Proteo® PC

Manual Programação PLC - ST

Edição: Abril de 2016, Revisão A

Revision History

Revision	Remarks
04/2016 Rev A	Primeira revisão do manual de programação de PLC do CNC Proteo®.

Sumário

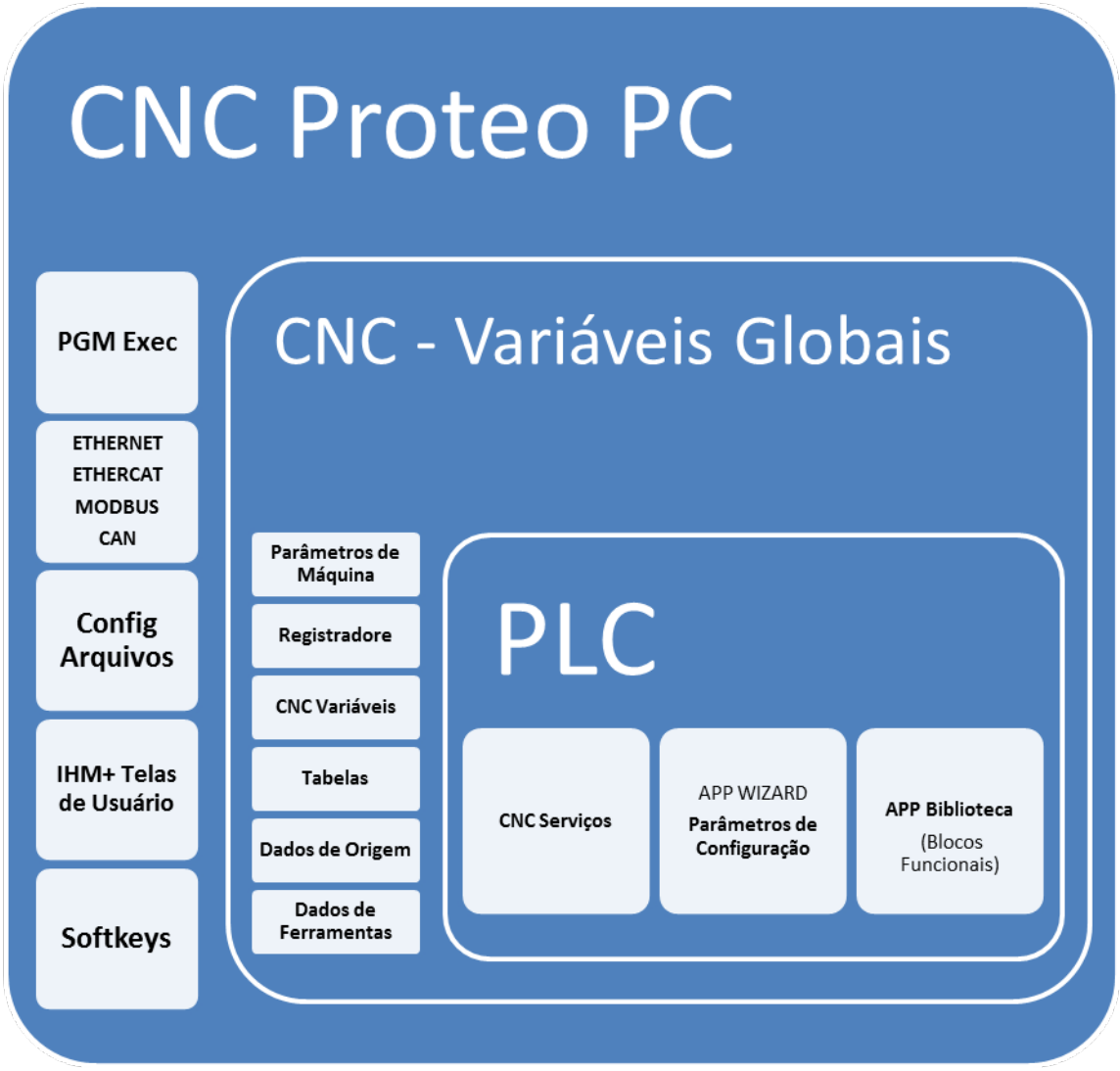
1 Introdução

CNC Proteo® PC – PLC utiliza IEC61131-3 ST (Texto Estruturado) para construir a interface com a Máquina.

Programação texto estruturado é uma linguagem muito poderosa. Se você tem experiência prévia em Lógica Ladder ou (FB) Bloco de Funções ou qualquer outra linguagem funcional como Basic, C ou Pascal, então você terá facilidade para aprender e em pouco templo dominar as técnicas de interface com a máquina com o CNC Proteo® PC.

Há muitos livros e treinamentos sobre a Programação de PLC definidas na norma IEC61131-3, o objetivo desta documentação não é reproduzir o que já está amplamente definido na norma e em outras publicações, mas buscamos transmitir uma compreensão básica da linguagem de texto estruturado no ambiente do Ativo™ e do CNC Proteo® PC. O nosso objetivo principal é capacitá-lo a utilizar nossas ferramentas para adaptar o CNC Proteo® PC a sua máquina no menor tempo possível.

2 Proteo® PC + PLC



CNC Proteo® PC

- PGM Exec : Execução de Programas do CNC, tem acesso ao mesmo ambiente de variaveis globais do CNC + PLC
- ETHERNET, ETHERCAT, MODBUS, CAN : Comunicação com outros equipamentos, incluíndo Acionamentos, módulos de Entrada/Saída, Inversores, Controladores de Temperatura e também Applicativos em ambiente Windows, como o Ativo™ , e aplicativos gerenciadores de dados.
- Config Arquivos: Arquivos de configuração como Arvore de Softkeys, Figuras, Telas de Usuário, Alarmes e Mensagens e muitos outros. Estes arquivos de dados ficam armazenados em uma estrutura de diretórios : \DATA
- IHM + Telas de Usuário : Interface home máquina composta por OBJETOS DE TELA organizados em “SCREENS”. Alguns objetos possuem conteúdo dinâmico, como por exemplo o monitor de execução ou o simulador gráfico, sensíveis a execução do programa. Outros apresentam informações estáticas e figuras. Muitos SCREENS são implementados para satisfazer condições de determinadas aplicações e tem significado especifico. Temos também as TELAS DE USUÁRIO (USER SCREENS), que são montadas especificamente para atender funcionalidades de uma aplicação específica.
- Softkeys: Organizadas em forma de “ARVORE” , representam teclas ou conjuntos de teclas. Utilizam “ICONS” para indicar ao operador um significado visual e um texto explicativo simplificado. As

softkeys enviam códigos de teclas ao CNC / PLC e podem ativar funcionalidades e também mudar o foco de uma tela para outra. A navegação pelos diversas telas da IHM ou telas de usuário está definida na árvore de Softkeys.

- Parâmetros de Máquina : Definem os limites e propriedades dos eixos bem como propriedades dos elementos do CNC e da máquina.
- Registradores: Memórias genéricas utilizadas tanto pelo Programa definido pelo usuário quanto ao PLC e telas de usuário. Constituem memória tanto de passagem de dados entre programas quanto um espaço de rascunho para execução de expressões e dados utilizados pelos ciclos fixos do CNC.
- CNC Variáveis : Registradores especiais conhecidos como “Variáveis Reservadas” permitem o acesso a dados conhecidos pelo CNC como a posição atual dos eixos, ou o último valor da velociade de avanço. A utilização destas variáveis reservadas demanda um conhecimento mais aprofundado do funcionamento do CNC e normalmente o acesso as mesmas é feito com o auxílio de especialistas.
- Tabelas: Armazenam informações de dados programados externamente (MS Excel) e podem ser utilizadas em programas paramétricos ou ciclos fixos do CNC, como em configurações de dados de processo utilizados em Scripts de Telas de Usuário e também pelo PLC. A utilização de Tabelas demanda um conhecimento mais aprofundado do funcionamento do CNC e normalmente o acesso as mesmas é feito com o auxílio de especialistas.
- Dados de Origem: Dados do Editor de Origens para definir as origens dos eixos. Existem várias origens que permitem definir coordenadas relativas ao ZERO MÁQUINA (G53) ou relativas aos chamados ZERO PEÇA (G54, 55, 56 e 57). Estes dados podem ser acessados via variáveis reservadas (CNC).
- Dados de Ferramentas : Dados do Editor de Ferramentas para definir as dimensões e propriedades das ferramentas, adequados para cada tipo de aplicação, notoriamente para tornos, fresas, retíficas e centros de usinagem em geral.
- CNC Serviços: Determinadas operações precisam ser feitas pelo CNC por exigirem reações compatíveis com a precisão e e velocidade de resposta. Os assim chamados “SERVIÇOS DO CNC” podem ser utilizados pelo PLC para executar funções específicas como por exemplo a busca de zero – referência, ou o movimento manual dos eixos, ou até mesmo o controle completo de um “SPINDLE” em um torno ou centro de usinagem.
- APP WIZARD / Parâmetros de Configuração : O Assistente de aplicação "APP WIZARD” orienta e ajuda o usuário a construir a sua aplicação base. Passo a passo o APP WIZARD dirige o usuário a selecionar opções solicitando a introdução de dados de configuração para determinados tipos predefinidos de máquinas e componentes do CNC. O Resultado final é um projeto completo de design do aplicativo , incluindo PLC, GUI , I / O Map e a documentação básico para o projeto elétrico.
- APP Biblioteca (Blocos Funcionais) : Nossa engenharia de aplicações disponibiliza uma biblioteca de poderosos blocos funcionais que facilitam muito a elaboração do PLC e da interface entre o CNC e a Máquina. Ao invéz de blocos elementares da linguágem ST como Flip-Flops e Funcões elementares, os blocos funcionais fornecidos constituem os elementos fundamentais para o funcionamento da máquina. Por exemplo o controle de altura integrado de uma máquina de corte plasma. Fornecemos os blocos funcionais básicos para resolver um problema complexo, mas necessário para qualquer máquina de corte de chapas. O PLC se torna muito mais simples graças aos blocos funcionais fornecidos.

3 Noções básicas de linguagem estruturada (ST)

Mesmo um pequeno programa em texto estruturado utiliza muitas das características tradicionais de outras linguagens de alto nível como o Basic, C ou Pascal, incluindo variáveis, operadores e instruções de fluxo de controle. O código pode parecer um pouco misterioso para os programadores que não têm pouco ou nenhum conhecimento em programação de alto nível. Mas este manual ensina o que você precisa saber sobre conceitos básicos da linguagem de programação Texto Estruturado

.

• Variáveis: Variáveis no programa texto estruturado são utilizadas para armazenar dados.

• Operadores: como operações aritméticas e de atribuição.

• Expressões: É uma combinação de operadores e variáveis em sequências conhecidas como expressões. Expressões são os blocos básicos de construção do seu código.

• Controle de Fluxo: Instruções de fluxo de controle para executar condicionalmente declarações ou para saltar para outra área no programa.

4 Por onde começar?

Texto estruturado é a forma mais rápida e fácil de criar programas em controladores PLC. Em ST você pode cercar o código de comentários de texto simples que irão ajudar você a produzir um código que outras pessoas também possam entender.

Se você já é um profissional experiente ou ainda está iniciando na programação de PLC, vai encontrar no Ativo™ um poderoso Editor de programas PLC, com uma ajuda sensível ao contexto do seu programa. O Ativo™ oferece um conjunto completo de ferramentas para simplificar e agilizar o desenvolvimento do programa PLC.

A parte "estruturado" refere-se aos recursos de programação de alto nível e "Texto" refere-se à capacidade de utilizar texto em vez de símbolos como em Lógica Ladder, Bloco de Funções ou Cartas de tempo.

Se você já usou qualquer linguagem de alto nível, como Basic, Pascal ou C, então você provavelmente vai ter as habilidades necessárias para criar um programa simples, sem quaisquer dificuldade.

Nós preparamos um rico conjunto de exemplos, organizados por Aplicação, para utilizar os Blocos Funcionais básicos e bibliotecas fornecidos junto com o Ativo™ e com o CNC Proteo® PC.

Nossos exemplos não estão focados na linguagem propriamente dita, mas na utilização de uma séries de BLOCOS DE FUNÇÕES e BIBLIOTECAS que facilitam a elaboração da interface com a máquina. É nosso objectivo principal ajudá-lo a chegar lá e construir o seu próprio código de partida com algo que realmente funcione.

