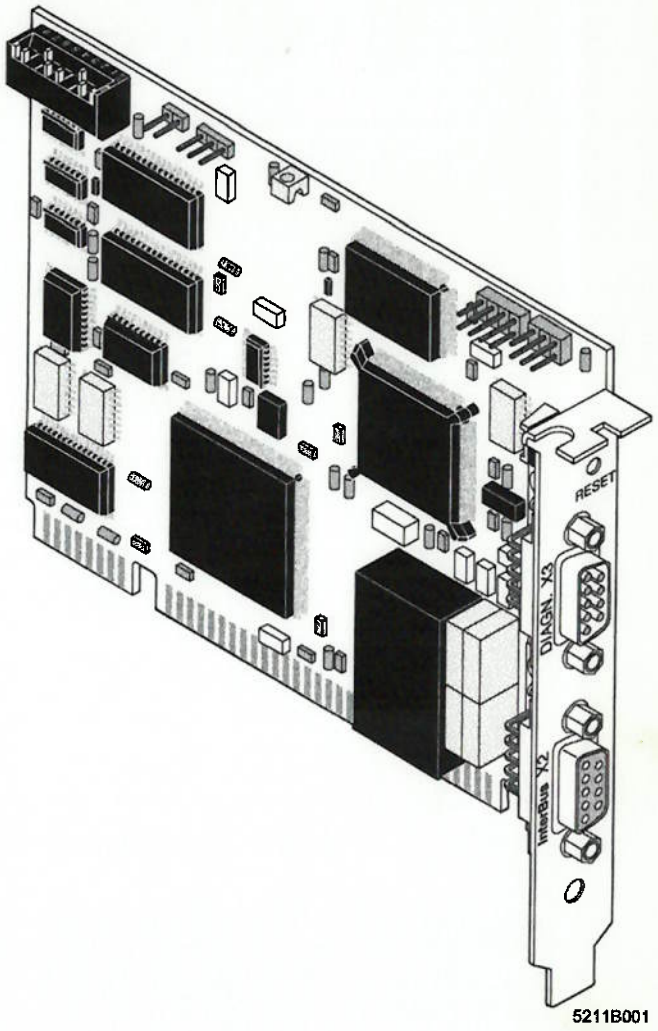


Figure 1: IBS PC ISA SC/I-T

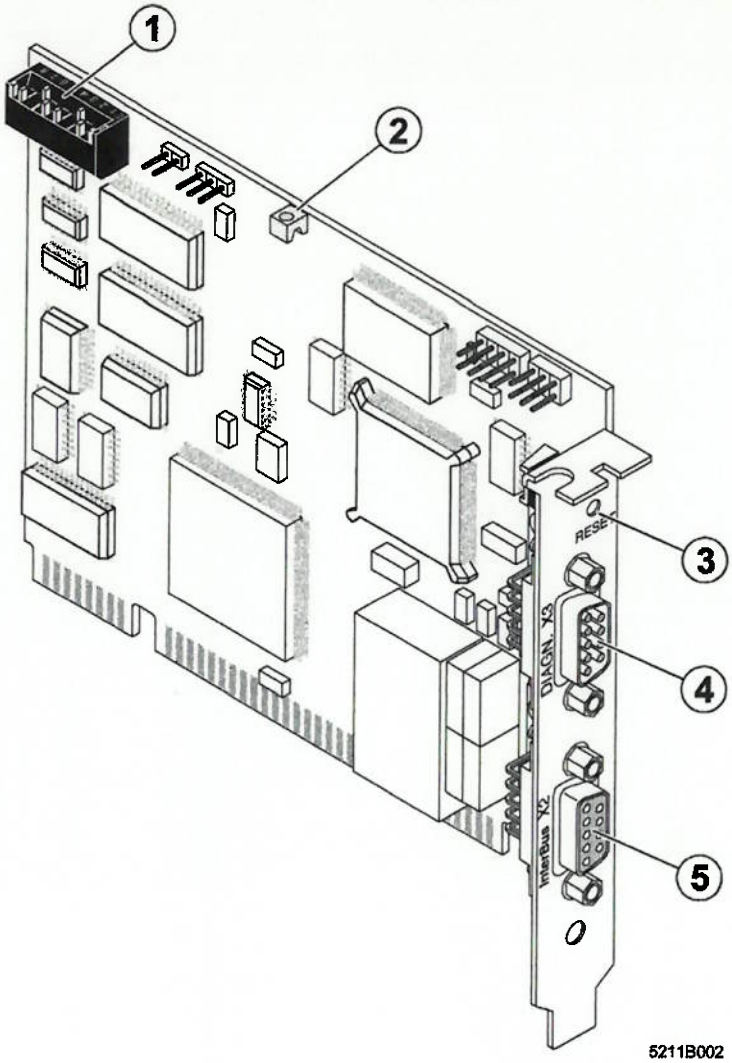


Figure 2: Design of the IBS PC ISA SC/I-T controller board

The controller board has the following components:

1 DIP switches for setting the board address
2 *Board Ready* diagnostic LED (green)
3 Reset pushbutton
4 Diagnostic interface (9-pos. SUB-D male connector)
5 INTERBUS remote bus interface (9-pos. SUB-D female connector)

Functional Units of the Controller Board

ISA-AT Bus Interface

The controller board has been designed as a 16-bit bus. Due to the mechanical structure the 16-bit expansion slot only uses interrupts 10, 11, 12 and 15. Therefore the board can also be operated in an 8-bit slot.

Remote Bus Interface

The remote bus interface connects the controller board with the INTERBUS devices (field devices). The remote bus interface has been designed as a 9-pos. D-SUB female connector on the front plate of the controller board. Phoenix Contact offers preassembled remote bus cables in common lengths. The design of the remote bus cable is shown in the figure below.

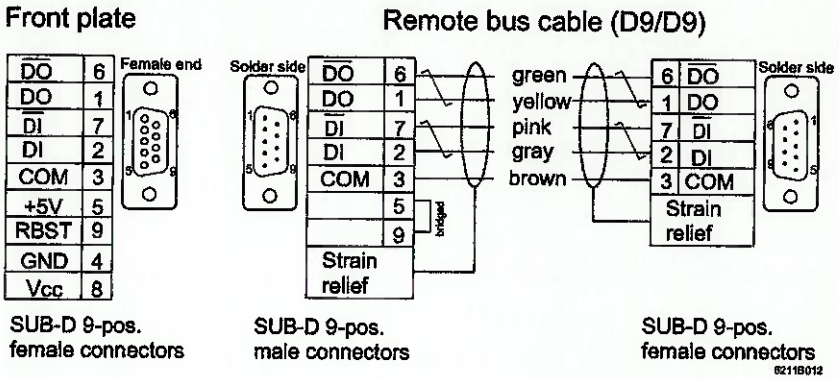


Figure 3: Remote bus interface and example for a remote bus cable (cable type D9/D9)

In addition, the IBS OPTOSUB-MA/M/L-LK interface connector (Order No. 27 50 11 2) allows you to fit your INTRBUS system with optical fibers. For different mounting positions the outputs of the interface connector can be on the right or left.

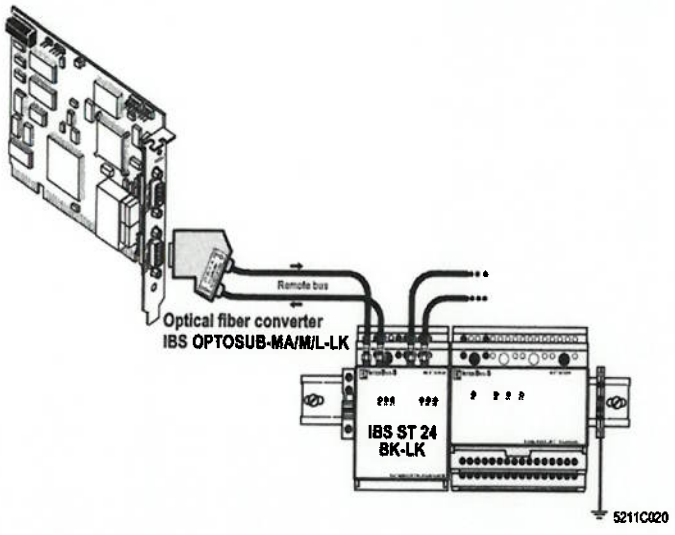


Figure 4: INTERBUS system with optical fibers

Diagnostic Interface (serial)

The controller board's serial interface (9-pos. D-SUB male connector) on the front plate connects an IBM-compatible PC with the *IBS CMD SWT G4 E* software (Order No.: 27 21 44 2). The controller board is connected to the PC via the *IBS PRG CAB* cable (Order No.: 28 06 86 2) shown below.

