Controladores Programáveis

Linguagem Descritiva BCM

Versão:2018-1
Ref.:31940092-8

BCM
automação

Linguagem Descritiva BCM

Este documento é propriedade da BCM ENGENHARIA LTDA. Seu conteúdo tem caráter exclusivamente informativo, cabendo à BCM o direito de promover alterações necessárias, sem aviso prévio.

É proibida a reprodução parcial ou total sem o expresso consentimento da BCM Engenharia Ltda.

ATENÇÃO

Os Controladores Programáveis BCM são equipamentos robustos e confiáveis. O seu projeto foi feito levando em conta as condições de operação particulares do ambiente industrial. Porém, nunca esqueça que todos os elementos de um sistema estão sujeitos a falhas. Desenvolva o projeto do sistema levando isto em consideração, obedecendo rigorosamente as recomendações deste manual e das normas de segurança vigentes em seu país ou região.

Um bom projeto do sistema e uma correta instalação são elementos fundamentais para o funcionamento satisfatório e confiável dos produtos.

Caso haja qualquer ponto duvidoso ou omisso, não hesite em consultar o Departamento de Assistência Técnica da BCM, o qual terá o maior prazer de lhe prestar todo o apoio necessário.

Telefone:(51) 3374.3899
Fax:(51) 3374.4141
***E-mail:*sistemas@bcmautomacao.com.br**
www.bcmautomacao.com.br

COMPATIBILIDADE COM O PROCP

As instruções descritas neste manual podem ser usadas com o PROCPV101.0, PROCPV201.0, PROCPV301.0 , PROCP V3k, V4k, V5k e V6.

Existem instruções que só são compiladas usando o PROCPV301.0 , PROCP V3k e PROCP V4k. Outras somente estão disponíveis para o PROCP V6. Estas instruções informam esta característica na sua descrição.

Para dúvidas a respeito da disponibilidade de instruções em versões do PROCP anteriores, consulte a BCM.

Índice

1 - CONTROLE DE PROCESSOS COM CPS 11

1.1 - DEFINIÇÃO E CARACTERISTICAS DO CP..... 11

1.2 - ABORDAGEM DO PROBLEMA E REPRESENTAÇÃO..... 13

1.3 - DOCUMENTAÇÃO DE PROJETOS..... 17

1.4 - TERMINOLOGIA E CONVENÇÕES USADAS..... 18

2 - A LINGUAGEM DESCRITIVA BCM..... 31

2.1 - ESTRUTURA DA LINGUAGEM DESCRITIVA BCM..... 31

2.2 - REGRAS DE PROGRAMAÇÃO NA LINGUAGEM DESCRITIVA..... 33

2.3 - COMO PROGRAMAR NA LINGUAGEM DESCRITIVA..... 39

2.4 - SINTAXE DAS INSTRUÇÕES E COMANDOS..... 40
2.4.1 - Comandos de configuração do CP 41
2.4.2 - Intruções de programação..... 57

2.5 - EXEMPLOS TIPICOS DE PROGRAMAÇÃO..... 113
2.5.1 - Leitura das Entradas Digitais..... 113
2.5.2 - Controle das Saídas Digitais..... 115
2.5.3 - Atribuição de Valor para Variáveis..... 115
2.5.4 - Uso de contadores..... 117
2.5.5 - Uso de temporizadores..... 120
2.5.6 - Temporizadores Associados ao Teste de Variável..... 125
2.5.7 - Leitura das Entradas Analógicas..... 126
2.5.8 - Controle das Saídas Analógicas..... 130
2.5.9 - Uso do Display..... 132
2.5.10 - Uso de Matrizes..... 145
2.5.11 - Controle PID..... 149
2.5.12 - Uso dos bancos de memória..... 157

2.6 - TRATAMENTO DE VARIÁVEIS..... 159
2.6.1 - Varáveis de Atribuição Livre..... 160
- Variáveis Indexadas..... 161
- Variáveis não Indexadas..... 161
2.6.2 - Variáveis de Entrada e Saídas 162
- Entradas Digitais..... 162
- Entradas Analógicas..... 163
- Saídas Analógicas..... 163
- Display..... 164
2.6.3 - Variáveis de atribuição Fixa..... 166
- Temporização..... 167
- Relógio..... 168
- Reconhecimento dos Módulos de E/S..... 169
- Controle de Programa..... 172
- Contadores Rápidos..... 176
- Interfaces com Periféricos..... 177
- Canais de Comunicação..... 183
- Medição do Tempo de Varredura..... 185

2.7 - REGRAS ESPECIAIS DE PROCESSAMENTO ARITMÉTICO..... 186

2.8 - COMO O PROGRAMA É EXECUTADO / CICLO DE VARREDURA..... 187

2.9 - PRIORIDADE NAS OPERAÇÕES..... 190

Índice Geral das Instruções e Comandos

COMANDOS DE CONFIGURAÇÃO

BCD - Tratamento das Chaves Thumbwheel..... 41
CLP - Define o tipo de Controlador..... 41
CONTINUA - Divide o programa em Várias Partes..... 43
CTR - Habilita as Rotinas para Tratamento de Contadores..... 44
DIM - Define o Dimensão de uma Matriz..... 44
DISPLAY - Habilita a Operação dos Displays..... 46
ESA - Habilita Rotinas para Tratamento de E/S Analógicas..... 47
FIM - Instrui o Compilador a Encerrar a Compilação..... 47
IFT - Habilita o Uso do Módulo IFT..... 47
INTERFACE - Define Parametros de Comunicação 48
NEST - Reserva um Espaço de Memória para uma Tabela..... 55
PAN - Habilita a Rotina para Tratamento do Módulo PAN..... 55
RTC - Habilita a Operação do Relógio em Tempo Real..... 55
SAIDA - Associa Variáveis as Saídas 55
TCP - Configura os Parâmetros do Canal Ethernet..... 56

INSTRUÇÕES DE PROGRAMAÇÃO

COPIA - Copia variáveis de uma matriz para outra..... 57
DECREMENTA - Diminui uma unidade na Variável..... 57
EAI - Programa os Valores Relativos ao Módulo EAI..... 58
EAN - Programa os Valores Relativos ao Módulo EAN..... 62
ESCISI - Escreve uma Variável em um Equipamento da Rede. 65
ESCMISI - Escreve na Memória do Módulo ISI..... 66
ESCREVE - Escreve Informações em Outro Equipamento..... 67
ESTADO - Marca o Inìcio das Instruções para cada Estado..... 69
EVENTOS - Configura o módulo EDE..... 69
FACA - Intrução que Executa Operações Aritméticas..... 78
IMPRIME - Imprime o conteudo da Variável..... 81
INCLUI - Inclui um trecho de programa em Ling. Descritiva..... 81
INCREMENTA - Incrementa uma Unidade na variável 82
ISI - Especifica a Porta Serial a ser Programada 82
LEISI - Lê uma Variável de um Equipamento em Rede..... 83
LEMISI - Faz a Leitura da Memória do Módulo ISI..... 84
LE - Adquire Informações de outro Equipamento na Rede..... 85
LIGA DESL - Liga e Desliga as Saídas Especificadas..... 87

INSTRUÇÕES DE PROGRAMAÇÃO (continuação)

MALHA/MAQUINA - Indica o início dos estados 87
MOSTRA - Apresenta as variáveis nos Displays..... 87
MOTOR - Sistemas de Posicionamento com Módulo MOT..... 92
PID - Executa Rotina PID no Controlador 101
PLIGA/PDESL - Liga e desliga pontos no módulo PAN..... 105
PRESET - Preseta Variáveis..... 105
RESET - Coloca a Variável Mencionada em Zero..... 106
ROTINA - Permite Inserir Rotinas 107
SE - Transição de Estado..... 112
SET - Coloca o Valor 1 Nas Variáveis Mencionadas..... 112
VA PARA - Mudança Imediata de Estado..... 112

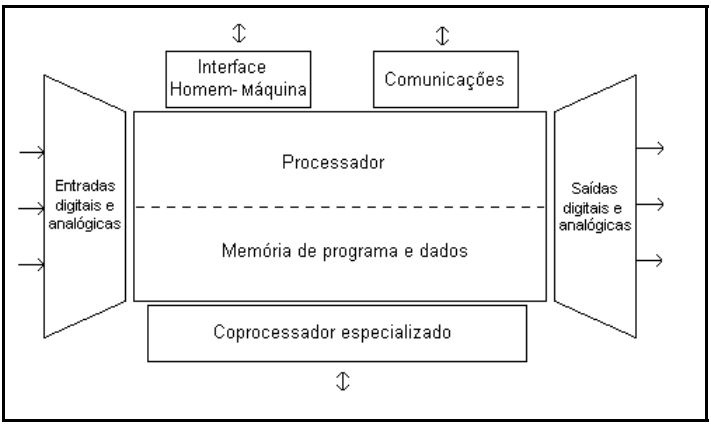
1 - CONTROLE DE PROCESSOS COM CPs

Criado para ser um substituto dos quadros de relés, o Controlador Programável (CP) ultrapassou há muito as expectativas originais. Com processadores de alto desempenho, variedade de modelos e capacidade de comunicação, o seu campo de aplicação hoje é quase ilimitado e o conhecimento de suas potencialidades torna-se cada vez mais necessário para todos os envolvidos no planejamento, operação e manutenção de processos industriais. Nos próximos anos a sua importância deverá continuar crescendo, pois as pesquisas de mercado indicam uma contínua expansão na demanda por CPs, confirmando o inexorável avanço da automação em todas as áreas de atividade.

1.1 - DEFINIÇÃO E CARACTERÍSTICAS DO CP

"O Controlador Programável é um dispositivo que, observando a arquitetura de computadores, possui uma memória programável pelo usuário e realiza as funções de controle, comando e supervisão de processos, com hardware e software compatíveis com estas aplicações."

A estrutura básica do Controlador Programável segue uma linha similar a dos computadores de uso geral: Dispositivos de Entrada/ Processamento/ Saída. Nos CPs estas estruturas assumem uma forma particular, adequada ao ambiente de controle de processos:



Entradas: As entradas digitais recebem o sinal de sensores, chaves,botoeiras, e outros equipamentos que fornecem sinais do tipo ligado/desligado. As entradas analógicas recebem sinais de tensão ou corrente de variação contínua, dentro de uma faixa e com significado especificado. Este módulo condiciona o sinal de entrada e torna-o disponível para o processador.

Processador: É o bloco que analisa, processa e decide. São executadas aí funções de decisão, operações matemáticas, contagens, temporização, diálogo com IHM, comunicação, etc.

Memória: A memória de programa contém as instruções armazenadas para execução das tarefas previstas para a aplicação. A memória de variáveis contém os valores e parâmetros correntes relacionados ao processo.

Saídas: Através das saídas, o CP age sobre o processo sob o seu controle. As saídas digitais fornecem comandos do tipo ligado/desligado. As saídas analógicas fornecem um sinal de tensão ou corrente com variação contínua, para acionar válvulas proporcionais, conversores ou outros equipamentos.

Co-processadores: Módulos com memória local e funções especializadas executam tratamento sofisticado de entradas analógicas, comunicações com alto desempenho, controle de posicionamento com alta velocidade, operações matemáticas e outras.

Interface Homem-Máquina: Este bloco coloca o operador (usuário) em contato direto com o CP e o processo sob controle. Podem ser passadas informações de condições de processo, alarmes, solicitações manuais e informações numéricas. A forma mais comum de interface Homem-Máquina no CP é um display alfanumérico com teclado.

Comunicações: Este bloco fornece o meio físico e os protocolos para que o CP se comunique com outros equipamentos integrantes do sistema. Os sistemas de controle atuais prevêem a integração de diversos dispositivos que podem incluir vários CPs, computadores, interfaces IHM externos, sensores e atuadores inteligentes, todos ligados em rede.

1.2 - ABORDAGEM DO PROBLEMA E REPRESENTAÇÃO

De forma geral, o processo de desenvolvimento de uma aplicação de Controlador Programável atende uma seqüência de etapas, com objetivos bem determinados. O cumprimento satisfatório de cada etapa irá garantir a segurança do trabalho e o sucesso do resultado final. Serão apresentadas em linhas gerais as características e requisitos de cada etapa.

Ciclo de Desenvolvimento de uma Aplicação

Definição - Nessa etapa será explicitado o que o sistema irá fazer, sob que condições, as relações do equipamento de automação com os diferentes usuários e as características das ligações com outros sistemas.

Projeto - Determinar uma solução viável e econômica para o problema a ser resolvido. Dessa fase já sairá uma configuração definitiva dos equipamentos a serem usados e as linhas gerais do software a ser desenvolvido.

Construção - Baseado nas especificações geradas nas etapas anteriores, o projeto será convertido em um programa ou programas que atendam os requisitos desejados. Paralelamente, os equipamentos necessários serão construídos de acordo com as solicitações do projeto.

Teste e Avaliação
Essa etapa deve responder basicamente a seguinte pergunta: O sistema atende as necessidades e expectativas do usuário? Em caso negativo, quais as modificações necessárias? Para responder a essas perguntas poderemos trabalhar com simuladores do sistema, testes no sistema real em condições normais e anormais, etc.

Definição

O primeiro passo na etapa de definição é a elaboração do documento denominado "Especificação do Problema", também chamado de "Especificação de Requisitos". Esse documento coloca claramente as necessidades do projeto e as expectativas do usuário. Somente uma redação completa e cuidadosa pode resultar em uma solução adequada e de boa qualidade. A Especificação do Problema é o instrumento fundamental para o diálogo entre o cliente do sistema e a equipe de desenvolvimento.

Em alguns casos, ao se iniciar o trabalho, a especificação pode já existir, ser elaborada pelo cliente ou pelo responsável pelo desenvolvimento.

Projeto

Na fase de projeto iremos construir uma solução para o problema. O resultado principal dessa fase é a "Especificação da solução" ou Especificação de Projeto. Baseado no conhecimento dos módulos e equipamentos disponíveis, o projetista irá determinar uma solução genérica completa para o problema. Será gerado um diagrama físico (hardware) e um diagrama lógico (software) da solução de automação, acompanhado de uma descrição detalhada de como os objetivos serão atingidos. Onde for conveniente, deve-se considerar a inclusão de normas universais relativas ao assunto, roteiros de teste, manutenção, etc.

O projetista nessa fase já irá sugerir uma divisão lógica para a abordagem da programação. Não iremos aqui abordar o desenvolvimento subseqüente da parte eletrônica (hardware) do projeto; iremos nos fixar na parte de software e como o programador irá trabalhar de aqui por diante.

No cotidiano do trabalho de automação com Controladores Programáveis notamos que frequentemente as especificações do problema e da solução são reunidas em um documento único. Tal procedimento é válido principalmente em processos mais simples de automação, uma vez que simplifica o trabalho do projetista.

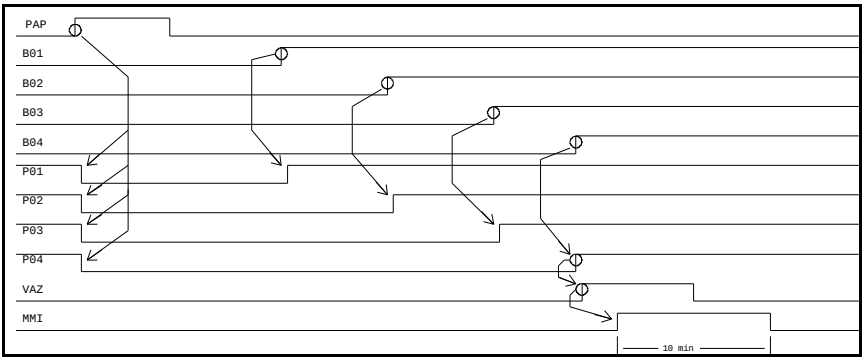
A principal exigência nessa forma de abordagem é que o responsável pela especificação deve conhecer simultaneamente o processo (para poder redigir a Especificação do Problema), os equipamentos disponíveis e os recursos de programação para poder projetar uma solução adequada. Com os equipamentos padronizados disponíveis e uma linguagem de alto nível para programação dos CPs, o próprio usuário pode frequentemente construir uma especificação completa e adequada sem grande dificuldade. Como podemos notar, a passagem da fase de projeto para a fase de construção do programa depende basicamente do cuidado com que as especificações foram elaboradas. Na linguagem descritiva BCM existe uma preocupação fundamental em aproximar a forma de raciocínio usada na especificação com a elaboração do programa.

Quando o projetista elabora a Especificação da Solução ele tem uma necessidade fundamental de descrever o fluxo do processo ou seja, como os eventos se sucedem no sistema, quais as causas e efeitos de cada evento e a seqüência lógica de tarefas que o controlador deve executar. Existem várias formas de fazer isso:

F Descrevendo literalmente a seqüência de eventos:

1 - Ao acionar o botão de partida PAR, as portas P01, P02, P03 e P04 devem ser abertas.
2 - Quando B01 atingir o peso programado, P01 fecha.
3 - Quando B02 atingir o peso programado, P02 fecha.
4 - Quando B03 atingir o peso programado, P03 fecha.
5 - Quando B04 atingir o peso programado, P04 fecha.
6 - Quando P01, P02, P03 e P04 estiverem novamente fechadas, VAZ é acionado, virando todos os ingredientes no recipiente grande.
7 - Após 15 segundos, MMI deve ser acionado.
8 - Após 10 minutos, MMI deve ser desligado, permanecendo em repouso.

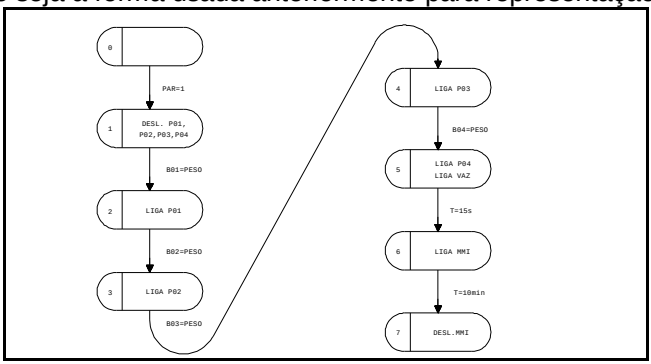
F Montando um diagrama de tempos:



F Elaborando um diagrama de estados:

Não é objetivo aqui discutir ou comparar os métodos tradicionais de representação. A linguagem BCM propõe uma forma alternativa para representação que substitui os anteriores. Além disso, elaborada convenientemente, essa forma de representação é usada diretamente para programar o controlador.

A estrutura fundamental na programação em linguagem BCM é o Diagrama de Estados. A passagem da especificação devidamente elaborada para o Diagrama de Estados é quase direta, exigindo um mínimo de elaboração adicional, qualquer que seja a forma usada anteriormente para representação:



1.3 - DOCUMENTAÇÃO DE PROJETOS

Ao lado da descrição de fluxo do processo, uma série de documentos devem ser gerados durante o desenvolvimento da aplicação. A lista a seguir apresenta a documentação padronizada adotada nos projetos BCM:

- Especificação do problema.
- Especificação da solução. Conceitualmente diferente, pode ser feita em conjunto com a especificação do problema em projetos simples e/ou quando o mesmo técnico detalha o processo e aplica o CP.
- Diagrama de blocos físico do sistema abordado.
- Listagem de entradas e saídas do CP - Símbolos e funções.
- Listagem de matrizes e variáveis utilizadas.
- Protocolos especiais de interface com outros CPs ou equipamentos (serial, paralela, via E/S, etc.)
- Divisão de malhas de estado e descrição da função de cada uma.
- Diagramas de estado para cada malha, com comentários.
- Listagem BCM completa.
- Descrição funcional, fluxogramas e listagem de rotinas assembler, se existirem.

Em muitos casos, é usual incluir um texto dedicado ao processo de desenvolvimento do software, descrevendo os arquivos usados, a nomenclatura, roteiros de trabalho, etc. Isso é feito para facilitar uma eventual necessidade futura de manutenção ou adaptações no programa.

1.4 - TERMINOLOGIA E CONVENÇÕES USADAS

Ao longo dos manuais BCM e no cotidiano do trabalho em automação e Controladores Programáveis, encontramos uma série de termos e conceitos típicos. No sentido de familiarizar o usuário com esta terminologia, apresentamos um glossário dos termos e expressões mais usuais. Em muitos casos, as informações apresentadas não se atem a rígidas definições formais. Acreditamos, no entanto, que este capítulo cumpre o objetivo de auxiliar a compreensão e eliminar dúvidas.

Álgebra Booleana - Notação matemática abreviada que representa relações e expressões lógicas associadas as funções básicas *e*, *ou*, *ou exclusivo e não*.

Algoritmo - Conjunto de regras e operações capazes de efetuar uma tarefa previamente especificada.

Apontador - Valor numérico usado para indicar a localização, na memória de dados ou de programa, de uma determinada informação, de um espaço livre ou do ponto inicial de execução de uma rotina.

ASCII - Abreviatura para American Standard Code for Information Interchange. Esse código representa todos os caracteres alfanuméricos e símbolos do teclado padrão através de números de 7 bits (0 a 127). Alguns códigos são ainda reservados para função de controle (troca linha, retorna carro, troca página, etc).

Assembler - Linguagem de baixo nível, orientada diretamente ao microprocessador do sistema, que descreve o conjunto de operações básicas disponíveis.

Barra - Um ou mais condutores que carregam coletivamente uma informação codificada, proveniente de um dispositivo fonte para um ou vários dispositivos receptores.

Baud Rate - Taxa (velocidade) de transmissão de um sinal digital codificado, calculada em bits por segundo. Por exemplo: 9600 baud equivale a uma velocidade de transmissão de 9600 bits por segundo.

BCD - Abreviatura de Binary Code Decimal. Sistema de numeração que representa separadamente cada dígito decimal pelo equivalente binário em 4 bits. Por exemplo 371 equivale a 0011.0111.0001 em código BCD.

Binário - Sistema de numeração de base 2, em contraposição ao sistema decimal, que usa a base 10. O sistema binário requer somente dois tipos de dígito (0,1) que agrupados representam qualquer quantia desejada pelo usuário.

BIT - Abreviatura de Binary Digit. Um bit é a menor unidade de informação disponível em um sistema binário. Representa a decisão entre dois estados possíveis, simbolizados normalmente 0 e 1. Pode ser usado também para representar os estados ligado/desligado ou verdadeiro/falso.

Buffer - Área de memória reservada para armazenar uma determinada informação na passagem de uma fase de processamento para outra.

Byte - Grupo de 8 bits processado em um sistema como uma entidade única.

Checksum - Resultado da soma de todos os valores de um determinado bloco de informação utilizado para verificação da integridade da informação armazenada. Uma determinada fase do processamento calcula o checksum e o armazena junto com o bloco. Em outro momento, o checksum é recalculado e comparado com o valor armazenado. Qualquer desigualdade indica que ocorreu uma alteração indevida dos dados.

Ciclo - Uma seqüência repetitiva de operações ou instruções dentro de um algoritmo. - Parte de uma onda que começando por um ponto de referência (zero), atinge um valor máximo, volta ao nível zero, atinge um valor mínimo, seguido por um retorno ao ponto zero de referência.

Clock - Circuito usado para gerar uma seqüência periódica de pulsos, usado para sincronização de todas as operações em um sistema digital. A freqüência de clock está associada à velocidade na qual os sinais são processados no sistema.

Comando - Informação codificada enviada a um processador digital (computador ou CP), que especifica um conjunto de características de funcionamento que este deve assumir.

Compatibilidade - Propriedade que uma parte de um sistema apresenta de poder ser substituída por outra parte similar, com um mínimo de alterações no funcionamento do sistema como um todo.

Compilador - Um programa específico, ou grupo de programas que traduzem as instruções e comandos de uma linguagem de alto nível em códigos na linguagem de máquina do processador utilizado.