Como qualquer outra linguagem de alto nível, o Texto Estruturado contém muitas funções e palavras-chave semelhantes a aquelas utilizadas de forma quase que universal e intuitiva. Os novatos podem criar aplicações úteis, aprendendo apenas algumas das palavras-chave, no entanto, o poder da linguagem permite que os profissionais especializados explorem todos os recursos de interface do CNC Proteo® PC, em nada devendo as outras linguagens de alto nível.

Se o seu objetivo for criar uma pequena função ou desenvolver programas de automação grandes e sofisticados, a linguagem de programação Texto Estruturado tem as ferramentas que você precisa. Cada linguagem de programação tem suas próprias vantagens e desvantagens. Uma das principais vantagens do texto estruturado é a sua capacidade para expressar equações logicas e simplificar equações matemáticas complexas. A documentação pode ser gerada enquanto se está programando através de comentários adequados explicando as funcionalidades e também o significado das variáveis e dos intertravamentos programados.

O Editor de Propramas do PLC em linguagem estruturada (ST) possui um “HELP” sensivel ao contexto que facilita a edição do programa PLC, utilizando cores para diferenciar as palavras reservadas e nomes de variáveis e funções inseridas pelo programador.

As palavras de comando constituem a unidade básica da Linguagem de Texto Estruturado e um conjunto destas palavras reservadas ou comandos, constituem um programa em ST.

ST é uma linguagem textual estruturada de alto nível com recursos semelhantes às linguagens "C" e Pascal. Com ela é possível escrever programas com todos os elementos essenciais de uma linguagem de programação moderna, tais como os comandos IF, THEN, ELSE, laços FOR e WHILE, criar variáveis locais e arrays, desenvolver e chamar funções, etc.

5 Exemplo de um programa em ST:

```
FUNCTION_BLOCK     CTU_TEST
VAR_INPUT
CU, R : BOOL;
PV : INT;
END_VAR
VAR_OUTPUT
Q : BOOL;
CV : INT;
END_VAR
IF R THEN
CV := 0 ;
ELSIF R_EDGE(CU) AND (CV < PV) THEN
CV := CV + 1;
END_IF ;
Q := (CV >= PV) ;
END_FUNCTION_BLOCK

PROGRAM Main
VAR
fb_ctu1 : CTU_TEST ;
results : ARRAY [0..9] OF INT;
Qs : ARRAY [0..9] OF BOOL;
clk, reset : BOOL ;
preset : INT ;
cycNo : INT ;
END_VAR

clk := (cycNo MOD 2) = 1;
preset := 3;
fb_ctu1(clk, reset, preset);
results[cycNo] := fb_ctu1.CV;
Qs[cycNo] := fb_ctu1.Q;
IF cycNo = 9 THEN
reset := TRUE;
END_IF;
IF cycNo = 1 THEN
reset := FALSE;
END_IF;
```

```
cycNo := (cycNo + 1) MOD 10;

END_PROGRAM
```

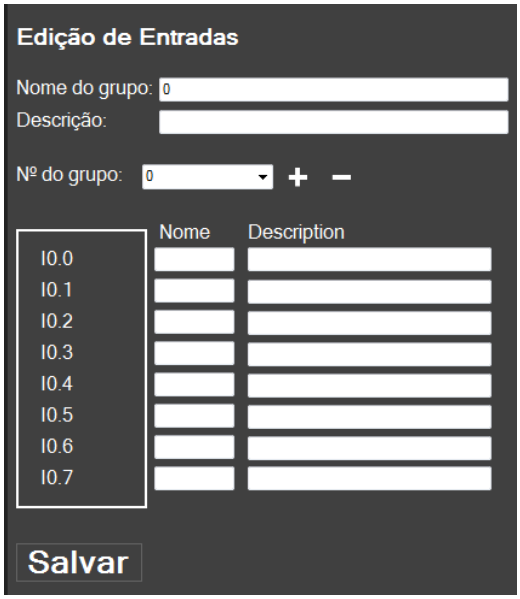
6 Estrutura de Linguagem ST

6.1 Tipos de Dados

Ao cadastrar as variáveis é necessário informar o tipo de dado da mesma. O Ativo™ suporta os seguintes tipos

No Ativo™ existe uma tela para o cadastro de variáveis. Esse cadastro é subdivido em variáveis e I/O's

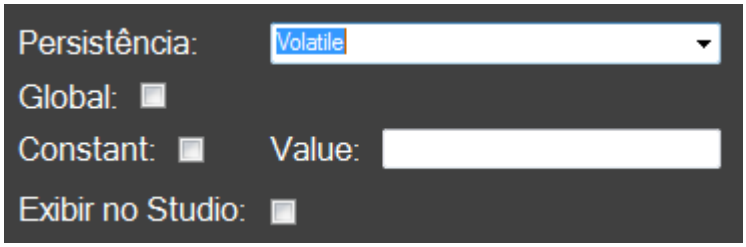
Abaixo temos duas telas de cadastro



Ao declarar variáveis podemos selecionar vários tipos.

Tipo	Bits	Descrição
Bool	1	Booleano
INT	16	Inteiro
UINT	16	Inteiro Sem sinal
SINT	8	Small Int
USINT	8	Small Int Sem Sinal
DINT	32	Double Int
UDINT	32	Double Int Sem sinal
LINT	64	Long Int
ULINT	64	Long Int Sem sinal
REAL	32	Real
ARRAY	0	Array

Outro ponto importante são as opções ligadas a cada variável declarada.



A declaração Persistência é usada para reter um valor variável mesmo quando a energia for desligada e ligada PLC.

Global é uma variável que pode ser utilizada em qualquer função ou blocos.

Constante, serve para indicar um valor para a variável, por exemplo, Pi que recebe o valor de 3.14. É possível declarar constantes de qualquer variável numérica (booleanos não são possíveis de fixar como true ou false).

O campo exibir no Studio será abordado posteriormente.

6.2 Instruções

A seguir listamos todas as instruções da linguagem ST que é utilizada para programar o PLC no Ativo™ .

Instruções de bit

Descrição	Função
:=	Atribuição
OR	Booleano OU
AND	Booleano E
XOR	Booleano OU Exclusivo
NOT	Negação
R_EDGE	Borda de Subida
F_EDGE	Borda de Descida

Comparações numéricas (DINT, REAL e LINT)

Descrição	Função
<	Menor Que
>	Maior Que
<=	Menor ou igual a
>=	Maior ou igual a
=	Igual
<>	Diferente

Instruções aritméticas (DINT, LINT)

Descrição	Função
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
MOD	Resto de divisão inteira
SQRT	Valor absoluto
ABS	

Instruções aritméticas (REAL)

Descrição	Função
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
MOD	Resto de divisão inteira
SQRT	Quadrado
ABS	Valor absoluto
**	Potenciação
SIN()	SENO
COS()	Cosseno
TAN()	Tangente
ATAN()	Arco Tangente

7 Expressões

Expressões são compostas por operadores e operandos.

Um operando pode ser uma variável, uma chamada de função, ou outra expressão. Os operadores da linguagem ST que são utilizados no Ativo™ estão listados nas tabelas acima.

Exemplos de expressões:

A := B;

A+B * SQRT(C)

Sempre que houver mais de um operando por operador, o operando mais a esquerda deve ser avaliado primeiro.

COS (A) - SIN (C)

A expressão COS (A) deve ser avaliada primeiro, seguida por COS (B), depois faz-se a subtração dos dois valores.

7.1 Expressões Booleanas

Expressões booleanas funcionam comparando operadores e gerando um resultado booleano(0 ou 1, TRUE ou FALSE).

A<>B AND C = D OR A>= C

O resultado dessa expressão só será verdadeiro (1), se todas as condições forem verdadeiras.

Esse tipo de expressão é utilizado principalmente dentro de comandos da linguagem. Como veremos mais a frente.

8 Elementos de ST

A linguagem ST possuí diversos elementos, estes que são operadores, símbolos, números e outros sinais próprios da linguagem.

8.1 Identificadores

Nome de variáveis, funções e blocos são considerados Identificadores.

É permitido usar letras, números e caracteres no cadastro dos nomes. A única regra que precisa ser obedecida é que não é permitido o uso de palavras reservadas do ST. No nosso cadastro é livre a quantidade de caracteres que será inserida para a um identificador.

8.2 Comentários

Para inserir um comentário deve-se inserir entre asteriscos, por ex (*exemplo*). Se for inserido corretamente a sintaxe ficará verde. Também é possível inserir comentário linhas, bastando digitar // antes da linha que deseja comentar.

No exemplo abaixo é possível observar os dois tipos de comentários.

```
51
52  //Emergência
53  //----------------------------------------------------------------------
54  // Botões de emergencia.
55      Emergency_CNC := I_Emergency OR Simulacao;
56
57  // Saída de emergencia
58      O_Emergency := Emergency_Output_Connected;
59
60      (*Mensagens
61      MESSAGE(0 , (NOT Emergency_CNC AND Initialization_Mode));
62      ALARM(0, (NOT Emergency_Output_Connected AND CNC_Initialized));
63      ----------------------------------------------------------------------    *)
64
```

8.3 Funções

Função é uma ação que recebe parâmetros e quando executada devolve um valor de retorno. Funções podem ser chamadas dentro de outra expressão quantas vezes forem necessárias, desde que elas não se tornem recursivas. Abaixo um exemplo de uma função sendo chamada no código ST.

Criamos uma função chamada média

```
1  Media :=(valor1 + valor2)/2;
```

Onde são passados dois parâmetros a ela, o valor1 e o valor2.

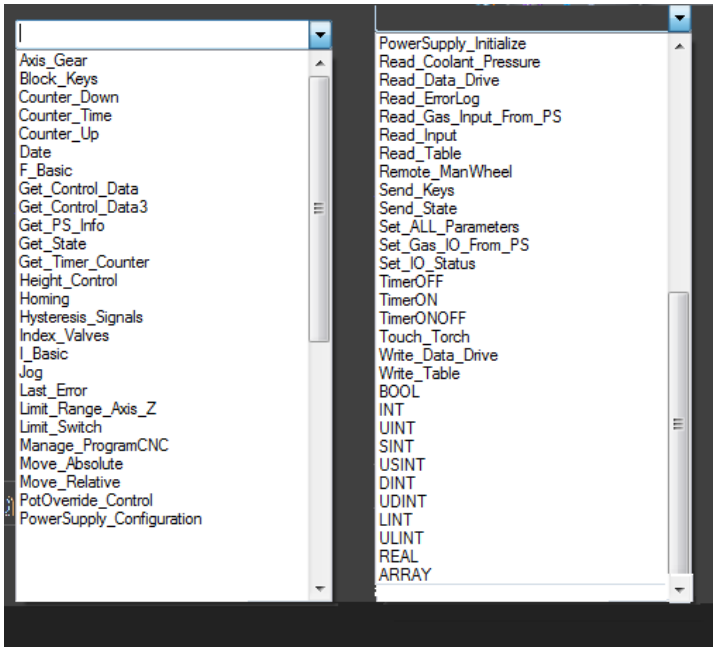
No programa principal devemos invocar essa função como é possível ver a seguir e inserimos os dois valores que serão calculados a média.

```
1  Result := Media(10, 5);
```

Result vai ter o valor de 7.5 pois a função média foi declarada com um retorno em Real.

8.4 Blocos

Bloco é um conjunto de informações onde você instancia-o para que ele execute uma ação em cima dessas informações. A principal diferença entre função e bloco, é a existência de instancia, não ter valor de retorno e é possível ter dois blocos do mesmo tipo com diferentes informações(diferentes instancias) executando a mesmas ações.



A seguir temos um exemplo do cadastro de um bloco que instância o bloco Homing.