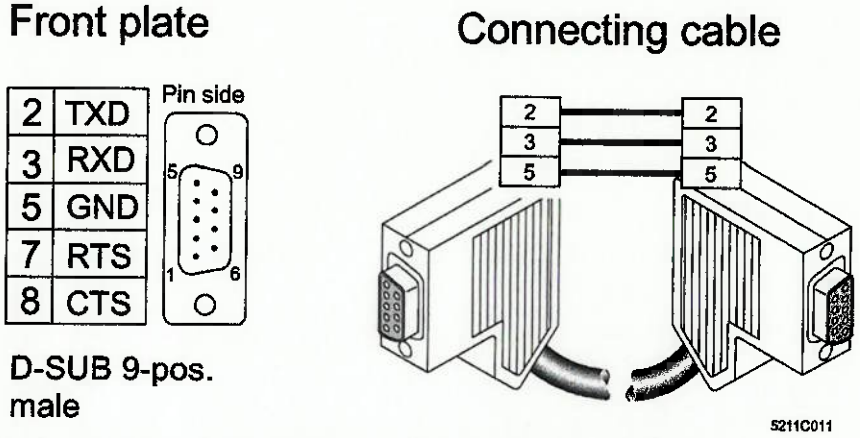


Figure 5: Diagnostic interface and connecting cable for PC connection

This software can be used to configure, parameterize and diagnose the INTERBUS system. With IBS CMD SWT G4, parameterization and configuration information can be stored on the controller board. This information, however, will be lost after the PC and controller board have been reset or switched off.
Via the serial interface (RS232) the firmware of the controller board can be downloaded. In this way it is possible to update the system to meet future demands.

DIP Switches

The DIP switches for setting the board address are located at the top of the board and in such a way that they can be set when the board has been inserted in the PC. The start address can be set to any value divisible by 8 in the range between 100_{hex} and $3F8_{hex}$. The default value is 120_{hex}. Eight subsequent addresses of the I/O channels are used.

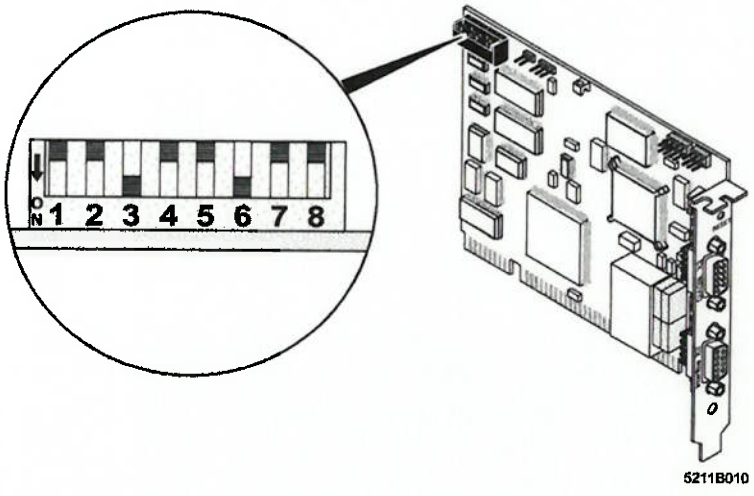


Figure 6: Default setting of the DIP switches (I/O address 120_{hex})

Table 1: Setting the DIP switches

DIP switch	2	3	4	5	6	7	8	
Value	200_{hex}	100_{hex}	80_{hex}	40_{hex}	20_{hex}	10_{hex}	8_{hex}	
I/O address								
100	OFF	ON	OFF	OFF	OFF	OFF	OFF	
108	OFF	ON	OFF	OFF	OFF	OFF	ON	
110	OFF	ON	OFF	OFF	OFF	ON	OFF	
118	OFF	ON	OFF	OFF	OFF	ON	ON	
120	**OFF**	**ON**	**OFF**	**OFF**	**ON**	**OFF**	**OFF**	**Default**
128	OFF	ON	OFF	OFF	ON	OFF	ON	
...	...	...	...	...	...	...	...	
...	...	...	...	...	...	...	...	
3F8	ON	ON	ON	ON	ON	ON	ON	

DIP switch 1 (test mode) is not used to set the address but the test mode. When the controller board is restarted with the test mode switched on, it starts the INTERBUS system with physical addressing and initializes it. During test mode the controller board does not respond to instructions of the host system (PC). Any parameter data stored on the controller board is not read in. The controller board initializes and starts up the INTERBUS system independently. No outputs are set.

The test mode is not intended for regular operation of the controller board. Set DIP switch 1 to OFF to deactivate the test mode.

Address Area Assignment of PC I/O Ports

The controller board assigns eight bytes within the I/O address area, five of them are assigned by registers. Only drivers use these registers.

Table 2: I/O address assignment

Offset	Description	Function
0	WIN_ADR_PC	Window address and enabling of host PC side
1	WIN_ADR_MPM	Window address and MPM side
2	IRQ_Control	IRQ selection and enabling
3	WDT_Control	Status, enabling and triggering of the watchdog
4	MA_Reset_Control	Reset pushbutton enabling and software reset of MA
5	—	Reserved
6	—	Reserved
7	—	Reserved

Diagnostic LED

When the PC is switched on, the basic functions of the processor and the firmware are checked on the controller board. If no errors have been detected, the green LED lights up after approx. 5 sec and the host interface is enabled.

Reset Pushbutton

The reset pushbutton is covered by a mounting sheet and can only be set with a pin, thus protecting it against wrong setting. The reset pushbutton causes the controller board to be reset completely. The PC is not affected by the reset.

Watchdog for Monitoring the Host

A watchdog circuit for monitoring the PC program ("system crash", "program hang-up") is located on the motherboard of the controller board. If the watchdog becomes active, the INTERBUS system is placed into a defined state (reset of all outputs).

The watchdog does not influence the host, which means that, for example, no host reset is carried out!

If the watchdog should be used, it must be enabled via the application program. The watchdog is disabled by default. After having enabled the watchdog, it cannot be disabled via the software. It can only be disabled by switching off the host or by a hardware reset.

The watchdog can be set to different times. The watchdog must be triggered by calling the *TriggerWatchDog ()* function in the application program within the time selected. Otherwise, the watchdog causes a reset of the INTERBUS system. Table 3 shows the different monitoring times.

Table 3: Possible watchdog monitoring times

Monitoring time
8.2 ms
16.4 ms
32.8 ms
65.5 ms
131.1 ms
262.1 ms
524.3 ms
1048.6 ms

It is not recommended to use the watchdog for host monitoring under Windows 3.1x. It is not ensured that your program allows watchdog triggering within the determined time. Example: Shift the frame of a window with your mouse. As long as you are holding on to the frame of the window with the mouse, the program will not continue to run.

SysFail Signal

For each MPM accessor, a special area is reserved within the MPM which, for example, is used for status messages or handshaking signals. The *SysFail* signal (system failure) - one of the status signals - is set if a system failure of the respective accessor occurs, e.g. if its watchdog has initiated a reset. The *GetSysFailRegister* allows to read out the *SysFail* signal of any MPM accessor.

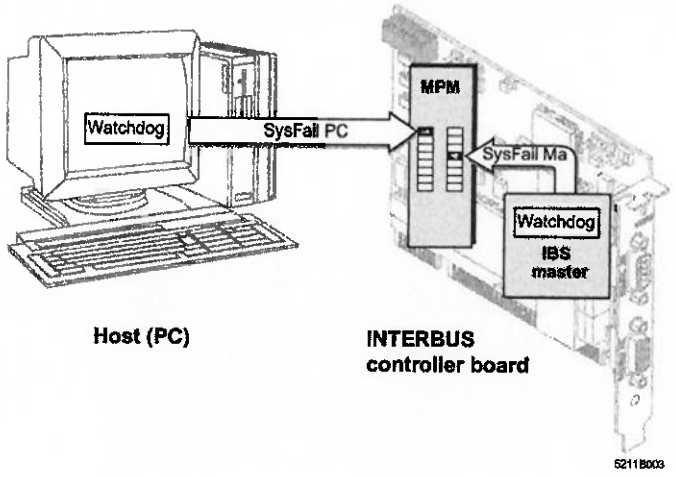


Figure 7: The *SysFail* signal in the MPM

Multi Port Memory (MPM)

The 64 kbyte MPM is a static RAM (SRAM) and exchanges data between the host PC and the controller board. The MPM is mapped into the host address area (between 640 kbytes and 1 MB). Since the address area is limited, a 4 kbyte memory window is used. The device driver shifts this window automatically to the required address area of the MPM. The window address in the host memory area remains constant!
Each address that is in the area between 80000_{hex} and $FF000_{hex}$ and is divisible by 4096 can be set by the software as a window address. The window is switched off after a reset of the PC and must be enabled by the driver software first.

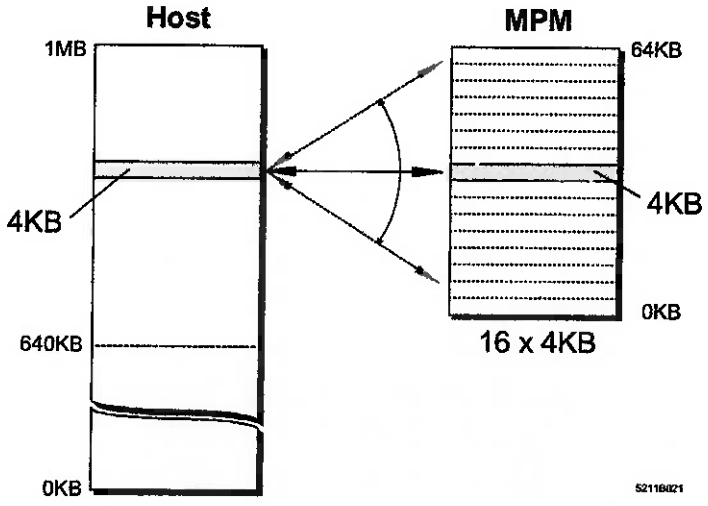


Figure 8: Shifting the MPM window by means of the device driver

Programming/Driver Software

The generation of own applications is supported by appropriate drivers (Device Drivers). These drivers are available for standard operating systems and programming languages. The drivers read from and write to the I/O addresses.