Controlador Programável - Dispositivo ou equipamento que, observando a arquitetura de computadores, possui uma memória programável pelo usuário, e realiza as funções de controle, comando e supervisão de processos industriais; com hardware e software compatíveis com estas aplicações. Também denominado Controlador Lógico Programável - abreviaturas **CP** ou **CLP**.

Controle Distribuído - Arquitetura usada em controle de processos, composta por 2 ou mais Controladores Programáveis interligados a um computador usado como unidade central. Cada CP controla uma parte do processo global e está em constante comunicação com a unidade central que supervisiona e coordena o conjunto. Estruturas de maior porte permitem ainda que essa unidade central esteja conectada a outro computador maior, que assume uma supervisão de nível superior.

Controle PID - Abreviatura para Controle Proporcional, Integral, Derivativo. Método de controle em laço fechado que consiste em aplicar ao atuador o sinal resultante da soma das funções proporcional, integral e derivativa aplicadas ao erro observado na variável medida.

Controle PID Adaptativo - Recurso adicional ao controle PID que utiliza um algoritmo especial para ajustar automaticamente os parâmetros proporcional, integral e derivativo do controlador, de acordo com o histórico de desempenho do processo. Como esses parâmetros dependem fundamentalmente das características físicas do meio, esse tipo de controlador adapta-se ao processo sem necessidade de calibração manual.

Controle PID Feedforward - Método de controle que acrescenta ao controle PID tradicional um fator de antecipação. Basicamente, a mudança em uma condição que terá influência futura na variável medida é considerada na composição da variável de saída, mesmo que a medida ainda não tenha sido afetada.

CPU - Abreviatura para Central Processing Unit. Parte de um sistema digital que contém a memória principal, os circuitos de processamento lógico e aritmético, além de outros circuitos auxiliares. Em um computador ou Controlador Programável, a CPU é a responsável pela interpretação e execução das instruções do usuário.

Dado - Informação codificada na forma digital, armazenada em uma determinada posição de memória, disponível para processamento pela CPU.

Depuração - Processo de pesquisa e eliminação de problemas e erros em um sistema. As fontes de erro consistem em problemas de fiação, partes com funcionamento incorreto ou descalibradas, instruções de programa com sintaxe errada e algoritmos de controle inadequados ao processo ou operação desejada.

Diagrama de Blocos - Método de representação que consiste em subdividir um sistema em várias partes, mostrando a função de cada bloco e apresentando graficamente a relação entre estes blocos.

Diagrama de Estados - Método de representação que apresenta as diversas etapas de um processo em uma montagem gráfica, evidenciando as ações que serão efetuadas em cada etapa, bem como a seqüência dessas etapas (estados) e as condições que permitem que o processo passe de um estado a outro.

Documentação - Todo o conjunto de informações relativas a um equipamento ou sistema, incluindo diagramas elétricos, de fiação, de montagem, listagens de programa, procedimentos de manutenção, instruções de instalação e operação, etc.

Edição -Trabalho de modificar um programa ou documentação no sentido de corrigir um problema, completar ou melhorar o desempenho do sistema.

Encoder - Dispositivo ótico, eletrônico ou eletromecânico, usado para medir e monitorar o movimento rotativo de uma peça. O encoder absoluto fornece um sinal codificado para cada posição do seu eixo. O encoder incremental fornece um determinado número de pulsos para cada unidade de movimento angular no eixo.

Endereço - Código que identifica a posição (localização) de um determinado dado ou programa na memória de um computador ou CP.

Entrada - Dispositivo de um computador ou CP que recebe uma determinada informação elétrica no meio ambiente e a repassa para o processamento na CPU.

Entrada Analógica - Dispositivo de entrada capaz de receber e interpretar um sinal elétrico variável continuamente em uma faixa de tensão ou corrente, onde a informação significativa é transportada através da magnitude dessa tensão ou corrente.

Entrada Digital - Dispositivo de entrada que recebe e interpreta uma informação do meio ambiente através de um código de dois níveis predeterminados de tensão ou corrente.

Eprom - Abreviatura de Erasable Programmable Read-Only Memory. Componente de memória capaz de ser gravada, apagada e regravada novamente. É usada normalmente para armazenar a parte fixa do programa de um CP ou um determinado conjunto de dados que não serão modificados durante a operação.

Estado - Na linguagem descritiva BCM, é o conjunto de comandos que, quando ativado, provoca ações que caracterizam uma etapa do processo. A ativação sucessiva de cada estado determina a seqüência de controle do processo.

E/S - Abreviatura para Entrada/Saída. Refere-se ao conjunto de dispositivos ou instruções de um CP responsáveis pela relação do controlador com as informações do meio ambiente.

E/S Inteligente - Refere-se genericamente a um módulo de entrada ou saída de um CP capaz de pré-processar as informações antes de enviá-las à CPU ou as saídas. Esse recurso é obtido através do uso de um processador localizado no próprio módulo, realizando funções de escala, linearização, cálculos e funções especiais, dispensando a CPU e o programa do usuário dessas atividades.

Flag - Variável introduzida em um programa com o objetivo de sinalizar a execução ou ocorrência de uma determinada ação ou evento. Em outro ponto do programa, essa variável será lida e condicionará a execução de uma instrução ou rotina.

Flash-eprom - Componente de memória capaz de ser gravada, apagada e regravada elétricamente. É usada normalmente para armazenar a parte fixa do programa de um CP ou um determinado conjunto de dados que poderão ser modificados durante a operação.

Fluxograma - Representação gráfica que apresenta as diversas etapas da execução de um programa. Todas as operações e testes do programa estão representados por blocos, onde cada um irá corresponder a uma instrução ou conjunto de instruções no processador.

Full Duplex - Modo de comunicação que permite que dois dispositivos troquem informações, cada um transmitindo e recebendo simultaneamente.

Half Duplex - Modo de comunicação que permite que dois dispositivos troquem informações nas duas direções, porém não simultaneamente, isto é, o dispositivo "A" transmite enquanto o dispositivo "B" somente recebe, e vice versa.

Hardware - Todo o conjunto de partes, dispositivos físicos e componentes que fazem parte de um sistema.

Hexadecimal - Sistema de numeração de base 16. São necessários dezesseis símbolos para representação, sendo que a padronização mais usual utiliza os algarismos "0" a "9" e as letras "A" a "F".

Hierarquia - Listagem ou classificação de um grupo de elementos de acordo com um conjunto de regras pré-determinadas. Refere-se também a ordem de execução ou atendimento de uma série de operações concorrentes.

Instrução - Código simbólico que representa uma seqüência de operações que serão efetuadas pela CPU sobre um conjunto específico de dados, de forma a obter resultados de acordo com regras previamente definidas.

Interface - Área ou dispositivo de ligação entre componentes, circuitos, meios físicos, ou sistemas entre os quais está previsto um intercâmbio de dados, comandos de operação ou energia para alimentação.

Interface Paralela - Dispositivo de interligação entre sistemas que transfere informação de um sistema a outro através de duas ou mais linhas codificadas coletivamente.

Interface Série - Dispositivo de interligação entre sistemas que transfere informação de um sistema a outro através de uma única via, codificada através da seqüência temporal dos sinais significativos.

Interrupção - Bloqueio na execução normal de um programa para atendimento de uma outra rotina considerada mais prioritária. Após o final dessa rotina, o programa volta à execução normal. O momento em que esse bloqueio ocorre é determinado por um sinal externo ao programa.

LCD - Abreviatura para Liquid Cristal Display. Refere-se a um tipo específico de display cujos segmentos refletem a luz seletivamente, controlados por sinais elétricos para a formação dos caracteres e algarismos.

LED - Abreviatura para Light Emitting Diode. Componente semicondutor que emite luz quando polarizado. É usado basicamente como indicador visual em equipamentos eletrônicos.

Linguagem - Conjunto de símbolos e regras que formam um código capaz de permitir a comunicação entre o usuário e um determinado equipamento ou sistema.

LSB - Abreviatura para Least Significant Bit. Bit menos significativo em uma palavra digital.

Matriz - Arranjo de um determinado número de elementos com um significado único. O agrupamento seqüencial desses elementos em linhas e/ou colunas está normalmente associado a uma regra de ocupação determinada.

Memória de Dados - Parte da memória de um computador ou CP reservada para armazenamento das constantes e variáveis do programa, valores usados em operações lógico-aritméticas, valores de contadores e temporizadores, estado corrente do processo, etc.

Memória de Programa - Parte da memória de um computador ou CP onde estão localizadas as instruções de operação, as quais definirão como serão manejados os dados da memória e as entradas externas para fornecer as saídas que o processo sob controle exige.

Memória Retentiva - Parte da memória de dados de um CP que mantem as informações armazenadas independentemente da presença de alimentação no equipamento. Normalmente essa característica é obtida através da alimentação permanente dos dispositivos de memória com pilhas ou baterias.

Mnemônico (Tag) - Abreviatura ou conjunto de símbolos usado para referenciar rapidamente um dispositivo, equipamento, atividade, operação, etc. Normalmente é composto por letras, números ou palavras-chave retiradas da descrição do dispositivo ou operação.

Modem - Abreviatura para Modulador/Demodulador. Equipamento usado para transmissão de dados digitais através de linhas telefônicas ou outros meios físicos, usando princípios de modulação dos sinais em fase ou freqüência.

Módulo - Parte ou elemento de um equipamento ou sistema capaz de ser separado do conjunto mantendo determinadas características próprias. Em hardware entende-se por módulo, uma parte que possa ser separada do equipamento, testada ou substituída individualmente. O mesmo conceito aplicado ao software se refere a partes de um programa que podem ser descritas, desenvolvidas, depuradas em separado e, posteriormente integradas ao sistema completo.

MSB - Abreviatura para Most Significant Bit. Bit mais significativo em uma palavra digital.

Multiplex - Ou multiplexação do processo que permite que dois ou mais sinais distintos sejam transmitidos em um mesmo canal de comunicação. Dois processos fundamentais são utilizados: a multiplexação no tempo divide os sinais em fatias no tempo, transmitindo uma amostra de cada sinal durante um período e numa seqüência determinada. A multiplexação em freqüência é conseguida fazendo com que cada um dos sinais module uma freqüência portadora diferente no transmissor. A soma dos sinais é enviada pela linha. No receptor, cada freqüência é filtrada, separada e os diversos sinais podem ser demodulados separadamente.

Nível Lógico - Valores de tensão ou corrente que podem ser associados aos dois estados possíveis em uma linha digital: "0" ou "1".

Operando - Valor numérico sobre o qual uma operação lógico- aritmética será efetuada.

Palavra - Grupo de dois ou mais bytes associados, tratados como uma unidade, com significado coletivo próprio.

Paridade - Ou bit de paridade. Bit não significativo adicionado a um byte digital, usado para detectar erros de transmissão desse byte de um meio a outro. Para um sistema funcionar é necessário que haja um dispositivo ou rotina que gere o bit de paridade correto (0 ou 1) e outro dispositivo que leia o byte completo, incluindo o bit de paridade, determinando se a paridade da palavra está correta ou não. O sistema, é dito de paridade par quando o número de bits"1" da palavra completa for par; e é dito de paridade ímpar, quando o número de bits "1" for ímpar.

Periférico - Dispositivo ou equipamento conectado a um computador ou CP para executar uma função específica, sem que possa ser considerada parte integrante do equipamento básico.

PID - Ver Controle PID.

Pointer - Ver Apontador.

Porta - Ponto de conexão de um equipamento ou sistema para intercâmbio de informações do sistema com o meio ou ligação de periféricos.

Preset - Valor predeterminado lançado na memória de dados como valor inicial para a partida de um processo ou rotina.

Protocolo - Conjunto de regras e convenções que especificam as características, formatos, níveis lógicos, temporização, etc., necessários para viabilizar a comunicação entre sistemas ou entre partes dentro de um sistema.

PROCP - É uma ferramenta usada no desenvolvimento de programas para os CP´s BCM. Possui versão para ambiente DOS e WINDOWS. Está disponível no mesmo ambiente ferramentas para editar, compilar, carregar e monitorar programas.

RAM - Abreviatura para Random Acess Memory. Componente de memória que permite o acesso aos dados armazenados para leitura ou escrita em um determinado tempo especificado, independentemente da localização desse dado na memória.

Realimentação - Sinal enviado pelo processo sob controle ao CP, indicando a resposta deste aos comandos de atuação aplicados pelo controlador.

Reset - Ação de fazer retornar um processo ou variável a uma condição inicial predeterminada.

Rotina - Série de instruções ou operações que executam uma tarefa específica.

Ruído Elétrico - Sinais elétricos indesejáveis em um sistema, capazes de perturbar ou modificar o funcionamento normal deste. A imunidade ao ruído é definida como a medida da capacidade de funcionamento do sistema na presença de um nível determinado de ruído elétrico.

Saída - Dispositivo de um computador ou CP que transforma os comandos enviados pela CPU em níveis elétricos adequados, capazes de atuar sobre o meio externo.

Saída Analógica - Dispositivo de um controlador programável que transforma o conteúdo de uma variável de programa em um valor de tensão ou corrente, disponível para atuação sobre um circuito externo. A magnitude da tensão ou corrente está associada ao valor da variável de acordo com uma regra pré-estabelecida.

Saída Digital - Dispositivo de um controlador programável que fornece uma tensão, corrente ou fecha um contato de acordo com o conteúdo de uma variável ou um comando específico do programa. A tensão ou corrente de saída pode assumir apenas dois níveis, correspondentes as situações de saída ligada ou desligada.

Sistema - Arranjo de componentes, dispositivos e/ou equipamentos ligados entre si como uma unidade organizada, capaz de executar um conjunto de funções e tarefas bem determinadas.

Sistema Operacional - Programa básico de um CP ou computador que controla a distribuição de memória, acesso aos periféricos, supervisão do sistema e outras funções trabalhadas independentemente do programa do usuário. O sistema operacional pode incluir ainda as rotinas básicas que desenvolvem as instruções disponíveis em um equipamento programável.

Software - Todos os programas, rotinas, sistemas operacionais e documentação participantes e relativos a um sistema que não possam ser considerados como partes físicas integrantes deste.

Sub-rotina - Parte de um programa ou rotina com uma função específica, armazenada em separado na memória de programa e que pode ser chamada e executada em vários pontos do programa, tantas vezes quanto necessário.

Transição - Na linguagem descritiva BCM, é a combinação de eventos que condiciona uma mudança de estado, isto é, a transferência da atividade de um estado para outro.

Varredura - Seqüência cíclica de operações dentro de um CP que engloba o atendimento dos periféricos, desenvolvimento das operações relativas aos estados correntes, inspeção das condições de transição, interface com o operador, supervisão do sistema, etc.

Watchdog - Tipo especial de temporizador, usado para monitorar o funcionamento correto de um controlador. O programa do CP gera um sinal alternado permanente para o circuito Watchdog, com um período máximo especificado. Caso haja uma falha do processador, o sinal alternado pára; nesse caso, o circuito Watchdog se encarrega de tomar uma providência que proteja o processo, acionando um alarme, desligando ou reinicializando o controlador.

2 - A LINGUAGEM DESCRITIVA BCM

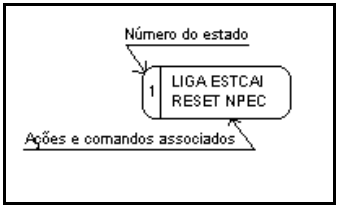
O Diagrama de Estados é basicamente uma forma padronizada de representação das regras de controle de um processo. Utilizando uma estrutura similar àquela adotada na eletrônica e na matemática, conhecida como "Diagrama de Variáveis de Estado", é possível descrever operações combinacionais ou seqüenciais, executadas dentro de um fluxo único ou em múltiplas ramificações paralelas e simultâneas. A forma gráfica adotada permite uma fácil visualização do conjunto, recurso fundamental na fase de projeto e especificação, oferecendo ao mesmo tempo uma forma exata para descrição de cada operação, viabilizando a programação do controlador a partir do Diagrama de Estados.

É importante ressaltar que qualquer programa para controle de processos digitais ou analógicos pode ser facilmente representado por um Diagrama de Estados. Além disso, a utilização do Diagrama de Estados elimina a necessidade de qualquer outro método de descrição do processo.

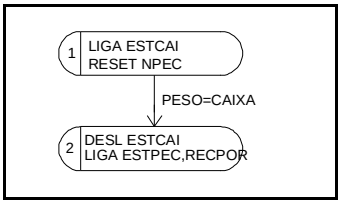
2.1 - ESTRUTURA DA LINGUAGEM DESCRITIVA BCM

A estrutura da Linguagem Descritiva BCM é formada por três elementos básicos: os ESTADOS, as TRANSIÇÕES e a MALHA DE CONTROLE:

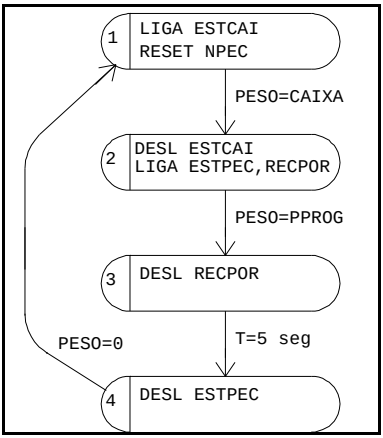
O ***Estado*** é o conjunto de comandos que, quando ativado, provoca ações que caracterizam uma etapa ou evento do processo. A ativação sucessiva dos estados determina a seqüência de controle do processo. Nos diagramas, o estado é representado por uma elipse. No canto esquerdo será colocado o número do estado. No espaço da direita serão descritas as ações efetuadas ou os comandos ativos para esse estado.



Uma condição de ***Transição*** é a combinação de eventos que determinam uma mudança de estado, isto é, a transferência da atividade de um estado para outro. No diagrama, a transição é representada por uma seta orientada que liga o estado de origem com o estado para a transferência. Sobre a seta são descritas as condições válidas para essa transição.



Uma ***Malha de controle*** é um diagrama de estados parcial, que corresponde a um ramo do processo, dentro do qual apenas um estado pode estar ativo a cada momento. O diagrama de estados completo do processo pode ser composto por uma ou mais malhas. Devido aos recursos de multiprocessamento do controlador, todas as malhas serão atendidas simultaneamente quando da execução do programa em tempo real.



O ***Diagrama de Estados*** apresenta de forma gráfica uma estrutura lógica completa através de um conjunto de malhas de controle. Cada malha assume o controle de uma parcela determinada do processo. O Controlador Programável oferece recursos para atendimento simultâneo de diversas malhas. Este recurso de processamento multitarefas é muito importante no desenvolvimento de soluções de controle para os sistemas reais e de difícil representação pela maioria das outras formas de documentação.

2.2 - REGRAS DE PROGRAMAÇÃO NA LINGUAGEM DESCRITIVA

O diagrama de estados é uma forma padronizada de representação das regras de controle de um processo. Utilizando uma estrutura similar a que é adotada na eletrônica e na matemática, conhecida como "Diagrama de Variáveis de Estado", é possível descrever operações combinacionais ou seqüenciais, executadas dentro de um fluxo único ou em múltiplas ramificações paralelas e
Simultâneas.

A forma gráfica adotada permite uma fácil visualização do conjunto, recurso fundamental na fase de projeto e especificação, oferecendo ao mesmo tempo uma forma exata para descrição de cada operação, viabilizando a programação do controlador a partir do Diagrama de Estados.

A seguir serão apresentadas, de forma sistemática e mais detalhada, as regras fundamentais para elaboração, análise e execução pelo controlador, adotadas na programação através de diagramas de estado:

F Regra 1

O estado é sempre representado por uma elipse, dividida em dois campos: No lado menor, a esquerda, é colocado o número de referência do estado; no lado direito são incluídos todos os comandos e ações ativos para esse estado.

F Regra 2

Os estados no diagrama são referenciados por um número situado na faixa entre 0 e 255. Cada malha recebe uma seqüência de numeração independente.

F Regra 3

A numeração adotada para os estados é usada apenas para referência. Não existe nenhum sentido de seqüência ou prioridade de execução pelo CP na numeração dos estados. A única consideração especial é que o controlador ao ser ligado, inicia a execução do programa sempre pelo estado 0 de cada malha.

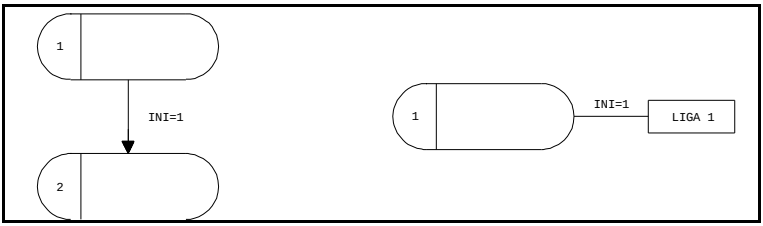
F Regra 4

Somente um estado pode estar ativo de cada vez num mesmo Diagrama de Estados. Apenas os comandos e operações desse estado serão executadas nesse momento. Além disso, apenas as condições de transição do estado ativo serão consideradas pelo controlador.

F Regra 5

Associado a cada estado são aceitas duas formas condicionais: As condições para transição de estado especificam as relações que, quando cumpridas fazem que o controlador transfira a atividade para o estado de destino; As condições para execução imediata fazem que as ações especificadas sejam executadas dentro do estado ativo apenas se as relações lógicas mencionadas forem válidas.

Transição de estado **Execução imediata**



F Regra 6

A transição é representada por uma seta reta ou curva, indicando o sentido da transferência da atividade quando do cumprimento das condições de transição.

F Regra 7

As operações e comandos associados à condição de execução imediata são representados dentro de um retângulo, ligado por uma linha ao estado.

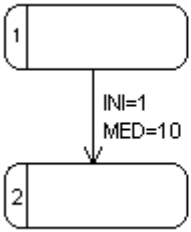
F Regra 8

O programa ignora todos os estados e transições executados, somente considerando as transições com origem no estado ativo. O programa só considera válidas as transições com origem no estado ativo, ignorando todas as demais. Assim, podemos dispensar a preocupação com o acionamento de entradas fora da seqüência desejada.

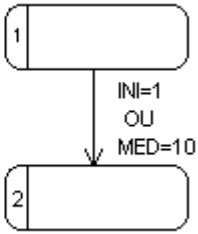
F Regra 9

São aceitas múltiplas condições de transição de um estado para outro. Quando duas ou mais expressões lógicas estão colocadas sobre uma mesma linha de transição, deve-se considerar que a transição ocorrerá quando todas as expressões forem válidas (ligação "E"). Para que a transição ocorra com apenas uma das expressões válidas (ligação "OU"), é necessário que a palavra "OU" apareça entre as expressões.

Ligação "E"

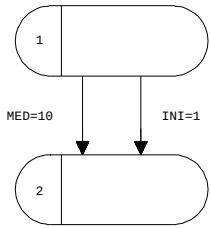


Ligação "OU"



F Regra 10

São aceitas duas ou mais linhas de transição entre um estado origem e um destino. Neste caso a transição ocorrerá quando uma das expressões for válida (corresponde a ligação "OU").

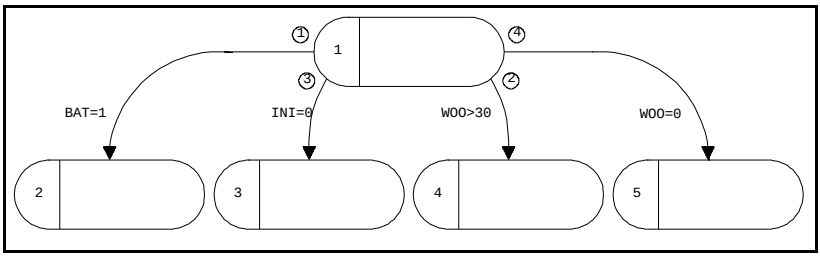


F Regra 11

O número de transições com origem em cada estado é ilimitado. No entanto um número muito grande de transições pode ocasionar uma falha por WATCH DOG. O programa pode seguir diversos caminhos a partir de um estado; o caminho a seguir será decidido pela última condição de transição aprovada.