Com o bloco Busca_Referencia cadastrado, basta utiliza-lo no seu PLC, abaixo um exemplo da utilização real dele.

```
214     Busca_Referencia(Reference_Mode, Start_Referencia_Timed OR Key_Code(2041),
215                                      (Key_Code(2042) OR NOT I_Stop),
216                                       Key_Code(2044), FALSE,
217                                       1, I_Ref1, I_Ref2, I_Ref3, FALSE, FALSE, FALSE, FALSE, FALSE);
218
```

É possível utilizar a saída dos blocos em sua lógica de PLC, para fazer isso deve-se usar o nome do bloco concatenado com um "ponto" (.) com a saída desejada. Observe o exemplo abaixo

Foi utilizado a saída REF_Running que é do bloco de busca referência.

```
IF R_EDGE(Busca_Referencia.REF_Running) THEN

    Start_Referencia:= FALSE;

END_IF;
```

8.5 Comando IF

IF é um comando condicional que executa as ações que estão depois de THEN somente se a condição for verdadeira. É possível também encadear com o comando ELSEIF e o ELSE

Abaixo o formato de como se utiliza a instrução IF

```
IF <EXPRESSION> THEN
    <ACTIONS>
ELSIF <EXPRESSION> THEN
    <ACTIONS>
ELSE
    <ACTIONS>
END_IF;
```

Abaixo temos um exemplo com valores numéricos e variáveis da utilização do IF

```
IF Result > 6 THEN
    OK := 1;
    ELSIF Result <6 AND <> 0 THEN
    OK := 0;
END_IF;
```

8.6 Comando CASE

Case é outro comando que também executa apenas um bloco de ações. A validação do case é feita comparando o valor da <EXPRESSION> com o valor dos <CASE> e executando apenas o que tem o valor igual. Abaixo temos o formato de como se utiliza o comando.

```
CASE <EXPRESSION> OF
    <case>:
    <ACTIONS>
    <case2>:
    <ACTIONS>
END_CASE;
```

Exemplo de programação

```
CASE Alarm_Number OF
        0:
        LampadaEmerg := TRUE;
        1:
        LampadaAmarela:= TRUE;
        2:
        LampadaAmarela:= FALSE;
END_CASE;
```

8.7 Comando WHILE

While é um tipo de comando que executa um bloco de ações enquanto a condição de validação for verdadeira. Ele sempre testa a validação antes de executar as ações.

```
WHILE <EXPRESSION> DO
    <ACTIONS>
END_WHILE;
```

Exemplo de programação do While.

```
Count := 0;
Init := 10;
Target := 2;
Temp := Init;

WHILE Temp > Target DO
    Temp := Temp / 2;
    Count := Count + 1;
END_WHILE;
```

8.8 Comando FOR

O comando FOR é uma estrutura de repetição assim como o comando WHILE. Ele utiliza uma variável para controlar a contagem e o incremento.

No inicio de cada contagem é feito uma verificação, se o <valor_inicial> atingiu a condição <valor_final>, a <ACTION> não será executada e o programa vai seguir até encontrar outra instrução. Caso o a condição ainda não foi atingida, o programa executa a <ACTION> e incrementa a <valor_inicial> com base no <valor_incremento>. Abaixo temos como se escreve o comando for.

```
FOR(<valor_inicial> TO <condicao_final> BY <valor_incremento> DO
    <ACTIONS>
END_FOR;
```

Exemplo do uso da função FOR

```
FOR Index := 0 TO 9 BY 1 DO
        TestArray[Index] := 7.0;
END_FOR;
```

9 Blocos Funcionais

Blocos funcionais funcionam como variável, ou seja, é necessário instanciar estes objetos.

A Instância é um objeto, ou seja, uma cópia da estrutura principal, por isso deve ser criada uma variável com o nome que o aplicador queira e identificar o tipo desta variável de acordo com o Bloco Funcional que se deseja instanciar. A variável passa a ser do tipo do bloco funcnional escolhido.

9.1 Axis_Gear

9.1.1 Funcionamento

- Este bloco executa o acoplamento de um eixo escravo a um eixo mestre, com isso pode-se fazer um eixo escravo seguir os movimentos do mestre com uma relação de acoplamento pré definida.

- A relação de acoplamento pode ser positiva ou negativa. Se positiva o eixo escravo segue o eixo mestre no mesmo sentido do movimento do mestre. Valores negativos fazem o eixo escravo se movimentar no sentido inverso ao do movimento do eixo mestre.

- O módulo da relação de acoplamento determina o relação entre o movimento do eixo mestre e do eixo escravo. Valores entre 0 e 1 fazer o eixo escravo se movimentar menos do que o movimento do mestre. Valores maiores que 1 fazem o eixo escravo se movimentar mais do que o mestre. Por exemplo, para relação igual a 2, para cada milímetro de movimento do eixo mestre, o escravo irá mover 2 mm.

9.1.2 Entradas

Enable (BOOL):

- Para que o bloco seja executado é necessário que o argumento Enable esteja com valor TRUE. Caso o valor esteja em FALSE o bloco é desabilitado e todas as saídas recebem valores nulos.

AG_Clutch_Connect (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado o acoplamento do eixo escravo ao mestre levando em consideração todos os valores atribuídos ao restante dos argumentos.

AG_Clutch_Disconnect (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE o acoplamento do eixo escravo é cancelado deixando-o liberado para movimentos independentes aos do mestre.

AG_Coupling_Angle_Tangent(BOOL):

- Ao solicitar um acoplamento com este argumento em TRUE, o número de um eixo no argumento de eixo escravo e este argumento em TRUE o eixo segue tangente à trajetória no plano definido pelo CNC via funções preparatórias G17 (plano XY), G18 (plano YZ) ou G19 (plano ZX), ou seja, no caso do plano XYn sendo o eixo C (rotativo) o eixo escravo, caso o XY execute um movimento fazendo um círculo, o eixo C faria um movimento tangente à circunferência descrita por XY.

AG_Coupling_Negative (BOOL):

- Ao solicitar o acoplamento com este argumento em TRUE, o eixo escravo só se movimenta quando o eixo mestre se desloca no sentido negativo.

AG_Coupling_Positive (BOOL):

- Ao solicitar o acoplamento com este argumento em TRUE o eixo escravo só se movimenta quando o eixo mestre se desloca no sentido positivo.

AG_Coupling_Offset (INT):

- Deslocamento que é executado no eixo escravo em relação ao eixo mestre. É executado apenas uma vez a cada transição de .....

AG_Coupling_Real_Point (BOOL):

- Quando solicitado o acoplamento com este argumento em TRUE, o acoplamento é feito pelo ponto real, ou seja, é acoplado a m um encoder, resolver ou outro transdutor de posição.

- Quando solicitado o acoplamento com este argumento em FALSE o acoplamento é feito pelo ponto teórico, ou seja, é acoplado ao ponto calculado pelo CNC.

AG_Master_Axis (INT):

- Neste argumento é atribuido o número do eixo correspondente ao eixo mestre determinado nos parâmetros de máquina (endereçamento dos drives).

AG_Sleave_Axis (INT):

- Neste argumento é atribuido o número do eixo correspondente ao eixo escravo determinado nos parâmetros de máquina (endereçamento dos drives).

AG_Ratio (REAL):

- Fator de acoplamento entre o escravo e o mestre pode ser programado 1 para uma relação 1:1, 2 para que o eixo escravo dê 2 voltas para 1 volta do mestre ou então 0.5 para que o eixo mestre de 2 voltas para 1 volta do escravo, este fator é variável podendo se colocar o fator necessário para sua aplicação.

- Caso se acople um eixo linear a um eixo rotativo cada um grau corresponde a 0.0001 mm.

ExeChannel_ID (INT):

- Este argumento serve para selecionar a qual canal de execução do CNC este acoplamento pertence, podendo ser programado 1 a 4, correspondente aos canais de execução existentes no CNC.

9.1.3 Saídas

Enabled (BOOL):

- Indica que o bloco esta habilitado e pronto para o uso.

Axis_Clutched (BOOL):

- Indica que o eixo escravo está acoplado ao respectivo eixo mestre.

Error (BOOL):

- Este argumento retorna TRUE caso ocorra qualquer erro no acoplamento (vide Error_Code).

Error_Code (INT):

- Quando um erro é sinalizado em "Error" este argumento recebe o código de erro gerado pelo bloco. O significado deste erro é apresentado na tabela de erros dos blocos.

9.1.4 Como se escreve

Axis_Gear(AG_Clutch_Connect, Enable, AG_Master_Axis, AG_Sleave_Axis, AG_Ratio, AG_Clutch_Disconnect, AG_Coupling_Real_Point, AG_Coupling_Positive, AG_Coupling_Negative, AG_Coupling_Angle_Tangent, AG_Coupling_Offset, ExeChannel_ID);

9.2 Height_Control

Parâmetro de máquina

- Existem alguns parâmetros de máquina que precisam ser programados para que este bloco funcione corretamente. Estes parâmetros devem ser programados na aba PLC do objeto de parâmetros do CNC. Neste caso o aplicador deve adicionar estes parâmetros na aplicação que estiver sendo desenvolvida.

P960 = Número do eixo associado ao THC1

P961 = Número do eixo associado ao THC2

P962 = Número do eixo associado ao THC3

P963 = Número do eixo associado ao THC4

P964 = Entrada analógica associada ao feedback analógico do controle de altura do THC1

P965 = Entrada analógica associada ao feedback analógico do controle de altura do THC2

P966 = Entrada analógica associada ao feedback analógico do controle de altura do THC3

P967 = Entrada analógica associada ao feedback analógico do controle de altura do THC4

P975 = janela de reação:

- Faixa de tensão em torno da tensão de controle, dentro da qual o controle de altura é congelado automaticamente. O centro da faixa de controle é a tensão de controle.

P976 = janela de controle:

- Faixa de tensão em que o THC é efetivamente controlado, fora desta janela o controle de altura do THC fica congelado. A janela de controle deve ser obrigatoriamente maior do que a janela de reação.

P977 = Ganho proporcional para malha do controle de altura (feedback analógico).

P978 = Ganho integral para malha do controle de altura (feedback analógico).

P979 = Ganho diferencial para malha do controle de altura (feedback analógico).

P981 = Tensão de fundo de escala, ou seja, tensão na tocha correspondente a 10 volts na entrada analógica do THC.

P984 = Porcentagem de tensão para detectar mudanças bruscas na tensão de arco, ou seja, caso haja um aumento súbito na tensão equivalente ao valor deste parâmetro o controle de altura do THC é congelado.

9.2.1 Funcionamento

- Este bloco é responsável por executar o controle do THC utilizado em máquinas de corte Plasma, onde um eixo perpendicular ao plano de corte ajusta sua altura dependendo das ondulações da chapa. Esta correção é feita por um feedback analógico, ou seja, é passada ao CNC uma referência analógica correspondente à tensão de arco para que a correção da altura seja executada.

- Todo o processo executado por este bloco consiste em não apenas habilitar o controle de altura, é responsável também por executar uma série de funções necessárias para o processo.

- Uma vez habilitado o bloco e solicitado iniciar o controle de altura o processo consiste em:

• Aguardar o tempo para transferência de arco, em que se espera o retorno da informação de arco transferido (Arc_Transfered) após iniciar a ignição da fonte plasma

• Posicionar o eixo na altura de perfuração com velocidade controlada (Pierce_Height, Pierce_Height_Velocity)

• Contar tempo de perfuração (Pierce_Time)

• Contar tempo de pula borra (Molten_Pool_Jump_Time)

• Posicionar na altura de corte com velocidade controlada (Cut_Heigth, Cut_Heigth_Velocity)

• Assim que posicionado na altura de corte iniciar o controle de altura

9.2.2 Entradas

Enable (BOOL):

- Para que o bloco seja executado é necessário que o argumento Enable esteja com valor TRUE. Caso o valor esteja em FALSE o bloco é desabilitado e todas as saídas recebem valores nulos.

Start_Heigth_Control (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado o início do processo de habilitação do controle de altura.

Arc_Transfer_Time (REAL):

- Tempo em segundos para aguardar a transferência de arco (Arc_Transfered), caso o arco não seja transferido até que este tempo se encerre o processo é abortado e é indicado erro (Fault_Arc_Transfer).

Arc_Transfered (BOOL):

- Sinal aguardado durante o tempo de transferência de arco (Arc_Transfer_Time). Durante este tempo é esperado que este sinal passe para estado lógico TRUE.

Pierce_Height (REAL):

- Posição absoluta onde será posicionado o THC para iniciar a perfuração da chapa.

Pierce_Height_Velocity (INT):

- Velocidade com que o CNC posiciona o THC na posição de perfuração

Pierce_Time (REAL):

- Tempo em que o eixo fica parado na posição de perfuração antes de ir para o próximo passo do processo.

Molten_Pool_Jump_Time (REAL):

- Tempo para pular borra. O controle de altura permanece na altura de perfuração pelo tempo estabelecido neste argumento.