Table 4: Driver software

Operating system	Driver type	Installation
DOS	TSR Program	Links the driver to *autoexec.bat* and protects the MPM window against a memory manager access by means of an appropriate entry in the *config.sys*.
Windows® 3.1x	Dynamic Link Library (DLL)	Creating key words and entering the board parameters into the *ibddiwin.ini* file
Windows® 95	Virtual Device Driver (VxD)	Determines key words and enters board parameters in the register data bank.
Windows® NT	Kernel Mode Driver	

The operating system-specific installation program controls the system settings of the PC (available address area, etc.) and sets up the board for operation.

A detailed description of the registers and MPM is given in the *Device Driver Development Kit* (on request) for programming your own driver software.

Basic Functions of the DOS Driver (Device Driver Interface, DDI)

Table 5: Overview of the DDI functions for DOS

Function	Task
DDI_DevOpenNode	Opens a data channel to a node
DDI_DevCloseNode	Closes a data channel to a node
DDI_MXI_SndMessage	Writes a message to the MPM
DDI_MXI_RcvMessage	Reads a message from the MPM
DDI_DTI_ReadData	Reads data from the MPM
DDI_DTI_WriteData	Writes data to the MPM

Table 6: Overview of hardware control functions

Function	Task
GetSysFailRegister	Reads out the contents of the SysFail register
EnableWatchDog	Enables a watchdog
TriggerWatchDog	Triggers a watchdog
GetWatchDogState	Reads out the status of a watchdog
ClearWatchDog	Resets the state of a watchdog
SetWatchDogTimeout	Sets the timeout value and enables the watchdog
GetWatchDogTimeout	Reads out the timeout value
GetIBSDiagnostic	Evaluates the operating state of the INTERBUS master board

Technical Data

General Data

Designation	IBS PC ISA SC/I-T
Order no.	27 19 23 4
Dimensions	135 mm x 107 mm

Power Supply

$V_{S,controller\ board}$ (PC supply)	5 V DC ±5%
Current consumption	0.5 A, typically

Host Interfaces

Connection method	ISA BUS (8/16 bit slot)
Bus system	ISA acc. to IEEE P966
Data width	8 bits
Address area	8 addresses in the I/O channel; 4 Kbytes in the memory address area
Interrupt	Selectable between IRQ 3, 5, 7, 9(2), 10, 11, 12, 15

Remote Bus Interface

Connection method	9-pos. SUB-D female connector
Type of interface	RS-422
Electrical isolation	Yes (test voltage 0.5 kV)

Diagnostic Interface

Connection method	10-pos. DIL pin strip, connecting line to 9-pos. D-SUB male connector
Type of interface	RS-232
Transmission rate	9600 Baud
Electrical isolation	No

INTERBUS Data

Number of devices	512 (256 remote bus devices/bus segments)
Number of process data	256 words (4096 digital inputs/outputs)
Number of PCP devices	62

Ambient Conditions

Temperature (acc. to EN 60 204-1)	Operation: 0°C to 70°C storage and transport: -25°C to 85°C
Humidity (acc. to EN 60 204-1)	Storage and operation: 75% on average, 85% occasionally (DIN 40040); no condensation
Air pressure	Operation: 860 hPa to 1080 hPa (up to 1500 m above sea level) storage and transportt:660 hPa to 1080 hPa (up to 3500 m above sea level)

Conformance with the EMC Directive 89/336/EEC

Noise Immunity Test acc. to EN 50082-2

Electrostatic discharge (ESD)	EN 61000-4-2/ IEC 1000-4-2	Criterion B 4 kV contact discharge 8 kV air discharge
Electromagnetic fields	ENV 50140/ IEC 1000-4-3	Criterion A Field strength: 10 V/m

Fast transients (Bursts)	EN 61000-4-4/ IEC 1000-4-4	Criterion B Supply lines: 4 kV Signal/data lines: 2 kV
Conducted interference	ENV 50141 IEC 1000-4-6	Criterion A Test voltage10 V

Noise Emission Test acc. to EN 55011

Emission of housing	EN 55011	Class A, group 1

Other tests in the engineering test:

Climatic test

Test of the insulation distances acc. to VDE 0160

Text of the air and creepage distances acc. to VDE 0110/0160

Mechanical tests

Keeping the degree of protection	IEC 529/ EN 60529	
Vibration test	IEC 68-2-6	Shock acceleration: 1.5 g, criterion 1 acc. to IEC 68-2-6
Shock test	IEC 68-2-27	Maximum acceleration: 30 g Shock duration: 18 ms
Drop and toggle	DIN IEC 68-2-31	
Free fall	DIN IEC 68-2-32	

Table 7: Ordering data

Description	Designation	Order No.
System kit with controller board, incl. driver software, User Manual, diagnostic cable and CMD software	IBS PC ISA SC SYSKIT E	27 21 91 8
Controller board	IBS PC ISA SC/I-T	27 19 23 4
User Manual for controller board incl. the driver software for DOS, Win 3.1x, Win 95 Win NT	IBS PC ISA SC UM E	27 47 52 3
INTERBUS Project Planning Manual	IBS SYS PRO UM E	27 51 00 1
INTERBUS Installation Manual	IBS SYS INST UM E	27 54 80 4
CMD software	IBS CMD SWT G4 E	27 21 44 2
Conversion of the INTERBUS remote bus interface to fiber optics technology	IBS OPTOSUB-MA/M/L-LK	27 50 11 2

Phoenix Contact GmbH & Co.
Flachsmarktstraße 8 - 28
32825 Blomberg
Germany
☎: + 49 - (0) 5235 - 3-00
Fax: + 49 - (0) 5235 - 3-41200

Notes:

APÊNDICE IV

IBS ST 24 BKM-T

Data Sheet

INTERBUS

Bus Terminal Module

Data Sheet Revision A 12/1995

Product Description

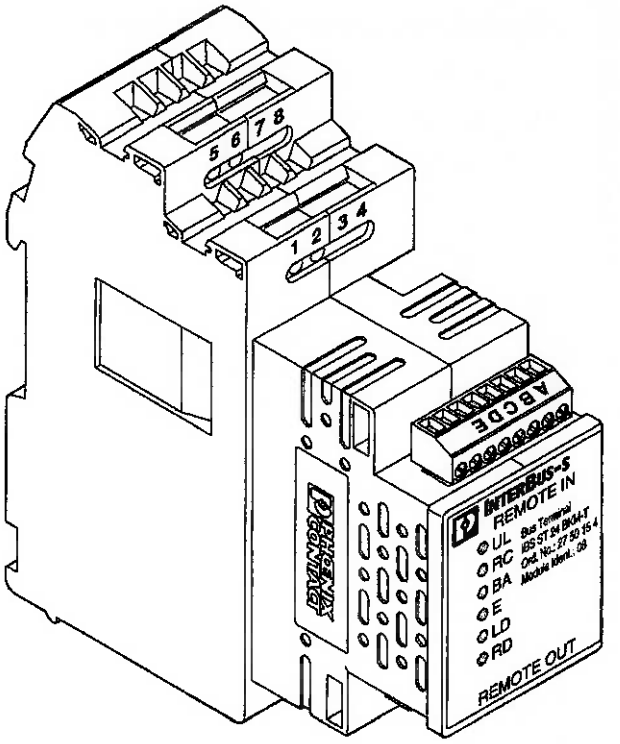
The IBS ST 24 BKM-T module connects an InterBus-ST compact station with an InterBus network.

Features

- InterBus protocol (DIN E 19258)
- 8-pos. MINI-COMBICON remote bus connectors
- Electrical isolation between the remote bus segments
- Electrical isolation between communication (logic) and I/O voltage
- Diagnostic LEDs
- DIN-rail mounting (DIN EN 50022)

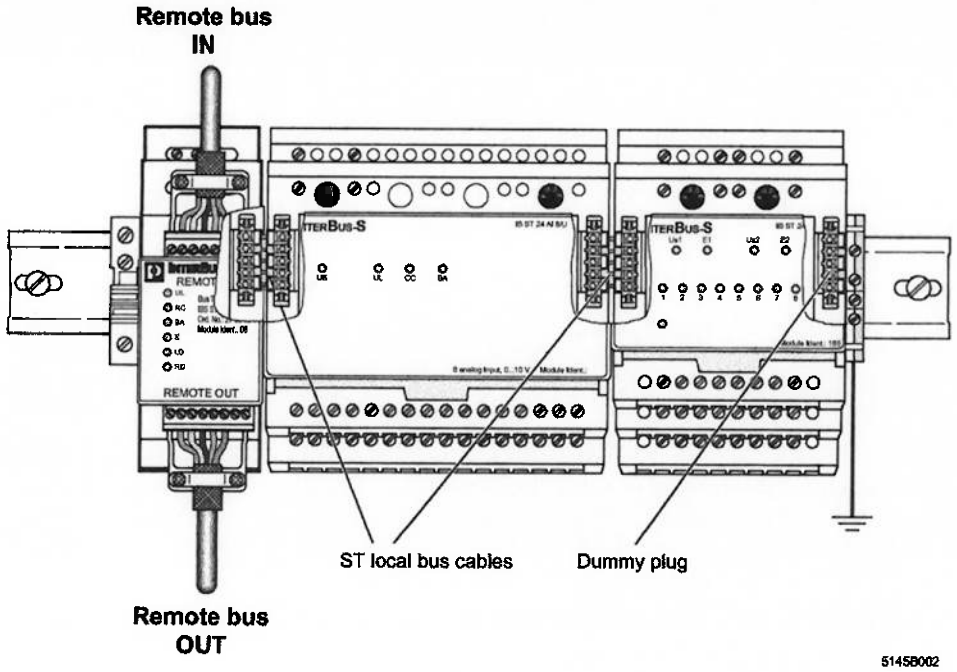
Applications

Connection of an InterBus-ST compact station to the InterBus remote bus.