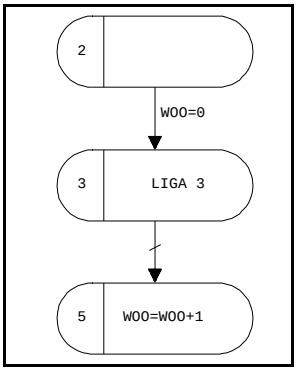
F Regra 12

No caso previsto pela regra anterior, é importante definir o que acontece se duas ou mais condições estiverem simultaneamente válidas. Quando a caracterização da ordem de prioridades for importante, será representada na forma a seguir: - (1) para o mais prioritário; (2) para a segunda prioridade; e assim por diante.



F Regra 13

Em determinadas situações dentro de um programa é necessário ou conveniente definir transições entre estados sem condição especificada. No caso do exemplo, quando o estado 3 fica ativo, a saída 3 é ligada; Imediatamente a seguir, o estado ativo passa a ser o 5, provocando a execução imediata da operação "W00=W00+1". No programa editado, esse tipo de transição é tratado pela instrução "VA PARA". A figura mostra o trecho-exemplo, salientando a forma de representação gráfica.



F Regra 14

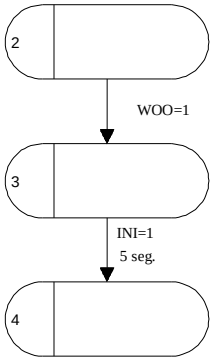
Outro tipo de condição de transição é a chamada "Condição Temporizada". Neste caso, ao invés de combinações lógicas de entradas ou variáveis, a transição está condicionada a passagem de um tempo. O tempo necessário para a transição aparece ao lado da linha no diagrama. No programa editado, esse tipo de transição é tratado pela instrução "ATRASO".

F Regra 15

Frequentemente aparece a necessidade de combinar uma condição temporizada com um ou mais testes de variáveis. O trecho do diagrama de estados abaixo mostra esse caso. O programa vai para o estado 4 depois de passar 5 segundos e se a entrada "INI" estiver acionada. Observar que:

- Os 5 segundos são contados a partir da entrada no estado 3, independentemente da situação de "INI".

- Se a entrada "INI" for acionada 5 segundos ou mais após a entrada no estado 3, a transição ocorrerá imediatamente.



F Regra 16

Cada malha componente do sistema recebe um número de referência. A primeira malha será a 0, a segunda será a 1 e assim por diante, até um número máximo de 64 malhas.

F Regra 17

Cada estado é sempre pertencente a uma e somente a uma malha. Além disso, um estado ativo com uma condição de transição válida transfere a atividade para o estado indicado na transição (estado de destino); Esse estado de destino pertencerá necessariamente a mesma malha do estado anterior.

F Regra 18

Do ponto de vista do diagrama de estados, cada malha é considerada um processo independente. Não pode haver transições de um estado de uma malha para outro em outra malha.

2.3 - COMO PROGRAMAR NA LINGUAGEM DESCRITIVA

Ü As caixas de diálogo que aparecerão ao longo do capítulo servem para situar/orientar o usuário sobre onde encontrar outras informações a respeito de determinado tópico.

A etapa seguinte à elaboração do diagrama de estados é a edição do programa do **CP**, propriamente dito.

A tecnologia empregada nas CPUs BCM permite a programação direta, isto é, o programa do usuário é transferido diretamente do microcomputador para a memória do CP através do canal serial RS 232.

Os recursos necessários para a edição e transferências do programa para o CP são:

· Computador padrão PC com Windows XP, Vista ou Windows 7
· Ambiente de desenvolvimento BCM PROCP
· Cabo de comunicação serial RS232 (específico para CPs BCM)

A seguir estão descritos os comandos e instruções para programação em linguagem descritiva BCM, segundo a sua função dentro do programa do usuário.

Sempre que for necessário, será apresentado um exemplo de aplicação para facilitar a compreensão e a utilização do(s) respectivo(s) comando(s) e/ou instrução(ões).

2.4 - SINTAXE DAS INSTRUÇÕES E COMANDOS

Para cada comando ou instrução, será apresentada a forma sintática, a descrição completa da operação e o conjunto de parâmetros e opções disponíveis. Sempre que for conveniente, serão incluídos exemplos procurando esclarecer o funcionamento da forma mais clara e precisa possível.

Em geral, o funcionamento dos comandos e instruções é idêntico para todos os modelos de Controladores Programáveis BCM. Sempre que houver diferenças, o texto chamará a atenção para todos os casos possíveis. Naturalmente, a aplicabilidade de cada recurso está associada a disponibilidade física desse recurso no modelo ou versão do Controlador Programável em questão.

- Os sinais "igual", "pontos", "vírgulas" e espaços em brancos devem ser seguidos rigorosamente na edição. Caso contrário, serão interpretados incorretamente pelo compilador.

- Os elementos representados em letras minúsculas significam constantes ou variáveis que devem ser introduzidas nas posições citadas, de acordo com o que consta em cada descrição.

- Os elementos entre colchetes "[ ]", são opcionais na composição da linha de programa. A sua inclusão ou não produz diferentes efeitos na operação do comando ou instrução.

- Os elementos entre chaves "{ }" são alternativos na composição da linha de programa. Dentro dos colchetes aparecem dois ou mais elementos entre colchetes. Para o funcionamento adequado, um dos elementos entre colchetes deve ser escolhido pelo programador. A inclusão de cada tipo de elemento produz um determinado efeito na operação do comando ou instrução, conforme descrito.

2.4.1 - Comandos de Configuração do CP

Estes tem a finalidade de especificar parâmetros, habilitar blocos funcionais do sistema operacional e instruir o compilador em relação a determinados procedimentos.
Com a excessão dos comandos **`CONTINUA' e `FIM'**, os demais comandos de configuração devem ser sempre colocados no início do programa, antes da primeira definição de malha.
O programador deve prestar especial atenção no momento da entrada destes comandos. A omissão da definição de um periférico faz com que este não opere; Parâmetros incorretos podem levar a erros de compilação ao longo do programa ou erros de execução no controlador.

- ***BCD=x***

  Este comando, válido apenas para o controlador BCM2085 com interface BCD, habilita a rotina de tratamento de chaves Thumbwheel. O parâmetro `x' define o número de chaves BCD por grupo. São aceitos os valores 2 ou 4.

- ***CLP=tipo[([ESD=X][ESA=Y][ESH=Z][ESP=1][VAF=1])]***

  Define o tipo de controlador programável BCM no qual o programa será aplicado. Este comando é obrigatório e deve se sempre o primeiro no programa. Os seguintes tipos estão disponíveis:

BCM2085	Configura o sistema para compilação, gravação de EPROMs e transferência para flash-Eprom nos controladores BCM2085 que usem a CPU3, CPU4 ou CPU4A.
CPU6	Configura o sistema para compilação de programas que serão usados no BCM2085/CPU6.
CPU6A	Configura o sistema para compilação de programas que serão usados no BCM2085/CPU6A.
CPU7	Configura o sistema para compilação de programas que serão usados no BCM2085/CPU7.
CPU7-10	Configura o sistema para compilação de programas que serão usados no BCM2085/CPU7-10. Válido para PROCP V4K ou superior.
CPU7-11	Configura o sistema para compilação de programas que serão usados no BCM2085/CPU7-11. Válido para PROCP V4K ou superior.
CPU8	Configura o sistema para compilação de programas que serão usados no BCM2085/CPU8. Válido para PROCP V4K 1.2 ou superior.

CPU8A	Configura o sistema para compilação e transferência de programas para o BCM2085B/CPU8A. Válido para PROCP V5K 3.0 ou superior.
CPU9	Configura o sistema para compilação e transferência de programas para o BCM2085B/CPU9. Válido para PROCP V5K 4.0 ou superior.
BCM1086	Configura o sistema para compilação, gravação de EPROMs e transferência para flash-Eprom nos controladores BCM1086 e BCM1086A.
BCM1088	Configura o sistema para compilação e transferência de programas para o controlador BCM1088.
BCM3011	Configura o sistema para compilação e transferência de programas para o controlador GP3000 (GP3011 e GP3020). Válido para o PROCPV3K ou superior.
GP1500	Configura o sistema para compilação e transferência de programas para o controlador GP1500. Válido para o PROCPV5K 5.0 ou superior.
GP3101	Configura o sistema para compilação e transferência de programas para o controlador GP3101. Válido para o PROCPV4K ou superior.
GP3001	Configura o sistema para compilação e transferência de programas para o controlador GP3001. Válido para o PROCPV4K 1.2 ou superior.
GP3005	Configura o sistema para compilação e transferência de programas para o controlador GP3005. Válido para o PROCPV4K 1.2 ou superior.
GP3006	Configura o sistema para compilação e transferência de programas para o controlador GP3006. Válido para o PROCPV4K 1.2 ou superior.
GP3009	Configura o sistema para compilação e transferência de programas para o controlador GP3009. Válido para o PROCPV4K 1.2 ou superior.
GP3020	Configura o sistema para compilação e transferência de programas para o controlador GP3020. Válido para o PROCPV4K 1.2 ou superior.
GP3210	Configura o sistema para compilação e transferência de programas para o controlador GP3210. Válido para o PROCPV5K 4.0 ou superior.
GP3220	Configura o sistema para compilação e transferência de programas para o controlador GP3220. Válido para o PROCPV5K 4.0 ou superior.

As opções entre parênteses são utilizadas para os controladores GP3011, GP3101, GP3210 e GP3220, definindo a configuração de módulos de entradas e saídas:

ESD	O parâmetro x (1 a 4) define o número de módulos GP3ESD.
ESA	O parâmetro y (1 a 4) define o número de módulos GP3ESA.
ESH	O parâmetro z (1 a 4) define o número de módulos GP3ESH.
ESP	Declara que se será usado o módulo ESP. Somente um módulo pode ser usado por controlador.
VAF	Declara que se será usado o módulo ESP. Somente um módulo pode ser usado por controlador.

Observação: o número máximo total de módulos é 4. Se um tipo de módulo não for usado este deve ser omitido.

ü *CONTINUA: (Narq.BCM)*

Esse comando permite que um programa aplicativo BCM seja dividido em vários arquivos.

Narq.BCM deve ser o nome de um arquivo com extensão .CLP ou .BCM localizado no mesmo diretório. O compilador, ao encontrar esse comando, interrompe a compilação e continua a partir do início do novo arquivo.

É usado principalmente para programas longos, que excedem a capacidade de memória do PROCP. Nestes casos, divide-se o programa em várias partes, com nomes de arquivo diferentes, usando o comando `CONTINUA' para uní-los no momento da compilação.

A partir do PROCP versão V5K, a limitação de 64kbytes por arquivo fonte em linguagem descritiva deixou de existir. A utilidade do comando CONTINUA permanece para organizar diferentes partes de um aplicativo longo.

Exemplo:

```
;Arquivo: "TESTE1.BCM"
MAQUINA 0:
ESTADO 0:
FACA MIN=0 E HOR=0
VA PARA 1
 "          "          "          "
ESTADO 30:
FACA* W00=SEC E W88=58888 E W89=HOR E W90=MIN
FACA* W01=AN1 E W02=AN2 E W03=AN3 E W04=AN4 E W05=AN5 E W06=AN6
CONTINUA:(TESTE2.BCM)

;Arquivo: "TESTE2.BCM"
MAQUINA 1:
ESTADO 0:
SE I01=1 OU I13=1 OU TF1=1 LIGA 1
SE I01=0 E I13=0 E TF1=0 DESL 1
 "          "          "          "
ESTADO 13:
SE V04<32768 ENTAO 12
VA PARA 0
```

ü *CTR[*]=x*

Este comando habilita as rotinas de tratamento dos contadores rápidos no BCM2085B (módulo CTR), no GP3000 (módulo ESA ou CTR) ou no BCM1086A (expansão interna EI3). O parâmetro `x' define o número de placas utilizadas no sistema. Deve ficar na faixa 1 a 5. Outro valor provocará erro na compilação. É importante observar que se este comando não estiver presente, as placas não serão acessadas e as respectivas variáveis assumirão valores indeterminados.
O uso de CTR=5 no controlador BCM2085B só é possível se o comando PAN não estiver habilitado. Esta restrição é determinada pelas limitações do espaço de endereçamento no BCM2085B.
O sinal opcional [*] informa ao sistema operacional que todas as placas CTR devem ser atendidas em cada Ciclo de Varredura. Sem o sinal, apenas uma placa será atendida em cada ciclo. Esta opção não tem efeito no BCM1086A**.**

ü *DIM aaa(xxx)*

Define a dimensão da matriz numérica **`aaa'**. O índice máximo da matriz, configurado em **`xxx'** pode estar na faixa 0 a 65535. O número total de elementos da matriz será esse valor **`xxx'** mais um.

ü *DIMD[n] aaa(xxx)*

Define que a matriz **`aaa'** será uma matriz de display, isto é, a cada elemento da matriz de 0 a 99 corresponderá um canal do display. O valor `n' opcional especifica a unidade de display e pode ser 1, 2 ou 3. Se omitido, o sistema assume a unidade 1. O valor **`xxx'** não terá importância; uma matriz de display possui sempre a dimensão 99.
O programa do usuário pode trabalhar livremente com uma matriz de display (testes, atribuição de valores, etc.), sem prejuízo das variáveis definidas como variáveis de display no menu **"EDITA"**, as quais podem continuar a ser usadas normalmente.

ü *DIMM aaa(xxx)*

Define a dimensão da matriz numérica **`aaa'**. O índice máximo da matriz, configurado em **`xxx'** pode estar na faixa 0 a 65535. O número total de elementos da matriz será esse valor **`xxx'** mais um. A diferença para a instrução `DIM' é que `DIMM' aloca a matriz na memória RAM da placa BCM2085/MEM, versão `RIM'. Apenas controladores com essa placa instalada aceitam esse comando.

ü *DIMX aaa(xxx)*

Declara a matriz **`aaa'**, alocando as suas variáveis na área de memória Ram a partir do endereço 10000h. O índice máximo da matriz, configurado em **`xxx'** pode estar na faixa 0 a 65535. O número total de elementos da matriz será esse valor **`xxx'** mais um.
Este comando pode ser usado para os CLPs modelo CPU8A, CPU9, GP3210 e GP3220, os quais possuem área de Ram superior a 10000h. A utilidade básica é permitir que, nas aplicações que usem muita memória, as variáveis livres não matriciais tenham espaço para ser alocadas pelo PROCP abaixo do endereço 10000h. Isto é importante porque várias instruções não admitem variáveis cujos endereços estejam alocados acima de 10000h.

Atenção: Nem todas as instruções da Linguagem Descritiva podem ser usadas com variáveis alocadas através de ***DIMX***! As seguintes instruções podem ser usadas: ***VA PARA, SE, PRESET, COPIA, FACA, SET, RESET, INCREMENTA*** e ***DECREMENTA.***

Exemplo:

```
DIMX MGDE1(20000)      ;Matriz alocada em 10000h
DIMX MGDE2(20000)      ;Matriz alocada em 19C42h
DIMX MGDE3(20000)      ;Matriz alocada em 23884h
DIM MAT(50)            ;Matriz alocada em 1A00h

MALHA 0:
ESTADO 0:
FACA VAR=0             ;Variável livre alocada em 1A66h
MOSTRA T(1,5) VAR      ;As variáveis usadas em MOSTRA devem ter endereço inferior
                       ;a 10000h!
```

ü *DISPLAY=xx.yyy*

Habilita a operação dos displays numéricos no BCM2085 ou BCM1086. O valor **[xx]** indica o número do último canal reservado para o display, na faixa 1 a 99. O valor **[yyy]** indica o máximo admissível no campo de valor dos displays, na faixa 1 a 65535. A colocação pelo programa de um valor fora da faixa configurada provoca o zeramento do valor, quando o canal correspondente for selecionado.

Exemplo: DISPLAY=12.680

Neste caso, se utilizarão 12 canais, possibilitando a entrada ou apresentação de valores dentro dos limites 0 a 680.

O Sistema Operacional permite a instalação de até três módulos de display numéricos programáveis pelo usuário e mais um de serviço. O comando DISPLAY habilita todos os módulos, mas os limites são válidos apenas para os módulos do usuário.

ü *DISPLAY=DPA[20]*

Habilita a operação do display alfanumérico no BCM2085, BCM1086 ou GP3000. O controle da apresentação de mensagens e valores no display é totalmente feito pela instrução MOSTRA. As seguintes observações são importantes quando da elaboração do programa:

- Ÿ Ao utilizar o comando DISPLAY=DPA, não podem existir variáveis definidas nas unidades de display 2 e 3.
- Ÿ Ao aparecer uma instrução "MOSTRA" no programa, é obrigatório que tenha sido colocado um comando DISPLAY=DPA no início do programa. Caso contrário, o compilador indica erro.
- Ÿ A opção **"DPA20"** é usada quando o display alfanumérico é de 20 colunas. Caso contrário, o compilador assume que o display possui 16 colunas.

ü *ESA=x*

Este comando habilita a rotina de tratamento de entradas e saídas analógicas nos CPs BCM1086A ou no BCM2085B com módulos ESA. É importante observar que se este comando não estiver presente, as entradas e saídas não serão acessadas e as respectivas variáveis assumirão valores indeterminados.
No controlador BCM2085B, o parâmetro **`x'** define o número de módulos ESA utilizadas no sistema. Se houver módulos BCM2085/EAN ou /EAI instalados, o valor **`x'** pode estar na faixa 1 a 4. Se não houver módulos EAN ou EAI, **`x'** pode estar na faixa 1 a 8.
No controlador BCM1086A, o comando ESA com **`x'** igual ou maior que 1 habilita o funcionamento das entradas e saídas analógicas nas expansões internas EI1, EI2 e EI4.

ü *FIM*

Instrui ao compilador para encerrar a compilação nesse ponto, independentemente da continuação do programa no arquivo. É usado principalmente durante a depuração, para isolar partes do programa que atrapalham o teste ou partes nas quais se suspeita haver erros.

ü *IFT=x*

Este comando habilita a rotina de tratamento de entradas analógicas de termopares na placa IFT do controlador BCM2085B. O parâmetro **`x'** define o número de placas utilizadas no sistema. Deve estar na faixa 1 a 8. Outro valor provocará erro na compilação. É importante observar que se este comando não estiver presente, as placas não serão acessadas e as respectivas variáveis assumirão valores indeterminados.

ü *INTERFACEi [(nnnn)]=yyy [DE zzz] [(PROTOCOLO=xxx TEM=m TOC=n TOR=o TOF=p NSE=q PAP=S/N BUF=&vvv PORTA=kkk)][EI]*

Configura os canais de comunicação do controlador programável - conforme o tipo de controlador, este pode ter de 1 a 4 canais, em variados padrões elétricos. Deve ser usada uma linha ***INTERFACE*** para configurar cada um dos canais disponíveis.

O parâmetro “i” é o número do canal de comunicação, dentre os disponíveis no CLP.

O valor “nnn” define a velocidade. Para o protocolo BCM1 a velocidade de comunicação pode ser 1200bps ou 9600bps. Para o protocolo BCM2 as seguintes velocidades são admitidas: *1200, 9600, 19200, 38400, 48000.* Nem todas as velocidades são suportadas por todos os produtos! Consulte a documentação específica de cada produto para verificar as velocidades aceitas!

O valor “yyy” é o código de acesso tal como será reconhecido pela outros integrantes da rede. O código permitido deve estar na faixa de 1 a 254.

Quando usado o protocolo BCM2, apenas um dos dois canais de comunicação do CP pode fazer a passagem de ficha (definido pela expressão [DE zzz]). Sobre este canal é que serão executadas as instruções LE/ESCREVE. O valor zzz especifica o número total de equipamentos ativos previstos para a rede.
Ver o *Manual de Protocolos e Redes de Comunicação BCM* para maiores informações.

O parâmetro “xxx” define o protocolo de comunicação que o canal deve assumir. A disponibilidade de cada protocolo depende do modelo de controlador programável.

As opções disponíveis em *PROTOCOLO* estão descritas no quadro a seguir. Consulte a documentação específica de cada modelo para mais detalhes da configuração para uso:

Produto	Canal 1	Canal 2	Canal 3	Canal 4
BCM1086A	BCM1 BCM2	BCM1 BCM2		
BCM1088	BCM1 BCM2	BCM1 BCM2		
CPU2085	BCM1 BCM2	BCM1 BCM2		
CPU6	BCM1 BCM2	BCM1 BCM2		
CPU6A	BCM1 BCM2	BCM1 BCM2		
CPU7	BCM1 BCM2	BCM1 BCM2		
CPU7-11	BCM2	BCM2		
GP3001	BCM2	BCM2		
GP3005	BCM2			
GP3006	BCM2			
GP3009	BCM1 BCM2	BCM1 BCM2		
GP3020	BCM1 BCM2	BCM1 BCM2		
GP3011	BCM1 BCM2	BCM1 BCM2		
GP3101	BCM2 MODBUSESCRAVO MODBUSMESTRE GPRS	BCM2 MODBUSESCRAVO MODBUSMESTRE	BCM2 MODBUSTCP GAT232 VEL=XX GAT485 VEL=XX BCM2MESTRE	
GP3210 & GP3220	BCM2 MODBUSESCRAVO MODBUSMESTRE GPRS	BCM2 MODBUSESCRAVO MODBUSMESTRE	BCM2TCP ou BCM2 MODBUSTCPMESTRE MODBUSTCPESCRAVO ou MODBUSTCP	
CPU8	BCM2	BCM2	BCM2	BCM2
CPU8A	BCM2 NMEA (GPS) MEINBERG(GPS) MODBUSESCRAVO MODBUSMESTRE	BCM2 MODBUSESCRAVO MODBUSMESTRE MODBUSTCP	BCM2 NMEA (GPS) MEINBERG(GPS) MODBUSESCRAVO MODBUSMESTRE	BCM2 MODBUSESCRAVO MODBUSMESTRE
CPU9	BCM2 NMEA (GPS) MEINBERG(GPS) MODBUSESCRAVO MODBUSMESTRE	BCM2 MODBUSESCRAVO MODBUSMESTRE	BCM2 NMEA (GPS) MEINBERG(GPS) MODBUSESCRAVO MODBUSMESTRE	BCM2TCP MODBUSTCPMESTRE MODBUSTCPESCRAVO

O parâmetro “m” (em TEM) define o tempo de espera gerado antes do envio de cada mensagem. Válido para o protocolo BCM2 e para o protocolo Modbus em alguns produtos. Pode variar de 0 a 2550 em passos de 10. A unidade é ms. Por omissão é assumido o valor 0.

O parâmetro “n” (em TOC) define o time-out entre caracteres para descartar uma mensagem em curso. Válido somente para o protocolo BCM2. Varia de 0 a 2550 (ms) em passos de 10. Se o valor programado for zero as mensagens nunca serão descartadas por time-out entre caracteres. Por omissão é assumido o valor 0.

O parâmetro “o” (em TOR) define o time-out para resposta de uma solicitação (para considerar que o destino não respondeu). Válido somente para o protocolo BCM2. Pode variar de 10 a 2550 (ms) em passos de 10. Por omissão é assumido o valor 500ms.

O parâmetro “p” (em TOF) define o time-out para criação de ficha. Válido somente para o protocolo BCM2. Varia de 0 a 2550 (ms) em passos de 10. Se o valor for 0 não cria ficha. Por omissão, é igual a (960+40*(endereço do CP))ms.

O parâmetro “q” (em NSE) define o número de série do produto. Válido somente para o protocolo BCM2. Pode variar 1 a 65535. Por omissão assume o valor préviamente gravado em Eeprom.

A opção PAP habilita ou não a alteração de parâmetros on-line. Válido somente para o protocolo BCM2. PAP=S permite a alteração e PAP=N não permite). Por omissão assume o valor PAP=S.

