

# **MasterTool® Programming Manual de Programação da Série Ponto**

Rev. F 09/2004

Cód. Doc: MP399101

Nenhuma parte deste documento pode ser copiada ou reproduzida de alguma forma sem o consentimento prévio e por escrito da ALTUS Sistemas de Informática S.A., que reserva-se o direito de efetuar alterações sem prévio comunicado.

Conforme legislação vigente no Brasil, do Código de Defesa do Consumidor, informamos os seguintes aspectos relacionados com a segurança de pessoas e instalações do cliente:

Os equipamentos de automação industrial, fabricados pela ALTUS, são robustos e confiáveis devido ao rígido controle de qualidade a que são submetidos. No entanto, equipamentos eletrônicos de controle industrial (controladores programáveis, comandos numéricos, etc.) podem causar danos às máquinas ou processos por eles controlados, no caso de defeito em suas partes e peças, erros de programação ou instalação, podendo inclusive colocar em risco vidas humanas.

O usuário deve analisar as possíveis consequências destes defeitos e providenciar instalações adicionais externas de segurança que, em caso de necessidade, atuem no sentido de preservar a segurança do sistema, principalmente nos casos da instalação inicial e de testes.

É imprescindível a leitura completa dos manuais e/ou características técnicas do produto, antes da instalação ou utilização do mesmo.

A ALTUS garante os seus equipamentos contra defeitos reais de fabricação pelo prazo de doze meses a partir da data da emissão da nota fiscal. Esta garantia é dada em termos de manutenção de fábrica, ou seja, o transporte de envio e retorno do equipamento até a fábrica da ALTUS, em Porto Alegre, RS, Brasil, ocorrerá por conta do cliente. A garantia será automaticamente suspensa caso sejam introduzidas modificações nos equipamentos por pessoal não autorizado pela ALTUS. A ALTUS exime-se de quaisquer ônus referentes a reparos ou substituições em virtude de falhas provocadas por agentes externos aos equipamentos, pelo uso indevido dos mesmos, bem como resultantes de caso fortuito ou por força maior.

A ALTUS garante que seus equipamentos funcionam de acordo com as descrições contidas explicitamente em seus manuais e/ou características técnicas, não garantindo a satisfação de algum tipo particular de aplicação dos equipamentos.

A ALTUS desconsiderará qualquer outra garantia, direta ou implícita, principalmente quando se tratar de fornecimento de terceiros.

Pedidos de informações adicionais sobre o fornecimento e/ou características dos equipamentos e serviços ALTUS, devem ser feitos por escrito. A ALTUS não se responsabiliza por informações fornecidas sobre seus equipamentos sem registro formal.

### DIREITOS AUTORAIS

Série Ponto, MasterTool e QUARK são marcas registradas da ALTUS Sistemas de Informática S.A. IBM é marca registrada da International Business Machines Corporation.

# Sumário

<b>PREFÁCIO</b>	<b>1</b>
<b>DESCRIÇÃO DESTE MANUAL</b>	<b>1</b>
<b>DOCUMENTOS DA SÉRIE PONTO</b>	<b>1</b>
<b>TERMINOLOGIA</b>	<b>2</b>
<b>CONVENÇÕES UTILIZADAS</b>	<b>3</b>
<b>SUPORTE TÉCNICO</b>	<b>4</b>
<b>REVISÕES DESTE MANUAL</b>	<b>5</b>
<b>INTRODUÇÃO</b>	<b>6</b>
<b>A LINGUAGEM DE PROGRAMAÇÃO</b>	<b>6</b>
<b>LINGUAGEM DE DIAGRAMA DE RELÉS</b>	<b>7</b>
<b>ELEMENTOS DE PROGRAMAÇÃO</b>	<b>7</b>
<b>ORGANIZAÇÃO DE MEMÓRIA DOS CPS DA SÉRIE PONTO</b>	<b>7</b>
<b>LÓGICAS</b>	<b>8</b>
<b>OPERANDOS</b>	<b>9</b>
IDENTIFICAÇÃO DE UM OPERANDO PELO ENDEREÇO	9
IDENTIFICAÇÃO DE UM OPERANDO PELO TAG	9
OPERANDOS UTILIZADOS NO MASTERTOOL	10
IDENTIFICAÇÃO DOS OPERANDOS SIMPLES	10
IDENTIFICAÇÃO DOS OPERANDOS CONSTANTE	11
IDENTIFICAÇÃO DOS OPERANDOS TABELA	12
OPERANDOS %E - RELÉS DE ENTRADA	13
OPERANDOS %S - RELÉS DE SAÍDA	13
OPERANDOS %A - RELÉS AUXILIARES	14
OPERANDOS %M - MEMÓRIAS	14
OPERANDOS %D - DECIMAIS	15
OPERANDOS %F - REAIS	16
OPERANDOS %I - INTEIRO	17
OPERANDOS %KM, %KI, %KD E %KF - CONSTANTES	17
OPERANDOS %TM, %TI, %TD E %TF - TABELAS	18
ACESSO INDIRETO	19
DECLARAÇÃO DE OPERANDOS	20
OPERANDOS RETENTIVOS	21
<b>INSTRUÇÕES</b>	<b>22</b>
RESTRIÇÕES QUANTO AO USO DE INSTRUÇÕES NOS CPS	23
REPRESENTAÇÃO GRÁFICA DAS INSTRUÇÕES	24
DESCRIÇÃO DA SINTAXE DAS INSTRUÇÕES	25
RESTRIÇÕES QUANTO AO POSICIONAMENTO DAS INSTRUÇÕES	25
<b>PROJETO DE PROGRAMAÇÃO</b>	<b>27</b>
ESTRUTURAÇÃO DE UM PROJETO DE PROGRAMAÇÃO	27
ESTADOS DE OPERAÇÃO DO CP	30
EXECUÇÃO DO PROJETO DE PROGRAMAÇÃO	32
ELABORAÇÃO DE PROJETOS DE PROGRAMAÇÃO	33

DEPURAÇÃO DE PROJETOS DE PROGRAMAÇÃO	37
TEMPOS DE CICLO DE EXECUÇÃO DO PROGRAMA	44
NÍVEIS DE PROTEÇÃO DO CP	45
INTERTRAVAMENTO DE COMANDOS NO CP	46

---

<b>INSTRUÇÕES</b>	<b>48</b>
-------------------	-----------

---

<b>LISTA DAS INSTRUÇÕES</b>	<b>48</b>
CONVENÇÕES UTILIZADAS	48
INSTRUÇÕES DO GRUPO RELÉS	50
CONTATOS	51
BOBINAS	52
SLT - BOBINA DE SALTO	53
PLS - RELÉ DE PULSO	55
RM, FRM - RELÉ MESTRE, FIM DE RELÉ MESTRE	56
INSTRUÇÕES DO GRUPO MOVIMENTADORES	57
MOV - MOVIMENTAÇÃO DE OPERANDOS SIMPLES	58
MOP - MOVIMENTAÇÃO DE PARTES (SUBDIVISÕES) DE OPERANDOS	59
MOB - MOVIMENTAÇÃO DE BLOCOS DE OPERANDOS	61
MOT - MOVIMENTAÇÃO DE TABELAS	63
CAB - CARREGA BLOCO	65
INSTRUÇÕES DO GRUPO ARITMÉTICAS	69
SOM - ADIÇÃO	69
SUB - SUBTRAÇÃO	71
MUL - MULTIPLICAÇÃO	72
DIV - DIVISÃO	73
AND - E BINÁRIO ENTRE OPERANDOS	74
OR - OU BINÁRIO ENTRE OPERANDOS	76
XOR - OU EXCLUSIVO ENTRE OPERANDOS	78
CAR - CARREGA OPERANDOS	80
INSTRUÇÕES DE COMPARAÇÃO DE OPERANDOS - IGUAL, MAIOR E MENOR	81
INSTRUÇÕES DO GRUPO CONTADORES	84
CON - CONTADOR SIMPLES	85
COB - CONTADOR BIDIRECIONAL	86
TEE - TEMPORIZADOR NA ENERGIZAÇÃO	87
TED - TEMPORIZADOR NA DESENERGIZAÇÃO	88
INSTRUÇÕES DO GRUPO CONVERSORES	89
B/D - CONVERSÃO BINÁRIO-DECIMAL	90
D/B - CONVERSÃO DECIMAL-BINÁRIO	91
INSTRUÇÕES DO GRUPO GERAL	92
LDI - LIGA/DESLIGA INDEXADO	93
TEI - TESTE DE ESTADO INDEXADO	95
SEQ - SEQUENCIADOR	97
CHP - CHAMA MÓDULO PROCEDIMENTO	101
CHF - CHAMA MÓDULO FUNÇÃO	102
ECH - ESCRITA DE OPERANDOS EM OUTRO CP PARA ETHERTNET	105
LTH - LEITURA DE OPERANDOS DE OUTRO CP PARA ETHERNET	110
LAH - LIBERA ATUALIZAÇÃO DE IMAGENS DOS OPERANDOS PARA ETHERNET	112
INSTRUÇÕES DO GRUPO LIGAÇÕES	113
LGH - LIGAÇÃO HORIZONTAL	113
LGN - LIGAÇÃO NEGADA	113
LGV - LIGAÇÃO VERTICAL	113

---

<b>MÓDULOS FUNÇÃO</b>	<b>114</b>
-----------------------	------------

---

<b>F-PID.033 - FUNÇÃO CONTROLE PID</b>	<b>115</b>
INTRODUÇÃO	115
PROGRAMAÇÃO	116
<b>F-RAIZN.034 - FUNÇÃO RAIZ QUADRADA</b>	<b>120</b>
INTRODUÇÃO	120
PROGRAMAÇÃO	120
<b>F-ARQ2.035 A F-ARQ31.042 - FUNÇÕES ARQUIVO DE DADOS</b>	<b>122</b>
INTRODUÇÃO	122
PROGRAMAÇÃO	122
<b>F-MOBT.043 - FUNÇÃO PARA MOVIMENTAÇÃO DE BLOCOS DE OPERANDOS TABELA</b>	<b>126</b>
INTRODUÇÃO	126
PROGRAMAÇÃO	126
<b>F-RELG.048 - FUNÇÃO PARA ACESSO AO RELÓGIO DE TEMPO REAL</b>	<b>128</b>
INTRODUÇÃO	128
PROGRAMAÇÃO	128
<b>F-PID16.056 - MÓDULO F PARA CONTROLE PID</b>	<b>131</b>
INTRODUÇÃO	131
PROGRAMAÇÃO	133
OPERANDOS	133
ENTRADAS E SAÍDAS	133
CARACTERÍSTICAS DE FUNCIONAMENTO	134
DESSATURACÃO DA AÇÃO INTEGRAL	134
MODO MANUAL	134
CONTROLE DIRETO E REVERSO	134
INTERVALO DE AMOSTRAGEM	134
TEMPO DE EXECUÇÃO	135
DESCRIÇÃO DAS POSIÇÕES DA TABELA DE PARÂMETROS	135
DESCRIÇÃO DO OPERANDO %A DE CONTROLE	136
NOTAS DE APLICAÇÃO	137
SUGESTÕES PARA AJUSTES DO CONTROLADOR PID	140
DETERMINAÇÃO DAS CONSTANTES DO CONTROLADOR ATRAVÉS DO PERÍODO E DO GANHO CRÍTICO	140
DETERMINAÇÃO DAS CONSTANTES DO CONTROLADOR ATRAVÉS DAS CONSTANTES DO PROCESSO	141
GANHOS X ESCALAS	143
EXEMPLO DE APLICAÇÃO	145
UTILIZAÇÃO DA F-PID16.056	148
COMPARAÇÃO COM O F-PID.033	148
<b>F-CTRL.059 - MÓDULO F PARA CONTROLE AVANÇADO</b>	<b>150</b>
INTRODUÇÃO	150
PROGRAMAÇÃO	154
<b>F-NORM.071 - FUNÇÃO PARA NORMALIZAÇÃO</b>	<b>157</b>
INTRODUÇÃO	157
PROGRAMAÇÃO	157
<b>F-COMPF.072 - FUNÇÃO PARA MÚLTIPLAS COMPARAÇÕES</b>	<b>160</b>
INTRODUÇÃO	160
PROGRAMAÇÃO	160
<b>F-AES.087 - FUNÇÃO PARA ATUALIZAÇÃO IMEDIATA DE ENTRADAS E SAÍDAS</b>	<b>162</b>
INTRODUÇÃO	162
PROGRAMAÇÃO	162
<b>F-ANDT.090, F-ORT.091 E F-XORT.092 - FUNÇÕES DE OPERAÇÕES LÓGICAS ENTRE OPERANDOS TABELA</b>	<b>164</b>
INTRODUÇÃO	164
PROGRAMAÇÃO	164
<b>F-STCP.044 - FUNÇÃO STATUS DO UCP</b>	<b>166</b>
INTRODUÇÃO	166
PROGRAMAÇÃO	166
<b>F-NEGT.093 - FUNÇÃO PARA NEGAÇÃO LÓGICA DE OPERANDOS TABELA</b>	<b>171</b>

INTRODUÇÃO	171
PROGRAMAÇÃO	171
<b><u>GLOSSÁRIO</u></b>	<b><u>173</u></b>
GLOSSÁRIO DA SÉRIE PONTO	173
GLOSSÁRIO DE REDES	173
GLOSSÁRIO GERAL	174
PRINCIPAIS ABREVIATURAS	177

# Prefácio

## Descrição deste Manual

Este manual apresenta a linguagem de programação utilizada nos controladores programáveis da Série Ponto da ALTUS, bem como orientações para a elaboração de programas aplicativos. Foi escrito supondo-se familiaridade com a utilização de microcomputadores padrão IBM-PC® e ambiente operacional Windows™.

O software programador MT4000 ou MT4100, referido a partir deste ponto como MasterTool® Programming ou simplesmente MasterTool, foi desenvolvido para a programação em linguagem de relés e blocos das séries de controladores programáveis Série Ponto da ALTUS.

Este manual está dividido em 4 capítulos e glossário.

O capítulo 1, **Introdução**, apresenta as características principais da programação dos CPs da Série Ponto da ALTUS.

O capítulo 2, **Programação da Linguagem de Diagramas e Relés**, apresenta os componentes da linguagem.

O capítulo 3, **Referência das Instruções**, descreve a função e a sintaxe de todas as instruções das linguagens.

O capítulo 4, **Referência dos Módulos Função**, descreve a função e a programação dos parâmetros de entrada e saída dos módulos função fornecidos pela ALTUS.

## Documentos da Série Ponto

Para obter informações adicionais sobre a Série Ponto podem ser consultados outros documentos (manuais e características técnicas) além deste. Estes documentos encontram-se disponíveis em [www.altus.com.br](http://www.altus.com.br).

Cada produto possui um documento denominado Característica Técnica (CT), e é neste documento que encontram-se as características do produto em questão. Caso o produto possua mais informações, ele pode ter também um manual de utilização (o código do manual é citado na CT).

Por exemplo, o módulo PO2022 tem todas as informações de características, de utilização e de compra, na sua CT. Por outro lado, o PO5063 possui, além da CT, um manual de utilização.

Aconselha-se os seguintes documentos como fonte de informação adicional:

- Características técnicas de cada produto
- Manual de Utilização do Programador MasterTool
- Manual de Utilização de UCPs Ponto

# Terminologia

Neste manual, as palavras “software”, “hardware” “mouse”, “tag” e “wire-info” são empregadas livremente, por sua generalidade e frequência de uso. Por este motivo, apesar de serem vocábulos em inglês, aparecerão no texto sem aspas.

As seguintes expressões são empregadas com frequência no texto do manual. Por isso, a necessidade de serem conhecidas para uma melhor compreensão.

- **CP:** Controlador Programável - entendido como um equipamento composto por uma UCP, módulos de entrada e saída e fonte de alimentação
- **UCP:** Unidade Central de Processamento, é o módulo principal do CP, que realiza o processamento dos dados

A palavra “módulo”, quando se referir a hardware, é utilizada para denominar cada um dos componentes de um equipamento.

A palavra “módulo”, quando se referir a software, é utilizada para denominar cada um dos componentes de um programa aplicativo.

Outras expressões podem ser encontradas no apêndice A, **Glossário**.



# Convenções Utilizadas

Os símbolos utilizados ao longo deste manual possuem os seguintes significados:

- Este marcador indica uma lista de itens ou tópicos.

maiúsculas PEQUENAS indicam nomes de teclas, por exemplo ENTER.

TECLA1+TECLA2 é usado para teclas a serem pressionadas simultaneamente. Por exemplo, a digitação simultânea das teclas CTRL e END é indicada como CTRL+END.

TECLA1, TECLA2 é usado para teclas a serem pressionadas sequencialmente. Por exemplo, a mensagem “Digite ALT, F10” significa que a tecla ALT deve ser pressionada e liberada e então a tecla F10 pressionada e liberada.

MAIÚSCULAS GRANDES indicam nomes de arquivos e diretórios.

Itálico indica palavras e caracteres que são digitados no teclado ou vistos na tela. Por exemplo, se for solicitado a digitar A:MASTERTOOL, estes caracteres devem ser digitados exatamente como aparecem no manual.

**NEGRITO** é usado para nomes de comandos ou opções, ou para enfatizar partes importantes do texto.

As mensagens de advertência apresentam os seguintes formatos e significados:

## **PERIGO:**

O rótulo **PERIGO** indica que risco de vida, danos pessoais graves ou prejuízos materiais substanciais resultarão se as precauções necessárias não forem tomadas.

## **CUIDADO:**

O rótulo **CUIDADO** indica que risco de vida, danos pessoais graves ou prejuízos materiais substanciais podem resultar se as precauções necessárias não forem tomadas.

## **ATENÇÃO:**



O rótulo **ATENÇÃO** indica que danos pessoais ou prejuízos materiais mínimos podem resultar se as precauções necessárias não forem tomadas.

## Suporte Técnico

Para acessar o Suporte Técnico ligue para (51) 5899500 em São Leopoldo, RS, ou para o Suporte Técnico mais próximo conforme a página da Altus na INTERNET:

- [www.altus.com.br](http://www.altus.com.br)
- E-MAIL: [altus@altus.com.br](mailto:altus@altus.com.br)

Caso o equipamento já esteja instalado, é aconselhável providenciar as seguintes informações antes de entrar em contato:

- Modelos de equipamentos utilizados e configuração do sistema instalado
- Número de série da UCP, revisão do equipamento e versão do software executivo, constantes na etiqueta fixada na sua lateral
- Informações do modo de operação da UCP, obtidas através do programador MASTERTOOL conteúdo do programa aplicativo, obtido através do comando **Comunicação, Módulos** do MasterTool Programming ou clicando em 
- Conteúdo do programa aplicativo (módulos), obtido através do programador MASTERTOOL
- Versão do programador utilizado que pode ser obtido a partir do comando Ajuda, Sobre MasterTool Programming ou selecionando o botão 

# Revisões deste Manual

O código de referência, da revisão e a data do presente manual estão indicados na capa. A mudança da revisão pode significar alterações da especificação funcional ou melhorias no manual.

O histórico a seguir lista as alterações correspondentes a cada revisão deste manual:

Revisão: A	Data: 11/2001
Aprovação: Luiz Gerbase	
Autor: Dimitrius Biroth Rocha	

Observações:

- Versão inicial

Revisão: B	Data: 02/2003
Aprovação: Luiz Gerbase	
Autor: Dimitrius Biroth Rocha e Rosana Casais	

Observações:

- Incluídos operandos reais (%F, %TF e %KF);
- Incluídos Módulos F\_CTRL.059 e F\_PID16.056;
- Incluído Módulo C Estendido.

Revisão: C	Data: 11/2003
Aprovação: Luiz Gerbase	
Autor: Dimitrius Biroth Rocha	

Observações:

- Inclusão das UCPs PO3042 e PO3142.

Revisão: D	Data: 02/2004
Aprovação: Luiz Gerbase	
Autor: Dimitrius Biroth Rocha	

Observações:

- Inclusão das UCPs PO3242 e PO3342.
- Inclusão das instruções ECH, LTH e LAH.

Revisão: E	Data: 06/2004
Aprovação: Luiz Gerbase	
Autor: Dimitrius Biroth Rocha	

Observações:

- Atualização de módulos F.

Revisão: F	Data: 09/2004
Aprovação: Luiz Gerbase	
Autor: Dimitrius Biroth Rocha	

Observações:

- Incluídos operandos inteiros 32 bits (%I, %TI e %KI);

# Introdução

Bem-vindo a Linguagem de Relés e Blocos ALTUS, a linguagem que permite a construção de programas aplicativos para os CPs ALTUS a partir do MasterTool Programming.

O programa aplicativo tem como objetivo a execução de tarefas de controle. Este programa, quando carregado no controlador programável (CP), faz com que este passe a exercer as funções de controle da máquina ou processo para o qual está sendo programado.

## A Linguagem de Programação

Os controladores programáveis surgiram para substituir painéis de controle a relés. Neste contexto, uma linguagem de programação que mais se aproximasse da experiência de técnicos e engenheiros seria a solução mais adequada para desenvolvimento de programas aplicativos de CPs.

Em vista disso, as instruções disponíveis para a construção do programa aplicativo no MasterTool são programadas em linguagem de relés e blocos, muito semelhante à linguagem de contatos elétricos e bobinas, utilizadas na descrição dos painéis de controle a relé.

A principal vantagem da utilização deste tipo de linguagem é seu rápido aprendizado, pois assemelha-se muito com os esquemas elétricos convencionais.

O acompanhamento e verificação de funcionamento de um programa aplicativo é similar ao de um esquema elétrico, com a vantagem de visualizar o estado dos contatos e bobinas na janela do MasterTool.

# Linguagem de Diagrama de Relés

Este capítulo descreve a linguagem de Relés e Blocos ALTUS detalhando os elementos da linguagem, a estruturação modular de um programa aplicativo e a função de cada módulo.

Ao final da leitura deste capítulo será possível estruturar um programa aplicativo bem como realizar a configuração de CPs e roteadores.

## Elementos de Programação

Um programa aplicativo é composto por 4 elementos básicos:

- módulos
- lógicas
- instruções
- operandos

Um programa aplicativo é composto por diversos **módulos**, permitindo uma melhor estruturação das rotinas de acordo com as suas funções. Os módulos são programados em linguagem de relés, seguindo a tendência mundial de normatização nesta área.

Um módulo de programa aplicativo é dividido em **lógicas de programação**. O formato de uma lógica de programa aplicativo utilizado nos CPs da Série Ponto permite até oito elementos em série e até quatro caminhos em paralelo.

As **instruções** são utilizadas para executar determinadas tarefas por meio de leituras e/ou alterações do valor dos operandos.

Os operandos identificam diversos tipos de variáveis e constantes utilizadas na elaboração de um programa aplicativo, podendo ter seu valor modificado de acordo com a programação realizada. Como exemplo de variáveis pode-se citar pontos de E/S e memórias contadoras.

Cada elemento componente do programa aplicativo é explicado em detalhes nas seções seguintes.

## Organização de Memória dos CPs da Série Ponto

O programa aplicativo é armazenado no controlador em uma área de memória dividida em bancos. Podem existir um ou mais bancos de memória RAM e flash EPROM, conforme o modelo do CP e a sua configuração de memória, cada banco possuindo 16, 32 ou 64 Kbytes.

Neste manual, na ajuda do MasterTool e no programador MasterTool, o nome EPROM refere-se indistintamente à memória para gravação permanente do programa aplicativo utilizada no CP, seja do tipo cartucho de EPROM ou flash EPROM.

Na janela do diretório de módulos do CP (opções Comunicação, Módulos) é possível visualizar a quantidade de memória livre em cada banco, para cada tipo existente no controlador. Ver Opção Módulos na seção Comando Comunicação no capítulo 4.

Os valores dos operandos numéricos (%M, %D, %F, %TM, %TD e %TF) são armazenados em área separada do programa, com diferentes tamanhos de acordo com o modelo do CP. Pode-se consultar a quantidade de memória de operandos livre na janela de edição do módulo C no quadro de operandos.

Para maiores informações sobre a janela de edição do módulo C, ver seção Janelas de Edição, no Manual de Utilização.

Os operandos binários (%E, %S e %A) possuem área permanentemente reservada para os seus valores na memória interna do microprocessador.

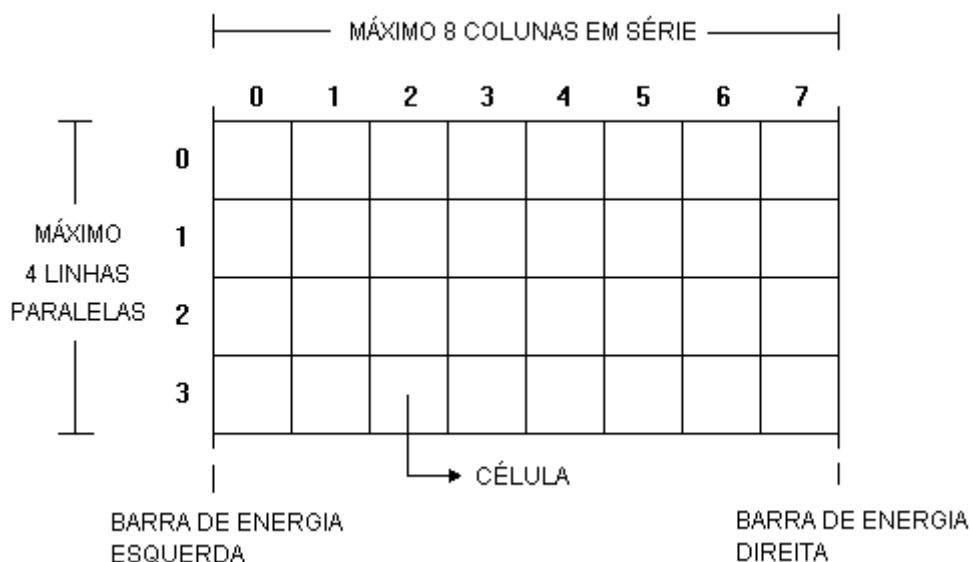
O uso da memória de operandos é apresentada em detalhes na seção **Declaração de Operandos**, neste mesmo capítulo.

Para maiores informações sobre as capacidades e a organização de memória de cada controlador, consultar os seus respectivos Manuais de Utilização (ver seção **Manuais Relacionados**, no prefácio deste manual).

## Lógicas

Chama-se **lógica** a matriz de programação formada por 32 células (elementos da matriz) dispostas em 4 linhas (0 a 3) e 8 colunas (0 a 7). Em cada uma das células podem ser colocadas instruções, podendo-se programar até 32 instruções em uma mesma lógica.

Cada lógica presente no programa simula um pequeno trecho de um diagrama de relés real. A figura 2-1 mostra o formato de uma lógica do programa aplicativo.



**Figura 2-1 Formato de uma Lógica**

As duas linhas laterais da lógica representam barras de energia entre as quais são colocadas as instruções a serem executadas.

Estão disponíveis para a programação instruções simbólicas tipicamente encontradas em diagramas, tais como contatos, bobinas, ligações e instruções representadas em caixas, como temporizadores, contadores e aritméticas.

A lógica deve ser programada de forma que bobinas e entradas das instruções de caixas sejam "energizadas" a partir do fechamento de um fluxo de "corrente" da esquerda para a direita entre as duas barras, através de contatos ou das saídas das caixas interligados. Entretanto, o fluxo de "corrente elétrica" simulado em uma lógica flui somente no sentido da barra de energia esquerda para a direita, diferentemente dos esquemas elétricos reais. O conceito utilizado simplifica sobremaneira o projeto lógico de relés, uma vez que não é necessário a preocupação com caminhos de fuga de corrente.

O processamento das instruções de uma lógica é realizado em colunas, desde a coluna 0 até a 7. Uma coluna é processada na ordem sequencial de suas linhas, desde a linha 0 até a linha 3. A figura 2-2

mostra a ordem de processamento das células da lógica. O número existente dentro de cada célula indica a sua ordem de processamento.

	0	1	2	3	4	5	6	7
0	1	5	9	13	17	21	25	29
1	2	6	10	14	18	22	26	30
2	3	7	11	15	19	23	27	31
3	4	8	12	16	20	24	28	32

Figura 2-2 Ordem de Processamento das Células na Lógica

## Operandos

Operandos são elementos utilizados pelas instruções do MasterTool na elaboração de um programa aplicativo. Os operandos podem definir valores constantes, definidos no momento da programação, ou variáveis, identificadas através de um endereço ou de um tag, com valores possíveis de serem alterados durante a execução do programa aplicativo.

### Identificação de um Operando pelo Endereço

A identificação e utilização de um operando pelo seu endereço é caracterizada pelo caractere % como primeiro caractere do nome. O restante do nome utilizado deve seguir às regras de formatação de endereço de operandos.

O formato de cada operando pode ser visto na seção Identificação de Operandos Simples e nas subseqüentes, neste mesmo capítulo.

### Identificação de um Operando pelo Tag

A identificação e utilização de um operando pelo seu tag é caracterizada pela utilização de um nome, com até 7 caracteres (alfanuméricos), que pode ser atribuído a qualquer operando, exceto constantes. Este nome passa a representar o operando nos processos de programação, monitoração, depuração e documentação de um programa aplicativo.

O MasterTool não permite a utilização de TAGs para operandos do tipo constante (%KM, %KI, %KD ou %KF).

Ex.:

Atribui-se o tag CONT1 ao operando %M0000.

Sempre que o operando %M0000 necessite ser utilizado na edição do programa aplicativo, pode-se utilizar o seu tag CONT1.

©DICA:

A escolha do nome do tag para o operando deve refletir ao máximo a função que o conteúdo do operando executa no programa aplicativo. Ex.: **TANQUE1**, armazena o volume do tanque 1.

A identificação de um operando pelo seu endereço poderá ser feita sempre, uma vez que todo operando possui um endereço. A identificação de um operando pelo seu tag, somente poderá ser feita após à atribuição do tag a um operando.

A atribuição de tags à operandos pode ser feita através do comando Operandos do menu Relatório ou diretamente no momento da programação. No segundo caso, ao preencher o nome de um operando de uma instrução com um tag não existente, é indicada a inexistência da definição do tag, e solicitado o tipo de operando para qual o tag deve ser criado.

Para maiores informações sobre criação e atribuição de tags para operandos, ver seções sobre o comando Relatório, Operandos, no capítulo 4 e Inserindo Tags e Comentários de Operandos, no Manual de Utilização do MasterTool.

Os operandos também podem ser visualizados através de seu wire-info associado. No entanto, um operando não pode ser forçado ou monitorado digitando-se o wire-info ao invés do tag ou endereço.

## Operandos Utilizados no MasterTool

São mostrados na tabela 2-1 os operandos disponíveis no MasterTool:

Tipo	Operando
%E	Relés de Entrada
%S	Relés de Saída
%A	Relés Auxiliares
%M	Memórias
%I	Inteiros
%D	Decimais
%F	Reais
%KM	Constantes Memórias
%KD	Constantes Decimais
%KF	Constantes Reais
%TM	Tabelas Memórias
%TI	Tabelas Inteiros
%TD	Tabelas Decimais
%TF	Tabelas Reais

Tabela 2-1 Operandos Utilizados no MasterTool

Os operandos dividem-se em 3 grupos:

- operandos simples
- operandos constante
- operandos tabela

## Identificação dos Operandos Simples

Os operandos simples são utilizados como variáveis de armazenamento de valores no programa aplicativo. Conforme a instrução que os utilizam, eles podem ser referenciados na sua totalidade ou em uma subdivisão (uma parte do operando). As subdivisões de operandos podem ser **palavra**, **octeto**, **nibble** ou **ponto**.

O formato geral de um operando simples pode ser visto na figura 2-3.

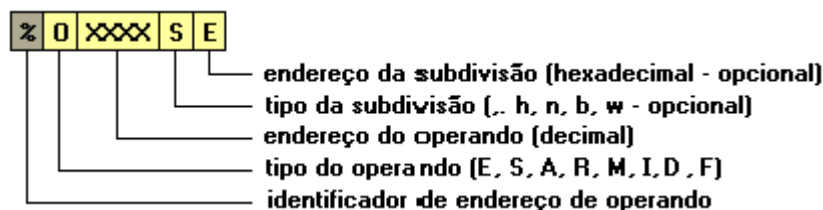


Figura 2-3 Formato de um Operando Simples



Tipo do operando:

- **%E** - entrada
- **%S** - saída
- **%A** - auxiliar
- **%M** – memória
- **%I** - inteiro
- **%D** - decimal
- **%F** - real

Tipo da subdivisão:

- **.** - ponto da palavra baixa (1 ponto)
- **h** - ponto da palavra alta (1 ponto)
- **n** - nibble (4 pontos)
- **b** - octeto (8 pontos)
- **w** - palavra (16 pontos)

Exemplos de Endereços:

- **%E0002.3** - ponto 3 do operando de entrada 2
- **%S0004.7** - ponto 7 do operando de saída 4
- **%A0039n1** - nibble 1 do operando auxiliar 39
- **%A0045** - octeto auxiliar 45
- **%I0234** – operando inteiro 234
- **%M0205** - operando memória 205
- **%M0205b0** - octeto 0 da memória 205
- **%D0029** - operando decimal 29
- **%D0034w1** - palavra 1 do operando decimal 34
- **%F0001** – operando real 1

Exemplos de tags:

- **FORNO**
- **LIMSUP**
- **CHAVE1**

## Identificação dos Operandos Constante

Os operandos constante são utilizados para a definição de valores fixos durante a edição do programa aplicativo.

O formato geral de um operando constante pode ser visto na figura 2-4.

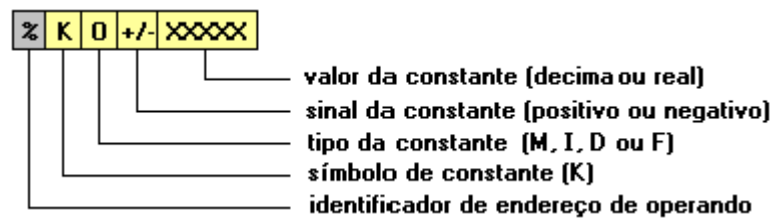


Figura 2-4 Formato de um Operando Constante

Tipo da constante:

- **%M** memória
- **%I** inteiro
- **%D** decimal
- **%F** real

Exemplos:

- **%KM+05172** - constante memória positiva
- **%KI-1** – constante inteira negativa
- **%KD-0974231** - constante decimal negativa
- **%KF+0153.78** – constante real positiva

## Identificação dos Operandos Tabela

Tabelas de operandos são conjuntos de operandos simples, constituindo arranjos unidimensionais. São utilizados índices para determinar a posição da tabela que se deseja ler ou alterar. São possíveis tabelas de operandos memória ou decimal.

O formato geral de um operando tabela pode ser visto na figura 2-5.

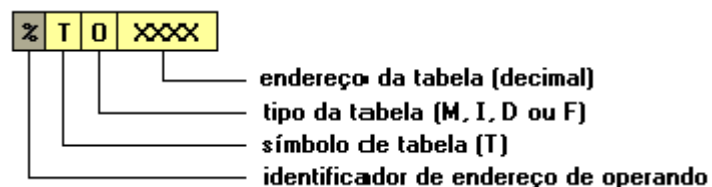


Figura 2-5 Formato de um Operando Tabela

Tipo da tabela:

- **%TM** memória
- **%TI** inteiro
- **%TD** decimal
- **%TF** real

Exemplos:

- **%TM0026** - tabela memória 26
- **%TI0020** – tabela inteiro 20
- **%TD0015** - tabela decimal 15

- %TF0069 – tabela real 69

## Operandos %E - Relés de Entrada

São operandos usados para referenciar pontos de módulos digitais de entrada. Sua quantidade é determinada pelo número de módulos de E/S que estão dispostos nos bastidores que compõem o sistema. Ver item **Configurando o Barramento** na seção **Configurando o Módulo C** no Manual de Utilização do MasterTool.

Os operandos %E são normalmente utilizados em instruções binárias (contatos, bobinas) e de movimentação. Ocupam um byte de memória (8 bits), armazenando os valores dos pontos diretamente em cada bit. Os valores dos operandos são armazenados na memória interna do microprocessador, não utilizando o espaço disponível ao programa aplicativo.

Os formatos dos operandos %E podem ser vistos na figura 2-6.

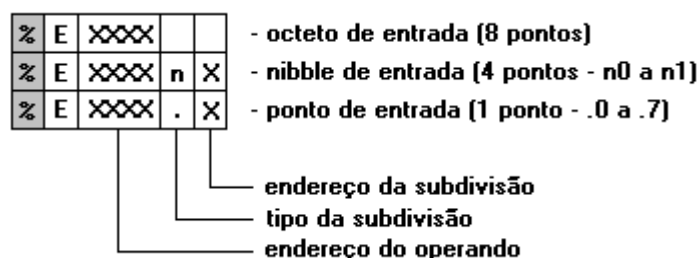


Figura 2-6 Formatos dos Operandos %E

Exemplos:

- %E0018.6 - ponto 6 do octeto de entrada 18
- %E0021n0 - nibble 0 do octeto de entrada 21
- %E0025 - octeto de entrada 25

## Operandos %S - Relés de Saída

São operandos usados para referenciar pontos de módulos digitais de saída. Sua quantidade é determinada pelo número de módulos de E/S que estão dispostos nos bastidores que compõem o sistema. Ver item **Configurando o Barramento** na seção **Configurando o Módulo C** no Manual de Utilização do MasterTool.

Os operandos %S são utilizados em instruções binárias (contatos, bobinas) e de movimentação. Ocupam um byte de memória (8 bits), armazenando os valores dos pontos diretamente em cada bit. Os valores dos operandos são armazenados na memória interna do microprocessador, não utilizando o espaço disponível ao programa aplicativo.

Os formatos dos operandos %S podem ser vistos na figura 2-7.

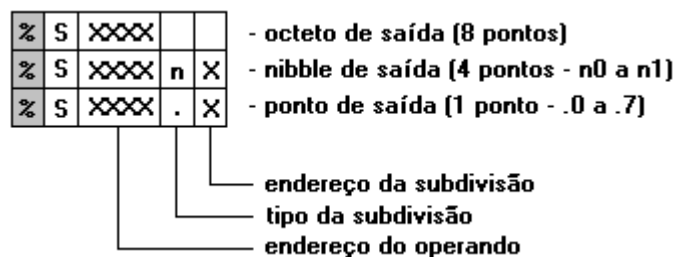


Figura 2-7 Formatos dos Operandos %S

Exemplos:

- %S0011.2 - ponto 2 do octeto de saída 11
- %S0010n1 - nibble 1 do octeto de saída 10
- %S0015 - octeto de saída 15

## Operandos %A - Relés Auxiliares

Os relés auxiliares são operandos usados para armazenamento e manipulação de valores binários intermediários no processamento do programa aplicativo. Sua quantidade nos controladores é fixa (ver seção Declaração de Operandos neste mesmo capítulo).

Operandos %A são utilizados em instruções binárias (contatos, bobinas) e de movimentação. Ocupam um byte de memória (8 bits), armazenando valores diretamente em cada bit. Os valores dos operandos são armazenados na memória interna do microprocessador, não utilizando o espaço disponível ao programa aplicativo.

Os formatos dos Operandos %A podem ser vistos na figura 2-8

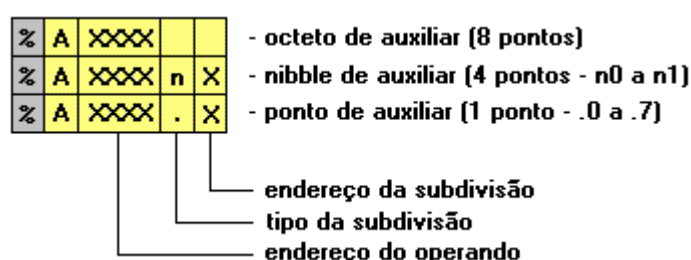


Figura 2-8 Formatos dos Operandos %A

Exemplos:

- %A0032.7 - ponto 7 do auxiliar de saída 32
- %A0087n1 - nibble 1 do auxiliar de saída 87
- %A0024 - octeto auxiliar 24

## Operandos %M - Memórias

Os operandos %M são usados para processamento numérico, armazenando valores em precisão simples, com sinal.

Os formatos dos operandos %M podem ser vistos na figura 2-9.

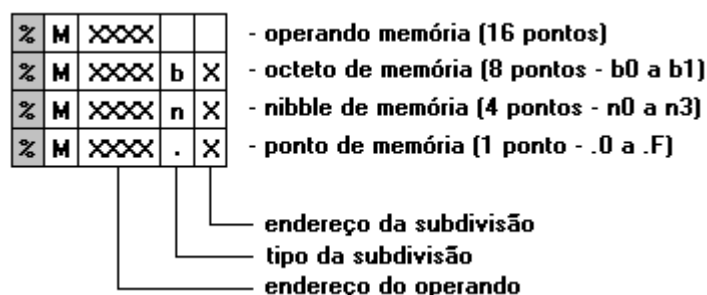
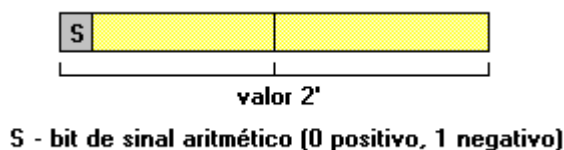


Figura 2-9 Formatos dos Operandos %M

A quantidade de operandos memória é configurável na declaração do módulo C, sendo o limite máximo dependente do modelo de CP em uso (ver seção **Declaração de Operandos** neste mesmo capítulo).

Os operandos %M são utilizados em instruções de movimentação, comparação, aritméticas, contagem, temporização e de conversão. Podem ser utilizados em contatos, da mesma forma que os operandos %E, %S e %A. Estes operandos ocupam dois bytes de memória (16 bits), armazenando o valor no formato complemento de dois (2'), conforme a figura 2-10.



**Figura 2-10 Formato do Operando Memória**

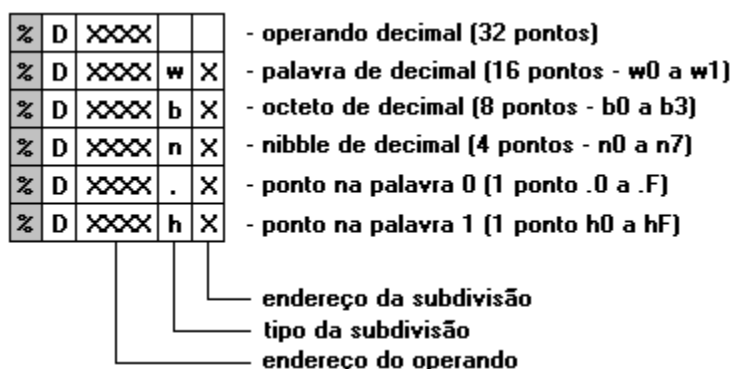
Exemplos:

- %M0032 - memória 32
- %M0072n1 - nibble 1 da memória 72
- %M0084.F - ponto 15 da memória 84

## Operandos %D - Decimais

Os operandos %D são usados para processamento numérico, armazenando valores em formato BCD com até 7 dígitos e sinal.

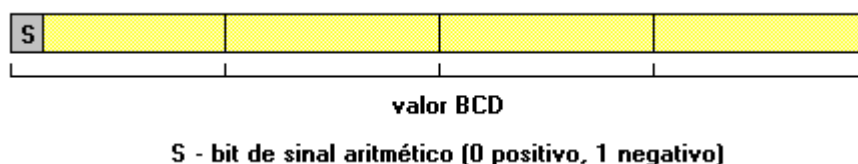
Os formatos dos operandos %D podem ser vistos na figura 2-11.



**Figura 2-11 Formatos dos Operandos %D**

A quantidade de operandos decimal é configurável na declaração do módulo C, sendo o limite máximo dependente do modelo de CP em uso (ver seção Declaração de Operandos neste mesmo capítulo).

Os operandos %D são utilizados em instruções de movimentação, comparação, aritméticas, e conversão. Podem ser utilizados em contatos, da mesma forma que os operandos %E, %S e %A. Estes operandos ocupam quatro bytes de memória (32 bits), armazenando o valor no formato BCD (cada dígito ocupa 4 bits), com sinal, conforme a figura a seguir:



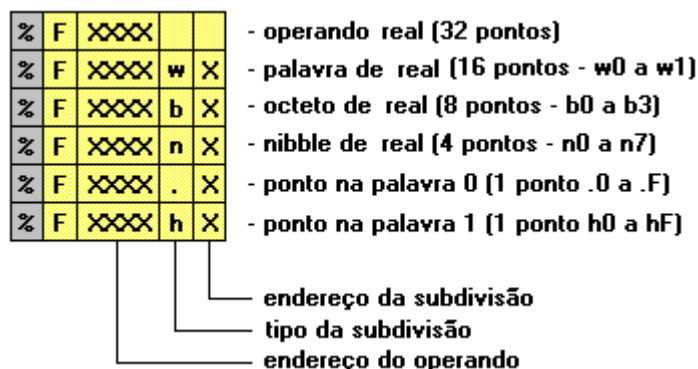
**Figura 2-12 Formato do Operando Decimal**

Exemplos:

- %D0041 - decimal 41
- %D0023b2 - octeto 2 do decimal 23
- %D0059n6 - nibble 6 da memória 59
- %D0172hA - ponto 10 da palavra 1 da memória 172

## Operandos %F – Reais

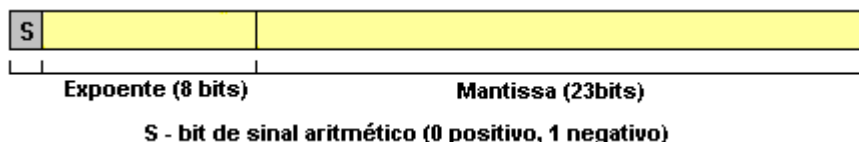
Os formatos dos operandos %F podem ser vistos na figura a seguir:



### Figura 2-13 Formatos dos Operandos %F

A quantidade de operandos real é configurável na declaração do módulo C, sendo o limite máximo dependente do modelo de CP em uso (ver seção **Declaração de Operandos** neste mesmo capítulo).

Os operandos %F são usados para processamento numérico, armazenando valores em 32 bits com ponto flutuante, precisão simples e sinal, conforme a norma IEEE 754. Estes operandos ocupam quatro bytes de memória (32 bits), armazenando o valor conforme a figura a seguir:



**Figura 2-14 Formato do Operando Real**

O valor de um operando real (%F) é obtido pela seguinte expressão:

$$\text{Valor} = \text{Sinal} \times 1.\text{Mantissa} \times 2^{(\text{Expoente} - 127)}$$

Sendo assim, a faixa de valores armazenáveis vai de -3,4028234663852886E+38 a 3.4028234663852886E+38.

Valores cujo módulo é maior que zero e menor que 1,1754943508222875E-38, são tratados como zero pelos CPs, por serem muito pequenos. Os CPs não tratam os seguintes casos especiais previstos na norma: números não normalizados, infinito e NaNs (not a number).

Exemplo:

- %F0032 – real 32

## Operandos %I - Inteiro

Os operandos %I são usados para processamento numérico, armazenando valores em precisão simples, com sinal. A diferença básica entre este tipo de operando e o operando Memória %M, é que o operando Inteiro %I possui 32 bits.

Os formatos dos operandos %I podem ser vistos na figura a seguir.

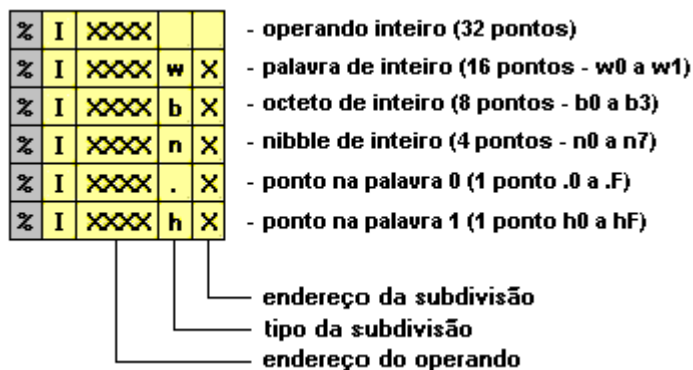


Figura 2-15 Formatos dos Operandos %I

A quantidade de operandos inteiro é configurável na declaração do módulo C, sendo o limite máximo dependente do modelo de CP em uso (ver seção Declaração de Operandos neste mesmo capítulo).

Os operandos %I são utilizados em instruções de movimentação, comparação, aritméticas, e conversão. Estes operandos ocupam quatro bytes de memória (32 bits), com sinal, conforme a figura a seguir:



Figura 2-16 Formato do Operando Inteiro

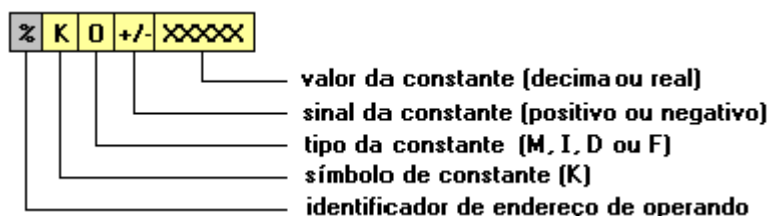
Exemplos:

- %I0041 - inteiro 41
- %I0023b2 - octeto 2 do inteiro 23
- %I0059n6 - nibble 6 do inteiro 59
- %I0172hA - ponto 10 da palavra 1 do inteiro 172

## Operandos %KM, %KI, %KD e %KF - Constantes

São operandos usados para a definição de valores fixos na elaboração do programa aplicativo. Existem tipos de constante, %KM, %KI, %KD e %KF, cada um seguindo um formato diferente de representação de valores, sendo idênticos aos operandos %M, %I, %D e %F, respectivamente.

O formato dos operandos constantes pode ser visto na figura a seguir.



**Figura 2-17 Formato dos Operandos Constantes**

Estes operandos são utilizados em instruções de movimentação, comparação, aritméticas, contagem e de temporização.

Exemplos:

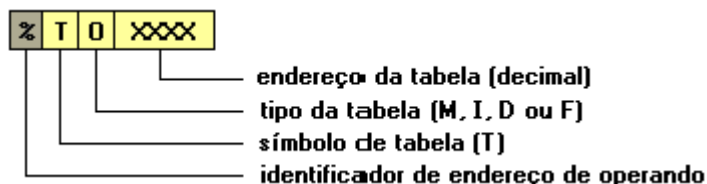
- %KM+00241 - constante memória + 241
- %KI+2000000000 – constante inteiro 2 bilhões ou  $2 \times 10^9$
- %KD-0019372 - constante decimal - 19.372
- %KF+0125.78 – constante real + 125.78
- %KF+3.1415E23 – constante real  $3.1415 \times 10^{23}$

As constantes reais podem conter até 8 dígitos significativos.