5145C001

Figure 1: IBS ST 24 BKM-T module



Figuure 2: Example of how to connect an IB ST compact station by means of the IBS ST 24 BKM-T module to the InterBus remote bus. An IB ST compact station may consist of up to 4 I/O modules. The modules are connected via ST local bus cables. The ST cables required come with the I/O module.

Connect the mounting rail via grounding terminals with protective earth (PE), as the modules are grounded when they are snapped onto the rail.

Table 1: Ordering data

Description	Type	Order No.
Bus terminal module	IBS ST 24 BKM-T	27 50 15 4
Zack "quick" strip for terminal labeling	ZB 6 ... see Catalog Part 3 *Marking Mounting*	
Mounting rail DIN EN 50022, 2 meters	NS 35/7,5 gelocht (perfor.) NS 35/7,5 ungelocht (unperforated)	08 01 68 1 08 01 73 3

Connection to the Bus with the Remote Bus Cable

The MINI-COMBICON connectors which come together with the module must be connected to the bus cable. Place the connectors with the keying tabs showing to the front plate of the BK module into the corresponding terminals strips (Figure 3). REMOTE IN accepts the incoming remote bus, REMOTE OUT the outgoing remote bus.

The supply voltage U_L for the electronics module must be supplied via the contacts 1 (+24 V DC) and 5 (GND) of the BK module because it is not supplied over the bus cable (Figure 3).

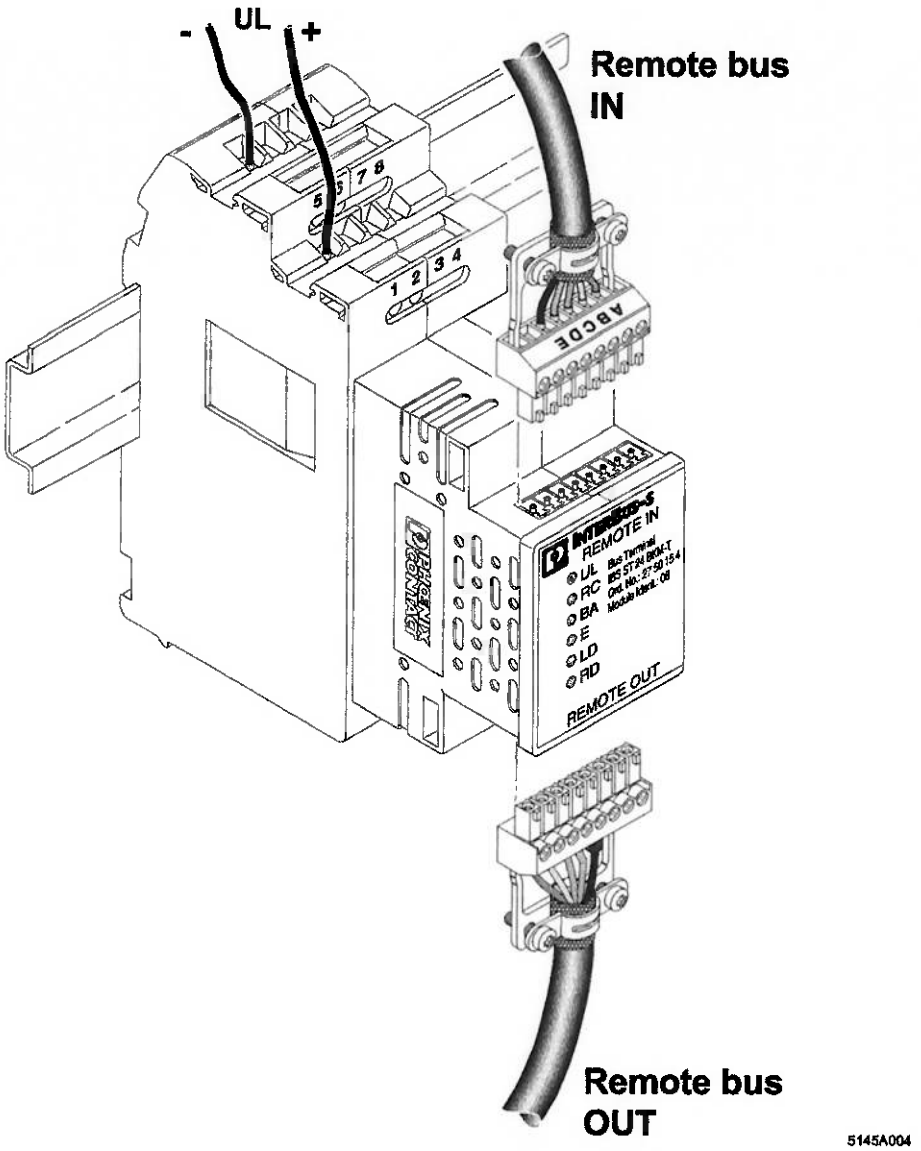


Figure 3: Connection of the bus cables and the supply voltage U_L

Please note that the bridge between the contacts K and L of the outgoing remote bus connector (REMOTE OUT) must be inserted in every module, if further remote bus modules follow. This bridge must not be inserted in the last module.

Connection

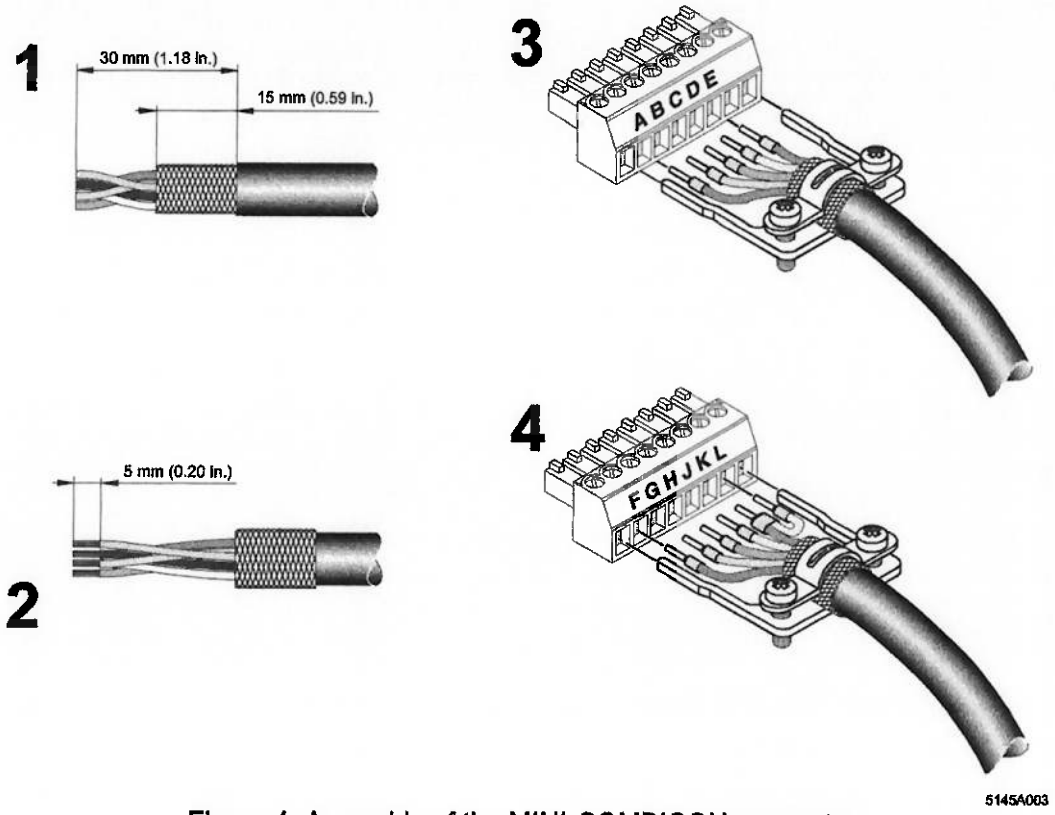


Figure 4: Assembly of the MINI-COMBICON connectors

Assembly of the Bus Cables

- Strip the outer cable sheath approx. 30 mm (4.1).
- Shorten the braided screen to 15 mm and place it around the cable sheath.
- Remove the protective foil (4.2).
- Cut off the white conductor close to the cable sheath, as it is not required.
- Strip the other conductors 5 mm and provide them with ferrules.
- Push the cable through the shield clamp and screw it tight so that as much of the braided screen as possible is clamped underneath.
- Wire the respective connectors acccording to Tables 2 and 3.
- Connect the shield clamp to contacts 1 and 8 of the connector. The clamp ensures a proper cable grip (Figure 4.3).