O parâmetro “*vvv*” (em *BUF)* define o endereço do buffer dos parâmetros de configuração/monitoração do respectivo protocolo. Em “***vvv***” deve ser colocado o nome da matriz usada para transportar esses parâmetros.

O parâmetro “kkk” (em *PORTA)* define a porta de conexão quando a comunicação é via canal ethernet TCP/IP. Pode variar de 1 a 65535. Quando existir esta opção, será obrigatória a programação dos parâmetros de rede através da instrução TCP. Se omitido, assume as seguintes portas, conforme o protocolo:

- BCM2: Porta 501
- MODBUSTCP: Porta 502
- GAT232: Porta 23
- GAT485: Porta 24

A opção **“EI”** indica que o controlador BCM1086A deve usar o canal serial da expansão interna, ao invés do canal serial do módulo principal.

Observações:

- Caso mais de um canal utilize o protocolo BCM2, os cinco parâmetros programáveis serão sempre iguais (TEM, TOC, TOR, TOF, NSE e PAP) para todos os canais. Se houverem definições contraditórias em diferentes linhas ***INTERFACE***, valerá a última definição.

- A opção "***BUF=&vvv***" somente está disponível e é obrigatória para os protocolos NMEA, MEINBERG, MODBUSMESTRE, MODBUSESCRAVO e MODBUSTCP. As demais opções (anteriores) somente estão disponíveis para o protocolo BCM2, nos canais 1 e 2 (seriais).

- Este conjunto completo de opções do comando ***INTERFACE*** é válido quando usado o PROCP V5k, versão 4 ou superior. Consulte a BCM para as opções disponíveis quando for usada uma versão anterior do PROCP!

Exemplo:

INTERFACE2(38400)=2 DE 10 (PROTOCOLO=BCM2 TEM=100 TOC=30 TOR=100 TOF=1250 NSE=1234 PAP=N)

Configura o canal 2 do controlador para o protocolo BCM2. É usada a velocidade de 38,4kbps e este CLP recebe o código 2 de uma rede com 10 CLPs.

Configurações específicas para o canal Ethernet do controlador GP3101:

Consulte o manual da linha GP3000 para um maior detalhamento da configuração do canal Ethernet em cada um dos modelos.

Os seguintes canais do Display #3 são utilizados para controle da comunicação ModbusTCP e portanto não podem ser utilizados para outras finalidades:

Variáveis de configuração (somente p/ conexão ativa):

Nome	Canal	Endereço	Descrição
IP_1	0	C400h	1º octeto do endereço IP do host (Supervisório)
IP_2	1	C402h	2º octeto do endereço IP do host
IP_3	2	C404h	3º octeto do endereço IP do host
IP_4	3	C406h	4º octeto do endereço IP do host
PORTA	4	C408h	Porta p/ conexão da comunicação Modbus com o host

Variáveis de monitoração (conexão ativa e passiva):

Nome	Canal	Endereço	Descrição
CONTC	5	C40Ah	Contador de tentativas de abertura de conexão TCP
RXPCM	6	C40Ch	Contador de pacotes Modbus recebidos
RXPCP	7	C40Eh	Contador de pacotes recebidos descartados

Conexão ativa nas funções Gateway pelos canais RS232 e RS485

As variáveis de interface de Gateway RS232 ativo e Gateway RS485 ativo devem ser alocadas nos seguintes endereços do display #3:

Prot.	Nome	Canal	Descrição
GW232	IP1_1	10	1º octeto do IP do equipamento destino (cliente)
	IP2_1	11	2º octeto do IP do equipamento destino (cliente)
	IP3_1	12	3º octeto do IP do equipamento destino (cliente)
	IP4_1	13	4º octeto do IP do equipamento destino (cliente)
	PORT1	14	Porta para conexão no equipamento destino
	DPCO1	15	Flag de disparo da conexão ativa (1) ativa; (3) encerra
	CNTD1	16	Contador de pedidos de conexão
GW485	IP1_2	17	1º octeto do IP do equipamento destino (cliente)
	IP2_2	18	2º octeto do IP do equipamento destino (cliente)
	IP3_2	19	3º octeto do IP do equipamento destino (cliente)
	IP4_2	20	4º octeto do IP do equipamento destino (cliente)
	PORT2	21	Porta para conexão no equipamento destino
	DPCO2	22	Flag de disparo da conexão ativa (1) ativa; (3) encerra
	CNTD2	23	Contador de pedidos de conexão

O padrão Ethernet permite que dois ou até três desses serviços sejam configurados simultaneamente. Isto é feito através da colocação de mais de uma linha ***INTERFACE3*** no programa.

O compilador deve indicar erro nos casos de conflito de alocação de uma porta serial para duas finalidades - Se for incluído no mesmo programa as instruções:
INTERFACE1=... e INTERFACE3=(PROTOCOLO=GAT232...)
ou:
INTERFACE2=... e INTERFACE3=(PROTOCOLO=GAT485...)

O compilador deve indicar erro no caso de haver a mesma programação de PORTA para diferentes serviços.

Exemplos:

INTERFACE3=(PROTOCOLO=GAT485 VEL=9600)
Configura o canal TCP/IP para operar como gateway com a porta serial RS 485(canal 2). A porta serial irá funcionar a 9600 baud.

INTERFACE3=(PROTOCOLO=MODBUSTCP PORTA=30)
Configura o canal TCP/IP para operar no protocolo MODBUS TCP, na porta lógica 30, sem abrir a comunicação na rede.

INTERFACE3=(PROTOCOLO=MODBUSTCP PORTA=30)
****
****
FACA TCPAT=1
Configura o canal TCP/IP para operar no protocolo MODBUS TCP, na porta lógica 30, tomando a iniciativa de abrir a comunicação na rede. Devem ser também configurados os parâmetros específicos para a operação ativa, conforme descrito acima.

INTERFACE3=(PROTOCOLO=GAT232 VEL=9600)
INTERFACE3=(PROTOCOLO=GAT485 VEL=9600)
INTERFACE3=(PROTOCOLO=MODBUSTCP PORTA=30)
Configura o canal TCP/IP para executar os três serviços simultaneamente. Funciona como gateway para o canal serial RS232 através da porta 23, funciona como gateway para o canal serial RS485 através da porta 24 e também opera no protocolo MODBUS TCP, através da porta lógica 30.

INTERFACE3=(PROTOCOLO=GAT232 VEL=9600 PORTA=10)
INTERFACE3=(PROTOCOLO=GAT485 VEL=9600 PORTA=20)
INTERFACE3=(PROTOCOLO=MODBUSTCP PORTA=30)
Configura o canal TCP/IP para executar os três serviços simultaneamente. Funciona como gateway para o canal serial RS232 através da porta 10, funciona como gateway para o canal serial RS485 através da porta 20 e também opera no protocolo MODBUS TCP, através da porta lógica 30.

ü *NEST=xxx*

Reserva um espaço de memória para a tabela de estados. Este espaço deve ser igual ou superior ao número de estados mais o número de testes independentes de estado de todo o programa. O valor para **'xxx'** é aceito na faixa 1 a 2000. Se omitido, o sistema reservará um espaço de 150 estados.

ü *PAN=x*

Este comando habilita a rotina de tratamento da placa BCM2085/PAN - Painel de leds. O parâmetro **`x'** define o número de placas utilizadas no sistema. Pode ser 1 ou 2. Outro valor provocará erro na compilação. Se este comando não estiver presente, as placas não serão acessadas.
É importante observar que a inclusão do comando PAN influi na capacidade de instalação de módulos BCM2085/CTR no sistema.

ü *RTC*

Habilita a operação do relógio de tempo real. Este comando é usado nos controladores BCM1086A, GP3000 e no BCM2085 com a placa `MEM' instalada.

ü *SAIDA r1{,r2,...rn} = aaa{,bbb,...nnn}*

Este comando está disponível para o controlador BCM2085B, com CPU7, CPU8 ou CPU8A. Associa as saidas digitais 'r1' ... até 'rn' às variáveis 'aaa' ... até 'nnn'. Com isto, as saídas poderão ser tratadas como variáveis.
A ordem das saídas r1{,r2,...rn} deverá ser crescente.
A variável 'aaa' deverá ser variável livre ou matricial. Se for matricial, só colocar o nome da variável, sem parênteses. A variável de índice 0 controla a primeira saída listada e assim por diante.
Quando a variável 'aaa' for matricial, não poderão ser definidas outras variáveis. Caso contrário, podem ser incluídas mais variáveis ('bbb' ... até 'nnn'), as quais deverão ser variáveis livres, não podendo ser variáveis indexadas, fixas ou de entrada digital.
O acionamento das saídas não declaradas por esta instrução, continuará sendo feita pelas instruções LIGA, DESL e outras do mesmo gênero. No Sistema Operacional, o bloco de tratamento das variáveis associadas às saídas será executado no início de cada ciclo. Cada variável associada que tiver o seu valor diferente de zero será ligada, senão a saída será desligada.

No programa BCM, todas as saídas associadas deverão ser declaradas depois dos comandos DIM (se houver) e NEST (se houver) e antes da descrição da MALHA 0. O compilador deverá gerar uma mensagem de erro, se esta ordem não for respeitada.
No Compilador, as instruções LIGA n, DESL n, SE condição LIGA n e/ou SE condição DESL n, cuja saida 'n' houver sido associada a uma variável pelo comando SAIDA, gerará uma mensagem de erro.

ü *TCPn IP=x GT=y NT=z*

Configura os parâmetros TCP/IP dos canais Ethernet. O parâmetro "n" define o número do canal, que pode ser 1,2,3 ou 4.

Atenção: A opção do parâmetro "n" passa a ser obrigatória para o PROCP V5k 4.0 ou superior e não *é* válida para as versões anteriores.

Este comando é válido para os seguintes produtos:

- GP3101, GP3210 e GP3220: Canal 3
- CPU8A: Canais 2 e 4
- CPU9: Canais 1, 2 e 4

Os demais parâmetros configuráveis são os seguintes:

IP=x - Define o número IP do equipamento na rede. O PROCP testa por números válidos (0 a 255 e diferente de 0.0.0.0). Deve estar presente e colocado no formato padrão de quatro octetos separados por pontos.

GT=y - Define o endereço IP do gateway . Deve estar presente e colocado no formato padrão de quatro octetos separados por pontos.

NT=z - Define o netmask. Deve estar presente e colocado no formato padrão de quatro octetos separados por pontos.

Exemplo:
TCP3 IP=192.168.150.140 GT=192.168.150.255 NT=255.255.255.200

2.4.2 - Instruções de Programação

ü *COPIA xxx(yyy) PARA zzz(www) QUANT=kkk*

Copia um conjunto de elementos de uma matriz para outra, a partir dos índices especificados. Os parâmetros ‘yyy’ e ‘www’ podem ser uma constante ou uma variável. O parâmetro ‘kkk’ indica a quantidade de elementos a copiar, podendo ser constante ou variável não indexada.
Várias linhas ***COPIA*** podem ser usadas num estado, em sequencia. Esta instrução é executada na transição, isto é, somente na primeira vez em que entra no estado.

Recurso disponível para: GP3210, GP3220, CPU7-11, CPU8, CPU8A e CPU9.

ü *DECREMENTA aaa,bbb,ccc,ddd(iii)*

Decrementa de uma unidade o conteúdo das variáveis mencionadas. Qualquer tipo de variável é aceita, inclusive variáveis matriciais. Equivale à instrução "FACA aaa=aaa-1 E bbb=bbb-1", etc., com mais clareza na edição, economia de memória e de tempo de processamento.
Podem ser usadas várias instruções DECREMENTA em um estado. A única exigência é que as linhas com esta instrução devem ser adjacentes. Além disso, todas as instruções SET, RESET, INCREMENTA e DECREMENTA de um estado devem estar agrupadas.

Exemplo:

```
ESTADO 5:          ;Este bloco de programa aciona a saída 1 por 10 vezes,
FACA CONT=10       ;com um tempo ligado de 1s e desligado de 1s.
VA PARA 6

ESTADO 6:
LIGA 1
SE ATRASO=10 ENTAO 7

ESTADO 7:
DESL 1
DECREMENTA CONT
SE ATRASO=10 ENTAO 6
SE CONT=0 ENTAO 8
```

ü *EAI*

Essa instrução controla o funcionamento da placa BCM2085/EAI, permitindo que o usuário, através do programa em alto nível, defina as características de funcionamento da placa conforme a aplicação. A mesma instrução permite adquirir os valores analógicos dos pontos de entrada e ler os valores processados localmente pela placa EAI, incluindo-os no programa da forma mais adequada.

A variável **"r"** especifica o número do ponto analógico que será atendido pelo comando. O Sistema Operacional suporta um máximo de 128 pontos de entradas analógicas, distribuidos em até 8 placas.
O comando define a operação a ser feita dentro da instrução `EAI'. Cada comando possui um conjunto próprio de parâmetros com sintaxe definida. Somente um comando pode ser usado em cada linha de programa.

ü *EAI r FILTRO [KP=www] [KI=yyy] [KD=zzz] [AI=uuu] [LIN=h] [ESC=xxx] [DES=vvv] [TA=qqq] [LS=sss] [LI=ttt]*

Programa os valores gerais relativos ao sistema de controle do canal **`r'** da placa EAI:

- KP Transfere a variável **`www'**, relativa ao coeficiente proporcional vezes 256, para o PID do canal **`r'** da placa EAI. São permitidos valores na faixa 0 a 65534. Se omitido, o coeficiente proporcional assume o valor 1.

- KI Transfere a variável **`yyy'**, relativa ao coeficiente integral vezes 256, para o PID do canal **`r'** da placa EAI. São permitidos valores na faixa 0 a 65534. Se omitido, assume o valor 0.

- KD Transfere a variável **`zzz'**, relativa ao coeficiente derivativo vezes 256, para o PID do canal **`r'** da placa EAI. Se omitido, assume o valor 0.

- AI Transfere a variável **`uuu'**, relativa ao acumulador integral, para o PID do canal `r' da placa EAI. Se omitido, o acumulador integral não será alterado.

- LIN O número **`h'** (sempre constante) especifica o código para uma função de linearização. Essa função é aplicada diretamente sobre a leitura do conversor A/D. Todos os processamentos posteriores (saída VME, controle PID) terão como base o valor já passado pela função especificada.
Note que cada ponto analógico pode ter uma função de linearização distinta dos demais, independentemente da placa onde estiver instalada.
- Função 0: Sem linearização (Leitura medida = Lad)
- Função 1: Linearização para termopar tipo T - Leitura = 1000 + (10 * Temperatura)
- Função 2: Linearização para termopar tipo S - Leitura = 1000 + (10 * Temperatura)
- Função 3: Linearização para termopar tipo K - Leitura = 1000 + (10 * Temperatura)
- Função 4: Linearização para termopar tipo J - Leitura = 1000 + (10 * Temperatura)
Está previsto o desenvolvimento de funções de linearização para outros tipos de termopares e transdutores não-lineares.

- ESC O conteúdo da variável **`xxx'** será o fator de escala aplicado a saída do conversor A/D, após a passagem pela função de linearização. É expresso em termos de unidades do A/D por unidade da variável da saída, vezes 256. São permitidos valores na faixa 0 a 65534. Se omitido, assume o valor 256.

- DES O conteúdo da variável **`vvv'** será o deslocamento (offset) de leitura. Especifica o valor que será somado a saída da função de linearização, já incluído o fator de escala. São permitidos valores na faixa 0 a 65534. Se omitido, assume o valor 0.

- TA Transfere a variável **`qqq'**, relativa ao tempo de amostragem para os fatores integral e derivativo do filtro PID relativo ao canal **`r'** da placa EAI. É expresso em décimos de segundo; a variável pode assumir valores entre 1 e 255 correspondendo a intervalos de amostragem entre 0,1 e 25,5 segundos. Se omitido, o módulo assume um intervalo de amostragem de 1 segundo.

- LS Transfere a variável **`sss'** para o registro de limite superior do filtro PID. O conteúdo dessa variável irá ser o limitante superior da saída do filtro PID. Ao atingir esse valor, o acumulador integral não aumenta mais, independentemente do valor do erro. São permitidos valores na faixa 0 a 65534.

- LI Transfere a variável **`ttt'** para o registro de limite inferior do filtro PID. O conteúdo dessa variável irá ser o limitante inferior da saída do filtro PID. Ao atingir esse valor, o acumulador integral não diminui mais, independentemente do valor do erro.

ü *EAI r LEITURA [VME=yyy] [REF=xxx] [PID=www]*

Esse comando adquire informações relativas ao canal **`r'** da placa EAI para tratamento em alto nível. Os parâmetros permitem especificar as variáveis que receberão estas informações:

- VME Valor Medido. Carrega o valor medido da entrada analógica **`r'** na variável **`yyy'** no momento da execução da instrução. Note que o valor carregado corresponde a saída da função de linearização, aplicados ainda os fatores de escala e deslocamento. A fórmula aplicável é a seguinte:
VME = (ESC / 256) * Fx(Lad) + DES

Onde: **ESC** -Fator de escala, programado em "FILTRO"
DES -Fator de deslocamento, programado em "FILTRO"
Fx( )-Função de linearização, selecionada em "FILTRO"
Lad -Leitura obtida no conversor A/D.

- REF Valor de Referência PID. O conteúdo da variável **`xxx'** será o valor de referência para aplicação do filtro PID. Ver o parâmetro `PID' para o detalhamento do algoritmo.

- PID Valor de Saída PID. Carrega o valor de saída do filtro PID na variável **`www'**, de acordo com o seguinte algoritmo:
PID=Kp * Err + Ki * SErr + Kd * dErr

Onde: Err é o erro de controle (REF - VME); SErr é a soma de todos os erros acumulados a cada intervalo de amostragem; dErr é a diferença entre o erro atual e o erro registrado no intervalo de amostragem anterior.

Os comandos associados a instrução `EAI' são ativados no estado, isto é, são repetidos uma vez a cada varredura enquanto o controlador estiver no estado.
Além dos comandos descritos para trabalho com o módulo EAI, o Sistema Operacional possui a variável de atribuição fixa `EANER' que indica o número de módulos ativos no sistema e o código de erro relativo a inicialização ou operação das placas.

Observar que não podem ser utilizados módulos EAN e EAI no mesmo controlador.

ü *EAN*

Essa instrução controla o funcionamento da placa BCM2085/EAN, permitindo que o usuário, através do programa em alto nível, defina as características de funcionamento da placa conforme a aplicação. A mesma instrução permite adquirir os valores analógicos dos pontos de entrada e ler os valores processados localmente pela placa EAN, incluindo-os no programa da forma mais adequada.

A variável **"r"** especifica o número do ponto analógico que será atendido pelo comando. O Sistema Operacional suporta um máximo de 64 pontos de entradas analógicas, distribuidos em até 8 placas.

O comando define a operação a ser feita dentro da instrução `EAN'. Cada comando possui um conjunto próprio de parâmetros com sintaxe definida. Somente um comando pode ser usado em cada linha de programa.

ü *EAN r FILTRO [KP=www] [KI=yyy] [KD=zzz] [AI=uuu] [LIN=h] [ESC=xxx] [DES=vvv] [TA=qqq] [LS=sss] [LI=ttt]*

Programa os valores gerais relativos ao sistema de controle do canal **`r'** da placa EAN:

- KP Transfere a variável **`www'**, relativa ao coeficiente proporcional vezes 256, para o PID do canal **`r'** da placa EAN. Se omitido, o coeficiente proporcional assume o valor 1.

- KI Transfere a variável **`yyy'**, relativa ao coeficiente integral vezes 256, para o PID do canal **`r'** da placa EAN. Se omitido, assume o valor 0.

- KD Transfere a variável **`zzz'**, relativa ao coeficiente derivativo vezes 256, para o PID do canal **`r'** da placa EAN. Se omitido, assume o valor 0.

- AI Transfere a variável **`uuu'**, relativa ao acumulador integral, para o PID do canal `r' da placa EAN. Se omitido, o acumulador integral não será alterado.

- LIN O número **`h'** (sempre constante) especifica o código para uma função de linearização. Essa função é aplicada diretamente sobre a leitura do conversor A/D. Todos os processamentos posteriores (saída VME, controle PID) terão como base o valor já passado pela função especificada.
Note que cada ponto analógico pode ter uma função de linearização distinta dos demais, independentemente da placa onde estiver instalada.
- Função 0: Sem linearização (Leitura medida = Lad)
- Função 1: Linearização para termopar tipo T - Leitura = 1000 + (10 * Temperatura)

Está previsto o desenvolvimento de funções de linearização para outros tipos de termopares e transdutores não-lineares.

- ESC O conteúdo da variável **`xxx'** será o fator de escala aplicado a saída do conversor A/D, após a passagem pela função de linearização. É expresso em termos de unidades do A/D por unidade da variável da saída, vezes 256. Se omitido, assume o valor 256.

- DES O conteúdo da variável **`vvv'** será o deslocamento (offset) de leitura. Especifica o valor que será somado a saída da função de linearização, já incluído o fator de escala. Se omitido, assume o valor 0.

- TA Transfere a variável **`qqq'**, relativa ao tempo de amostragem para os fatores integral e derivativo do filtro PID relativo ao canal **`r'** da placa EAN. É expresso em décimos de segundos; a variável pode assumir valores entre 1 e 255 correspondendo a intervalos de amostragem entre 0,1 e 25,5 segundos. Se omitido, o módulo assume um intervalo de amostragem de 1 segundo.

- LS Transfere a variável **`sss'** para o registro de limite superior do filtro PID. O conteúdo dessa variável irá ser o limitante superior da saída do filtro PID. Ao atingir esse valor, o acumulador integral não aumenta mais, independentemente do valor do erro.

- LI Transfere a variável **`ttt'** para o registro de limite inferior do filtro PID. O conteúdo dessa variável irá ser o limitante inferior da saída do filtro PID. Ao atingir esse valor, o acumulador integral não diminui mais, independentemente do valor do erro.

ü ***EAN r LEITURA [VME=yyy] [REF=xxx] [PID=www] [OVE=zzz]***

Esse comando adquire informações relativas ao canal **`r'** da placa EAN para tratamento em alto nível. Os parâmetros permitem especificar as variáveis que receberão estas informações:

- VME Valor Medido. Carrega o valor medido da entrada analógica **`r'** na variável **`yyy'** no momento da execução da instrução. Note que o valor carregado corresponde a saída da função de linearização, aplicados ainda os fatores de escala e deslocamento. A fórmula aplicável é a seguinte:
VME= (ESC / 256) * x(Lad) + DES
Onde:
ESC -Fator de escala, programado em "FILTRO"
DES -Fator de deslocamento, programado em "FILTRO"
Fx( )-Função de linearização, selecionada em "FILTRO"
Lad -Leitura obtida no conversor A/D.

- REF Valor de Referência PID. O conteúdo da variável **`xxx'** será o valor de referência para aplicação do filtro PID. Ver o parâmetro `PID' para o detalhamento do algoritmo**.**

- PID Valor de Saída PID. Carrega o valor de saída do filtro PID na variável **`www'**, de acordo com o seguinte algoritmo:
PID = Kp * Err + Ki * SErr + Kd * dErr

Onde: Err é o erro de controle (REF - VME) SErr é a soma de todos os erros acumulados a cada intervalo de amostragem, dErr é a diferença entre o erro atual e o erro registrado no intervalo de amostragem anterior.

- OVE Sinalizador de overflow na leitura. Ao ser executada a instrução, o sistema carrega o valor 1 na variável **zzz,** se o canal '**r'** estiver em overflow. Se o sinal de entrada estiver dentro da faixa, a instrução carrega o valor zero.

Os comandos associados a instrução `EAN' são ativados no estado, isto é, são repetidos uma vez a cada varredura enquanto o controlador estiver no estado.
Além dos comandos descritos para trabalho com o módulo EAN, o Sistema Operacional possui a variável de atribuição fixa `EANER' que indica o número de módulos ativos no sistema e o código de erro relativo a inicialização ou operação das placas.