### Operandos %TM, %TI, %TD e %TF - Tabelas

Tabelas de operandos são conjuntos de operandos simples, constituindo arranjos unidimensionais com a finalidade de armazenar valores numéricos. Cada tabela possui uma quantidade de posições configurável, onde cada posição pode conter exatamente os mesmos valores de um operando %M, %I, %D ou %F, se a tabela for do tipo %TM, %TI, %TD ou %TF, respectivamente.

O formato dos operandos tabelas pode ser visto na figura a seguir:



**Figura 2-18 Formato dos Operandos Tabelas**

A quantidade de tabelas e o número de posições de cada uma é configurável na declaração do módulo C. Podem ser definidas até 255 tabelas totais e até o máximo de 255 posições em cada tabela, respeitando-se o limite da memória de operandos do CP.

As tabelas são utilizadas em instruções de movimentação.

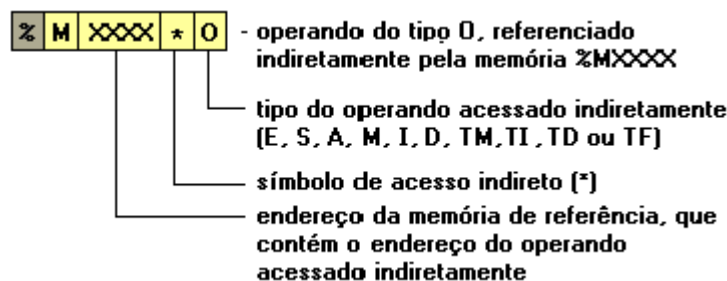


## Acesso Indireto

Esta forma de acesso é usada em conjunto com um operando memória %M para referenciar indiretamente outros operandos do sistema.

O sinal \*, colocado na frente de um tipo de operando, indica que este é referenciado pelo endereço contido na memória especificada à esquerda do sinal.

O formato do acesso indireto pode ser visto na figura a seguir.



**Figura 2-19 Formato de um Acesso Indireto**

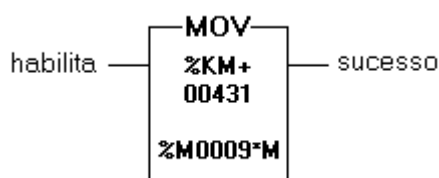
No MasterTool, o acesso indireto às tabelas é representado sem o asterisco.

O acesso indireto é utilizado em instruções de movimentação, comparação, contagem e de temporização.

Exemplos:

- %M0043\*E - octeto de entrada referenciada indiretamente pela memória 43
- %M1824\*A - octeto auxiliar referenciado indiretamente pela memória 1824
- %M0371TD - tabela de decimais referenciada indiretamente pela memória 371
- %M0009\*M - operando memória referenciado indiretamente pela memória 9
- %M0045\*F - operando real referenciado indiretamente pela memória 45

Exemplo:



Esta instrução movimenta o valor +431 para o operando memória cujo endereço é o valor correntemente armazenado em %M0009. Se %M0009 contiver o valor 32, então o valor +431 será armazenado em %M0032. Se %M0009 contiver o valor 12 então o valor constante será armazenado em %M0012.

### ATENÇÃO:

É responsabilidade do programa aplicativo que o valor contido na memória de referência (%M0009, no exemplo) represente endereços válidos, não contendo valores negativos ou acima dos endereços existentes para o tipo de operando referenciado indiretamente. As instruções não realizam os acessos indiretos inválidos, normalmente possuindo um sinal de saída para a indicação do erro.

Se no programa do exemplo anterior houvessem sido declarados 256 operandos %M, o valor de %M0009 deveria estar entre 0 e 255 para que a instrução fosse corretamente executada. Caso o valor não estivesse nesta faixa, o acesso não seria realizado.

## Declaração de Operandos

Os operandos %E, %S e %A ocupam áreas de memórias próprias, permanentemente reservadas no microprocessador do CP. A quantidade destes operandos nos controladores, portanto, é constante.

Por representarem valores fixos, os operandos constante (%KM, %KI, %KD e %KF) também não ocupam espaço em memória, sendo armazenados no próprio programa aplicativo na etapa de programação. Não há limites no número de operandos constante utilizados no programa.

A declaração dos operandos é realizada através da janela de edição do módulo C do MasterTool, sendo armazenada no módulo C. A quantidade de operandos declarada deve se adequar à capacidade máxima de memória disponível. Ver itens **Configurando Operandos Simples**, **Configurando Operandos Tabelas** e **Configurando Operandos Retentivos** na seção **Configurando o Módulo C** no Manual de Utilização do MasterTool.

Deve-se declarar uma quantidade mínima de operandos memória (%M) que comporte os bytes de diagnóstico utilizados nos módulos do barramento.

A reserva dos operandos %M, %I, %D e %F é realizada em blocos de 256 bytes. No caso de operandos memória, esta quantidade corresponde a 128 operandos. Em operandos decimais e reais, correspondem a 64 operandos.

Os operandos %TM, %TI, %TD e %TF são declarados informando-se o número de tabelas necessário para cada tipo e o número de posições que cada tabela contém. É possível a definição de até 255 tabelas totais e até 255 posições para uma tabela, respeitando-se o limite da memória RAM de operandos.

A tabela a seguir, mostra o espaço de memória ocupado por cada tipo de operando e onde os seus valores são armazenados.

Operando	Ocupação de Memória	Localização
%E - entrada	1 byte	Microprocessador
%S - saída	1 byte	Microprocessador
%A - auxiliar	1 byte	Microprocessador
%KM - constante M	-	-
%KI - constante I	-	-
%KD - constante D	-	-
%KF - constante F	-	-
%M - memória	2 bytes	RAM de operandos
%I - inteiro	4 bytes	RAM de operandos
%D - decimal	4 bytes	RAM de operandos
%F - real	4 bytes	RAM de operandos
%TM - tabela M	2 bytes por posição	RAM de operandos
%TI - tabela I	4 bytes por posição	RAM de operandos
%TD - tabela D	4 bytes por posição	RAM de operandos
%TF - tabela F	4 bytes por posição	RAM de operandos

Tabela 2-2 Ocupação de Memória e Localização dos Operandos

## Operandos Retentivos

Operandos retentivos são operandos que têm seus valores preservados quando a UCP é desenergizada (desligada). Os operandos não retentivos têm o seu valor zerado no momento em que o controlador programável é ligado.

Todos os operandos tabela são sempre retentivos. É possível configurar a quantidade de operandos %M (memória), %I (inteiro), %D (decimal), %F (real), %S (saída) e %A (auxiliar) retentivos.

Os operandos retentivos são configurados a partir dos últimos endereços até os iniciais, obedecendo a mesma regra dos operandos simples. Ou seja, a reserva é realizada em blocos de 256 bytes para operandos numéricos. A declaração dos operandos %S e %A é realizada de octeto em octeto.

Por exemplo, se existem 512 operandos %M declarados (%M0000 a %M0511) e deseja-se que 128 desses operandos sejam retentivos, serão considerados retentivos os operandos %M0384 ao %M0511.

Ver item **Configurando Operandos Retentivos** na seção **Configurando o Módulo C** no Manual de Utilização do MasterTool.

# Instruções

Os CPs ALTUS utilizam a linguagem de relés e blocos para a elaboração do programa aplicativo, cuja principal vantagem, além de sua representação gráfica, é ser similar a diagramas de relés convencionais.

A programação desta linguagem, realizada através do MasterTool, utiliza um conjunto de poderosas instruções apresentadas no capítulo **Referência das Instruções**, neste manual.

As instruções do MasterTool podem ser divididas em 7 grupos:

- RELÉS contendo as instruções:
  - RNA                      contato normalmente aberto
  - RNF                      contato normalmente fechado
  - BOB                      bobina simples
  - BBL                      bobina liga
  - BBD                      bobina desliga
  - SLT                      bobina de salto
  - PLS                      relé de pulso
  - RM                      relé mestre
  - FRM                      fim de relé mestre
- MOVIMENTADORES contendo as instruções:
  - MOV                      movimentação de operandos simples
  - MOP                      movimentação de partes de operandos
  - MOB                      movimentação de blocos de operandos
  - MOT                      movimentação de tabelas de operandos
  - CAB                      carrega bloco de constantes
- ARITMÉTICOS contendo as instruções:
  - SOM                      soma
  - SUB                      subtração
  - MUL                      multiplicação
  - DIV                      divisão
  - AND                      função "e" binário entre operandos
  - OR                      função "ou" binário entre operandos
  - XOR                      função "ou exclusivo" binário entre operandos
  - CAR                      carrega operando
  - =                      igual
  - <                      menor
  - >                      maior
- CONTADORES contendo as instruções:
  - CON                      contador simples
  - COB                      contador bidirecional
  - TEE                      temporizador na energização
  - TED                      temporizador na desenergização
- CONVERSORES contendo as instruções:



- B/D conversão binário - decimal
- D/B conversão decimal - binário
- GERAIS contendo as instruções:
  - LDI liga ou desliga pontos indexados
  - TEI teste de estado de pontos indexados
  - SEQ seqüenciador
  - CHP chama módulo procedimento
  - CHF chama módulo função
  - ECH escrita de operandos em outro CP para Ethernet
  - LTH leitura de operandos de outro CP para Ethernet
  - LAH libera atualização de imagem de operandos para Ethernet
- LIGAÇÕES contendo as instruções:
  - LGH ligação horizontal
  - LGV ligação vertical
  - LGN ligação negada

Algumas funções especiais desempenhadas pelos CPs são somente obtidas com os Módulos Função, que são chamados pela instrução CHF. No Capítulo 4 **Referência aos Módulos Função** apresenta uma lista destes módulos, disponíveis para a Série Ponto e que são fornecidos juntamente com o MasterTool Programming.

Os tempos de execução de cada instrução para os CPs da Série Ponto devem ser consultados no manual do respectivo CP.

## Restrições Quanto ao Uso de Instruções nos CPs

A linguagem de relés e blocos é perfeitamente compatível entre os CPs programados pelo MasterTool. Devido a características de funcionamento, contudo, algumas instruções não estão disponíveis em todos os controladores. Na tabela 2-3 estão relacionadas as instruções e os controladores nos quais as mesmas não podem ser utilizadas.

-  - indica que o CP possui a instrução
-  - indica que o CP não possui a instrução

UCPs Instrução	AL-600	AL-3003 AL-3004	QK600, QK800, QK801	PL101, PL102, PL103, PL104 PL105	AL-2000, AL-2002, AL-2003, AL-2004, QK2000	PO3042 PO3142 PO3045 PO3145	PO3242 PO3342
MES							
AES							
CES							
A/D							
D/A							
ECR							
LTR							
LAI							
ECH							
LTH							
LAH							

Tabela 2-3 Instruções Inexistentes em Determinados CPs

O MasterTool não permite inserir no programa aplicativo uma instrução que não possa ser executada no CP para o qual está configurado.

**ATENÇÃO:**

Ao editar um módulo de programa aplicativo, o tipo de UCP declarado no item **Modelo de UCP** da janela de edição do módulo C deve ser o da UCP onde o programa será executado.

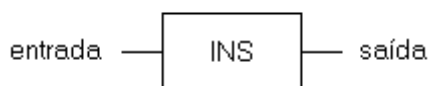
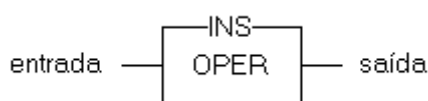
**ATENÇÃO:**

Caso se deseje mudar o tipo de UCP para outro, após o programa estar editado, deve-se procurar e remover as instruções que não podem ser utilizadas no novo tipo de UCP. Este procedimento deve ser realizado em todos os módulos do programa.

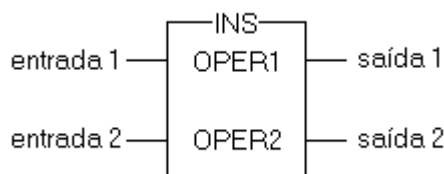
## Representação Gráfica das Instruções

Nas figuras a seguir estão representadas as configurações máximas de entradas e saídas em cada tipo, não sendo necessariamente todas utilizadas em uma determinada instrução.

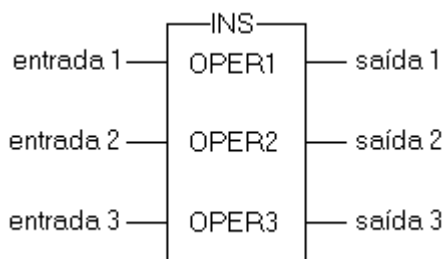
### Instruções com uma célula



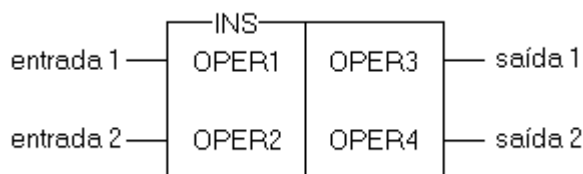
### Instruções com duas células



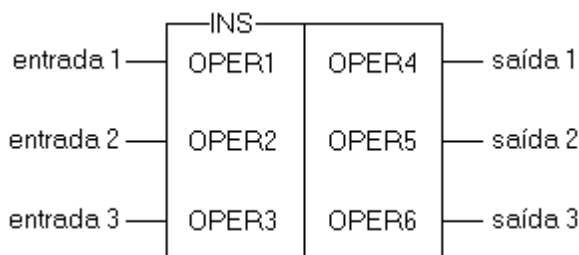
### Instruções com três células



### Instruções com quatro células



### Instruções com seis células



### Descrição da Sintaxe das Instruções

A descrição dos operandos possíveis de serem programados nas células de cada instrução é realizada de acordo com o formato mostrado na figura 2-17.

OPER1	OPER2	OPER3
LISTA DE OPERANDOS POSSÍVEIS NA CÉLULA	LISTA DE OPERANDOS POSSÍVEIS NA CÉLULA	LISTA DE OPERANDOS POSSÍVEIS NA CÉLULA

Figura 2-20 Formato da Sintaxe das Instruções

Várias combinações diferentes de operandos podem ser especificadas para uma mesma instrução.

Exemplo:

OPER1	OPER2	OPER1	OPER2
%M	%M %M*M %KM	%D	%D %M*D %KD

Esta declaração de sintaxe significa que, como primeiro operando, podem ser utilizados %M ou %D. Se o primeiro operando for %M, o segundo somente poderá ser %KM, %M ou %M \* M (acesso indireto a memória). Se o primeiro for %D, o segundo somente poderá ser %KD, %D ou %M \* D (acesso indireto a decimal).

### Restrições Quanto ao Posicionamento das Instruções

Existem regras a serem respeitadas quanto ao posicionamento das instruções nas 8 colunas da lógica. Pode-se dividir as instruções em três categorias:

- Instruções que podem ser editadas somente na coluna 7:
  - BOB                      bobina simples
  - BBL                      bobina liga
  - BBD                      bobina desliga
  - SLT                      bobina de salto
  - RM                      relé mestre
  - FRM                      fim de relé mestre

- Instruções que podem ser editadas nas colunas 0 a 6:
  - RNA relé normalmente aberto
  - RNF relé normalmente fechado
  - PLS relé de pulso
  - LGH ligação horizontal
  - LGV ligação vertical
  - LGN ligação negada
  - DIV divisão
  - MOB movimentação de blocos de operandos
  - > maior
  - < menor
  - = igual
  - SEQ seqüenciador
  - CHF chama módulo função
- Instruções que podem ser editadas em todas as colunas:
  - MOV movimentação de operandos simples
  - MOP movimentação de partes de operandos
  - MOT movimentação de tabelas de operandos
  - CAB carrega bloco de constantes
  - SOM soma
  - SUB subtração
  - MUL multiplicação
  - AND função 'e' binário entre operandos
  - OR função 'ou' binário entre operandos
  - XOR função 'ou exclusivo' binário entre operandos
  - CON contador simples
  - COB contador bidirecional
  - TEE temporizador na energização
  - TED temporizador na desenergização
  - B/D conversão binário - decimal
  - D/B conversão decimal - binário
  - CAR carrega operando
  - LDI liga ou desliga pontos indexados
  - TEI teste de estado de pontos indexados
  - CHP chama módulo procedimento
  - ECH escrita de operandos em outro CP para Ethernet
  - LTH leitura de operandos de outro CP para Ethernet
  - LAH libera atualização de imagem de operandos para Ethernet

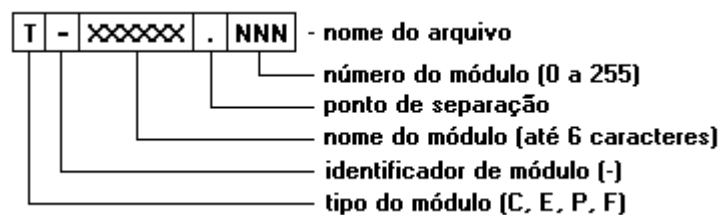


# Projeto de Programação

## Estruturação de um Projeto de Programação

Funcionalmente, um projeto de programação, pode ser visto como uma coleção de módulos utilizados para realizar uma tarefa específica, também conhecido como programa aplicativo. Isto permite uma visão hierárquica do projeto com a criação de sub-rotinas e funções.

Os módulos são chamados para a execução pelo software executivo (sistema operacional do CP) ou por outros módulos, através de instruções apropriadas. Quando armazenado em disco, o projeto de programação corresponde a um conjunto de arquivos, onde cada arquivo contém um módulo, denominados como mostra a figura 2-18.



**Figura 2-21 Formato do Nome dos Módulos em Arquivo**

Exemplo: **F-PID.033**

Em alguns locais deste manual e na Ajuda os módulos de programa são referenciados somente pelo seu tipo e número, quando não for relevante o nome utilizado no mesmo.

Exemplo: **E018**

### ATENÇÃO:

O nome do arquivo correspondente a um módulo de programa não deve ser alterado através de outro aplicativo do Windows™. Para alterar o nome de um arquivo, deve-se ler e salvar o mesmo com o nome desejado através do MasterTool. Ver seção **Salvando um Módulo com Outro Nome** no Manual de Utilização do MasterTool.

Se o nome do arquivo for modificado através de outro aplicativo do Windows™, poderá ser atribuído um nome inválido para o mesmo, não podendo mais ser lido para o MasterTool ou carregado no CP.

Existem 4 tipos de módulos que podem fazer parte de um projeto de programação:

- **Módulo C** (Configuração): existe um módulo de configuração por projeto, contendo os parâmetros de configuração do CP (C000).
- **Módulo C Estendido** (Configuração): este módulo de configuração existe quando o usuário utiliza em seu projeto uma determinada característica da UCP que necessita de um módulo de configuração estendido. Para maiores informações consultar o manual de utilização do MasterTool Programming (C003 a C009).
- **Módulo E** (Execução): podem existir até 4 módulos de execução por projeto. Os mesmos são chamados somente pelo sistema operacional do CP (E000, E001, E018 e E020).
- **Módulo P** (Procedimento): podem existir até 112 módulos procedimento por projeto. Eles contêm trechos de programa aplicativo, sendo chamados por instruções colocadas em módulos de execução, procedimento ou função. Após serem executados, o processamento retorna para a instrução seguinte à de chamada. Os módulos P funcionam como sub-rotinas, não permitindo a passagem de parâmetros para o módulo chamado (P000 a P111).
- **Módulo F** (Função): podem existir até 112 módulos função por projeto. Eles contêm trechos de programa aplicativo escritos de forma genérica, permitindo a passagem de parâmetros para o módulo chamado, de forma a poderem ser reaproveitados em vários programas aplicativos.

diferentes. São semelhantes a instruções, podendo ser chamados por módulos de execução, procedimento ou função. (F000 a F111).

### Módulo C - Configuração

O módulo C contém os parâmetros de configuração do CP. Sua criação é pré-requisito para a edição dos demais módulos do projeto de programação no MasterTool. A definição dos parâmetros contidos no mesmo é realizada através da janela de edição de módulo C. Para maiores detalhes sobre como configurar um módulo C, ver seção **Configurando o Módulo C** no Manual de Utilização do MasterTool.

Há somente um módulo C por projeto de programação, tendo como nome o próprio nome do projeto e o número 000.

#### Conteúdo de um módulo C:

- **Declaração do Barramento de módulos de E/S:** especifica a configuração dos módulos de E/S a serem utilizados no controlador programável, indicando a distribuição dos mesmos e módulos especiais no barramento do CP. A declaração dos módulos define, desta forma, o número de pontos e os endereços de E/S a serem utilizados no programa aplicativo. A mesma é realizada através da janela de edição do módulo C. Para maiores informações sobre como configurar o barramento, ver o item **Configurando o Barramento** na seção **Configurando o Módulo C** no Manual de Utilização do MasterTool.
- **Declaração de Operandos:** especifica o número de operandos simples e tabelas de operandos que serão utilizados no projeto de programação, dentro de cada tipo disponível. Permite também a definição da retentividade dos operandos, ou seja, quais operandos devem manter seu conteúdo com a falta de energia do sistema.
  - **Declaração de Operandos Simples:** permite a definição da quantidade de operandos Memória (%M), Inteiro (%I), Decimal (%D) e Real (%F). É realizada através da janela de edição de módulo C. Para maiores informações sobre como declarar operandos simples, ver o item **Configurando Operandos Simples** na seção **Configurando o Módulo C** no Manual de Utilização do MasterTool.
  - **Declaração de Operandos Tabela:** permite a definição do número de tabelas de operandos Memória (%TM), de operandos Inteiro (%TI), de operandos Decimal (%TD), de operandos Real (%TF) e do número de posições de cada uma. Uma tabela representa um conjunto de operandos, sendo a sua definição realizada através da janela de edição de módulo C. Para maiores informações sobre como configurar operandos tabela, ver o item **Configurando Operandos Tabela** na seção **Configurando o Módulo C** no Manual de Utilização do MasterTool.
  - **Declaração de Operandos Retentivos:** especifica o número de operandos simples que são retentivos, dentro dos operandos já declarados. Operandos retentivos são aqueles que continuam com o seu conteúdo inalterado com a falta de energia do CP, sendo os não retentivos zerados com a reinicialização do sistema. Os operandos tabela são todos retentivos. A declaração é realizada através da janela de edição de módulo C. Para maiores informações sobre como configurar operandos retentivos, ver o item **Configurando Operandos Retentivos** na seção **Configurando Operandos Retentivos** no Manual de Utilização do MasterTool.
- **Declaração dos Parâmetros Gerais da UCP:** são parâmetros genéricos necessários para o funcionamento do controlador programável, tais como o tipo de UCP na qual o programa aplicativo será carregado, o período de chamada dos módulos acionados por interrupção e o tempo máximo de ciclo de varredura. Estes parâmetros são declarados através da janela de edição de módulo C. Para maiores informações sobre como configurar os parâmetros gerais, ver seção **Configurando o Módulo C** no Manual de Utilização do MasterTool.
- **Declaração dos Parâmetros da Rede ALNET I:** especifica os diversos parâmetros necessários ao funcionamento da comunicação em rede ALNET I. Estes parâmetros são configurados na janela de edição de módulo C. Para maiores informações sobre como configurar parâmetros da rede ALNET I, ver item **Configurando Parâmetros da Rede ALNET I** na seção **Configurando o Módulo C** no Manual de Utilização do MasterTool.

- **Declaração dos Parâmetros da Rede Ethernet:** especifica os diversos parâmetros necessários ao funcionamento da comunicação em rede Ethernet, para os controladores programáveis que permitem o seu uso. Estes parâmetros são configurados na janela de edição de módulo C. Para maiores informações sobre como configurar parâmetros da rede Ethernet, ver item Configurando Parâmetros da Rede Ethernet na seção Configurando o Módulo C no Manual de Utilização do MasterTool.

### Módulo C Estendido – Configuração

Estes módulos contêm configurações de determinadas características das UCPs. Estes módulos são totalmente gerenciados pelo usuário, isto é, deve ser criado e apagado conforme necessidade do usuário. Isto se deve ao fato de que a quantidade deste tipo de módulo varia de acordo com cada aplicação, podendo não ter nenhum a ter até 7 módulos (C003 a C009).

Para maiores informações consultar Manual de Utilização do MasterTool.

### Módulo E - Execução

Os módulos E contêm trechos do programa aplicativo, sendo chamados para a execução pelo software executivo. Existem diversos módulos E, diferenciando-se entre si pelo modo como são chamados à execução, conforme o seu número.

Tipos de módulos E:

- **E000 - Módulo de Inicialização:** é executado uma única vez, ao se energizar o CP ou na passagem de modo programação para execução com o MasterTool, antes da execução cíclica do módulo E001.
- **E001 - Módulo Sequencial de Programa Aplicativo:** contém o trecho principal do programa aplicativo, sendo executado ciclicamente.
- **E018 - Módulo Acionado por Interrupção de Tempo:** o trecho de programa aplicativo colocado neste módulo é chamado para a execução em intervalos de tempo periódicos. Define-se o período de chamada do mesmo nos parâmetros gerais do módulo C, podendo ser escolhido entre 50 ms, 25 ms, 10 ms, 5 ms, 3,125 ms, 2,5 ms, 1,25 ms e 0,625 ms. Ao ser transcorrido o tempo programado, a execução sequencial do programa aplicativo é interrompida e o módulo E018 é executado. Após o seu final, o sistema retorna a execução para o ponto do processamento sequencial onde o módulo E0001 havia sido interrompido. O tempo continua a ser contado durante a chamada do módulo E018, devendo a sua execução ser o mais breve possível para não haver o aumento excessivo no tempo de ciclo do módulo E001.

#### ATENÇÃO:

O tempo de execução do módulo E018 não pode ser maior ou igual ao período de chamada. Caso isto aconteça, o CP entra em modo erro sendo exibida a mensagem **Reentrada no módulo E018**, na janela **Informações** (comando **Comunicação, Estado, Informações**).

### Módulo P - Procedimento

Os módulos P contêm trechos de programas aplicativos chamados a partir de módulos E, P ou F através da instrução CHP (Chama Procedimento).

Este tipo de módulo não possui passagem de parâmetros, sendo similar ao conceito de sub-rotina.

O número máximo de módulos deste tipo é 112 (P000 a P111).

O módulo P é útil para conter trechos de programas aplicativos que devem ser repetidos várias vezes no programa principal, sendo assim programados uma só vez e chamados quando necessário, economizando memória de programa.

Podem ser usados também para uma melhor estruturação do programa principal, dividindo-o em segmentos de acordo com a sua função e declarando-os em diversos módulos P. Neste caso, o módulo de execução contínua E001 somente chama os módulos P na sequência desejada.

Exemplos:

- P-MECAN.000 - realiza o intertravamento mecânico da máquina
- P-TEMPER.001 - realiza o controle de temperaturas

- P-VIDEO.002 - realiza o interfaceamento homem-máquina
- P-IMPRES.003 - gerência a impressão de relatórios

### Módulo F - Função

Os módulos F contêm trechos de programas aplicativos chamados a partir de módulos E, P ou F, através da instrução CHF (Chama Função).

Na chamada dos módulos F é possível a passagem de valores como parâmetros para o módulo chamado. Estes módulos são usualmente escritos de forma genérica para serem aproveitados por vários programas aplicativos, em linguagem de relés ou de máquina, sendo semelhantes às instruções da linguagem de relés. Os valores dos parâmetros são enviados e devolvidos através de listas de operandos existentes na instrução de chamada e no módulo F.

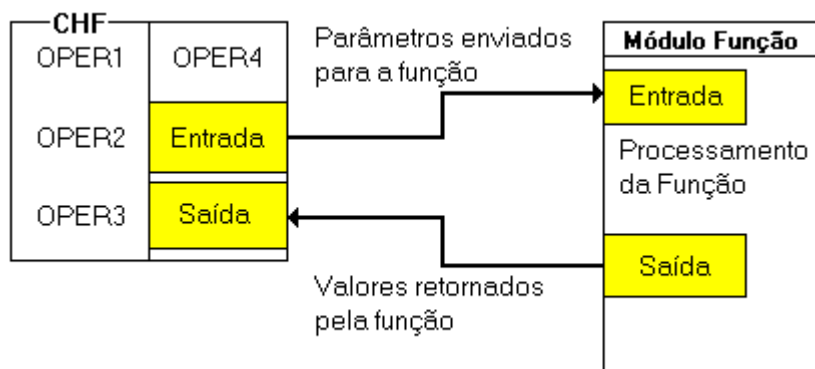
Na edição de um instrução CHF, devem ser definidas 2 listas de operandos que são utilizadas para:

- enviar parâmetros para execução do módulo função (Entrada)
- receber os valores retornados pelo módulo função (Saída)

Na edição do módulo função, também devem ser definidas 2 listas de operandos, utilizando o comando **Edição, Editar, Parâmetros**, que são utilizados para:

- receber parâmetros da instrução CHF (Entrada)
- enviar valores de retorno para a instrução CHF (Saída)

A passagem de parâmetros é realizada através da cópia dos valores dos operandos declarados (passagem de parâmetros por valor). A figura 2-19 apresenta o fluxo de dados entre a instrução CHF e o módulo função.



**Figura 2-22 Passagem de Parâmetros para o Módulo F**

Maiores informações a respeito da passagem de parâmetros podem ser encontradas na descrição da instrução CHF neste mesmo manual. É permitida a passagem de todos os tipos de operandos.

Exemplos:

- F-LINEAR.002 - executa a linearização de valores lidos de um sensor
- F-PID.033 - realiza cálculos para implementação de laço PID de controle

### Estados de Operação do CP

Existem cinco estados ou modos de operação do CP: inicialização, execução, programação, ciclado e erro. O estado em que o controlador programável se encontra é indicado nos LEDs do painel frontal da UCP, podendo também ser consultado pelo MasterTool, através da caixa de diálogo **Estado** (opções **Comunicação, Estado**, a partir do menu principal). Para obter informações específicas sobre os modos de operação, consultar o manual de utilização do controlador utilizado.

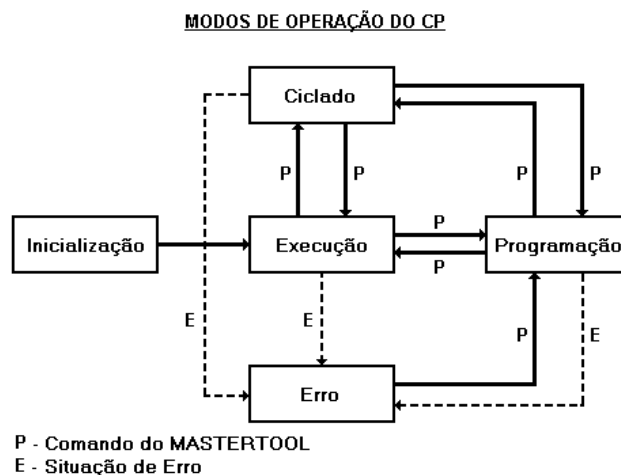
- **Estado Inicialização:** o CP inicializa as diversas estruturas de dados de uso do programa executivo e realiza consistências no projeto de programação presente na memória. Este estado ocorre após a energização do controlador, passando após alguns segundos para o estado execução. Caso não exista programa aplicativo na memória, o CP passa para modo erro.

Enquanto o CP está inicializando, pode-se acionar o comando Comunicação, Estado, Programação, ou atalho equivalente na barra de ferramentas, fazendo com que o CP passe diretamente para o estado de programação, ao invés de executar o programa aplicativo. Este procedimento é útil para a reinicialização de CPs com programas contendo erros graves de programação.

Por exemplo, um módulo com um laço infinito de execução, programado com uma instrução de salto para uma lógica anterior, provoca o acionamento do circuito de cão-de-guarda da UCP sempre que for ligada, após o estado de inicialização. Executando-se o procedimento anterior logo após a energização, o CP passa para o estado programação após inicializar, permitindo o apagamento ou a substituição do programa.

- **Estado Execução:** normalmente o controlador programável se encontra neste estado, varrendo continuamente os pontos de entrada e atualizando os pontos de saída de acordo com a lógica programada. Este estado indica que o CP está executando corretamente um programa aplicativo.
- **Estado Programação:** o programa **aplicativo** não é executado, não havendo a leitura dos pontos de entrada, sendo as saídas desativadas e a memória do CP é compactada. O CP permanece inoperante, esperando comandos do MasterTool. Este modo normalmente é utilizado para a carga dos módulos do projeto de programação pelo MasterTool, através do canal serial. Ao passar para estado execução ou ciclado a partir do estado programação, os operandos são zerados.
- **Estado Ciclado:** quando em modo ciclado, o controlador programável não executa ciclicamente o módulo E001, permanecendo à espera de comandos do MasterTool. Cada comando **executa ciclo** acionado no MasterTool (opções **Comunicação, Estado, Executa Ciclo** a partir do menu principal ou atalho equivalente) dispara uma única varredura do programa aplicativo (módulo E001), permanecendo o CP à espera de um novo comando após a execução da mesma. Quando o CP passa para modo ciclado, a contagem de tempo nos temporizadores pára, sendo os mesmos incrementados de uma unidade de tempo a cada duas varreduras executadas. As chamadas para o módulo de interrupção de tempo E018 não são realizadas neste modo. O módulo E020, acionado pela entrada de interrupção externa, continua sendo chamado neste modo.
- **Estado de Erro:** indica que houve alguma anomalia no CP durante o processamento do projeto de programação. O tipo de erro ocorrido pode ser consultado através da caixa de diálogo (opções **Comunicação, Estado, Informações** a partir do menu principal), enquanto o CP estiver neste estado. A saída do estado de erro somente é possível passando-se o controlador programável para modo programação.

Em condições normais, o controlador programável pode estar nos modos execução, programação e ciclado, sendo esses modos acionados através de comandos do MasterTool (opções **Execução, Programação e Ciclado** da caixa de diálogo **Estado**, ou seus atalhos equivalentes na **Barra de Ferramentas**). Na ocorrência de alguma situação de funcionamento errôneo nestes modos, o CP passa para estado de erro. A recuperação do modo erro somente é possível passando-se o controlador programável para modo programação. A figura 2-20 apresenta as possibilidades de troca de estados.



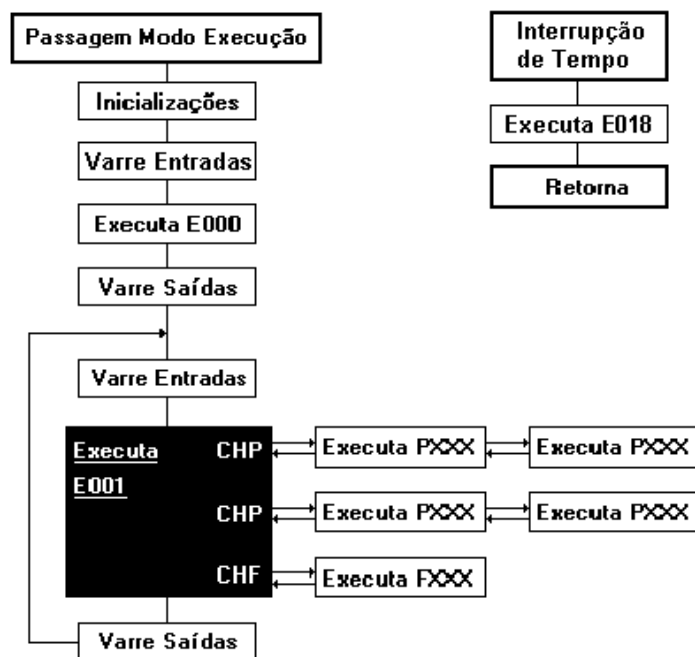
**Figura 2-23 Estados de Operação do CP**

Nos modos execução, programação e ciclado é possível carregar e ler módulos do projeto de programação pelo canal serial do controlador programável, bem como monitorar e forçar quaisquer operandos utilizados. Essas operações não são possíveis caso o CP esteja em modo erro.

Os operandos que não são retentivos são zerados na passagem de modo programação para execução ou programação para ciclado, permanecendo os demais inalterados.

## Execução do Projeto de Programação

Quando o CP é energizado ou após a passagem para modo execução, as inicializações do sistema são realizadas de acordo com o conteúdo do módulo C, sendo logo após executado o módulo E000 uma única vez. O controlador programável passa então para o processamento cíclico do módulo E001, atualizando as entradas e saídas e chamando o módulo E018, quando existente, a cada período de tempo de interrupção programado. A figura 2-21 mostra esquematicamente a execução do programa aplicativo.



**Figura 2-24 Execução do Projeto de Programação**

## Elaboração de Projetos de Programação

### Considerações Gerais

Um projeto de programação é composto ao menos por um módulo C (configuração) e um módulo E001 (execução). A condição mínima para a execução de um projeto de programação é a presença destes dois módulos na UCP do controlador programável.

O primeiro passo para a edição de um projeto de programação no MasterTool é a criação ou a leitura de um projeto. O módulo de configuração do projeto é criado automaticamente na criação de um novo projeto, uma vez que neste módulo estão contidas as declarações dos módulos de entrada e saída e operandos utilizados em todo o projeto. Cada módulo que contenha trechos de programa aplicativo (E, P ou F) necessita que o módulo C esteja presente no MasterTool para que possa ser editado.

Após a criação ou leitura de um projeto, pode-se editar o mesmo adicionando módulos já existentes, criando módulos novos para o projeto ou excluindo módulos que já façam parte do projeto.

O MasterTool permite que vários módulos sejam carregados e permaneçam simultaneamente em sua memória

### Considerações sobre Operandos

Os diversos módulos que compõem um projeto de programação devem preferencialmente ter sido programados utilizando-se o mesmo módulo C. Caso um módulo já programado necessite ser usado em outro projeto de programação, os operandos utilizados pelo módulo devem obrigatoriamente estar declarados no módulo C do novo projeto.

Os operandos disponíveis no controlador programável são de uso comum a todos os módulos do projeto de programação presentes no CP (operandos globais). Em consequência deste fato, dois módulos quaisquer podem estar inadvertidamente acessando o mesmo operando, ocorrendo erros no funcionamento de ambos.

Ao elaborar um projeto de programação, deve-se reservar operandos em número suficiente para o mesmo, preferencialmente separando-os em grupos, cada grupo utilizado somente por um módulo. Os operandos utilizados nos módulos F programados em linguagem de relés e blocos também podem ser acessados por quaisquer outros módulos de programa presentes no CP, mesmo os operandos utilizados na passagem de parâmetros. Para garantir o seu caráter genérico, cada módulo F deve utilizar um grupo de operandos diferente dos demais utilizados no programa aplicativo.

### Utilização dos Módulo P e F

Dentro de um módulo do projeto de programação podem ser colocadas as instruções de chamada de outros módulos. As instruções CHP e CHF chamam, respectivamente, módulos de procedimento e função. Elas realizam o gerenciamento das chamadas dos módulos, verificando a existência ou não dos mesmos no diretório do controlador programável, baseadas em seus tipos e números.

Existem 32 níveis de chamada, ou seja, podem ser executadas até 32 chamadas consecutivas de módulos sem ser finalizada a execução de nenhum. Deve-se considerar que o módulo E018 (se existir) e os módulos por ele chamados também ocupam níveis de chamada.

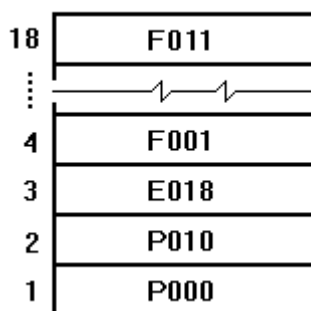
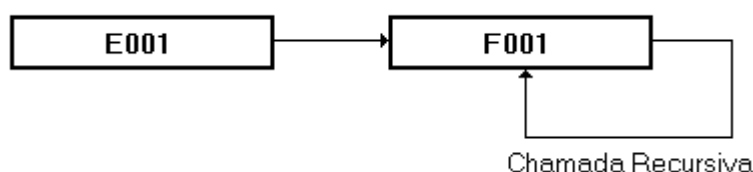


Figura 2-25 Número Máximo de Níveis de Chamada de Módulos

Quando o número máximo de chamadas acumuladas sem retorno for ultrapassado, o sistema não mais as realizará, prosseguindo com a execução normal do programa aplicativo. Nos casos de ocorrência de chamadas para módulos inexistentes ou o excesso do número de chamadas totais, são mostradas mensagens de advertência na janela **Informações** (opções **Comunicação**, **Estado**, **Informações** a partir do menu principal), pois estas situações poderão causar erros no processamento conforme a lógica programada.

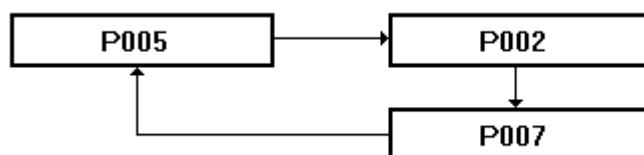
É possível a chamada de um módulo por ele mesmo (programação por recursividade) tomando-se os cuidados necessários, ou seja, deve ser previsto no trecho de programa aplicativo com recursividade um momento em que não há mais chamadas para o mesmo módulo. Embora seja possível, o uso de tal procedimento não é aconselhável em controladores programáveis, devido ao grande tempo de processamento que um pequeno trecho de programa aplicativo pode necessitar para ser executado e à facilidade da ocorrência de laços infinitos de execução (loops).



**Figura 2-26 Chamada Recursiva de Módulos**

Deve-se evitar a programação indevida de laços de chamada de módulos sem fim ("dead locks"). Caso um módulo do projeto de programação chame outro e este também realize uma chamada para o primeiro, se as instruções de chamada nos dois módulos não forem desabilitadas ambos permanecerão chamando-se mutuamente até a passagem do controlador programável para modo erro, por excesso de tempo de execução do programa aplicativo.

A mesma situação pode ocorrer com chamadas encadeadas entre diversos módulos, quando um módulo chamado volte a chamar algum módulo inicial da cadeia. Por exemplo, se o módulo P005 chamar o P002, este chamar P007 e este chamar novamente o P005, o processamento poderá permanecer neste laço se nenhuma instrução de chamada for desabilitada.



**Figura 2-27 Laço de Chamada de Módulos**

### Utilização do Módulo E018

O módulo E018 deve ser utilizado quando for necessário um processamento rápido para alguns pontos de entrada e saída do controlador programável, como por exemplo, no sensoreamento de fins-de-curso em sistemas de posicionamento rápido. Para este caso deve ser empregada um módulo Função F-AES.087 (Atualização de pontos de E/S, maiores detalhes no Capítulo 4), realizando dentro do módulo E018 um processamento semelhante a um laço completo de execução do programa principal. As entradas são lidas, o trecho de programa aplicativo desejado é executado e as saídas são atualizadas.

Da mesma forma, este módulo torna-se útil quando se deseja uma resposta dos acionamentos de saída após um tempo fixo dos estímulos das entradas, independente do tempo de varredura do programa principal, que pode ser variável. Essa característica é importante também em sistemas de controle de posição.



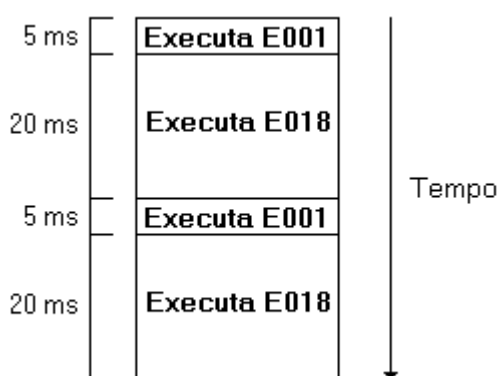
Outra aplicação para o módulo E018 é a geração de bases de tempo menores que 100 ms para o programa principal. Podem ser gerados temporizadores com precisão de 50 ms, 10 ms ou menos, se necessário, através do uso de instruções contadoras dentro do módulo de interrupção de tempo.

Este módulo é útil quando se deseja um controle preciso de tempos no processamento do CP.

### Cuidados na Utilização do Módulo E018

Alguns cuidados especiais são necessários na programação do módulo E018. Como o mesmo é chamado de modo síncrono a cada período fixo de tempo, interrompendo o processamento do módulo E001, o seu tempo de execução deve ser o mais breve possível para não aumentar excessivamente o tempo de ciclo total do programa aplicativo.

Se o intervalo entre as chamadas do módulo E018 for programado para 25 ms, por exemplo, e o seu tempo de execução for 20 ms, restarão somente 5 ms para a execução do programa principal antes que o E018 seja chamado novamente. Esta situação aumenta de forma considerável o tempo de ciclo do módulo E001.



**Figura 2-28 Cuidados na Utilização do Módulo E018**

Caso a execução do módulo E018 demore mais do que o intervalo de tempo programado para suas chamadas, o CP passa para o estado de erro, sendo exibida a mensagem "Reentrada no módulo E018" na janela **Informações** (opções **Comunicação**, **Estado**, **Informações** a partir do menu principal). Nesta situação, deve-se aumentar o período de chamada utilizado ou diminuir o tempo de execução do módulo E018 para que o projeto de programação possa ser executado corretamente.

As instruções permanecem com o mesmo comportamento quando executadas dentro do módulo E018, exceto em relação a algumas características particulares. Os temporizadores (TEE e TED) continuam a contar o tempo a cada 100 ms, qualquer que seja o período de acionamento programado para o módulo, exatamente como na execução cíclica. O relé de pulso (PLS) aciona a sua saída durante uma execução do módulo E018, zerando a mesma na próxima chamada. As instruções CHP e CHF podem ser usadas da mesma forma como no programa principal, devendo os módulos acionados pelas mesmas obedecerem às mesmas regras de programação válidas para o módulo E018 propriamente dito. O número máximo de níveis de chamada de módulos utilizados dentro do módulo E018 deve ser acrescentado ao máximo nível empregado em E001, devendo esta soma ser menor que o limite do sistema (18 níveis).

### Utilização dos Operandos na Programação dos Módulos E018

Outro cuidado necessário diz respeito ao compartilhamento de dados entre o módulo E018 e os demais presentes no controlador programável. As interrupções podem ocorrer em qualquer ponto do programa principal de execução cíclica (módulo E001 ou módulos P ou F chamados pelo mesmo), inclusive durante o processamento das suas instruções. Como os operandos são todos de uso comum a qualquer módulo do projeto de programação, deve-se tomar o cuidado para não utilizar inadvertidamente no módulo E018 algum operando que seja utilizado em outro módulo do projeto de programação, pois este uso, conforme o caso, pode ocasionar o funcionamento incorreto.

Para possibilitar o compartilhamento de dados entre o módulo E018 e outro módulo qualquer de execução cíclica devem ser utilizadas as instruções MOV (movimentação de operandos simples) e MOB (movimentação de blocos de operandos), para criar uma imagem dos operandos que contém os dados a serem compartilhados. Estas instruções devem ser utilizadas nos módulos pertencentes ao ciclo normal de execução e não no módulo E018.

Por exemplo, se for necessário que o módulo E018 utilize o valor contido em uma memória usada no programa principal, deve-se passar o valor desta memória para outra através da instrução MOV, devendo o módulo E018 utilizar somente esta última. A instrução MOV deve estar no programa principal, e não no módulo E018.

O fluxo contrário de dados também exige a criação de operandos imagem. Se o módulo E018 manipula uma tabela e o programa principal precisa utilizar os valores desta tabela, esses valores devem ser copiados para uma segunda tabela de uso exclusivo do programa principal, através de uma instrução MOB. A instrução MOB deve estar no programa principal, e não no módulo E018.

Uma situação semelhante ocorre para as instruções bobinas. Se algum ponto de um operando é modificado no programa principal por uma bobina, não é permitida a alteração de qualquer ponto pertencente a todo octeto do mesmo operando nos módulos E018. Esta restrição não existe quando os octetos utilizados pertencem à faixa %S0000 a %S0015.

Entretanto, é possível que pontos de um operando sejam alterados no módulo E018 por uma bobina e sejam somente testados por outro módulo com instruções contatos, por exemplo. A situação contrária também é permitida, ou seja, os pontos de um operando modificados no programa principal por bobinas podem ser testados no módulo E018 por contatos.

Outro cuidado a ser tomado diz respeito à atualização das entradas e saídas dentro do módulo E018.

Preferencialmente devem ser atualizadas dentro destes módulos somente as entradas utilizadas no seu processamento, utilizando-se o módulo função F-AES.087. Como o programa aplicativo de execução cíclica pode ser interrompido em qualquer local por estes módulos, se neles forem atualizadas as imagens das entradas do programa principal, estas poderão assumir valores diversos em pontos diferentes do programa aplicativo durante o mesmo ciclo de execução. Este fato pode provocar erros se um operando de entrada for utilizado em vários lugares do programa principal, pois normalmente é suposto que seu valor permaneça inalterado na mesma varredura.

Devido a este fato, é aconselhável o uso de octetos de entrada exclusivos para o módulo E018, se for necessária a sua atualização dentro do mesmo, não sendo estes octetos utilizados no programa principal.

Caso seja necessária a atualização de entradas utilizadas simultaneamente nas interrupções e no processamento cíclico, o valor das mesmas pode ser copiado para operandos auxiliares ou memórias no início do programa principal, sendo estes operandos utilizados no restante do mesmo. Pode-se também não atualizar as imagens das entradas no módulo E018 com o módulo função F-AES.087, mas somente ler diretamente os valores dos módulos de E/S para operandos memória também através do módulo função F-AES.087, e utilizar estas memórias em contatos para realizar o processamento nos módulos de interrupção.

A atualização de octetos de saída no módulo E018 (através do módulo função F-AES.087) é possível, desde que os pontos pertencentes a estes octetos sejam acionados por bobinas somente dentro destes módulos.

No módulo E018, não se deve escrever valores com o módulo função F-AES.087 em módulos de saída declarados no barramento através do MasterTool, pois a varredura das saídas também realiza atualização de valores nos mesmos.

Quando um módulo E018 está sendo executado e a compactação for disparada, os mesmos poderão ser transferidos para outra posição na memória pela rotina de compactação. Durante esta transferência a sua chamada será desabilitada, podendo algumas interrupções ocorrerem sem que o


módulo E018 seja processado. Deve-se atentar para este efeito da compactação sobre a execução do módulo acionado por interrupção. Durante a compactação dos demais módulos, todavia, o módulo E018 continuará sendo executado.

## Depuração de Projetos de Programação

Várias facilidades estão previstas no controlador programável para auxiliar a depuração dos projetos de programação, sendo descritas a seguir.

### Informações do Estado do CP

Diversas informações sobre o estado do controlador podem ser obtidas com o acionamento das opções **Comunicação, Estado, Informações** no MasterTool.