Table 2: Pin assignment of the REMOTE IN connector (Figure 4.3)

Pin	1	2	3	4	5	6	7	8
Terminal designation		A	B	C	D	E		
Signal description	Shield	$\overline{DO}$	DO	$\overline{DI}$	DI	Ground	Free	Shield
Conductor color		Green	Yellow	Pink	Gray	Brown		

Table 3: Pin assignment of the REMOTE OUT connector (Figure 4.4)

Pin	1	2	3	4	5	6	7	8
Terminal designation		F	G	H	J	K	L	
Signal description	Shield	$\overline{DO}$	DO	$\overline{DI}$	DI	Ground	RBST	Shield
Conductor color		Green	Yellow	Pink	Gray	Brown		

Connecting I/O Modules with ST Local Bus Cables

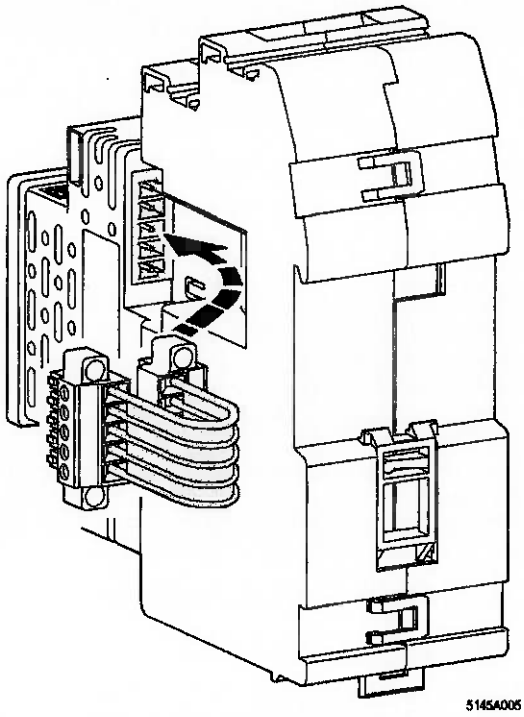


Bild 5: Inserting the ST cables into the BK module for the connection of I/O modules

Please note, that for the BK module the ST cable for the local bus must be inserted **before the module is snapped** onto the mounting rail, since the electronics module is not pluggable as it is the case for other ST components (Figure 5).
In the last module the open IB ST interface must be isolated with the dummy plug. This plug comes with the BK module.

Table 4: The most important data for the programmer

ID code	8 (08_{hex})
Length code	00_{hex}
Input address area in the control or computer system	0 bytes
Output address area in the control or computer system	0 bytes
PCP address area in the control or computer system	0 bytes
Register length on the bus	0 bytes
Programmable functions: - Disconnection of the IB ST compact station - Disconnection of the outgoing remote bus - Reset of the outgoing remote bus - Reset of the IB ST compact station - Monitoring of the remote bus cable	

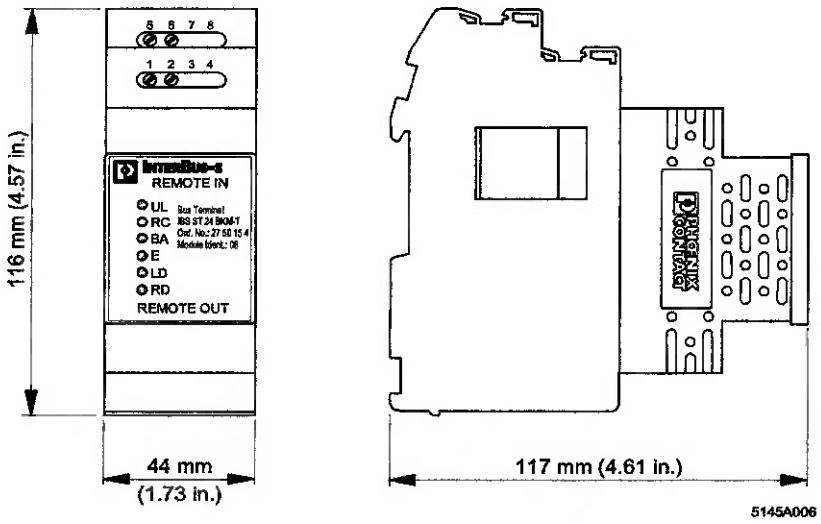


Figure 6: Module dimensions

Technical Data

General

Type	IBS ST 24 BKM-T
Degree of protection	IP 20
Air and creepage distances	VDE 0110 part1, 01/89
Permissible operating temperature	0°C to 55°C
Permissible strorage temperature	-20°C to 70°C
Dimensions (w*h*d)	44 mm * 116 mm * 117 mm

Interfaces

InterBus	
- Incoming remote bus	MINI-COMBICON connector, 8-pos.
- Outgoing remote bus	MINI-COMBICON connector, 8-pos.
IB-ST interface	ST cable (is delivered with the I/O modules)
Number of ST modules that can be connected	4
Maximum distance to the next remote bus device	400 m
Eelctrical isolation between incoming and outgoing remote bus	Test voltage 500 V AC (50 Hz)
Electrical isolation between incoming remote bus and IB ST interface	Test voltage 500 V AC (50 Hz)

Power Supply

Terminal points	1 (+), 5 (-) (see Figure 3)
Nominal voltage US	24 V (DC)
Permissible ripple	3.3 V_{pp} within the permissible voltage range
Permissible voltage range	20 V to 30 V (DC), ripple included
Current comsumption primary	350 mA
Current output secondary (ST cable)	
- Maximum	500 mA (9 V)

Since there is no fuse inside the module, a reversal of the polarity of the input voltage and an input current of > 2 A can damage the electronics module. Therefore you should connect an external fuse of 2 A.

Local Diagnostic Indicators

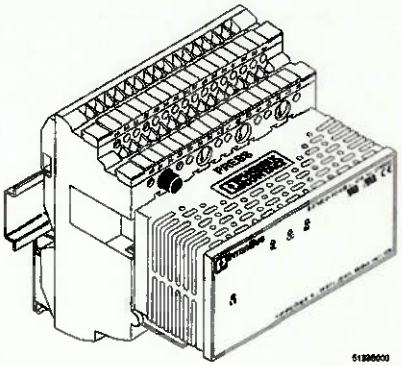
UL	Green LED	Supply voltage for the electronics module
	on:	Supply voltage U_L present
	off:	Supply voltage U_L not present – Internal power pack of the BK module defective
RC	Green LED	Remote bus cable check
	on:	Incoming remote bus cable connected
	off:	Incoming remote bus cable interrupted or defective
BA	Green LED	Remote bus active
	on:	Data transmission on InterBus active
	off:	No data transmission
LD	Red LED	Local bus status
	on:	The I/O modules of this compact station are disabled.
RD	Red LED	Remote bus status
	on:	Outgoing remote bus disabled
E	Red LED	Error
	on:	The controller board located an error in this IB ST compact station.

TNR 92 70 841 DNR 2598

APÊNDICE V

IB ST (ZF) 24 AO 4/SF4

Analog Output Module With 4 Channels



Data Sheet 5133B 01/1999

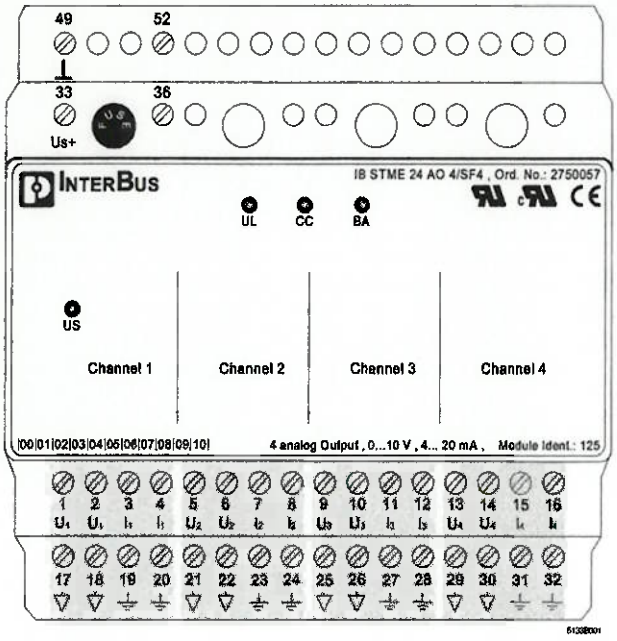


Figure 1 IB ST 24 AO 4/SF4 module

General information pertaining to ST modules can be found in the IBS SYS PRO UM E Manual.

Ground the mounting rail. The module is grounded by snapping it onto the mounting rail.