Cut_Heigth (REAL):

- Posição absoluta onde será posicionado o THC para que se inicie o corte.

Cut_Heigth_Velocity (INT):

- Velocidade com que o CNC posiciona o THC para posição de corte.

Disable_Heigth_Control (BOOL):

- Na transição de FALSE para TRUE é solicitado ao bloco que desabilite o controle de altura, ou seja, o CNC deixa de olhar o feedback analógico para corrigir a altura de corte.

FreezeControl (BOOL):

- Solicita ao bloco que congele a altura atual do THC, ou seja, na transição de FALSE para TRUE o bloco mantem o THC na altura em que se encontrava antes dessa função ser solicitada. Esta função só é ativada caso o controle de altura esteja habiltado.

Reference_Voltage (INT):

- Tensão de correspondente à altura de corte programada (Cut_Heigth). Quando o CNC posiciona o THC na altura de corte, espera-se que a tensão de arco corresponda ao valor deste argumento. Ao se posicionar na altura de corte o controle de altura espera que a tensão de arco assuma este valor.

THC_ID_Number (INT):

- Número do THC que se quer habilitar. O CNC está preparado para controlar até 4 unidades de controle de altura, ou seja, podemos programar valores de 1 a 4.

9.2.3 Saídas

Enabled (BOOL):

- Indica que o bloco esta habilitado e pronto para o uso.

Block_Ohmic:

- Indica que o toque ôhmico está bloqueado.

Control_Heigth_Enabled (BOOL):

- Indica que o processo para habilitar o controle foi concluído e está operando.

Control_Heigth_Freezed (BOOL):

- Indica que o controle de altura está congelado.

Cut_Heigth_InMotion (BOOL):

- Indica que o CNC está posicionando o THC na altura de corte.

Pierce_Heigth_InMotion (BOOL):

- Indica que o CNC está posicionando o THC na altura de perfuração.

Voltage (INT):

- Mostra a tensão recebida pelo feedback analógico.

Wait_Arc_Transfer_Time (BOOL):

- Indica o tempo decorrido durante a transferência de arco.

Wait_Molten_Pool_Jump_Time (BOOL):

- Indica o tempo decorrido no processo de pular borra.

Wait_Pierce_Time (BOOL):

- Indica o tempo decorrido durante o processo de perfuração.

Fault_Arc_Transfer (BOOL):

- Indica que ouve alguma falha no momento que estava aguardando o tempo

para transferência de arco.

Error (BOOL):

- Este argumento retorna TRUE caso ocorra qualquer erro no processo (vide Error_Code).

Error_Code (INT):

- Quando um erro é sinalizado em "Error" este argumento recebe o código de erro gerado pelo bloco. O significado deste erro é apresentado na tabela de erros dos blocos.

9.2.4 Como se escreve

Height_Control(Enable, Arc_Transfered, Arc_Transfer_Time, Pierce_Height, Pierce_Time, Molten_Pool_ Jump_Time, Cut_Heigth, Cut_Heigth_Velocity, Start_Heigth_Control, Pierce_Height_Velocity, THC_ID_ Number, Reference_Voltage, FreezeControl, Disable_Higth_Control);

9.3 Homing

9.3.1 Funcionamento

Álém dos sinais de entrada do bloco é necessário setar os parâmetros Px42 e Px43 para máquinas principais e alternativa respectivamente.

Estes parâmetros definem a sequência em que os eixos buscarão referência. Devem ser programados valores de 1 a 8 para definir a ordem e caso seja programado 0 a referência deste eixo estará desabilitada.

Caso 2 eixos tenham o mesmo valor programado, o de menor número será referenciado primeiro.

9.3.2 Entradas

REF_Enable (bit):

- Habilitação do bloco. Habilita o processo de busca de referência. O sinal de sistema Reference_Mode é ativado e normalmente deve ser usado para manter o bloco habilitado. Caso enable caia no meio do processo de busca, o processo de busca é cancelado.

REF_Start (bit):

- A transição de FALSE para TRUE inicia a sequência de captura de referência de todos os eixos de acordo com os parâmetros e argumentos do bloco.

REF_Stop (bit):

- A transição de FALSE para TRUE neste sinal coloca o processo em pausa. Uma transição de FALSE para TRUE em Ref_Start retoma o processo no ponto em que foi interrompido.

REF_Abort (bit):

- Abortar a referência, caso queira cancel todo o processo e permite reiniciar a sequência a partir do primeiro eixo definido na ordem de busca.

REF_Target (bit):

- Caso FALSE: processo necessita de sinais indicando a posição desejada para a referência dos eixos (sensores de referência associados a REF_KEYx).

- Caso TRUE: processo assume que os eixos já estão posicionados de modo a capturar a marca de referência nestas posições (captura em uma volta do eixo). Para isso deve-se ter certeza de que os eixos estão posicionados corretamente ou então posicionar os eixos em JOG antes de iniciar a busca. Isso faz com que os eixos peguem a referência no local onde foram posicionados.

REF_ExeChannel (byte):

- Assossia a busca a um canal de execução no Proteo® PC. Existem 4 canais de execução de 0 a 3, cada canal de execução é um CNC distinto.

REF_KEYx (8 bits):

- Caso REF_Target igual FALSE deve-se associar um sinal de referência para cada eixo existente, num máximo de 8 eixos por canal de execução (REF_KEY1 a REF_KEY8).

9.3.3 Saidas

REF_Enabled (BOOL):

- Indica que o bloco está habilitado caso tenha sido inicializado sem erros.

REF_Running (BOOL):

- Indica que o CNC está executando o referenciamento em algum eixo.

REF_Paused (bit):

- Indica referenciamento pausado caso seja dado RER_Stop tenha sido acionado no meio do processo de busca de referência.

REF_Aborted (bit):

- Indica cancelamento do processo de busca de referência caso REF_Abort tenha sido acionado ou ocorra algum erro na execução do bloco.

REF_OK (bit):

- Memoriza status de referência OK ao encerrar o processo de busca de referência de todos os eixos. O sinal de sistema Reference_Mode é desativado, o que permite desabilitar o bloco de referenciamento.

Atenção: ao desabilitar o bloco este sinal não é cancelado.

REF_Error (BOOL):

- Este argumento retorna TRUE caso ocorra qualquer erro no processo (vide Error_Code).

REF_Error_Code (INT) – em processo de implementação:

- Quando um erro é sinalizado em "Error" este argumento recebe o código de erro gerado pelo bloco. O significado deste erro é apresentado na tabela de erros dos blocos.

REF_AxisID (byte):

- Retorna número do eixo que está executando o seu referenciamento. O valor é "0" caso nenhum eixo esteja executando referenciamento.

9.3.4 Como se escreve

Homming(REF_Enable, REF_Start, REF_Stop, REF_Abort, REF_Target, REF_ExeChannel, REF_KEY1, REF_KEY2, REF_KEY3, REF_KEY4, REF_KEY5, REF_KEY6, REF_KEY7, REF_KEY8);

9.4 Jog – Movimento manual dos eixos

9.4.1 Funcionamento

- O bloco permite executar movimentos nos eixos controlados pelo CNC no sentido indicado nos argumentos do bloco e com velocidade controlada. Esta velocidade é dada pelo argumento "Velocity".

- O movimento é mantido enquanto os argumentos que solicitam este movimento permanecerem TRUE.

9.4.2 Entradas

Enable (BOOL):

- Habilita o movimento manual dos eixos quando o valor for TRUE. Caso o valor esteja em FALSE o bloco é desabilitado e todas as saídas recebem valores nulos.

Axis_ID (INT):

- Este argumento recebe o número do eixo que se quer movimentar, valores 1 a 8, que identificam o eixo a ser movimentado. Estes valores correspondem às abas de parâmetros de eixos na tela de parâmetros do CNC.

Por exemplo, valor 1 corresponde ao eixo cujos parâmetros são definidos na aba Eixo 1 da tela de parâmetros do CNC.

ExeChannel (byte):

- Define o canal de execução do CNC ao qual o eixo está associado. Valores de 1 a 4.

Minus_Axis (BOOL):

- Solicita ao bloco executar o movimento no sentido negativo do eixo indicado em "Axis_ID". O movimento permanece habilitado enquanto este argumento estiver TRUE. Ao assumir o status FALSE o movimento é interrompido.

Plus_Axis (BOOL):

- Solicita ao bloco executar o movimento no sentido positivo do eixo indicado em "Axis_ID". O movimento permanece habilitado enquanto este argumento estiver TRUE. Ao assumir o status FALSE o movimento é interrompido.

Velocity (INT):

- Velocidade com que o movimento jog é executado quando solicitado.

9.4.3 Saídas

Enabled (BOOL):

- Indica que o bloco está habilitado e pronto para o uso.

InMotion (BOOL):

- Indica que o eixo associado ao "Axis_ID" está em movimento.

Error (BOOL):

- Este argumento retorna TRUE caso ocorra algum erro no processo (vide Error_Code).

Error_Code (INT):

- Quando um erro é sinalizado em "Error" este argumento recebe o código de erro gerado pelo bloco. O significado deste erro é apresentado na tabela de erros dos blocos.

9.4.4 Como se escreve

Jog(Enable, ExeChannel_ID, Plus_Axis, Minus_Axis, Axis_ID, Jog_Velocity);

9.5 Limit_Switch

9.5.1 Funcionamento

- Este bloco indica se o erro após acionado os flags positivo ou negativo é um alarme ou mensagem.

- Caso seja uma mensagem é indicado apenas mostrando TRUE nos argumentos de saída _Acted.

- Caso seja um alarme além dos argumentos _Acted também fica em TRUE "Alarm" e é solicidado um abort aos movimentos do CNC.

- Cada instância do bloco atua para um único eixo.

9.5.2 Entradas

Enable (BOOL):

- Para que o bloco seja executado é necessário que o argumento Enable esteja com valor TRUE. Caso o valor esteja em FALSE o bloco é desabilitado e todas as saídas recebem valores nulos.

Limit_Switch_NF (BOOL):

- Este argumento estabelece a polaridade na detecção de fim de curso. Caso TRUE, são identificadas falhas de fim de curso quando o argumento Negative_Switch ou Positive_Switch estiverem na condição FALSE. Caso este argumento esteja na condição FALSE, todos os erros são gerados quando o argumento Negative_ Switch ou Positive_Switch estiverem na condição TRUE.

Negative_Switch (BOOL):

- Devem ser atribuidos a este argumento todos os status dos sensores de fim de curso negativo da máquina.

- Caso Limit_Switch_NF esteja em TRUE os erros serão gerados quando este argumento estiver em FALSE.

Positive_Switch (BOOL):

- Devem ser atribuidos a este argumento todos os status dos sensores de fim de curso positivo da máquina.

- Caso Limit_Switch_NF esteja em TRUE os erros seram gerados quando este argumento estiver em FALSE.

Release (BOOL):

- Caso este argumento esteja em TRUE todos os erros gerados por este bloco são ignorados.

Clear_Alarm (BOOL):

- A transição de FALSE para TRUE concela a saída Alarm. Caso a condição que causou o alarme não seja retirada, Clear_Alarm não cancela o erro.

9.5.3 Saídas

Enabled (BOOL):

- Indica que o bloco está habilitado e pronto para o uso.

Alarm (BOOL):

- Indica que o erro gerado é um Alarm com isso é solicidato uma interrupção de movimento do CNC.

Negative_Switch_Acted (BOOL):

- Indica que um fim de curso negativo está acionado, ou seja, o argumento de entrada Negative_Switch está em condição de falha.

Positive_Switch_Acted (BOOL):

- Indica que um fim de curso positivo esta acionado, ou seja, o argumento de entrada Positive_Switch está em condição de falha.

9.5.4 Como se escreve

Limit_Switch(Enable, Clear_Alarm, Positive_Switch, Negative_Switch, Release, Limit_Switch_NF);

9.6 Move_Absolute

9.6.1 Funcionamento

- Move_Absolute é um Bloco Funcional por isso ele funciona como uma váriavel, ou seja, é necessário instânciar este objeto.

- Instância é um objeto, ou seja, uma cópia da estrutura principal, por isso deve ser criado uma variável com o nome que o aplicador queira e dizer que o tipo desta variável é do tipo "Move_Absolute".