Atalho: 

- **Modelo da UCP** - indica o tipo do controlador com o qual o MasterTool está comunicando.
- **Versão do Executivo** - mostra o número da versão do programa executivo que o CP contém.
- **Modo de Operação** - indica o modo de operação atual do CP: execução, programação, ciclado ou erro.
- **Mensagem de Erro/Advertência** - se o CP estiver em modo erro, é apresentada uma mensagem indicando a causa do erro ocorrido. Caso o CP esteja em qualquer outro modo, a mensagem indica a existência de problemas que não causam a mudança para modo erro (por exemplo, a bateria do CP descarregada). Ver **Mensagens de Erro**, apêndice A do Manual de Utilização do MasterTool.
- **Saídas** - indica se as saídas estão habilitadas ou desabilitadas.
- **Relés Forçados** - indica se existe algum ponto de entrada ou saída forçado.
- **Troca de Módulos com CP Energizado** - indica a possibilidade de troca de módulos com CP energizado.
- **Compactando RAM** - indica se o CP está compactando a memória RAM do programa aplicativo.
- **Copiando Módulo** - indica se algum módulo está sendo carregado no CP, transferido da RAM para a flash EPROM ou da flash EPROM para a RAM, ou se o CP está apagando a memória flash.
- **Nível de Proteção** - mostra o nível de proteção atual do CP.
- **Tempos de Ciclo** - mostra o valor instantâneo, médio, máximo e mínimo do tempo de ciclo do programa aplicativo. Ver seção **Tempos de Ciclo de Execução do Programa** neste mesmo capítulo.
- **Período de Acionamento de E018** - mostra o período de chamada do módulo acionado por interrupção de tempo E018, se estiver presente no CP.

As janelas de estado do CP (opções Comunicação, Estado, Informações), diretório de módulos (opções **Comunicação, Módulos**) e monitoração (opções Comunicação, Monitorar Operandos ou Monitorar Bloco de Operandos ou Monitorar Tabelas) fornecem diversas informações úteis para a verificação do bom funcionamento do controlador. Estas informações podem ser obtidas à distância, caso o CP esteja ligado em rede. Sempre que o MasterTool for conectado a algum CP, aconselha-se a consulta dessas informações como o primeiro procedimento a ser tomado.

### Monitoração

Através do MasterTool é possível monitorar os valores de um ou mais operandos do CP em qualquer modo de operação, exceto em modo erro.

Os valores dos operandos contidos em uma lógica de programa aplicativo também podem ser visualizados diretamente sobre a mesma facilitando a verificação de seu funcionamento.

Para maiores informações sobre como realizar a monitoração, ver itens **Monitorando Operandos Simples**, **Monitorando Operandos Tabelas** e **Monitorando Programas** na seção **Comunicando com o CP ou Roteador** no Manual de Utilização do MasterTool.

A monitoração de operandos no CP ocorre no final do ciclo de execução do programa aplicativo. Devido a este fato, situações incoerentes podem ser visualizadas na monitoração de lógicas, se os valores dos operandos forem modificados nas lógicas posteriores à monitorada.

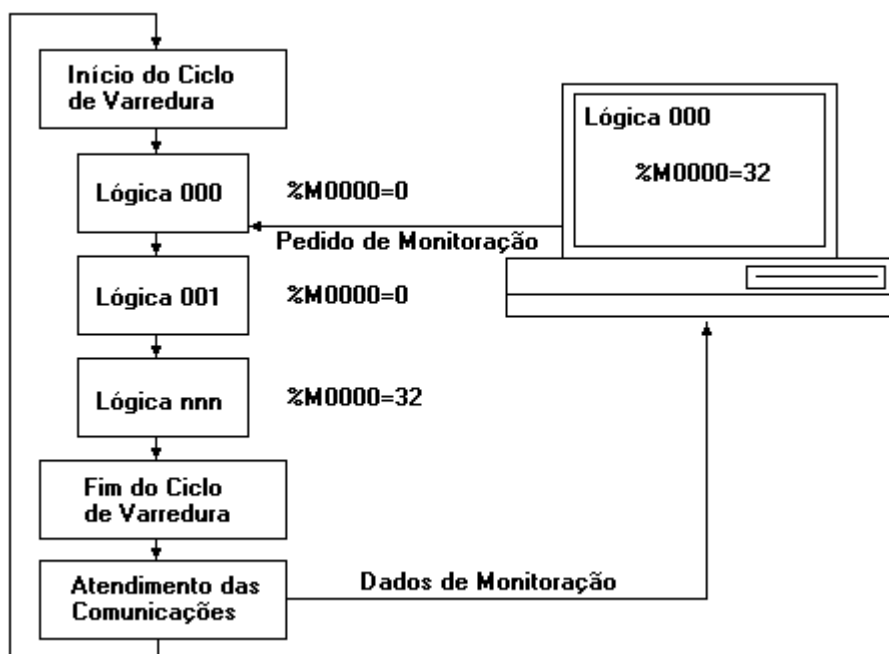


Figura 2-29 Situação Incoerente na Monitoração de Lógicas

### Forçamento

Os valores dos operandos também podem ser forçados com o MasterTool, ou seja, pode-se modificar o conteúdo de qualquer operando do projeto de programação. O forçamento de operandos é permitido em qualquer modo de operação, exceto em modo erro. Ver itens **Forçando Operandos Simples** e **Forçando Operandos Tabela** na seção **Comunicando com o CP ou Roteador** no Manual de Utilização do MasterTool.

Os operandos %A, %M, %I, %D, %F, %TM, %TI, %TD e %TF têm o seu valor alterado somente por uma varredura, logo após o envio do comando ao CP. Para que o valor forçado permaneça no operando, não pode haver no programa nenhuma instrução que o modifique.

O forçamento dos operandos %E e %S é realizado de forma permanente no controlador. Após o envio do comando ao CP, o valor é forçado em todas as varreduras do programa aplicativo, até que o operando seja liberado. O LED FC no painel da UCP permanece ligado se houver algum operando %E ou %S forçado.

Os valores forçados em operandos %E sobrepõem os obtidos na leitura dos módulos de entrada, antes do início de cada ciclo de execução do programa aplicativo. O programa é executado com o valor forçado, como se o ponto de entrada correspondente estivesse com este valor, podendo ser visualizado na monitoração.

Por exemplo, caso o operando %E0002.5 esteja forçado com o valor 1, o programa aplicativo será executado com este valor para este operando, não importando o estado do ponto no módulo de entrada correspondente. A monitoração de %E0002.5 mostra sempre o valor 1.

Os valores forçados nos operandos %S são diretamente enviados para os módulos de saída, independente do valor obtido após a execução do programa aplicativo. A monitoração mostra o valor

forçado, que corresponde ao valor assumido pelo ponto correspondente ao operando no módulo de saída.

Por exemplo, caso o operando %S0024.3 esteja forçado com o valor 0, o ponto respectivo no módulo de saída permanecerá desligado, não importando o estado da bobina que contenha o operando no programa aplicativo. A monitoração de %S0024.3 mostra sempre o valor 0.

**ATENÇÃO:**

Podem ser visualizadas situações incoerentes na monitoração de lógicas com operandos %S forçados. Isto ocorre porque o valor monitorado pode ser diferente do valor realmente obtido pelo programa aplicativo.

**ATENÇÃO:**

Todos os forçamentos de operandos %E e %S são removidos com a desenergização do CP. O forçamento destes operandos deve ser utilizado de forma temporária, somente para auxiliar a depuração do projeto de programação. Não devem ser deixados operandos %E ou %S forçados em caráter permanente, pois serão liberados com a desenergização e posterior energização do controlador.

Os operandos %E e %S deixam de ser forçados pelo CP através do comando de liberação de forçamento. A liberação consiste em anular o forçamento anteriormente determinado. Os operandos %E voltam a ter seus valores atualizados de acordo com os módulos de entrada, enquanto que os módulos de saída recebem os valores obtidos no processamento do programa aplicativo.

**ATENÇÃO:**

A operação de forçamento não atua sobre operandos %E ou %S atualizados com o módulo função F-AES.087. Esta instrução executa a leitura para operandos %E ou a escrita de operandos %S no momento em que é executada, não considerando os efeitos de forçamento sobre os mesmos. Por este motivo, recomenda-se que não sejam forçados os operandos atualizados pelo o módulo função F-AES.087 que estejam ativas no programa.

Para maiores informações sobre como liberar operandos forçados, ver item **Liberando Operandos Forçados** na seção **Comunicando com o CP ou Roteador** no Manual de Utilização do MasterTool.

**Desabilitação das Saídas**

Para a segurança na posta-em-marcha quando se utiliza o programa aplicativo diretamente na máquina, os acionamentos das saídas do controlador programável podem ser desabilitados através do comando desabilita. O programa aplicativo continua a ser executado no CP, com a varredura das entradas e cálculo dos valores das saídas, porém com todos os pontos de saída mantidos desacionados. Os operandos %S podem ser monitorados e conferidos com os valores esperados para os mesmos.

Para maiores informações sobre como desabilitar as saídas, ver item **Habilitando e Desabilitando as Saídas** na seção **Comunicando com o CP ou Roteador** no Manual de Utilização do MasterTool.

**ATENÇÃO:**

Se o CP for desenergizado, a desabilitação dos pontos de saída é removida. Ou seja, quando o CP for novamente energizado, o estado dos operandos da memória será normalmente transferido, ao final de cada varredura, para os pontos de saída. A desabilitação deve ser utilizada de forma temporária, somente para auxiliar a depuração do projeto de programação.

**Modificações no Programa**

A carga de módulos durante a execução do projeto de programação (carga "on line") possibilita sucessivas modificações e envios do módulo em depuração para o controlador programável. Deste modo não é necessária a reinicialização do programa aplicativo de controle nem a troca do estado do controlador programável a cada alteração efetuada em um módulo.

**ATENÇÃO:**

Após qualquer modificação realizada no módulo C do projeto de programação, o mesmo deve ser enviado para o CP.

**ATENÇÃO:**

Se a declaração de operandos simples ou tabelas for modificada, aconselha-se reinicializar o CP, passando para modo programação, carregando o módulo C e retornando para modo execução. Podem ocorrer erros no funcionamento alterando-se a configuração de operandos e enviando o módulo C com o controlador em modo execução.

Após um certo número de cargas sucessivas em modo execução, entretanto, pode se tornar necessária a compactação da memória RAM de programa pelos motivos explicados na seção **Gerenciamento de Módulos do Projeto de Programação no CP**, neste capítulo. Este tipo de carga somente é possível se houver memória livre suficiente no CP para armazenamento do módulo a ser enviado.

Ao acabar a depuração de um módulo de programa, sugere-se a transferência do mesmo para a memória flash EPROM ou a sua gravação no cartucho de EPROM, liberando o espaço disponível na memória RAM de programa.

**Modo Ciclado**

A execução do projeto de programação em modo ciclado torna-se útil na verificação do funcionamento de intertravamentos rápidos no programa aplicativo. As demais facilidades de depuração continuam atuando da mesma forma como no modo execução (monitoração, forçamento, carga e outras operações com módulos).

Em modo ciclado, os valores dos operandos permanecem constantes entre os ciclos, exceto os pontos de entrada (%E) que continuam sendo continuamente atualizados, mostrando seus valores reais.

**Gerenciamento de Módulos do Projeto de Programação**

Os módulos que compõem o programa aplicativo são independentes entre si, não necessitando de ligação ("link") através de programas auxiliares. A carga dos módulos no controlador programável pelo canal serial pode ser realizada em qualquer ordem, permitindo que somente o módulo alterado seja carregado no CP, em caso de manutenção de projetos de programação.

**ATENÇÃO:**

Para a UCP do sistema, somente o tipo do módulo e seu número são relevantes, sendo o seu nome desconsiderado. Se dois módulos com tipo e número iguais mas com nomes diferentes forem carregados no CP, somente o último carregado será considerado.

O controlador programável organiza um diretório interno onde são armazenadas diversas informações a respeito dos módulos contidos no mesmo, podendo ser consultado pelo MasterTool através do comando diretório de módulos (opções **Comunicação**, **Módulos** a partir do menu principal). Quando este comando for acionado, uma caixa de diálogo é aberta, mostrando na sua parte superior, dois quadros chamados **Módulos em RAM** e **Módulos em EPROM** com a lista dos nomes e a ocupação de memória de cada módulo presente no CP.

No quadro **Memória Ocupada** é informado o número total de módulos e o espaço de memória total ocupado pelos mesmos (soma de todas as ocupações individuais), além do espaço total ocupado em RAM ou EPROM.

No quadro **Memória Livre** estão expostas as quantidades de memória RAM e EPROM disponível para a carga de novos módulos, em cada banco de memória existente no controlador programável.

Somente os módulos presentes no diretório são considerados válidos para a execução no CP.

Um módulo de programa presente no diretório do CP pode estar somente em um tipo de memória, RAM ou EPROM, nunca simultaneamente em ambas. Os módulos carregados pelo canal serial são sempre armazenados na memória RAM de programa aplicativo.

### Compactação

A memória de programa do controlador programável está dividida em um ou mais bancos, dependendo do modelo da UCP utilizada (ver tabela 2-1 na seção **Organização de Memória dos CPs** neste capítulo).

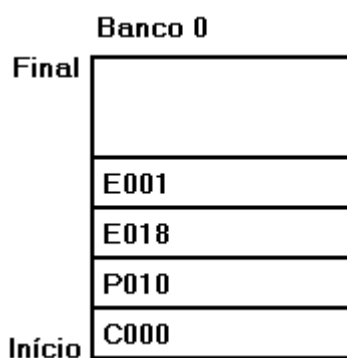
À medida que os módulos que compõem o projeto de programação são enviados para o CP através do canal serial, os mesmos ocupam o primeiro banco de memória RAM, desde o seu início até o seu final. Quando o espaço restante no primeiro banco não for suficiente para carregar o próximo módulo, este será carregado no banco seguinte, se este existir.

A cada carga de novo módulo no controlador programável, o software executivo testa se há espaço suficiente para o mesmo desde o primeiro até o último banco disponível. A carga de um novo módulo somente é possível se houver memória livre à disposição para o seu armazenamento.

Dentro de um banco de memória RAM, a carga de um módulo é realizada sempre a partir da primeira posição após o último módulo presente. Se um módulo no início do banco for removido, os módulos que estão após o mesmo devem ser transferidos para ocupar o seu espaço de memória, para que este espaço esteja disponível no final do banco para a carga de outros módulos. Este procedimento denomina-se **compactação da memória RAM** do programa aplicativo.

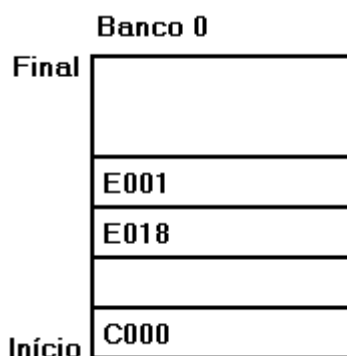
### Exemplo:

Suponha-se que o primeiro banco de memória do controlador programável esteja inicialmente com os seguintes módulos:



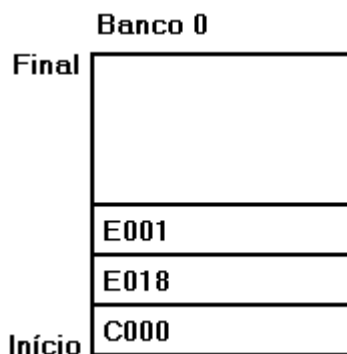
**Figura 2-30 Compactação de Memória RAM**

Se o módulo P010 for removido do CP o banco 0 passará a ter a seguinte organização:



**Figura 2-31 Compactação da Memória RAM - 2**

O espaço anteriormente ocupado por P010 não é aproveitável pelo controlador programável, pois a carga de um novo módulo somente é possível após o último, o módulo E001. Após realizar a compactação de memória do CP, o banco 0 passa para a seguinte configuração:



**Figura 2-32 Compactação da Memória RAM - 3**

Os módulos E018 e E001 são transferidos para o espaço anteriormente ocupado pelo módulo P010, tornando este espaço disponível ao final da memória do banco para a carga de outro módulo.

Se o controlador programável estiver em modo programação ou ciclado, os bancos de memória RAM de programa são mantidos automaticamente compactados pelo programa executivo. Em modo execução, todavia, deve-se disparar a compactação manualmente, através do MasterTool (opções **Comunicação, Módulos, Compactar RAM** desde o menu principal). Este procedimento é comum quando são realizadas diversas cargas de módulos em modo execução (cargas "on line"), tipicamente quando um módulo está sendo depurado, necessitando de sucessivas alterações e transmissões para o CP.

**ATENÇÃO:**

Dependendo da localização dos módulos na memória, o procedimento de compactação pode aumentar em muito o tempo de alguns ciclos do programa aplicativo, quando realizado em modo execução. Deve-se estar consciente dos efeitos deste aumento de tempo de processamento. Aconselha-se que a compactação não seja disparada caso a máquina sob controle esteja em operação ou com seus acionamentos principais ativos.

Devido a este mecanismo de gerenciamento de módulos no controlador programável, é possível que a soma da memória disponível nos bancos do CP com o valor ocupado pelos módulos seja menor que a memória de programa total da UCP, se esta estiver em modo execução. Este fato significa que a memória de programa não está compactada. Após a compactação, entretanto, a soma dos valores ocupados com a memória livre deve ser igual à memória total.

No MasterTool não existe uma Compactação de Flash, igualmente como possui a Compactação de RAM. O método para se "compactar" a Flash é carregar os módulos para a RAM, limpar a Flash e somente então recarregar os módulos para a Flash.

### Utilização da Memória Flash EPROM

Os controladores contêm Memória flash EPROM e esta é colocada na placa do circuito do CP, não sendo necessário removê-la para gravar ou apagar programas. Estas operações são realizadas pelo próprio controlador, através de comandos do MasterTool. Esta memória pode ser gravada parcialmente, porém não permite o seu apagamento parcial do seu conteúdo. Ou seja, somente é possível desgravar todo o conteúdo da memória no seu apagamento.

A configuração de memória de cada modelo de CP é apresentada na seção Organização de Memória dos CPs neste capítulo.

### Transferência de módulos de RAM para Flash:

Após serem carregados na memória RAM de programa, através do canal serial do CP, os módulos do projeto de programação podem ser transferidos para a flash EPROM. Este comando somente é utilizável nos CPs que possuam memória flash. Para maiores informações sobre como transferir módulos de RAM para flash EPROM, ver item **Transferindo Módulos de RAM para Flash EPROM** na seção **Comunicando com o CP ou Roteador** no Manual de Utilização do MasterTool.



Pode-se transferir um único módulo ou um conjunto de módulos, mesmo com o CP executando o programa. A transferência em modo execução é realizada parcialmente em cada varredura, podendo demorar vários segundos até ser completada, principalmente se houver alto tempo de ciclo de execução. No final da transferência, o módulo em RAM é automaticamente apagado e as informações do diretório modificadas.

O gerenciamento da carga do módulo na flash EPROM é idêntico ao da memória RAM, mostrada na seção anterior **Compactação**. Ou seja, o módulo da RAM é gravado no primeiro banco de flash que possua espaço livre suficiente para o conter, após o último módulo do banco. A pesquisa do espaço livre ocorre na ordem sequencial dos bancos (0, 1, 2 e 3).

### **Transferência de módulos de EPROM para RAM:**

Os módulos presentes na memória flash EPROM ou no cartucho de EPROM também podem ser transferidos para a memória RAM de programa. Para maiores informações sobre como transferir módulos de EPROM para RAM, ver item **Transferindo Módulos de EPROM para RAM** na seção **Comunicando com o CP ou Roteador** no Manual de Utilização do MasterTool.

Pode-se transferir um único módulo ou um conjunto de módulos, mesmo com o CP executando o programa. A transferência em modo execução é realizada parcialmente em cada varredura, podendo demorar vários segundos até ser completada, principalmente se houver alto tempo de ciclo de execução. No final da transferência, as informações do diretório são modificadas.

O gerenciamento da carga do módulo na flash EPROM é idêntico ao da memória RAM, mostrada na seção anterior **Compactação**.

### **Apagamento e reabilitação de módulos em EPROM:**

O comando de apagamento pode ser utilizado para módulos armazenados na memória EPROM do CP. Como o apagamento de EPROMs somente é possível para todo o seu conteúdo, este comando apenas retira as informações do diretório de módulos, não realizando um apagamento real da memória.

O mesmo ocorre se um módulo gravado em EPROM for substituído por um novo módulo de mesmo tipo e número carregado pelo canal serial. O novo módulo é armazenado em RAM, permanecendo o antigo em EPROM, sendo mostrado no diretório apenas o novo em RAM.

O módulo removido através do comando de apagamento ou substituído com carga de um novo módulo pode ser recuperado para o diretório, pois o seu conteúdo ainda está gravado na memória EPROM. Esta recuperação é possível com o comando de reabilitação de módulos.

A reabilitação faz com que um módulo inexistente no diretório reapareça situado em EPROM, ou que um já existente em RAM seja substituído por um antigo em EPROM.

Para maiores informações sobre como apagar ou reabilitar módulos, ver itens **Apagando Módulos do CP ou Roteador** e **Reabilitando Módulos em EPROM** na seção **Comunicando com o CP ou Roteador** no Manual de Utilização do MasterTool.

### **Apagamento da memória EPROM:**

Com o apagamento total da memória EPROM, todos os módulos são removidos, ficando todo o seu espaço disponível para a gravação de novos módulos.

Para apagar o cartucho de EPROM deve ser utilizado um dispositivo apagador apropriado, após a remoção do cartucho do CP.

Para apagar a memória flash EPROM, utiliza-se as opções **Comunicação, Módulos, Apaga Flash** estando o CP em modo programação. O apagamento pode demorar vários segundos, dependendo da capacidade da memória flash utilizada no CP. Para maiores informações sobre como apagar a memória flash, ver item **Apagando a Memória Flash EPROM** na seção **Comunicando com o CP ou Roteador** no Manual de Utilização do MasterTool.

## Tempos de Ciclo de Execução do Programa

O tempo máximo possível para a execução de um ciclo completo do programa aplicativo no controlador programável é configurável de 100 ms a 800 ms. Ou seja, a execução completa de uma varredura do módulo E001 não pode se estender por mais do que o valor configurado, incluindo as chamadas para módulos P e F e acionamentos do módulo de interrupção de tempo E018. O software executivo realiza uma verificação contínua no tempo de ciclo, passando automaticamente para estado de erro caso este limite seja ultrapassado.

Pode-se verificar os tempos de execução do programa aplicativo através da janela de informações do CP (opções **Comunicação**, **Estado**, **Informações** a partir do menu principal), sendo informados diversos tempos de ciclos de execução, especificados a seguir:

- **Tempo de ciclo instantâneo:** indica o tempo de ciclo da última varredura executada pelo CP antes de enviar as informações do seu estado para o MasterTool. Este item é útil em modo ciclado, quando mostra o tempo de execução do último ciclo disparado no controlador programável.
- **Tempo de ciclo médio:** indica a média de tempos de execução das últimas 256 varreduras realizadas pelo CP. Em modo execução este parâmetro fornece uma idéia geral do tempo de processamento do programa aplicativo, ao contrário do tempo de ciclo instantâneo, que pode estar mostrando um valor atípico de uma varredura isoladamente. Como este tempo é calculado somente a cada 256 varreduras, por vezes o seu valor necessita de alguns segundos para ser atualizado, principalmente em caso de aumento brusco no tempo de execução (inclusão de novos módulos no controlador programável, por exemplo).
- **Tempo de ciclo máximo:** indica o maior tempo entre todos os ciclos realizados desde a passagem do CP para modo execução ou ciclado.
- **Tempo de ciclo mínimo:** indica o menor tempo entre todos os ciclos realizados desde a passagem do CP para modo execução ou ciclado.

Os tempos de ciclo são indicados em milissegundos (ms), sendo as contagens inicializadas na passagem de modo programação para execução ou programação para ciclado.

O atendimento da comunicação serial com o MasterTool aumenta o tempo de ciclo do programa aplicativo no CP, podendo, em alguns casos, ultrapassar o tempo de ciclo máximo selecionado. Caso o tempo de execução limite for ultrapassado somente devido aos comandos da comunicação serial (monitoração, forçamento e demais), o CP não entrará em estado de erro. É possível, portanto, a indicação de um tempo de ciclo máximo maior que o selecionado sem que o controlador programável tenha entrado em modo erro.

O procedimento de compactação da memória de programa do controlador programável também segue a regra anterior. Em alguns casos, a rotina de compactação necessita copiar um módulo muito extenso na memória entre dois ciclos do programa aplicativo, aumentando exageradamente o tempo de execução de uma varredura. Nesta situação o CP não entrará em estado de erro.

Deve-se tomar cuidados especiais quando os tempos de ciclo de execução se aproximam do tempo máximo selecionado. O simples fato do programa aplicativo estar executando corretamente com as condições mais comuns dos pontos de entrada não garante que seu tempo de varredura, em condições reais de funcionamento da máquina, permaneça dentro do valor limite.

### ATENÇÃO:

Cada projeto de programação deve ser examinado cuidadosamente em busca de situações que irão provocar os maiores tempos de execução. Essas situações devem ser simuladas e os tempos medidos, verificando se não são excessivos. Este procedimento deve ser realizado mesmo em projeto de programação com tempos de ciclo bem abaixo do limite, para assegurar o seu bom funcionamento.

É possível que em algumas varreduras isoladas o tempo de ciclo exceda o tempo máximo selecionado sem que o CP passe para modo erro, caso estas varreduras esporádicas não causem atrasos nos temporizadores do sistema.

**ATENÇÃO:**

Se o CP indicar o tempo de ciclo máximo maior que o selecionado sem que tenha havido uma compactação de memória, mesmo que continue normalmente em modo execução, deve-se examinar o programa para diminuir o seu tempo de ciclo nas situações que causem maiores tempos.

**☺DICA:**

Existem alguns procedimentos típicos para diminuir o tempo de execução de programas aplicativos muito extensos. Um bom gerenciamento na chamada de módulos pode diminuir sensivelmente o tempo de ciclo total, sendo realizadas chamadas de poucos módulos do programa aplicativo em cada varredura, não permitindo que todos sejam disparados em um mesmo ciclo. O uso de instruções de salto dentro dos módulos, diminui o tempo de execução dos mesmos, pois um trecho de programa aplicativo saltado é desconsiderado pelo software executivo. As instruções relé mestre e fim de relé mestre (RM e FRM) não possuem esta propriedade, pois o segmento de programa aplicativo delimitado pelas mesmas continua a ser executado mesmo quando a bobina RM o desabilita.

**☺DICA:**

Deve-se realizar inicializações de valores em operandos ou tabelas dentro do módulo E000, idealizado especialmente para este propósito. A execução do módulo E000, por não ser cíclica, pode demorar mais que o tempo máximo, sendo este tempo desconsiderado na contagem do tempo da primeira varredura do módulo E001. Pelo modo como é executado, torna-se sem sentido a programação de temporizadores (TEE, TED) no módulo E000.

## Níveis de Proteção do CP

As UCPs da Série Ponto possuem um mecanismo de proteção do projeto de programação e dos operandos, permitindo o bloqueio da carga de módulos de programa, forçamentos de valores ou mesmo leituras de módulos e monitoração por operadores não autorizados.

Estas características são interessantes em processos críticos, para evitar modificações acidentais nos dados ou no programa de controle ou na necessidade de sigilo dos mesmos.

O bloqueio de operações é realizado através de níveis de proteção, que podem ser definidos apenas por operadores que conheçam uma senha preestabelecida. O controlador pode operar em quatro diferentes níveis de proteção:

- **Nível 0** - todas as operações no CP são permitidas.
- **Nível 1** - não é possível alterar o projeto de programação (apagar ou carregar novos módulos de programa) ou mudar o estado do CP. Pode-se forçar e monitorar operandos e ler módulos do programa.
- **Nível 2** - Possui as mesmas restrições do nível 1 e também não é possível ler módulos do programa.
- **Nível 3** - Possui as mesmas restrições dos níveis 1 e 2 e também não é possível monitorar ou forçar operandos nem mudar o estado do CP. Também não será permitido consultar o diretório de módulos nem liberar forçamento de operandos.

A troca do nível de proteção é realizada com as opções **Comunicação, Estado, Proteção** no MasterTool, devendo-se digitar a senha de acesso correta para efetivá-la. O nível de proteção do CP pode ser consultado com o MasterTool através das opções **Comunicação, Estado, Informações**.

A utilização de níveis de proteção diferentes de zero permite que somente pessoas autorizadas, que conheçam a senha, modifiquem o programa ou os dados do CP. Operadores não autorizados, mesmo dispondo do MasterTool, ficam impedidos de realizar alterações inadvertidas.

A senha de acesso pode ter de um a oito caracteres alfanuméricos. É definida ou trocada com as opções **Comunicação, Estado, Senha**, devendo-se digitar a senha anterior e a nova senha duas vezes, para ser confirmada a mudança.

O CP é fornecido sem senha. Não é necessário digitar qualquer valor no campo senha anterior para a definição da primeira senha.

**ATENÇÃO:**

A senha deve ser escrita e guardada em lugar seguro. Em caso de perda da senha programada no CP, entrar em contato com a ALTUS.

A proteção do CP atua não somente para as operações realizadas com o MasterTool, mas também para os comandos recebidos pelas redes ALNET I e ALNET II (via Ethernet), com as mesmas características definidas para cada nível.

Para maiores informações sobre como alterar o nível de proteção e a senha do CP, ver itens **Alterando o Nível de Proteção** e **Alterando a Senha** na seção **Comunicando com o CP ou Roteador** no Manual de Utilização do MasterTool.

## Intertravamento de Comandos no CP

Na Série Ponto pode-se utilizar as redes de comunicação ALNET I e ALNET II (via Ethernet) em conjunto. Quando interligado desta forma, é possível a recepção simultânea de dois comandos cuja execução concorrente seja indesejável, devido às suas características. Por exemplo, o CP pode receber um comando de transferência de módulo da EPROM para a RAM pela ALNET II enquanto o mesmo módulo está sendo carregado na ALNET I.

Situações semelhantes ocorrem com os comandos de transferência de módulos de programa da memória EPROM para RAM, de RAM para flash ou de apagamento da memória flash. A execução destes comandos pode estender-se por vários segundos, durante os quais o CP pode receber outros comandos que entrem em conflito com a operação em curso. Por exemplo, o CP pode receber um comando para apagar a memória flash enquanto um módulo estava sendo transferido para a mesma memória.

Para resolver as possíveis situações de conflito, há um mecanismo de intertravamento para a execução de alguns comandos disponíveis no CP. Estes comandos não podem ser executados caso o CP esteja realizando uma operação específica. Existem dois sinais internos, **carregando módulo (CM)** e **compactando RAM (CR)**, que são usados para este propósito. As tabelas 2-4 e 2-5 mostram os comandos que utilizam o intertravamento e o acionamento dos sinais.

O estado dos sinais carregando módulo e compactando RAM pode ser verificado na janela de informações do CP, opções **Comunicação, Estado, Informações** no MasterTool. Enquanto qualquer um dos sinais estiver acionado, o LED DG do painel do CP informa o estado.

Operação realizada no CP	Comando bloqueado (ALNET I, ALNET II)	Sinal ligado
Carga de Módulos	Carga de módulos Transferência de EPROM para RAM Transferência de RAM para Flash Pedido de carga de módulos Reabilitação de módulos em EPROM Apagamento de Flash EPROM Compactação	CM
Transferência de EPROM para RAM	Carga de módulos Transferência de EPROM para RAM Transferência de RAM para Flash Pedido de carga de módulos Reabilitação de módulos em EPROM Apagamento de Flash EPROM Compactação	CM
Transferência de RAM para Flash	Carga de módulos Transferência de EPROM para RAM Transferência de RAM para Flash Pedido de carga de módulos Reabilitação de módulos em EPROM Apagamento de Flash EPROM Compactação	CM
Apagamento de Flash EPROM	Carga de módulos Transferência de EPROM para RAM Transferência de RAM para Flash Pedido de carga de módulos Reabilitação de módulos em EPROM	CM

	Apagamento de Flash EPROM Compactação	
Legenda: CM - Carregando Módulo		

Tabela 2-4 Intertravamento de Comandos no CP (carregando módulo)

Operação realizada no CP	Comando bloqueado (ALNET I, ALNET II)	Sinal ligado
Compactação	Carga de módulos Transferência de EPROM para RAM Transferência de RAM para Flash Pedido de carga de módulos Reabilitação de módulos em EPROM Remoção de módulos Compactação	CR
Legenda: CR - Compactando RAM		

Tabela 2-5 Intertravamento de Comandos no CP (Compactando RAM)

Por exemplo, enquanto um módulo está sendo carregado no CP pela rede ALNET I, os comandos de carga de módulos, transferência de EPROM para RAM, transferência de RAM para flash, pedido de carga de módulos, reabilitação de módulos em EPROM, apagamento de Flash EPROM e compactação não podem ser executados, caso sejam recebidos através da outra rede. Se forem recebidos pelo CP, uma resposta indicando a impossibilidade de sua execução é transmitida para o solicitante.

# Instruções

Este capítulo apresenta a lista de instruções integrantes da Linguagem de Diagramas e Relés ALTUS, descrevendo o formato, o uso, a sintaxe e fornecendo exemplos de cada instrução.

## Lista das Instruções

Os CPs ALTUS utilizam a linguagem de relés e blocos para a elaboração do programa aplicativo, cuja principal vantagem, além de sua representação gráfica, é ser similar a diagramas de relés convencionais.

A programação desta linguagem, realizada através do MasterTool, utiliza um conjunto de poderosas instruções apresentadas nas seções seguintes.

As instruções do MasterTool podem ser divididas em 7 grupos:

- RELÉS
- MOVIMENTADORES
- ARITMÉTICOS
- CONTADORES
- CONVERSÕES
- GERAIS
- LIGAÇÕES

## Convenções Utilizadas

Foram utilizadas diversas convenções para a apresentação dos grupos e instruções tornando melhor a visualização e reconhecimento dos itens descritos, visando com isto um aprendizado mais simples e uma fonte de consulta direta aos tópicos desejados.

### Apresentação dos Grupos

A descrição de cada grupo segue o seguinte roteiro.

1. grupo é descrito com um título contendo o nome do grupo.
2. Logo após o título, é realizada uma breve descrição das características comuns às instruções do grupo.
3. Finalizando a apresentação do grupo, é exibida uma tabela contendo na primeira coluna o nome da instrução, na segunda coluna a descrição do nome da instrução e na terceira coluna a sequência de teclas para realizar a inserção da instrução diretamente pelo teclado.

### Exemplo:

#### Instruções do Grupo Relés

As instruções do grupo **Relés** são utilizadas para o processamento lógico dos diagramas de relés. Através das mesmas pode-se manipular os valores dos pontos digitais de entrada (%E) e saída (%S), bem como pontos de operandos auxiliares (%A), memória (%M) e decimal (%D).

São usadas também para desvio do fluxo e controle do processamento do programa aplicativo.

Nome	Descrição do Nome	Seqüência de Edição	Barra de Ferramentas
<b>RNA</b>	contato normalmente aberto	ALT, R, A	
<b>RNF</b>	contato normalmente fechado	ALT, R, F	
<b>BOB</b>	bobina simples	ALT, R, B	
<b>SLT</b>	bobina de salto	ALT, R, S	
<b>BBL</b>	bobina liga	ALT, R, L	
<b>BBQ</b>	bobina desliga	ALT, R, D	
<b>PLS</b>	relé de pulso	ALT, R, P	
<b>FRM</b>	fim de relé mestre	ALT, R, M	
<b>RM</b>	relé mestre	ALT, R, R	

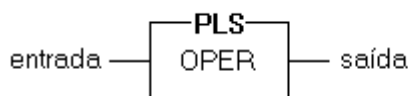
### Apresentação das Instruções

A descrição de cada instrução é feita da seguinte maneira.

1. A instrução é descrita com um título contendo o nome da instrução e a descrição do nome. Uma figura apresentando como a instrução é visualizada no diagrama de relés contendo seus operandos, entradas e saídas. Abaixo da figura, é exibida uma breve descrição do significado de cada operando.
2. O item **Descrição** contém informações descrevendo o funcionamento da instrução conforme as entradas habilitadas e os tipos de operando utilizados. Neste item também são descritas as saídas que serão acionadas após a execução da instrução.
3. O item **Sintaxe** descreve as combinações de operandos que podem ser utilizados na instrução. Este item somente esta presente nas instruções que possuam operandos.
4. O item **Exemplo** fornece um exemplo de utilização da instrução descrevendo seu comportamento. Este item somente esta presente nas instruções que requeiram uma detalhamento maior de seu funcionamento.
5. Podem existir outros itens descrevendo uma característica específica da instrução caso haja necessidade.

Exemplo:

#### PLS - Relé de Pulso



#### Descrição:

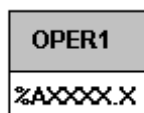
A instrução relé de pulso gera um pulso de uma varredura em sua saída, ou seja, permanece energizada durante uma varredura do programa aplicativo quando o estado da sua entrada passar de desenergizado para energizado.

O relé auxiliar declarado serve como memorizador, evitando limitações quanto ao número de instruções de pulso presentes no programa aplicativo.

#### ATENÇÃO:

O valor do relé auxiliar não deve ser utilizado em nenhum outro ponto do programa aplicativo.

#### Sintaxe:



## Instruções do Grupo Relés

As instruções do grupo **Relés** são utilizadas para o processamento lógico dos diagramas de relés. Através das mesmas pode-se manipular os valores dos pontos digitais de entrada (%E) e saída (%S), bem como pontos de operandos auxiliares (%A), memória (%M) e decimal (%D).

São usadas também para desvio do fluxo e controle do processamento do programa aplicativo.










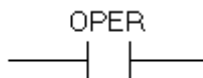
Nome	Descrição do Nome	Seqüência de Edição	Barra de Ferramentas
<b>RNA</b>	Contato normalmente aberto	ALT, R, A	
<b>RNF</b>	Contato normalmente fechado	ALT, R, F	
<b>BOB</b>	Bobina simples	ALT, R, B	
<b>SLT</b>	Bobina de salto	ALT, R, S	
<b>BBL</b>	Bobina liga	ALT, R, L	
<b>BBD</b>	Bobina desliga	ALT, R, D	
<b>PLS</b>	Relé de pulso	ALT, R, P	
<b>FRM</b>	Fim de relé mestre	ALT, R, M	
<b>RM</b>	Relé mestre	ALT, R, R	

Tabela 3-1 Instruções do Grupo Relés



## Contatos

- **RNA** contato normalmente aberto



- **RNF** contato **normalmente** fechado



### Descrição:

Estas instruções refletem, logicamente, o comportamento real de um contato elétrico de um relé no programa aplicativo.

O contato normalmente aberto fecha conforme o estado do seu operando associado. Caso o ponto do operando esteja no estado lógico **1** ou **0**, o contato normalmente aberto está fechado ou aberto, respectivamente.

O contato normalmente fechado possui comportamento oposto ao normalmente aberto. Caso o ponto do operando associado esteja no estado lógico **1** ou **0**, o contato normalmente fechado está aberto ou fechado, respectivamente.

Quando um contato está fechado, a instrução transmite o estado lógico da sua entrada para a sua saída. Se estiver aberto, o valor da entrada não é colocado na saída.

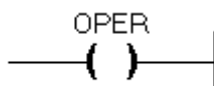
### Sintaxe:

OPER1
%EXXXX.X
%SXXXX.X
%AXXXX.X
%MXXXX.X
%DXXXX.X
%DXXXXhX

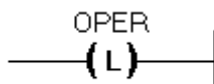
Tabela 3-2 Sintaxe das Instruções RNA e RNF

## Bobinas

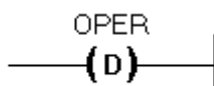
- **BOB** Bobina Simples



- **BBL** Bobina Liga



- **BBD** Bobina Desliga



### Descrição:

As instruções bobina modificam o estado lógico do operando na memória imagem do controlador programável, conforme o estado da linha de acionamento das mesmas.

A bobina simples liga ou desliga o ponto do operando conforme a linha de acionamento, enquanto que as bobinas do tipo liga e do tipo desliga somente ligam ou desligam os operandos quando a linha está energizada ("set"/"reset").

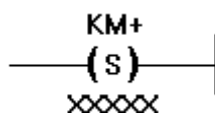
Estas instruções somente podem ser posicionadas na coluna 7 da lógica.

### Sintaxe:

OPER1
%SXXXX.X
%AXXXX.X
%MXXXX.X
%DXXXX.X
%DXXXXhX

Tabela 3-3 Sintaxe das Instruções BOB, BBL e BBD

## SLT - Bobina de Salto



### Descrição:

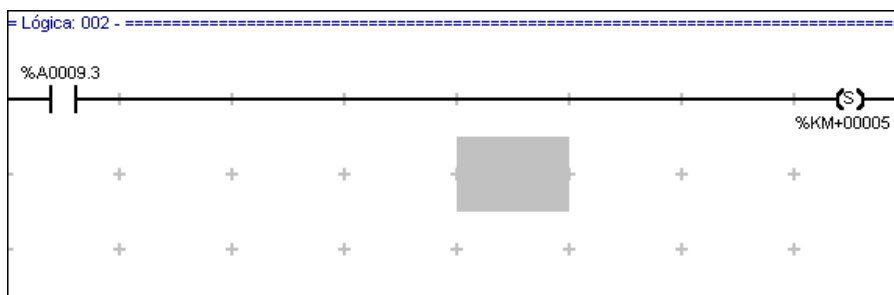
A instrução bobina de salto serve para controlar a seqüência de execução de um programa aplicativo, sendo usada para desviar o processamento do mesmo para uma lógica determinada.

Seu operando é uma constante que determina o número de lógicas a serem saltadas a partir da energização da bobina. A determinação da lógica destino é realizada pela soma da constante que acompanha a instrução com o número da lógica onde a mesma se encontra.

Quando a linha de acionamento da bobina de salto estiver desenergizada, o salto não ocorre, e a instrução seguinte àquela em que esta bobina está declarada é executada.

### Exemplo:

Supondo que a instrução a seguir esteja na lógica 2, a execução do programa aplicativo é desviada para a lógica 7 se a linha de acionamento estiver energizada, ou seja, se o valor do ponto %A0009.3 for 1. Se o valor deste ponto for 0, a execução continuará normalmente na lógica 003.



**Figura 3-1 Exemplo da Instrução SLT**

Esta instrução pode ser posicionada somente na coluna 7 da lógica.

Pode ser utilizada nesta instrução uma constante %KM com valor zero ou mesmo com valor negativo. Se programada com o valor zero, a lógica destino é a mesma que contém a bobina de salto, quando esta é energizada. Ou seja, o processamento é desviado para o início da própria lógica da bobina. Se o valor programado é negativo, o processamento é desviado para uma lógica anterior à lógica que contém a bobina de salto.

### ATENÇÃO:

O uso de constante zero ou negativa corresponde a um uso não convencional da instrução. Caso deseje-se utilizá-la, deve-se tomar os cuidados necessários para evitar a entrada em laço infinito de execução "loop" ou o aumento excessivo do tempo de ciclo do programa aplicativo. Recomenda-se, contudo, a utilização da bobina de salto somente com constantes positivas maiores que zero.

O controle da execução nestas situações deve ser realizado através de um intertravamento que desligue o salto para a lógica anterior, após um certo número de laços executados no trecho de retorno.

Caso a lógica destino ultrapasse a última lógica do programa aplicativo, o CP salta para o final do programa e continua seu ciclo normal.

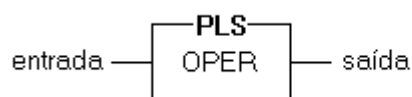
Caso a lógica destino de um salto de retorno seja menor do que a primeira lógica do programa aplicativo, a execução é reiniciada a partir da lógica 0.

Sintaxe:

OPER1
%KM+XXXXX
%KM-XXXXX

Tabela 3-4 Sintaxe da Instrução SLT

## PLS - Relé de Pulso



### Descrição:

A instrução relé de pulso gera um pulso de uma varredura em sua saída, ou seja, permanece energizada durante uma varredura do programa aplicativo quando o estado da sua entrada passar de desenergizado para energizado.

O relé auxiliar declarado serve como memorizador, evitando limitações quanto ao número de instruções de pulso presentes no programa aplicativo.

### ATENÇÃO:

O valor do relé auxiliar não deve ser modificado em nenhum outro ponto do programa aplicativo.

### Sintaxe:

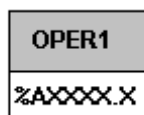


Tabela 3-5 Sintaxe da Instrução PLS

## RM, FRM - Relé Mestre, Fim de Relé Mestre

- **RM** Relé Mestre



- **FRM** Fim de Relé Mestre



### Descrição:

As instruções relé mestre e fim de relé mestre são utilizadas para delimitar trechos de programas aplicativos, energizando ou não a barra lógica de alimentação nos mesmos, conforme o estado da sua linha de acionamento.

Estas instruções não necessitam de operandos, podendo ser posicionadas somente na coluna 7 da lógica.

Quando a entrada da instrução RM estiver desenergizada, a barra lógica de alimentação é desenergizada desde a lógica seguinte até a lógica que contém a instrução FRM.

Como estas instruções atuam sempre na lógica seguinte a que estão contidas é aconselhável o seu posicionamento sempre como últimas instruções da lógica em que estiverem presentes. Assim sendo, o trecho de programa aplicativo delimitado visualmente pelas instruções no diagrama corresponde exatamente ao controlado pelas mesmas, evitando assim má interpretação de seu funcionamento.

### ATENÇÃO:

As instruções CON, COB, TEE e TED contém saídas energizadas mesmo sem o acionamento das suas entradas. Estas saídas permanecem energizadas mesmo dentro de um trecho sob comando de um relé mestre desenergizado, podendo causar acionamentos indesejáveis.

## Instruções do Grupo Movimentadores

Estas instruções são utilizadas para a manipulação e transferência de valores numéricos entre constantes, operandos simples ou tabelas de operandos.







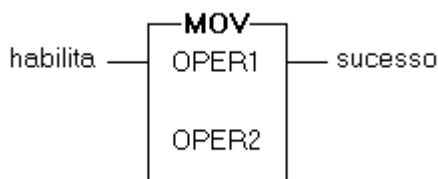
Nome	Descrição do Nome	Sequência de Edição	Barra de Ferramentas
<b>MOV</b>	Movimentação de operandos simples	ALT, M, V	
<b>MOP</b>	Movimentação de partes de operandos	ALT, M, P	
<b>MOB</b>	Movimentação de blocos de operandos	ALT, M, B	
<b>MOT</b>	Movimentação de tabelas de operandos	ALT, M, T	
<b>CES</b>	Conversão de entradas ou saídas	ALT, M, S	
<b>CAB</b>	carrega bloco	ALT, M, C	

Tabela 3-6 Instruções do Grupo Movimentadores

## MOV - Movimentação de Operandos Simples



OPER1 - operando origem

OPER2 - operando destino

### Descrição:

Esta instrução move o conteúdo de operandos simples, quando a entrada habilita é acionada.

O operando que ocupa a primeira célula da instrução (OPER1) é o operando origem, cujo valor é movimentado para o operando destino, especificado na segunda célula (OPER2).

Se o formato do operando destino for menor que o do origem, os octetos mais significativos do valor origem são desprezados. Se o formato do destino for maior, seus octetos mais significativos são zerados. Se o operando origem for real e o destino memória, o valor movido corresponderá a parte inteira (trunca) do número real. Se a movimentação for realizada, a saída sucesso é acionada.

Se os índices indiretos excederem os limites de operandos declarados no módulo de configuração, a movimentação não é efetuada e a saída sucesso não é ligada.

Não é permitida a movimentação de subdivisões de operandos. Para isto, deve ser usada a instrução MOP.

Quando o operando destino da instrução é um inteiro (%M) e pelo menos um dos demais operandos da instrução é um real (%F) o resultado armazenado será truncado, ou seja, armazena-se no operando M apenas a parte inteira do resultado da operação, desprezando-se a parte fracionária.

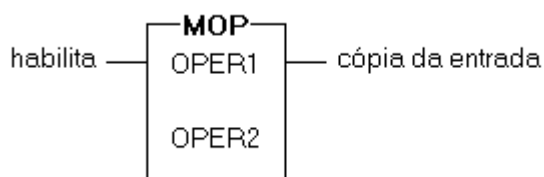
### Sintaxe:

OPER1	OPER2	OPER1	OPER2
%E	%E		
%S	%S		
%A	%A		
%M	%M	%M	
%I	%I	%F	%M
%D	%D	%I	%F
%M*E	%M*E	%M*M	%I
%M*S	%M*S	%M*F	%M*M
%M*A	%M*A	%M*I	%M*F
%M*M	%M*M	%KM	%M*I
%M*I	%M*I	%KF	
%M*D	%M*I	%KI	
%KM	%M*D		
%KD			

Tabela 3-7 Sintaxe da Instrução MOV



## MOP - Movimentação de Partes (Subdivisões) de Operandos



OPER1 - operando origem

OPER2 - operando destino

### Descrição:

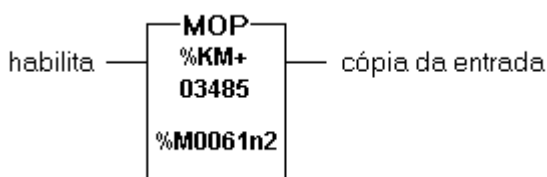
Esta instrução move conteúdos de partes de operandos simples (palavras, octetos, "nibbles", pontos) quando a entrada habilita é energizada. Não é realizada a conversão entre tipos de operandos, apenas a movimentação dos valores.

O operando que ocupa a primeira célula da instrução (OPER1) é o operando origem, cujo valor é movimentado para o operando destino especificado na segunda célula (OPER2). O tipo de subdivisão usado no primeiro operando deve ser o mesmo do segundo.

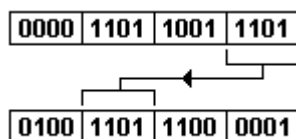
### ATENÇÃO:

Se a movimentação é realizada de uma constante para um operando, é considerada sempre a subdivisão menos significativa da constante igual à declarada no operando destino. Devido a esta característica, sugere-se que sempre seja declarado na constante origem o valor real a ser movimentado, para maior clareza do programa.

### Exemplo:



O operando destino da instrução está declarado com subdivisão de nibble. Portanto, o nibble menos significativo da constante origem (com valor igual a 1101 em binário, 13 em decimal) será movido para o nibble 2 da memória M0061.