Terminal Assignment

Terminal	Signal
Us+	24 V I/O supply voltage
⊥	Ground of the supply voltage
U1 to U4	Voltage outputs channels 1 to 4
I1 to I4	Current outputs channels 1 to 4
▽	Analog ground connection
⏚	Ground potential

Local Diagnostic and Status Indicators

Des.	Color	Meaning
UL	Green	Supply voltage for the electronics module
CC	Green	Cable check
BA	Green	Bus active
US	Green	24 V I/O supply voltage

Internal Circuit Diagram

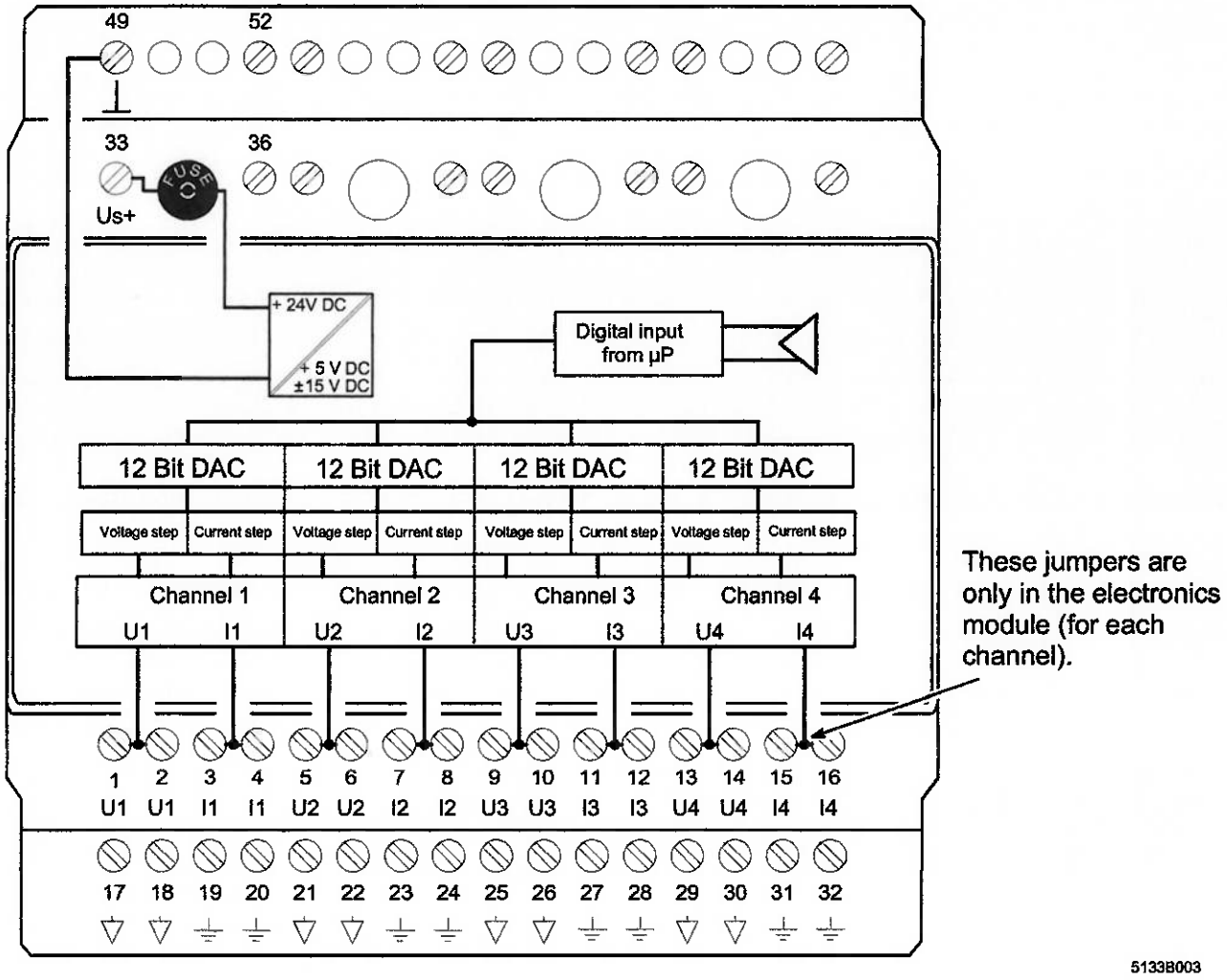


Figure 2 Internal wiring of the current and voltage channels

Electrical Isolation of the Single Function Areas

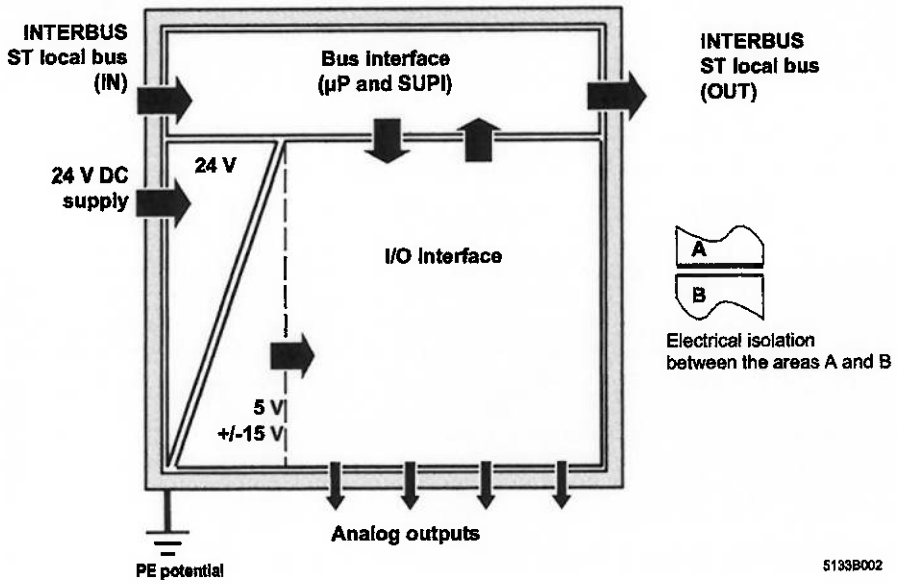


Figure 3 Electrical isolation of the single function areas

Connection Examples

Connection of the Supply Voltage

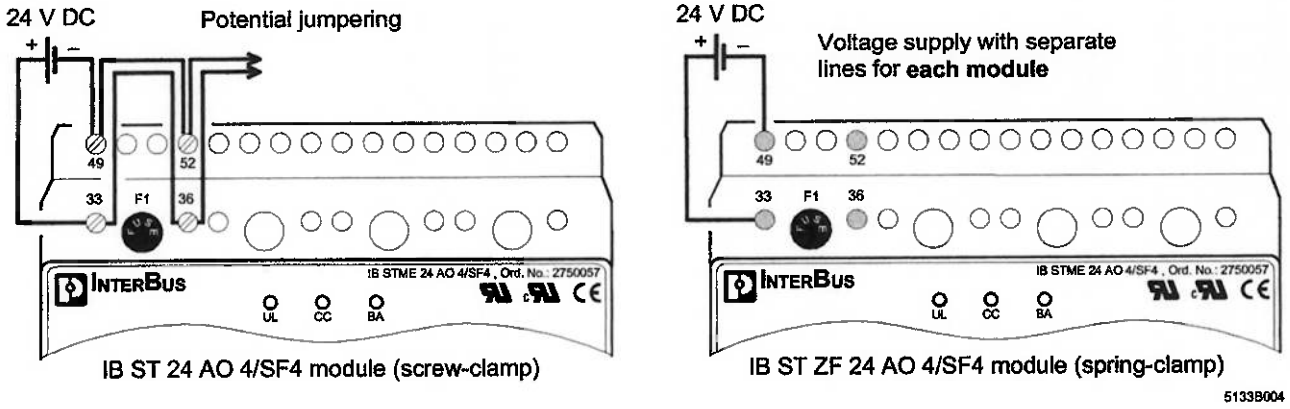


Figure 4 Connection of the supply voltage

Potential jumpering in the screw-clamp module:
To connect more modules, an external bridge is required between the terminals 33 (Us) and 36, and 49 (⊥) and 52.