- Solicita o movimento absoluto do eixo associado a "Axis".

9.6.2 Entradas

Enable (BOOL):

- Para que o bloco seja executado é necessário que o argumento Enable esteja com valor TRUE. Caso o valor esteja em FALSE o bloco é desabilitado e todas as saídas recebem valores nulos.

Axis (INT):

- Este argumento recebe o número do eixo que se quer movimentar, valores que podem ser programados de 1 a 8, identificamos qual o eixo a ser movimentado observando na tela de parâmetros qual o endereço programado em cada aba correspondente aos parâmetros de eixo.

Position (REAL):

- Posição absoluta onde será posicionado o eixo indicado em "Axis".

Start (BOOL):

- Solicita iniciar o movimento para a posição indicada em "Position".

Stop (BOOL):

- Solicita parar o movimento que estiver sendo executado.

Velocity (INT):

- Velocidade com que será executado o movimento.

ExeChannel_ID (byte):

- Canal de execução associa a busca a um canal de execução no Proteo® PC existem 4 canais de execução de 0 a 3, cada canal de execução é um CNC distinto.

9.6.3 Saídas

Enabled (BOOL):

- Indica que o bloco esta habilitado e pronto para o uso.

Aborted (BOOL):

- Indica que o um movimento que estava sendo executado foi interrompido.

Done (BOOL):

- Indica que um movimento solicitado foi concluído.

InMotion (BOOL):

- Indica que o eixo associado ao "Axis" esta em movimento.

Error (BOOL):

- Qualquer erro de execução este argumento retorna TRUE. (vide Error_Code).

Error_Code (INT):

- Quando um erro é sinalizado em "Error" neste argumento se recebe o código de erro gerado pelo bloco, o que este erro representa pode ser visto na tabela de erros dos blobos.

9.6.4 Como se escreve

Move_Absolute(Enable, Position, Axis, Start, Stop, ExeChannel_ID, Velocity);

9.7 Move_Relative

9.7.1 Funcionamento

- Move_Relative é um Bloco Funcional por isso ele funciona como uma váriavel, ou seja, é necessário instânciar este objeto.

- Instância é um objeto, ou seja, uma cópia da estrutura principal, por isso deve ser criado uma variável com o nome que o aplicador queira e dizer que o tipo desta variável é do tipo "Move_Relative".

- Solicita o movimento incremental do eixo associado a "Axis".

9.7.2 Entradas

Enable (BOOL):

- Para que o bloco seja executado é necessário que o argumento Enable esteja com valor TRUE.

Caso o valor esteja em FALSE o bloco é desabilitado e todas as saídas recebem valores nulos.

Axis (INT):

- Este argumento recebe o número do eixo que se quer movimentar, valores que podem ser programados de 1 a 8, identificamos qual o eixo a ser movimentado observando na tela de parâmetros qual o endereço programado em cada aba correspondente aos parâmetros de eixo.

Position (REAL):

- Valor do incremento de posição, este valor pode ser positivo para incrementar a posição, ou negativo para decrementar a posição atual.

Direction (BOOL):

- O incremento para valores positivos normalmente desloca o eixo para o sentido positivo, quando direction é colocado em TRUE o deslocamento passa a ser negativo para valores de incremento positivo.

Start (BOOL):

- Solicita iniciar o movimento para a incrementar/decrementar posição.

Stop (BOOL):

- Solicita parar o movimento que estiver sendo executado.

Velocity (INT):

- Velocidade com que será executado o movimento.

ExeChannel_ID (byte):

- Canal de execução associa a busca a um canal de execução no Proteo® PC existem 4 canais de execução de 0 a 3, cada canal de execução é um CNC distinto.

9.7.3 Saídas

Enabled (BOOL):

- Indica que o bloco esta habilitado e pronto para o uso.

Aborted (BOOL):

- Indica que o um movimento que estava sendo executado foi interrompido.

Done (BOOL):

- Indica que um movimento solicitado foi concluído.

InMotion (BOOL):

- Indica que o eixo associado ao "Axis" esta em movimento.

Error (BOOL):

- Qualquer erro de execução este argumento retorna TRUE. (vide Error_Code).

Error_Code (INT):

- Quando um erro é sinalizado em "Error" neste argumento se recebe o código de erro gerado pelo bloco, o que este erro representa pode ser visto na tabela de erros dos blobos.

9.7.4 Como se escreve

Move_Relative(Axis, Enable, ExeChannel_ID, Position, Start, Stop, Velocity, Direction);

9.8 Send_Keys

9.8.1 Funcionamento

- Send_Keys é um Bloco Funcional por isso ele funciona como uma váriavel, ou seja, é necessário instânciar este objeto.

- Instância é um objeto, ou seja, uma cópia da estrutura principal, por isso deve ser criado uma variável com o nome que o aplicador queira e dizer que o tipo desta variável é do tipo "Send_Keys".

- Este bloco solícita ao CNC reconhecer uma tecla, qualquer tecla pode ser enviada, foco, números, modo, ou teclas específicas do CNC.

9.8.2 Entradas

Enable (BOOL):

- Para que o bloco seja executado é necessário que o argumento Enable esteja com valor TRUE. Caso o valor esteja em FALSE o bloco é desabilitado e todas as saídas recebem valores nulos.

Key (INT):

- Código de tecla a ser reconhecida pelo CNC, pode ser qualquer tecla como foco, números, modo, ou teclas específicas do CNC.

Send (BOOL):

- Na transição de FALSE para TRUE é solicitado ao CNC reconhecer a tecla atribuada a "Key".

9.8.3 Saídas

Enabled (BOOL):

- Indica que o bloco esta habilitado e pronto para o uso.

Key_Done (BOOL):

- Indica que a tecla solicitada ao CNC foi enviada.

Error (BOOL):

- Qualquer erro de execução este argumento retorna TRUE. (vide Error_Code).

Error_Code (INT):

- Quando um erro é sinalizado em "Error" neste argumento se recebe o código de erro gerado pelo bloco, o que este erro representa pode ser visto na tabela de erros dos blobos.

9.8.4 Como se escreve

Send_Keys(Enable, Send, Key);

9.9 Send_State

9.9.1 Funcionamento

- Send_State é um Bloco Funcional por isso ele funciona como uma váriavel, ou seja, é necessário instânciar este objeto.

- Instância é um objeto, ou seja, uma cópia da estrutura principal, por isso deve ser criado uma variável com o nome que o aplicador queira e dizer que o tipo desta variável é do tipo "Send_State".

- Este bloco solícita ao CNC reconhecer um estado de softkeys que podem ser estados horizontais e verticais.

9.9.2 Entradas

Enable (BOOL):

- Para que o bloco seja executado é necessário que o argumento Enable esteja com valor TRUE. Caso o valor esteja em FALSE o bloco é desabilitado e todas as saídas recebem valores nulos.

Horizontal_State (INT):

- Estado horizontal que se quer reconhecer na árvore de softkeys.

Vertical_State (INT):

- Estado vertical que se quer reconhecer na árvore de softkeys.

Send (BOOL):

- Na transição de FALSE para TRUE é solicitado ao CNC reconhecer os estados solicitados.

9.9.3 Saídas

Enabled (BOOL):

- Indica que o bloco esta habilitado e pronto para o uso.

State_Done (BOOL):

- Indica que o estado solicitado ao CNC foi enviado.

Error (BOOL):

- Qualquer erro de execução este argumento retorna TRUE. (vide Error_Code).

Error_Code (INT):

- Quando um erro é sinalizado em "Error" neste argumento se recebe o código de erro gerado pelo bloco, o que este erro representa pode ser visto na tabela de erros dos blobos.

9.9.4 Como se escreve

Send_State(Enable, Horizontal_State, Vertical_State, Send);

9.10 TimerOFF

9.10.1 Funcionamento

- TimerOFF é um Bloco Funcional por isso ele funciona como uma váriavel, ou seja, é necessário instânciar este objeto.

- Instância é um objeto, ou seja, uma cópia da estrutura principal, por isso deve ser criado uma variável com o nome que o aplicador queira e dizer que o tipo desta variável é do tipo "TimerOFF".

- Quando habilitado o tempo definido em "Presset_Time" é iniciado, no caso do "TimerOFF" o status da saída "Output_Time" inicialmente fica em TRUE e ao terminar o tempo o status da saída vai para FALSE.

9.10.2 Entradas

Enable (BOOL):

- Para que o bloco seja executado é necessário que o argumento Enable esteja com valor TRUE. Caso o valor esteja em FALSE o bloco é desabilitado e todas as saídas recebem valores nulos.

Presset_Time (REAL):

- Tempo para ser contado no temporizador, a base de tempo é "ms".

Restart_Time(BOOL):

- Reinicia a contagem de tempo.

9.10.3 Saídas

Elapsed_Time (REAL):

- Tempo decorrido, mostra o tempo que ja se passou a partir do momento que o tempo ja se iniciou, a base de tempo é "ms".

Output_Time (BOOL):

- Indica o status da contagem de tempo.

9.10.4 Como se escreve

TimerOFF(Enable, Preset_Time, Restart_Time);

9.11 TimerON

9.11.1 Funcionamento

- TimerON é um Bloco Funcional por isso ele funciona como uma váriavel, ou seja, é necessário instânciar este objeto.

- Instância é um objeto, ou seja, uma cópia da estrutura principal, por isso deve ser criado uma variável com o nome que o aplicador queira e dizer que o tipo desta variável é do tipo "TimerON".

- Quando habilitado o tempo definido em "Presset_Time" é iniciado, no caso do "TimerON" o status da saída "Output_Time" inicialmente fica em FALSE e ao terminar o tempo o status da saída vai para TRUE e permanece em TRUE enquanto o argumento "Enable" estiver em TRUE.

9.11.2 Entradas

Enable (BOOL):

- Para que o bloco seja executado é necessário que o argumento Enable esteja com valor TRUE. Caso o valor esteja em FALSE o bloco é desabilitado e todas as saídas recebem valores nulos.

Presset_Time (REAL):

- Tempo para ser contado no temporizador, a base de tempo é "ms".

Restart_Time(BOOL):

- Reinicia a contagem de tempo.

9.11.3 Saídas

Elapsed_Time (REAL):

- Tempo decorrido, mostra o tempo que ja se passou a partir do momento que o tempo ja se iniciou, a base de tempo é "ms".

Output_Time (BOOL):

- Indica o status da contagem de tempo.

9.11.4 Como se escreve

TimerON(Enable, Preset_Time, Restart_Time);

9.12 Touch_Torch

Parâmetro de máquina

P968 = Entrada digital associada ao sensor ohmico do THC1

P969 = Entrada digital associada ao sensor ohmico do THC2

P970 = Entrada digital associada ao sensor ohmico do THC3

P971 = Entrada digital associada ao sensor ohmico do THC4

P974 = Lag para detecção de toque por torque.

9.12.1 Funcionamento

- Touch_Torch é um Bloco Funcional por isso ele funciona como uma váriavel, ou seja, é necessário instânciar este objeto.

- Instância é um objeto, ou seja, uma cópia da estrutura principal, por isso deve ser criado uma variável com o nome que o aplicador queira e dizer que o tipo desta variável é do tipo "Touch_Torch".

- Este bloco é responsável por executar a detecção de chapa e presset para zero na coodenada detectada.

- O processo consiste em mover o eixo no sentido negativo até que se encontre a chapa, após encontrada é feito um presset de coodenada para zero e após o presset é posicionado em uma altura de transferência.

- A chapa pode ser encontrada de 2 formas, por sensor ohmico ou torque.

- Quando por sensor ohmico assim que o sensor é acionado a coordenada do eixo em questão é pressetada para zero.

- O sensor ohmido é definido nos parâmetros "P968 para THC 1", "P969 para THC 2", "P970 para THC 3" e "P971 para THC 4".

- Quando por torque, ao ser detectado a chapa é somado um offset a coodenada do eixo em questão e então é feito o presset para zero.

- O processo de torque é feito através da detecção de "lag(P974)", quando o lag no processo de captura de chapa for maior que o definido em parâmetro é acionada a detecção por torque.

- O toque por torque sempre esta habilitado, o por sensor ohmico pode ser desabilitado por argumento de entrada do bloco.

9.12.2 Entradas

Enable (BOOL):

- Para que o bloco seja executado é necessário que o argumento Enable esteja com valor TRUE. Caso o valor esteja em FALSE o bloco é desabilitado e todas as saídas recebem valores nulos.