**Figura 3-2 Exemplo da Instrução MOP**

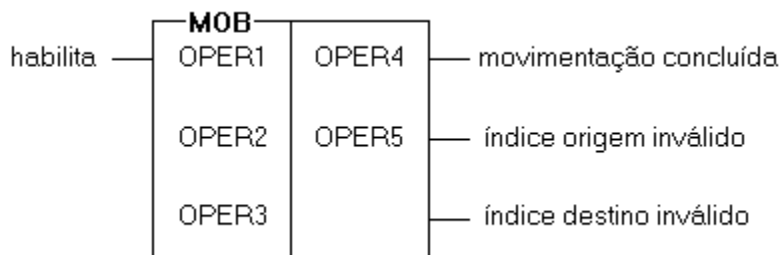
Os demais bits que compõem a constante são desprezados, ou seja, o resultado da movimentação seria idêntico utilizando-se uma constante %KM00013. O exemplo apresentado utiliza um valor excedente ao da movimentação para melhor ilustrar o funcionamento da MOP. Para melhor interpretação do programa deve-se utilizar o valor %KM00013.

Sintaxe:

OPER1	OPER2	OPER1	OPER2
%EXXX.X %SXXX.X %AXXX.X %MXXX.X %DXXX.X %DXXXhX %FXXX.X %FXXXhX %IXXX.X %IXXXhX %KMXXXXX %KDXXXXX	%EXXX.X %SXXX.X %AXXX.X %MXXX.X %DXXX.X %DXXXhX %FXXX.X %FXXXhX %IXXX.X %IXXXhX %IXXXhX	%MXXXbX %DXXXbX %FXXXbX %IXXXbX %EXXX %SXXX %AXXX %KMXXXXX %KDXXXXX	%MXXXbX %DXXXbX %FXXXbX %IXXXbX
OPER1	OPER2	OPER1	OPER2
%EXXXnX %SXXnX %AXXXnX %MXXnX %DXXnX %FXXnX %IXXXnX %KMXXXXX %KDXXXXX	%EXXXnX %SXXnX %AXXXnX %MXXnX %DXXnX %FXXnX %IXXXnX %IXXnX	%MXXXbX %DXXXbX %FXXXbX %IXXXbX	%EXXX %SXXX %AXXX
OPER1	OPER2	OPER1	OPER2
%DXXXwX %FXXXwX %IXXXwX %MXXX %KMXXXXX %KDXXXXX	%DXXXwX %FXXXwX %IXXXwX	%DXXXwX %FXXXwX %IXXXwX	%MXXX

Tabela 3-8 Sintaxes da Instrução MOP

## MOB - Movimentação de Blocos de Operandos



OPER1 - primeiro operando do bloco origem

OPER2 - número de transferências a realizar

OPER3 - operando de controle

OPER4 - primeiro operando do bloco destino

OPER5 - número de transferências por varredura

### Descrição:

Esta instrução realiza a cópia dos valores de um bloco de operandos origem para um bloco destino.

Especifica-se o primeiro operando do bloco origem em OPER1 e o primeiro operando do bloco destino em OPER4. O número total de transferências a serem realizadas é declarado no parâmetro OPER2, devendo também ser especificados o número de transferências por varredura (OPER5) e uma memória acumuladora para a contagem do número de transferências (OPER3).

Se o bloco origem ou destino for uma tabela, a transferência tem início na sua primeira posição.

Se o formato do operando destino for menor do que o origem, os octetos mais significativos do valor origem são desprezados. Caso contrário, os octetos mais significativos do destino são zerados.

O número de transferências por varredura é limitado em 255 operandos. Na medida do possível deve-se evitar um número elevado de transferências na mesma varredura, para diminuir o tempo de execução do programa.

Em cada instrução MOB é utilizada uma memória como operando de controle (OPER3), que deve estar zerada antes da primeira execução.

### ATENÇÃO:

O operando de controle não deve ter seu conteúdo alterado em nenhuma parte do programa aplicativo, sob pena de prejudicar a execução correta da instrução. Cada ocorrência desta instrução no programa deve possuir um operando de controle exclusivo, diferente dos demais. Este operando não pode ser retentivo.

Quando ligadas, as saídas da segunda e terceira células indicam, respectivamente, que pelo menos um dos operandos componentes do bloco origem ou destino tem endereço superior ao número máximo declarado para o operando ou tabela utilizado, não sendo realizada nenhuma movimentação. Caso o valor do segundo operando seja negativo, a saída **índice origem inválido** é acionada.

A saída da primeira célula é acionada na varredura em que a movimentação for completada.

### ATENÇÃO:

A entrada **habilita** deve permanecer ativa até que a movimentação esteja concluída. Como esta instrução é executada em múltiplos ciclos de execução, não deve ser saltada enquanto não estiver terminada a movimentação.

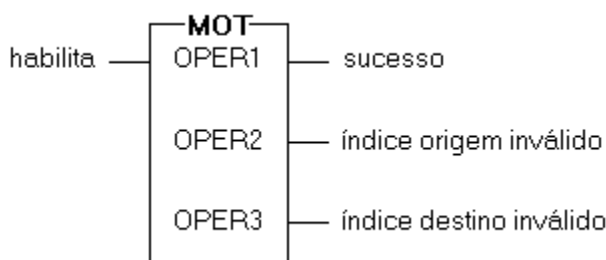
### Sintaxe:

OPER1	OPER2	OPER3	OPER4	OPER5
%E %S %A %M %I %D %TM %TD	%KM	%M	%E %S %A %M %I %D %TM %TD	%KM

OPER1	OPER2	OPER3	OPER4	OPER5
%F %TF	%KM	%M	%F %TF	%KM

Tabela 3-9 Sintaxe da Instrução MOB

## MOT - Movimentação de Tabelas



OPER1 - tabela origem ou operando origem

OPER2 - índice da tabela

OPER3 - operando destino ou tabela destino

### Descrição:

Esta instrução permite duas operações: transferir o valor de uma posição de tabela para um operando simples ou de um operando simples para uma posição de tabela.

O operando que ocupa a primeira célula da instrução (OPER1) é o operando origem, cujo valor é movimentado para o operando destino especificado na terceira célula (OPER3). OPER2 contém a posição da tabela declarada em OPER1 ou OPER3.

### Leitura de conteúdo de tabela:

Permite ler o conteúdo de uma posição de tabela e carregá-lo em um operando memória ou decimal.

A instrução é programada da seguinte forma:

- **OPER1** - especifica o endereço da tabela a ser lida
- **OPER2** - especifica a posição (%KM) a ser lida ou a memória (%M) que contém esta posição
- **OPER3** - especifica para onde o conteúdo da posição de tabela deve ser transferido

Se o primeiro operando referenciar indiretamente uma tabela não especificada, ou se o valor do segundo operando for negativo ou maior que a última posição definida para a tabela, a transferência não é realizada e a saída **índice origem inválido** é acionada. Se o terceiro operando referenciar indiretamente um operando não declarado, a transferência não é realizada e a saída **índice destino inválido** é acionada.

### Escrita de valores em tabela:

Permite escrever um valor constante ou o conteúdo de um operando memória ou decimal em uma posição de tabela.

A instrução é programada da seguinte forma:

- **OPER1** - especifica o operando origem
- **OPER2** - especifica a posição (%KM) a ser escrita na tabela ou a memória (%M) que contém esta posição
- **OPER3** - especifica o endereço da tabela para onde é transferido o conteúdo

Se o primeiro operando referenciar indiretamente um operando não declarado, a transferência do conteúdo não é realizada e a saída **índice origem inválido** é acionada. Se o valor do segundo operando for negativo ou maior que a última posição definida para a tabela, ou se o terceiro operando referenciar indiretamente uma tabela não especificada, a transferência do conteúdo não é realizada e a saída **índice destino inválido** é acionada.

Esta instrução simplifica a programação de uma série de algoritmos envolvendo decodificações, sequenciamentos, geração de curvas, armazenamento e comparação de valores, entre outros.

Sintaxe:

Leitura:

Escrita:

OPER1	OPER2	OPER3
%TM %M*TM	%KM %M	%M %M*M

OPER1	OPER2	OPER3
%KM %M %M*M	%KM %M	%TM %M*TM

OPER1	OPER2	OPER3
%TD %M*TD	%KM %M	%D %M*D

OPER1	OPER2	OPER3
%KD %D %M*D	%KM %M	%TD %M*TD

OPER1	OPER2	OPER3
%TF %M*TF	%KM %M	%F %M*F

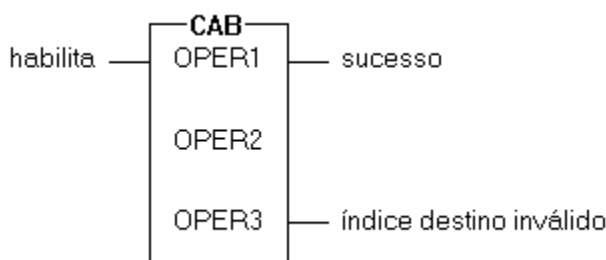
OPER1	OPER2	OPER3
%KF %F %M*F	%KM %M	%TF %M*TF

OPER1	OPER2	OPER3
%TI %M*TI	%KM %M	%I %M*I

OPER1	OPER2	OPER3
%KI %I %M*I	%KM %M	%TI %M*TI

Tabela 3-10 Sintaxes da Instrução MOT

## CAB - Carrega Bloco



OPER1 - operando inicial ou tabela a ser carregada

OPER2 - número de operandos ou posições de tabela

OPER3 - tabela de constantes a serem carregadas

### Descrição:

Esta instrução permite a carga de até 255 valores constantes em um bloco de operandos ou em tabelas.

O operando inicial ou tabela a ser carregada é especificado no primeiro parâmetro (OPER1), o número de operandos ou posições da tabela a serem carregados no segundo operando (OPER2) e o valor das constantes no terceiro (OPER3).

O valor do segundo operando deve ser positivo, menor ou igual a %KM+128.

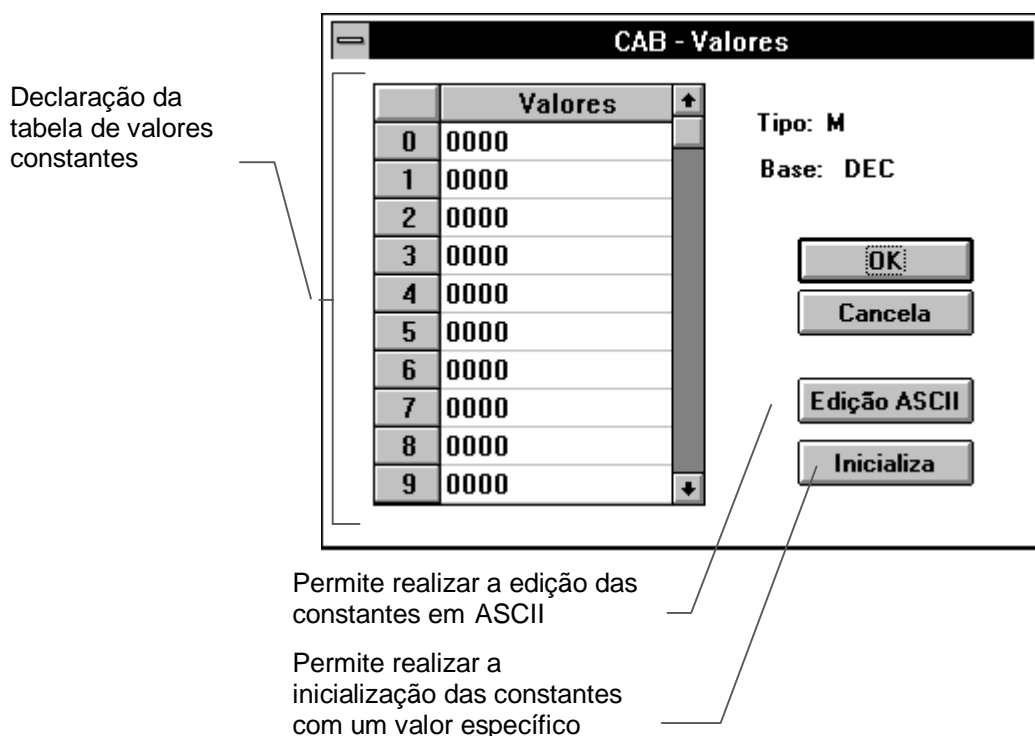
O terceiro operando (OPER3) é composto por uma tabela de valores constantes a serem carregados. Estes valores são declarados selecionando o botão **Bloco**, sendo aberta uma janela de edição no MasterTool. As constantes são do tipo %KM se o tipo do primeiro operando for %E, %S, %A, %M, %TM, do tipo %KD se o primeiro operando for %D ou %TD e do tipo %KF se o primeiro operando for %F ou %TF. Caso o primeiro operando seja um octeto (%E, %S ou %A), somente serão movimentados os valores dos octetos menos significativos de cada constante declarada.

Também é possível realizar a declaração dos valores da tabela em ASCII. Este modo permite que seja digitado um texto que será carregado na tabela com os valores ASCII relativos a cada caractere. Neste modo é possível a inserção de endereços ou tags de operandos que devem representar o seu valor no momento em que a instrução for executada. O endereço ou tag do operando deve ser digitado entre chaves ({}).

Ex.: Supondo que %M0000 tenha o valor 35 e que tenha sido carregado o seguinte texto em ASCII "Valor de %M0000: {%M0000}.".O texto fica o seguinte:

Valor de %M0000: 00035.

Quando o botão **Bloco** é selecionado é exibida a caixa de diálogo **CAB - Valores**:



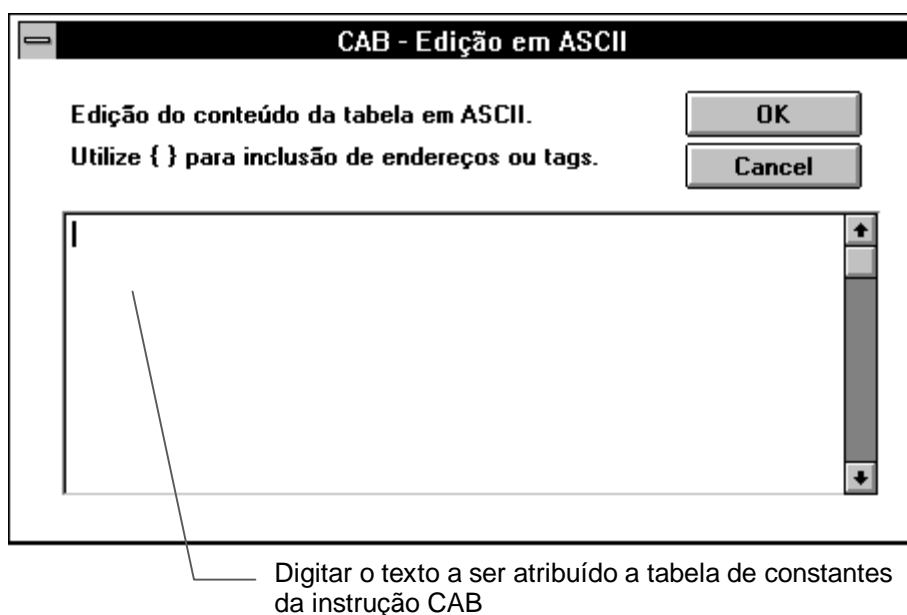
**Figura 3-3 Caixa de Diálogo CAB - Valores**

**Para realizar a edição das constantes**

1. Posicionar o cursor no índice a ser editado. Caso seja necessário rolar as páginas podem ser utilizadas as teclas PAGE DOWN e PAGE UP ou a barra de rolagem vertical.
2. Digitar o valor da constante.

**Para realizar a edição em ASCII**

1. Selecionar o botão **Edição ASCII**. É exibida a caixa de diálogo **CAB - Edição em ASCII**.
2. Digitar o texto que se deseja carregar nas constantes da CAB e selecionar o botão Ok.

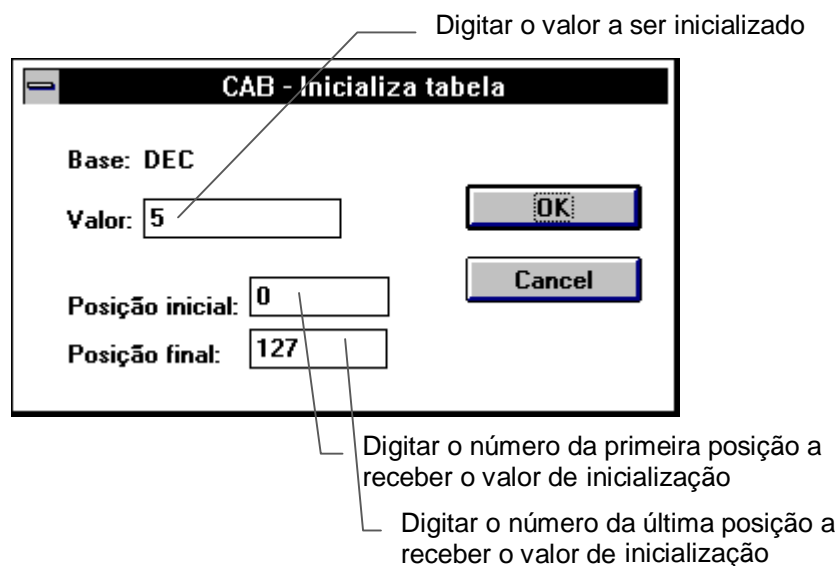


**Figura 3-4 Caixa de Diálogo CAB - Edição em ASCII**



**Para inicializar as constantes com um valor específico**

1. Selecionar o botão **Inicializar**. É exibida a janela **CAB - Inicializa tabela**.
2. No item **Valor**, digitar o valor a ser inicializado nas constantes.
3. No item **Posição inicial**, digitar o número da primeira posição a receber o valor de inicialização.
4. No item **Posição final**, digitar o número da última posição a receber o valor de inicialização.
5. Selecionar o botão **Ok**.

**Figura 3-5 CAB - Inicializa Tabela**

A saída **índice destino inválido** é acionada quando algum operando não puder ser acessado ou uma posição de tabela não existir. A saída sucesso é acionada sempre que a instrução for executada corretamente. Se a saída **índice destino inválido** foi acionada, nenhuma carga de constantes ocorreu.

A carga dos valores constantes é inteiramente realizada em uma só varredura do programa aplicativo, podendo ocasionar um tempo de ciclo excessivo quando o mesmo for extenso. Na maior parte dos programas aplicativos, a instrução CAB pode ser executada somente na inicialização do mesmo (carga de tabelas cujos conteúdos serão somente lidos) ou em alguns momentos especiais, não precisando ser chamada em todas as varreduras. Nestes casos, recomenda-se a sua programação no módulo de programa aplicativo de inicialização (E000) ou que seja acionada apenas nos momentos de carga necessários.

**Sintaxe:**

OPER1	OPER2	OPER3
%E %S %A %M %TM %M*E %M*S %M*A %M*M %M*TM	%KM	TABELA DE VALORES MEMÓRIA

OPER1	OPER2	OPER3
%D %TD %M*D %M*TD	%KM	TABELA DE VALORES DECIMAIS

OPER1	OPER2	OPER3
%F %TF %M*F %M*TF	%KM	TABELA DE VALORES REAIS

OPER1	OPER2	OPER3
%I %TI %M*I %M*TI	%KM	TABELA DE VALORES INTEIRO

Tabela 3-14 Sintaxes da Instrução CAB

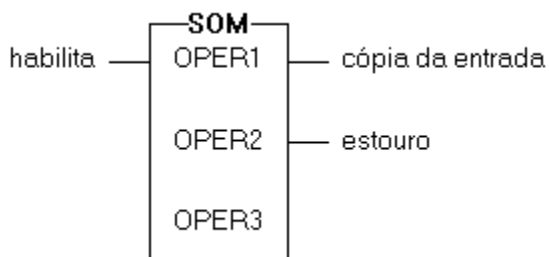
## Instruções do Grupo Aritméticas

As instruções aritméticas modificam os valores dos operandos numéricos, permitindo a realização de cálculos aritméticos e lógicos entre os mesmos. Permitem também comparações entre valores de operandos.

Nome	Descrição do Nome	Seqüência de Edição	Barra de Ferramentas
<b>SOM</b>	adição	ALT, A, S	
<b>SUB</b>	subtração	ALT, A, B	
<b>MUL</b>	multiplicação	ALT, A, M	
<b>DIV</b>	divisão	ALT, A, D	
<b>AND</b>	função E binário entre operandos	ALT, A, A	
<b>OR</b>	função OU binário entre operandos	ALT, A, O	
<b>XOR</b>	função OU EXCLUSIVO entre operandos	ALT, A, X	
<b>CAR</b>	carrega operandos	ALT, A, C	
<b>IGUAL</b>	igual	ALT, A, I	
<b>MENOR</b>	menor	ALT, A, N	
<b>MAIOR</b>	maior	ALT, A, R	

Tabela 3-15 Instruções do Grupo Aritméticas

### SOM - Adição



OPER1 - primeira parcela

OPER2 - segunda parcela

OPER3 - total

#### Descrição:

Esta instrução realiza a soma aritmética de operandos. Quando a entrada **habilita** é energizada, os valores dos operandos especificados nas duas primeiras células são somados e o resultado armazenado no operando da terceira célula.

Se o resultado da operação for maior ou menor do que o armazenável, a saída **estouro** é energizada e o máximo ou mínimo valor armazenável é atribuído a variável total como resultado.

Se a entrada **habilita** não está energizada, todas as saídas são desenergizadas e o valor de OPER3 não é alterado.

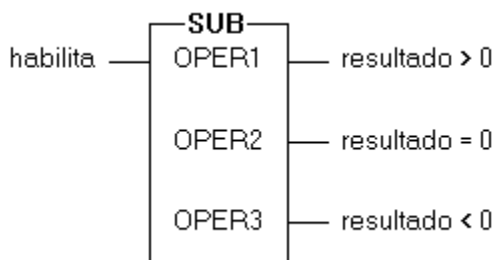
Sintaxe:

OPER1	OPER2	OPER3
%KD %D	%KD %D	%D

OPER1	OPER2	OPER3
%KF %F %KM %M %KI %I	%KF %F %KM %M %KI %I	%F %M %I

Tabela 3-16 Sintaxes da Instrução SOM

## SUB - Subtração



OPER1 - primeira parcela

OPER2 - segunda parcela

OPER3 - resultado

### Descrição:

Esta instrução realiza a subtração aritmética entre operandos. Quando **habilita** é energizada, o valor do operando da segunda célula é subtraído do valor do operando da primeira célula. O resultado é armazenado na memória especificada na terceira célula.

As linhas de saída **resultado > 0**, **resultado = 0** e **resultado < 0** podem ser usadas para comparações e são acionadas de acordo com o resultado da subtração.

Se a entrada **habilita** não está energizada, todas as saídas são desenergizadas e OPER3 permanece inalterado.

Se o resultado da operação excede o maior ou menor valor armazenável no operando, o respectivo valor limite é considerado como resultado.

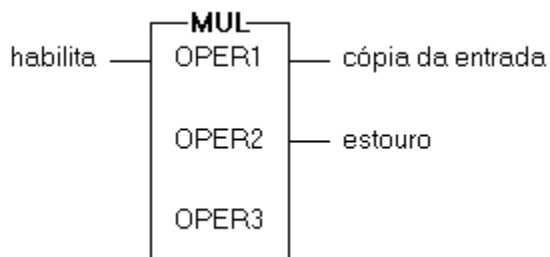
### Sintaxe:

OPER1	OPER2	OPER3
%KD %D	%KD %D	%D

OPER1	OPER2	OPER3
%KF %F %KM %M %KI %I	%KF %F %KM %M %KI %I	%F %M %I

Tabela 3-17 Sintaxes da Instrução SUB

## MUL - Multiplicação



OPER1 - multiplicando

OPER2 - multiplicador

OPER3 - produto

### Descrição:

Esta instrução realiza a multiplicação aritmética de operandos. Quando a entrada **habilita** está energizada, ocorre a multiplicação do conteúdo do operando especificado na primeira célula pelo especificado na segunda.

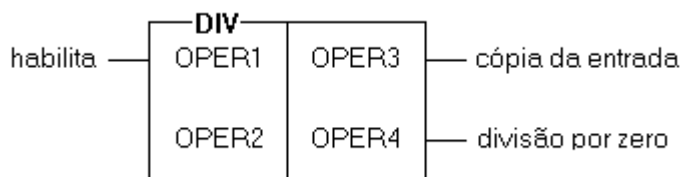
O resultado é armazenado na memória especificada na terceira célula. Caso este exceda o valor máximo armazenável em uma memória, o resultado final é este valor e a saída **estouro** é energizada. Se a entrada **habilita** é desenergizada, nenhuma saída é ligada e OPER3 permanecerá inalterado.

### Sintaxe:

OPER1	OPER2	OPER3
%KF	%KF	
%F	%F	%F
%KM	%KM	%M
%M	%M	%I
%KI	%KI	
%I	%I	

Tabela 3-18 Sintaxe da Instrução MUL

## DIV - Divisão



OPER1 - dividendo

OPER2 - divisor

OPER3 - quociente

OPER4 - resto

### Descrição:

Esta instrução realiza a divisão aritmética de operandos. Quando a entrada **habilita** está energizada, ocorre a divisão do valor do operando da primeira célula pelo da segunda, sendo o resultado armazenado na memória especificada na terceira célula e o resto da operação colocado no quarto operando. Os operandos da primeira e segunda células podem ser do tipo memória ou constante.

Se o valor do segundo operando for zero, a saída divisão por zero é acionada e em OPER3 é colocado o valor máximo ou mínimo armazenável no operando, conforme o sinal de OPER1. Neste caso, em OPER4 (resto) será armazenado zero. As saídas da instrução somente são energizadas se a entrada **habilita** estiver acionada. Se não estiver acionada, OPER3 e OPER4 permanecerão inalterados.

Sempre que o OPER1 (dividendo), OPER2 (divisor) ou OPER3 (quociente) for um operando do tipo real o quarto parâmetro (resto) será desconsiderado.

### Sintaxe:

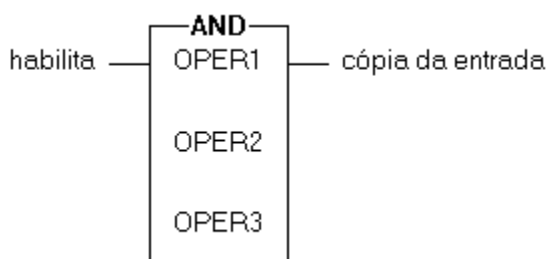
OPER1	OPER2	OPER3	OPER4
%KM %M %KI %I	%KM %M %KI %I	%M %I	%M %I

OPER1	OPER2	OPER3	OPER4
%KF %F %KM %M %KI %I	%KF %F %KM %M %KI %I	%F %M %I	%M (NU)

Tabela 3-19 Sintaxe da Instrução DIV

NU= Não Utilizado, apenas preencher com uma memória qualquer.

## AND - E Binário entre Operandos



OPER1 - primeiro operando

OPER2 - segundo operando

OPER3 - resultado

### Descrição:

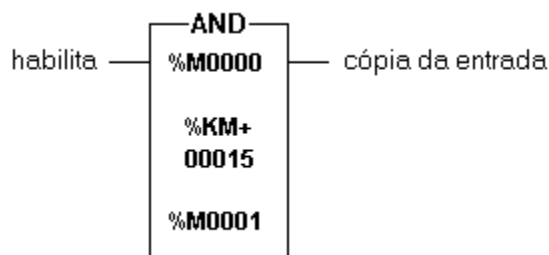
Esta instrução realiza a operação "e" binário entre os dois primeiros operandos, armazenando o resultado no terceiro.

A operação é realizada ponto a ponto entre os operandos. A tabela a seguir mostra as combinações da operação "e" ponto a ponto possíveis.

ponto OPER1	ponto OPER2	ponto OPER3 (resultado)
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 3-20 Operações Ponto a Ponto (AND)

### Exemplo:



Neste exemplo deseja-se preservar o valor do nibble menos significativo de %M0000, zerando o resto do operando. Se %M0000 contém 215 (11010111 binário), o resultado do "e" binário com 15 (00001111 binário) é 7 (00000111 binário).

	Decimal	Binário
	215	00000000 11010111 (conteúdo de %M0000)
AND	15	AND 00000000 00001111 (valor de %KM+00015)
	7	00000000 00000111 (resultado em %M0001)

Portanto, o valor 7 decimal é armazenado em %M0001.

### Sintaxe:

OPER1	OPER2	OPER3
%KM %M	%KM %M	%M

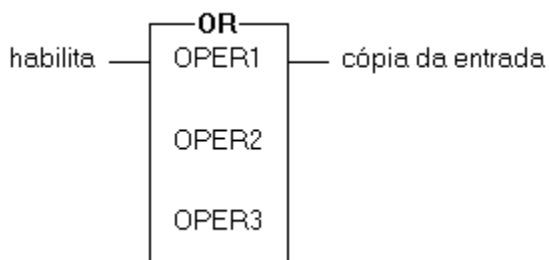
OPER1	OPER2	OPER3
%KD %D	%KD %D	%D



OPER1	OPER2	OPER3
%I %KI	%I %KI	%I

Tabela 3-21 Sintaxes da Instrução AND

## OR - Ou Binário entre Operandos



OPER1 - primeiro operando

OPER2 - segundo operando

OPER3 - resultado

### Descrição:

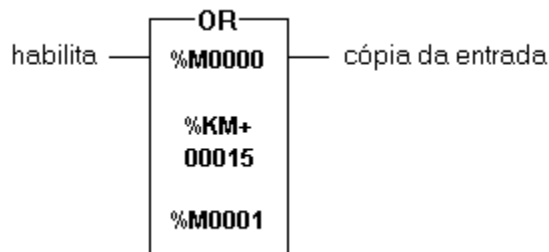
Esta instrução realiza a operação "ou" binário entre os valores dos dois primeiros operandos, armazenando o resultado no terceiro.

A operação é realizada ponto a ponto entre os operandos. A tabela a seguir mostra as combinações da operação "ou" ponto a ponto possíveis.

ponto OPER1	ponto OPER2	ponto OPER3 (resultado)
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 3-22 Operações Ponto a Ponto (OR)

### Exemplo:



Neste exemplo deseja-se forçar o nibble menos significativo de %M0000 para 1, preservando-se o valor nos outros nibbles. Se %M0000 contém 28277 (0110111001110101 binário) o resultado é 28287 (0110111001111111 binário).

	Decimal	Binário
	28277	01101110 01110101 (conteúdo de %M0000)
OR	15	00000000 00001111 (valor de %KM+00015)
	28287	01101110 01111111 (resultado em %M0001)

### Sintaxe:

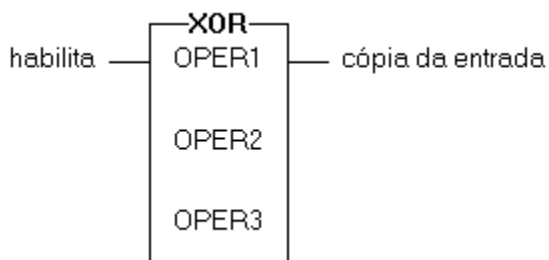
OPER1	OPER2	OPER3
%KM %M	%KM %M	%M

OPER1	OPER2	OPER3
%KD %D	%KD %D	%D

OPER1	OPER2	OPER3
%I %KI	%I %KI	%I

Tabela 3-23 Sintaxes da Instrução OR

## XOR - Ou Exclusivo entre Operandos



OPER1 - primeiro operando

OPER2 - segundo operando

OPER3 - resultado

### Descrição:

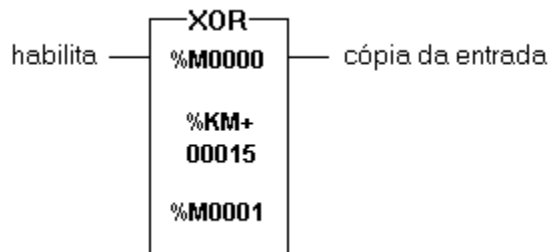
Esta instrução realiza a operação "ou exclusivo" binário entre os dois primeiros operandos, armazenando o resultado no terceiro.

A operação é realizada ponto a ponto entre os operandos. A tabela a seguir mostra as combinações da operação "ou exclusivo" ponto a ponto possíveis.

ponto OPER1	ponto OPER2	ponto OPER3 (resultado)
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 3-24 Operações Ponto a Ponto (XOR)

### Exemplo:



Neste exemplo deseja-se inverter os pontos contidos no nibble menos significativo de %M0000, preservando o resto do operando. Se %M0000 contém 1612 (0000011001001100 binário), o resultado é 1603 (0000011001000011 binário)

Decimal	Binário
1612	00000110 01001100 (conteúdo de %M0000)
XOR 15	XOR 00000000 00001111 (valor de %KM+00015)
1603	00000110 01000011 (resultado em %M0001)

Portanto, o valor 1603 decimal é armazenado em M001.

### Sintaxe:

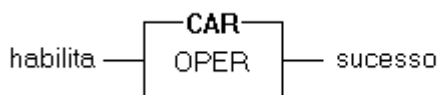
OPER1	OPER2	OPER3
%KM %M	%KM %M	%M

OPER1	OPER2	OPER3
%KD %D	%KD %D	%D

OPER1	OPER2	OPER3
%I %KI	%I %KI	%I

Tabela 3-25 Sintaxes da Instrução XOR

## CAR - Carrega Operandos



OPER - operando a ser carregado

### Descrição:

A instrução carrega operando realiza a carga do valor do operando especificado em registrador especial interno ao CP, para subsequente uso das instruções de comparação (maior, menor, igual). O operando permanece carregado até a próxima instrução de carga, podendo ser utilizado por várias lógicas, inclusive em ciclos de varredura subsequentes.

A saída sucesso é acionada se a carga for realizada. Se algum acesso indireto a operando não for possível (índice inválido), a saída sucesso não é acionada.

Ver considerações e exemplos apresentados na seção seguinte, Instruções de Comparação de Operandos.

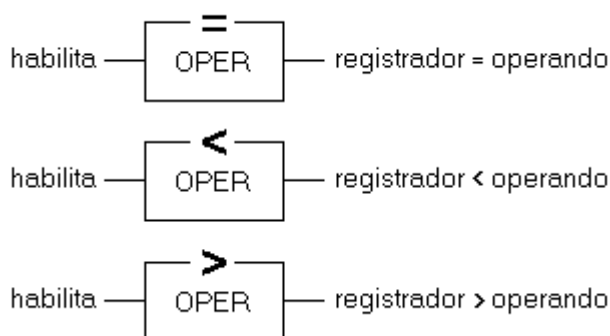
Não pode ser feita a comparação entre operandos decimais e operandos flutuante.

### Sintaxe:

OPER1
%E
%S
%A
%M
%D
%F
%I
%KM
%KD
%KF
%KI
%M*E
%M*S
%M*A
%M*M
%M*D
%M*F
%M*I

Tabela 3-26 Sintaxe da Instrução CAR

## Instruções de Comparação de Operandos - Igual, Maior e Menor



OPER - operando a ser comparado

### Descrição:

As instruções maior, menor e igual realizam comparações do operando especificado com o valor previamente carregado no registrador interno com a instrução CAR (Carrega Operando), fornecendo o resultado da comparação em suas saídas. Caso algum acesso indireto seja inválido, a saída é desacionada.

Por exemplo, a instrução maior energiza a sua saída se o valor do operando presente na última instrução CAR ativa for maior que o valor do seu operando. As instruções igual e menor operam de forma idêntica, mudando apenas o tipo de comparação realizada.

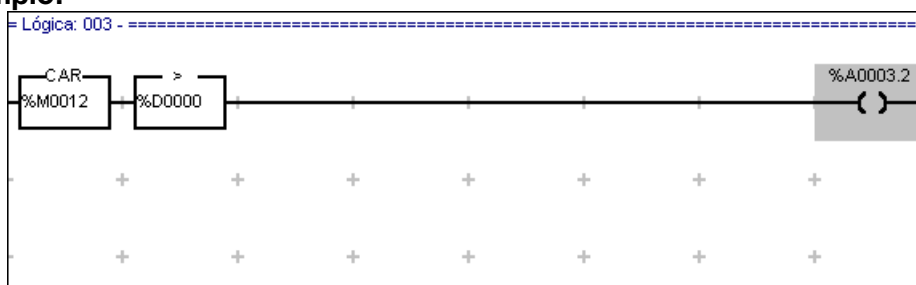
Se os operandos a serem comparados são do mesmo tipo, são comparados conforme o seu formato de armazenamento (considerando o seu sinal). Se não são do mesmo tipo, são comparados ponto a ponto (como valores binários sem sinal).

Não pode ser feita a comparação entre operandos decimais e operandos flutuante.

### ATENÇÃO:

Sugere-se que sempre sejam comparados operandos de tipos iguais, para evitar má interpretação nos resultados quando os operandos possuírem valores negativos. Ver o exemplo a seguir.

### Exemplo:



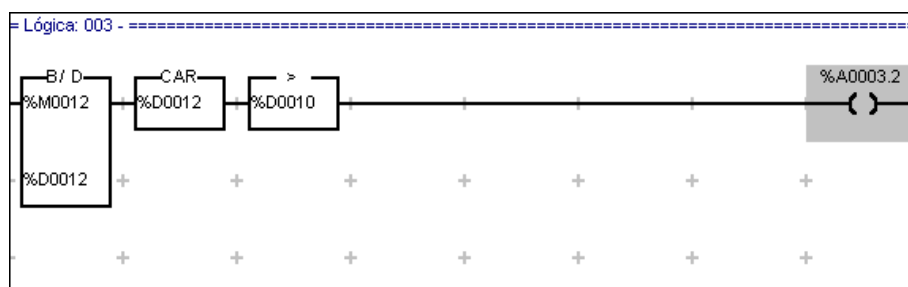
**Figura 3-6 Exemplo das Instruções de Comparação**

Como os tipos dos operandos são diferentes (%M e %D), a comparação é realizada ponto a ponto, sem considerar os sinais aritméticos. Devido a este fato, se %M0012 possuir valor -45 e %D0010 possuir valor +21, o operando %A0003.2 será energizado, como se o valor de %M0012 fosse maior que %D0010, o que não ocorre na realidade.

<b>%M0012</b>	= -45					1111	1111	1101	0011
<b>%D0000</b>	= +21	0000	0000	0000	0000	0000	0000	0010	0001

Para considerar os sinais na comparação do exemplo, deve-se converter o valor do operando memória para um decimal, utilizando este último na instrução CAR, como mostrado na lógica a seguir:

O valor 1111 1111 1101 0011 (%M0012) é maior que 10 0001 (%D0010) na comparação ponto a ponto, mesmo representando um valor negativo.

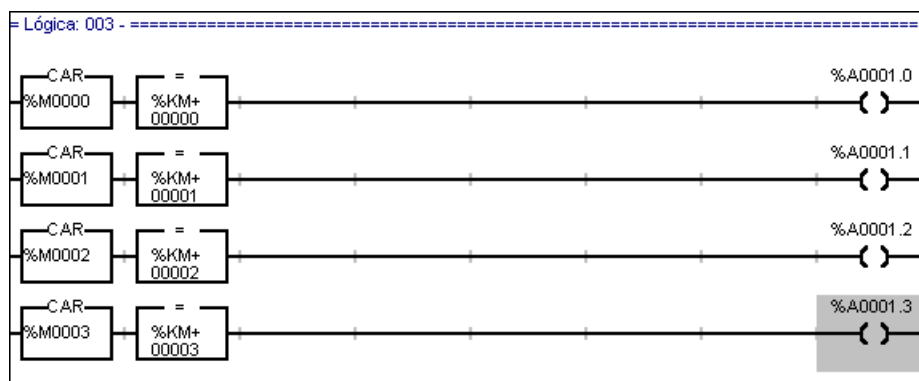


**Figura 3-7 Exemplo das Instruções de Comparação**

#### ATENÇÃO:

Devido à ordem de processamento das instruções na lógica, deve-se cuidar o posicionamento das instruções de comparação para evitar erros na interpretação no seu funcionamento. Ver seção Lógicas neste mesmo capítulo e o exemplo a seguir.

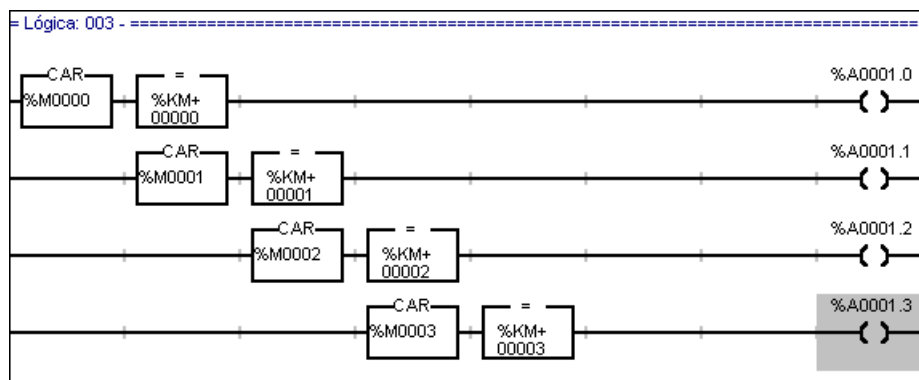
#### Exemplo:



**Figura 3-8 Utilização Incorreta da Instrução CAR**

Na lógica apresentada, deseja-se comparar os valores dos operandos %M0000, %M0001, %M0002 e %M0003 com as constantes %KM00000, %KM00001, %KM00002 e %KM00003, respectivamente. Entretanto, o funcionamento ocorre de forma diversa do que o aspecto visual sugere. Como o processamento da lógica ocorre em colunas, no final da execução da coluna 0 estará carregado o valor de %M0003 para as comparações na coluna 1. Na realidade, somente o valor do operando %M0003 será comparado com as constantes presentes na coluna 1.

Para o funcionamento desejado, a lógica deve ser programada da seguinte forma:



**Figura 3-9 Utilização Correta da Instrução CAR**



**ATENÇÃO:**

Para evitar interpretações errôneas no funcionamento das instruções de comparação, sugere-se o uso de apenas uma instrução CAR por coluna da lógica.

**Sintaxe:**

OPER1
%E
%S
%A
%M
%D
%F
%I
%KM
%KD
%KF
%KI
%M*E
%M*S
%M*A
%M*M
%M*D
%M*F
%M*I

Tabela 3-27 Sintaxe das Instruções Maior, Igual e Menor

## Instruções do Grupo Contadores

As instruções contadoras são utilizadas para realizar contagens de eventos ou de tempo no programa aplicativo.





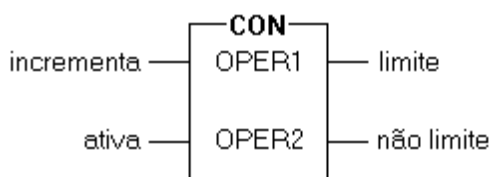
Nome	Descrição do Nome	Seqüência de Edição	Barra de Ferramentas
CON	contador simples	ALT, C, N	
COB	contador bidirecional	ALT, C, B	
TEE	temporizador na energização	ALT, C, T	
TED	temporizador na desenergização	ALT, C, D	

Tabela 3-28 Instruções do Grupo Contadores

## CON - Contador Simples



OPER1 - contador

OPER2 - limite de contagem

### Descrição:

Esta instrução realiza contagens simples, com o incremento de uma unidade em cada acionamento.

A instrução contador simples possui dois operandos. O primeiro, sempre do tipo %M, especifica a memória que contabiliza os eventos. O segundo estabelece o valor limite de contagem para energização da saída da célula superior e pode ser do tipo %KM, %M ou operando %M referenciado indiretamente.

Se a entrada **ativa** esta desenergizada, a memória em OPER1 é zerada, a saída **não limite** energizada e a saída **limite** desenergizada.

Quando a entrada **ativa** está energizada, cada transição de ligação na entrada **incrementa** aumenta o valor do operando contador (OPER1) de uma unidade.

Se o valor do primeiro operando igualar-se ao do segundo operando, a saída **limite** é energizada. A variável contadora não é incrementada com novas transições na entrada **incrementa**, permanecendo com o valor limite. Se for menor, a saída **limite** é desenergizada. O estado lógico da saída **não limite** é exatamente o oposto da saída **limite**, mesmo estando a instrução desativada.

Em caso de acesso indireto inválido para o segundo operando da instrução, a saída não limite é energizada.

### ATENÇÃO:

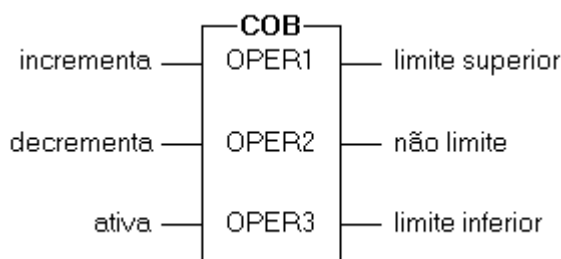
Com a entrada **ativa** desativada, a saída **não limite** permanece sempre energizada, mesmo quando a instrução estiver em um trecho comandado pela instrução RM (relé mestre). Deve-se ter cuidado para não realizar acionamentos indesejáveis na lógica devido a este fato.

### Sintaxe:

OPER1	OPER2
%M	%KM %M %M*M

Tabela 3-29 Sintaxe da Instrução CON

## COB - Contador Bidirecional



OPER1 - contador

OPER2 - passo de contagem

OPER3 - limite de contagem

### Descrição:

Esta instrução realiza contagens com o valor de incremento ou decremento definido por um operando. A instrução contador bidirecional permite contagens em ambos os sentidos, isto é, incrementa ou decrementa o conteúdo de um operando do tipo memória.

O primeiro operando contém a memória acumuladora do valor contado, enquanto que o segundo especifica o valor do incremento ou decremento desejado. O terceiro operando contém o valor limite da contagem.

A contagem ocorre sempre que a entrada **ativa** está energizada e as entradas **incrementa** ou **decrementa** sofrerem uma transição de desligadas para ligadas. Se ambas as entradas sofrem a transição no mesmo ciclo de varredura do programa, não há incremento nem decremento no valor da memória declarada em OPER1.

Caso o valor do incremento seja negativo, a entrada **incrementa** provoca decrementos e a entrada **decremento** provoca incrementos no valor da contagem.

Se o valor do primeiro operando tornar-se maior ou igual ao do terceiro operando, a saída **limite superior** é energizada, não havendo incremento.

Se o valor do primeiro operando tornar-se igual ou inferior a zero, a saída **limite inferior** é acionada, sendo armazenado zero no primeiro operando.

Se o valor do primeiro operando está entre zero e o limite, a saída **não limite** é acionada. Se a entrada **ativa** não está energizada, a saída **limite inferior** é energizada e o primeiro operando é zerado.

Em caso de acesso indireto inválido para qualquer um dos operandos da instrução, a saída limite inferior é energizada.

### ATENÇÃO:

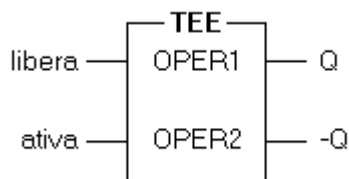
Com a entrada **ativa** desativada, a saída **limite inferior** permanece sempre energizada, mesmo quando a instrução estiver em um trecho comandado pela instrução RM (relé mestre). Deve-se ter cuidado para não realizar acionamentos indesejáveis na lógica devido a este fato.

### Sintaxe:

OPER1	OPER2	OPER3
%M	%M	%M
%M*M	%M*M	%M*M
	%KM	%KM

Tabela 3-30 Sintaxe da Instrução COB

## TEE - Temporizador na Energização



OPER1 - acumulador de tempo

OPER2 - limite de tempo (décimos de segundos)

### Descrição:

Esta instrução realiza contagens de tempo com a energização das suas entradas de acionamento.

A instrução TEE possui dois operandos. O primeiro (OPER1) especifica a memória acumuladora da contagem de tempo. O segundo operando (OPER2) indica o tempo máximo a ser acumulado. A contagem de tempo é realizada em décimos de segundos, ou seja, cada unidade incrementada em OPER1 corresponde a 0,1 segundo.

Enquanto as entradas **libera** e **ativa** estiverem simultaneamente energizadas, o operando OPER1 é incrementado a cada décimo de segundo. Quando OPER1 for maior ou igual a OPER2, a saída **Q** é energizada e **-Q** desenergizada, permanecendo OPER1 com o mesmo valor de OPER2.

Desacionando-se a entrada **libera**, há a interrupção na contagem do tempo, permanecendo OPER1 com o mesmo valor. Desacionando-se a entrada **ativa**, o valor em OPER1 é zerado.

Se OPER2 for negativo ou o acesso indireto for inválido, OPER1 é zerado e a saída **-Q** é energizada.

O estado lógico da saída **Q** é o oposto da saída **-Q**, mesmo estando a instrução desativada.

### ATENÇÃO:

Com a entrada **ativa** desativada, a saída **-Q** permanece sempre energizada, mesmo quando a instrução estiver em um trecho comandado pela instrução RM (relé mestre). Deve-se ter cuidado para não realizar acionamentos indesejáveis na lógica devido a este fato.

### Diagrama de Tempos:

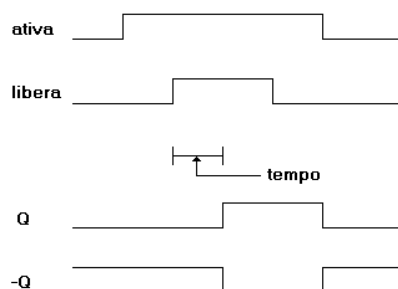


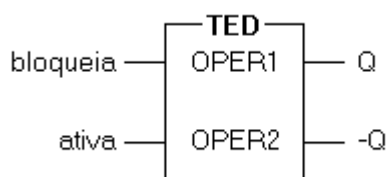
Figura 3-10 Diagrama de Tempos da Instrução TEE

### Sintaxe:

OPER1	OPER2
%M	%M %M*M %KM

Tabela 3-31 Sintaxe da Instrução TEE

## TED - Temporizador na Desenergização



OPER1 - acumulador de tempo

OPER2 - limite de tempo (décimos de segundo)

### Descrição:

Esta instrução realiza contagens de tempo com a desenergização da sua entrada de acionamento.

A instrução TED possui dois operandos. O primeiro (OPER1) especifica a memória acumuladora da contagem de tempo. O segundo operando (OPER2) indica o tempo máximo a ser acumulado. A contagem de tempo é realizada em décimos de segundos, ou seja, cada unidade incrementada em OPER1 corresponde a 0,1 segundo.

Enquanto a entrada **ativa** estiver energizada e a entrada **bloqueia** desenergizada, o operando OPER1 é incrementado a cada décimo de segundo. Quando OPER1 for maior ou igual a OPER2, a saída Q é desenergizada e -Q energizada, permanecendo OPER1 com o mesmo valor de OPER2.