Connecting Actuators

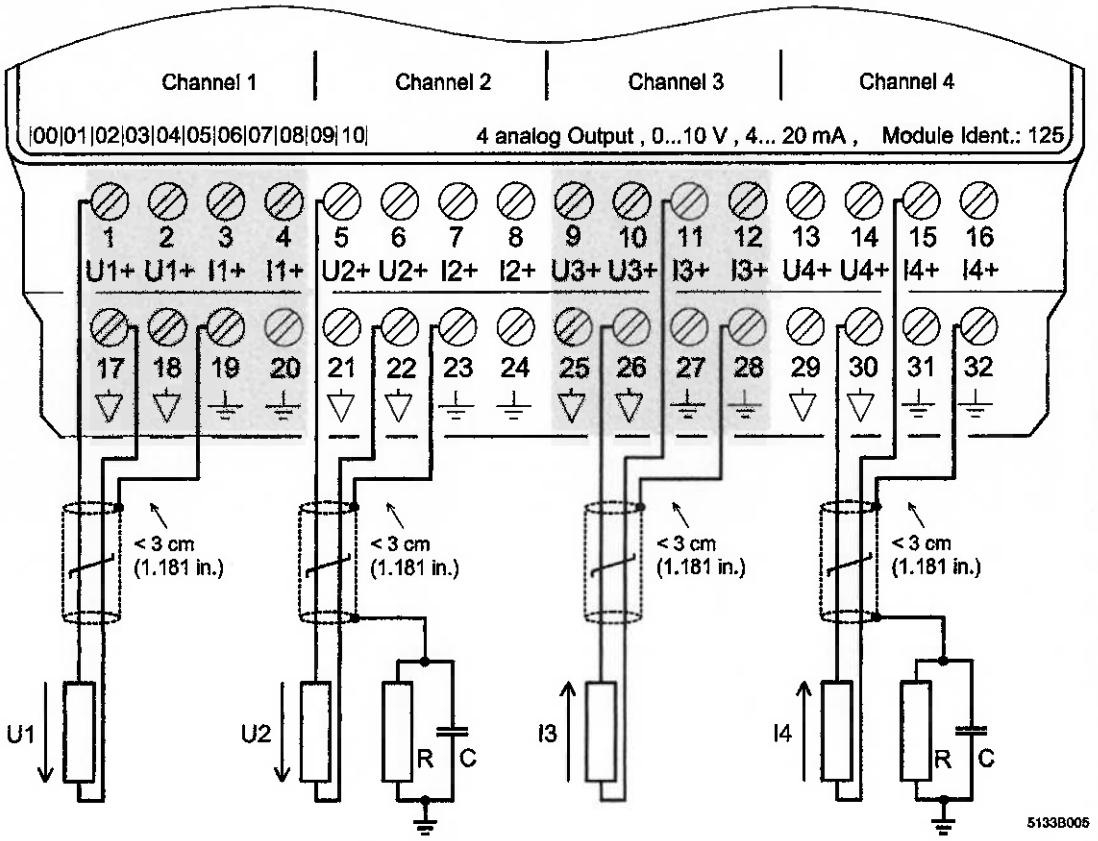


Figure 5 Connecting actuators to the module

Channel 1: Voltage tap with shield connection for cable lengths **below 10 m (33 ft.)**
Channel 2: Voltage tap with shield connection for cable lengths **exceeding 10 m (33 ft.)**
Channel 3: Current tap with shield connection for cable lengths **below 10 m (33 ft.)**
Channel 4: Current tap with shield connection for cable lengths **exceeding 10 m (33 ft.)**

When using cables with less than 10 m (33 ft.) length, connect one end of the shield to the IB ST (ZF) 24 AO 4/SF module (see Figure 5).
When using cables with more than 10 m (33 ft.) length in environments with heavy noise, we recommend connecting the shield also through an RC element to PE potential of the actuator (see Figure 5).
Typcially, the capacitor C should be rated between 1 and 15 nF. The resistor R should have at least 10 MΩ . The terminal points of the shield are directly connected with PE potential.

Terminals 1 and 2, 3 and 4, 5 and 6, 7 and 8, 9 and 10, 13 and 14, 15 and 16 only have the same potential if the electronics module is plugged in.

Programming Data

ID code	$7D_{hex}$ (125_{dec})
Length code	4_{hex}
Input address area	0 bytes
Output address area	8 bytes
Parameter channel (PCP)	0 bytes
Register length	8 bytes

INTERBUS OUT Process Data Word

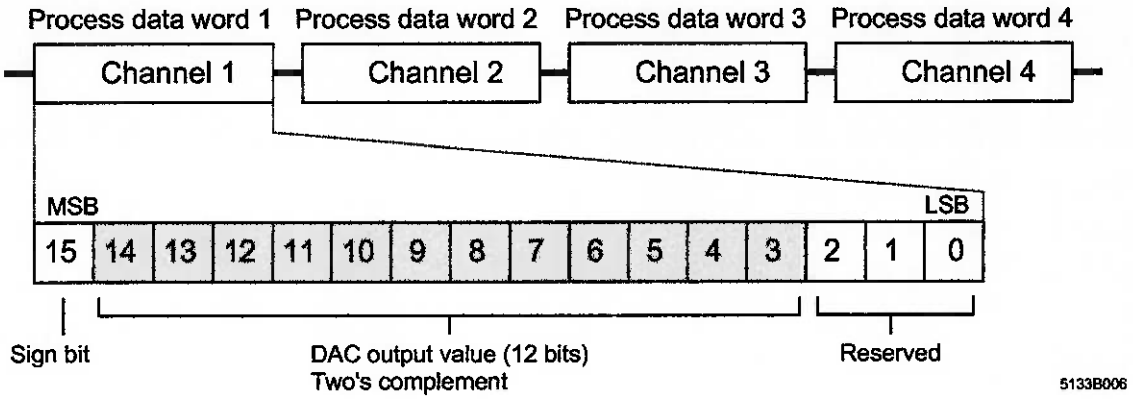


Figure 6 Sequence of the four INTERBUS OUT process data words in the INTERBUS ring and representation of the bits of the first process data word

The bits 0, 1, and 2 are reserved. The value of these bits is insignificant. Bit 15 is the sign bit. If it has the value 0, only positive values are output. As only positive voltage and current values occur the sign bit must contain zero. If bit 15 has the value 1, the output value is 4 mA or 0 V.

Assignment of the Module Terminals to the INTERBUS Reference

INTERBUS reference	Word	Word x															
	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Byte	Byte 1								Byte 0							
	Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Module terminals channel 1	Signal	Sign bit	Terminals 1 and 2: voltage output Terminals 3 and 4: current output														
	Analog ground		Terminals 17 and 18														
	Shield		Terminals 19 and 20														
Module terminals channel 2	Signal	Sign bit	Terminals 5 and 6: voltage output Terminals 7 and 8: current output														
	Analog ground		Terminals 21 and 22														
	Shield		Terminals 23 and 24														
Module terminals channel 3	Signal	Sign bit	Terminals 9 and 10: voltage output Terminals 11 and 12: current output														
	Analog ground		Terminals 25 and 26														
	Shield		Terminals 27 and 28														
Module terminals channel 4	Signal	Sign bit	Terminals 13 and 14: voltage output Terminals 15 and 16: current output														
	Analog ground		Terminals 29 and 30														
	Shield		Terminals 31 and 32														

INTERBUS OUT Process Data Word for the Voltage Outputs (Example)																		
Voltage output **0 V to 10 V**	Analog value (V)	Process data word																
		Hex.	Binary (two's complement) MSB															LSB
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
10 V minus 1 quantization step	9.9976	7FF8	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
10 V minus 2 quantization steps	9.9951	7FF0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
Half of the output range final value	5.0000	4000	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 quantization step	0.00244	0008	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Zero	0.0000	0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTERBUS OUT Process Data Word for the Current Outputs (Example)																		
Current output **4 mA to 20 mA**	Analog value (mA)	Process data word																
		Hex.	Binary (two's complement) MSB															LSB
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
20 mA minus 1 quantization step	19.9961	7FF8	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
20 mA minus 2 quantization steps	19.9922	7FF0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
Half of the output range final value	12.0000	4000	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4 mA plus 1 quantization step	4.00391	0008	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Output range start	4.0000	0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Technical Data

General	
Housing dimensions (width x height x depth)	118 mm (4.65 in.) x 116 mm (4.57 in.) x 117 mm (4.61 in.)
Operating mode	Process data operation with 4 words
Connection method of the actuators	2- and/or 3-wire technology
Total power consumption	5.1 W, typical
Permissible operating temperature	Voltage outputs: From 0°C (32°F) to 55°C (131°F) Current outputs: from 0°C (32°F) to 50°C (122°F)
Permissible storage temperature	From -25°C to 70°C (-13°F to 158°F)
Degree of protection	IP 20, DIN 40050, IEC 60529
Class of protection	Class 3 VDE 0106, IEC 60536
Humidity	75% on average, 85% occasionally, no condensation
Air pressure (operation)	From 80 kPa to 106 kPa, 2000 m (6562 ft.) above sea level
Electrical isolation	Test voltage
Bus/outputs	500 V AC, 1 min., 50 Hz
Supply voltage/outputs	500 V AC, 1 min., 50 Hz
Supply voltage/protective conductor	500 V AC, 1 min., 50 Hz
I/O voltage/protective conductor	500 V AC, 1 min., 50 Hz
Emitted interference	EN 50081-2, Class A
Processor monitoring	Watchdog circuit
Preferred installation position	Panel mounting
Protective ground connection	Via DIN rails
Weight	600 g, typical

Interface	
INTERBUS ST interface	ST cable (supplied with the module)

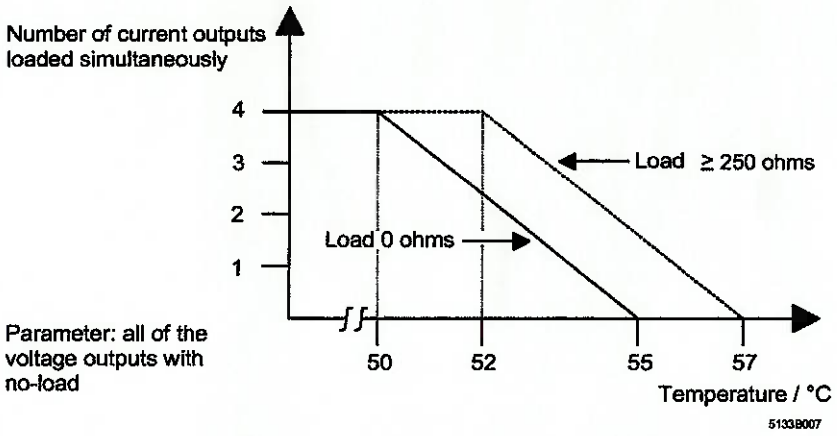
Power Consumption	
Communications power	9 V
Current consumption from the local bus	100 mA, typical; 130 mA, maximum
Power consumption from the local bus	1 W, typical
I/O supply voltage Us	24 V DC
Current consumption Us	170 mA, typical
Power consumption from power pack and application side (24 V supply)	4.1 W, typical
Total current consumption	5.1 W, typical