Enable_Ohmic_Sensor (BOOL):

- Este argumento habilita a detecção de chapa por sensor ohmico.

Pierce_Height (REAL):

- Posição absoluta onde será posicionado o THC para iniciar a perfuração da chapa.

Porc_Transfer_Heigth (INT):

- Argumento que recebe um valor de porcentagem, este argumento é utilizado para calcular a altura de "transferência".

- Altura de transferência é uma porcentagem de "Pierce_Height".

Start_Touch (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado o inicio do processo de toque que faz a captura da chapa.

Touch_Offset (REAL):

- Valor para ser somado a cota do eixo em questão no momento que o toque por torque for detectado.

Touch_Velocity (INT):

- Velocidade com que se desloca o eixo em questão para captura da chapa.

Transfer_Heigth_Velocity (INT):

- Velocidade com que o CNC posiciona o eixo em questão para posição de transferência.

THC_ID_Number (INT):

- Número do THC que se quer habilita, existem 4 THC possíveis, ou seja, podemos programa valores de 1 a 4.

9.12.3 Saídas

Enabled (BOOL):

- Indica que o bloco esta habilitado e pronto para o uso.

Done (BOOL):

- Indica que todo o precesso do bloco foi concluído.

Ohmic_Sensor_Enabled (BOOL):

- Quando este argumento em TRUE a detecção por sensor ohmico esta habilitada, quando em FALSE desabilita a captura por sensor ohmico.

- A captura deve ser desabilitada sempre que se depara com uma situaçao em que o sensor pode ser acionado de forma enganosa como por exemplo para cortes embaixo de água.

Running_Touch (BOOL):

- Este argumento indica que o processo de captura de chapa, ou seja, o movimento para o sentido negativo, mas ainda não detectada a chapa esta em curso.

Transfer_Heigth_InMotion (BOOL):

- Indica que o CNC esta posicionando o eixo em questão na posição de transferência.

Error (BOOL):

- Qualquer erro de execução este argumento retorna TRUE. (vide Error_Code).

Error_Code (INT):

- Quando um erro é sinalizado em "Error" neste argumento se recebe o código de erro gerado pelo bloco, o que este erro representa pode ser visto na tabela de erros dos blobos.

9.12.4 Como se escreve

Touch_Torch(Enable, Start_Touch, Touch_Velocity, Porc_Transfer_Heigth, Pierce_Hight, THC_ID_Number, Transfer_Heigth_Velocity, Enable_Ohmic_Sensor, Touch_Offset);

9.13 Counter_Up

9.13.1 Funcionamento

- Contador crescente, incrementa a saída do bloco (Counter) de acordo com o valor de "Counter_Base".

- Sempre que um pulso for dado em "Count" é incrementado uma unidade de "Counter_Base" em "Counter".

- Caso "Time_Interval" esteja em zero ao manter "Counter" em TRUE não é incrementado novamente, ou seja, a contagem fica apenas na transição de FALSE para TRUE de "Count".

- Caso "Time_Interval" esteja com valor diferente de 0 uma histerese (Time_Hysteresis_Counter) é aplicada a "Count" e a partir dai sempre que "Count" ficar em TRUE o valor é incrementado automáticamente no tempo dado em Time_Interval".

- "Time_Hysteresis_Counter" e "Time_Interval" são dados em unidade de milisegungos.

- Quando o valor de "Counter" for maior ou igual ao de "Preset" é indicado com o valor TRUE na saída "Done".

- Mesmo após "Done" igual a TRUE se "Count" for acionado o valor de "Counter" continua sendo incrementado até que seja dado um "Reset".

9.13.2 Entradas

Count (BOOL):

- Na transição de FALSE para TRUE incrementa o contador (Counter).

Reset (BOOL):

- Reinicia o contador, ou seja, "Counter" vai para o valor 0.

Preset (REAL):

- Valor alvo da contagem, o contador é incrementado até que chegue a este valor, após isso é indicado na saída do bloco(Done) que o valor alvo foi atingido.

Time_Hysteresis_Counter (INT):

- Define o tempo de histerese para "Count", histerese é o atraso para iniciar o incremento automático a partir do momento em que "Count" permanecer em TRUE.

Counter_Base (REAL):

- Base para incremento de "Counter".

- Os valores programados podem ser inteiros ou com casas decimais.

Time_Interval (INT):

- Intervalo entre incrementos de "Counter" a partir do momento que "Count" fica com valor TRUE.

- Caso este argumento em zero o incremento automático é ignorado.

9.13.3 Saídas

Done (BOOL):

- Indica que "Counter" atingiu o valor indicado em "Preset".

Counter (REAL):

- Valor acumulado da contagem, mesmo após chegar ao valor de "Preset" a contagem continua.

9.13.4 Como se escreve

Counter_Up(Count, Reset, Preset, Time_Hysteresis_Counter, Counter_Base, Time_Interval);

9.14 Counter_Down

9.14.1 Funcionamento

- Contador decrescente, decrementa a saída do bloco (Counter) de acordo com o valor de "Counter_Base".

- Sempre que um pulso for dado em "Count" é decrementado uma unidade de "Counter_Base" em "Counter".

- Caso "Time_Interval" esteja em zero ao manter "Counter" em TRUE não é decrementado novamente, ou seja, a contagem fica apenas na transição de FALSE para TRUE de "Count".

- Caso "Time_Interval" esteja com valor diferente de 0 uma histerese (Time_Hysteresis_Counter) é aplicada a "Count" e a partir dai sempre que "Count" ficar em TRUE o valor é decrementado automáticamente no tempo dado em "Time_Interval".

- "Time_Hysteresis_Counter" e "Time_Interval" são dados em unidade de milisegungos.

- Quando o valor de "Counter" for menor ou igual ao de "Preset" é indicado com o valor TRUE na saída "Done".

- Mesmo após "Done" igual a TRUE se "Count" for acionado o valor de "Counter" continua sendo decrementado até que seja dado um "Reset".

9.14.2 Entradas

Count (BOOL):

- Na transição de FALSE para TRUE decrementa o contador (Counter).

Reset (BOOL):

- Reinicia o contador, ou seja, "Counter" vai para o valor de Preset.

Preset (REAL):

- Valor inicial da contagem, o contador é decrementado até que chegue a zero, após isso é indicado na saída do bloco (Done) que o valor zero foi atingido.

Time_Hysteresis_Counter (INT):

- Define o tempo de histerese para "Count", histerese é o atraso para iniciar o decremento automático a partir do momento em que "Count" permanecer em TRUE.

Counter_Base (REAL):

- Base para decremento de "Counter".

- Os valores programados podem ser inteiros ou com casas decimais.

Time_Interval (INT):

- Intervalo entre decrementos de "Counter" a partir do momento que "Count" fica com valor TRUE.

- Caso este argumento em zero o decremento automático é ignorado.

9.14.3 Saídas

Done (BOOL):

- Indica que "Counter" atingiu o valor zero".

Counter (REAL):

- Valor acumulado da contagem, mesmo após chegar ao valor zero a contagem continua.

9.14.4 Como se escreve

Counter_Down(Count, Reset, Preset, Time_Hysteresis_Counter, Counter_Base, Time_Interval);

9.15 Counter_Down

9.15.1 Funcionamento

- Na transição de FALSE para TRUE de "Start" é inicada a contagem de tempo como a de um cronômetro, mostrando nas saídas do bloco os tempos em milisegundo, minutos e hora, os milisegundos são contados até 1000 e zerados, caso queira fazer contadores maiores em milisegundos basta ir somando esta variável a outra memoria definida pelo aplicador.

- Na transição de FALSE para TRUE de "Stop" o tempo é pausado podendo ser rotamada a contagem com um novo "Start".

- Na transição de FALSE para TRUE de "Reset" os tempos são zerados.

- Quando o valor dos presets de minutos e horas forem alcançados uma saída "Done" do bloco terá seu valor igual a TRUE, mesmo após "Done" igual a TRUE o tempo permanece contando até que se de um "Stop".

9.15.2 Entradas

Start (BOOL):

- Inicia a contagem de tempo.

Stop (BOOL):

- Pausa a contagem de tempo.

Reset (BOOL):

- Zera a contagem de tempo.

Preset_Minutes (INT):

- Valor alvo para minutos, quando o tempo chegar as condições de horas e minutos presetadas é indicado na saída (Done) do bloco.

Preset_Hours (INT):

- Valor alvo para horas, quando o tempo chegar as condições de horas e minutos presetadas é indicado na saída (Done) do bloco.

9.15.3 Saídas

MilliSeconds (INT):

- Mosta o tempo decorrido em mili segundos, quando este contador chegar a 1000 o valor é zerado.

Seconds (INT):

- Mostra o valor decorrido em segundos, quando este contador chegar a 60 ele é zerado.

Minutes (INT):

- Mostra o valor decorrido em minutos, quando este contador chegar a 60 ele é zerado.

Hours (INT):

- Mostra o valor decorrido em horas, este contador pode contar até 1092 horas antes de estourar a contagem.

Done (BOOL):

- Indica que o valor alvo em "Preset_Minutes" e "Preset_Hours" foi atingido.

9.15.4 Como se escreve

Counter_Time(Start, Stop, Reset, Preset_Minutes,Preset_Hours);

9.16 Date

9.16.1 Funcionamento

- Retorna para uso no PLC a data atual no real time clock do CNC.

9.16.2 Entradas

Refresh (BOOL):

- Solita atualizar as saídas do bloco com a data atual no CNC.

9.16.3 Saídas

Seconds (INT):

- Retorna os segundos do real time clock do CNC.

Minutes (INT):

- Retorna os minutos do real time clock do CNC.

Hours (INT):

- Retorna as horas do real time clock do CNC.

Day (INT):

- Retorna o dia do real time clock do CNC.

Month (INT):

- Retorna os meses do real time clock do CNC.

Year (INT):

- Retorna o ano do real time clock do CNC.

Como se escreve

Date(Refresh);

9.17 TimerONOFF

9.17.1 Funcionamento

Funcionamento

- TimerONOFF é um Bloco Funcional por isso ele funciona como uma váriavel, ou seja, é necessário instânciar

este objeto.

- Instância é um objeto, ou seja, uma cópia da estrutura principal, por isso deve ser criado uma variável

com o nome que o aplicador queira e dizer que o tipo desta variável é do tipo "TimerONOFF".

- Quando habilitado o tempo definido em "Preset_Time_ON" é iniciado(Output_Time = TRUE), quando este tempo é encerrado a saída

"Output_Time" vai para "FALSE" ao encerrar o tempo defenido em "Preset_Time_OFF" é reiniciada a contagem.

- Para que a contagem seja encerrada é necessário desabilitar o bloco, ou seja, "Enable = FALSE".

9.17.2 Entradas

Enable(BOOL):

- Para que o bloco seja executado é necessário que o argumento Enable esteja com valor TRUE.

Caso o valor esteja em FALSE o bloco é desabilitado e todas as saídas recebem valores nulos.

Preset_Time_OFF(REAL):

- Tempo para ser contado no temporizador com a saída "Output_Time" em "FALSE", a base de tempo é "ms".

Preset_Time_ON(REAL):

- Tempo para ser contado no temporizador com a saída "Output_Time" em "TRUE", a base de tempo é "ms".

9.17.3 Saidas

Elapsed_Time(REAL):

- Tempo decorrido, mostra o tempo que ja se passou a partir do momento que o tempo ja se iniciou

(Tempo total soma entre "Preset_Time_OFF" e "Preset_Time_ON"), a base de tempo é "ms".

Output_Time(BOOL):

- Indica o status da contagem de tempo.

9.17.4 Como se escreve

TimerONOFF(Enable, Preset_Time_ON, Preset_Time_OFF);

9.18 Remote_ManWheel

9.18.1 Funcionamento

9.18.2 Entradas

9.18.3 Saidas

9.19 Read_Data_Drive

9.19.1 Funcionamento

9.19.2 Entradas

9.19.3 Saidas

9.20 Write_Data_Drive

9.20.1 Funcionamento

9.20.2 Entradas

9.20.3 Saidas

9.21 Block_Keys

9.21.1 Funcionamento

9.21.2 Entradas

9.21.3 Saidas

9.22 Manage_ProgramCNC

9.22.1 Funcionamento

9.22.2 Entradas

9.22.3 Saidas

10 Funções

10.1 Alarme

10.1.1 Funcionamento

- Aprensenta uma mensagem (Alarme) com a cor vermelha no topo da tela do CNC, essa mensagem permanece enquanto "BOOL_STATE" estiver em TRUE, uma vez que "BOOL_STATE" passe para FALSE é necessário que se pressione a telca CE para que esta mensagem em vermelho seja cancelada.