A saída Q fica energizada sempre que a entrada **ativa** estiver energizada e OPER1 for menor do que OPER2.

Acionando-se a entrada **bloqueia**, há a interrupção na contagem do tempo, enquanto que desacionando a entrada **ativa**, o tempo do acumulador é zerado e a saída Q é desacionada.

Se OPER2 for negativo ou o acesso indireto for inválido, OPER1 é zerado e a saída Q é energizada.

O estado lógico da saída -Q é o oposto da saída Q, mesmo estando a instrução desativada.

### ATENÇÃO:

Com a entrada **ativa** desativada, a saída -Q permanece sempre energizada, mesmo quando a instrução estiver em um trecho comandado pela instrução RM (relé mestre). Deve-se ter cuidado para não realizar acionamentos indesejáveis na lógica devido a este fato.

### Diagrama de Tempos:

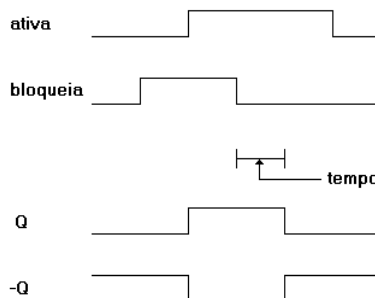


Figura 3-11 Diagrama de Tempos da Instrução TED

### Sintaxe:

OPER1	OPER2
%M	%M %M*M %KM

Tabela 3-32 Sintaxe da Instrução TED

## Instruções do Grupo Conversores

Esse grupo possui instruções que permitem a conversão entre os formatos de armazenamento dos valores utilizados nos operandos do programa aplicativo e acessos a módulos analógicos no barramento de entrada e saída.



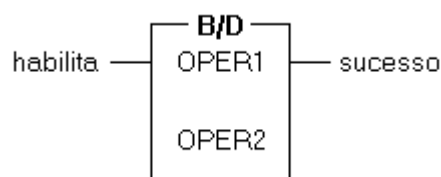
Nome	Descrição do Nome	Seqüência de Edição	Barra de Ferramentas
<b>BIN/DEC</b>	conversão binário-decimal	ALT, V, B	
<b>DEC/BIN</b>	conversão decimal-binário	ALT, V, D	

Tabela 3-33 Instruções do Grupo Conversores

## B/D - Conversão Binário-Decimal



OPER1 - origem

OPER2 - destino

### Descrição:

Esta instrução converte valores armazenados em formato binário, contidos em operandos memória (%M), para o formato decimal (BCD), armazenando-os em operandos decimais (%D).

O valor binário contido no primeiro operando (OPER1) é convertido para valor decimal e armazenado no segundo operando (OPER2). A saída **sucesso** é acionada se a conversão for corretamente realizada. Se algum acesso indireto inválido a operando ocorrer, a saída **sucesso** não é energizada.

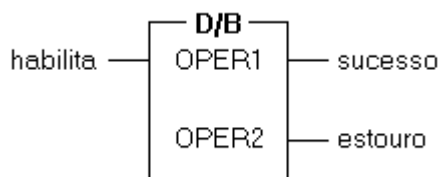
### Sintaxe:

OPER1	OPER2
%M	%D
%M*M	%M*D

Tabela 3-34 Sintaxe da Instrução B/D



## D/B - Conversão Decimal-Binário



OPER1 - origem

OPER2 - destino

### Descrição:

Esta instrução converte valores armazenados em formato decimal, contidos em operandos decimais (%D), para o formato binário, armazenando-os em operandos memórias (%M).

O valor decimal contido no primeiro operando (OPER1) é convertido para valor binário e armazenado no segundo operando (OPER2). A saída **sucesso** é acionada se a conversão for corretamente realizada. Se algum acesso indireto inválido a operando ocorrer, a saída **sucesso** não é energizada.

Caso o valor convertido resultar em um valor maior do que os máximos armazenáveis em operandos %M, a saída **sucesso** não é energizada, sendo armazenado o valor limite no operando destino. Neste caso, a saída **estouro** é energizada.

### Sintaxe:

OPER1	OPER2
%D	%M
%M*D	%M*M

Tabela 3-35 Sintaxe da Instrução D/B

## Instruções do Grupo Geral

As instruções do grupo gerais permitem teste e acionamentos de pontos indiretamente, implementações de máquinas de estado e chamadas de procedimentos e funções.









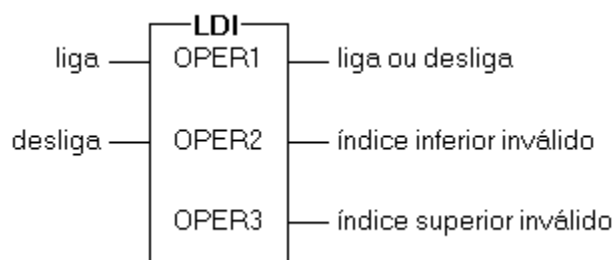
Nome	Descrição do Nome	Seqüência de Edição	Barra de Ferramentas
<b>LDI</b>	liga ou desliga pontos indexados	ALT, G, L	
<b>TEI</b>	teste de estado de pontos indexados	ALT, G, T	
<b>SEQ</b>	Seqüenciador	ALT, G, S	
<b>CHP</b>	chama módulo procedimento	ALT, G, P	
<b>CHF</b>	chama módulo função	ALT, G, F	
<b>ECH</b>	escrita de operandos em outro CP para Ethernet	ALT, G, E	
<b>LTH</b>	leitura de operandos de outro CP para Ethernet	ALT, G, T	
<b>LAH</b>	libera atualização de imagens para Ethernet	ALT, G, A	

Tabela 3-40 Instruções do Grupo Geral

## LDI - Liga/Desliga Indexado



OPER1 - endereço do ponto a ser ligado ou desligado

OPER2 - endereço limite inferior

OPER3 - endereço limite superior

### Descrição:

Esta instrução é utilizada para ligar ou desligar pontos indexados por uma memória, delimitados por operandos de limite inferior e superior.

O primeiro operando especifica a memória cujo conteúdo referencia o operando auxiliar, entrada ou saída a ser ligado ou desligado. Deve ser declarado como operando de acesso indireto a operando %E ou %A (%MXXXX\*E ou %MXXXX\*A). Mesmo quando a instrução for utilizada para ligar ou desligar pontos de saída (%S), a representação deste operando será como acesso indireto a entrada (%MXXXX\*E).

O segundo operando especifica o endereço do primeiro relé de saída ou auxiliar válido na instrução. Deve ser especificado com subdivisão de ponto (%SXXXX.X ou %AXXXX.X).

O terceiro operando especifica o endereço do último relé de saída ou auxiliar válido na instrução. Deve ser especificado com subdivisão de ponto (%EXXXX.X, %SXXXX.X ou %AXXXX.X).

Se as entradas **liga** ou **desliga** forem acionadas, o ponto especificado pelo valor contido no operando memória (OPER1) é ligado ou desligado se estiver dentro da área de endereços limitada por OPER2 e OPER3. Por exemplo, se estes operandos correspondem a %S0003.3 e %S0004.5, respectivamente, esta instrução só atua para os elementos de %S0003.3 a %S0003.7 e de %S0004.0 a %S0004.5.

Se o relé ou auxiliar apontado pela memória índice estiver fora dos limites definidos pelos parâmetros da segunda e terceira células, a saída **índice superior inválido** ou **índice inferior inválido** é ligada. A saída da primeira célula é acionada se qualquer uma das entradas **liga** ou **desliga** é energizada e o acesso for corretamente realizado.

Caso as entradas permaneçam desacionadas, todas as saídas da instrução permanecem desenergizadas.

Se ambas as entradas forem energizadas simultaneamente, nenhuma operação é realizada, e todas as saídas da instrução são desenergizadas.

Em OPER1 deve ser carregado um valor que especifique o ponto desejado para ligar ou desligar, de acordo com a seguinte fórmula:

$$\text{VALOR OPER1} = (\text{OCTETO} * 8) + \text{PONTO}$$

### Exemplo:

Por exemplo, se S0010.5 é o ponto que se deseja ligar indiretamente, então:

$$\text{OCTETO} = 10$$

$$\text{PONTO} = 5$$

$$\text{VALOR OPER1} = (10 * 8) + 5 = 85$$

O valor a ser carregado em OPER1 é 85.

**ATENÇÃO:**

Esta instrução permite ligar ou desligar indiretamente pontos de operandos %E, sobrepondo o valor da varredura dos módulos de entrada após a sua execução.

**Sintaxe:**

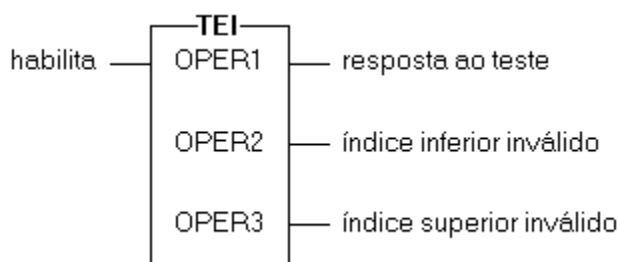
OPER1	OPER2	OPER3
% <b>M</b> *E	%E	%E

OPER1	OPER2	OPER3
% <b>M</b> *S	%S	%S

OPER1	OPER2	OPER3
% <b>M</b> *A	%A	%A

Tabela 3-41 Sintaxes da Instrução LDI

## TEI - Teste de Estado Indexado



OPER1 - endereço do ponto a ser testado

OPER2 - endereço limite inferior

OPER3 - endereço limite superior

### Descrição:

Esta instrução é utilizada para testar o estado de pontos indexados por uma memória, delimitados por operandos de limite inferior e superior.

O primeiro operando especifica a memória cujo conteúdo referencia o operando auxiliar ou relé de saída a ser testado. Deve ser declarado como operando de acesso indireto a operando %E ou %A (%MXXX\*E ou %MXXX\*A). Mesmo quando a instrução for utilizada para testar pontos de saída (%S), a representação deste operando será como acesso indireto a entrada (%MXXX\*E).

O segundo operando especifica o endereço do primeiro relé de saída ou auxiliar válido na instrução. Deve ser especificado com subdivisão de ponto (%EXXX.X, %SXXX.X ou %AXXX.X).

O terceiro operando especifica o endereço do último relé de saída ou auxiliar válido na instrução. Deve ser especificado com subdivisão de ponto (%EXXX.X, %SXXX.X ou %AXXX.X).

Se a entrada **habilita** estiver energizada, o estado do relé ou auxiliar especificado pelo valor contido na memória índice (OPER1) é examinado. Conforme esteja em 1 ou 0, a saída **resposta** é ligada ou não.

O ponto indexado pela memória é testado se estiver dentro da área de endereços limitada por OPER2 e OPER3. Por exemplo, se estes operandos correspondem a %S0003.3 e %S0004.5, respectivamente, esta instrução só atua para os elementos de %S0003.3 a %S0003.7 e de %S0004.0 a %S0004.5.

Se o relé ou auxiliar apontado pela memória índice estiver fora dos limites definidos pelos parâmetros da segunda e terceira células, a saída **índice superior inválido** ou **índice inferior inválido** é ligada e a saída da primeira célula desligada. Esta verificação é feita somente no momento em que a entrada **habilita** é energizada.

O cálculo do valor a ser armazenado no primeiro operando, para referência do ponto desejado, é o mesmo especificado na instrução **LDI**.

### Sintaxe:

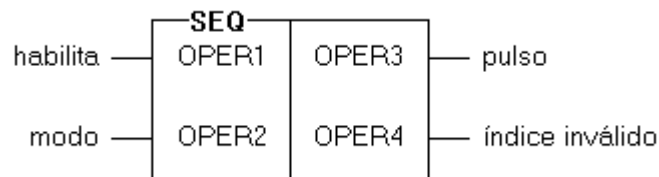
OPER1	OPER2	OPER3
%M*E	%E	%E

OPER1	OPER2	OPER3
%M*S	%S	%S

OPER1	OPER2	OPER3
% <b>M</b> * <b>A</b>	% <b>A</b>	% <b>A</b>

Tabela 3-42 Sintaxes da Instrução TEI

## SEQ - Seqüenciador



OPER1 - tabela de condições ou primeira tabela de estados

OPER2 - índice da(s) tabela(s) (estado atual)

OPER3 - operando base da primeira série de condições

OPER4 - operando base da segunda série de condições

### Descrição:

Esta instrução permite a programação de seqüenciamentos complexos com condições de evolução específicas para cada estado. Sua forma de programação é semelhante a "máquinas de estado".

A instrução pode ser executada em dois modos: o modo 1000 e o modo 3000. Quando a entrada **modo** está desenergizada, a instrução é executada no modo 1000, e quando ela está energizada, a instrução é executada no modo 3000. No modo 3000 seqüenciamentos mais complexos podem ser programados.

### Modo 1000:

Neste modo ocorre uma seqüência de evolução fixa dos estados. A evolução sempre ocorre do estado atual para o seguinte, e do último para o primeiro.

O primeiro operando especifica uma tabela onde cada posição contém o endereço de um ponto de operando auxiliar que é testado como condição de evolução para o próximo estado.

O segundo operando especifica uma memória que armazena o estado atual e serve de índice para a tabela especificada no primeiro operando.

O terceiro operando é irrelevante, porém deve ser especificado um operando do tipo memória ou auxiliar nesta célula, pois o MasterTool realiza a consistência conforme o modo 3000.

O quarto operando é irrelevante, porém deve ser especificado um operando do tipo memória ou auxiliar nesta célula, pois o MasterTool realiza a consistência conforme o modo 3000.

Quando a entrada **habilita** está desenergizada, as saídas **pulso** e **índice inválido** ficam desenergizadas, independente de qualquer outra condição. Quando a entrada **habilita** estiver energizada, a saída de **pulso** fica normalmente energizada, e a saída de **índice inválido** fica normalmente desenergizada.

Além disso, quando a entrada **habilita** está energizada, a posição da tabela (OPER1) indexada pelo estado atual (OPER2) é acessada e o ponto de operando auxiliar referenciado nesta posição da tabela é examinado. Se este ponto estiver energizado, o conteúdo de OPER2 é incrementado (ou zerado, se estiver apontando para a última posição da tabela OPER1) e na saída **pulso** ocorre um pulso de desenergização com duração de um ciclo de programa. Se o ponto examinado estiver desenergizado nada ocorre e o valor da memória em OPER2 permanece inalterado.

A saída **índice inválido** é ativada se a memória OPER2 (estado atual) contiver um valor que indexa uma posição não existente na tabela especificada em OPER1. Isto pode ocorrer modificando-se a memória OPER2 em um ponto do programa aplicativo fora da instrução SEQ (na inicialização de OPER2, por exemplo). Deve-se ter o cuidado de definir e inicializar a tabela especificada em OPER1 com valores legais.

Na tabela especificada em OPER1 devem ser carregados valores em formato decimal que especifiquem pontos de operandos auxiliares que devem ser testados como condições de evolução. O cálculo destes valores é especificado pela equação:

$VALOR = (\text{endereço do operando} * 8) + \text{endereço da subdivisão}$

**Exemplo:**

Se %A0030.2 é o ponto que se deseja usar como condição de evolução a partir do estado 4, então:

endereço do operando = 30

Endereço da subdivisão = 2

$VALOR = (30 * 8) + 2 = 242$

O valor a ser carregado na posição 4 da tabela OPER1 deve ser 242 para que o ponto %A0030.2 cause a evolução para o próximo estado, que é o estado 5 (ou o estado 0, se a tabela tiver 5 posições).

**Modo 3000:**

Neste modo é possível definir a sequência de evolução e escolher um entre dois caminhos a partir do estado atual. Portanto, 2 graus de liberdade a mais são oferecidos em relação ao modo 1000, permitindo-se implementar máquinas de estado bem mais complexas.

Existe, entretanto, menos liberdade para escolher as condições de evolução em relação ao modo 1000, além de ser necessário o uso de mais memória (tabelas) no modo 3000.

O primeiro operando especifica a primeira de duas tabelas subseqüentes que são utilizadas pela instrução. As duas tabelas devem ter o mesmo tamanho. Cada posição da primeira tabela contém o próximo estado caso a condição associada ao operando 3 esteja energizada. Cada posição da segunda tabela contém o próximo estado caso a condição associada ao operando 4 esteja energizada.

O segundo operando especifica uma memória que indica qual o estado atual e serve de índice para as tabelas especificadas no primeiro operando.

O terceiro operando especifica um operando que serve de base para determinar a condição de evolução a partir do estado OPER2 para o estado indexado por OPER2 na primeira tabela.

O quarto operando especifica um operando que serve de base para determinar a condição de evolução a partir do estado OPER2 para o estado indexado por OPER2 na segunda tabela.

Quando a entrada **habilita** está desenergizada, as saídas **pulso** e **índice inválido** ficam desenergizadas, independente de qualquer outra condição. Quando a entrada **habilita** está energizada, a saída de **pulso** fica normalmente energizada, e a saída de **índice inválido** fica normalmente desenergizada.

Além disso, quando a entrada **habilita** está energizada, a instrução busca o valor da memória OPER2 (estado atual) e testa a respectiva condição de evolução com base em OPER3. Se esta condição estiver energizada, o operando OPER2 é carregado com um novo estado, indexado pelo próprio operando OPER2 na primeira tabela especificada por OPER1. Caso a condição de evolução associada a OPER2 e com base em OPER3 estiver desenergizada, testa-se a condição de evolução associada a OPER2 e com base em OPER4. Se esta última condição estiver energizada, o operando OPER2 é carregado com um novo estado, indexado pelo próprio operando OPER2 na segunda tabela especificada por OPER1. Se pelo menos uma das 2 condições acima estiver energizada, uma transição de estado ocorrerá, e um pulso de desenergização com duração de um ciclo de programa aplicativo ocorrerá na saída **pulso** da instrução. Se nenhuma das 2 condições estiver energizada, nada acontece, e o valor da memória OPER2 (estado atual) permanece inalterado, bem como a saída **pulso** continua energizada.

A saída **índice inválido** é ativada se a memória OPER2 contiver um valor que indexa uma posição não existente nas tabelas especificadas em OPER1. Isto pode ocorrer modificando-se a memória OPER2 em um ponto do programa aplicativo fora da instrução SEQ (na inicialização de OPER2, por exemplo) ou dentro da própria instrução SEQ, caso algumas das posições das tabelas especificadas em OPER1 contenham valores inválidos para serem o próximo estado. Deve-se ter o cuidado de definir as 2 tabelas especificadas por OPER1 com o mesmo tamanho, e deve-se inicializá-las com valores legais (exemplo: se as tabelas tiverem 10 posições, somente valores entre 0 e 9 devem ser carregados em posições desta tabela, pois somente estes podem ser estados legais).



As condições de evolução associadas ao estado atual (OPER2) são determinadas com base em OPER3 (próximo estado é carregado a partir da primeira tabela) ou com base em OPER4 (próximo estado é carregado a partir da segunda tabela). Sabendo-se que os operandos OPER3 e OPER4 são do tipo memória (16 bits) ou do tipo auxiliar (8 bits), suponha-se o seguinte:

ESTADO = conteúdo do operando OPER2 (estado atual)

END3 = endereço de OPER3

END4 = endereço de OPER4

END1 = endereço do ponto a ser testado, com base em OPER3

SUB1 = subdivisão do ponto a ser testado, com base em OPER3

END2 = endereço do ponto a ser testado, com base em OPER4

SUB2 = subdivisão do ponto a ser testado, com base em OPER4

Os pontos testados como condição de evolução associada a cada tabela serão:

M<END1>.<SUB1> ou A<END1>.<SUB1> (primeira tabela)

e M<END2>.<SUB2> ou A<END2>.<SUB2> (segunda tabela)

onde:

END1 = END3 + ESTADO / 16 (se operando %M)

END1 = END3 + ESTADO / 8 (se operando %A)

SUB1 = RESTO (ESTADO / 16) (se operando %M)

SUB1 = RESTO (ESTADO / 8) (se operando %A)

END2 = END4 + ESTADO / 16 (se operando %M)

END2 = END4 + ESTADO / 8 (se operando %A)

SUB2 = RESTO (ESTADO / 16) (se operando %M)

SUB2 = RESTO (ESTADO / 8) (se operando %A)

### Exemplo:

Sejam:

OPER1 = %TM000

OPER2 = %M0010

OPER3 = %M0100

OPER4 = %A0020

Onde:

%TM000	Posição	Valor
	000	00001
	001	00002
	002	00004
	003	00001
	004	00000

%TM001	Posição	Valor
	000	00001
	001	00003
	002	00001
	003	00004
	004	00000

**%M0010** = 00001

%M0100	XXXXX
%M0101	XXXXX
%M0102	XXXXX
%M...	...

%A0020	XXXXX
%A0021	XXXXX
%A0022	XXXXX
%A...	...

Então as condições de evolução a partir do estado 1 serão:

Para a primeira tabela:

- $100 + 1/16 = 100$
- $\text{resto}(1/16) = 1$
- ponto a ser testado = %M0100.1

Para a segunda tabela:

- $20 + 1/8 = 20$
- $\text{resto}(1/8) = 1$
- ponto a ser testado = %A0020.1

Baseado nas condições de %M0100.1 e %A0020.1 teremos, a partir de uma das tabelas, o novo estado do operando %M0010:

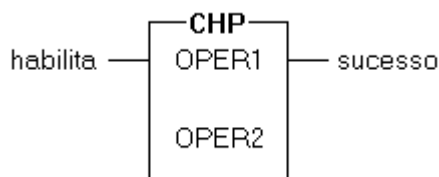
%M0100.1	%A0020.1	%M0010	Observação
0	0	00001	não há mudança de estado
0	1	00003	mudança de estado conforme %TM001
1	0	00002	mudança de estado conforme %TM000
1	1	00002	mudança de estado conforme %TM000 (OPER3 tem prioridade sobre OPER4)

**Sintaxe:**

OPER1	OPER2	OPER3	OPER4
%TM	%M	%M	%M
%M*TM	%M*M	%A	%A

Tabela 3-43 Sintaxe da Instrução SEQ

## CHP - Chama Módulo Procedimento



OPER1 - nome do módulo a chamar

OPER2 - número do módulo a chamar

### Descrição:

Esta instrução realiza o desvio do processamento do módulo corrente para módulo Procedimento especificado nos seus operandos, se o mesmo estiver presente no CP. Ao final da execução do módulo chamado, o processamento retorna para a instrução seguinte à CHP. Não há passagem de parâmetros para o módulo chamado.

O primeiro operando (OPER1) é documental e especifica o nome do módulo a ser chamado. O segundo operando (OPER2) especifica o número deste módulo, sendo implícito o fato do módulo chamado ser do tipo procedimento.

Caso o módulo chamado não exista, a saída **sucesso** é desenergizada e a execução continua normalmente após a instrução. O nome do módulo não é considerado pelo CP para a chamada mas apenas o seu número. Caso exista o módulo P com o mesmo número do módulo chamado, porém com nome diferente, este módulo mesmo assim é executado.

Ver seção **Utilização dos Módulos P e F** no capítulo 2 deste manual.

### Exemplo:

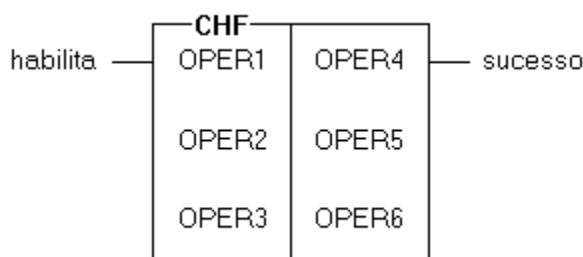


### Sintaxe:

OPER1	OPER2
NOME	NÚMERO

Tabela 3-44 Sintaxe da Instrução CHP

## CHF - Chama Módulo Função



OPER1 - nome do módulo a chamar

OPER2 - número de parâmetros a enviar

OPER3 - número de parâmetros a retornar

OPER4 - número do módulo a chamar

OPER5 - lista dos parâmetros a enviar

OPER6 - lista dos parâmetros a retornar

### Descrição:

A instrução chama módulo função realiza o desvio do processamento do módulo corrente para o módulo especificado na mesma, se este estiver presente no CP. Ao final da execução do módulo chamado, o processamento retorna para a instrução seguinte à CHF.

Devem ser declarados o nome e o número do módulo como operandos OPER1 e OPER4, respectivamente, sendo implícito o fato do módulo chamado ser do tipo função. Caso o módulo chamado não exista no controlador, a execução continua normalmente após a instrução de chamada, com a saída sucesso da mesma desligada. O nome do módulo não é levado em consideração pelo CP, estando no programa aplicativo apenas como referência documental, sendo considerados para a chamada apenas o seu tipo e número. Caso exista o módulo F com o mesmo número do módulo chamado mas com nome diferente, este módulo é executado.

É possível a passagem de valores de operandos (parâmetros) para o módulo chamado e no sentido inverso, após a sua execução. Na quinta célula da instrução (OPER5) é especificada uma lista de operandos a serem enviados para o módulo chamado. Antes da execução do módulo, os valores destes operandos são copiados para os operandos especificados na lista de parâmetros de entrada do módulo F, declarados na opção **Parâmetros** do MasterTool quando o mesmo foi programado.

Após a execução do módulo F chamado, os valores dos operandos declarados na sua lista de parâmetros de saída (opção **Parâmetros** do MasterTool na sua programação) são copiados para os operandos declarados na lista de operandos a retornar da instrução CHF (OPER6). Finalizada a cópia de retorno, o processamento continua na instrução seguinte à de chamada.

### ATENÇÃO:

O MasterTool não realiza nenhuma consistência em relação aos operandos programados como parâmetros, tanto na instrução CHF como no módulo F.

A lista de operandos a serem enviados para o módulo F deve conter o mesmo número de operandos com o mesmo tipo dos declarados como parâmetros de entrada do módulo, para que a cópia dos seus valores seja realizada corretamente. A cópia dos operandos é realizada na mesma ordem em que os mesmos estão dispostos nas listas. Caso uma das duas listas possua menos operandos em relação à outra, os valores dos operandos excedentes não serão copiados. Se os operandos possuírem tipos diferentes, a cópia dos valores será realizada com as mesmas regras usadas na instrução MOV (movimentação simples de operandos). Este princípio é válido também para as listas de parâmetros de retorno do módulo F.

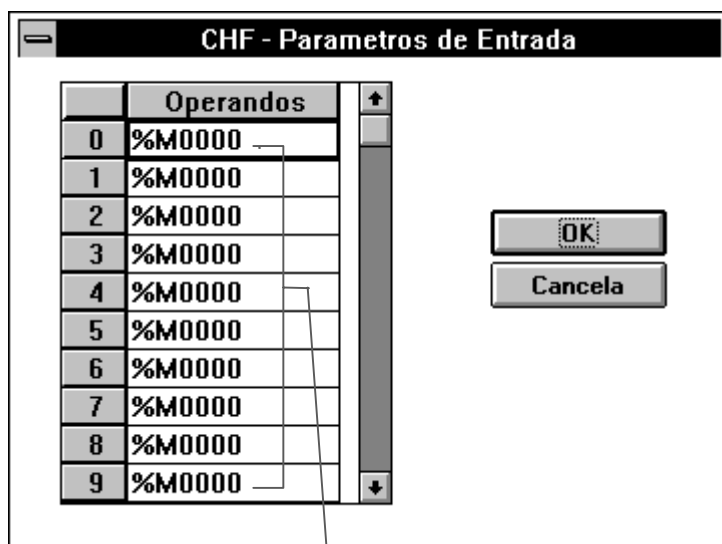
A passagem de parâmetros é realizada com a cópia dos valores dos operandos declarados (passagem de parâmetros por valor), embora esses operandos ainda continuem de uso global, utilizáveis por qualquer módulo presente no CP. Os módulos F podem ser programados de forma genérica, para serem reutilizados em diversos programas aplicativos como novas instruções. É aconselhável que os mesmos empreguem operandos próprios, não usados por nenhum outro módulo presente no programa aplicativo, evitando alterações inadvertidas em operandos utilizados em outros módulos.

É possível a passagem de operandos simples e constantes para o módulo F. A passagem de tabelas como parâmetros não é permitida, devido ao grande tempo que seria necessário para a cópia do conteúdo das mesmas. Entretanto, o endereço de uma tabela pode ser passado para o módulo F contido em um operando memória e dentro deste módulo serem realizados acessos indiretos à mesma.

Não é permitida a passagem de operandos com subdivisão para o módulo F, tais como %M004.2, %A0021n1, etc. Devem ser usados somente operandos simples para esta passagem.

#### Para realizar a edição de parâmetros.

1. Declarar o número de parâmetros a enviar e retornar em OPER2 e OPER3, limitados em 10 para cada um (%KM+00000 a %KM+00010).
2. Selecionar o botão Entrada. É exibida a janela CHF - Parâmetros de Entrada.
3. Posicionar o cursor no índice a ser editado e digitar o endereço ou tag do operando desejado para aquela posição.
4. Repetir o passo 3 até que todos os operandos utilizados como parâmetros de entrada tenham sido editados.
5. Selecionar o botão Ok.
6. Para editar os parâmetros de saída da CHF, repetir o passo 2 selecionando o botão **Saída**, e após repetir os passos 3, 4 e 5.

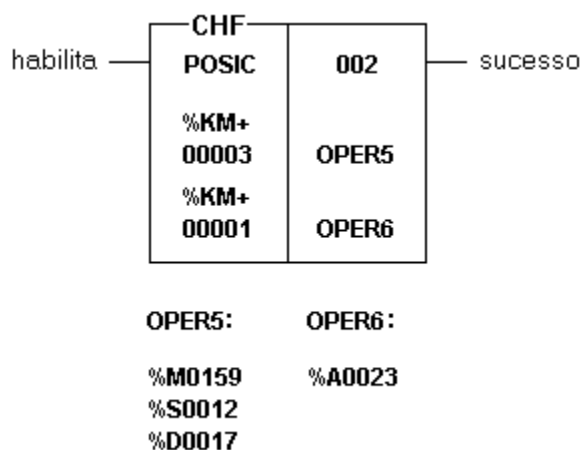


Digitar os operandos utilizados como parâmetros de entrada

**Figura 3-12 Caixa de Diálogo CHF - Parâmetros de Entrada**

Ver seção Utilização dos Módulos P e F no capítulo 2 deste manual.

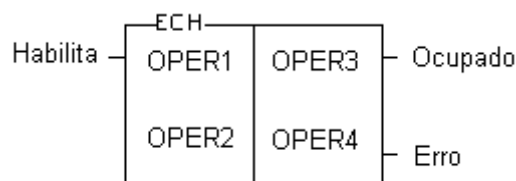
Caso o valor de OPER2 ou OPER3 seja maior do que 10, o MasterTool considera tal valor como igual a 10 (%KM+00010).

**Exemplo:****Sintaxe:**

OPER1	OPER2	OPER3	OPER4	OPER5	OPER6
NOME	%KM	%KM	NÚMERO	%KM %KD %KF %TM %TD %TF %M %D %F %E %S %A	%KM %KD %KF %TM %TD %TF %M %D %F %E %S %A

Tabela 3-45 Sintaxe da Instrução CHF

## ECH - Escrita de Operandos em Outro CP para Ethernet



OPER1 – endereço IP do controlador remoto

OPER2 – não utilizado

OPER3 - operando de controle da instrução

OPER4 - janela de edição dos operandos

### Descrição:

Esta instrução realiza a escrita de valores de operandos do controlador onde está sendo executada em operandos presentes em outros CPs, através da rede Ethernet de comunicação. Para o seu uso, portanto, é imprescindível que o controlador que a execute esteja conectado a outros CPs pela Ethernet.

Através da ECH podem ser transferidos valores de operandos individuais ou de conjuntos de operandos, sendo possível a programação de até 6 comunicações diferentes em uma mesma instrução.

Para programar a instrução, deve-se declarar na primeira célula (OPER1) o endereço IP do controlador programável destino que irá receber os valores escritos.

Na terceira célula (OPER3) deve ser declarado um operando decimal (%D) para ser utilizado pela própria instrução no controle do seu processamento.

### ATENÇÃO:

O operando %D programado em OPER3 não pode ter o seu valor modificado em nenhum outro ponto do programa aplicativo para o correto funcionamento da ECH. Conseqüentemente, cada nova instrução ECH ou LTH inserida no programa aplicativo deve utilizar um operando %D diferente das demais. Este operando não pode ser retentivo.

### Para realizar a edição dos parâmetros da ECH

1. Selecionar o botão **CP**. É exibida a caixa de diálogo **Parâmetros**.

**Figura 3-13 Caixa de Diálogo Parâmetros**

Esta caixa de diálogo está dividida em duas partes: CP LOCAL e CP REMOTO, cada qual contendo três colunas. Nas três colunas que compõem o CP local pode-se definir o operando ou o grupo de operandos cujos valores serão enviados para o controlador programável destino. Nas colunas pertencentes ao CP remoto, são declarados os operandos que irão receber os valores no controlador

destino, podendo ser de tipos diferentes do CP local. A caixa de diálogo possui seis linhas, permitindo que sejam definidas até seis comunicações diferentes na mesma instrução ECR para o mesmo controlador destino.

Os operandos especificados para o CP local são consistidos pelo MasterTool de acordo com as declarações constantes no módulo C presente no mesmo, por pertencerem ao programa aplicativo que está sendo editado. Os operandos declarados para o CP remoto não sofrem consistências quanto a tipo e endereços, por pertencerem a um programa aplicativo de outro controlador programável. Entretanto, o número de bytes ocupados pelo bloco de operandos declarados no CP local deve ser igual ao número de bytes ocupado pelos operandos do CP remoto em cada comunicação, para que a escrita seja realizada corretamente. O número máximo de bytes possível de ser ocupado por um bloco de operandos em cada comunicação é limitado em 220.

A seguir estão relacionados os tipos de operandos possíveis de serem programados para o CP local e remoto, com a disposição correta dos mesmos nas colunas de edição e os seus respectivos significados.

CP LOCAL ou CP REMOTO	Significado
%EXXXX	Operando individual %EXXXX
%SXXXX	Operando individual %SXXXX
%AXXXX	Operando individual %AXXXX
%MXXXX	Operando individual %MXXXX
%DXXXX	Operando individual %DXXXX
%FXXXX	Operando individual %FXXXX
%EXXXX .. %EYYYY	Grupo de operandos %EXXXX a %EYYYY
%SXXXX .. %SYYYY	Grupo de operandos %SXXXX a %SYYYY
%AXXXX .. %AYYYY	Grupo de operandos %AXXXX a %AYYYY
%MXXXX .. %MYYYY	Grupo de operandos %MXXXX a %MYYYY
%DXXXX .. %DYYYY	Grupo de operandos %DXXXX a %DYYYY
%FXXXX .. %FYYYY	Grupo de operandos %FXXXX a %FYYYY
%TMXXXX      YYY	Tabela %TMXXXX posição YYY
%TDXXXX      YYY	Tabela %TDXXXX posição YYY
%TFXXXX      YYY	Tabela %TFXXXX posição YYY
%TMXXXX      III ..      FFF	Tabela %TMXXXX posições III a FFF
%TDXXXX      III ..      FFF	Tabela %TDXXXX posições III a FFF
%TFXXXX      III ..      FFF	Tabela %TFXXXX posições III a FFF

**Tabela 3-47 Operandos para CP Local e Remoto em ECH**

O MasterTool permite a livre edição dos operandos dentro de uma mesma linha, possibilitando a troca de colunas com o auxílio das teclas de setas de movimentação horizontais. As consistências são realizadas na tentativa de troca de linha (setas verticais) ou confirmação do conteúdo editado na janela com a tecla ENTER. Pode-se desistir das alterações realizadas acionando-se a tecla ESC, permanecendo a instrução com o conteúdo anterior à abertura da janela de edição.

A tabela seguinte mostra o número de octetos ocupado por cada tipo de operando possível de ser programado nas definições de escritas.



Operando	Número de bytes
%E	1
%S	1
%A	1
%M	2
%F	4
%D	4
%TM	2 por posição
%TD	4 por posição
%TF	4 por posição

Tabela 3-48 Ocupação em Bytes dos Operandos da ECH

O cálculo do número de bytes ocupado nas declarações de CP local e remoto é realizado multiplicando-se o número de operandos declarados pelo número de octetos do tipo correspondente. Na tabela a seguir são mostrados alguns exemplos.

CP LOCAL ou CP REMOTO	Cálculo	Bytes
%E0004	1 operando x 1 byte	1
%S0020	1 operando x 1 byte	1
%A0018	1 operando x 1 byte	1
%M0197	1 operando x 2 bytes	2
%D0037	1 operando x 4 bytes	4
%E0005 .. %E0008	4 operandos x 1 byte	4
%S0024 .. %S0031	8 operandos x 1 byte	8
%A0089 .. %A0090	2 operandos x 1 byte	2
%M0002 .. %M0040	39 operandos x 2 bytes	78
%D0009 .. %D0018	10 operandos x 4 bytes	40
%TM0031 101	1 posição x 2 bytes	2
%TD0002 043	1 posição x 4 bytes	4
%TM0000 000 .. 002	3 posições x 2 bytes	6
%TD0007 021 .. 025	5 posições x 4 bytes	20

Tabela 3-49 Exemplos de Ocupação em Bytes

Ao ser acionada a entrada **habilita**, é disparada a comunicação da primeira escrita presente na ECH, sendo energizada a saída **ocupado** da mesma. No momento em que esta comunicação for completada, a instrução dispara a próxima escrita, independentemente do estado da entrada de habilitação, repetindo este procedimento para as demais comunicações existentes nesta instrução. Ao final da última escrita, a saída **ocupado** da ECH é desenergizada, com o disparo de um pulso com duração de uma varredura na saída **erro** caso não tenha sido possível realizar alguma comunicação.

Nos seis primeiros nibbles do operando D programado em OPER3 são colocados os estados das seis comunicações da instrução. Os últimos dois nibbles são utilizados para o controle do seu processamento.

Operando %D programado em OPER3 nas instruções ECH e LTR

7	6	5	4	3	2	1	0
		1	2	3	4	5	6

Controle da Instrução      Estado das Comunicações

Estado da comunicação 1 - Nibble 5  
 Estado da comunicação 2 - Nibble 4  
 Estado da comunicação 3 - Nibble 3  
 Estado da comunicação 4 - Nibble 2  
 Estado da comunicação 5 - Nibble 1  
 Estado da comunicação 6 - Nibble 0

**Figura 3-14 Operando de Controle da Instrução ECH e LTH**

O estado da comunicação armazenado em cada nibble é codificado da seguinte forma:

- 0 - comunicação com sucesso
- 1 - operando não definido
- 2 - endereço do controlador local igual ao remoto (comunicação para o próprio CP)
- 3 - bloco de operando inválido
- 4 - tipo de operando inválido
- 5 - time-out de transmissão de pacote
- 6 - não há espaço na fila de espera de transmissão
- 7 - falta de buffer de transmissão
- 8 - time-out de requisição
- 9 - erro de hardware
- 10 - CP remoto protegido

Em resumo, ao ser disparada a execução de uma instrução ECH todas as comunicações existentes na mesma são realizadas, mesmo que a sua entrada de habilitação seja desenergizada. Quando todas as escritas forem completadas, a próxima instrução ECH ou LTH encontrada no programa aplicativo com a entrada **habilita** energizada torna-se ativa, começando a processar as suas comunicações.

#### ATENÇÃO:

O programa aplicativo não pode realizar saltos sobre a instrução ECH ativa ou deixar de executar o módulo que a contém, para assegurar o seu correto processamento.

Em um programa aplicativo sendo executado no CP, em um dado momento, somente uma instrução de acesso à rede Ethernet (ECH ou LTH) é considerada ativa, mesmo que existam várias instruções com entrada habilita acionadas. A saída ocupado determina qual a instrução ativa, podendo ser utilizada para sincronizar as comunicações com o programa aplicativo. Para evitar sobrecargas no tráfego de informações na rede, aconselha-se disparar as instruções ECH periodicamente, evitando mantê-las permanentemente habilitadas no programa aplicativo, se possível. Um procedimento recomendado é desligar a entrada **habilita** logo que a saída **ocupado** for energizada, evitando um novo disparo da instrução após seu término.

Se a instrução for programada especificando-se o endereço IP igual ao endereço do próprio controlador que a executa (escrita de valores de si próprio), a saída erro é energizada.

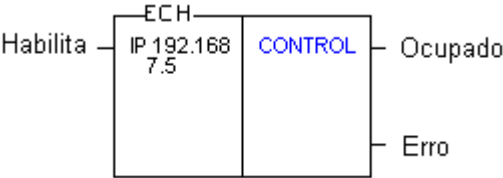
Caso nenhum operando tenha sido definido em OPER4, as saídas **erro** e **ocupado** ficam desenergizadas.

Sintaxe da Instrução:

OPER1	OPER2	OPER3	OPER4
Endereço IP		%D	COMUNICAÇÕES

Tabela 3-50 Sintaxe da Instrução ECH

Exemplo:

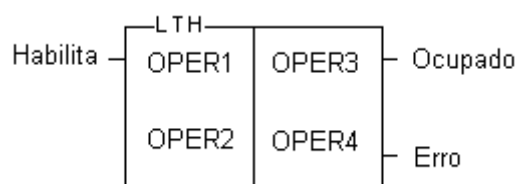


Conteúdo da janela de edição em OPER4 da ECH

COM	CP Local				CP Remoto			
1	%M0004				%A0014	..	%A0015	
2	%S0038		..	%S0041	%D0027			
3	%TD0007	028	..	030	%M0009	..	%M0014	
4	%M0006				%M0018			
5	%A0013		..	%A0020	%D0003	..	%D0004	
6	%TM0019	000	..	004	%TM0032	018	..	022

Esta instrução realiza escritas no controlador programável com o endereço IP igual a 192.168.7.5. São definidas seis comunicações para a mesma, transferindo dados de diversos tipos entre os CPs. A comunicação 0 envia o conteúdo de um operando memória no CP local para dois operandos auxiliares no CP remoto, sendo transferidos 2 octetos. As comunicações 1, 2, 3, 4 e 5 transferem, respectivamente, 4, 12, 2, 8 e 10 octetos entre os controladores.

## LTH - Leitura de Operandos de Outro CP para Ethernet



OPER1 - endereço IP do controlador remoto

OPER2 – não utilizado

OPER3 - operando de controle da instrução

OPER4 - janela de janela de edição dos operandos

### Descrição:

Esta instrução realiza a leitura de valores de operandos presentes em outros controladores programáveis para operandos do controlador programável onde está sendo executada, através da rede Ethernet de comunicação. Para o seu uso, portanto, é imprescindível que o CP que a execute esteja conectado a outros CPs pela Ethernet.

Através da LTH podem ser lidos valores de operandos individuais ou de conjuntos de operandos, sendo possível a programação de até 6 comunicações de leitura diferentes em uma mesma instrução.

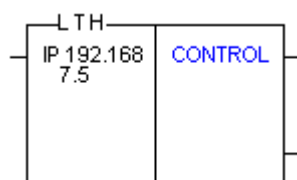
A programação da instrução LTH é idêntica à ECH, observando as mesmas restrições. Na LTH, a transferência dos valores ocorre dos operandos declarados no CP remoto para o CP local, sendo esta a única diferença entre ambas.

### Sintaxe da Instrução:

OPER1	OPER2	OPER3	OPER4
Endereço IP		%D	COMUNICAÇÕES

Tabela 3-51 Sintaxe da Instrução LTH

### Exemplo:



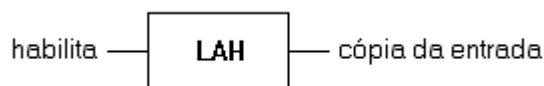
Conteúdo da janela de edição em OPER4 em uma LTH:

COM	CP Local				CP Remoto			
1	%M0004				%A0014	..	%A0015	
2	%S0038 .. %S0041				%D0027			
3	%TD0007	028	..	030	%M0009	..	%M0014	
4	%M0006				%M0018			
5	%A0013 .. %A0020				%D0003	..	%D0004	
6	%TM0019	000	..	004	%TM0032	018	..	022

Esta instrução realiza leituras no controlador programável com o endereço IP igual a 192.168.7.5. São definidas seis comunicações para a mesma, transferindo dados de diversos tipos entre os CPs. A comunicação 0 lê o conteúdo de dois operandos auxiliares no CP remoto para um operando memória

no CP local, sendo transferidos 2 octetos. As comunicações 1, 2, 3, 4 e 5 transferem, respectivamente, 4, 12, 2, 8 e 10 octetos entre os controladores programáveis.

## LAH - Libera Atualização de Imagens dos Operandos para Ethernet



### Descrição:

A instrução libera atualização da imagem de operandos realiza o processamento das comunicações pendentes da rede Ethernet para o CP local.

Ao retornar para o processamento do software executivo, ao final de cada varredura, o CP processa as requisições de leitura e outros serviços que tenham sido solicitados para o mesmo por outros CPs presentes na rede, durante a execução do programa aplicativo.

O controlador programável possui uma área de memória reservada para o armazenamento de até 32 comunicações recebidas durante o laço de execução do programa aplicativo, enquanto o software executivo não as processa. Caso o programa aplicativo possua tempo de execução relativamente alto e o controlador programável receba muitas requisições de serviços da rede, pode ocorrer a situação do CP não conseguir atendê-las, chegando ao limite de 32 comunicações pendentes à espera de processamento. Neste caso, o CP devolve uma resposta ao requisitante indicando a impossibilidade de atender a sua comunicação.

A instrução LAH executa o processamento de recepções e transmissões pendentes no CP, diminuindo a possibilidade de ocorrência da situação descrita anteriormente e reduzindo o tempo de atendimento às requisições. É recomendado o seu uso em programas aplicativos com alto tempo de ciclo, devendo ser inserida em pontos intermediários dos módulos, dividindo-os em trechos com aproximadamente 20 ms de tempo de execução.

### ATENÇÃO:

Os valores dos operandos do programa aplicativo podem ser modificados após a execução de uma LAH, pois outro equipamento ligado à rede pode estar solicitando escritas nos mesmos. Deve-se considerar a influência deste fato ao se inserir esta instrução no programa aplicativo.

## Instruções do Grupo Ligações

As instruções do grupo ligações permitem a construção de caminhos em série e em paralelo bem como a inversão do sinal.




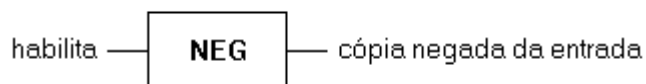
Nome	Descrição do Nome	Seqüência de Edição	Barra de Ferramentas
LGH	ligação horizontal	ALT, L, H	
LGN	ligação negada	ALT, L, N	
LGV	ligação vertical	ALT, L, V	

Tabela 3-52 Instruções do Grupo Ligações

### LGH - Ligação Horizontal



### LGN - Ligação Negada



### LGV - Ligação Vertical



#### Descrição:

As ligações são elementos auxiliares na construção dos diagramas de relés, para interligar as demais instruções.

A ligação negada inverte na sua saída o estado lógico da sua entrada.

# Módulos Função

Este capítulo contém a descrição dos módulos Função (F) que acompanham o MasterTool, disponíveis para os controladores programáveis da Série Ponto.

Os módulos Função implementam diversas rotinas de uso específico ou para o acesso aos módulos especiais de E/S pelo programa aplicativo, sendo similares às instruções, porém carregados como módulos do programa. A sua execução é disparada por outros módulos através da instrução CHF.

Os módulos que acompanham o MasterTool são programados em linguagem de máquina, não podendo ser lidos para o programador e visualizados como os módulos em diagrama de relés. Devem ser carregados diretamente do disco para o CP (opções **Comunicação, Ler/Enviar Módulo**).

Cada modelo de CP possui um conjunto de módulos função. A lista a seguir mostra as funções existentes para os CPs da Série Ponto.

- F-PID.033
- F-RAIZN.034
- F-ARQ2.035 à F-ARQ31.042
- F-MOBT.043
- F-RELG.048
- F-PID16.056
- F-CTRL.059
- F-NORM.071
- F-COMPF.072
- F-AES.087
- F-ANDT.090
- F-ORT.091
- F-XORT.092
- F-NEGT.093

Durante a instalação do MasterTool são copiados diversos módulos com o mesmo nome, sendo armazenados em subdiretórios diferentes, conforme o tipo de UCP ao qual se destinam. Mesmo possuindo o mesmo nome, estes módulos diferem no seu conteúdo.

## ATENÇÃO:

Os arquivos contidos no subdiretório de um CP não devem ser copiados para o de outro CP, sob o risco de perda de módulos. Deve-se carregar no controlador somente os módulos contidos no subdiretório correspondente à UCP utilizada.

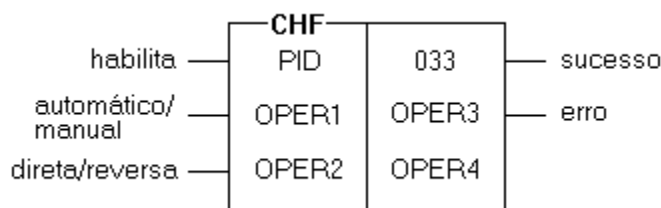
Em caso de dúvidas sobre o tipo de UCP para o qual o módulo foi programado, utilizar o comando de informações de arquivos (comando **Arquivo, Informações do Módulo** no MasterTool).

O tempo de execução de cada módulo função varia de acordo com o CP utilizado, portanto tal informação pode ser obtida no manual do respectivo CP.

Existem ainda muitos outros módulos funções disponíveis, que são mais específicos a algumas aplicações ou produtos. A lista completa de todos os módulos função disponíveis, bem como a forma de obtê-los pode ser encontrada no documento **Lista de Módulos F**.



## F-PID.033 - Função Controle PID



### Introdução

A função F-PID.033 implementa o algoritmo de controle proporcional, integral e derivativo. A partir de um valor medido (VM) e do ponto de ajuste desejado (PA) a função calcula o valor de atuação (VA) para o sistema controlado. Este valor é calculado periodicamente, levando em consideração os fatores proporcionais, integrais e derivativos programados. O diagrama em blocos da função é mostrado na figura 4-1.