I/O Supply Voltage (U_S)	
Nominal value	24 V DC
Permissible ripple	3.6 V_{pp} within the permissible voltage range
Permissible voltage range (including ripple)	Operation: 18.5 V DC 30.2 V DC
Current consumption of U_S	170 mA, typical
Electrical isolation	Via DC/DC converter
Test voltage	500 V AC (50 Hz for 1 min.)
Protection against polarity reversal	Via diodes connected in series
Overload protection	Fuse F1 in base element IBS TR5 0,4 AT
Transient protection	Yes
Failure detection	Yes
Power consumption from power pack and application side (24 V supply)	4.1 W, typical

Analog Voltage Outputs	
Number	4
Output area	0 V to 10 V
Output current	5 mA, maximum
Permissible load resistor	2 kΩ, minimum
Representation of output value	16 bit two's complement
Quantization	12 bits (4096 steps) or 2.44 mV/quantization step
D/A conversion time of all channels	1 ms (incl. slew rate), maximum

Analog Voltage Outputs	(Continued)
Transient protection	Yes
Basic error limit	±0.05% of the output range final value
Derating	No derating
Separate shield connection	Yes

Analog Current Outputs	
Number	4
Output range	4 mA to 20 mA
Load range	0 Ω to 500 Ω
Representation of output value	16 bit two's complement
Quantization	12 bits (4096 steps) or 3.91 µA/quantization step
D/A conversion time of all channels	1 ms (incl. slew rate), maximum
Transient protection	Yes
Basic error limit	±0.1% of the output range final value



Derating curve of the current outputs of the IB ST (ZF) 24 AO 4/SF4 module

Tolerance Behavior and Temperature Response of the Voltage Outputs (Error Indications Refer to the Total Final Value of the Output Range (10 V Voltage))		
Offset error at 23°C (73.4°F)	**Typical**	**Maximum**
Total offset voltage	±0.025%	±0.10%
Gain error at 23°C (73.4°F)	±0.12%	±0.15%
Differential non-linearity	< ±0.02%	±0.02%

Tolerance Behavior and Temperature Response of the Voltage Outputs (Error Indications Refer to the Total Final Value of the Output Range (10 V Voltage))		
Temperature response at 0°C (32°F) to 55°C (131°F)		
Offset voltage drift T_{KVO}	< ±7 ppm/K	±10 ppm/K
Gain drift T_{KG}	±12 ppm/K	±40 ppm/K
Total voltage drift $T_{Ktot} = T_{KVO} + T_{KG}$	±19 ppm/K	±50 ppm/K
Total error of the voltage outputs (**0°C (32°F) to 55°C (131°F)**) Offset error + gain error + linearity error + drift error	±0.18%	±0.25%

Tolerance Behavior and Temperature Response of the Current Outputs (Error Indications Refer to the Total Value of the Output Range (20 mA Current))		
Offset error at 23°C (73.4°F)	**Typical**	**Maximum**
Offset current I_{ioc}	±0.06%	±.15%
Gain error at 23°C (73.4°F)	±0.19%	±0.3%
Differential non-linearity	< ±0.02%	±0.02%
Temperature response at 0°C (32°F) to 50°C (122°F)		
Offset current drift T_{KIO}	< ±7 ppm/K	±10 ppm/K
Gain drift T_{KG}	±13 ppm/K	±80 ppm/K
Total current drift $T_{Ktot} = T_{KIO} + T_{KG}$	±20 ppm/K	±90 ppm/K
Total error of the current outputs (**0°C (32°F) to 50°C (122°F)**) Offset error + gain error + linearity error + drift error	±0.25%	±0.5%

With 0°C (32°F) to 50°C (122°F) the total error for the current outputs is typically ±0.25% with reference to the output range final value of 20 mA. With 0°C (32°F) to 55°C (131°F) the total error of the voltage outputs is typically ±0.18% with reference to the output range final value of 10 V.

Additional Tolerances Influenced by Electromagnetic Fields	
Type of electromagnetic interference	Deviation
Radiated-noise immunity acc. to IEC 60801-3: 1984 (field strength 10 V/m)	< ±0.5%
Conducted interference IEC 60801-6: 1992 Class 3	< ±0.5%
Transient interferences (burst) acc. to IEC 60801-4 Class 3: 1988	< ±0.25%
Transient interferences (burst) acc. to IEC 60801-4 Class 4: 1988	< ±0.25%

All error indications are typical and refer to the output range final value of 10 V.

Response of the Control System or Computer to a Hardware Signal for Different Control or Computer Systems

Signal	Control or Computer System	Status After the Switching Operation		
		INTERBUS OUT Process Data Words 1 to 4	Analog Outputs	
			U_{out}	I_{out}
NORM*	AEG-Schneider Automation	0000	0 V	4 mA
BASP	Siemens S5	0000	0 V	4 mA
CLAB	Bosch	0000	0 V	4 mA
SYSFAIL	VME	0000	0 V	4 mA
CLEAR OUT	Klöckner-Moeller IPC	0000	0 V	4 mA
SYSFAIL	PC	0000	0 V	4 mA

* On controller boards for AEG-Schneider Automation control systems it is possible to set the NORM signal in such a way that the INTERBUS OUT process data words 1 to 4 and the analog outputs keep the last value.

Response of the Voltage and Current Outputs to a Control Command of the INTERBUS Controller Board

Command	Status After the Switching Operation		
	INTERBUS OUT Process Data Words 1 to 4	Analog Outputs	
		U_{out}	I_{out}
STOP	Keep last value	Keep last value	Keep last value
ALARM STOP (reset)	Keep last value	Keep last value	Keep last value

Output Behavior of the Voltage and Current Outputs

Supply voltage of the module	Supply Voltage of the BK Module	Bus Status	INTERBUS OUT Process Data Words 1 to 4	Behavior / Status of the Analog Outputs
From 0 V to 24 V	24 V	In operation	xxxx	Keep last value
From 0 V to 24 V	0 V	In operation	0000	0 V/4 mA
From 24 V to 0 V	Varies	In operation	yyyy	0 V/0 mA
24 V	From 0 V to 24 V	In operation	xxxx	Keep last value
0 V	From 0 V to 24 V	In operation	0000	0 V/0 mA
Varies	From 24 V to 0 V	In operation	yyyy	0 V/4 (0) mA
From 0 V to 24 V	From 0 V to 24 V	In operation	xxxx	Keep last value
24 V	24 V	Interrupted	xxxx	Keep last value
24 V	0 V	Interrupted	0000	0 V/4 mA
0 V	24 V	Interrupted	0000	0 V/0 mA
24 V	24 V	Reset	xxxx	Keep last value
24 V	0 V	Reset	0000	0 V/4 mA
0 V	24 V	Reset	0000	0 V/4 mA

Key:

$xxxx_{hex}$ Hexadecimal encoded value according to table on page 7.

Keep last value Output of the last output value before a power supply or a bus has broken down. The value is stored in the module and is encoded available in the process data words INTERBUS OUT 1 to 4 ($xxxx_{hex}$). During breakdown, the output is 0 V/4 mA. If the module is connected for the first time the memory contains the value 0V/4 mA.

$yyyy_{hex}$ Hexadecimal encoded output value which is sent from the control or computer system in the current bus cycle.

Mechanical Tests	
Vibration test (EN 60068-2-6; IEC 60068-2-6)	Acceleration amplitude above the limit frequency: 2g (operation); 5g (transport)
Shock test (EN 60068-2-27; IEC 60068-2-27)	Maximum acceleration: 15g Shock duration: 11 ms Maximum acceleration: 25g Shock duration: 6 ms
Mechanical noise (EN 60068-2-64; IEC 60068-2-64)	Acceleration density 30 to 200 Hz: 0.25 m^2/s^3 Effective acceleration: 7.8g

Module Error Messages	
Breakdown of the I/O voltages from ±15 V DC and/or 5 V DC	Yes
Breakdown of fuse F1 for the I/O supply voltage	Yes
Breakdown of the I/O supply voltage U_S 24 V DC	Yes

Ordering Data

Description	Designation	Order No.
Analog output module (screw-clamp)	IB ST 24 AO 4/SF4	27 50 57 8
Analog output module (spring-clamp)	IB ST ZF 24 AO 4/SF4	27 50 58 1
Electronics module	IB STME 24 AO 4/SF4	27 50 05 7
Replacement terminal block (screw-clamp terminals)	IB STTB 24 AO 4/SF	27 53 05 4
Replacement terminal block (spring-clamp terminals)	IB STTB ZF 24 AO 4/SF	27 50 86 6
Fuse 400 mA (slow-blow)	IBS TR5 0,4 AT	27 53 47 8