ü ***ESCISInn[*] aaa NO EQ=xxx EM [#]bbb [BLOCO=ccc] [STA=ddd]***

Esta instrução faz a escrita de uma variável ou de um bloco de memória no equipamento número 'xxx' conectado à rede de comunicação ligada à porta 'nn' de uma placa BCM2085B/ISI instalada. Os dados enviados da CPU começam na posição 'aaa'. Só pode ser executada uma instrução ESCISI por estado.
O parâmetro 'nn', sempre constante, especifica o número da porta serial da placa ISI (podendo variar de 1 a 36). Os parâmetros 'aaa, 'xxx, 'bbb' e 'ccc' podem ser constantes ou variáveis, não podendo ser variáveis indexadas. O parâmetro 'ddd' será sempre variável. O parâmetro 'aaaendereço dos dados na CPU. Se 'aaa' for parâmetro 'xxx' especifica o número do equipamento destino para escrita na rede de comunicação. O parâmetro 'bbb' especifica o endereço no equipamento destino. Se 'bbb' for variável, o endereço inicial no equipamento destino será o conteúdo da variável 'bbb'. Os parâmetros 'ccc' e 'ddd' (se existirem), especificam respectivamente o tamanho do bloco (nº de bytes) a ser escrito e a variável onde será armazenado o status da comunicação. A opção "#" (antes de 'bbb') é usada para indicar que a escrita
no equipamento destino será direto na memória da ISI, caso contrário será na memória da CPU destino. A omissão do parâmetro 'ccc' (tamanho do bloco), indicará que a escrita será de apenas uma variável.

A opção '*' indica que a instrução será executada no estado, senão a mesma será executada somente na transição.

Observações:

1) Se 'aaa' ou 'bbb' forem constante, seu valor deverá ser hexadecimal e precedido do caracter $. Por exemplo, o valor BFFE deverá ser escrito: $BFFE.

2) O valor de 'xxx' e 'ccc' , se constante, deverá ser decimal. O valor de 'xxx' poderá variar de 1 a 127 e o valor de 'ccc' de 1 a 240.

3) A opção 'BLOCO' é obrigatória se o endereço no equipamento destino for direto na memória da placa ISI (opção #).

Instruções relacionadas:

ESCREVE
ESCMISI
LE
LEISI
LEMISI
ISI

ü *ESCMISIn[*] aaa EM bbb [BLOCO=ccc]*

Esta instrução faz a escrita na memória da placa ISI (variável ou bloco), sem provocar uma transmissão pelo canal de comunicação da ISI. Só poderá ser executada uma instrução ESCMISI por estado.

O parâmetro 'n', sempre constante (1 a 12), especifica o número do módulo ISI onde será feito a escrita. Os outros parâmetros podem ser constantes ou variáveis, não podendo ser variáveis indexadas, entretanto podem ser usadas variáveis do tipo matricial, sem o índice. O parâmetro 'aaa' especifica o endereço dos dados na CPU. Se 'aaa' for variável, o endereço inicial na CPU será o endereço da própria variável 'aaa'. O parâmetro 'bbb' especifica o endereço na memória da ISI. Se 'bbb' for variável, o endereço inicial da ISI será o conteúdo da variável 'bbb'. O parâmetro 'ccc' (se existir) especifica o tamanho do bloco (nº de bytes) a ser escrito, senão a escrita será de uma variável.

A opção '*' indica que a instrução será executada no estado, senão a mesma será executada somente na transição.

Observação: Se 'aaa' ou 'bbb' forem constante, seu valor deverá ser hexadecimal e precedido do caracter $. Por exemplo, o valor BFFE deverá ser escrito: $BFFE. O valor de 'bbb' só poderá variar de 8000H a BFFFH. O valor de 'ccc', se constante, deverá ser decimal

ü *ESCREVE[*] aaa NO EQ=xxx EM yyyy [BLOCO=zzz] [STA=bbb]*

O Controlador Programável irá escrever um determinado bloco de informação em outro equipamento da rede de comunicação, através do canal serial. Para maiores informações a respeito do Protocolo Geral de Comunicação, ver o manual do software de desenvolvimento.
A opção **"*"** tem função similar àquela adotada na instrução "FACA". Sem o asterisco, a escrita é executada somente uma vez a cada entrada no estado correspondente. Com o asterisco, a operação fica sendo repetida enquanto o CP estiver no estado.

O parâmetro **aaa** especifica o nome da variável cujo endereço será o ponto de partida para a escrita. Pode ser especificada uma variável isolada ou matricial com índice constante.

A constante **xxx** especifica o número (código) do equipamento sobre o qual será feita a escrita.

A constante hexadecimal **yyyy** especifica o endereço de carga do bloco no equipamento de destino. Neste caso a constante não deverá ser precedida do sinal “$”.
A constante opcional **zzz** especifica o tamanho do bloco (em bytes) de memória a ser transferido. Pode estar na faixa 1 a 240. Se omitido, a instrução assume que o bloco possui dois bytes (uma variável).

O parâmetro **bbb** especifica a variável que irá receber os códigos referentes às informações de status do processo de comunicação. Os códigos de status definidos são:

0 - Tarefa completada com sucesso
1 - Tarefa agendada no buffer para execução pelo SO
2 - Tarefa em curso de execução pelo SO
3 - Erro por falta de resposta no equipamento indicado
4 - Erro por sobrecarga do buffer de tarefas.

Só é permitido usar uma instrução ESCREVE por estado. No manual de Protocolos e Redes de comunicação poderão ser encontradas mais informações sobre o significado dos códigos e a seqüência de tarefas de comunicação.

Exemplos:
ESTADO 2:
ESCREVE HAA NO EQ=15 EM 8965*

O controlador irá escrever a variável HAA no equipamento 15, na posição de memória 8965/66H. A operação será repetitiva - A escrita ficará sendo feita enquanto o programa estiver no estado 2.

ESTADO 2:
ESCREVE XAM NO EQ=15 EM 8400 BLOCO=100 STA=AAB

O controlador irá copiar a área de memória iniciada pela variável XAM no equipamento 15, a partir da posição de memória 8400H. A área tem um tamanho de 100 bytes e o status da operação poderá ser visto através do código presente na variável AAB. A escrita será feita só uma vez, quando o sistema entrar no estado 2.

Instruções relacionadas :
ESCISI
ESCMISI
LE
LEISI
LEMISI

- ***ESTADO r :***

Marca para o compilador o início das instruções relativas ao estado **`r'**. Os estados podem ser numerados de zero até 254.
A numeração dos estados no momento da edição do programa pode ser feita em qualquer ordem, pulando números ou misturando a seqüência de numeração. Não há qualquer sentido de prioridade ou influência no funcionamento, que dependa dos números escolhidos para os estados ou da ordem de edição. A única consideração importante é que o controlador ao ser ligado, inicia sempre pelo estado zero.

- ***EVENTOS r FILTRO BUF=aaa TAM=bbb [DEB=ccc] [MASC=xxyy] [TRAN=d] [EST=e] [MESTRE] [DUPLOS=f,g,h,...] [TDUP=iii]***

- ***EVENTOS r FILTRO BUF=aaa TAM=bbb [DEB=ccc] [MASC=xxyy] [TRAN=d] [EST=e] [MESTRE] [DUPLOS=fff] [TDUP=iii]***

Esta instrução programa os parâmetros da placa EDE número “r”, conforme descrito abaixo. Deve ser executada no início do programa e antes de qualquer outra instrução ***EVENTOS***:

BUF

O parâmetro 'aaa' especifica o nome da variável cujo endereço será o endereço inicial do buffer onde serão armazenados os eventos lidos da placa EDE, através do comando EVENTOS r LEITURA. A variável 'aaa' deverá ser do tipo matricial **(sem índice)**.

Toda vez que esta instrução é executada, a posição 2 da matriz 'aaa' (que indica o número de eventos lidos) é zerada.

TAM

A constante 'bbb' especifica o número máximo de eventos (tamanho) que poderão ser carregados no buffer citado acima. A matriz 'aaa' deverá ser declarada no inicio do programa com a instrução DIM e seu tamanho. deverá ser maior ou igual a 3 * 'bbb' + 2. O valor de 'bbb' pode variar de 1 a 255.

CUIDADO: A declaração errada desta constante ou da matriz poderá acarretar uma escrita de dados em área de memória imprópria, sem indicação de erro pelo compilador.

DEB

A constante decimal 'ccc' especifica o tempo de debounce em 'ms' para todas as entradas da respectiva placa. O valor pode variar de 1 a 254. Por omissão, o valor assumido será 10 ms.

MASC

As constantes hexadecimais 'xx' e 'yy' especificam a máscara para o registro de eventos das 32 entradas digitais. O valor hexadecimal 'xx' corresponde respectivamente às 16 últimas entradas da placa (32 a 17) e o valor 'yy' às 16 primeiras (16 a 1). Para uma entrada gerar evento (entrada não mascarada), o bit correspondente deverá ser setado pra '1'. O valor por omissão será '1' para todos os bits.
Obs.: O compilador PROCP sempre irá programar a máscara, independente dela ser omitida.

TRAN

O número 'd' (sempre constante) indica se a ocorrência de transiente (bounce) sem a mudança de estado gera um evento. Se o valor for '0', o transiente não gera evento. Se for '1', gera evento. Por omissão, o valor assumido será '0'.

EST

O número 'e' (sempre constante) especifica o que ocorrerá se houver estouro da capacidade na carga dos eventos. Se o valor for '0', serão descartados os primeiros eventos que foram armazenados na EDE. Se o valor for '1', serão descartados os últimos eventos. Por omissão, o valor assumido será 1. (A 1ª versão de S.O. para a placa EDE suportará apenas este caso).

MESTRE

Se este parâmetro for atribuído, a placa EDE que está sendo programada será considerada a placa mestre, isto é, a que gera o clock de sincronismo para as outras EDEs que estiverem presentes no mesmo bastidor. Só pode haver uma placa mestre por CPU. Se uma placa EDE já foi programada como mestre, e é dado um novo comando programando outra placa EDE como mestre, a última programação é a que será assumida.

Quando este parâmetro for atribuído, poderá haver um atraso de até 0,5 ms no relógio da placa EDE usado para o registro dos eventos.

DUPLOS

As constantes decimais ‘f’, ‘g’, ‘h’, ... especificam quais serão as entradas duplas da respectiva placa EDE. Somente podem ser configuradas entradas ímpares (1 a 31) e a segunda entrada de cada ponto será a entrada par imediatamente a seguir.

Se for usado o parâmetro ‘fff’, este especifica o nome da variável , que inclusive pode ser indexada. Cada bit da variável configura 2 entradas da placa, sendo que bit em zero indica entrada simples e bit em 1 indica entrada dupla. O bit 0 configura as entradas 1 e 2, o bit 1 configura as entradas 3 e 4 e assim sucessivamente.

Por omissão, todas as entradas serão configuradas como simples. O uso desse parâmetro só é válido para o BCM2085B/EDE-11.

Este recurso só está disponível no Controlador Programável que use o módulo BCM2085B/CPU7-11 ou /CPU8.

TDUP

O parâmetro ‘iii’ pode ser uma constante decimal ou uma variável (inclusive indexada) e especifica o tempo que deve ser aguardado para registro do evento duplo de posição indeterminada (em ms). Por omissão, o valor assumido será 10.000 ms (10s).

Observação: O tempo real p/ registro do evento de transição será igual a: TDUP + DEB

Os parâmetros DUPLOS e TDUP, quando presentes, devem ser configurados simultaneamente com constantes ou simultaneamente como variáveis.

Por omissão, todas as entradas serão configuradas como simples. O uso desse parâmetro só é válido para o BCM2085B/EDE-11.

Este recurso só está disponível no Controlador Programável que use o módulo BCM2085B/CPU7-11.

ü *EVENTOS RELOGIO*

Esta instrução reprograma o relógio de todas as placas EDE presentes no sistema. Esta instrução sempre deverá ser executada no início do programa do usuário, para que o relógio de todas as placas EDE seja atualizado. A passagem do programa do usuário pelo estado com essa instrução deve ser de apenas um ciclo (usar VA PARA para sair do estado). A execução dessa instrução gera dois eventos, um com o horário anterior à atualização e outro com o horário atual.

Observações:

1) Quando for usada a instrução ***EVENTOS RELOGIO***, não poderá estar habilitada a sincronização do relógio via GPS (controlada pela variável PRGPS) e vice-versa. Lembrar que a sincronização via GPS já faz a reprogramação dos relógios das EDEs. Para mais detalhes, consulte o item “Sincronização com receptores GPS” nos capítulos CPU8 e CPU9 do manual de instalação e características BCM2085B.

2) Ao executar ***EVENTOS RELOGIO***, o conteúdo das variáveis 'RHOR', 'RMIN' e 'RSEC' do Sistema Operacional (que indicam a hora, minuto e segundo) será carregado em todas as placas EDE presentes no sistema. O controle de sincronismo será feito automaticamente pela placa EDE mestre. A diferença máxima de tempo entre o relógio de tempo real e o relógio das placas EDE (após programados) será igual ao tempo de ciclo de programa +- 10 ms.

3) Se o conteúdo das variáveis RHOR, RMIN e RSEC for um valor inválido para hora, será carregado o horário 00:00:00 nas EDEs.

4) Após esta instrução ser executada, haverá um intervalo de até 1 segundo em que nenhum evento será registrado (se ocorrer). Isto ocorre em função do ajuste do relógio das placas EDE. **Atenção:** Durante este tempo não poderá ser executado nenhuma instrução EVENTOS.

ü *EVENTOS r LEITURA [NUM=aaa]*

Esta instrução adquire as informações dos eventos da placa 'r'.

NUM

O conteúdo da variável 'aaa' será o número de eventos que o usuário deseja ler na placa EDE e carregar no buffer programado pela instrução EVENTOS r FILTRO. Esta variável não poderá ser matricial nem de entrada digital. O valor poderá estar entre 0 e 255. Se a EDE tiver menos eventos do que o valor programado em 'aaa', apenas serão lidos os existentes. Além disso, eles serão carregados no buffer a partir da posição do último evento carregado, até que todo o buffer seja preenchido. Quando o buffer estiver preenchido e for dada uma nova instrução de leitura, a mesma não terá efeito.

Se este parâmetro for omitido, o número de eventos que será lido será o número de eventos atual que existe na placa EDE ou o número de eventos que faltam para preencher o buffer programado em FILTRO. Vale o que for menor.Abaixo está descrito a estrutura do buffer criado pela instrução EVENTOS r FILTRO e que é usado pela instrução EVENTOS r LEITURA:

BUF (0)	livre
BUF (1)	livre
BUF (2)	nº eventos
BUF (3),(4),(5)	evento 1
BUF (6),(7),(8)	evento 2
BUF (9),(10),(11)	evento 3
	evento N (onde N=nº de eventos programado em TAM.

Observação: A posição BUF(2) da matriz contém o número de eventos carregados no buffer. Este valor será atualizado pelo sistema operacional da CPU, todas as vezes que for executada uma instrução LEITURA, até atingir o valor máximo (TAM) programado em FILTRO. O usuário poderá tratar os eventos e zerar o conteúdo de BUF(2), todas as vezes que o valor máximo for atingido, para que novos eventos possam ser carregados no buffer. O conteúdo de BUF(2) também é zerado quando é executado a instrução EVENTOS r FILTRO. A estrutura de cada evento transferido pela EDE segue um formato que será descrito a seguir. Existem dois padrões de formato, um para o produto EDE e EDE-10 e outro padrão para o produto EDE-11.

Estrutura para os produtos EDE (311022200-8) e EDE-10 (31022203-3):

Cada evento será constituído dos seguintes parâmetros:

1 - hora
2 - minuto
3 - segundo
4 - milisegundo
5 - nº do ponto (1 a 384)
6 - tipo de evento: - entrada foi p/ '0'
- entrada foi p/ '1'
- ocorreu bounce (transiente) **
- acerto do relógio: hora antiga
- acerto do relógio: hora nova
- overrum na leitura

A estrutura de cada evento será

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1º palavra	0	0	0	H	H	H	H	H	0	0	M	M	M	M	M	M

onde : H = hora
M = minuto

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2º palavra	S	S	S	S	S	S	L	L	L	L	L	L	L	L	L	L

onde : S = segundo
L = milisegundo

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3º palavra	0	T	T	T	0	0	0	P	P	P	P	P	P	P	P	P

onde: T=tipo Tipo: 000 -> “0"
P=nº do ponto 001 -> “1"
010 -> bounce +
011 -> bounce -
100 -> hora antiga
101 -> hora nova
110 -> overrun na leitura das entradas

** O evento “bounce” indica que ocorreu um transiente numa entrada, sem ter havido mudança de estado.

Estrutura para o produto EDE-11 (31022205-7):

Cada evento é constituído dos seguintes parâmetros:
1 - contador de dias (0 a 7)
2 - hora (0 a 23)
3 - minuto (0 a 59)
4 - segundo (0 a 59)
5 - milissegundo (0 a 999)
6 - nº do ponto (1 a 384)
7 - tipo de evento (0 a 20)

A estrutura de cada evento está descrita abaixo:

1ª palavra	C	C	C	H	H	H	H	H	0	0	M	M	M	M	M	M

onde: C = contador de dias
H = hora
M = minuto

2ª palavra

S	S	S	S	S	S	L	L	L	L	L	L	L	L	L	L

onde: S = segundo
L = milissegundo

3ª palavra

T	T	T	T	T	0	0	P	P	P	P	P	P	P	P	P

onde: T = tipo
P = nº do ponto

Tipos:

```
00000  (0)  -> OFF                -|
00001  (1)  -> ON                 | p/ ponto
00100  (4)  -> bounce -           | simples
00101  (5)  -> bounce +          -|

01000  (8)  -> indeterminado - -|
01001  (9)  -> OFF                | p/ ponto
01010 (10) -> ON                  | duplo
01011 (11) -> indeterminado +-|

01100 (12) -> não usado
01101 (13) -> não usado
01110 (14) -> não usado
01111 (15) -> não usado

10000 (16) -> hora antiga{*}
10001 (17) -> hora nova{*}
10010 (18) -> overrun na leitura das entradas
10011 (19) -> overflow no buffer de eventos
10100 (20) -> hora nova c/erro{*}
```

Observação: O evento 'bounce' indica que ocorreu um transiente numa entrada, sem ter havido mudança de estado.

Os eventos marcados com {*} são aqueles produzidos pela instrução *EVENTOS RELÓGIO.*

ü *EVENTOS r ENTRADAS BUF=bbb*

Esta instrução realiza a leitura das 32 entradas digitais da placa 'r' e carrega o valor da leitura na matriz 'bbb'. Esta operação é tratada por estado ou seja, durante toda a permanência no estado, o buffer será continuamente atualizado pela leitura das entradas.

BUF

O parâmetro 'bbb' especifica o nome da variável cujo endereço será o endereço inicial do buffer onde será armazenado o resultado da leitura das 32 entradas. A variável 'bbb' deverá ser do tipo matricial e seu tamanho deverá ser de no mínimo 32 posições (32 variáveis). Na primeira posição será armazenado o estado da entrada 1, na segunda posição o estado da entrada 2 e assim sucessivamente. Valor 0 = entrada desligada. Valor 1 = entrada ligada.
A matriz 'bbb' deverá ser declarada no início do programa através da instrução DIM.

ü *EVENTOS r TESTE BUF=bbb*

Esta instrução realiza o teste das 32 entradas digitais da placa 'r' e carrega o valor do teste na matriz 'bbb'.

BUF

O parâmetro 'bbb' especifica o nome da variável cujo endereço será o endereço inicial do buffer onde será armazenado o resultado do teste das entradas. A variável 'bbb' deverá ser do tipo matricial e seu tamanho deverá ser de no mínimo 2 variáveis (4 bytes). O primeiro byte armazenado será o byte do estado das primeiras 8 entradas digitais da placa e assim sucessivamente. O bit em '0' significa que a respectiva entrada está boa. O bit em '1', indica que a entrada está com defeito.
A matriz 'bbb' deverá ser declarada no início do programa através da instrução DIM.

Quando é executado a instrução EVENTOS r TESTE, a placa EDE fica cega durante aproximadamente 70 ms, ou seja, durante este tempo ela não registra a ocorrência de nenhum evento (se ocorrer). Esta instrução faz o autoteste das 32 entradas digitais da respectiva placa. Se os valores lidos pela CPU são iguais a zero, é porque todas as entradas estão OK, caso contrário existe erro em alguma delas, conforme definido nas instruções da linguagem BCM.

ü *FACA[*]operação1<E>operação2<E>.....operaçãoN*

Esta instrução executa operações aritméticas entre variáveis e/ou constantes. Todas as variáveis e constantes são de 16 bits (faixa de 0 a 65535). Podem ser usadas variáveis de qualquer tipo, exceto aquelas relativas às entradas digitais. Além disso, algumas variáveis de atribuição fixa reservadas só para leitura não podem ser usadas para receber resultados (Exemplo: CT1 a CT9).

Apesar de todos os operandos serem de 16 bits, as operações internas de multiplicação e divisão são realizadas em 32 bits para aumentar a precisão do resultado em cálculos encadeados.

O sinal opcional [*], quando presente, faz que a operação esteja associada ao estado e não a transição. Neste caso, a operação será repetida em todos os ciclos de varredura, atualizando a saída na variação dos operandos. Caso contrário, a operação é feita apenas quando a malha entra no estado (na transição).

As operações ao longo de uma linha são sempre resolvidas da esquerda para direita. Não há prioridade de execução para os operadores

Exemplo:

```
MALHA 1:
ESTADO 0:
FACA*  CONT=GAMA+DELT-453*8  E  VAR6=VAR5-VAR4+VAR3
;operação por estado.
SE INI=1 ENTAO 1

ESTADO 1:
FACA GAMA=GAMA+1        ;operação por transição.
SE INI=0 ENTAO 0
```

Podem ser usadas várias linhas “FACA” em um estado, porém só uma pode estar sem asterisco. Com várias linhas “FACA”, coloque asteriscos em todas as linhas se você quiser operações de estado. Se você quiser operações de transição deixe a última linha “FACA” sem asterisco.

Diversos operadores estão disponíveis, conforme descrito a seguir:

- Operações aritméticas básicas: Soma(+), Subtração(-), Multiplicação(*), Divisão(/).

- Operação lógica “E”: ***FACA XXX=YYY&ZZZ***

- Operação lógica “OU”: ***FACA XXX=YYY|ZZZ***

- Operação lógica “XOR”: ***FACA XXX=YYY^ZZZ***

- Operação lógica “NOT”: ***FACA XXX=~YYY***

- Operação lógica “Deslocamento à esquerda”: ***FACA XXX = YYY << ZZZ*** (O conteúdo de YYY é deslocado por ZZZ passos, e o resultado é lançado em XXX).

- Operação lógica “Deslocamento à direita”: ***FACA XXX=YYY>>ZZZ***
(O conteúdo de YYY é deslocado por ZZZ passos, e o resultado é lançado em XXX).

- Operação lógica “Isolar um bit”: ***FACA XXX=YYY.ZZZ***
(Isola o bit ZZZ da variável YYY e lança o resultado 0 ou 1 em XXX)

- Operação lógica “Resto da divisão”: ***FACA XXX=YYY%ZZZ***
(A variável XXX recebe o resto da divisão de YYY por ZZZ)

- Operação lógica “Raiz quadrada”: ***FACA XXX=#YYY***

- Separação do endereço da variável (parte baixa): ***FACA XXX=&MAT(0)***
(A variável XXX irá assumir a parte baixa (bits 0 a 15) do endereço em RAM da variável MAT(0))

- Separação do endereço da variável (parte alta): ***FACA XXX=&&MAT(0)***
(A variável XXX irá assumir a parte alta (bits 16 a 23) do endereço em RAM da variável MAT(0)).