As características mais importantes apresentadas pelo laço de controle implementado são:

- desaturação da ação integral (anti-reset windup)
- acompanhamento da saída no modo manual e comutação manual/automática balanceada (output tracking e bumpless transfer)
- ação direta ou reversa
- limites de saída máximo e mínimo ajustáveis
- ação derivativa calculada sobre várias amostragens
- capacidade de realizar integral discreta
- deslocamento com sinal
- tempo de execução de 1,6 ms no pior caso
- resolução de saída de 1:1000

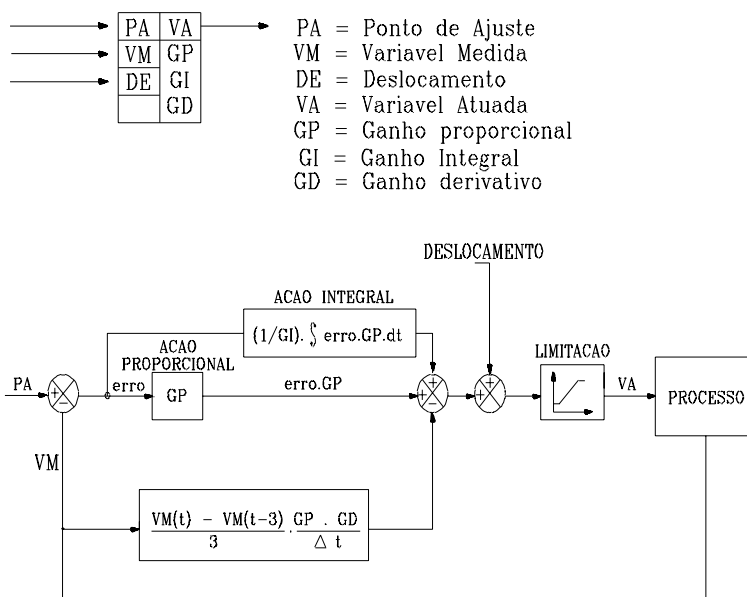


Figura 4-1 Diagrama em Blocos da Função PID

O uso da função PID no programa aplicativo permite uma série de facilidades que são facilmente integradas ao sistema, sem o uso de controladores externos. Por exemplo:

- função automático/manual
- inibição do fator integral ou derivativo
- laços cascadeados
- geração de curvas de ponto de ajuste
- modificação dos parâmetros de controle pelo programa
- modificação da política de controle em função do estado do processo

## Programação

### Operandos

As células da instrução CHF utilizada para a chamada da função são programadas do seguinte modo:

- **OPER1** - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deverá ser obrigatoriamente uma constante memória com valor 5 (%KM+00005).
- **OPER2** - Especifica o número de parâmetros que são passados para a função em OPER4. Este operando deverá ser obrigatoriamente uma constante memória com valor 0 (%KM+00000).
- **OPER3** - Contém os parâmetros que são passados para a função, declarados através de uma janela visualizada no MasterTool quando a instrução CHF for editada. O número de parâmetros editáveis é especificado em OPER1, sendo fixo em 5 para este módulo:

%TMXXXX - Tabela que contém os parâmetros utilizados pelo algoritmo de controle. Deve conter 16 posições.

%MXXXX - Memória que contém o valor medido do processo.

%MXXXX - Contém o ponto de ajuste (set point), que é o valor desejado para a variável medida. O seu valor pode ser modificado conforme a política de controle desejada.

%MXXXX - Memória que contém o valor de atuação no processo.

%AXXXX - Octeto auxiliar que contém pontos de controle da função PID.

- **OPER4** - Não utilizado.

### Entradas e Saídas

Descrição das entradas:

- **habilita** - quando esta entrada está energizada a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso o número de parâmetros ou seu tipo sejam diferentes das necessidades da função, todas as saídas da instrução são desenergizadas. Se estiverem corretos, o cálculo do controle PID é realizado.
- **automático(0)/manual(1)** - quando energizada, o operando de atuação não recebe o valor calculado pela função (modo manual).
- **direta(0)/reversa(1)** - especifica a forma de ação do controle.

Descrição das saídas:

- **sucesso** - é energizada quando a função foi corretamente executada.
- **erro** - é energizada caso ocorra erro na especificação dos operandos ou tentativa de acesso a operandos não declarados.

### Parâmetros Adicionais

Além dos operandos programados na instrução de chamada CHF, outros parâmetros devem ser carregados na tabela declarada em OPER3. Esta tabela deve conter 16 posições, sendo utilizada para definir os parâmetros utilizados pelo algoritmo de controle e armazenar resultados intermediários. A tabela 4-1 apresenta os parâmetros que devem ser carregados em cada posição de tabela, bem como seus valores mínimos e máximos.

P o	Parâmetro armazenado	Fórm.	Variação permitida	Valor tab
00	Ganho proporcional x 10	GP x 10	GP: 1,0 a 100,0	10 a 1000
01	Fat integral - parte frac.	dt / GI	GI: 1 a 1000 s/rep	0,0001 a 10,000
02	Fat integral - parte int		dt: 0,1 a 10 s	
03	Fat derivativo - parte frac.	GD / 3dt	GD: 1 a 1000 s	0,0333 a 3333,3333
04	Fat derivativo - parte int		dt: 0,1 a 10 s	
05	Deslocamento	DE	0 a 1000	0 a 1000
06	Valor mínimo da saída		0 a 1000	0 a 1000
07	Valor máximo da saída		0 a 1000	0 a 1000
08	Não utilizada			
09	Variável medida N - 1			0 a 1000
10	Variável medida N - 2			0 a 1000
11	Variável medida N - 3			0 a 1000
12	Erro			0 a 1000
13	Ação proporcional x 10			0 a 65535
14	Ação integral - parte frac x 10			0 a 65535
15	Ação integral - parte int x 10			0 a 65535

Tabela 4-1 Parâmetros Adicionais da PID

Para possibilitar uma maior velocidade de execução, alguns parâmetros devem ser carregados na tabela já pré-calculados. Sendo valores relativamente fixos, evita-se desta forma que sejam recalculados a cada chamada da função.

Os parâmetros que devem ser pré-calculados são:

- Ganho proporcional X 10 (posição 0) - É calculado multiplicando-se o ganho proporcional desejado por 10.
- Fator multiplicativo integral (posições 1 e 2) - É calculado dividindo-se o intervalo de amostragem (dt) pelo ganho integral desejado. A unidade de dt é segundos, sendo o seu valor mínimo de 0,1 segundos e máximo de 10,0 segundos e deve ser igual ao intervalo de tempo em que a rotina é executada. A unidade de GI é segundos/repetição, podendo variar de 1 até 1000 segundos/repetição. GI igual a 1 segundo/repetição significa o máximo efeito integral.
- Fator multiplicativo derivativo (posições 3 e 4) - É calculado dividindo-se o ganho derivativo (GD) pelo intervalo de amostragem (dt) e pelo valor 3. A unidade de GD é segundos, podendo variar de 1 até 1000 segundos. GD igual a 1000 segundos significa máximo efeito derivativo. Recomenda-se que quanto maior o valor de GD, maior deve ser o intervalo de amostragem. Mesmo para valores de GD = 1 segundo, o intervalo de amostragem deve ser maior que 0,2 segundos. Caso não seja tomado este cuidado, o termo derivativo produzirá apenas "ruído" e a ação de controle será muito brusca.
- Deslocamento (posição 5) - Permite que seja introduzido um deslocamento ("bias") no valor de atuação, evitando que erros negativos causem saturação no valor mínimo de saída. Geralmente este valor é ajustado para 50% (500) ou igual ao ponto de ajuste, se o ganho proporcional é pequeno.
- Valores mínimo e máximo de saída (posições 6 e 7) - São valores opcionais que limitam a excursão do valor de atuação, podendo serem modificados dinamicamente em função das condições operacionais. Se o valor máximo for maior ou igual a 1000 e o valor mínimo igual a 0, não é realizada nenhuma limitação.

O valor medido, o valor de atuação, o deslocamento, os valores máximo e mínimo têm como variação a faixa de 0 a 1000, o que corresponde a uma variação de 0 a 100% nas variáveis do processo.

As demais posições da tabela são utilizadas exclusivamente pela função PID, não devendo ser modificadas pelo programa aplicativo. A posição 12 (erro) pode ser consultada pelo programa. As posições 14 e 15 acumulam o fator integral, podendo serem zeradas, se necessário. Recomenda-se que estas posições estejam zeradas no início do processamento para evitar que valores aleatórios fiquem armazenados.

Além da tabela de parâmetros, alguns pontos de controle são utilizados pela função, contidos no octeto auxiliar especificado (%AXXXX).

- %AXXXX.4 - Sinal da ação integral - É utilizado pela função PID. Quando desenergizado, o termo integral é positivo, caso contrário é negativo. Pode ser lido pelo programa, se desejado.
- %AXXXX.5 - Sinal do deslocamento Indica para a função qual é o sinal do deslocamento, devendo ser acionado pelo programa. O ponto desenergizado indica deslocamento positivo. Quando energizado, o deslocamento é negativo.
- %AXXXX.6 - Inibe ação derivativa - Quando energizado, a função não executa a ação derivativa.
- %AXXXX.7 - Inibe ação integral - Quando energizado, a ação integral não é calculada, permanecendo a sua atribuição como o último valor calculado antes da inibição, a menos que os valores limites sejam excedidos.

### Características do Funcionamento

A desaturação da ação integral (anti-reset windup) é feita de modo a evitar que o termo integral continue a acumular erro quando um distúrbio no processo causa a saturação da saída do controlador em alguns dos limites. No momento em que o valor de saída atinge algum dos limites (máximo ou mínimo), o termo integral é fixado em seu valor corrente, impedindo o seu crescimento indefinido, sem influenciar na saída. Isto assegura que haverá uma resposta do controlador tão logo desapareça o distúrbio que o levou a saturar a saída.

A função pode ser executada em modo manual, energizando-se a segunda entrada da instrução CHF. Neste modo, a rotina não mais modifica o valor da saída de atuação, mas o acompanha (output tracking). Isto é, em função do valor da saída fixo e do valor medido do processo, os termos proporcional e derivativo são calculados e o termo integral é forçado para um valor adequado, de modo que, quando ocorrer a transição de manual para automático, a rotina possa reassumir o controle com o valor inicial da saída igual ao último valor da saída no modo manual. Chama-se este fato de comutação manual/automática balanceada (bumpless transfer).

A forma de controle pode ser direta ou reversa. Esta seleção é realizada desenergizando ou energizando a terceira entrada da instrução CHF. Caso o processo seja tal que o valor medido cresce quando o valor da saída de atuação cresce, a ação direta deve ser selecionada. Se o valor medido decresce com o aumento da saída de atuação, então a ação reversa deve ser utilizada.

O intervalo entre amostragens de um laço PID pode variar de 0,1 a 10,0 segundos. É de responsabilidade do usuário programar um "disparador" da função, ou seja, um trecho de programa aplicativo que somente habilite a rotina PID nos intervalos de tempo desejados. Note-se ainda que o valor do intervalo de amostragem usado para o cálculo dos fatores multiplicativos integral e derivativo deve coincidir com o intervalo de tempo das chamadas do "disparador". Como cada execução da rotina pode despendar até 3 ms, é aconselhável que cada laço de controle diferente seja disparado em diferentes varreduras do programa.

### Exemplo de Aplicação

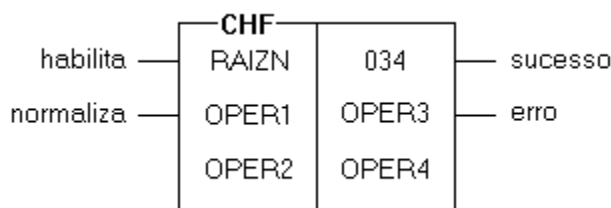
Como exemplo de utilização, sejam os seguintes valores de ajuste desejados para um laço de controle:

PA	=	62
GP	=	5 (GP = 100 / banda proporcional em %)
GI	=	100 segundos/repetição
GD	=	5 segundos
dt	=	1 segundo
DES	=	50%
MAX	=	80%
MIN	=	0%

Os valores que devem ser carregados na tabela de parâmetros são:

Posição	Valor	
0	50	GP X 10 (50)
1	100	dt / GI (0,0100)
2	0	
3	6666	GD / 3dt (1,6666)
4	1	
5	500	DES
6	0	MIN
7	800	MAX
8	620	PA

## F-RAIZN.034 - Função Raiz Quadrada



### Introdução

A função F-RAIZN.034 extrai a raiz quadrada de um valor fornecido em um operando memória ou real. No caso de operandos memória, o resultado pode ser normalizado para uma escala previamente definida.

O cálculo realizado corresponde à seguinte expressão:

**Op Destino** = Raiz Quadrada (Op Fonte) \* Constante de Normalização / 256

A normalização executada em conjunto com o processamento da raiz quadrada garante uma boa precisão dos resultados, pois são utilizadas variáveis internas com maior capacidade de armazenamento do que os operandos memória.

Esta função é tipicamente utilizada na linearização das leituras de transdutores que fornecem valores em escala quadrática, ou seja, com a saída proporcional ao quadrado do sinal medido.

### Programação

#### Operandos

As células da instrução CHF utilizada para a chamada da função são programadas do seguinte modo:

- **OPER1** - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deverá ser obrigatoriamente uma constante memória com valor 3 (%KM+00003).
  - **OPER2** - Deve ser um operando do tipo constante memória com valor 0 (%KM+00000). Determina o número de parâmetros possíveis de serem programados na janela de edição de OPER4. Como esta função não necessita de nenhum parâmetro em OPER4, o valor de OPER2 é 0.
  - **OPER3** - Contém os parâmetros que são passados para a função, declarados através de uma janela visualizada no MasterTool quando a instrução CHF for editada. O número de parâmetros editáveis é especificado em OPER1, sendo fixo em 3 para este módulo:
- %MXXXX ou %FXXXX - Operando com o valor a ser extraída a raiz quadrada (fonte). Este valor deve ser positivo para que o cálculo seja realizado.
  - %MXXXX ou %KM+XXXXX - Operando memória ou constante para a normalização de fundo de escala da raiz quadrada extraída. O valor programado é dividido por 256 e multiplicado pela raiz do operando fonte, gerando o valor do operando destino, quando a segunda entrada da instrução estiver energizada. Este operando não é utilizado quando os operandos fonte e destino são do tipo real.
  - %MXXXX ou %FXXXX - Operando que recebe o resultado da raiz quadrada normalizada (destino). Deve necessariamente ser do mesmo tipo que o operando fonte.
- **OPER4** - Não utilizado.

#### Entradas e Saídas

Descrição das entradas:

- habilita - quando esta entrada está energizada a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso os mesmos estejam incorretos, todas as saídas da instrução são desenergizadas. Se estiverem corretos, os cálculos são realizados, sendo acionadas as saídas sucesso ou erro.
- normaliza - quando energizada, realiza o ajuste de fundo de escala para o valor da raiz quadrada obtido. Se desenergizada, o valor do operando memória destino receberá simplesmente a raiz quadrada do operando fonte.

Descrição das saídas:

- sucesso - indica que o cálculo da raiz e a sua normalização foram realizados corretamente. Quando desenergizada, indica que a entrada habilita não está acionada, o módulo não está carregado no CP, os operandos não foram corretamente definidos ou existem valores negativos armazenados nos mesmos.
- erro - esta saída é energizada sempre que ocorre um dos seguintes erros:
  - existem valores negativos no operando fonte ou na constante de normalização
  - erro na especificação dos operandos ou tentativa de acesso a operandos não declarados
  - operando fonte com tipo diferente do operando destino

**ATENÇÃO:**

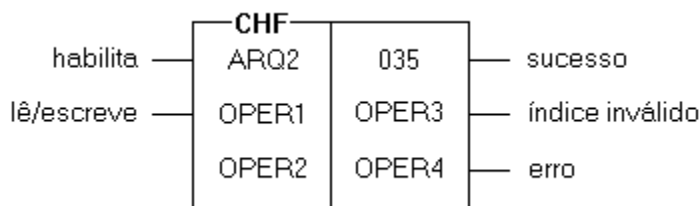
Na versão 1.00 de F-RAIZN.034 a saída de erro não é acionada na tentativa de acesso a operandos não declarados.

**Exemplo de Aplicação**

Para normalizar o valor do operando destino de forma que tenha a mesma escala do operando fonte, o valor a ser declarado no operando de normalização deve ser igual à raiz quadrada do valor do operando fonte multiplicado por 256.

Por exemplo, seja o caso de um transdutor que fornece valores de 0 a 1024, proporcionais ao quadrado de uma vazão, e deseja-se que estes valores sejam linearizados para a mesma escala de valores (0 a 1024). A constante de normalização programada é 8192 (raiz quadrada  $(1024) * 256$ ).

## F-ARQ2.035 a F-ARQ31.042 - Funções Arquivo de Dados



### Introdução

As funções arquivo de dados permitem o uso da memória do programa aplicativo para armazenar grandes quantidades de informações, utilizando conceitos de registros e campos. Desta forma obtém-se grande flexibilidade no aproveitamento dos bancos de memória do CP, além de um aumento substancial na capacidade de armazenamento de dados.

Existem diversas funções que implementam arquivos de dados, sendo idênticas no modo de programação e no funcionamento, diferindo-se apenas quanto à capacidade de armazenamento. Os módulos disponíveis são:

- F-ARQ2.035 - Função arquivo com 2 Kbytes de dados
- F-ARQ4.036 - Função arquivo com 4 Kbytes de dados
- F-ARQ8.037 - Função arquivo com 8 Kbytes de dados
- F-ARQ12.038 - Função arquivo com 12 Kbytes de dados
- F-ARQ15.039 - Função arquivo com 15 Kbytes de dados
- F-ARQ16.040 - Função arquivo com 16 Kbytes de dados
- F-ARQ24.041 - Função arquivo com 24 Kbytes de dados
- F-ARQ31.042 - Função arquivo com 31 Kbytes de dados

Cada arquivo pode possuir até 255 registros, numerados de 0 a 254, sendo que cada registro pode possuir até 255 campos, também numerados de 0 a 254. Note-se, entretanto, que a quantidade total de memória ocupada não pode exceder a capacidade do módulo.

Cada campo ocupa o mesmo número de bytes do operando onde serão realizadas as escritas ou leituras do arquivo.

### Programação

#### Operandos

As células da instrução CHF utilizada para a chamada da função são programadas do seguinte modo:

- **OPER1** - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deverá ser obrigatoriamente uma constante memória com valor 5 (%KM+00005).
- **OPER2** - Especifica o número de parâmetros que são passados para a função em OPER4. Este operando deverá ser obrigatoriamente uma constante memória com valor 0 (%KM+00000).
- **OPER3** - Contém os parâmetros que são passados para a função, declarados através de uma janela visualizada no MasterTool quando a instrução CHF for editada. O número de parâmetros editáveis é especificado em OPER1, sendo fixo em 5 para este módulo:

%MXXXX,                      %DXXXX,                      %AXXXX,                      %EXXXX,  
 %SXXXX, %TMXXXX, %TDXXXX, %KM+XXXXXX ou %KD+XXXXXXXX -



Operando de onde os dados serão lidos nas operações de escrita no arquivo ou para onde os dados serão copiados nas leituras do arquivo (parâmetro 1).

%MXXXX - Número do registro de/para o qual será copiado o dado (parâmetro 2). Deve estar compreendido entre 0 e o número de registros totais menos 1.

%MXXXX - Número do campo de/para o qual será copiado o dado (parâmetro 3). Deve estar compreendido entre 0 e o número de campos totais menos 1.

%KM+XXXXX - Número de registros totais (1 a 255) desejados para o arquivo (parâmetro 4).

%KM+XXXXX - Número de campos totais (1 a 255) desejados para o arquivo (parâmetro 5).

- **OPER4** - Não utilizado.

### Entradas e Saídas

Descrição das entradas:

- **habilita** - quando esta entrada está energizada a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso o número de parâmetros ou seu tipo sejam diferentes das necessidades do módulo, a saída de erro é energizada. Se estiverem corretos, é realizada uma tentativa de acesso ao arquivo.
- **lê/escreve** - quando energizada, o valor do primeiro parâmetro é copiado para o registro e o campo especificados no segundo e terceiros parâmetros. Se estiver desenergizada, o valor é lido do campo e copiado para o primeiro parâmetro.

Descrição das saídas:

- **sucesso** - indica que o acesso ao arquivo de dados foi corretamente realizado.
- **índice inválido** - esta saída é ligada se:
  - o campo a ser lido ou escrito não foi especificado
  - a declaração de registros e campos excede a capacidade de memória do módulo
  - houver tentativa de leitura quando o primeiro parâmetro for uma constante
  - houver tentativa de escrita estando o módulo armazenado em memória EPROM
- **erro** - é energizada caso ocorra erro na especificação dos parâmetros ou tentativa de acesso a operandos não declarados.

### Descrição do Funcionamento

Para declaração correta do número de campos e registros do arquivo, deve ser realizado o seguinte cálculo:

Ocupação do arquivo = Núm. registros X Núm. campos X Núm. bytes por campo  
 (parâmetro 4)                      (parâmetro 5)

O número de bytes por campo ocupado por tipo de operando pode ser obtido da tabela 4-2.

Parâmetro 1	Número de bytes por campo
%MXXXX	2
%DXXXX	4
%AXXXX	1
%EXXXX	1
%SXXXX	1
%TMXXXX	2
%TDXXXX	4
%KM+XXXXX	2
%KD+XXXXXXXX	4

Tabela 4-2 Ocupação dos Campos dos Arquivos

O valor obtido no cálculo anterior deve ser menor ou igual à capacidade total da função utilizada, conforme a tabela 4-3.

Função	Capacidade (bytes)
F-ARQ2.035	2048
F-ARQ4.036	4096
F-ARQ8.037	8192
F-ARQ12.038	12288
F-ARQ15.039	15360
F-ARQ16.040	16384
F-ARQ24.041	24576
F-ARQ31.042	31744

Tabela 4-3 Capacidade das Funções Arquivos de Dados

**ATENÇÃO:**

Em um mesmo programa podem ser declaradas diversas instruções CHF para acesso ao mesmo arquivo. Em todas estas instruções os operandos com os valores a serem escritos ou que recebem os valores lidos (parâmetro 1 em OPER3) devem possuir o mesmo número de bytes por campo.

Portanto, é possível ler ou escrever indistintamente operandos %E, %S e %A de um arquivo ou %KM, %M e %TM de outro. Contudo, não devem ser realizados acessos com operandos %M ou %D em um mesmo arquivo.

Se o primeiro parâmetro for uma tabela (%TM ou %TD), todos os campos do registro indicado no segundo parâmetro são copiados, ou seja, a transferência de dados é realizada entre o registro e a tabela, sendo que o valor do terceiro parâmetro (número do campo) é ignorado.

Caso a tabela possua menos posições do que o número de campos do registro, serão transferidos somente os campos que correspondem às posições existentes. Caso a tabela possua mais posições do que o número de campos do registro, serão transferidos somente os campos existentes.

A operação de escrita de dados copia-os para a própria área de memória ocupada pelo módulo função.

**ATENÇÃO:**

Caso o módulo F-ARQ esteja armazenado em Flash EPROM, não é possível escrever dados no arquivo, somente ler dados. Para que a escrita de dados nos arquivos ser realizada, os módulos F correspondentes aos mesmos devem estar na memória RAM do programa aplicativo.

**ATENÇÃO:**

Durante a leitura de um módulo arquivo de dados do CP com o programador MasterTool ou durante a sua transferência de RAM para Flash, não deve ser realizada nenhuma escrita de dados no mesmo

Isto porque a escrita de dados modifica o módulo lido, sendo considerado inválido pelo programador ou pelo CP devido à alteração do seu "checksum".

As funções arquivos de dados são módulos de programa aplicativo podendo ser carregados ou lidos do CP e armazenados em disquetes. Por exemplo, seja o caso de um CP controlando uma máquina injetora, armazenando diversos parâmetros de configuração em um módulo F-ARQ8.037. Depois que os parâmetros forem armazenados, este módulo F pode ser lido e armazenado em um disquete, para carga em outras máquinas injetoras iguais.

**Exemplo de Aplicação**

Como exemplo, caso seja desejado um arquivo com 120 registros e com 8 campos por registro para armazenar operandos %D, a ocupação de memória será:

Ocupação do arquivo = 120 registros X 8 campos/registro X 4 bytes/campo

Ocupação do arquivo = 3840

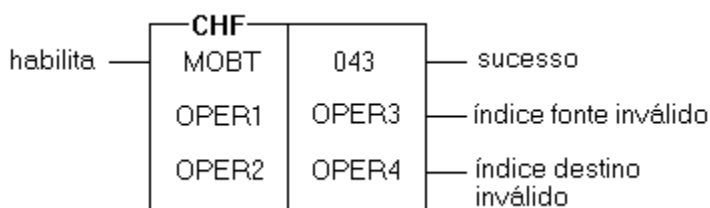
A configuração desejada ocupa 3840 bytes, devendo ser utilizado o módulo F-ARQ4.036, pois o mesmo permite o armazenamento de até 4096 bytes.

Os parâmetros programados em OPER3 da instrução CHF utilizada para o acesso ao arquivo são:

- %D0020 - operando para onde será lido ou com o valor a ser escrito no arquivo
- %M0100 - contém o número do registro a ser lido ou escrito, devendo estar compreendido entre 0 e 119 (120 registros totais)
- %M0101 - contém o número do campo a ser lido ou escrito, devendo estar compreendido entre 0 e 7 (8 registros totais)
- %KM+00120 - número total de registros

**%KM+00008 - número total de campos**

## F-MOBT.043 - Função para Movimentação de Blocos de Operandos Tabela



### Introdução

A função F-MOBT.043 realiza a cópia de blocos de operandos numéricos (%M ou %D) ou posições de tabelas (%TM ou %TD). Podem ser copiados até 255 valores de operandos simples para tabelas e vice-versa, transferindo-se também posições de uma tabela para outra. É possível especificar a posição inicial do bloco a ser copiado na tabela fonte e na tabela destino.

### Programação

#### Operandos

As células da instrução CHF utilizada para a chamada da função são programadas do seguinte modo:

- **OPER1** - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deverá ser obrigatoriamente uma constante memória com valor 5 (%KM+00005).
- **OPER2** - Deve ser um operando do tipo constante memória com valor 0 (%KM+00000). Determina o número de parâmetros possíveis de serem programados na janela de edição de OPER4. Como esta função não necessita de nenhum parâmetro em OPER4, o valor de OPER2 é 0.
- **OPER3** - Contém os parâmetros que são passados para a função, declarados através de uma janela visualizada no MasterTool quando a instrução CHF for editada. O número de parâmetros editáveis é especificado em OPER1, sendo fixo em 5 para este módulo:
  - %MXXXX, %DXXXX, %TMXXXX ou %TDXXXX - Operando inicial de onde os valores são copiados (operando fonte).
  - %KMXXXX - Posição inicial a ser transferida da tabela fonte. Este parâmetro é desconsiderado se o operando fonte for um operando simples (%M ou %D).
  - %MXXXX, %DXXXX, %TMXXXX ou %TDXXXX - Operando inicial para onde os valores são copiados (operando destino).
  - %KMXXXX - Posição inicial para onde serão copiados os valores na tabela destino. Este parâmetro é desconsiderado se o operando destino for um operando simples (%M ou %D).
  - %KMXXXX - Número de operandos simples ou posições de tabela a serem transferidos a partir do operando ou da posição inicial declarados nos parâmetros anteriores. Deve ser menor ou igual a 255.
- **OPER4** - Não utilizado.

**Entradas e Saídas**

Descrição das entradas:

- habilita - quando esta entrada está energizada a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso os mesmos estejam incorretos, são acionadas as saídas de índice inválido.

Descrição das saídas:

- sucesso - indica que a movimentação foi corretamente realizada.
- índice fonte inválido - indica que houve erro na especificação do operando fonte:
  - o operando não está declarado no módulo C
  - o tipo do parâmetro 2 não é %KM
  - não existe a posição inicial programada, se o operando fonte for tabela
  - não existem operandos ou posições de tabela suficientes para realizar a movimentação
- índice destino inválido - indica que houve erro na especificação do operando destino:
  - o operando não está declarado no módulo C
  - o tipo do parâmetro 4 não é %KM
  - não existe a posição inicial programada, se o operando destino for tabela
  - não existem operandos ou posições de tabela suficientes para realizar a movimentação

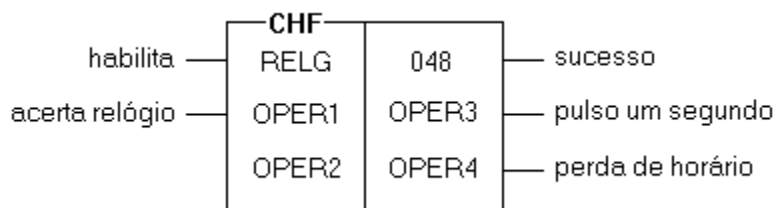
Caso as duas saídas de índice inválido sejam acionadas simultaneamente, ocorreu algum dos seguintes erros:

- o número de parâmetros programado em OPER1 é diferente de cinco
- o tipo do parâmetro 5 não é %KM
- o número total de posições a serem transferidas é maior que 255

**ATENÇÃO:**

Esta função permite a movimentação de um grande número de operandos em uma única varredura. Deve-se utilizá-la com cuidado para que o tempo máximo de ciclo do programa não seja excedido.

## F-RELG.048 - Função para Acesso ao Relógio de Tempo Real



### Introdução

A função F-RELG.048 realiza o acesso ao relógio de tempo real contido na UCP. O relógio possui horário e calendário completos, permitindo o desenvolvimento de programas aplicativos que dependam de bases de tempo precisas. A informação de tempo é mantida mesmo com a falta de alimentação do sistema, pois a UCP é alimentada por baterias.

Esta função possui características semelhantes à função F-SINC.049, pois ambas executam acessos ao mesmo relógio, diferindo apenas quanto aos métodos de acerto. Elas podem ser utilizadas simultaneamente em um mesmo programa, caso necessário.

### Programação

#### Operandos

As células da instrução CHF utilizada para a chamada da função são programadas do seguinte modo:

- **OPER1** - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deverá ser obrigatoriamente uma constante memória com valor 2 (%KM+00002).
- **OPER2** - Deve ser um operando do tipo constante memória com valor 0 (%KM+00000). Determina o número de parâmetros possíveis de serem programados na janela de edição de OPER4. Como esta função não necessita de nenhum parâmetro em OPER4, o valor de OPER2 é 0.
- **OPER3** - Contém os parâmetros que são passados para a função, declarados através de uma janela visualizada no MasterTool quando a instrução CHF for editada. O número de parâmetros editáveis é especificado em OPER1, sendo fixo em 2 para este módulo:
  - %MXXXX ou %TMXXXX - Especificação dos operandos para onde são lidos os valores do relógio. Se este parâmetro for especificado como memória, os valores são lidos para a memória declarada e as 6 subseqüentes. Se for especificado como tabela, os valores são colocados a partir da posição 0 até 6. Caso os operandos não estejam declarados, a leitura dos valores de tempo não é realizada e as saídas da instrução são desligadas. É possível o uso de tabelas com mais de 7 posições, sendo que a função ignora as posições excedentes. Os valores são lidos dos operandos na seguinte seqüência:

Operando	Posição Tabela	Conteúdo	Formato
%MXXXX	0	Segundos	000XX
%MXXXX + 1	1	Minutos	000XX
%MXXXX + 2	2	Horas	000XX
%MXXXX + 3	3	Dia do mês	000XX
%MXXXX + 4	4	Mês	000XX
%MXXXX + 5	5	Ano	000XX
%MXXXX + 6	6	Dia da semana	000XX

Tabela 4-4 Valores Lidos do Relógio (F-RELG.048)

O conteúdo destes operandos pode ser lido a qualquer momento, mas são atualizados com a hora real do relógio apenas quando a instrução for executada. Não devem ser modificadas em nenhum outro ponto do programa aplicativo. É utilizado o formato 24 horas na contagem do tempo. Os dias da semana são contados com valores de 1 a 7:

Valor	Dia da Semana
1	Domingo
2	Segunda-feira
3	Terça-feira
4	Quarta-feira
5	Quinta-feira
6	Sexta-feira
7	Sábado

Tabela 4-5 Valores dos Dias da Semana (F-RELG.048)

- **%MXXXX ou %TMXXXX** - Especificação dos operandos de onde são acertados os valores do relógio, com o acionamento de alguma das entradas de acerto da função. Se este parâmetro for especificado como memória, os valores são copiados da memória declarada e as 6 subseqüentes. Se for especificado como tabela, os valores são copiados da posição 0 até 6. Caso os operandos não estejam declarados, o acerto não é realizado e as saídas da instrução são desligadas. Os valores a serem copiados para o relógio devem ser colocados nos operandos na mesma seqüência dos operandos de leitura (segundos, minutos, horas, dia do mês, mês, ano e dia da semana).
- **OPER4** - Não utilizado.

### Entradas e Saídas

Descrição das entradas:

- **habilita** - quando esta entrada está energizada a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso os mesmos estejam incorretos, todas as saídas da instrução são desenergizadas. Se estiverem corretos, os valores de tempo do relógio são transferidos para os operandos memória ou para a tabela declarada como primeiro parâmetro em OPER3, a saída sucesso é energizada e a saída pulso um segundo é ligada por uma varredura a cada segundo.
- **acerta relógio** - quando energizada, os valores dos operandos declarados como segundo parâmetro em OPER3 são acertados no relógio, caso estejam com valores corretos. Enquanto esta entrada estiver acionada o tempo não é contado, permanecendo a saída pulso um segundo desenergizada.

### Exemplo:

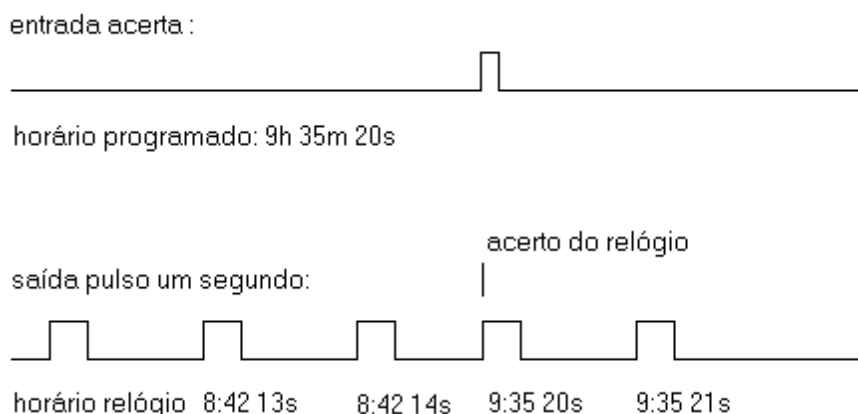


Figura 4-2 Exemplo de Diagrama de Tempos da Entrada Acerta

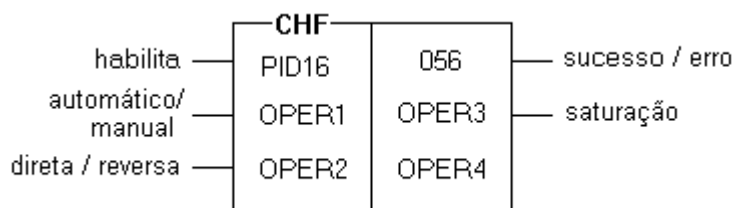
Descrição das saídas:

- **sucesso** - é energizada quando a função foi corretamente executada.

- pulso um segundo - indica se houve uma mudança no contador de segundos do relógio. O pulso dura uma varredura e pode ser usado para sincronizar eventos do programa aplicativo.
- perda de horário - esta saída é ligada caso o relógio tenha ficado sem a alimentação da bateria durante falha na alimentação principal. É desligada com o acerto do relógio.



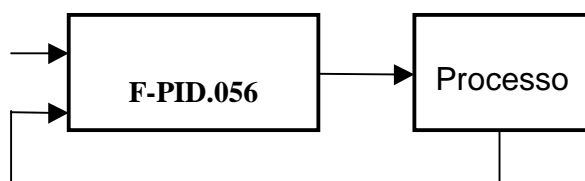
## F-PID16.056 - Módulo F para controle PID



### Introdução

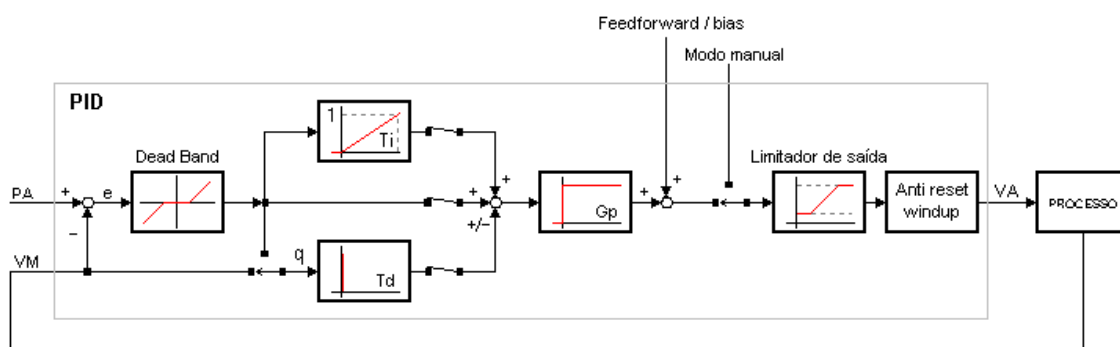
A função F-PID16.056, disponível para as UCPs AL-2003, AL-2004, PO3145, PO3142 e PO3242, implementa o algoritmo de controle proporcional, integral e derivativo. A partir de um valor medido (VM) e do ponto de ajuste desejado (PA) a função calcula o valor de atuação (VA) para o processo controlado. Este valor é calculado periodicamente, levando em consideração os fatores proporcional, integral e derivativo programados. Trata-se de um algoritmo de controle PID tipo ISA onde o ganho proporcional é o ganho do controlador, aplicado tanto ao erro como às parcelas integral e derivativa do controlador.

O módulo F pode ser representado pelo diagrama de blocos mostrado na figura 4-10.



**Figura 4-10 Diagrama de blocos do F-PID16.056**

Detalhes do diagrama de blocos do controlador PID podem ser observados na figura 4-11.



**Figura 4-11 Diagrama de blocos detalhado do F-PID16.056**

O algoritmo de controle básico utilizado para o controlador PID é descrito pela equação abaixo.

$$VA = Gp \cdot (e + \frac{1}{Ti} \cdot \int e \cdot dt \pm Td \cdot \frac{\partial q}{\partial t}) + Bias$$

Onde:

- $VA$  é a valor de atuação;
- $Gp$  é o ganho proporcional do controlador;
- $e$  é o erro do sistema ( $PA-VM$ );
- $Ti$  é a constante de tempo integral (s/rep);
- $Td$  é a constante de tempo derivativo em (s);
- $dt$  é o período de amostragem;
- $q$  representa o erro do sistema (+) ou a variável medida (-), conforme seleção;
- $PA$  é o ponto de ajuste;
- $VM$  é a variável medida no processo que está sendo controlado;
- $Bias$  é um deslocamento inserido através de um ponto de soma após o cálculo do algoritmo.

Além do algoritmo de controle básico descrito anteriormente, o módulo F-PID16.056 possui as seguintes características:

- Operação em 4 quadrantes (valores positivos e negativos em entradas e saídas);
- Pode ser usado em modo cascata, implementando algoritmos de controle complexos;
- Operação em 16 bits;
- Uso de parâmetros ( $Gp$ ,  $Ti$ ,  $Td$ ) diretamente no formato ISA;
- Seleção de termo derivativo agindo em função do erro (positivo) ou da variável medida (negativo);
- Inibição individual dos termos derivativo, integral ou proporcional;
- Ação derivativa calculada sobre três amostragens (filtro);
- Ação direta ou reversa;
- Limites de saída ajustáveis;
- Desaturação da ação integral (“anti reset windup”);
- Limitação da taxa de crescimento;
- “Feedforward / bias”;
- Modo manual ou automático;
- “Output tracking” para transição suave (bumpless) do modo manual para o modo automático;
- Zona morta configurável aplicada sobre o erro.

## Programação

### Operandos

As células da instrução CHF (figura 3) utilizadas para a chamada da função são programadas do seguinte modo:

- OPER1 - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deverá ser obrigatoriamente uma constante memória com valor 6 (%KM+00006).
- OPER2 - Especifica o número de parâmetros que são passados para a função em OPER4. Este operando deverá ser obrigatoriamente uma constante memória com valor 0 (%KM+00000).
- OPER3 - Contém os parâmetros que são passados para a função, declarados quando a instrução CHF for editada. O número de parâmetros editáveis é especificado em OPER1, sendo fixo em 6 para este módulo:
  - %TMXXXX - Tabela que contém os parâmetros utilizados pelo algoritmo de controle. Deve conter pelo menos 30 posições.
  - %MXXXX - Memória que contém o valor medido do processo (VM), normalmente obtido através de uma entrada analógica.
  - %MXXXX - Contém o ponto de ajuste (PA), que é o valor desejado para a variável medida ("set point").
  - %MXXXX - Memória que contém o valor de atuação (VA) gerado pelo algoritmo de controle. Em modo manual o algoritmo de controle não atua sobre esta variável, que pode ser manipulada pelo usuário.
  - %MXXXX - Memória para "*feedforward/bias*". O valor deste operando será somado ao valor de saída do controlador PID antes da limitação (limites superior e inferior declarados na tabela de parâmetros).
  - %AXXXX - Octeto auxiliar que contém pontos de controle da função PID.
- OPER4 - Não utilizado.

### Entradas e Saídas

#### Descrição das entradas:

- habilita - quando esta entrada está energizada a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso o número de parâmetros ou seu tipo sejam diferentes das necessidades da função, haverá a desenergização da saída sucesso/erro. Se estiverem corretos, o cálculo do controle PID é realizado.
- automático(0)/manual(1) - quando energizada, a variável de atuação não é alterada pela função podendo ser alterada manualmente (modo manual).
- reversa (0)/ direta (1) - especifica a forma como o controlador atuará sobre o processo.

#### ATENÇÃO:

O conceito de modo direto e reverso do módulo F-PID.033 está invertido em relação ao F-PID16.056.

#### Descrição das saídas:

- sucesso(1)/erro(0)- é energizada quando a função foi corretamente executada. Sempre que ocorrerem erros na especificação dos operandos, tentativa de acesso a operandos não declarados ou parâmetros inválidos, esta saída não é energizada, indicando erro.
- saturação – quando energizada indica que a saída do controlador atingiu a saturação, ou no limite máximo, ou no limite mínimo.

### Utilização

Este módulo F está disponível para os CPs AL-2003, AL-2004, PO3145, PO3142 e PO3242.

## Características de Funcionamento

### Dessaturação da Ação Integral

A dessaturação da ação integral (“anti-reset windup”) é feita de modo a evitar que o termo integral continue a acumular erro quando um distúrbio no processo causa a saturação da saída do controlador em um dos limites. No momento em que o valor de saída atinge um dos limites (máximo ou mínimo), o termo integral é fixado em seu valor corrente, impedindo o seu crescimento indefinido. Isto assegura que haverá uma resposta do controlador tão logo desapareça o distúrbio que o levou a saturar a saída.

### Modo Manual

A função pode ser executada em modo manual, energizando-se a segunda entrada da instrução CHF. Neste modo, a rotina não modifica o valor da saída de atuação, mas o acompanha (“output tracking”). Ou seja, em função do valor da saída no modo manual e do valor medido do processo, os termos proporcional e derivativo são calculados e o termo integral é forçado para um valor adequado de modo que, quando ocorrer a transição de manual para automático, a rotina possa reassumir o controle com o valor inicial da saída igual ao último valor da saída no modo manual. Chama-se este comportamento de comutação manual/automática balanceada (“bumpless transfer”). É importante observar que este recurso somente funciona quando a ação integral está habilitada. Embora em modo manual, a saída de atuação não pode assumir valores maiores que os limites declarados na tabela de parâmetros. Quando isto ocorrer o valor da saída de atuação será forçado para o limite mais próximo.

### Controle Direto e Reverso

A forma de controle pode ser direta ou reversa. Esta seleção é realizada desenergizando ou energizando a terceira entrada da instrução CHF.

Em caso de controle direto, caso ocorra um aumento no valor medido (VM), o controlador deve aumentar a saída de atuação (VA) a fim de controlar o processo.

Em caso de controle reverso, caso ocorra um aumento no valor medido (VM), o controlador deve diminuir a saída de atuação (VA) a fim de controlar o processo.

Considerando-se dois exemplos que utilizam a mesma válvula, controlada por uma saída analógica 4-20 mA (VA varia de 0 a 4095). Suponha que com  $VA = 0$  (4 mA) a válvula esteja totalmente fechada, e que com  $VA = 4095$  (20 mA) a válvula esteja totalmente aberta.

No primeiro exemplo, deseja-se controlar o nível de um tanque ( $VM = \text{nível do tanque}$ ), através de uma válvula de esgotamento do tanque. Portanto, quanto mais se abrir a válvula, mais rápido o nível do tanque diminui. Neste caso, caso aumente o nível (VM) do tanque, o controlador deve aumentar VA para abrir a válvula. Portanto, trata-se de um controle direto.

No segundo exemplo, deseja-se controlar a vazão através da válvula. Portanto, quanto mais se abrir a válvula, maior será a vazão. Neste caso, caso aumente a vazão (VM) através da válvula, o controlador deve diminuir VA para fechar a válvula. Portanto, trata-se de um controle reverso.

### Intervalo de amostragem

O intervalo entre amostragens de um laço PID pode variar de 0,01 a 10 segundos. É de responsabilidade do usuário programar um “disparador” da função, ou seja, um trecho de programa aplicativo que somente habilite a rotina F-PID16.056 nos intervalos de tempo desejados. Aconselha-

se utilizar um módulo E018, pois este módulo é executado dentro de um intervalo de tempo fixo que pode ser utilizado para gerar uma ou mais bases de tempo para a execução de um ou mais laços PID. Nota-se ainda que o valor do intervalo de amostragem ( $dt$ ) declarado na tabela de parâmetros do controlador deve coincidir com o intervalo de tempo das chamadas do "disparador".

## Tempo de execução

O pior caso de execução de um laço de controle com o F-PID16.056 atinge o tempo de 360  $\mu$ s. Este tempo é válido para as UCPs AL-2003, PO3145, PO3142 e PO3242.

## Descrição das Posições da Tabela de Parâmetros

A tabela com os parâmetros do controlador também é utilizada para armazenar variáveis de uso interno, o somatório da ação integral e as variáveis medidas ou erros de ciclos anteriores para o cálculo da ação derivativa. Cada posição desta tabela está descrita abaixo, na tabela 1.

Posição da tabela	Parâmetros	Descrição
00	GP (x10)	Ganho proporcional (sem unidade). Os valores possíveis estão dentro do intervalo de 0,1 a 3000. O ganho proporcional deve ser multiplicado por 10 para ser declarado neste campo, assumindo o intervalo de 1 a 30000. Valores fora desta faixa farão com que o módulo F entre em modo erro e não execute o algoritmo de controle.
01	Ti (x10)	Constante de tempo integral (s/repetição). Os valores possíveis estão dentro do intervalo de 0,1 a 3200 s/rep. A constante de tempo integral deve ser multiplicada por 10 para ser declarada neste campo, assumindo o intervalo de 1 a 32000. Observa-se aqui que quanto menor o valor de Ti, maior será a ação integral. Valores fora desta faixa farão com que o módulo F entre em modo erro e não execute o algoritmo de controle.
02	Td (x100)	Constante de tempo derivativa (s). Os valores possíveis estão dentro do intervalo de 0 a 320 s em unidades de 0,01s. A constante de tempo derivativo deve ser multiplicada por 100 para ser declarada neste campo, assumindo o intervalo de 0 a 32000. Valores fora desta faixa farão com que o módulo F entre em modo erro e não execute o algoritmo de controle. Se for atribuído o valor zero à constante de tempo derivativa, a ação será calculada com valor zero, não influenciando na saída de atuação. Recomenda-se desabilitar a ação derivativa quando esta não é utilizada.
03	dt (x100)	Intervalo de amostragem do processo que está sendo controlado (s). Os valores possíveis estão dentro do intervalo de 0,01 a 10 s. O intervalo de amostragem deve ser multiplicado por um fator de 100 para ser declarado neste campo, assumindo o intervalo de 1 a 1000. Valores fora desta faixa farão com que o módulo F entre em modo erro e não execute o algoritmo de controle. É responsabilidade do usuário disparar o módulo F neste intervalo de tempo.
04	Valor máximo de saída	Valor máximo de saída permitido. Pode assumir valores de -30000 a +30000. Deve necessariamente ser maior que o valor mínimo de saída. Valores fora desta faixa farão com que o módulo F entre em modo erro e não execute o algoritmo de controle.
05	Valor mínimo de saída	Valor mínimo de saída permitido. Pode assumir valores de -30000 a +30000. Deve necessariamente ser menor que o valor máximo de saída. Valores fora desta faixa farão com que o módulo F entre em modo erro e não execute o algoritmo de controle.
06	Zona morta	Zona morta. Pode assumir valores de 0 a +30000. Sempre que o valor absoluto do erro for menor que o valor definido neste campo, o controlador será executado considerando o erro como zero. Para desabilitar este recurso basta declarar o valor zero para a zona morta. Valores fora desta faixa farão com que o módulo F entre em modo erro e não execute o algoritmo de controle.
07	Variação máxima permitida.	O valor declarado neste campo indica o valor absoluto da variação máxima que a saída do controlador pode ter a cada intervalo de amostragem ( $dt$ ). Pode assumir valores de 1 a 30000. O valor 1 representa uma variação muito pequena enquanto que o valor 30000 representa uma grande variação a cada ciclo de amostragem. Declarando zero neste campo a variação máxima permitida não é verificada, permitindo-se qualquer variação. Valores fora desta faixa farão com que o módulo F entre em modo erro e não execute o algoritmo de controle.

08	AI acumulada x100 (Hi)	O valor presente neste campo é a ação integral acumulada. Visando obter mais resolução numérica, a ação integral acumulada é armazenada multiplicada por um fator de 100. São utilizados três operandos de 16 bits para guardar a parte alta, baixa e parte fracionária da ação integral. Estes campos devem ser inicializados com zero para evitar que algum valor aleatório fique armazenado.
09	AI acumulada x100 (Lo)	
10	AI acumulada x100 (frac)	
11	Valor de atuação anterior (VA)	Este campo é de uso restrito do módulo F e não deve ter seu conteúdo alterado. O valor deste campo é a variável de atuação do ciclo anterior, utilizada para limitar a variação máxima.
12-13	Reservado	Operando reservados.
14	VM(t-3) ou erro(t-3)	Este campo é de uso restrito do módulo F e não deve ter seu conteúdo alterado. Histórico dos três últimos erros ou variáveis medidas, utilizadas para cálculo do termo derivativo. A ação derivativa pode agir tanto em função do erro como da variável medida, porém jamais deve-se trocar a seleção de erro para variável medida ou vice-versa durante o processo de controle.
15	VM(t-2) ou erro(t-2)	
16	VM(t-1) ou erro(t-1)	
17-27	Uso interno	Estas posições da tabela são utilizadas exclusivamente pela função PID, não devendo ser modificadas pelo programa aplicativo.
28-29	Reservado	Operandos reservados.