- Podemos utilizar até 64 alarmes na aplicação PLC onde os indices vão de 0 a 63;

10.1.2 Entradas

INT_NUMBER:

- Número da mensagem assossiada ao ID definido na tela de alarmes e mensagens.

BOOL_STATE:

- A lógica assossiada a esse campo deve retornar um valor TRUE OR FALSE para aprensentar ou não a mensagem assossiada.

10.1.3 Como se escreve

ALARM(INT_NUMBER, BOOL_STATE);

10.2 Blink_Message

10.2.1 Funcionamento

- Aprensenta uma mensagem com a cor amarela no topo da tela do CNC, essa mensagem

permanece piscando enquanto "BOOL_STATE" estiver em TRUE, uma vez que "BOOL_STATE" passe para FALSE a menssagem é cancelada.

- Podemos utilizar até 32 mensagens na aplicação PLC onde os indices vão de 0 a 31.

10.2.2 Entradas

INT_NUMBER:

- Número da mensagem assossiada ao ID definido na tela de alarmes e mensagens.

BOOL_STATE:

- A lógica assossiada a esse campo deve retornar um valor TRUE OR FALSE para aprensentar ou não a mensagem assossiada.

10.2.3 Como se escreve

BLINK_MESSAGE(INT_NUMBER, BOOL_STATE);

Delete this text and replace it with your own content.

10.3 Message

Funcionamento

- Aprensenta uma mensagem com a cor amarela no topo da tela do CNC, essa mensagem

permanece piscando enquanto "BOOL_STATE" estiver em TRUE, uma vez que BOOL_STATE" passe para FALSE a menssagem é cancelada.

- Podemos utilizar até 32 mensagens na aplicação PLC onde os indices vão de 0 a 31;

Entradas

INT_NUMBER:

- Número da mensagem assossiada ao ID definido na tela de alarmes e mensagens.

BOOL_STATE:

- A lógica assossiada a esse campo deve retornar um valor TRUE OR FALSE

para aprensentar ou não a mensagem assossiada.

Como se escreve

BLINK_MESSAGE(INT_NUMBER, BOOL_STATE);

10.4 Timed_Message

10.4.1 Funcionamento

- Aprensenta uma mensagem com a cor amarela no topo da tela do CNC, essa mensagem

é aparesentada por 4 segundos na transisão de FALSE para TRUE de "BOOL_STATE".

- Podemos utilizar até 32 mensagens na aplicação PLC onde os indices vão de 0 a 31;

10.4.2 Entradas

INT_NUMBER:

- Número da mensagem assossiada ao ID definido na tela de alarmes e mensagens.

BOOL_STATE:

- A lógica assossiada a esse campo deve retornar um valor TRUE OR FALSE

para aprensentar ou não a mensagem assossiada.

10.4.3 Como se escreve

TIMED_MESSAGE(INT_NUMBER, BOOL_STATE);

10.5 Change_SFK

Funcionamento

- Altera o status de uma softkey "TOGGLE" ou "ONOFF", ou seja, podemos forçar esta softkeys para o estado pressionado ou solto.

10.5.1 Entradas

Change (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado alterar o status da softkey.

Memory_Bit (INT):

- Deve ser associado o número do bit da softkey que se quer alterar, este número pode variar de 0 a 31. È associado este bit as propriedades de softkeys no módulo Studio do Ativo, propriedade "Bit de memória".

State (BOOL):

- Status que se quer alterar, pode ser TRUE ou FALSE.

10.5.2 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.
- Esta função sempre retorna o status atual sa softkey apontada em "Memory_Bit".

10.5.3 Como se escreve

Change_SFK(State, Memory_Bit, Change);

10.6 Key_Code

10.6.1 Funcionamento

- Compara se o código associado a entrada do bloco é igual a uma tecla enviada pelo CNC.
- Todos os códigos que o CNC envia são de teclas, ou seja, softkeys pressionadas ou qualquer tecla do teclado do CNC que for pressionada.

10.6.2 Entradas

Code (INT):

- Código para ser compado a todos os códigos de tecla que o CNC enviar.

10.6.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.
- O retorno será TRUE caso o código definido em "Code" for igual ao código que o CNC enviar.
- O retorno será FALSE sempre que o código que o CNC enviar for diferente de "Code".

10.6.4 Como se escreve

Key_Code (Code);

10.7 Function_M

10.7.1 Funcionamento

- Compara se o código associado a entrada do bloco é igual a uma função M chamada em um programa CNC.

- Caso a saída do bloco seja TRUE signigica que a função que se quer monitorar (Function_Code) foi chamada em um programa CNC, caso contrário o retorno da função é FALSE.

- Caso o argumento "Hold_Function_M" esteja em TRUE quando for detectado que a função chamada é igual a que se quer monitorar, a execução do CNC fica parada na linha da função M até que o argumento "Hold_ Function_M" caia para FALSE.

10.7.2 Entradas

Function_Code (INT):

- Neste argumento se programa qual o número da função M que se quer monitorar.

- Caso queira saber se existe uma função independente do código chamar a função com a constante Any_ Function_MST.

Hold_Function_M (BOOL):

- Neste argumento se programa TRUE ou FALSE

- Se programa TRUE caso queira segurar a execução do programa CNC quando detectar que a função "M" monitorada foi chamada.

- Caso FALSE o não segura a execução do programa CNC.

10.7.3 Saída

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.

- O retorno será TRUE caso a função "M" chamada em um programa CNC for igual ao número monitorado no argumento "Function_Code".

- O retorno será FALSE sempre que a função for diferente do argumento "Function_Code".

10.7.4 Como se escreve

Function_M(Hold_Function_M, Function_Code);

10.8 Function_S

10.8.1 Funcionamento

- Compara se o código associado a entrada do bloco é igual a uma função S chamada em um programa CNC.

- Caso a saída do bloco seja TRUE signigica que a função que se quer monitorar (Function_Code) foi chamada em um programa CNC, caso contrário o retorno da função é FALSE.

- Caso o argumento "Hold_Function_S" esteja em TRUE quando for detectado que a função chamada é igual a que se quer monitorar, a execução do CNC fica parada na linha da função S até que o argumento "Hold_ Function_S" caia para FALSE.

10.8.2 Entradas

Function_Code (INT):

- Neste argumento se programa qual o número da função S que se quer monitorar.

- Caso queira saber se existe uma função independente do código chamar a função com a constante Any_ Function_MST.

Hold_Function_S (BOOL):

- Neste argumento se programa TRUE ou FALSE

- Se programa TRUE caso queira segurar a execução do programa CNC quando detectar que a função "S" monitorada foi chamada.

- Caso FALSE o não segura a execução do programa CNC.

10.8.3 Saída

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.

- O retorno será TRUE caso a função "S" chamada em um programa CNC for igual ao número monitorado no argumento "Function_Code".

- O retorno será FALSE sempre que a função for diferente do argumento "Function_Code".

10.8.4 Como se escreve

Function_S(Hold_Function_S, Function_Code);

10.9 Function_T

10.9.1 Funcionamento

- Compara se o código associado a entrada do bloco é igual a uma função T chamada em um programa CNC.

- Caso a saída do bloco seja TRUE signigica que a função que se quer monitorar (Function_Code) foi chamada em um programa CNC, caso contrário o retorno da função é FALSE.

- Caso o argumento "Hold_Function_T" esteja em TRUE quando for detectado que a função chamada é igual a que se quer monitorar, a execução do CNC fica parada na linha da função T até que o argumento "Hold_ Function_T" caia para FALSE.

10.9.2 Entradas

Function_Code (INT):

- Neste argumento se programa qual o número da função T que se quer monitorar.

- Caso queira saber se existe uma função independente do código chamar a função com a constante Any_ Function_MST.

Hold_Function_T (BOOL):

- Neste argumento se programa TRUE ou FALSE

- Se programa TRUE caso queira segurar a execução do programa CNC quando detectar que a função "T" monitorada foi chamada.

- Caso FALSE o não segura a execução do programa CNC.

10.9.3 Saída

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.

- O retorno será TRUE caso a função "T" chamada em um programa CNC for igual ao número monitorado no argumento "Function_Code".

- O retorno será FALSE sempre que a função for diferente do argumento "Function_Code".

10.9.4 Como se escreve

Function_T(Hold_Function_T, Function_Code);

10.10 Preset_A

10.10.1 Funcionamento

- Executa um preset de coordenada do eixo A.

- Preset é assumir a coordenada indicada em "Coordinate".

10.10.2 Entradas

Coordinate (REAL):

- Coordenada para o qual se quer presetar o eixo A.

Origin (INT):

- Podemos fazer este preset em 6 origens diferentes que são de 54 a 59.

- Se pode observar estas origens entrando na tela de "Origens" que é o objeto 361 de telas do CNC.

- 361: código de tecla do CNC que solicita mostrar a tela de origens.

Preset (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado o preset para alterar a coordenada do eixo A.

10.10.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.

- Esta função retorna TRUE por um ciclo de PLC no momento que a solicitação de preset foi aceita.

10.10.4 Como se escreve

Preset_A(Coordinate, Preset, Origin);

10.11 Preset_B

10.11.1 Funcionamento

- Executa um preset de coordenada do eixo B.

- Preset é assumir a coordenada indicada em "Coordinate".

10.11.2 Entradas

Coordinate (REAL):

- Coordenada para o qual se quer presetar o eixo B.

Origin (INT):

- Podemos fazer este preset em 6 origens diferentes que são de 54 a 59.
- Se pode observar estas origens entrando na tela de "Origens" que é o objeto 361 de telas do CNC.
- 361: código de tecla do CNC que solicita mostrar a tela de origens.

Preset (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado o preset para alterar a coordenada do eixo B.

10.11.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.
- Esta função retorna TRUE por um ciclo de PLC no momento que a solicitação de preset foi aceita.

10.11.4 Como se escreve

Preset_B(Coordinate, Preset, Origin);

10.12 Preset_C

10.12.1 Funcionamento

- Executa um preset de coordenada do eixo C.
- Preset é assumir a coordenada indicada em "Coordinate".

10.12.2 Entradas

Coordinate (REAL):

- Coordenada para o qual se quer presetar o eixo C.

Origin (INT):

- Podemos fazer este preset em 6 origens diferentes que são de 54 a 59.
- Se pode observar estas origens entrando na tela de "Origens" que é o objeto 361 de telas do CNC.
- 361: código de tecla do CNC que solicita mostrar a tela de origens.

Preset (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado o preset para alterar a coordenada do eixo C.

10.12.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.
- Esta função retorna TRUE por um ciclo de PLC no momento que a solicitação de preset foi aceita.

10.12.4 Como se escreve

Preset_C(Coordinate, Preset, Origin);

10.13 Preset_U

10.13.1 Funcionamento

- Executa um preset de coordenada do eixo U.
- Preset é assumir a coordenada indicada em "Coordinate".

10.13.2 Entradas

Coordinate (REAL):

- Coordenada para o qual se quer presetar o eixo U.

Origin (INT):

- Podemos fazer este preset em 6 origens diferentes que são de 54 a 59.
- Se pode observar estas origens entrando na tela de "Origens" que é o objeto 361 de telas do CNC.
- 361: código de tecla do CNC que solicita mostrar a tela de origens.

Preset (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado o preset para alterar a coordenada do eixo U.

10.13.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.
- Esta função retorna TRUE por um ciclo de PLC no momento que a solicitação de preset foi aceita.

10.13.4 Como se escreve

Preset_U (Coordinate, Preset, Origin);

10.14 Preset_V

10.14.1 Funcionamento

- Executa um preset de coordenada do eixo V.
- Preset é assumir a coordenada indicada em "Coordinate".

10.14.2 Entradas

Coordinate (REAL):

- Coordenada para o qual se quer presetar o eixo V.

Origin (INT):

- Podemos fazer este preset em 6 origens diferentes que são de 54 a 59.
- Se pode observar estas origens entrando na tela de "Origens" que é o objeto 361 de telas do CNC.
- 361: código de tecla do CNC que solicita mostrar a tela de origens.