Observação: Essa opção de endereço da parte alta só é válida para os produtos com área de variáveis maior que 64kB (CPU8A, CPU9, GP3210, GP3220)

Quando o valor de uma constante é atribuido a uma variável, essa constante pode ser expressa no formato decimal (normal) hexadecimal ou binário.
Para o formato hexadecimal, adicionar ”h” minúsculo após a constante.
Para o formato binário adicionar o “b” minúsculo após a constante.

Exemplos:

FACA XXX=30h ;Equivale a FACA XXX=48
FACA XXX=10101b ;Equivale a FACA XXX=21

Recurso disponível para: GP3101, GP3210, GP3220, CPU7-11, CPU8, CPU8A e CPU9, usando o PROCP-V5k 4.0 ou superior. Para as versões anteriores do PROCP, somente as quatro operações básicas são permitidas em FACA.

ü FACA[*] xxx(yyy)={a,b,c,d,...}

Carrega uma lista de constantes numa matriz, onde ‘xxx’ deve ser uma variável matricial e ‘yyy’ o índice inicial da matriz, podendo ser constante ou variável. Os parâmetros ‘a’, ‘b’, ‘c’, ‘d’ ... são constantes a serem carregadas na matriz a partir da posição indicada pelo índice. A opção ‘*’ indica que a instrução é executada no estado, caso contrário é executada da transição.

Atenção: Na sintaxe da instrução, o uso dos colchetes {} é obrigatório.

Este formato da instrução FACA é mais eficiente do que diversos FACAs encadeados. Executa com maior velocidade e economiza espaço de memória flash em relação a instruções FACA encadeadas. Ocupa 2 bytes por parâmetro carregado.

Recurso disponível para: GP3101, GP3210, GP3220, CPU7-11, CPU8, CPU8A e CPU9.

- ***IMPRIME [!][/][rr][(]aaa[)]{[!][/][rr]bbb}***

  Com parênteses, imprime o string **`aaa'**. Sem parênteses imprime o conteúdo da variável **`aaa'**. O valor **`r'** especifica a posição para impressão a partir do último caracter impresso. O sinal [!] instrui para troca de página e o sinal [/] instrui para troca de linha.
  A aplicação da instrução "IMPRIME" deve levar em conta os seguintes cuidados:
  Apenas uma linha "IMPRIME" pode ser colocada em cada estado. Não há, no entanto, restrições ao comprimento da linha ou número de termos a imprimir.
  O sistema operacional BCM trata o interface com impressora através de um buffer de 256 caracteres, transferidos a taxa de um caracter por ciclo do controlador. O programador deve cuidar para não exceder a capacidade do buffer. Se um trecho impresso exige mais de 256 caracteres, as instruções de impressão devem ser separadas por um tempo suficiente para que haja a descarga do buffer.
  Do ponto de vista do preenchimento do buffer, a contagem de caracteres é feita da seguinte forma: 5 caracteres por variável; 1 caracter por letra ou número entre parênteses; 2 caracteres por símbolo de troca linha, 1 caracter por símbolo de troca página; e **`r'** caracteres para posicionamento, conforme o próprio valor **`r'**.

  Observação: esta instrução pode ser usada apenas com CPU3, CPU4 e CPU4A com interface para impressora paralela pelo módulo MEM.

- ***INCLUI:<arquivo.bcx>***

  Usado para incluir um trecho de programa em Linguagem Descritiva, que será inserido na posição da instrução INCLUI. A instrução aceita tanto arquivos no formato BCM quanto arquivos no formato .BCX. O arquivo formado .BCX é desenvolvido normalmente usando o editor do PROCP, mas salvo usando uma nova opção do menu ARQUIVO: “SALVAR COMO BCX”. Nesta opção, o PROCP pede o nome do programa e uma senha, gravando o arquivo de forma escriptada, não editável no formato ASCII. O editor do PROCP abre arquivos . BCX, fazendo automaticamente a decriptação , desde que o usuário coloque a senha correta.

ü *INCREMENTA aaa,bbb,ccc,ddd(iii)*

Incrementa de uma unidade o conteúdo das variáveis mencionadas. Qualquer tipo de variável é aceita, inclusive variáveis matriciais. Equivale à instrução "FACA aaa=aaa+1 E bbb=bbb+1",etc., com mais clareza na edição, economia de memória e de tempo de processamento.
Podem ser usadas várias instruções INCREMENTA em um estado. A única exigência é que as linhas com esta instrução devem ser adjacentes. Além disso, todas as instruções SET, RESET, INCREMENTA e DECREMENTA de um estado devem estar agrupadas.

ü *ISInn VEL=CV1 EER=CV2 PRO=CV3 [EUR=CV4] [TEM=CV5] [TOC=CV6] [TOR=CV7] [TOF=CV8] [NSE=CV9]*

Configura os canais seriais do módulo BCM2085B/ISI. O estado com essa instrução não deve estar ativo quando em comunicação - só passar por esse estado para configurar e sair em seguida.

Onde:

- ***nn:*** constante que especifica a porta serial a ser programada (podendo variar de 1 a 36). Cada placa tem 3 portas seriais.
- ***CV1:*** constante ou variável que especifica a velocidade da porta serial, conforme o código:

1 - 1200bauds	5 - 96.000bauds
2 - 9600bauds	6 - 100.000bauds
3 - 19200bauds	7 - 250.000bauds
4 - 48.000bauds	

- ***CV2:*** constante ou variável que especifica o endereço da porta na rede.
- ***CV3:*** constante ou variável que especifica o protocolo a ser usado pela porta serial. O significado depende do modelo de ISI utilizado. Consulte o Manual de Instalação e Características Técnicas BCM2085B - capítulo ISI.
- ***CV4:*** constante ou variável que especifica o endereço do último equipamento ativo na rede.

- ***CV5:*** constante ou variável que especifica o tempo de espera antes do envio de cada mensagem (em ms/10). Se omitido o tempo é zero.
- ***CV6:*** constante ou variável que especifica o time-out entre caracteres para descartar uma mensagem (em ms/10). Se omitido o tempo é zero.
- ***CV7:*** constante ou variável que especifica o time-out para resposta de uma solicitação (em ms/10). Se omitido o tempo é 500ms.
- ***CV8:*** constante ou variável que especifica o time-out para criação da ficha (em ms/10). Se omitido, o tempo é igual a: 96+4*EER
- ***CV9:*** constante que força um número de série para o módulo ISI.

Observações:
- Todas as constantes só poderão assumir valores de 0 a 255, exceto CV9 que poderá assumir valores de 0 a 65535.
- Para velocidades acima de 48000 bauds pode haver restrições de recursos e números de canais. Consulte a BCM.
- Conforme o modelo de ISI utilizado, alguns parâmetros podem ser ou não aplicáveis. Consulte o Manual de Instalação e Características Técnicas BCM2085B - capítulo ISI.

ü *LEISInn[*] [#]aaa NO EQ=xxx EM bbb [BLOCO=ccc] [STA=ddd]*

Esta instrução faz a leitura de uma variável ou de um bloco de memória no equipamento número 'xxx' conectado à rede de comunicação ligada à porta 'nn' de uma placa ISI instalada. Os dados lidos são carregados na CPU a partir da posição dada por 'bbb'. Só pode ser executada uma instrução LEISI por estado.

O parâmetro 'nn', sempre constante, especifica o número da porta serial da placa ISI sempre constante (1 a 36). Os parâmetros 'aaa', 'xxx', 'bbb' e 'ccc' podem ser constantes ou variáveis, não podendo ser variáveis indexadas. O parâmetro 'ddd' será sempre variável. O parâmetro 'aaa' especifica o endereço no equipamento destino. Se 'aaa' for variável, o endereço inicial no equipamento destino será o conteúdo da variável 'aaa'. O parâmetro 'xxx' especifica o número do equipamento destino para leitura na rede de comunicação. O parâmetro 'bbb' especifica o endereço inicial para carga na CPU. Se 'bbb' for variável, o endereço inicial de carga na CPU será o endereço da própria variável 'bbb'.

Os parâmetros 'ccc' e 'ddd' (se existirem), especificam respectivamente o tamanho do bloco (nº de bytes) a ser lido e a variável onde será armazenado o status da comunicação. A opção "#" (antes de 'aaa') é usada para indicar que a leitura no equipamento destino será direto na memória da ISI, caso contrário será na memória da CPU destino. A omissão do parâmetro 'ccc' (tamanho do bloco), indicará que a leitura será de apenas uma variável.

A opção '*' indica que a instrução será executada no estado, senão a mesma será executada somente na transição.

Obervações:

1) Se 'aaa' ou 'bbb' forem constante, seu valor deverá ser hexadecimal e precedido do caracter $. Por exemplo, o valor BFFE deverá ser escrito: $BFFE.

2) O valor de 'xxx' e 'ccc' , se constante, deverá ser decimal. O valor de 'xxx' poderá variar de 1 a 127 e o valor de 'ccc' de 1 a 252.

3) A opção 'BLOCO' é obrigatória se o endereço no equipamento destino for direto na memória da placa ISI (opção #).

ü *LEMISIn[*] aaa EM bbb [BLOCO=ccc]*

Esta instrução faz a leitura de memória da placa ISI (variável ou bloco), sem provocar uma leitura no canal de comunicação da ISI. Só poderá ser executada uma instrução LEMISI por estado.
O parâmetro 'n', sempre constante (1 a 12), especifica o número do módulo ISI onde será feito a leitura. Os outros parâmetros podem ser constantes ou variáveis, não podendo ser variáveis indexadas, entretanto podem ser usadas variáveis do tipo matricial, sem o índice. O parâmetro 'aaa' especifica o endereço na memória ISI. Se 'aaa' for variável, o endereço inicial da ISI será o conteúdo da variável 'aaa'. O parâmetro 'bbb' especifica o endereço inicial de carga na CPU. Se 'bbb' for variável, o endereço inicial de carga na CPU será o endereço da própria variável 'bbb'. O parâmetro 'ccc' (se existir) especifica o tamanho do bloco (nº de bytes) a ser lido, senão a leitura será de uma variável.

A opção '*' indica que a instrução será executada no estado, senão a mesma será executada somente na transição.
OBS.: Se 'aaa' ou 'bbb' forem constante, seu valor deverá ser hexadecimal e precedido do caracter $. Por exemplo, o valor BFFE deverá ser escrito: $BFFE. O valor de 'aaa' só poderá variar de 8000H a BFFFH. O valor de 'ccc', se constante, deverá ser decimal.

- ***LE[*] yyyy NO EQ=xxx EM aaa [BLOCO=zzz] [STA=bbb]***

O Controlador Programável irá adquirir um determinado bloco de informação em outro equipamento da rede de comunicação, através do canal serial. Para maiores informações a respeito do Protocolo Geral de Comunicação, ver o manual de Protocolos e Redes de comunicação.

A opção **"*"** tem função similar àquela adotada na instrução "FACA". Sem o asterisco, a leitura é executada somente uma vez a cada entrada no estado correspondente. Com o asterisco, a operação fica sendo repetida enquanto o CP estiver no estado.

A constante hexadecimal **yyyy** especifica o endereço de aquisição do bloco no equipamento fonte. A constante **xxx** especifica o número (código) do equipamento onde será feita a aquisição.

O parâmetro **aaa** especifica o nome da variável cujo endereço será o ponto de partida para a carga dos valores adquiridos. A variável deve ser isolada ou matricial com índice constante - não pode ser usada uma variável matricial com índice variável.

A constante opcional **zzz** especifica o tamanho do bloco (em bytes) de memória a ser transferido. Pode estar na faixa 1 a 240. Se omitido, a instrução assume que o bloco possui dois bytes (uma variável).

O parâmetro **bbb** especifica a variável que irá receber os códigos referentes às informações de status do processo de comunicação. Os códigos de status definidos são:

0 - Tarefa completada com sucesso
1 - Tarefa agendada no buffer para execução pelo SO
2 - Tarefa em curso de execução pelo SO
3 - Erro por falta de resposta no equipamento indicado
4 - Erro por sobrecarga do buffer de tarefas
5 - Erro na resposta (parâmetros inválidos ou erro de CRC)

Só é permitido usar uma instrução LE por estado.
No manual de Protocolos e Redes de comunicação poderão ser encontradas mais informações sobre o significado dos códigos e a seqüência de tarefas de comunicação.

Exemplos:

ESTADO 2:
LE 8600 NO EQ=15 EM HAA*

O controlador irá ler as posições de memória 8600/01H no equipamento 15 e carregar o valor lido na variável HAA. A operação será repetitiva - A aquisição ficará sendo refeita enquanto o programa estiver no estado 2.

ESTADO 2:
LE 8400 NO EQ=15 EM XAM BLOCO=100 STA=AAB

O controlador irá copiar a área de memória iniciada na posição de memória 8400H do equipamento 15 na memória do CP a partir da variável XAM. A área tem um tamanho de 100 bytes e o status da operação poderá ser visto através do código presente na variável AAB. A aquisição será feita só uma vez, quando o sistema entrar no estado 2.

Instruções Relacionadas
ESCREVE
ESCISI
ESCMISI
LEISI
LEMISI

- ***LIGA r1{,rn} {<LIGA><DESL> s1{,sn}}***
- ***DESL r1{,rn} {<LIGA><DESL> s1{,sn}}***

Liga e/ou desliga as saídas especificadas em r1,rn,s1,sn quando a malha entra no estado corrente. Não há hierarquia nestas operações. Na ocorrência de instruções conflitantes, o resultado na saída é imprevisível.

Atenção: Uma forma alternativa de controlar as saídas digitais é através das variáveis EPSDD e NPSDD (ver item 2.6.3.6). Quando este recurso é usado para controlar as saídas digitais, as instruções LIGA e DESL não devem ser usadas.

Exemplo:

DISPLAY=1.99
INTERFACE1(9600)=1
MALHA 0:
ESTADO 4:
LIGA 1,2,5 DESL 3,4
DESL 7
LIGA 8

- ***MALHA r/ MAQUINA r:***

Marca para o compilador o início dos estados e instruções relativas a malha de estado **`r'**. As malhas são numeradas de 0 em diante, com um limite máximo que depende do modelo de CLP usado.

Na edição do programa, as definições das malhas devem ser feitas em ordem crescente, sem numerações vagas entre uma malha e outra. A primeira deve ser sempre a malha 0; a seguir vem a malha 1; depois a malha 2; e assim por diante até o final do programa. A Linguagem Descritiva BCM utiliza os dois termos "Malha" e "Máquina" com o mesmo significado. O compilador do PROCP aceita indistintamente as duas formas.

- ***MOSTRA [r](l,c)"string1",aaa,"string2"***

Esta instrução é utilizada para a apresentação de variáveis e mensagens nos displays alfanuméricos dos controladores BCM1086A, GP3000 e BCM2085B (todas as CPUs, exceto CPU9).

Apresenta uma combinação de mensagens e conteúdos de variáveis, posicionados no display na linha 'l' (1 ou 2) e iniciados na coluna 'c' (1 a 32). A variável "aaa" pode ser isolada ou matricial.

O atributo 'r' permite executar algumas tarefas adicionais relacionadas à apresentação:

T - Limpa a tela antes de apresentar
L - Limpa a linha antes de apresentar
Dn - Alinha o valor (conteúdo da variável) à direita, com 'n' dígitos.
Z - Inclui os zeros à esquerda dos valores, se houver 'Dn'.
Sn - Condiciona a apresentação ao valor da variável de atribuição fixa WWW, igual a 'n'. O parâmetro 'n' pode estar entre 0 e 99.

Podem ser utilizados vários atributos combinados, se necessário.

Atributos da Instrução 'MOSTRA':
O significado de cada um dos atributos citados acima está detalhado a seguir:

Atributo `T´ (Limpa a tela):
A inclusão deste atributo em uma instrução `MOSTRA' faz com que todas as mensagens e variáveis carregadas anteriormente na tela sejam apagadas.

Atributo `L´ (Limpa a linha):
A inclusão deste atributo em uma instrução `MOSTRA' faz com que as mensagens que aparecem na linha em questão e as variáveis que iniciam na linha em questão sejam apagadas do display. É importante notar que no caso de uma mensagem que ocupe um pedaço de cada linha, somente será apagada a parte contida na linha que estiver sendo apagada. No caso de variáveis, serão apagadas todas as variáveis cuja primeira posição se localize na linha em questão.

Atributo `Dn´ (Alinha à direita com `n' dígitos):
Este atributo modifica a forma de tratamento dos zeros à esquerda da variável a ser apresentada. Se este atributo for omitido, a variável será apresentada com apagamento dos zeros à esquerda e alinhamento pela esquerda. Este atributo fará a variável ser apresentada com apagamento dos zeros à esquerda mas com alinhamento pela direita, sendo reservados `n' dígitos para a variável.
Este atributo influi apenas na apresentação da variável contida na mesma instrução `MOSTRA', não influenciando na apresentação das demais variáveis da tela.

Atributo `Z´ (Apresenta os zeros à esquerda):
Se estiver presente o atributo `Dn', este atributo faz com que sejam apresentados os zeros à esquerda no valor da variável, até o limite do número de dígitos definido pelo parâmetro `n'.
Este atributo influi apenas na apresentação da variável contida na mesma instrução `MOSTRA', não influenciando na apresentação das demais variáveis da tela.

Atributo `Sn´ (Apresentação condicional):
Uma instrução `MOSTRA' que contenha este atributo só será executada se o valor da variável `WWW' for igual ao parâmetro `n'. As mensagens e variáveis que houverem sido carregadas no display por alguma instrução `MOSTRA' com atributo de apresentação condicional serão apagadas quando uma nova instrução `MOSTRA' com atributo de apresentação condicional for executada. Este atributo não está disponível para o BCM2085/CPU6.

Limitações e cuidados especiais da instrução 'MOSTRA':

- Em cada instrução `MOSTRA' poderão ser definidos até três elementos, mas apenas uma variável.

- O tamanho máximo das mensagens definidas nas instruções `MOSTRA' é 32 caracteres. As mensagens ou variáveis que iniciem na primeira linha e não caibam inteiramente nela continuarão na segunda linha.

- Os Sistemas Operacionais dos CLPs limitam a um máximo de 15, o número de variáveis apresentadas simultaneamente no display.

- Informações (variáveis ou mensagens) carregadas no display por uma instrução `MOSTRA' só desaparecerão do display se forem eliminadas por outra instrução 'MOSTRA'.

- Uma variável que tenha sido carregada no display terá seu valor corrente atualizado automaticamente pelo Sistema Operacional a partir do momento em que houver sido carregada e até o momento em que seja eliminada do display, mesmo que o programa saia do estado que contém a instrução `MOSTRA' que carregou a variável no display.

- Variáveis carregadas no display por uma instrução `MOSTRA' sem atributo de apresentação condicional só serão eliminadas pelos atributos `T' e `L'. **Carregar uma mensagem ou outra variável na posição ocupada por uma variável carregada anteriormente não elimina esta variável, e causa conflito no display**.

- A execução de uma instrução `MOSTRA' com atributo de apresentação condicional eliminará do display todas as informações (mensagens e variáveis) que houverem sido carregadas pela execução anterior de outra instrução `MOSTRA' com atributo de apresentação condicional, mesmo que esta instrução tenha sido executada em outro estado ou mesmo em outra malha do programa.

- Nos controladores GP3011, BCM2085B e BCM1086A, a execução simultânea de duas ou mais instruções `MOSTRA' com atributo de apresentação condicional faz com que apenas as informações referentes à última delas sejam apresentadas. No GP 3101 podem ser mostradas duas ou mais instruções com apresentação condicional.

- Informações que excedam a área do display não serão apresentadas, e o display não apresentará nenhuma indicação da ocorrência deste erro. Assim, se uma variável for programada para começar na posição 2,15 e o seu valor corrente tiver cinco dígitos, apenas os dois mais significativos serão apresentados, e a informação apresentada no display não será válida.

- Se houver um elemento mensagem após uma variável com alinhamento pela esquerda, o tamanho desta mensagem não poderá exceder 7 caracteres. Um elemento mensagem colocado após uma variável com alinhamento pela esquerda na mesma instrução `MOSTRA' terá sua posição de apresentação no display variável conforme o número de dígitos do valor corrente da variável correspondente.

- No caso de o espaço previsto para a variável ser menor do que o número de dígitos do seu valor corrente, ocorre a sobreposição do valor da variável sobre a mensagem. Neste caso, se o número de dígitos do valor corrente da variável diminuir, fazendo desaparecer a sobreposição, a mensagem original será recuperada.

- Não ocorre a apresentação de todos os dígitos do valor corrente de uma variável se estes não couberem inteiramente na área do display. É importante notar que neste caso o display não apresenta nenhuma indicação de erro, ao contrário do que ocorre se uma variável for apresentada com o atributo `Dn' e o número de dígitos do seu valor corrente for maior do que o parâmetro 'n' (ocorrência de overflow), caso em que o espaço reservado para a variável é preenchido com asteriscos.

Exemplos de aplicações da instrução 'MOSTRA':

```
ESTADO 1:
MOSTRA TL(1,4)"Peso:",W00,"g"
MOSTRA (2,2)"* Em Operacao *"
SE ABC=1 ENTAO 2
```

Peso: 392g
* Em operação *

* Variando o valor, a posição do "g" se modifica *

Peso:57g
* Em operação *

```
ESTADO 2:
MOSTRA TD4(1,2)"Total:",W00,","
MOSTRA D2Z(1,13)W01,"g"
SE ABC=2 ENTAO 3
```

Total:5315,17g

* O dígito menos significativo da variável `W00' será sempre apresentado imediatamente à esquerda da vírgula, e os zeros à esquerda da variável serão substituídos por espaços. A variável `W01' será apresentada imediatamente à direita da vírgula, com apenas dois dígitos, e sem apagamento dos zeros à esquerda.

```
ESTADO 3:
FACA WWW=3
MOSTRA S1(2,1)"Mensagem Um"

MOSTRA S2(2,1)"Mensagem Dois"
MOSTRA S3(2,1)"Mensagem Tres"
MOSTRA S4(2,1)"Mensagem Quatro"
MOSTRA S5(2,1)"Mensagem Cinco"
SE ABC=3 ENTAO 4
```

```
Total:5315,17g
Mensagem Três
```

```
ESTADO 4:
FACA WWW=1 E W02=0
MOSTRA S1(2,8)"WWW=",WWW
MOSTRA S2(2,8)"W02=",W02
MOSTRA S3(2,1)"Mensagem Tres"
```

```
Total:5315,17g
WWW
```

ü ***MOTOR***

Essa instrução permite a implementação de sistemas de posicionamento através do módulo BCM2085/MOT.
O módulo MOT pode ser aplicado em dois tipos básicos de sistema - Com motor DC ou com motor de passo. Nos sistemas acionados por motor DC, a informação de posição da unidade é realimentada para a MOT através de um encoder. Nos sistemas acionados por motor de passo, não há realimentação. O número de pulsos fornecido ao motor define com precisão o deslocamento.
As várias opções de comandos e parâmetros permitem um completo entrelaçamento entre cada unidade de posicionamento e o resto do sistema. A variável **`r'** especifica o número do eixo que será atendido pelo comando. O sistema operacional suporta até um máximo de 16 placas no controlador.

O comando define a operação a ser feita dentro da instrução `MOTOR'. Cada comando possui um conjunto próprio de parâmetros com sintaxe definida. Somente um comando pode ser usado em cada linha.

ü *MOTOR r DESL*

Esse comando interrompe qualquer movimento do motor **`r'** imediatamente. Não possui parâmetros. O movimento pode ser retomado pelo comando "MOTOR LIGA".

ü *MOTOR r LIGA [INI=iii FIM=jjj] [SVE]*

Esse comando liga o motor **`r'** imediatamente. É aplicado em duas situações:
- Dá início a um movimento instruído pelo comando "SEQUENCIA"
- Retoma um movimento interrompido pelo comando "DESL"

Sem os parâmetros INI e FIM, o motor **`r'** retoma a operação que estava sendo executada até o comando DESL Caso haja parâmetros e uma seqüência programada, os parâmetros instruem os passos inicial e final do movimento a ser executado.