Tabela 6: Descrição das posições da tabela de parâmetros do F-PID16.056

Sempre que ocorrer alguma alteração nos parâmetros GP, Ti, Td ou dt o módulo F-PID16.056 necessita um ciclo de execução para adaptar o controlador aos novos parâmetros, não executando o controle neste ciclo e mantendo a variável de atuação (VA) inalterada.

## Descrição do Operando %A de Controle

O operando auxiliar de controle do módulo F é utilizado conforme a tabela 2.

Bit	Descrição
0	inibe(1) / habilita(0) ação integral
1	inibe(1) / habilita(0) ação derivativa
2	inibe(1) / habilita(0) ação proporcional
3	Ação derivativa em função do erro (1) ou da variável do processo (0)
4	Reservado
5	Uso interno
6	Uso interno
7	Uso interno

Tabela 7: Descrição do operando auxiliar de controle

Através do operando auxiliar de controle da função F-PID16.056 é possível desabilitar a ação proporcional, integral e/ou derivativa e também selecionar a ação derivativa agindo em função do erro ou da variável medida no processo. Quando alguma ação de controle (seja ela proporcional, integral ou derivativa) não for utilizada, esta deve ser desabilitada ligando o bit correspondente.

Desabilitando a ação de controle proporcional deixa-se de gerar ação de controle proporcional ao erro, mas o ganho do sistema continua sendo aplicado sobre as ações integral e derivativa. Para um integrador puro, por exemplo, deve-se habilitar somente a ação integral, ajustar a constante de tempo Ti desejada e atribuir à constante GP (ganho proporcional) um ganho unitário.

A ação derivativa agindo em função da variável medida é recomendada para a maioria das aplicações, pois evita grandes variações na saída VA quando o ponto de ajuste PA é modificado. Para aplicações especiais existe a possibilidade de seleção da ação derivativa em função do erro do sistema.

Os bits de uso interno são de uso exclusivo da função F-PID16.056 e não devem ter seu conteúdo alterado.

## Notas de Aplicação

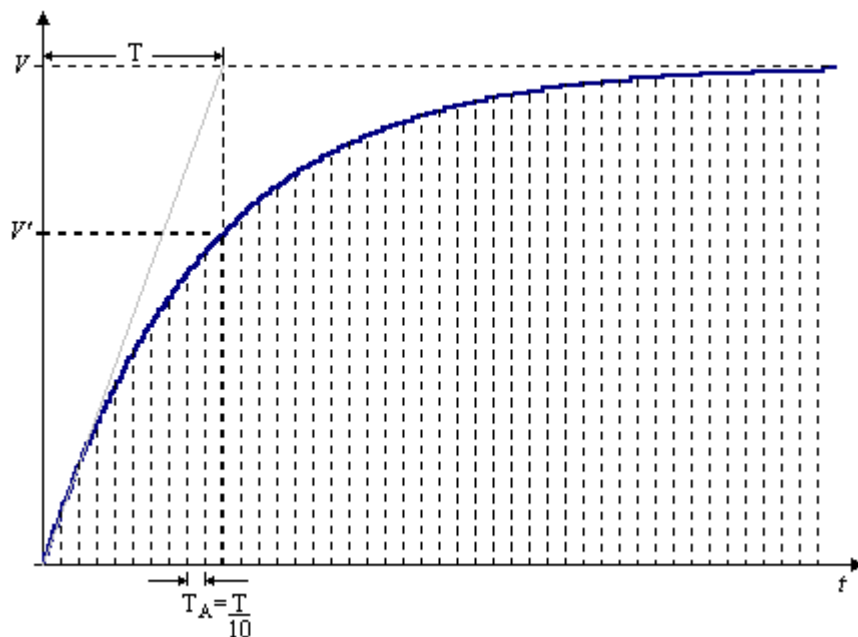
### Seleção do Tempo de Amostragem

A eficiência do controlador digital está diretamente relacionada com o intervalo de amostragem utilizado. A medida que este intervalo diminui, o resultado do controlador digital aproxima-se do resultado de um controlador analógico. Aconselha-se utilizar um tempo de amostragem da ordem de um décimo da constante de tempo do sistema, ou seja

$$T_A = \frac{T}{10}$$

onde  $T_A$  é o tempo de amostragem utilizado e  $T$  é a constante de tempo do sistema.

Exemplo: Pode-se obter a constante de tempo de um sistema de primeira ordem a partir do seu gráfico da resposta da variável de atuação (VA) a um degrau no ponto de ajuste PA com o laço de controle aberto (PID desabilitado ou em modo manual), conforme a figura 4.



**Figura 33: Constante de tempo do sistema e intervalo de amostragem**

Esta figura demonstra a obtenção da constante de tempo do sistema por dois modos distintos. O mais usual é tomar como constante de tempo do sistema o tempo necessário para o sistema atingir 63,212% do valor final. Outro modo é traçar a primeira derivada da curva da resposta ao degrau, a constante de tempo é aquela onde esta reta cruza o valor final da resposta do sistema.

Uma vez definida a constante de tempo, basta definir o intervalo de amostragem da ordem de um décimo deste valor.

É importante lembrar que na Série Ponto a atualização das entradas e saídas ocorre na mesma ordem de tempo de um ciclo do CP. Sempre que o tempo de ciclo do CP for maior que o tempo de amostragem aconselha-se o uso da instrução F-AES.087.

### Feedforward/Bias

Através do operando memória utilizado para *feedforward/bias* é possível injetar alguma variável do sistema na saída do controlador e/ou aplicar um deslocamento na mesma.

O objetivo do feedforward é medir os principais distúrbios do processo e calcular a mudança necessária na variável de atuação para compensá-los antes que estes causem alterações na variável controlada. A manipulação dos distúrbios do processo pode ser feita através dos blocos de controle avançado (F-CTRL.059) que disponibilizam blocos avanço-atraso, derivador com retardo temporal e retardo temporal de primeira ordem.

Pode-se citar como exemplo, um sistema onde a variável a ser controlada é a temperatura de uma mistura quente. Numa determinada fase do processo é necessário derramar água fria nesta mistura. Sem o feedforward, seria necessário esperar a água fria mudar o estado da mistura para então o controlador gerar a ação corretiva. Utilizando o *feedforward*, um valor associado à temperatura da água fria seria injetado na saída do controlador, fazendo com que este tome uma “ação corretiva” antes mesmo da água fria começar a alterar o estado da mistura quente, agilizando a resposta do controlador.

O “bias” é utilizado sempre que se deseja aplicar algum deslocamento sobre a saída do controlador.

### Controle em Cascata

Provavelmente o controle em cascata é uma das técnicas de controle avançado mais utilizadas na prática. É composto por pelo menos duas malhas de controle. A figura 5 mostra um controlador em cascata com duas malhas.

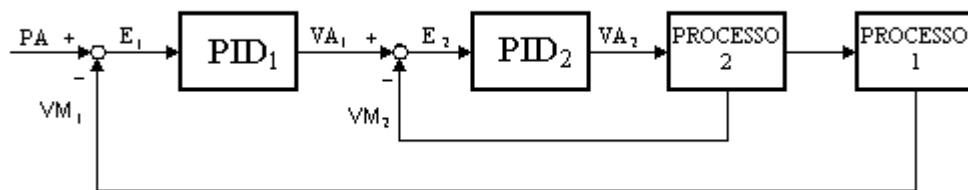
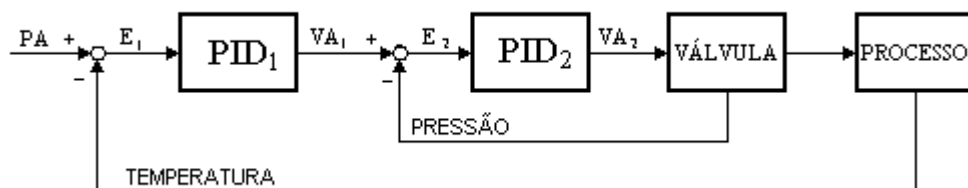


Figura 34: Controlador em cascata com duas malhas

A malha externa é chamada de controlador mestre e a malha interna de controlador escravo. O controlador mestre tem seu ponto de ajuste fixo e sua saída fornece o ponto de ajuste do controlador escravo ( $VA_1$ ). A variável de atuação do controlador escravo ( $VA_2$ ) atuará sobre o processo 2 que, por sua vez, atuará sobre o processo 1, fechando a malha do controlador mestre.

Este tipo de controlador é aplicado, por exemplo, no controle de temperatura pela injeção de vapor. Além da variação da temperatura, que deve ser controlada, o sistema está sujeito a variações de pressão na linha de vapor. Torna-se então desejável um controlador de vazão escravo atuando em função das variações de pressão e um controlador mestre para manipular a referência do escravo controlando então a temperatura do processo. Este exemplo pode ser representado graficamente conforme a figura 6.





**Figura 35: Aplicação de um controlador em cascata**

Caso fosse utilizado somente um controlador de temperatura atuando diretamente sobre a válvula de vapor, não haveria como compensar eventuais variações de pressão na linha de vapor.

Existem três principais vantagens no uso de controladores em cascata:

- Qualquer distúrbio que afete o controlador escravo é detectado e compensado por este controlador antes de afetar a variável controlada pelo controlador mestre;
- Aumento da controlabilidade do sistema. No caso do controle de temperatura pela injeção de vapor, a resposta do sistema é melhorada devido ao controlador de vazão aumentando a controlabilidade do laço principal.
- Não linearidades de um laço interno são manipuladas dentro deste laço e não percebidos pelo laço externo. No exemplo anterior, as variações de pressão são compensadas pelo controlador escravo e o controlador mestre “enxerga” apenas uma relação linear entre a válvula e a temperatura.

**Considerações Importantes**

Para se utilizar controladores em cascata deve-se tomar os seguintes cuidados:

- Como o ponto de ajuste dos controladores escravos é manipulado conforme a saída dos controladores mestres, poderão ocorrer variações bruscas no erro do controlador escravo. Se os controladores escravos estiverem com a ação derivativa agindo em função do erro surgirão ações derivativas com grandes valores. Portanto aconselha-se utilizar os controladores escravos com a ação derivativa em função da variável medida.
- O controlador escravo deve ser rápido o suficiente para eliminar os distúrbios de seu laço antes que estes afetem o laço do controlador mestre.

## Sugestões para Ajustes do Controlador PID

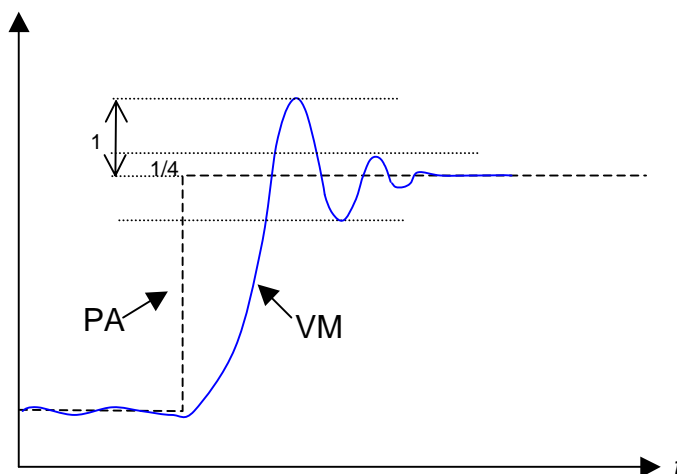
A seguir são apresentados dois métodos para a determinação das constantes do controlador PID. O primeiro método consiste na determinação das constantes em função do período de oscilação e do ganho crítico, enquanto que o segundo determina as constantes do controlador em função da constante de tempo ( $T$ ), do tempo morto ( $T_m$ ) e do ganho estático do sistema ( $K$ ). Para maiores detalhes aconselha-se a leitura da literatura referenciada.

### ATENÇÃO:

A Altus Sistemas de Informática não se responsabiliza por eventuais danos causados por erros de configuração das constantes do controlador ou parametrização. Recomenda-se que pessoa devidamente qualificada execute esta tarefa.

## Determinação das Constantes do Controlador Através do Período e do Ganho Crítico

Este método gera uma resposta amortecida cuja taxa de amortecimento é igual a  $1/4$ . Isto é, depois de sintonizar um laço através deste método, espera-se um resposta como mostrada na figura 7:



**Figura 36:Resposta com Amortecimento de 1/4**

O ganho crítico é definido como o ganho de um controlador proporcional que gera uma oscilação de amplitude constante no sistema em malha fechada enquanto que o período crítico é o período desta oscilação. O ganho crítico é uma medida de controlabilidade do sistema, ou seja, quanto maior o ganho crítico mais fácil será o controle do sistema. O período crítico de oscilação é uma medida da velocidade de resposta do sistema em malha fechada, ou seja, quanto maior o período de oscilação mais lento será o sistema. No decorrer deste capítulo o ganho crítico será denominado como  $G_{Pc}$  e o período crítico como  $T_c$ .

É importante lembrar que ganhos ligeiramente menores que  $G_{Pc}$  geram oscilações cujo período decresce com o tempo, enquanto que ganhos maiores que  $G_{Pc}$  geram oscilações cuja amplitude cresce com o tempo. No caso de ganhos maiores que  $G_{Pc}$  é preciso ter cuidado para não tornar o sistema criticamente instável.

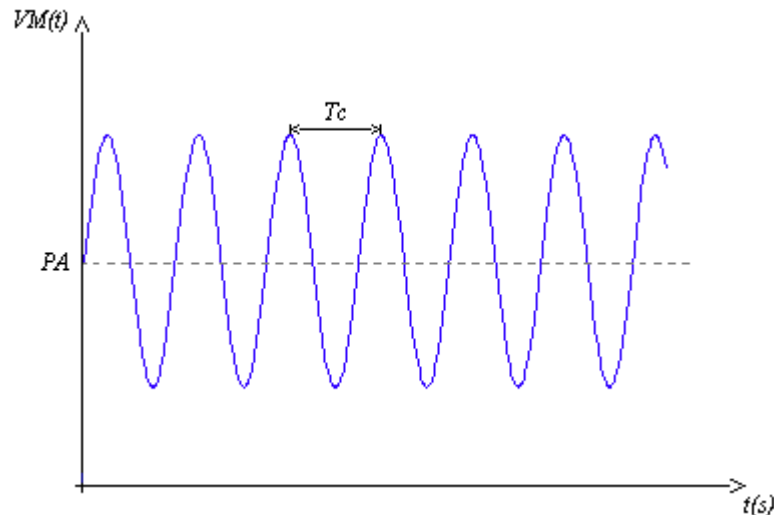
O processo para a determinar  $G_{Pc}$  e  $T_c$  consiste em fechar a malha com o controlador em modo automático desabilitando a ação integral e a derivativa. Os passos são os seguintes.

- Remover a ação integral e derivativa através do operando %A de controle;
- Aumentar o ganho proporcional com pequenos incrementos. Depois de cada incremento inserir um pequeno distúrbio no sistema através de um pequeno degrau no ponto de ajuste (PA). Verificar o comportamento do sistema (VM), a amplitude de oscilação deve aumentar à medida

que o ganho aumenta. O ganho crítico ( $GP_c$ ) será aquele que gerar oscilações com amplitude constante (ou quase constante) conforme a figura 8;

- Medir o período destas oscilações ( $T_c$ ).

Para determinar as constantes do controlador basta aplicar os valores de  $GP_c$  e  $T_c$  nas fórmulas da tabela 3.



**Figura 37: Representação gráfica de um sistema oscilando quando sujeito ao  $GP_c$**

Tipo de Controlador	Constantes
Proporcional (P)	$GP = 0,5 \cdot GP_c$
Proporcional e Integral (PI)	$GP = 0,45 \cdot GP_c$ $Ti = \frac{T_c}{1,2}$
Proporcional, Integral e Derivativo (PID)	$GP = 0,75 \cdot GP_c$ $Ti = \frac{T_c}{1,6}$ $Td = \frac{T_c}{10}$

**Tabela 8: Fórmulas para determinar as constantes do controlador**

### Determinação das Constantes do Controlador Através das Constantes do Processo

Este método se aplica bem a processos lineares, de primeira ordem (similar a um circuito RC) e com tempo morto. Na prática, muitos processos industriais se adaptam a este modelo.

O método requer, inicialmente, determinar as seguintes características do processo em laço aberto:

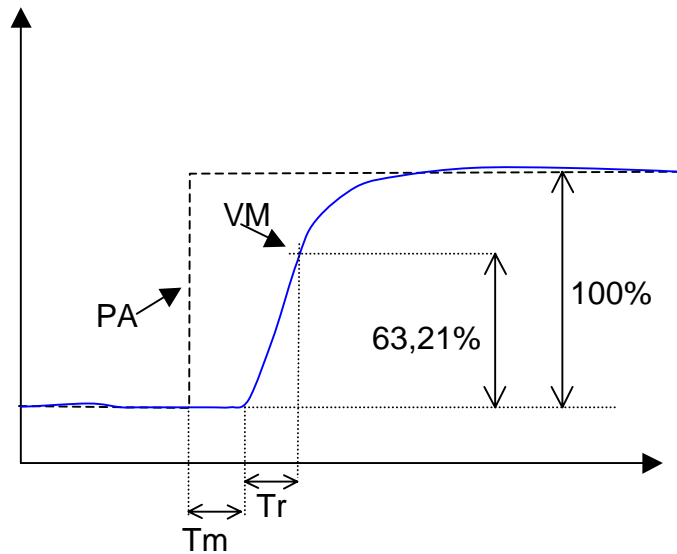
- $K$ : Ganho estático do processo. Definido como a razão entre uma variação de  $VM$  e uma variação de  $VA$ , ou seja,  $K = \Delta VM / \Delta VA$ ;
- $T_m$ : Tempo morto, definido como o tempo entre o início de uma variação na saída  $VA$  ( $t_0$ ) e o início da reação do sistema.
- $T$ : Constante de tempo do sistema, definido como o tempo que a variável medida leva para excursionar 63,212% de seu valor final.

Além disso, o método requer dois parâmetros adicionais, que não são características do processo em si, e devem ser informados pelo usuário:

$T_r$ : tempo de resposta desejado após a sintonia do laço. Trata-se de uma característica interessante, pois através deste parâmetro o usuário pode informar um requisito de performance do laço controlado.

$dt$ : tempo de amostragem em segundos, isto é, o período de chamada da F-PID16.056 e atualização da entrada VM e saída VA. A constante  $dt$  simboliza um tempo morto adicional, que deve ser somado a  $T_m$ . Na prática, soma-se  $dt/2$  ao valor de  $T_m$ , pois este é o tempo morto médio inserido.

O tempo de resposta  $T_r$  pode ser comparado com uma “constante de tempo” do laço fechado, conforme ilustra a figura 9.



**Figura 38: Especificação do Tempo de Resposta  $T_r$**

O parâmetro  $T_r$ , na figura 9, mostra o tempo de resposta desejado. Trata-se do tempo medido entre o início da resposta do sistema (após o tempo morto  $T_m$ ), e o momento em que VM atinge 63,21% de sua excursão total. Através de  $T_r$  o usuário pode especificar um “requisito de performance” para o laço controlado. Deve-se ter o cuidado de não especificar tempos de resposta menores que um décimo da constante de tempo do sistema, pois do contrário o sistema pode ficar instável. Quanto menor o valor de  $T_r$ , maior o ganho necessário.

A seguir, descreve-se como determinar, através de um teste de laço aberto, os demais parâmetros ( $K$ ,  $T_m$  e  $T$ ), que caracterizam o processo. Um modo simples para determinar estas constantes do processo é colocar o módulo F-PID16.056 em modo manual, gerar um pequeno degrau em VA e plotar a resposta de VM no tempo. Para processos lentos isto pode ser feito manualmente, mas para processos rápidos aconselha-se o uso de um osciloscópio ou qualquer outro dispositivo que monitore a variação de VM. O degrau em VA deve ser grande o suficiente para causar uma variação perceptível em VM.

As figuras 10 e 11 representam respectivamente um degrau na saída VA, aplicado no instante  $t_0$ , e a resposta de um sistema linear de primeira ordem com tempo morto.

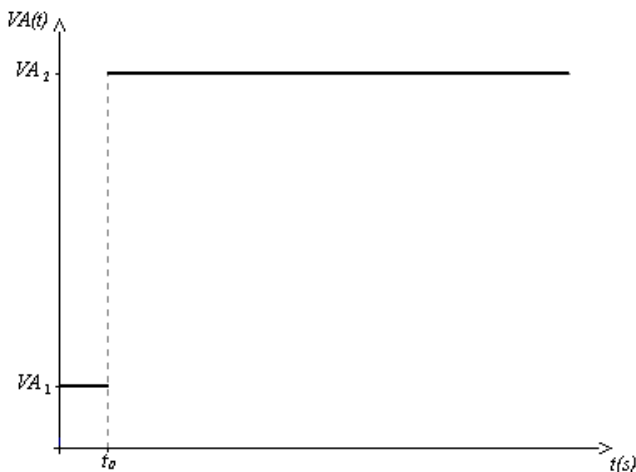


Figura 39: Degrau em VA

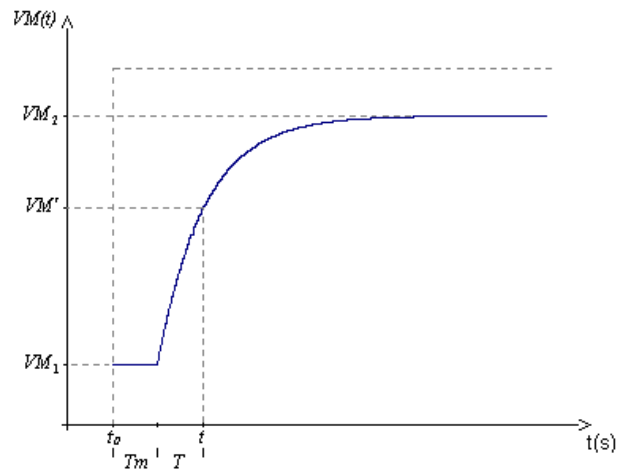


Figura 40: Resposta do sistema ao degrau

Através das figuras 10 e 11 pode-se obter todas as constantes necessárias para a determinação dos parâmetros do controlador. O ganho estático do processo é obtido através da razão entre a variação da variável medida e a variação da variável de atuação, ou seja:

$$K = \frac{VM_2 - VM_1}{VA_2 - VA_1}$$

O tempo morto, Tm, é o tempo entre o momento de aplicação do degrau em VA (t0) e o início da resposta do sistema.

A constante de tempo do sistema, T, é o tempo entre o início da reação do sistema e 63,212% do valor final de VM (VM'), isto é:

$$0,63212 = \frac{VM' - VM_1}{VM_2 - VM_1}$$

A partir das constantes do sistema, K, Tm e T, pode-se obter os parâmetros do controlador utilizando as fórmulas da tabela 4.

Tipo de Controlador	Constantes
Proporcional, Integral e Derivativo (PID)	$GP = \frac{T}{K * (Tr + Tm + dt/2)}$ $Ti = T$ $Td = Tm/2 + dt/4$

Tabela 9: Fórmulas para determinação dos parâmetros do controlador

## Ganhos X Escalas

É importante lembrar que o ganho proporcional somente executará sua ação de modo correto quando tanto a entrada como a saída do sistema utilizarem as mesmas escalas. Por exemplo, um controlador proporcional com ganho unitário e entrada (VM) utilizando a faixa de 0 a 1000 somente será realmente unitário se a faixa de saída (VA) também for de 0 a 1000.

Em muitos casos as escalas de entrada e saída são diferentes. Pode-se citar como exemplo um sistema onde o cartão de entrada analógica é de 4-20 mA, onde 4 mA corresponde ao valor 0, e 20 mA corresponde ao valor 30000. E o cartão de saída analógica é de 0V a 10 V, onde 0 V corresponde ao valor 0, e 10V corresponde ao valor 1000. Em casos como o deste exemplo, o ajuste de escalas pode ser feito através do ganho proporcional ao invés de uma normalização dos valores de entrada ou de saída.

Uma estratégia que pode ser adotada é, inicialmente, determinar o ganho em termos percentuais (independente de escalas), sem se preocupar com o tipo de módulos de entrada e saída analógicas utilizados. Posteriormente, após determinado este ganho, deve-se executar a correção de escalas, antes de introduzir o ganho proporcional no módulo F-PID16.056.

A estratégia consiste em determinar o ganho proporcional do sistema utilizando a faixa percentual (0% a 100%) tanto da variável medida (VM) como do valor de atuação (VA), sem levar em consideração os valores absolutos, tanto de VM como de VA.

Isto levará à determinação de um ganho proporcional denominado GP%. Este ganho GP% não pode ser utilizado diretamente na F-PID16.056. Antes é necessário fazer uma correção de escalas, que considere os valores absolutos destas variáveis.

**Atenção:**

Na seção anterior, *Sugestões para Ajustes do Controlador PID*, são sugeridos métodos de ajuste nos quais a correção de escalas é implícita ao método, não devendo ser considerada. No capítulo seguinte, *Exemplo de Aplicação*, a correção de escalas também é desnecessária, pois utilizou-se um dos métodos abordados na seção *Sugestões para Ajustes do Controlador PID*.

A correção de escalas é ilustrada a partir de um exemplo descrito a seguir.

Considere um sistema de ar condicionado onde o módulo de entrada analógica está lendo um resistor PTC (coeficiente térmico positivo) e o módulo de saída analógica gera uma tensão de 0 a 10V para atuar sobre a válvula responsável pela circulação da água que resfria o ar insuflado.

O módulo de entrada trabalha com uma faixa de 0 a 30000, porém a faixa útil é de 6634 a 8706 com o seguinte significado:

- $EA_0 = 6634 = 0\% = 884,6\Omega$  (corresponde a mínima temperatura que pode ser medida)
- $EA_1 = 8706 = 100\% = 1160,9\Omega$  (corresponde a máxima temperatura que pode ser medida)

O módulo de saída utiliza a mesma faixa de 0 a 30000 sem restrições e com o seguinte significado:

- $SA_0 = 0 = 0\% = 0V$  (corresponde a mínima vazão de água pela válvula)
- $SA_1 = 30000 = 100\% = 10V$  (corresponde a máxima vazão de água pela válvula)

Supondo que o ganho GP% foi previamente determinado, o ganho GP pode ser calculado pela seguinte equação:

$$GP = GP\% \cdot R$$

onde:

$$R = \frac{SA_1 - SA_0}{EA_1 - EA_0}$$

Para o exemplo anterior:

$$R = \frac{30000 - 0}{8706 - 6634} = 14,478$$

Esta razão  $R$  é uma constante que, quando multiplicada pelo ganho proporcional do controlador, compensa as diferenças entre as faixas de entrada e saída sem a necessidade de uma normalização direta.

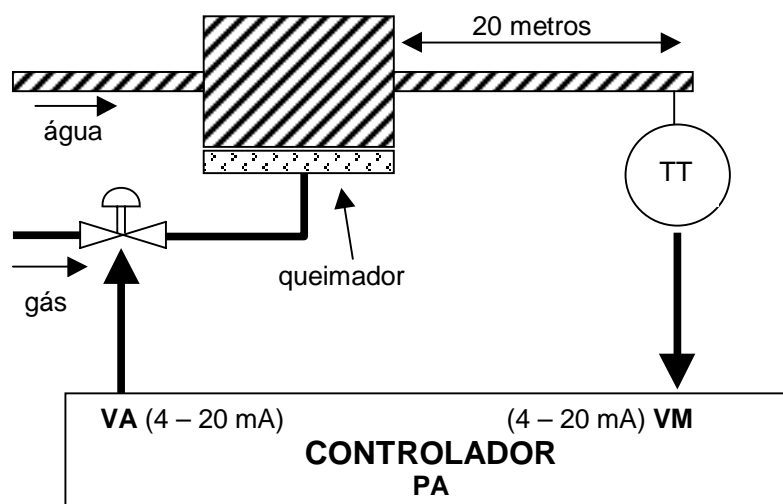
## Exemplo de Aplicação

Neste capítulo, será mostrado um exemplo prático de utilização do módulo F-PID16.056, abrangendo diversas fases do projeto do processo e do seu sistema de controle.

### Descrição do Processo

O processo exemplo tem como objetivo o fornecimento de água aquecida, com temperatura controlada, para um consumidor. O aquecimento será feito através de um queimador de gás, sendo controlado a partir da variação de vazão de gás através de uma válvula.

A figura 12 ilustra este processo.



**Figura 41: Processo de Aquecimento de Água**

Na figura, observa-se que o transmissor de temperatura (TT) fica perto do consumidor, que fica a 20 metros do ponto de aquecimento da água. Processos como este são bons exemplos de como podem ser introduzidos “tempos mortos”. Isto porque a água aquecida no ponto de aquecimento leva algum tempo para percorrer a distância até o ponto de medição junto do consumidor. Tempos mortos foram discutidos anteriormente (ver figura 11).

Algumas hipóteses foram assumidas no modelamento deste processo:

Assume-se que a água que chega ao ponto de aquecimento sobre o queimador tem temperatura fixa, de 30 °C.

Assume-se que a vazão de água é constante.

Assume-se que a pressão do gás é constante.

A seguir define-se algumas características deste processo e dos elementos utilizados:

A água aquecida deve ter sua temperatura programável entre 50 °C e 80 °C.

O transmissor de temperatura TT tem saída de 4 – 20 mA, e se comporta de forma linear, de tal maneira que 4 mA correspondem a 30 °C e 20 mA correspondem a 130 °C.

Assume-se que, para aumentar em 10 °C a temperatura da água, é necessário injetar 1 m<sup>3</sup>/h de gás. Este comportamento é linear.

A válvula de gás se fecha com 4 mA, injetando 0 m<sup>3</sup>/h de gás. Por outro lado, com 20 mA, ela injeta 8 m<sup>3</sup>/h de gás.

### Descrição dos Módulos Analógicos

Conforme pode ser visto na figura 12, necessita-se de uma saída analógica 4 – 20 mA, e de uma entrada analógica de 4 – 20 mA, como interfaces entre o controlador e o processo.

Internamente ao controlador, estas faixas de 4 – 20 mA correspondem a faixas numéricas em operandos M (VM e VA). Estas faixas de valores numéricos podem variar em função dos módulos de entrada e saída analógica selecionados. Neste exemplo, assume-se o seguinte:

entrada analógica VM (0 a 30000):

VM = 0 ---> 4 mA ---> 30 °C

VM = 30000 ---> 20 mA ---> 130 °C

saída analógica VA (0 a 10000):

VA = 0 ---> 4 mA = 0 ---> 0 m<sup>3</sup>/h

VA = 10000 ---> 20 mA ---> 8 m<sup>3</sup>/h

### Ponto de Ajuste

O operando PA deve ser utilizado para programar a temperatura desejada, entre 50 °C e 80 °C.

Como este operando deve ser comparado com VM, ele deve ter a mesma faixa numérica de VM, ou seja:

PA = 0 ---> 30 °C

PA = 30000 ---> 130 °C

Ou para restringir a faixa entre 50 °C e 80 °C:

PA = 6000 ---> 50 °C

PA = 15000 ---> 80 °C

### Blocodiagrama Geral e Valores Limites

A figura 13 mostra um blocodiagrama geral do sistema (controlador + processo), onde dentro do controlador mostra-se o módulo F-PID16.056. Observar que PA, VM e VA são operandos M.

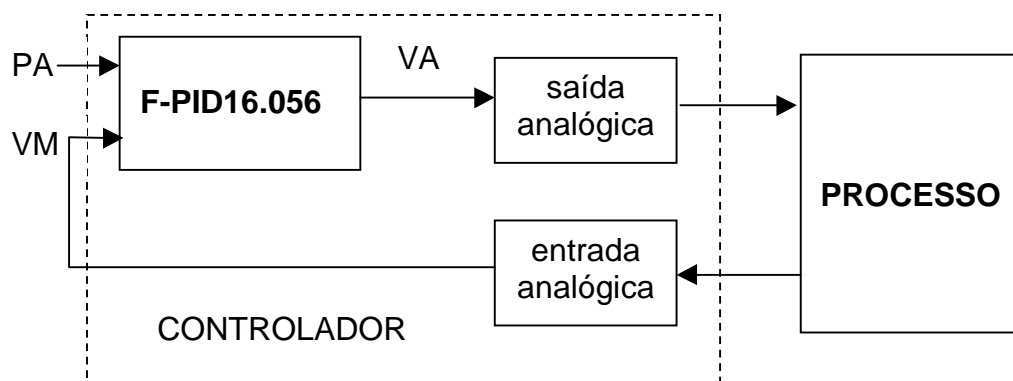


Figura 42: Blocodiagrama Geral

PA:

mínimo = 6000 (50 °C)

máximo = 15000 (80 °C)

VM:

mínimo = 0 (30 °C)

máximo = 30000 (130 °C)



VA:

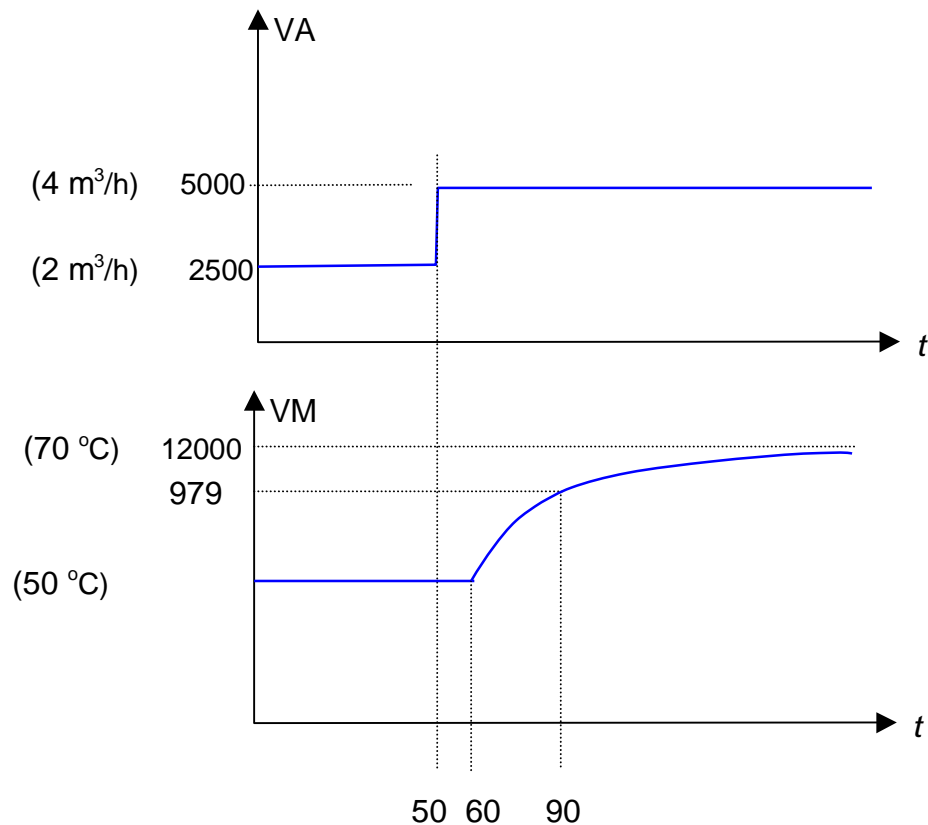
mínimo = 0 ( $0 \text{ m}^3/\text{h}$ )

máximo = 7500  $\rightarrow$  ( $6 \text{ m}^3/\text{h}$ )

Observa-se que no caso de VA, embora a válvula tenha capacidade de injetar  $8 \text{ m}^3/\text{h}$ , deseja-se limitar esta vazão em  $6 \text{ m}^3/\text{h}$ .

### Parâmetros do Processo

A figura 14 mostra o resultado de um teste de malha aberta sobre o processo. Para executar este teste, utilizou-se diretamente as variáveis VA e VM, com suas unidades internas.



**Figura 43: Teste de Malha Aberta**

A partir desta figura pode-se determinar os 3 parâmetros básicos, conforme explicado anteriormente no capítulo **Notas de Aplicação**.

$T_m = 10$  segundos (tempo morto, visto que o degrau foi aplicado em  $t = 50 \text{ s}$  e a resposta iniciou em  $t = 60 \text{ s}$ ).

$T = 30$  segundos (constante de tempo, visto que a resposta iniciou em  $t = 60 \text{ s}$ , e atingiu 63,21% da excursão em  $t = 90 \text{ s}$ :

$$9792 = 6000 + (12000 - 6000) * 0,6321.$$

$K = 2.4$  (ganho estático do processo)

$$2.4 = 12000 - 6000$$

$$5000 - 2500$$

### Sintonia do Controlador

Já que foi realizado o teste de malha aberta, será utilizado o segundo método de sintonia descrito no capítulo **Notas de Aplicação**.

Para utilizar este método, além dos parâmetros do processo determinados na seção anterior ( $T_m$ ,  $T$  e  $K$ ), também é necessário que o usuário informe outros 2 parâmetros:

$T_r$ , ou tempo de resposta desejado. Neste exemplo, será arbitrado em 10 segundos (um terço da constante de tempo em malha aberta).

$dt$ , ou tempo de ciclo da F-PID16.056. Conforme comentado anteriormente, este tempo deve ser 10 vezes menor do que a constante de tempo em malha aberta, ou ainda menor. Portanto, o valor deve ser menor que 3 segundos. Selecionou-se  $dt = 1$  segundo.

Agora, é possível aplicar as equações do método:

$$GP = T / (K * (T_r + T_m + Dt/2)) = 30 / (2.4 * (10 + 10 + 1/2)) = 0,609$$

$$Ti = T = 30 \text{ s/rep}$$

$$Td = T_m/2 + Dt/4 = 10/2 + 1/2 = 5.25 \text{ s}$$

### Utilização da F-PID16.056

A cada um segundo, deve-se executar a F-PID16.056, acionando sua entrada HABILITA durante uma única varredura.

A entrada AUTOMÁTICO/MANUAL pode ser controlada durante a operação do processo. Normalmente o processo estará em automático.

Para este processo, a entrada REVERSA/MANUAL deverá estar no estado 0 (reversa). O processo exige controle reverso pois, no caso de um aumento de VM, o controlador deve diminuir VA a fim de controlar o processo. Em outros termos, se a temperatura aumenta, deve-se fechar a válvula.

Operando TMXXXX:

$$\text{posição 0} = GP \times 10 = 6$$

$$\text{posição 1} = Ti \times 10 = 300$$

$$\text{posição 2} = Td \times 100 = 525$$

$$\text{posição 3} = dt \times 100 = 100$$

$$\text{posição 4} = \text{valor máximo da saída} = 7500$$

$$\text{posição 5} = \text{valor mínimo da saída} = 0$$

$$\text{posição 6} = \text{zona morta} = 0 \text{ (desabilitada)}$$

$$\text{posição 7} = \text{variação máxima permitida} = 0 \text{ (desabilitada)}$$

posições 8 a 29, inicializar com zeros somente na energização do CP

Operando AXXXX de controle: zerar todos os bits na inicialização.

### Comparação com o F-PID.033

O módulo F-PID16.056 foi desenvolvido visando melhorar a interface com o usuário, otimizar o tempo de execução e torná-lo compatível com variáveis de 16 bits ou menos resolução.

As principais alterações foram:

- Entradas e saídas com intervalo (“range”) de -30000 a 30000;
- Entrada de parâmetros sem pré-cálculo (entrada direta de  $G_p$ ,  $T_i$ ,  $T_d$  e  $dt$ );

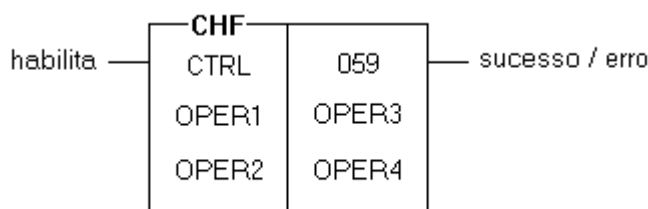
- Intervalo da amostragem (dt) de 10ms a 10s, enquanto que no F-PID.033 o limite mínimo é de 100ms.

Juntamente com estas alterações, um conjunto de novas características foi acrescentado ao F-PID anterior. A seguir a tabela 5 traz uma comparação das características entre o módulo F-PID16.056 e o F-PID.033.

CARACTERÍSTICA	F-PID.033	Novo PID
Parâmetros programados diretamente no formato ISA ( $G_p$ , $T_i$ , $T_d$ )		X
Ação derivativa calculada em função do erro ou da variável medida.		X
Ação derivativa calculada sobre 3 amostragens	X	X
Ação direta ou reversa	X	X
Intervalo de entrada e saída de -100% a +100%		X
Dead band		X
Dessaturação da ação integral ("anti-reset windup")	X	X
Entrada para <i>feedforward</i> / <i>bias</i> (deslocamento)	X	X
Inibição do termo derivativo	X	X
Inibição do termo integral	X	X
Inibição do termo proporcional		X
Limitação da taxa de crescimento		X
Limites de saída ajustáveis	X	X
Modo manual / automático	X	X
Acompanhamento da saída no modo manual e comutação manual/automática balanceada ("Output tracking" e "bumpless transfer").	X	X

**Tabela 10: Comparação entre o F-PID.033 e o F-PID16.056**

## F-CTRL.059 - Módulo F para Controle Avançado



### Introdução

O módulo função F-CTRL.059 implementa os algoritmos de controle avanço/atraso (lead/lag), retardo de primeira ordem e derivador com retardo de primeira ordem. Cada modo de operação (algoritmo) é selecionado através de um índice no módulo F-CTRL.059.

A partir de um valor de entrada, o módulo calcula o valor de saída em função do algoritmo selecionado. Todos os módulos utilizam duas constantes, uma constante de tempo  $T$  e uma segunda constante  $K$  cuja função varia conforme o algoritmo selecionado. Os algoritmos são executados de modo discreto, necessitando que o tempo de disparo da função seja declarado juntamente com os parâmetros.

Estas funções são utilizadas em algoritmos de controle avançado para otimização da malha de controle. São geralmente utilizadas em conjunto com uma função PID.

### Retardo de 1ª Ordem

Quando selecionado o algoritmo de retardo de 1ª ordem, o módulo F aplica sobre o valor do operando de entrada ( $V_i$ ) um retardo temporal de 1ª ordem. O valor de saída ( $V_o$ ) desta função é proporcional à entrada, porém, retardado conforme uma função exponencial.

Este algoritmo necessita de duas constantes. Uma constante de tempo  $T$  que, numa analogia com um circuito RC, representa sua constante de tempo de carga (63,212% do valor final) e uma constante de ganho proporcional  $K$ .

No domínio frequência ( $s$ ), o retardo temporal de 1ª ordem efetua a seguinte função de transferência:

$$V_o(s) = \frac{K}{1 + T \cdot s} \times V_i(s)$$

Onde  $V_i(s)$  e  $V_o(s)$  são as transformadas de Laplace dos sinais de entrada e saída.

Sua resposta ao degrau é representada pela figura 4-16, onde pode ser observada a sua constante de tempo  $T$  associada ao valor  $V'$ , que representa 63,212% da diferença entre o valor inicial e o valor final.

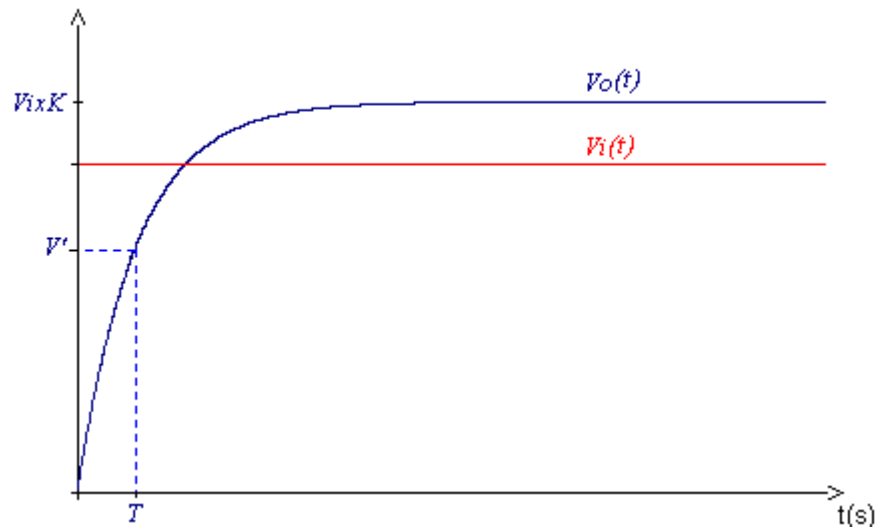


Figura 4-16 Retardo Temporal de 1ª Ordem

### Derivador com Retardo de 1ª Ordem

Quando este algoritmo é selecionado, o módulo F aplica sobre o valor do operando de entrada a sua derivada juntamente com um retardo temporal de 1ª ordem. O valor de saída é a derivada da entrada  $V_i$  com amortecimento retardado conforme uma função exponencial.

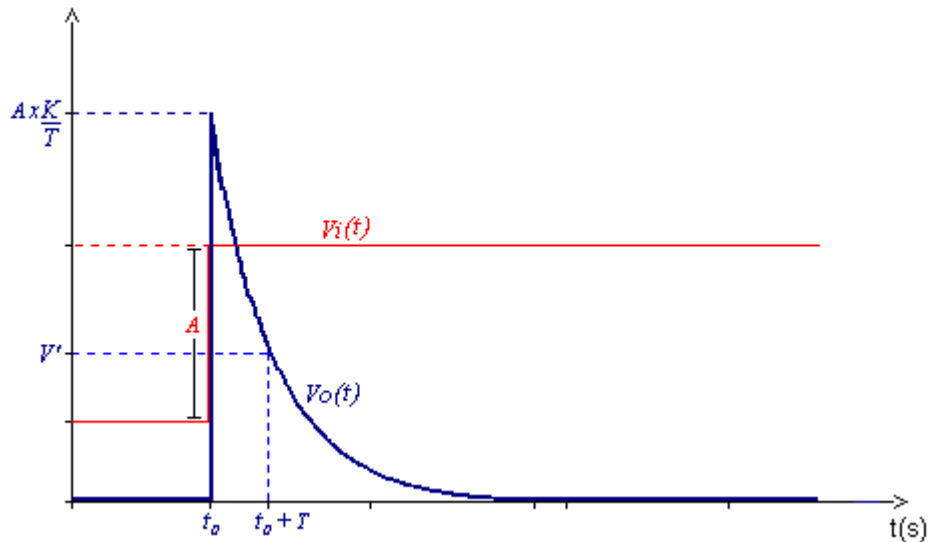
Este algoritmo necessita de duas constantes. Uma constante de tempo  $T$  que pode ser entendida, na mesma analogia do retardo temporal de primeira ordem, como a constante de tempo de descarga de um circuito RC. A segunda constante é a constante derivativa  $K$  que, dividida pela constante  $T$ , determinará uma outra constante que pode ser tratada como um ganho.

No domínio frequência (s), o derivador com retardo temporal de 1ª ordem efetua a seguinte função de transferência:

$$V_o(s) = \frac{K \cdot s}{1 + T \cdot s} \times V_i(s)$$

Onde  $V_i(s)$  e  $V_o(s)$  são as transformadas de Laplace dos sinais de entrada e saída.

Sua resposta ao degrau é representada pela figura 4-17. No instante inicial ( $t = t_0$ ) observa-se que a saída desta função ( $V_o$ ) é o degrau de entrada, com amplitude  $A$ , multiplicada pela razão  $K/T$ . No instante  $t = t_0 + T$ , a saída do sistema equivale a  $V'$ , ou seja, 36,788% de  $A \times K/T$ . Quando a entrada  $V_i$  fica constante, a saída desta função retorna a zero com um retardo temporal de 1ª ordem.



**Figura 4-17 Derivador com Retardo Temporal de 1ª Ordem**

É importante lembrar que o degrau não é visto pelo módulo F-CTRL.059 como uma variação instantânea, mas sim como uma variação entre duas amostragens. Caso contrário sua derivada teria valor infinito.

#### **Avanço/ Atraso (lead/lag)**

Quando este algoritmo é selecionado, o módulo F aplica sobre o valor do operando de entrada o avanço ou o atraso conforme a relação entre as constantes declaradas.

O algoritmo necessita de duas constantes. Uma constante de tempo  $T$  que, da mesma forma que os algoritmos anteriores, pode ser entendida como a constante de tempo de um circuito RC. E uma constante  $K$  que, juntamente com a constante  $T$ , definirá o comportamento do algoritmo como avanço ou atraso.

Sempre que a constante de tempo  $T$  for maior que a constante  $K$ , o algoritmo se comportará como atraso. Quando  $K$  for maior que  $T$  seu comportamento será de avanço. As constantes  $K$  e  $T$  são também conhecidas como constantes de avanço e atraso, respectivamente.

No domínio frequência (s), o avanço/atraso efetua a seguinte função de transferência:

$$Vo(s) = \frac{1 + K \cdot s}{1 + T \cdot s} \times Vi(s)$$

Onde  $Vi(s)$  e  $Vo(s)$  são as transformadas de Laplace dos sinais de entrada e saída.

A resposta ao degrau do avanço é representada pela figura 4-18. No instante  $t = t_0$  observa-se que a saída desta função  $Vo(t_0)$  equivale a  $V''$  que pode ser descrita como

$$V'' = Vi(t) + A \times \frac{K}{T}, \text{ para } t < t_0,$$

ou seja, o valor da entrada antes do salto somado ao produto da amplitude do degrau aplicado na entrada ( $A$ ) pela razão  $K/T$ . No instante  $t = t_0 + T$ , a saída do sistema equivale a  $V'$ , ou seja, 36,788% da diferença entre o valor máximo ( $V''$ ) e o valor de  $Vi(t)$  para  $t > t_0$  somado a um deslocamento igual a  $Vi(t_0)$ .