Preset (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado o preset para alterar a coordenada do eixo V.

10.14.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.
- Esta função retorna TRUE por um ciclo de PLC no momento que a solicitação de preset foi aceita.

10.14.4 Como se escreve

Preset_V(Coordinate, Preset, Origin);

10.15 Preset_W

10.15.1 Funcionamento

- Executa um preset de coordenada do eixo W.
- Preset é assumir a coordenada indicada em "Coordinate".

10.15.2 Entradas

Coordinate (REAL):

- Coordenada para o qual se quer presetar o eixo W.

Origin (INT):

- Podemos fazer este preset em 6 origens diferentes que são de 54 a 59.
- Se pode observar estas origens entrando na tela de "Origens" que é o objeto 361 de telas do CNC.
- 361: código de tecla do CNC que solicita mostrar a tela de origens.

Preset (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado o preset para alterar a coordenada do eixo W.

10.15.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.

- Esta função retorna TRUE por um ciclo de PLC no momento que a solicitação de preset foi aceita.

10.15.4 Como se escreve

Preset_W(Coordinate, Preset, Origin);

10.16 SFK_Memory_Status

10.16.1 Funcionamento

- Altera o status de uma softkey "TOGGLE" ou "ONOFF", ou seja, podemos forçar esta softkeys para o estado pressionado ou solto.

10.16.2 Entradas

Memory_Bit (INT):

- Deve ser associado o número do bit da softkey que se quer alterar, este número pode variar de 0 a 31. È associado este bit as propriedades de softkeys no módulo Studio do Ativo, propriedade "Bit de memória".

10.16.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.

- Esta função sempre retorna o status atual sa softkey apontada em "Memory_Bit".

10.16.4 Como se escreve

SFK_Memory_Status(Memory_Bit);

10.17 Preset_X

10.17.1 Funcionamento

- Executa um preset de coordenada do eixo X.

- Preset é assumir a coordenada indicada em "Coordinate".

10.17.2 Entradas

Coordinate (REAL):

- Coordenada para o qual se quer presetar o eixo X.

Origin (INT):

- Podemos fazer este preset em 6 origens diferentes que são de 54 a 59.
- Se pode observar estas origens entrando na tela de "Origens" que é o objeto 361 de telas do CNC.
- 361: código de tecla do CNC que solicita mostrar a tela de origens.

Preset (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado o preset para alterar a coordenada do eixo X.

10.17.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.
- Esta função retorna TRUE por um ciclo de PLC no momento que a solicitação de preset foi aceita.

10.17.4 Como se escreve

Preset_X(Coordinate, Preset, Origin);

10.18 Preset_Y

10.18.1 Funcionamento

- Executa um preset de coordenada do eixo Y.
- Preset é assumir a coordenada indicada em "Coordinate".

10.18.2 Entradas

Coordinate (REAL):

- Coordenada para o qual se quer presetar o eixo Y.

Origin (INT):

- Podemos fazer este preset em 6 origens diferentes que são de 54 a 59.
- Se pode observar estas origens entrando na tela de "Origens" que é o objeto 361 de telas do CNC.
- 361: código de tecla do CNC que solicita mostrar a tela de origens.

Preset (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado o preset para alterar a coordenada do eixo Y.

10.18.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.
- Esta função retorna TRUE por um ciclo de PLC no momento que a solicitação de preset foi aceita.

10.18.4 Como se escreve

Preset_Y(Coordinate, Preset, Origin);

10.19 Preset_Z

10.19.1 Funcionamento

- Executa um preset de coordenada do eixo Z.
- Preset é assumir a coordenada indicada em "Coordinate".

10.19.2 Entradas

Coordinate (REAL):

- Coordenada para o qual se quer presetar o eixo Z.

Origin (INT):

- Podemos fazer este preset em 6 origens diferentes que são de 54 a 59.
- Se pode observar estas origens entrando na tela de "Origens" que é o objeto 361 de telas do CNC.
- 361: código de tecla do CNC que solicita mostrar a tela de origens.

Preset (BOOL):

- Quando este argumento sofre uma transição de FALSE para TRUE é solicitado o preset para alterar a coordenada do eixo Z.

10.19.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.
- Esta função retorna TRUE por um ciclo de PLC no momento que a solicitação de preset foi aceita.

10.19.4 Como se escreve

Preset_Z(Coordinate, Preset, Origin);

10.20 Block_SFK

10.20.1 Funcionamento

- Bloqueia uma softkeys, ou seja, ao pressiona-la não é enviado ao PLC os códigos associados a esta softkey.

10.20.2 Entradas

Memory_Bit (INT):

- Deve ser associado o número do bit da softkey que se quer bloquear, este número pode variar de 0 a 31.

È associado este bit as propriedades de softkeys no módulo Studio do Ativo™ , propriedade "Bit de habilitação".

10.20.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.
- Esta função sempre retorna o status do bloqueio solicitado.

10.20.4 Como se escreve

Block_SFK(Memory_Bit, Condition);

10.21 R_EDGE

10.21.1 Funcionamento

- Detecta a transição de subida, ou seja, transição de FALSE para TRUE de um booleano.
- Retorna TRUE por um ciclo de PLC.

10.21.2 Entradas

BOOL:

- Detecta a transição de FALSE para TRUE deste argumento, booleano pode ser uma comparação, ou o retorno de uma outra função e etc...

10.21.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.
- Retorna TRUE por um ciclo de PLC, quando condições de entrada forem satisfeitas.

10.21.4 Como se escreve

R_EDGE(BOOL)

10.22 F_EDGE

10.22.1 Funcionamento

- Detecta a transição de descida, ou seja, transição de TRUE para FALSE de um booleano.
- Retorna TRUE por um ciclo de PLC.

10.22.2 Entradas

BOOL:

- Detecta a transição de TRUE para FALSE deste argumento, booleano pode ser uma comparação, ou o retorno de uma outra função e etc...

10.22.3 Saídas

- Esta função tem um retorno booleano, ou seja, sempre retorna TRUE ou FALSE.

- Retorna TRUE por um ciclo de PLC, quando condições de entrada forem satisfeitas.

10.22.4 Como se escreve

F_EDGE(BOOL)

10.23 R_PARAM

10.23.1 Funcionamento

- Retorna o valor de um parâmetro de máquina ou de um parâmetro de PLC.

- Parâmetro PLC retornado por esta função são os contidos na aba PLC

da tela de parâmetros.

10.23.2 Entradas

INT:

- Número do parâmetro que se quer ler, podem ser programados valores de 0 a 999 que são os números de parâmetros existentes.

10.23.3 Saídas

- Retorna o valor do parametro indicado no argumento “INT”, o valor retonado sempre é um valor “REAL”.

10.23.4 Como se escreve

R_PARAM(INT)

10.24 W_PARAM

10.24.1 Funcionamento

- Altera o valor do parâmetro indicado na entrada da função.

10.24.2 Entradas

INT:

- Número do parâmetro que se quer alterar o valor, podem ser programados valores de 0 a 999 que são os números de parâmetros existentes.

REAL:

- Valor a ser atribuido ao parâmetro indicado no argumento “INT”.

10.24.3 Saídas

Esta função não tem retorno.

10.24.4 Como se escreve

W_PARAM(INT, REAL)

10.25 R_HVAR

10.25.1 Funcionamento

- Retorna o valor de uma variavel H.

- Variaveis H são as memórias utilizadas para ciclos fixos que são acessadas diretamente pelos programas de CNC (ISO).

10.25.2 Entradas

INT:

- Número da variavel H que se quer ler, podem ser programados valores de 0 a 1023 que são o número de variaveis H existentes.

10.25.3 Saídas

- Retorna o valor de uma variavel H indicado no argumento “INT”, o valor retonado sempre é um valor “REAL”.

10.25.4 Como se escreve

R_HVAR(INT)

10.26 W_PARAM

10.26.1 Funcionamento

- Altera o valor da variavel H indicado na entrada da função.

10.26.2 Entradas

INT:

- Número da variavel H que se quer alterar o valor, podem ser programados valores de 0 a 1023 que são os números de parâmetros existentes.

REAL:

- Valor a ser atribuido a variavel H indicado no argumento “INT”.

10.26.3 Saídas

Esta função não tem retorno.

10.26.4 Como se escreve

W_PARAM(INT, REAL)

10.27 R_COMMAND_VAR

10.27.1 Funcionamento

- Retorna o valor de uma variavel reservada.

- Variaveis reservadas são as memórias utilizadas para ler informações ou para solicitar funções ao CNC.

- Uma lista das variaveis reservadas existentes pode ser visto no documento “Variaveis Reservadas”.

10.27.2 Entradas

INT:

- Número da variavel reservada que se quer ler, podem ser programados valores de 10000 a 19999 que são o número de variaveis reservadas existentes, nem todas posições de memoria executa alguma ação.

10.27.3 Saídas

- Retorna o valor da variavel reservada indicada no argumento “INT”, o valor retonado sempre é um valor “REAL”.

10.27.4 Como se escreve

R_COMMAND_VAR(INT)

10.28 W_COMMAND_VAR

10.28.1 Funcionamento

- Altera o valor de uma variavel reservada.
- Variaveis reservadas são as memórias utilizadas para ler informações ou para solicitar funções ao CNC.
- Uma lista das variaveis reservadas existentes pode ser visto no documento “Variaveis Reservadas”.

10.28.2 Entradas

INT:

- Número da variavel reservada que se quer alterar o valor, podem ser programados valores de 0 a 19999 que são os números de variaveis reservadas existentes.

REAL:

- Valor a ser atribuido a variavel reservada indicada no argumento “INT”.

10.28.3 Saídas

- Não retorna valor.

10.28.4 Como se escreve

W_COMMAND_VAR(INT, REAL)

Sobre Kollmorgen

Fabricante nacional de CNCs, com mais de 30 anos de atuação no mercado brasileiro, a MCS Engenharia foi adquirida em 2013 pela Kollmorgen, empresa com mais de 60 anos no mercado de controle de movimento, presente no Brasil desde 2007, oferecendo soluções inovadoras em termos de confiabilidade, desempenho e facilidade de uso.

Através do conhecimento global em movimento e qualidade, é líder de mercado e tem profundo conhecimento em associar e integrar produtos padronizados e personalizados. Fornecemos aos OEMs a vantagem competitiva de que precisam para ter sucesso.

Nossa infra-estrutura, conhecimento, paixão e experiência são provas da nossa busca pelo movimento perfeito. E por causa de nossa herança de customização, vemos oportunidades, não obstáculos.

Todos os dias, ao redor do mundo, exploramos os limites do movimento. Veja como fazemos.

Nossa experiência é incomparável

Com conhecimento em aplicação e customização rápida e prototipagem, Kollmorgen supera os outros em ajudar você a construir o equipamento diferenciado e colocá-lo ao mercado mais rapidamente. As nossas soluções combinam software de programação, serviços de engenharia e componentes best-in-class de movimento para uma solução única.

Oferecemos a maior variedade de produtos da Indústria

Produtos padrão, modificados e personalizados - possibilita a mais ampla gama de soluções para sua necessidade. Pode utilizar os nossos sistemas integrados ou componentes para aperfeiçoar e reduzir o tempo de desenvolvimento. A melhor solução muitas vezes não é clara. Mas nossa experiência em aplicação nos permite modificar produtos padrão ou desenvolver soluções totalmente personalizadas em toda a nossa linha de produtos.

Somos seu parceiro global com recursos locais

Aproveitando uma equipe de mais de 1.800 funcionários e mais de 60 anos de experiência em aplicações para minimizar os riscos e fornecer ótimos componentes de movimento para sua máquina. Temos excelentes centros de engenharia e atendimento ao cliente em todas as principais regiões do mundo. Temos uma cadeia global de suprimentos com baixo custo de produção em todo o mundo para conduzir excelente custo-benefício, continuidade e prontidão. Nossos recursos são incomparáveis.

Estes são os fatos e a nossa filosofia: Acreditamos que o maximizar o movimento seja o diferencial de sua máquina e do seu equipamento.

South America

MCS Kollmorgen

Avenida Tamboré, 1077

Barueri - São Paulo

Internet: www.kollmorgen.com

Tel: +55 - 11 - 4191-4026