Observe que os comandos "MOVE" e "HOME" não exigem o comando "LIGA", pois são disparados imediatamente. Os parâmetros são os seguintes:

- INI Índice inicial. A variável associada especifica o primeiro passo de movimento, que será iniciado imediatamente após o lançamento da instrução. Aceita valores na faixa 1 a 254.

- FIM Índice final. A variável associada especifica o último passo de movimento, após o qual a unidade pára. Aceita valores na faixa 1 a 254.

- SVE Especifica se a velocidade secundária permanece setada ou não no final de um movimento ou de uma seqüência. Sem a opção, a velocidade secundária desliga no final ou na interrupção do movimento. Com SVE a velocidade secundária permanece com o último valor programado.

ü ***MOTOR r MOVE [POP=www] [VEP=yyy] [NVE=zzz] [ACP=vvv] [SVE=yyy]***

Executa um movimento único, com características determinadas pelos parâmetros:

- POP Posição programada. A variável associada indica a posição programada para o final do movimento.

Em motores DC: POP = Pulsos do encoder / ESC
POP máximo aceitável: 1.000.000.000 / ESC + DES

Em motores de passo: POP = Número de passos
POP máximo aceitável: 65.535

A omissão desse parâmetro coloca o sistema no modo "velocidade", ou seja, o movimento só pode ser interrompido por outro comando, não existindo posição final especificada. Neste modo, só é permitido ordenar um movimento na direção positiva.

- VEP Velocidade programada. A variável associada indica a velocidade com que o eixo vai se aproximar da posição programada.

Em motores DC: VEP = Pulsos do encoder por segundo / ESC
VEP máximo aceitável: 19.000.000 / ESC

Em motores de passo: VEP = Passos por segundo
VEP máximo aceitável: 1.000

Obs.: Distorção máxima dos pulsos de saída: +/- 500 us (com comando: MOTOR r LEITURA POR=MOV=BLQ=)

Acima de 500 Hz não devem ser usados comandos de acesso à placa MOT enquanto ela está gerando os pulsos de saída, pois isto pode ocasionar a perda ou acréscimo de alguns pulsos.
A omissão desse parâmetro faz com que o eixo busque a posição com a máxima velocidade possível.

- NVE Nova velocidade. Ao atingir a posição programada, o eixo entra no modo velocidade, especificada pela variável indicada em NVE. A unidade de medida e o limite são os mesmos do parâmetro VEP. Se o valor for zero ou se o parâmetro for omitido, o movimento termina na posição programada. {nota 1}

- ACP Aceleração programada. A variável associada especifica a aceleração solicitada para que o eixo assuma a velocidade VEP.
ACP =Pulsos do encoder por segundo por segundo / ESC
ACP máximo aceitável: 65.535
A omissão desse parâmetro faz com que o eixo busque a velocidade programada com a maior aceleração possível. {nota 1}

- SVE Velocidade secundária. A variável associada especifica a tensão de saída no canal analógico secundário da placa MOT. Aceita valores na faixa 0 a 255, correspondendo de forma linear a tensões entre -10 e +9,92V (128 ==> 0V). Se omitido, a saída permanece em V. {nota 1}.

ü ***MOTOR r SEQUENCIA POP=www VEP=yyy [TEP=zzz] [ACP=vvv] [SVE=yyy] [IIN=aaa] [IFN=bbb]***

Executa uma seqüência de movimentos, com características determinadas pelas matrizes indicadas nos parâmetros: {nota 1}

- POP Matriz das posições programadas. Apresenta uma lista das posições que o eixo deve atingir em passos sucessivos. As considerações adotadas em relação as unidades de medida e aos limites são as mesmas do comando "MOVE". Esse parâmetro não pode ser omitido no comando SEQUENCIA.

- VEP Matriz das velocidades programadas. Indica as velocidades nas quais o eixo deverá atingir as posições programadas em POP, a cada passo. As considerações adotadas em relação as unidades de medida e aos limites são as mesmas do comando "MOVE". Esse parâmetro não pode ser omitido no comando SEQUENCIA.

- TEP Matriz dos tempos de espera. Ao invés de especificar uma posição final para um passo, o usuário pode desejar que o eixo se mova ou fique parado por um tempo determinado. Quando o valor de um passo na matriz indicada por TEP for diferente de zero, a posição na matriz POP é ignorada e o eixo se move na velocidade VEP pelo tempo especificado (em 1/100 s). Se omitido, não serão habilitados tempos de espera no movimento. Aceita valores na faixa 0 a 65535.

- ACP Matriz das acelerações programadas. Indica a aceleração para mudança de velocidade a cada mudança de direção ou partida do motor. Esse valor não será considerado nas trocas de velocidade sem mudança de direção.
As considerações adotadas em relação as unidades de medida e aos limites são as mesmas do comando "MOVE". A omissão desse parâmetro faz com que o eixo busque cada nova velocidade programada com a maior aceleração possível.

- SVE Matriz das velocidades secundárias. Cada variável especifica a tensão de saída que será lançada no canal analógico a cada passo. Aceita valores na faixa 0 a 255, correspondendo de forma linear a tensões entre -10 e +9,92 V (128 ==> 0 V). Se omitido, a saída permanece em 0 V.

- IIN A variável associada especifica o índice inicial das matrizes que irão programar o primeiro passo na placa MOT. Se omitido, o primeiro passo corresponderá ao índice zero da matriz. Aceita valores na faixa 0 a 255. {nota 2}

- IFN A variável associada especifica o índice final das matrizes que irão programar o último passo na placa MOT. Se omitido, o compilador assume o índice 9 da matriz. Aceita valores na faixa 0 a 255. {nota 2}

ü ***MOTOR r FILTRO [KP=www] [KI=yyy] [KD=zzz] [ESC=xxx] [DES=vvv] [TOL=uuu] [IAD=aaa] [LIG=bbb] [MOD=t]***

Programa os valores gerais relativos ao sistema de controle do módulo MOT. Esta instrução será executada por transição, isto é, uma vez a cada entrada no estado. Não é permitido incluir mais de uma instrução desse tipo num único estado.

- KP A variável associada especifica a constante proporcional do PID da placa MOT. Se omitido, Kp assume o valor 0. Aceita valores na faixa 0 a 32767. {nota 1}

- KI A variável associada especifica a constante integral do PID da placa MOT. Se omitido, Ki assume o valor 0. Aceita valores na faixa 0 a 32767. {nota 1}

- KD A variável associada especifica a constante derivativa do PID da placa MOT. Se omitido, Kd assume o valor 0. Aceita valores na faixa 0 a 32767. {nota 1}

- ESC O conteúdo da variável associada especifica o fator de escala de todas as referências de posição, afetando também a velocidade e aceleração. É expresso em termos de pulsos por unidade representada. Se omitido, assume o valor 1. {nota 1}

- DES Deslocamento (offset) de posição. O conteúdo da variável associada especifica a posição do ponto "Home" do encoder. Se omitido, assume o valor 0.

- TOL O conteúdo da variável associada especifica a tolerância em unidades de posição para a consideração de erro de posicionamento "ERR" do comando "LEITURA". {nota 1}

- IAD O conteúdo da variável associada especifica o intervalo de amostragem derivativo para o controle PID {nota 1}. É calculado pela seguinte fórmula: IAD = (1,164 * Interv. de amostragem em ms) - 1. (Para CPU3) Aceita valores na faixa 0 a 255.

- LIG O conteúdo da variável associada especifica o limite para o acumulador integral do filtro PID. O acumulador integral é o termo (KI. Se(n)) da equação PID. {nota 1}. Aceita valores na faixa 0 a 32767.

- MOD Define o modo geral de operação para a placa MOT. O número 1 especifica que o motor **`r'** é um sistema DC com encoder (modo por omissão). O número 2 especifica que o motor **`r'** é um sistema com motor de passo.

Definições básicas do filtro PID da placa MOT:
O sinal analógico de saída para atuação do motor tem o seu valor determinado pela seguinte equação:

U(n) = KP.e(n) + KI.Se(n) + KD.[e(n') - e(n'-1)] onde:

U(n) - Sinal de saída para atuação na amostragem n.

E(n) - Erro de posição no instante da amostragem n.

Se(n) - Soma integral dos erros de posição desde a inicialização até a amostragem n.

E(n) - Erro de posição no instante da amostragem derivativa n'.

E(n'-1) - Erro de posição no instante da amostragem derivativa imediatamente anterior a n'.

N - Índice da amostragem básica. A taxa de amostragem básica é fixa e serve para renovar o cálculo do erro e incrementar a soma integral. Na placa MOT com CPU3 é igual a 856 us.

N' - Índice da amostragem derivativa. A taxa de amostragem derivativa é igual a um múltiplo da amostragem básica, programado em IAD.

Especificação do Fator Escala

O fator escala “ESC” na placa MOT é um recurso importante para converter o número de pulsos do encoder para uma unidade física conveniente em cada caso. Explicaremos o mecanismo de cálculo através de um exemplo:

"Temos um sistema mecânico de movimentação linear tracionado por uma rosca sem fim ligada ao eixo de um encoder de 2500 ciclos por volta. O passo da rosca é de 5mm. Calcular o fator de escala para que o controlador programável trabalhe com as variáveis em centésimos de milímetro. "

Um encoder de 2500 ciclos por volta gera 4*2500 = 10000 pulsos na MOT por volta completa. O passo de 5mm (500 centésimos) equivale a 10000 pulsos na MOT. Devemos programar 20 em "ESC" dentro do comando "FILTRO".

ü ***MOTOR r HOME [VEL[N]=yyy]***

Instrui a placa MOT a buscar o ponto de referência inicial de posicionamento. Omitindo o parâmetro "VEL", o registro de posição é resetado imediatamente, sem movimentar o motor.

Quando existir o parâmetro "VEL", o motor **`r'** se move até que encontre o sinal Home:

- VEL O conteúdo da variável **`yyy'** define a velocidade para busca do Home. Com o sinal "N", a busca se dá no sentido negativo, caso contrário, a busca é feita no sentido positivo.

ü ***MOTOR r LEITURA [POR=www] [VER=yyy] [MOV=www] [ERR=zzz] [BLQ=ttt]***

Esse comando adquire informações na unidade de posicionamento para tratamento em alto nível. Os parâmetros permitem especificar as variáveis que receberão estas informações:

- POR Posição real instantânea. Carrega a posição do eixo **r** na variável **www** no momento da execução da instrução. A unidade de medida adotada é a mesma utilizada na programação - número de pulsos do encoder, a menos que o comando FILTRO defina outra unidade.

- VER Velocidade real instantânea. Carrega a posição do eixo r na variável **yyy** no momento da execução da instrução. A unidade de medida adotada é a mesma utilizada na programação - número de pulsos do encoder por segundo, a menos que o comando FILTRO defina outra unidade. {nota 1}

- MOV A variável especificada assume o valor 1 enquanto o motor estiver se movendo, qualquer que seja o comando. Ao terminar o movimento ou no caso de uma interrupção, a variável assume o valor zero. {nota 3}

- ERR A variável especificada assume o valor 1 caso o erro de posicionamento exceda a tolerância especificada no comando "FILTRO". Caso contrário, a variável assume o valor zero. {nota 1}

- BLQ A variável especificada assume o número do bloco corrente em execução. Essa opção tem sentido apenas durante a execução de uma seqüência. {nota 3}
A maioria dos comandos associados a instrução `MOTOR' são executados quando da transição do estado, não sendo repetidos durante a permanência no estado. Da mesma forma que nas instruções `FACA', `IMPRIME' e outras, existem flags para atuação única. Ao entrar no estado, o sistema executa uma vez todas as instruções `MOTOR' existentes, inibindo a repetição nas varreduras seguintes. A única exceção ocorre com o comando `LEITURA'. Esse comando não é inibido e fica sendo repetido enquanto o estado estiver ativo.
Afora os comandos descritos, o sistema operacional possui a variável de atribuição fixa `MOTER' que indica o número de módulos ativos no sistema e o código de erro relativo a inicialização ou operação das placas.

Nota 1
Essas opções são válidas apenas dentro do modo para motor DC. No modo para motor de passo não tem sentido ou não foram desenvolvidas no sistema.

Nota 2 :
Devido ao ciclo de carga dos parâmetros na placa MOT, a duração do ciclo de varredura que executa a instrução ‘SEQUENCIA’ ´é fortemente influenciada pela diferença entre os fatores ‘IFN’- ‘IIN´. Tipicamente, o tempo de varredura aumenta 0,5 ms para cada unidade.

Nota 3 :
As informações passadas por esses comandos se referem ao movimento teórico, isto é, ao movimento previsto pela MOT com base nos objetivos de posição, velocidade e aceleração. O comando `ERR' é usado para informar se o mecanismo está realmente cumprindo o objetivo.

- ***PID:r jjj=VS RE=aaa MD=bbb [KP=ccc] [KI=ddd] [KD=eee] [TA=fff] [DE=ggg] [LS=hhh] [LI=iii]***

Esta instrução executa uma rotina PID no controlador BCM. Se colocada associada a um estado, a rotina será executada enquanto o estado estiver ativo. Se for colocada independente de estado, a rotina ficará operando permanentemente.
O parâmetro **`r'** define o número da rotina instalada. Podem ser definidas até 32 rotinas independentes com parâmetros próprios. Se várias rotinas PID forem definidas com o mesmo número, a última definição ativada invalidará todas as anteriores. Essa circunstância permite a definição de rotinas PID com parâmetros diferentes para diferentes fases do processo, conforme a necessidade.

A instrução PID executa automaticamente a cada ciclo a seguinte equação:

VS = Kp * Err + Ki * SErr.dT + Kd * dErr/dT

Onde: Err é o erro de controle (Referência - Medição).
SErr é a soma de todos os erros acumulados.

Em relação a um controlador analógico ideal, a rotina PID executada digitalmente impõe uma série de detalhes a serem observados:

a) O diferencial de tempo, usado para cálculo do erro diferencial e integrado, definido no sistema ideal como infinitamente pequeno, é substituído pelo conceito de Tempo de Amostragem (TA). Esse valor, programável pelo usuário, determina o intervalo sobre o qual o erro é somado no acumulador de erro; sendo também usado para calcular a variação diferencial das leituras tomadas entre duas amostragens.

Definido desta maneira, a equação realmente executada fica sendo a seguinte:

VS = DE + Kp * Err + Ki * J(r) + Kd * (Err - Err')

Onde: J(r) é a soma dos erros acumulados a cada período de amostragem.
Err' é o erro calculado no período de amostragem anterior.
DE é uma constante programável somada a variável de saída.

b) As variáveis usadas não possuem resolução e capacidade infinita. Deverão ser sempre observados os seguintes limites:

VS, MD, RE, DE	- Valores positivos na faixa 0 a 65535 (16 bits).
kp, Ki, Kd	- Valores positivos na faixa 0,004 a 255 (16 bits).
Err	- Valores em complemento de 2 na faixa -32768 a 32767 (16 bits).
J(r)	- Valores positivos na faixa 0 a 65535 (16 bits).

Os cálculos internos são todos desenvolvidos em complemento de 2, com 24 bits.

c) O software é projetado para que, em cada ciclo do CP, uma rotina PID seja atendida. Deve haver cuidado para que o tempo de amostragem escolhido seja sempre maior que o tempo requerido pelo sistema para atendimento de todas as rotinas.

d) Para aperfeiçoar a seqüência de controle, é provido um recurso lógico para evitar o disparo do acumulador integral, fenômeno que provoca um atraso muito grande na estabilização do sistema. A atualização do acumulador integral fica associada ao desempenho da saída. Assim, o acumulador não é incrementado se a saída estiver no limite superior (LS). Da mesma forma, o acumulador não é decrementado se a saída estiver no limite inferior. Isso estabelece que a ação integral só ocorre dentro da chamada faixa proporcional.

A seguir são apresentados os parâmetros de controle acessíveis ao usuário. Estes podem ser constantes ou variáveis, conforme as definições:

VS Variável de saída do controle. Pode ser qualquer variável não indexada. A faixa de saída vai de 0 a 65535 a menos que haja uma limitação via LS ou LI.

RE Nome da variável ou constante da referência para o controle. Pode ser qualquer variável não indexada. Deve estar na faixa 0 a 65535.

MD Nome da variável de medição do processo. Pode ser qualquer variável não indexada. Deve estar na faixa 0 a 65535.

KP Coeficiente proporcional. Pode ser uma constante ou variável não indexada. Se constante, deve estar na faixa de 0,004 a 255. Se variável, o conteúdo da variável será dividido por 256 para entrar no algoritmo de controle. Se omitido, o coeficiente proporcional assume o valor 1.

KI Coeficiente integral. Pode ser uma constante ou variável não indexada. Se constante, deve estar na faixa de 0,004 a 255. Se variável, o conteúdo da variável será dividido por 256 para entrar no algoritmo de controle. Se omitido, assume o valor 0.

KD Coeficiente derivativo. Pode ser uma constante ou variável não indexada. Se constante, deve estar na faixa de 0,004 a 255. Se variável, o conteúdo da variável será dividido por 256 para entrar no algoritmo de controle. Se omitido, assume o valor 0.

TA Tempo de amostragem em segundos. Constante que define o intervalo entre incrementos do acumulador integral e entre dois cálculos do erro diferencial. Deve estar na faixa 0,1 a 25,5. Se omitido assume o valor 1. Apenas uma definição de TA é admissível no programa e será válida para todas as rotinas PID existentes.

DE Deslocamento (offset) de saída. Constante na faixa 0 a 65535 que desloca a saída em relação ao valor calculado pelo algoritmo básico.

LS Limite superior de saída. Constante na faixa 0 a 65535 que limita o valor máximo da variável de saída. Se omitido, a saída pode assumir qualquer valor.

LI Limite inferior de saída. Constante na faixa 0 a 65535 que limita o valor mínimo da variável de saída. Se omitido, a saída pode assumir qualquer valor.

Observação: Ao colocar constantes nos parâmetros PID, o separador decimal deve ser ponto, e não vírgula (notação inglesa).

Exemplo:

A rotina a seguir implementa um controle PID com saída PWM, modulando a saída digital 1. As variáveis de referência e os coeficientes podem ser ajustados em outras malhas ou em outros equipamentos através da rede de comunicação.

MALHA 0:
PID:1 VSAI=VS RE=TEMP MD=TEMPM KP=COEFP KI=COEFI TA=0.5 LS=100

ESTADO 0:
DESL 1
SE VSAN>0 E DEC>100 ENTAO 1

ESTADO 1:
LIGA 1
RESET DEC
SE DEC>VSAN ENTAO 0

- ***PLIGA r1{,rn} {<PLIGA><PDESL> s1{,sn}}***
- ***PDESL r1{,rn} {<PLIGA><PDESL> s1{,sn}}***

Liga e/ou desliga os leds da placa PAN especificados em r1,rn,s1,sn quando a malha entra no estado corrente. Não há hierarquia nestas operações. Na ocorrência de instruções conflitantes, o resultado na saída é imprevisível. A sintaxe e lógica de operação é similar a instrução LIGA/DESL, usada para comandar as saídas digitais.

- ***PRESET:<arquivo.pre>,n***

Esta instrução é usada para presetar variáveis, equivalendo a uma série encadeada de intruções FACA. O conteudo do arquivo ***arquivo.pre*** descreve as variáveis que devem ser presetadas e os seus valores. No momento da compilação, o PROCP codifica o preset de cada variável no formato (objeto) da instrução FACA.
O arquivo em formato ASCII descreve o formato da tela de parametrização, os nomes das variáveis, o valor de preset e uma legenda descritiva do significado de cada variável. O aplicativo parametrizador, integrado ao PROCP, permite a edição do arquivo de preset em um formato gráfico flexível e adequado a cada aplicação.

O parâmetro ‘n´ define a forma que o arquivo de preset será tratado pelo PROCP e pelo Sistema Operacional:
Se ‘n´ for 0 ou omitido: Compila na forma “FACA .... E .....”, ocupando 5 bytes por variável presetada.
Se ‘n´ for 1: Compila na forma “FACA xxx(yyy)={a,b,c,d....}”, ocupando 2 bytes por variável presetada.

ü *PRESET xxx(yyy)=zzz QUANT=kkk*

Carrega um intervalo de uma matriz com um único valor, onde ‘xxx’ é o nome da matriz e ‘yyy’ seu índice, que pode ser constante ou variável. O parâmetro ‘zzz’ é o valor a carregar na matriz, que pode ser constante ou variável não indexada. O parâmetro ‘kkk’ indica a quantidade de elementos a copiar, podendo ser constante ou variável não indexada.
Esta instrução é executada na transição, isto é, é executada somente na primeira vez em que entra no estado.
Recurso disponível para: GP3101, GP3210, GP3220, CPU7-11, CPU8, CPU8A e CPU9.

ü *RESET aaa,bbb,ccc,ddd(iii)*

Coloca o valor 0 nas variáveis mencionadas. Qualquer tipo de variável é aceita, inclusive variáveis matriciais. Equivale à instrução "FACA aaa=0 E bbb=0", etc., com mais clareza na edição, economia de memória e de tempo de processamento.
Podem ser usadas várias instruções RESET em um estado. A única exigência é que as linhas com esta instrução devem ser adjacentes. Além disso, todas as instruções SET, RESET, INCREMENTA e DECREMENTA de um estado devem estar agrupadas.

ü *ROTINA[(xxxx)]=r1.r2.r3...rn*

Esta função permite ao programador inserir uma rotina assembler que será executada uma vez por ciclo do controlador. Os valor r1, r2, etc. são códigos em linguagem de máquina (hex), cuja saída deve terminar sempre com "RET". O compilador aceita qualquer número de linhas com código. Apenas deve haver no máximo 80 colunas por linha e as instruções ***ROTINA*** devem ser as últimas do programa.
O valor **`xxxx'** (obrigatoriamente 4 dígitos hex) é opcional e define o endereço de relocação da rotina. Se for omitido, a rotina será relocada imediatamente após o final da tabela de instruções. Ocorrerá um erro de compilação se o endereço apontado para a relocação for menor que o endereço final da tabela de instruções.

ü *ROTINA[(xxxx)]=(narq)*

Nesta forma, a instrução `ROTINA' permite a inclusão de Blocos de Função disponíveis numa biblioteca, para a execução de funções especiais não abrangidas pela linguagem Descritiva.
O arquivo com o Bloco de Função deve estar presente no mesmo diretório do programa fonte .BCM, caso contrário, um erro específico será gerado.
A instrução ***ROTINA*** deve ser sempre a última num programa em linguagem Descritiva. Conforme as características dos Blocos de Função, podem ser encadeados vários módulos através da inserção de várias instruções `ROTINA', desde que sejam sempre as últimas instruções presentes no programa.

O valor **`xxxx'** define o endereço de alocação do Bloco de Função. A documentação deste irá informar qual o endereço que deve ser usado. Ocorrerá um erro de compilação se o endereço apontado para a relocação for menor que o endereço final da tabela de instruções.
Alguns Blocos de Função permitem que a alocação seja livre e automática. Neste caso, o valor **`xxxx'** pode ser omitido e o Bloco será alocado imediatamente após o final da tabela de instruções.
Para os produtos GP3210, GP3220 e CPU9, o endereço será de 32 bits (8 caracteres hex separados 4 a 4 por um ponto). Para os demais produtos, é usado um endereço de 16 bits (4 caracteres hex).

O parâmetro ***(narq)*** define o nome do arquivo do Bloco de Função. Deve ser posto o nome com a extensão do arquivo, conforme descrito na sua documentação.