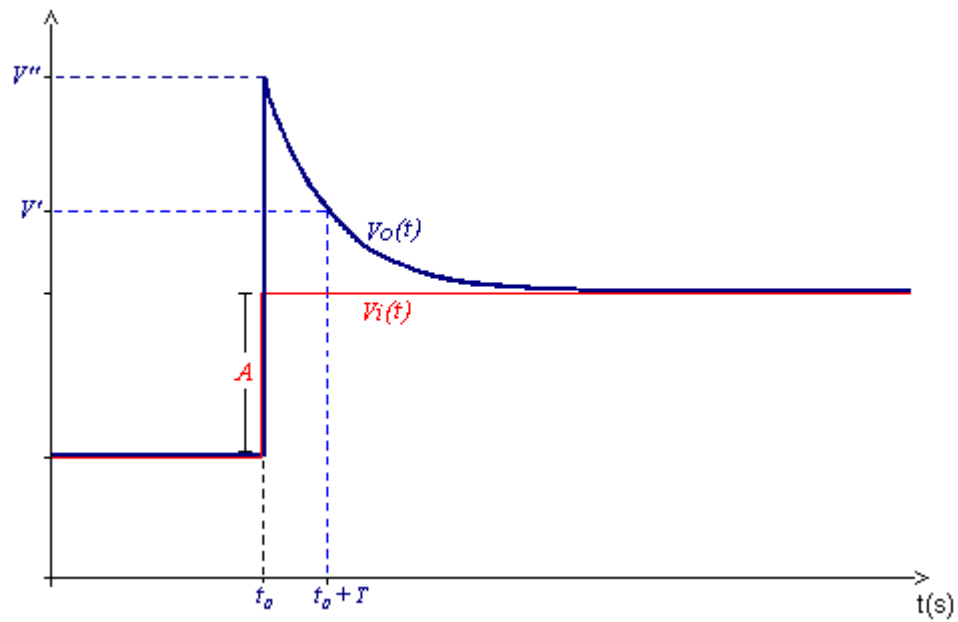


Figura 4-18 Avanço

A resposta ao degrau do atraso é representada pela figura 4-19. No instante  $t = t_0$  observa-se que a saída desta função  $V_o(t_0)$  equivale a  $V''$  que, da mesma forma que o avanço, pode ser descrita como:

$$V'' = V_i(t) + A \times \frac{K}{T}, \text{ para } t < t_0,$$

diferindo-se do gráfico do avanço porque  $K$  é menor que  $T$ . No instante  $t = t_0 + T$ , a saída do sistema equivale a  $V'$ , ou seja, 63,212% da diferença entre o valor de  $V_i(t)$ , para  $t > t_0$ , e o valor de  $V''$ , somado a um deslocamento igual a  $V_i(t_0)$ .

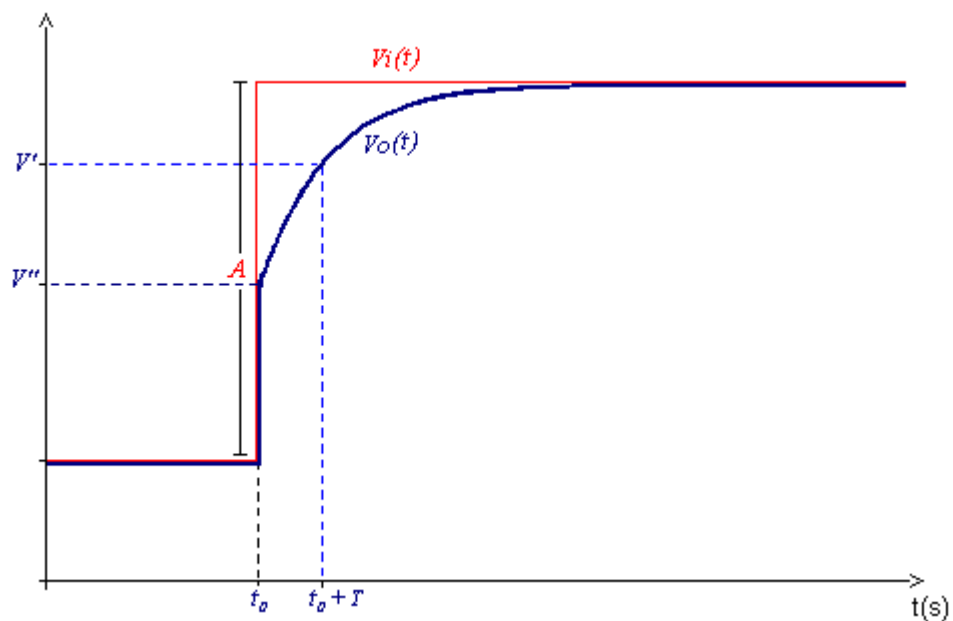


Figura 4-19 Atraso

## Programação

### Operandos

As células da instrução CHF utilizada para a chamada da função são programadas do seguinte modo:

- **OPER1** - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deverá ser obrigatoriamente uma constante memória com valor 8 (%KM+00008).
- **OPER2** - Especifica o número de parâmetros que são passados para a função em OPER4. Este operando deverá ser obrigatoriamente uma constante memória com valor 0 (%KM+00000).
- **OPER3** - Contém os parâmetros que são passados para a função, declarados quando a instrução CHF for editada. O número de parâmetros editáveis é especificado em OPER1, sendo fixo em 8 para este módulo:
  - %KMXXXX - Constante memória que indica qual o algoritmo será utilizado, pode assumir os seguintes valores:
    - %KM0000 – retardo temporal de primeira ordem;
    - %KM0001 – derivativo com retardo temporal de primeira ordem;
    - %KM0002 – avanço/atraso (lead/lag) .
  - % KMXXXX - Constante com o valor do intervalo de amostragem . Assume valores de 0,01 a 10s, devendo ser multiplicado por 100 para ser declarado neste campo.
  - %MXXXX - Memória com o valor da constante de tempo  $T$ . Assume valores de 0,01 a 320s, devendo ser multiplicado por 100 para ser declarado neste campo.
  - %MXXXX - Memória com o valor da constante  $K$ . Assume valores de 0,01 a 320, devendo ser multiplicado por 100 para ser declarado neste campo.
  - %MXXXX - Memória com o valor de entrada num intervalo de -32768 a +32767.
  - %MXXXX - Memória com o valor de saída num intervalo de -30000 a +30000.
  - %MXXXX - Uso interno. Não deve ser alterado.
  - %MXXXX - Uso interno. Não deve ser alterado.

- **OPER4** - Não utilizado.

### Entradas e Saídas

Descrição das entradas:

- habilita - quando esta entrada está energizada a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso o número de parâmetros ou seu tipo sejam diferentes das necessidades da função, haverá a desergenização da saída sucesso/erro. Se estiverem corretos, o cálculo do algoritmo selecionado é realizado.

Descrição das saídas:



- sucesso(1) /erro (0) - É energizada quando a função foi corretamente executada. Não é energizada caso ocorram erros na especificação dos operandos, tentativa de acesso a operandos não declarados ou parâmetros inválidos.

**Características de Funcionamento**

A cada intervalo de amostragem o valor de entrada da função é aplicado sobre o algoritmo e a saída da função atualizada.

Observa-se que o algoritmo é aplicado de forma discreta de modo que o tempo de amostragem (dt) deve ser da ordem de 10 vezes menor que a constante de tempo T para se obter um resultado satisfatório. O intervalo entre amostragens de um laço do módulo F-CTRL.059 pode variar de 0,01 a 10 segundos. É de responsabilidade do usuário programar um “disparador” da função, ou seja, um trecho de programa aplicativo que somente habilite o módulo F nos intervalos de tempo desejados. Aconselha-se utilizar um módulo E018, este módulo é executado dentro de um intervalo de tempo fixo que pode ser utilizado para gerar uma ou mais bases de tempo para a execução de um ou mais laços do F-CTRL.059. Nota-se ainda que o valor do intervalo de amostragem declarado nos parâmetros deve coincidir com o intervalo de tempo das chamadas do "disparador".

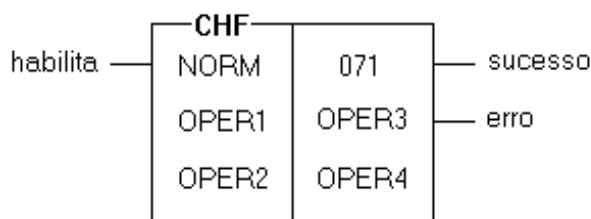
É importante lembrar que a atualização das entradas e saídas ocorrem na mesma ordem de tempo de um ciclo do CP. Sempre que o tempo de ciclo do CP for maior que o tempo de amostragem aconselha-se o uso do módulo função F-AES.087 para a Série Ponto.

**Tempos de Execução**

- Retardo temporal de primeira ordem: 298  $\mu$ s
- Derivativo com retardo temporal de primeira ordem: 338  $\mu$ s
- Avanço / Atraso (lead/lag): 338  $\mu$ s

Estes tempos são válidos para os CPs AL-2003, AL-2004, PO3145, PO3142 e PO3242.

## F-NORM.071 - Função para Normalização



### Introdução

A função F-NORM.071 normaliza operandos inteiros, implementando o seguinte algoritmo de normalização:

$$M(saída) = \frac{(M(entrada) - A) \cdot (C)}{B - A}$$

Onde A, B e C são constantes.

### Programação

- OPER1 - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deverá ser obrigatoriamente uma constante memória com valor 6 (%KM+00006).
- OPER2 - Deve ser um operando do tipo constante memória com valor 0 (%KM+00000). Determina o número de parâmetros possíveis de serem programados na janela de edição de OPER4. Como esta função não necessita de nenhum parâmetro em OPER4, o valor de OPER2 é 0.
- OPER3 - Contém os parâmetros que são passados para a função, declarados através de uma janela visualizada no MasterTool quando a instrução CHF for editada. O número de parâmetros editáveis é especificado em OPER1, sendo 6 para esta chamada:
  - %KMXXXXXX – número de operandos;
  - %MXXXXXX – primeiro operando de entrada;
  - %MXXXXXX – primeiro operando de saída;
  - %KMXXXXXX – início da faixa de entrada (A);
  - %KMXXXXXX – fim da faixa de entrada (B);
  - %KMXXXXXX – fim da faixa de saída (C);

A partir da versão 1.10 do F-NORM.071, disponível para os CPs AL-2003, AL-2004 e Série Ponto, pode-se utilizar um sétimo parâmetro (D), que define o início da faixa de entrada. Quando forem declarados 6 parâmetros na CHF da F-NORM.071 versão 1.10, será admitido que o sétimo parâmetro (D) é igual a zero e executado o mesmo algoritmo de normalização.

O algoritmo de normalização passa a ser escrito como:

$$M(saída) = \frac{(M(entrada) - A) \cdot (C - D)}{B - A} + D$$

Depois de declarar o valor 7 (%KM+00007) para o OPER1, os parâmetros passados através de OPER3 são os seguintes:

- %KMXXXXXX – número de operandos;
- %MXXXXXX – primeiro operando de entrada;
- %MXXXXXX – primeiro operando de saída;
- %KMXXXXXX – início da faixa de entrada (A);
- %KMXXXXXX – fim da faixa de entrada (B);
- %KMXXXXXX – fim da faixa de saída (C);
- %KMXXXXXX – início da faixa de saída (D) (somente na versão 1.10).

### Operação

A F-NORM.071 implementa o seguinte cálculo:

$$M[\text{saída}] = (M[\text{entrada}] - A) * C / (B - A)$$

sendo:

- M[entrada] - faixa de operandos inteiros de entrada
- M[saída] - faixa de operandos inteiros de saída
- A - offset para a entrada
- B - valor referência da entrada para normalizar
- C - valor normalizado da saída correspondente à B

A saída é a normalização da entrada e de modo que para um dado de entrada com o valor A a saída correspondente é 0, e para um valor de entrada B a saída

correspondente será C. Fora dessa faixa, o valor da saída será proporcional ao da entrada, conforme a fórmula dada.

A função trabalha com faixa de até 127 operandos (1 a 127)

### Entradas e Saídas

Descrição das entradas da função:

- **habilita** - quando esta entrada está energizada a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso os mesmos estejam incorretos, a saída erro da instrução é energizada, e as demais ficam desenergizadas. Se os parâmetros estiverem corretos, somente a saída sucesso é energizada.

Descrição das saídas da função:

- **sucesso** - indica que os parâmetros da chamada estão corretos e que a função foi corretamente executada. Vide observação.

- **erro** - é ligada caso ocorra erro nos parâmetros da chamada. Vide observação.

OBS: quando as duas saídas (sucesso e erro) estiverem energizadas é porque o intervalo de operandos M de entrada é o mesmo que o intervalo de operandos M de saída (nos parâmetros).

### Utilização

Esta função pode ser utilizada nas UCPs AL-600, AL-2000/MSP, AL-2002, AL-2003, AL-2004, PL104, PL105, PL106, QK800, QK801, QK2000 e Série Ponto.

### Exemplo

Seja a entrada um valor inteiro proveniente de uma instrução A/D, com variação de 0 a 4095. Deseja-se normalizar a saída para 0 a 100, correspondendo à valores de entrada entre 800 a 4000. Teremos:

$$A = 800$$

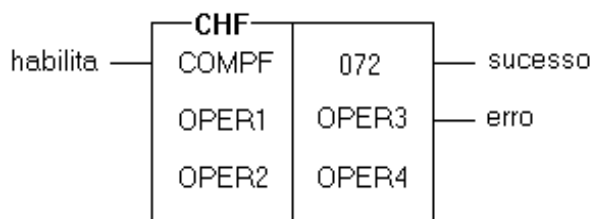
$$B = 4000$$

$$C = 100$$

Resultados:

Entrada	Saída
0	-25
800	0
2400	50
4000	100
4095	102

## F-COMPF.072 - Função para Múltiplas Comparações



### Introdução

A função F-COMPF.072 divide um operando em faixas especificadas, apresentando saída em forma binária, onde o bit ligado indica que o operando pertence à faixa respectiva.

### Programação

As células da instrução CHF utilizada para a chamada são programadas do seguinte modo:

- **OPER1** - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deverá ser obrigatoriamente uma constante memória com valor 4 (%KM+00004).
- **OPER2** - Deve ser um operando do tipo constante memória com valor 0 (%KM+00000). Determina o número de parâmetros possíveis de serem programados na janela de edição de OPER4. Como esta função não necessita de nenhum parâmetro em OPER4, o valor de OPER2 é 0.
- **OPER3** - Contém os parâmetros que são passados para a função, declarados através de uma janela visualizada no MasterTool quando a instrução CHF for editada. O número de parâmetros editáveis é especificado em OPER1, sendo 4 para esta chamada:
  - %KM+XXXX - Número de operandos %MXXXX a examinar
  - %MXXXX - Operando de entrada inicial para a comparação
  - %MXXXX - Operando de saída inicial para os bits indicadores
  - %TMXXXX - Tabela que especifica até 16 faixas de valores para qualificar os operandos de entrada (ver formato a seguir)

Posição Tabela	Conteúdo
0	Reservada
1	Reservada
2	Início faixa 0
3	Fim da faixa 0
4-31	<continuam as definições de faixas>
32	Início faixa 15
33	Fim da faixa 15

Tabela 4-11 Definição das faixas

A tabela deve ter tamanho mínimo de 4 posições (1 faixa). Para otimizar o tempo de execução da função, recomenda-se que a tabela seja definida com o tamanho exato para conter as definições de faixas necessárias.

- **OPER4** - Não utilizado.

### Operação

O início e o fim de cada faixa de comparação são especificados como números inteiros.

O operando é considerado na faixa se cumprir:

(início da faixa)  $\leq$  %MXXXX  $<$  (fim da faixa)

A cada faixa na tabela %TMXXXX corresponde um bit no operando %MXXXX, sendo que o bit 0 do operando de saída corresponde a faixa 0 e assim sucessivamente. Os bits correspondentes às faixas não definidas são sempre 0. As faixas podem ser sobrepostas.

O número de operandos a processar é dado pelo primeiro parâmetro (%KM+XXXX), podendo ser definido de 1 a 127.

### Entradas e Saídas

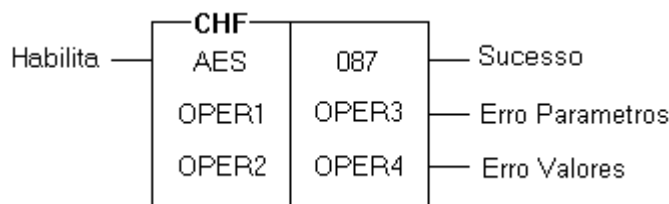
Descrição das entradas da função:

- **habilita** - quando esta entrada está energizada a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso os mesmos estejam incorretos, a saída **erro** da instrução é energizada, e as demais ficam desenergizadas. Se os parâmetros estiverem corretos, somente a saída sucesso é energizada.

Descrição das saídas da função:

- **sucesso** - indica que os parâmetros da chamada estão corretos e que a função foi corretamente executada.
- **erro** - é ligada caso ocorra erro nos parâmetros da chamada

## F-AES.087 - Função Para Atualização Imediata de Entradas e Saídas



### Introdução

Esta instrução executa uma atualização imediata na memória imagem e nos módulos das posições físicas especificadas. Sua atuação é idêntica à varredura dos pontos de E/S realizada pelo programa executivo ao final de cada varredura, porém com o número de posições limitados.

Caso a UCP seja uma PO3242 ou uma PO3342 esta função permite atualizar também os dispositivos da rede PROFIBUS.

### Programação

As células da instrução CHF utilizada para a chamada são programadas do seguinte modo:

- **OPER1** - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deverá ser obrigatoriamente uma constante memória com valor 2 (%KM+00002).
- **OPER2** - Deve ser um operando do tipo constante memória com valor 0 (%KM+00000).
- **OPER3** - Contém os parâmetros que são passados para a função, declarados quando a instrução CHF for editada. O número de parâmetros editáveis é especificado em OPER1, sendo fixo em 2 para este módulo F:
  - MXXXX, KMXXXX Especificação da posição física inicial do barramento a ser atualizada. O módulo existente na posição aqui declarada terá seu valor atualizado do operando correspondente para as saídas, no caso de um módulo de saída, ou então as entradas serão lidas para o operando correspondente, no caso de um módulo de entrada. O valor da posição física deve ser de 0 a 39.

Rede PROFIBUS: a posição deve ser a de um módulo PO4053. Se a rede for redundante, deve-se dar apenas a posição de um dos módulos para que toda a rede seja atualizada.

  - MXXXX, KMXXXX - Especificação de quantas posições do barramento serão atualizadas, incluindo a posição inicial, e a partir dela. Por exemplo, se for especificado o valor 2, serão atualizadas a posição declarada como a posição inicial, e a posição seguinte à ela. O valor deste parâmetro deve ser de 1 a 10.
- **OPER4** - Não utilizado.

### Entradas e Saídas

Descrição das entradas da função:

- **Habilita** - quando esta entrada é energizada, a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso estejam todos corretos, a função é executada e as posições são atualizadas. Caso haja algum parâmetro incorreto, as saídas são acionadas indicando o erro e a atualização não é realizada.



Descrição das saídas da função:

- **Sucesso** - Esta saída é acionada quando a função foi corretamente executada e as posições do barramento atualizadas.
- **Erro Parâmetros**: esta saída é acionada quando algum operando não declarado é passado como parâmetro. Quando isto acontece, nenhuma posição é atualizada.
- **Erro Valores**: esta saída é acionada quando algum dos parâmetros contém um valor inválido. Isto acontece se o valor está fora da faixa permitida para esta função, ou se não existe nenhum módulo declarado no módulo C para a posição que deve ser atualizada.

Se nenhuma das três saídas for energizada, significa que a função não pôde ser executada por um intertravamento interno do CP. Isso pode acontecer, por exemplo, se a função estiver dentro de um módulo E018, e o E018 for executado no mesmo instante em que um módulo C é carregado no CP.

### Tempo de execução

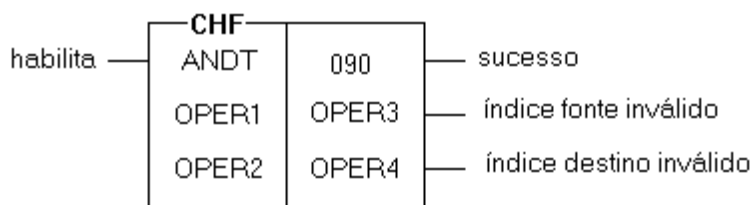
Caso seja utilizada para atualizar a rede PROFIBUS o tempo de atualização da rede deve ser considerado (consultar o manual de utilização do PO4053 MU209903).

### Redundância

Caso a posição do módulo referido seja de uma PO4053 de um par redundante, basta apenas uma chamada de F-AES para que as duas PO4053 da redundância sejam atualizadas.

**O uso da função F-AES.087 com forçamento de pontos de E/S ao mesmo tempo gera variações no pontos de E/S físicos. Esta situação deve ser evitada.**

## F-ANDT.090, F-ORT.091 e F-XORT.092 - Funções de Operações Lógicas entre Operandos Tabela



### Introdução

As funções F-ANDT.090, F-ORT.091 e F-XORT.092 permitem realizar operações lógicas AND (e), OR (ou) ou XOR (ou exclusivo), respectivamente, entre operandos simples (M ou D) e/ou tabelas (TM ou TD). Podem ser realizadas até 255 operações lógicas em uma única chamada da função. É necessário que os três operandos (fonte1, fonte2 e destino) sejam do mesmo tipo (memória ou decimal).

### Programação

As células da instrução CHF utilizada para a chamada são programadas do seguinte modo:

- **OPER1** - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deverá ser obrigatoriamente uma constante memória com valor 4 (KM+00004).
- **OPER2** - Deve ser um operando do tipo constante memória com valor 0 (KM+00000). Determina o número de parâmetros possíveis de serem programados na janela de edição de OPER4. Como estas funções não necessitam de nenhum parâmetro em OPER4, o valor de OPER2 é 0.
- **OPER3** - Contém os parâmetros que são passados para a função, declarados através de uma janela visualizada no AL-3830 quando a instrução CHF for editada. O número de parâmetros editáveis é especificado em OPER1, sendo fixo em 4 para estes módulos:
  - MXXXX, DXXXX, TMXXXX ou TDXXXX - Operando simples ou tabela cujo valor será utilizado para realização da lógica (operando fonte 1).
  - MXXXX, DXXXX, TMXXXX ou TDXXXX - Operando simples ou tabela cujo valor será utilizado para realização da lógica (operando fonte 2).
  - MXXXX, DXXXX, TMXXXX ou TDXXXX - Operando simples ou tabela onde o valor resultante da lógica será armazenado (operando destino).
  - KMXXXX - Número de operandos simples ou posições da tabela com as quais se fará a operação lógica.
- **OPER4** - Não utilizado.

**Entradas e Saídas**

Descrição das entradas da função:

- **habilita** - quando esta entrada está energizada a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso os mesmos estejam incorretos, são acionadas as saídas de índice inválido.

Descrição das saídas da função:

- **sucesso** - indica que a movimentação foi corretamente realizada.
- **índice fonte inválido** - indica que houve erro na especificação do operando fonte:
  - o operando não está declarado no módulo C
  - não existem posições suficientes para realizar a lógica
- **índice destino inválido** - indica que houve erro na especificação do operando destino:
  - o operando não está declarado no módulo C
  - não existem posições suficientes para realizar a lógica

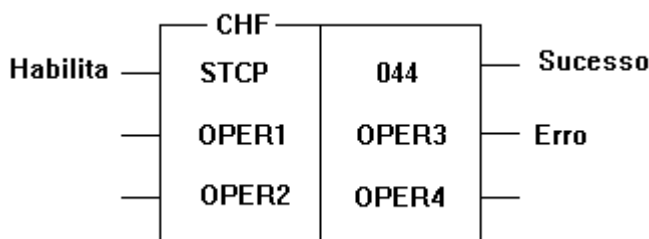
Caso as duas saídas de índice inválido sejam acionadas simultaneamente, ocorreu algum dos seguintes erros:

- o número de parâmetros programado em OPER1 é diferente de quatro
- o tipo de um ou mais parâmetros em OPER4 não é válido
- os parâmetros em OPER4 são de tipos diferentes (memória e decimal)
- o número total de posições a serem transferidas é maior que 255

**ATENÇÃO:**

Estas funções permitem executar operações lógicas de um grande número de operandos em uma única varredura. Deve-se utilizá-las com cuidado para que o tempo máximo de ciclo do programa não seja excedido.

## F-STCP.044 - Função Status do UCP



### Introdução

A função F-STCP.044 retorna o status da UCP em um bloco de operandos M ou em uma TM. Este status corresponde aos mesmos parâmetros que são respondidos no comando 37 do protocolo ALNETI.

### Programação

As células da instrução CHF utilizada para a chamada da função são programadas do seguinte modo:

- **OPER1** - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deve ser obrigatoriamente uma constante memória com valor 1 (KM+00001).
- **OPER2** - Deve ser um operando do tipo constante memória com valor 0 (KM+0000). Este operando define o número de parâmetros possíveis de serem programados na janela de edição de OPER4. Como esta função não necessita de nenhum parâmetro em OPER4, o valor de OPER2 é 0.
- **OPER3** - Contém os parâmetros que são passados para a função, declarados através de uma janela visualizada no MasterTool, AL-3830 ou AL-3832 quando a instrução CHF é editada. O número de parâmetros editáveis é especificado em OPER1, sendo fixo em 1 para este módulo:

TMXXXX ou MXXXX: Tabela ou bloco de operandos M onde será escrito o status que foi lido do CP. A tabela deve conter 50 posições. Em caso de operandos M a função escreve no operando declarado e nos 49 subsequentes, sendo que estes devem estar declarados no módulo C do projeto.

- **OPER4** - Não utilizado.

### Entradas e Saídas

Descrição das Entradas:

- **habilita** - quando esta entrada está energizada a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso o número de parâmetros ou seu tipo sejam diferentes das necessidades da função, haverá a energização da saída de indicação de erro. também é consistido se existe um número mínimo de operandos declarados após o indicado na função. Se estes estiverem corretos, busca os parâmetros de status e copia para os operandos.

Descrição das Saídas:

- **sucesso** - é energizada quando a função foi corretamente executada.
- **erro** - é energizada caso ocorra erro na especificação dos operandos ou tentativa de acesso a operandos não declarados.

### Descrição dos Operandos de Status

Segue descrição dos operandos e o índice no qual se encontram dentro da tabela da função. Em caso de operando M, o operando declarado representa o índice 00.

Operando	Identificação	Descrição
00	Identificação do Modelo de UCP	00H - AL-3003 01H - AL-3004 20H - AL-2000 21H - AL-2002 22H - QK2000 23H - AL-2003 24H - AL-2004 40H - AL-600 41H - QK600 50H - QK800 51H - QK801 A0H - PL101 A1H - PL102 A2H - PL103 A3H - PL104 A4H - PL105 A5H - PL106 A6H - PL107 B0H - GR310 B1H - GR316 B2H - GR330 B3H - GR350 B4H - GR351 B6H - GR370 B7H - GR371 C0H - PO3042 C1H - PO3142 C2H - PO3242 C3H - PO3342 C6H - PO3045 C7H - PO3145 C8H - PO3245 C9H - PO3345
01	Caracter 0 de identificação auxiliar da UCP	String de identificação auxiliar da UCP, podendo ter até 8 caracteres, no formato ASCII
02	Caracter 1 de identificação auxiliar da UCP	
03	Caracter 2 de identificação auxiliar da UCP	
04	Caracter 3 de identificação auxiliar da UCP	
05	Caracter 4 de identificação auxiliar da UCP	
06	Caracter 5 de identificação auxiliar da UCP	
07	Caracter 6 de identificação auxiliar da UCP	
08	Caracter 7 de identificação auxiliar da UCP	
09	Versão do Executivo (Parte Alta)	Formato V.RC, onde V é o número da Versão, R é o número da Revisão e o C é o número da última Correção. Parte alta armazena o valor V e a parte baixa armazena R e C nos nibles 1 e 0, respectivamente.
10	Versão do Executivo (Parte Baixa)	

11	Modo de Operação 1 do CP	<div> <div>F E D C B A 9 8</div> <div>exe prg cic tst cop for cpt sai</div> </div> <p>           exe: CP em modo execução            prg: CP em modo programação            cic: CP em modo ciclado            tst: CP em modo teste            cop: copiando módulo de EPROM para RAM            for: há forçamento(s) de relé(s)            cpt: compactando RAM            sai: saídas digitais desabilitadas         </p> <div> <div>7 6 5 4 3 2 1 0</div> <div>trc apg prt</div> </div> <p>           apg: apagando flash EPROM            prt: nível de proteção do CP (valor de 0 - sem proteção - a 3 - proteção máxima)            trc: troca de módulos de E/S com o CP energizado         </p>
12	Código de Mensagem 1	
13	Código de Mensagem 2	
14	Código de Mensagem 3	
15	Código de Mensagem 4	
16	Tempo de ciclo instantâneo	Em ms
17	Tempo de ciclo médio	Em ms
18	Tempo de ciclo máximo	Em ms
19	Tempo de ciclo mínimo	Em ms
20	Período E018	00H - 50ms 01H - 25ms 02H - 10ms 03H - 5ms 04H - 3,125ms 05H - 2,5ms 06H - 1,25ms 07H - 0,625ms FFH - Sem E018
21	Operando Reservado	
22	Tempo máximo da varredura de programa	00 - 100ms 01 - 200ms 02 - 300ms 03 - 400ms 04 - 500ms 05 - 600ms 06 - 700ms 07 - 800ms

23	Estado da RAM do programa aplicativo	<div>Informações gerais do estado da RAM de programa aplicativo e indicador dos bancos de RAM do programa aplicativo existentes:</div> <div><div>F E D C B A 9 8</div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div>cpt</div></div></div> <div>cpt: RAM compactada (0) ou não (1)</div> <div><div>7 6 5 4 3 2 1 0</div><div><div>B7</div><div>B6</div><div>B5</div><div>B4</div><div>B3</div><div>B2</div><div>B1</div><div>B0</div></div></div> <div>bits 7-0: bancos existentes</div>
24	Espaço livre em bytes da RAM programa aplicativo banco 0	
25	Espaço livre em bytes da RAM programa aplicativo banco 1	
26	Espaço livre em bytes da RAM programa aplicativo banco 2	
27	Espaço livre em bytes da RAM programa aplicativo banco 3	
28	Espaço livre em bytes da RAM programa aplicativo banco 4	
29	Espaço livre em bytes da RAM programa aplicativo banco 5	
30	Espaço livre em bytes da RAM programa aplicativo banco 6	
31	Espaço livre em bytes da RAM programa aplicativo banco 7	
32	Estado da EPROM do programa aplicativo	<div>Indicador dos bancos de EPROM do programa aplicativo existentes:</div> <div><div>F E D C B A 9 8</div><div><div>B15</div><div>B14</div><div>B13</div><div>B12</div><div>B11</div><div>B10</div><div>B9</div><div>B8</div></div></div> <div><div>7 6 5 4 3 2 1 0</div><div><div>B7</div><div>B6</div><div>B5</div><div>B4</div><div>B3</div><div>B2</div><div>B1</div><div>B0</div></div></div> <div>bits F-0: bancos existentes</div>
33	Espaço livre em bytes da EPROM programa aplicativo banco 0	
34	Espaço livre em bytes da EPROM programa aplicativo banco 1	
35	Espaço livre em bytes da EPROM programa aplicativo banco 2	
36	Espaço livre em bytes da EPROM programa aplicativo banco 3	

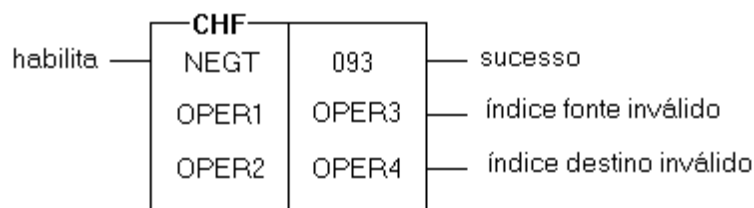
37	Espaço livre em bytes da EPROM programa aplicativo banco 4	
38	Espaço livre em bytes da EPROM programa aplicativo banco 5	
39	Espaço livre em bytes da EPROM programa aplicativo banco 6	
40	Espaço livre em bytes da EPROM programa aplicativo banco 7	
41	Espaço livre em bytes da EPROM programa aplicativo banco 8	
42	Espaço livre em bytes da EPROM programa aplicativo banco 9	
43	Espaço livre em bytes da EPROM programa aplicativo banco 10	
44	Espaço livre em bytes da EPROM programa aplicativo banco 11	
45	Espaço livre em bytes da EPROM programa aplicativo banco 12	
46	Espaço livre em bytes da EPROM programa aplicativo banco 13	
47	Espaço livre em bytes da EPROM programa aplicativo banco 14	
48	Espaço livre em bytes da EPROM programa aplicativo banco 15	
49	Operando reservado	

**Utilização**

Esta função pode ser utilizada em todas UCPs Altus.



## F-NEGT.093 - Função para Negação Lógica de Operandos Tabela



### Introdução

A função F-NEGT.093 realiza a negação lógica de operandos simples (M ou D) ou tabelas (TM ou TD). Podem ser negadas até 255 posições em uma única chamada a função. O resultado da alteração pode ser armazenado no próprio operando, substituindo o valor original, ou em um outro operando, desde que ele seja do mesmo tipo do primeiro (memória ou decimal).

### Programação

As células da instrução CHF utilizada para a chamada são programadas do seguinte modo:

- **OPER1** - Especifica o número de parâmetros que são passados para a função em OPER3. Este operando deverá ser obrigatoriamente uma constante memória com valor 3 (KM+00003).
- **OPER2** - Deve ser um operando do tipo constante memória com valor 0 (KM+00000). Determina o número de parâmetros possíveis de serem programados na janela de edição de OPER4. Como esta função não necessita de nenhum parâmetro em OPER4, o valor de OPER2 é 0.
- **OPER3** - Contém os parâmetros que são passados para a função, declarados através de uma janela visualizada no AL-3830 quando a instrução CHF for editada. O número de parâmetros editáveis é especificado em OPER1, sendo fixo em 3 para este módulo:
  - MXXXX, DXXXX, TMXXXX ou TDXXXX - Operando simples ou tabela cujos valores serão negados (operando fonte).
  - MXXXX, DXXXX, TMXXXX ou TDXXXX - Operando simples ou tabela onde os valores negados serão armazenados (operando destino).
  - KMXXXX - Número de operandos simples ou posições da tabela a serem negadas. Deve ser menor ou igual a 255.
- **OPER4** - Não utilizado.

### Entradas e Saídas

Descrição das entradas da função:

- **habilita** - quando esta entrada está energizada a função é chamada, sendo analisados os parâmetros programados na instrução CHF. Caso os mesmos estejam incorretos, são acionadas as saídas de índice inválido.

Descrição das saídas da função:

- **sucesso** - indica que a movimentação foi corretamente realizada.
- **índice fonte inválido** - indica que houve erro na especificação do operando fonte:
  - o operando não está declarado no módulo C
  - não existem posições suficientes para realizar a lógica
- **índice destino inválido** - indica que houve erro na especificação do operando destino:
  - o operando não está declarado no módulo C
  - não existem posições suficientes para realizar a lógica

Caso as duas saídas de índice inválido sejam acionadas simultaneamente, ocorreu algum dos seguintes erros:

- o número de parâmetros programado em OPER1 é diferente de três
- o tipo de um dos parâmetros em OPER3 não é válido
- o tipo do operando destino é diferente do operando fonte
- o número total de posições a serem transferidas é maior que 255

**ATENÇÃO:**

Esta função permite a negação de um grande número de operandos em uma única varredura. Deve-se utilizá-la com cuidado para que o tempo máximo de ciclo do programa não seja excedido.

# Glossário

## Glossário da Série Ponto

- **Barramento:** Conjunto de módulos de E/S interligados a uma UCP ou Cabeça de Rede de Campo.
- **Barramento Local:** Conjunto de módulos de E/S interligados a uma UCP.
- **Barramento Remoto:** Conjunto de módulos de E/S interligados a uma cabeça de rede de campo.
- **Base :** Componente onde são inseridos os módulos de E/S, UCPs, fontes e demais módulos da Série Ponto.
- **Cabeça de Rede de Campo:** Módulo escravo de uma rede de campo. É responsável pela troca de dados entre seus módulos e com um mestre de rede de campo.
- **Cabo de Expansão:** Cabo que interliga os expansores de barramento.
- **Cabo da Rede de Campo:** Cabo que conecta os nós de uma rede de campo, tal como a Interface de Rede de Campo e as Cabeça de Rede de Campo.
- **Código Chave Mecânica:** Dois dígitos que são definidos por meio de chaves mecânicas, programáveis na base com objetivo de impedir a montagem de módulos não compatíveis.
- **Código Comercial:** É o código do produto, formado pelas letras PO e seguidos por quatro números.
- **Endereço da Cabeça de Rede de Campo:** É o endereço de um nó da rede de campo. É ajustado na base do módulo de cabeça de rede de campo.
- **Expansor de Barramento:** Módulo que interliga um segmento de barramento em outro
- **Fiação de campo:** Cabos que conectam sensores, atuadores e outros dispositivos do processo/máquina nos módulos de E/S da Série Ponto.
- **Interface de Rede de Campo:** Módulo mestre de redes de campo, localizado no barramento local destinado a fazer a comunicação com cabeças de rede de campo.
- **Segmento de barramento:** Parte de um barramento. Um barramento local ou remoto pode ser dividido em no máximo quatro segmentos de barramento.
- **Terminação de Barramento:** Componente que deve ser conectado no último módulo de um barramento.
- **Trilho:** Elemento metálico com perfil normalizado segundo a norma DIN50032, também chamado de trilho TS35.
- **UCP:** Unidade Central de Processamento, responsável pela execução do programa aplicativo.

## Glossário de Redes

- **Acesso ao meio:** Método utilizado por todos os nós de uma rede de comunicação para sincronizar as transmissões de dados e resolver possíveis conflitos de transmissões simultâneas.
- **Backoff:** Tempo que um nó de uma rede tipo CSMA/CD aguarda antes de voltar a transmitir dados após a ocorrência de colisão no meio físico.
- **Baud rate:** Taxa com que os bits de informação são transmitidos através de uma interface serial ou rede de comunicação. ( medido em Bits/segundo )
- **Bridge (ponte) :** Equipamento para conexão de duas redes de comunicação dentro de um mesmo protocolo.
- **Broadcast:** Disseminação simultânea de informação a todos os nós interligados a uma rede de comunicação.
- **Canal serial:** Interface de um equipamento que transfere dados no modo serial.

- **CSMA/CD.** Disciplina de acesso ao meio físico, baseada na colisão de dados, utilizada pelas redes ETHERNET.
- **EIA RS-485:** Padrão industrial (nível físico) para comunicação de dados.
- **Escravo:** Equipamento ligado a uma rede de comunicação que só transmite dados se for solicitado por outro equipamento denominado mestre.
- **Frame:** Uma unidade de informação transmitida na rede.
- **Gateway:** Equipamento para a conexão de duas redes de comunicação com diferentes protocolos.
- **Mestre:** Equipamento ligado a uma rede de comunicação de onde se originam solicitações de comandos para outros equipamentos da rede.
- **Multicast:** Disseminação simultânea de informação a um determinado grupo de nós interligados a uma rede de comunicação.
- **Nó ou nodo:** Qualquer estação de uma rede com capacidade de comunicação utilizando um protocolo estabelecido.
- **Peer to peer:** é um tipo de comunicação onde dois parceiros trocam dados e/ou avisos sem depender de um mestre.
- **Protocolo:** Regras de procedimentos e formatos convencionais que, mediante sinais de controle, permitem o estabelecimento de uma transmissão de dados e a recuperação de erros entre equipamentos.
- **Rede de comunicação determinística:** Rede de comunicação onde a transmissão e recepção de informações entre os diversos nós é garantida com um tempo máximo conhecido.
- **Rede de comunicação mestre-escravo:** Rede de comunicação onde as transferências de informações são iniciadas somente a partir de um único nó (o mestre da rede) ligado ao barramento de dados. Os demais nós da rede (escravos) apenas respondem quando solicitados.
- **Rede de comunicação multimestre.** Rede de comunicação onde as transferências de informações são iniciadas por qualquer nó ligado ao barramento de dados.
- **Rede de comunicação:** Conjunto de equipamentos (nós) interconectados por canais de comunicação.
- **Sub rede:** Segmento de uma rede de comunicação que interliga um grupo de equipamentos (nós) com o objetivo de isolar o tráfego local ou utilizar diferentes protocolos ou meio físicos.
- **Time-out:** Tempo preestabelecido máximo para que uma comunicação seja completada, que, se for excedido, provoca a ocorrência de um erro de comunicação.
- **Token:** é uma marca que indica quem é o mestre do barramento no momento.

## Glossário Geral

- **Algoritmo:** Sequência finita de instruções bem definidas objetivando a resolução de problemas.
- **Arrestor:** Dispositivo de proteção contra raios carregado com gás inerte.
- **Barramento:** Conjunto de sinais elétricos agrupados logicamente com a função de transferir informação e controle entre diferentes elementos de um subsistema.
- **Bit:** Unidade básica de informação, podendo estar no estado 0 ou 1.
- **Byte:** Unidade de informação composta por oito bits.
- **Ciclo de varredura:** Uma execução completa do programa aplicativo de um controlador programável.
- **Circuito de cão-de-guarda:** Circuito eletrônico destinado a verificar a integridade no funcionamento de um equipamento.
- **Controlador Programável:** Equipamento que realiza controle sob o comando de um programa aplicativo escrito em linguagem de relés e blocos. Compõe-se de uma UCP, fonte de alimentação e estrutura de entrada/saída.
- **Database:** banco de dados.

- **Default:** valor pré-definido para uma variável, utilizado em caso de não haver definição.
- **Diagnóstico.** Procedimento utilizado para detectar e isolar falhas. É também o conjunto de dados usados para tal determinação, que serve para a análise e correção de problemas.
- **Download:** carga de programa ou configuração nos módulos.
- **Encoder:** transdutor para medidas de posição.
- **Endereço de módulo:** Endereço pelo qual o CP realiza acessos a um determinado módulo de E/S colocado no barramento.
- **EPROM (Erasable Programmable Read Only Memory) :** Memória somente de leitura, apagável e programável. Não perde seu conteúdo quando desenergizada.
- **Estação de supervisão:** Equipamento ligado a uma rede de CPs ou instrumentação com a finalidade de monitorar ou controlar variáveis de um processo.
- **E2PROM:** Memória não volátil, que pode ser apagada eletricamente.
- **E/S (entrada/saída):** Dispositivos de entrada e/ou saída de dados de um sistema. No caso de CPs, correspondem tipicamente a módulos digitais ou analógicos de entrada ou saída, que monitoram ou acionam o dispositivo controlado.
- **Flash EPROM.** Memória não volátil que pode ser apagada eletricamente.
- **Hardkey:** Conector normalmente ligado à interface paralela do microcomputador com a finalidade de impedir a execução de cópias ilegais de um software.
- **Hardware:** Equipamentos físicos usados em processamento de dados, onde normalmente são executados programas (software).
- **IEC Pub. 144 (1963):** norma para proteção contra acesso incidentais ao equipamento e vedação para água, pó ou outros objetos estranhos ao equipamento.
- **IEC 1131:** Norma genérica para operação e utilização de Controladores Programáveis.
- **IEC-536-1976:** Norma para proteção contra choque elétrico
- **IEC-801-4:** norma para testes de imunidade a interferências por trem de pulsos
- **IEEE C37.90.1 (SWC- Surge Withstand Capability):** norma para proteção contra ruídos tipo onda oscilatória.
- **Interface:** Dispositivo que adapta elétrica e/ou logicamente a transferência de sinais entre dois equipamentos.
- **Interrupção:** Evento com atendimento prioritário que temporariamente suspende a execução de um programa.
- **Kbytes:** Unidade representativa de quantidade de memória. Representa 1024 bytes.
- **LED (Light Emitting Diode):** Tipo de diodo semicondutor que emite luz quando estimulado por eletricidade. Utilizado como indicador luminoso.
- **Linguagem Assemble:** Linguagem de programação do microprocessador, também conhecida como linguagem de máquina.
- **Linguagem de programação:** Um conjunto de regras, de convenções e de sintaxe utilizado para a elaboração de um programa.
- **Linguagem de Relés e Blocos ALTUS:** Conjunto de instruções e operandos que permitem a edição de um programa aplicativo para ser utilizado em um CP.
- **Lógica:** Matriz gráfica onde são inseridas as instruções da linguagem de diagrama de relés que compõem um programa aplicativo. Um conjunto de lógicas ordenadas sequencialmente constitui um módulo de programa.
- **Menu:** Conjunto de opções disponíveis e exibidas no vídeo por um programa, a serem selecionadas pelo usuário a fim de ativar ou executar uma determinada tarefa.

- **Módulo de configuração (Módulo C) :** Módulo único em um programa de CP que contém diversos parâmetros necessários ao funcionamento do controlador, tais como a quantidade de operandos e a disposição dos módulos de E/S no barramento.
- **Módulo de E/S:** Módulo pertencente ao subsistema de Entradas e Saídas.
- **Módulo função (Módulo F):** Módulo de um programa de CP que é chamado a partir do módulo principal (módulo E) ou a partir de outro módulo função ou procedimento, com passagem de parâmetros e retorno de valores, servindo como uma sub-rotina.
- **Módulo procedimento (Módulo P):** Módulo de um programa de CP que é chamado a partir do módulo principal (módulo E) ou a partir de outro módulo procedimento ou função, sem a passagem de parâmetros.
- **Módulo (quando se referir a hardware):** Elemento básico de um sistema completo que possui funções bem definidas. Normalmente é ligado ao sistema por conectores podendo ser facilmente substituído.
- **Módulo (quando se referir a software):** Parte de um programa aplicativo capaz de realizar uma função específica. Pode ser executado independentemente ou em conjunto com outros módulos trocando informações através da passagem de parâmetros.
- **Módulos execução (Módulo E):** Módulos que contêm o programa aplicativo, podendo ser de três tipos: E000, E001 e E018. O módulo E000 é executado uma única vez na energização do CP ou na passagem de programação para execução. O módulo E001 contém o trecho principal do programa que é executado ciclicamente, enquanto que o módulo E018 é acionado por interrupção de tempo.
- **Nibble:** Unidade de informação composta por quatro bits.
- **Octeto:** Conjunto de oito bits numerados de 0 a 7.
- **Operandos:** Elementos sobre os quais as instruções atuam. Podem representar constantes, variáveis ou conjunto de variáveis.
- **PC (Programmable Controller):** Abreviatura de Controlador Programável em inglês.
- **Ponte-de-ajuste:** Chave de seleção de endereços ou configuração, composta por pinos presentes na placa do circuito e um pequeno conector removível, utilizado para a seleção.
- **Posta-em-marcha:** Procedimento de depuração final do sistema de controle, quando os programas de todas as estações remotas e UCPs são executados em conjunto, após terem sido desenvolvidos e verificados individualmente.
- **Programa aplicativo:** É o programa carregado em um CP, que determina o funcionamento de uma máquina ou processo.
- **Programa executivo:** Sistema operacional de um controlador programável; controla as funções básicas do controlador e a execução de programas aplicativos.
- **RAM (Random Access Memory):** Memória onde todos os endereços podem ser acessados diretamente de forma aleatória e a mesma velocidade. É volátil, ou seja, seu conteúdo é perdido quando desenergizada, a menos que possua bateria para retenção dos valores.
- **Ripple:** Ondulação presente em tensão de alimentação contínua.
- **Sistema redundante:** Sistema que contém elementos de reserva ou duplicados para executar determinada tarefa, que podem tolerar determinados tipos de falha sem que execução da tarefa seja comprometida.
- **Software:** Programas de computador, procedimentos e regras relacionadas à operação de um sistema de processamento de dados.
- **Soquete:** Dispositivo no qual se encaixam circuitos integrados ou outros componentes, facilitando a substituição dos mesmos e simplificando a manutenção.
- **Subsistema de E/S:** Conjunto de módulos de E/S digitais ou analógicos e interfaces de um Controlador Programável.
- **Tag:** Nome associado a um operando ou a uma lógica que permite uma identificação resumida de seu conteúdo.
- **Toggle.** Elemento que possui dois estados estáveis, trocados alternadamente a cada ativação.

- **Troca a quente:** Procedimento de substituição de módulos de um sistema sem a necessidade de desenergização do mesmo. Normalmente utilizado em trocas de módulos de E/S.
- **UCP ativa:** Em um sistema redundante, é a UCP que realiza o controle do sistema, lendo os valores dos pontos de entrada, executando o programa aplicativo e acionando os valores das saídas.
- **UCP inoperante:** UCP que não está no estado ativo (controlando o sistema) nem no estado reserva (supervisionando a UCP ativa), não podendo assumir o controle do sistema.
- **UCP redundante:** Corresponde à outra UCP do sistema, em relação à que o texto do manual está se referindo. Por exemplo, a UCP redundante da UCP 2 é a UCP 1 e vice versa.
- **UCP reserva:** Em um sistema redundante, é a UCP que supervisiona a UCP ativa, não realizando o controle do sistema, estando pronta para assumir o controle em caso de falha na UCP ativa.
- **UCP:** Unidade central de processamento. Controla o fluxo de informações, interpreta e executa as instruções do programa e monitora os dispositivos do sistema.
- **Upload:** leitura de programa ou configuração dos módulos.
- **Varistor:** Dispositivo de proteção contra surto de tensão.
- **Word:** Unidade de informação composta por dezesseis bits.

## Principais Abreviaturas

- BAT: Bateria
- BT: Teste de Bateria, do inglês "Battery Test"
- CT: Características Técnicas
- CP: Controlador Programável
- DP: Abreviatura para Decentralized Periphery
- EEPROM: "Electric Erasable Programmable Read Only Memory"
- EMI: Electromagnetic Interference. Interferência Eletromagnética
- EPROM: "Erasable Programmable Read Only Memory"
- ER: Erro
- ESD: ElectroStatic Discharge. Descarga devida a eletricidade estática.
- EX: Execução
- E2PROM: "Electric Erasable Programmable Read Only Memory"
- E/S: Entradas e Saídas
- FC: Forçamento
- Flash EPROM: "Flash Erase Programmable Read Only Memory"
- FMS: Abreviatura para Fieldbus Message System
- INTERF.: Interface
- ISOL.: Isolado(s), Isolamento
- LED: diodo emissor de luz, do inglês "Light Emitting Diode"
- Máx.: máximo ou máxima
- Mín.: mínimo ou mínima
- Obs.: observação ou observações
- PAs: Pontes de Ajuste
- PA: Abreviatura para Process Automation

- PG: Programação
- PID: controle Proporcional, Integral e Derivativo.
- RAM: "Random Access Memory"
- ref.: referência
- RX: Recepção Serial
- SELEC.: Seleccionável
- TX: Transmissão serial
- UCP: Unidade Central de Processamento
- UTIL.: Utilização
- WD: cão-de-guarda , do inglês "watchdog"