- ***SE aaa<=><#><<><>>bbb{<E><OU> ccc<=><#><>><<>ddd}***
  ENTAO r

A malha corrente muda para o estado **`r'** caso se cumpra as condições previstas. A variável **`aaa'** pode ser de qualquer tipo; o operando **`bbb'** pode ser constante ou variável, não podendo ser variável indexada ou de entrada digital. Estão disponíveis os testes `igual' (=); `diferente' (#); maior que (>) e `menor que' (<). A expressão é avaliada sequencialmente da esquerda para direita, sendo que não há hierarquia na seqüência de testes.

Exemplo:

DISPLAY=99.10
INTERFACE1(9600)=1
MAQUINA 0:
ESTADO 0:
SE ALFA#4 OU VAR1=VAR2 ENTAO 20

- ***SE aaa<=><#><<><>>bbb {<E><OU>ccc<=><#><>><<>ddd} <LIGA><DESL> r1{,rn}***

A saída **`r'** é ligada ou desligada caso se cumpram as condições previstas. **`aaa'** deve ser uma variável de qualquer tipo; **`bbb'** pode ser variável ou constante. Estão disponíveis os testes `igual' (=); `diferente' (#); maior que (>) e `menor que' (<). A expressão é avaliada sequencialmente da esquerda para direita. Não há hierarquia na seqüência de testes.

EXEMPLO:
DISPLAY=20.9999
INTERFACE1(9600)=1
MAQUINA 0:

ESTADO 0:
SE ENT1<ENT2 E ENT3>6 LIGA 8,10,12

- ***SE{(}aaa<=><#><>><<>bbb{)}<E><OU> {(}ccc<=><#><<><>>ddd {)}} ENTÃO r***

A malha corrente mudará para o estado “r” caso se cumpram as condições previstas. A variável “aaa” poderá ser de qualquer tipo; o operando “bbb” poderá ser constante ou variável, não podendo ser variável matricial com índice variável nem entrada digital. Pelo uso de parênteses poderá ser dado prioridade na seqüência dos testes. Os testes disponíveis serão: “igual” (=), “diferente” (#), “maior que” (>) e “menor que” (<).
Recurso disponível para: CPU7-11, CPU8, CPU8A e CPU9.

Observações:

- Quando a variável “aaa” for de entrada digital, a variável “bbb” obrigatóriamente deverá ser constante (valor 0 ou 1). Neste caso, o único teste disponível será o de “igual” (=).
- Esta instrução admite até 8 níveis de parênteses.

- ***SE {(}aaa<=><#><>><<>bbb{)} {<E><OU> {(}ccc<=><#><<><>> ddd{)}} ENTÃO fff=yyy [SENÃO ggg=zzz]***

A variável “aaa” poderá ser de qualquer tipo; o operando “bbb” poderá ser constante ou variável, não podendo ser variável matricial com índice variável nem entrada digital. As variáveis “fff” e “ggg” não poderão ser indexadas, nem de entrada digital. Os operandos “yyy” e “zzz” poderão ser constante ou variável, não podendo ser variável matricial com índice variável nem entrada digital. Se o resultado final da operação for verdadeiro, “fff” assumirá o valor de “yyy”, senão permanecerá inalterado ou “ggg” assumirá o valor “zzz”, caso exista a diretiva “SENÃO”. Pelo uso de parênteses poderá ser dado prioridade na seqüência dos testes. Os testes disponíveis serão: “igual” (=), “diferente” (#), “maior que” (>) e “menor que” (<).
Recurso disponível para: CPU7-11, CPU8, CPU8A e CPU9.

Observações:

- Quando a variável “aaa” for de entrada digital, a variável “bbb” obrigatóriamente deverá ser constante (valor 0 ou 1). Neste caso, o único teste disponível será o de “igual” (=).
- Esta instrução admite até 8 níveis de parênteses

ü *SE ATRASO=xx{ <E> aaa<=><#><>><<>bbb} ENTAO r*

A malha corrente muda para o estado **`r'** após decorrido o tempo especificado na constante **`xx'** (em décimos de segundo) e caso se cumpram as condições previstas. Não há hierarquia na seqüência de testes. **`aaa'** deve ser uma variável de qualquer tipo; **`bbb''** pode ser variável ou constante. Estão disponíveis os testes `igual' (=); `diferente' (#); maior que (>) e `menor que' (<).

Observações:

- O termo ***ATRASO=xx*** deve ser obrigatoriamente o primeiro na seqüência de testes.
- Somente o teste ‘igual' (=) é admissível no termo ***ATRASO***.
- Somente é permitida uma linha com ***ATRASO*** por estado.

Exemplo:

DISPLAY=99.99999
INTERFACE1(9600)=1
MAQUINA 1:
ESTADO 0:
SE ATRASO=40 E VAR#ENT ENTAO 1

ü *SE ATRASO=xx {<E><OU> {(}aaa<=><#><>><<>bbb{)}} ENTÃO r*

A malha corrente mudará para o estado “r” após decorrido o tempo especificado na constante ”xx” (em décimos de segundo) e/ou caso se cumpram as condições previstas. A variável “aaa” poderá ser de qualquer tipo; o operando “bbb” poderá ser constante ou variável, não podendo ser variável indexada ou de entrada digital. Pelo uso de parênteses poderá ser dado prioridade na seqüência dos testes. Os testes disponíveis serão: “igual” (=), “diferente” (#), “maior que” (>).
Recurso disponível para: GP3101, GP3210, GP3220, CPU7-11, CPU8, CPU8A e CPU9.

Observações:

- Somente é permitida uma linha com ***ATRASO*** por estado.
- Quando a variável “aaa” for de entrada digital, a variável “bbb” obrigatóriamente deverá ser constante (valor 0 ou 1). Neste caso, o único teste disponível será o de “igual” (=).
- Esta instrução admite até 8 níveis de parênteses

- ***SE ATRASO=xx {<E><OU> {(}aaa<=><#><>><<>bbb{)}} ENTÃO r [SENÃO ggg=zzz]***

A variável “aaa” poderá ser de qualquer tipo; o operando “bbb” poderá ser constante ou variável, não podendo ser variável indexada ou de entrada digital. As variáveis “fff” e “ggg” não poderão ser indexadas, nem de entrada digital. Os operandos “yyy” e “zzz” poderão ser constante ou variável, não podendo ser variável indexada nem de entrada digital. Se o resultado final da operação for verdadeiro, “fff” assumirá o valor de “yyy”, senão permanecerá inalterado ou “ggg” assumirá o valor “zzz”, caso exista a diretiva “SENÃO”. Pelo uso de parênteses poderá ser dado prioridade na seqüência dos testes. Os testes disponíveis serão: “igual” (=), “diferente” (#), “maior que” (>) e “menor que” (<).
Recurso disponível para: GP3101, GP3210, GP3220, CPU7-11, CPU8, CPU8A e CPU9.

Observações:

- Quando a variável “aaa” for de entrada digital, a variável “bbb” obrigatóriamente deverá ser constante (valor 0 ou 1). Neste caso, o único teste disponível será o de “igual” (=).
- Esta instrução admite até 8 níveis de parênteses

- ***SE ATRASO=xx {<E><OU>aaa<=><#><>><<>bbb} LIGA r1{,rn}***
  SE ATRASO=xx {<E><OU>aaa<=><#><>><<>bbb} DESL r1{,rn}

As saídas ´**r1, ... rn**´são ligadas ou desligadas caso se cumpram as condições previstas. **`aaa´**deve ser uma variável de qualquer tipo; **`bbb´** pode ser variável ou constante. Estão disponíveis os testes “igual” (=); “diferente” (#); “maior que” (>) e “menor que” (<). A expressão é avaliada sequencialmente da esquerda para direita. Não há hierarquia na seqüência de testes.

ü *SET aaa,bbb,ccc,ddd(iii)*

Coloca o valor 1 nas variáveis mencionadas. Qualquer tipo de variável é aceita, inclusive variáveis matriciais. Equivale à instrução "FACA aaa=1 E bbb=1", etc., com mais clareza na edição, economia de memória e de tempo de processamento.

Podem ser usadas várias instruções SET em um estado. A única exigência é que as linhas com esta instrução devem ser adjacentes. Além disso, todas as instruções SET, RESET, INCREMENTA e DECREMENTA de um estado devem estar agrupadas.

ü *VA PARA aaa*

Mudança imediata do estado corrente para o estado `aaa' independentemente de qualquer condição. O parâmetro `aaa' pode ser uma constante ou uma variável, quando usado o controlador GP3210, GP3220, CPU8, CPU8A e CPU9. Para os demais modelos de CLP, só são aceitas constantes em `aaa'.

Exemplo:

DISPLAY=9.100
INTERFACE1(9600)=1
MAQUINA 0:

ESTADO 0:
VA PARA 1

ESTADO 1:

2.5 - EXEMPLOS TÍPICOS DE PROGRAMAÇÃO

2.5.1 - Leitura de Entradas Digitais

As entradas digitais são interfaces que recebem e interpretam informações do processo sob controle.

Uma entrada digital pode assumir dois níveis lógicos:

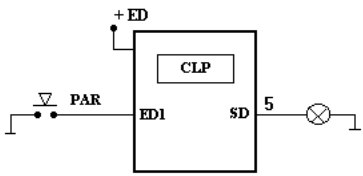
Nível Lógico	Significado	Representação Elétrica
1	Sensor acionado/contato fechado	
0	Sensor desacionado/contato aberto	

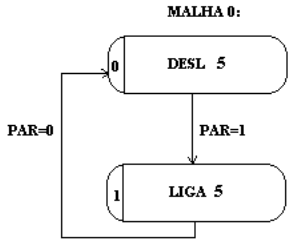
Na linguagem descritiva BCM as entradas digitais são consideradas um tipo específico de variável. Esta variável **DEVE** ser associada à respectiva entrada física do CP em que estiver conectada no instante da edição do programa no compilador PROCP.

LEMBRE-SE:O nome de uma variável deve ter de 3 a 5 caracteres, sendo que o primeiro caracter deve ser SEMPRE uma letra.

A leitura de uma entrada digital é feita através da instrução condicional **SE.**

Exemplo: - Leitura de Entradas Digitais





Listagem do programa

CLP=BCM1086	;Especifica tipo de CP a ser utilizado
INTERFACE1(9600)=1	;Habilita canal serial RS232 na velocidade de 9600 baud
DISPLAY=DPA	;Habilita funcionamento do display alfanumérico
MALHA 0:	;Especifica número da malha de controle
ESTADO 0:	;Especifica número do estado corrente
DESL 5	;Mantém saída 5 desligada
SE PAR=1 ENTAO 1	;Se variável PAR for acionada salta p/o estado 1
ESTADO 1:	;Especifica número do estado corrente
LIGA 5	;Mantém saída 5 ligada
SE PAR=0 ENTAO 0	;Se variável PAR for desacionada retorna ao estado 0 (zero)
FIM	;Informa fim da edição do programa

LEMBRE-SE: A variável **PAR** deve ser associada (declarada) à uma entrada física do CP quando da edição do programa no PROCP.

2.5.2 - Controle de Saídas Digitais

Exemplo: - Acionamento de Saídas Digitais

CLP=BCM1086	;Especifica tipo de CP a ser utilizado
INTERFACE1(9600)=1	;Habilita canal serial RS232 na velocidade de 9600 baud
DISPLAY=DPA	;Habilita tratamento do display alfanumérico
MALHA 0:	;Especifica número da malha de controle
ESTADO 0:	;Especifica número do estado corrente
DESL 1.2.3	;Mantém saídas 1,2 e 3 desligadas
LIGA 4.5.6	;Mantém saidas 4,5 e 6 ligadas
FIM	;Informa fim da edição do programa

2.5.3 - Atribuição de Valor para Variáveis

Em determinadas situações pode ser necessário fazer com que uma variável assuma determinado valor. Existem três situações onde isso pode ser necessário:

* Fazer com que uma variável assuma valor **0** - RESET
* Fazer com que uma variável assuma valor **1** - SET
* Fazer com que uma variável assuma valor**>1** - PRESET

Para estes casos pode-se utilizar as seguintes instruções:

IMPORTANTE: Estas instruções não se aplicam às variáveis de entradas digitais, entradas analógicas e variáveis aplicadas à placa CTR.

Exemplo: Atribuir UM valor para uma variável

CLP=BCM1086	;Especifica tipo de CP a ser utilizado
INTERFACE1(9600)=1	;Habilita canal serial RS232 na velocidade de 9600 baud
DISPLAY=DPA	;Habilita funcionamento do display alfanumérico
MALHA 0:	;Especifica número da malha de controle
ESTADO 0:	;Especifica o número do estado corrente
FACA VAR3=5	;Preseta VAR3 com valor igual a 5

Exemplo: Atribuir valor ZERO para uma variável

CLP=BCM1086	;Especifica tipo de CP a ser utilizado
INTERFACE1(9600)=1	;Habilita canal serial RS232 na velocidade de 9600 baud
DISPLAY=DPA	;Habilita funcionamento do display alfanumérico
MALHA 0:	;Especifica o número da malha de controle
ESTADO 0:	;Especifica número do estado corrente
RESET VAR1	;Reseta variável VAR1

Exemplo: Atribuir valor 1 para uma variável

CLP=BCM1086	; Especifica tipo de CP a ser utilizado
INTERFACE1(9600)=1	;Habilita canal serial RS232 na velocidade de 9600 baud
DISPLAY=DPA	;Habilita funcionamento do display alfanumérico
MALHA 0:	;Especifica o número da malha de controle
ESTADO 0:	;Especifica número do estado corrente
SET VAR1	;Seta variável VAR1

Para monitorar as variáveis do programa, o PROCP tem um recurso chamado “Monitora Variáveis”. Este recurso pode ser acessado através do “**Menu Monitora”** opção **“Variáveis”**. Veja o manual do software de desenvolvimento para linguagem descritiva PROCP.

2.5.4 - Uso de Contadores

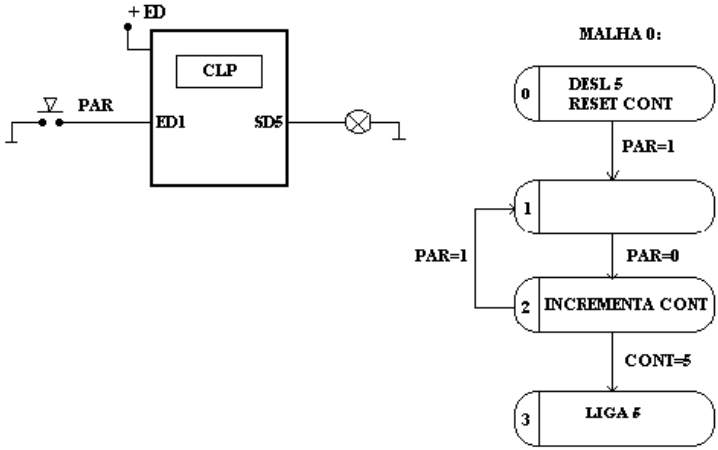
A linguagem descritiva BCM permite que sejam criados inúmeros contadores internos. Para a criação de um contador interno basta usar uma variável de atribuição livre.
De um modo geral, estas variáveis devem ser resetadas ou presetadas, conforme o caso, no estado **0** da malha em que forem utilizadas.

LEMBRE-SE: O nome de uma variável deve ter de 3 a 5 caracteres, sendo que o primeiro caracter deve ser SEMPRE uma letra.

Contadores Crescentes

Podem ser utilizadas duas instruções para implementar um contador crescente.

Exemplo: - Contador Crescente - Ligar uma lâmpada após 5 toques no botão PAR



Listagem do Programa

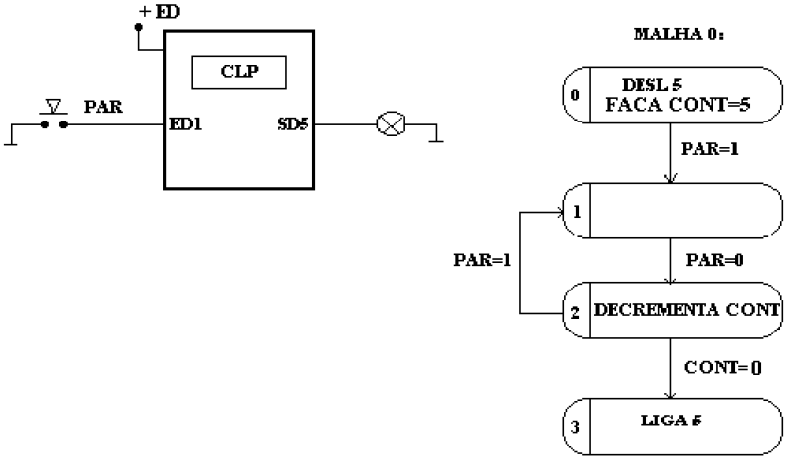
CLP=BCM1086	;Especifica o tipo de CP a ser utilizado
INTERFACE1(9600)=1	;Habilita canal serial RS232 na velocidade de 9600 baud
DISPLAY=DPA	;Habilita funcionamento do display alfanumérico
MALHA 0:	;Especifica número da malha de controle
ESTADO 0:	;Especifica o número do estado corrente
DESL 5	; Mantém saída digital 5 desligada
RESET CONT	; Reseta variável interna CONT
SE PAR=1 ENTAO 1	;Testa se botão PAR foi acionado e salta para o estado 1
ESTADO 1:	; Especifica o número do estado corrente
SE PAR=0 ENTAO 2	; Testa se botão PAR foi desacionado e salta para o estado 2
ESTADO 2:	; Especifica o número do estado corrente
INCREMENTA CONT	; Incrementa em uma unidade a variável CONT
SE PAR=1 ENTAO 1	; Se botão PAR foi acionado, retorna ao estado 1
SE CONT=5 ENTAO 3	; Se variável CONT atingiu valor 5, muda para estado 3
ESTADO 3:	; Especifica número do estado corrente
LIGA 5	; Mantém saída digital 5 ligada
FIM	; Informa FIM da edição do programa

LEMBRE-SE: A variável **PAR** deve ser associada à uma entrada física do CP quando da edição do programa no PROCP.
A variável CONT, por ser uma variável interna do programa, não precisa ser declarada.

Contadores Decrescentes

Podem ser utilizadas duas instruções para implementar contadores decrescentes.

Exemplo: - Contador Decrescente - Ligar uma lâmpada após 5 toques no botão PAR.



Listagem do Programa

CLP=BCM1086	;Especifica tipo de CP a ser utilizado
INTERFACE1(9600)=1	;Habilita canal serial RS232 na velocidade de 9600 baud
DISPLAY=DPA	;Habilita o display alfanumérico
MALHA 0:	;Especifica número da malha de controle
ESTADO 0:	;Especifica o número do estado corrente
DESL 5	;Mantém saída digital 5 desligada
FACA CONT=5	;Preseta variável interna CONT com valor igual a 5
SE PAR=1 ENTAO 1	;Se botão PAR for acionado vai p/estado 1

```
ESTADO 1:              ;Especifica o número do estado corrente
SE PAR=0 ENTAO 2       ; Se botão PAR for desacionado salta p/estado
2

ESTADO 2:              ; Especifica o número do estado corrente
DECREMENTA CONT        ; Decrementa em uma unidade a variável
                       CONT

SE PAR=1 ENTAO 1       ; Se botão PAR foi acionado, retorna ao estado
                       1
SE CONT=0 ENTAO 3      ; Se variável CONT atingiu valor 0, muda para
                       estado 3

ESTADO 3:              ; Especifica número do estado corrente
LIGA 5                 ; Mantém saída digital 5 ligada
FIM                    ; Informa FIM da edição do programa
```

LEMBRE-SE: A variável **PAR** deve ser associada à uma entrada física do CP quando da edição do programa no PROCP.

A variável CONT, por ser uma variável interna do programa, não precisa ser declarada.

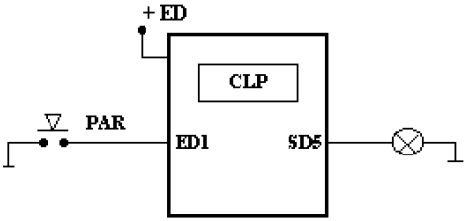
2.5.5 - Uso de Temporizadores

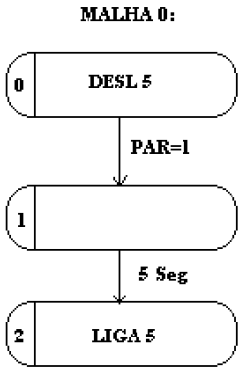
Freqüentemente aparece a necessidade da utilização de uma “Condição de Transição Temporizada” (***regras 14 e 15 do diagrama de estados)***. Neste caso a transição do estado corrente para o estado de destino está condicionada a passagem de um determinado tempo.

A linguagem descritiva BCM dispõe de três recursos para esta função:

1º) Utilizando-se a instrução condicional SE ATRASO.

Exemplo: - Uso da Instrução SE ATRASO - Ligar uma lâmpada, 5 segundos após pressionado o botão PAR.



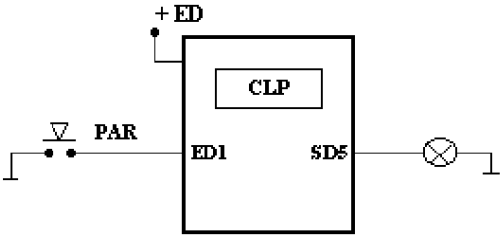


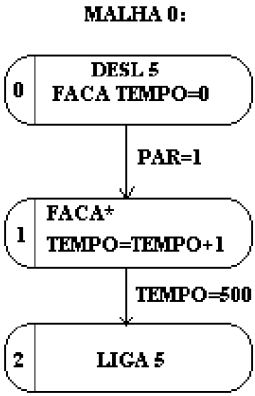
Listagem do Programa

```
CLP=BCM1086              ; Especifica tipo de CP a ser utilizado
INTERFACE1(9600)=1       ;Habilita canal serial RS232 na velocidade de
                         9600 baud
DISPLAY=DPA              ;Habilita funcionamento do display alfanumérico

MALHA 0:                 ; Especifica número da malha de controle
ESTADO 0:                ; Especifica o número do estado corrente
DESL 5                   ; Mantém saída digital 5 desligada
SE PAR=1 ENTAO 1         ; Se o botão PAR for acionado salta para o
                         estado 1.
ESTADO 1:
SE ATRASO=50 ENTAO 2 ; Aguarda 5 segundos (50 décimos) e muda
                         para estado 2
ESTADO 2:                ; Especifica número do estado corrente
LIGA 5                   ; Mantém saída digital 5 ligada
```

2º) Utilizando-se a instrução FACA.

Exemplo: - Utilização de um Temporizador Interno





Listagem do Programa

```
CLP=BCM1086              ; Especifica tipo de CP a ser utilizado
INTERFACE1(9600)=1       ; Habilita canal serial RS232 na velocidade de
                         9600 baud
DISPLAY=DPA              ; Habilita funcionamento do display
alfanumérico

MALHA 0:                 ; Especifica número da malha de controle
ESTADO 0:                ; Especifica o número do estado corrente
FACA TEMPO=0             ; Cria a variável TEMPO com valor inicial 0
(zero)
DESL 5                   ; Mantém saída digital 5 desligada
SE PAR=1 ENTAO 1         ;Se botão PAR for acionado vai p/estado 1

ESTADO 1:                ; Especifica o número do estado corrente
FACA* TEMPO=TEMPO+1;Incrementa temporizador interno TEMPO
SE TEMPO=500 ENTAO 2;Testa se decorreu 5 seg. (500 centésimos) e
                         muda para o estado 2
```

```
ESTADO 2:        ; Especifica o número do estado corrente
LIGA 5           ; Mantém saída digital 5 ligada
FIM              ; Informa fim da edição do programa
```

3º) Utilizando as variáveis de atribuição fixa para temporização (HOR, MIN,SEC,DEC).

A linguagem descritiva BCM dispõe de um grupo de variáveis de atribuição fixa, já definidas no Sistema Operacional, para utilização em temporização. São elas:

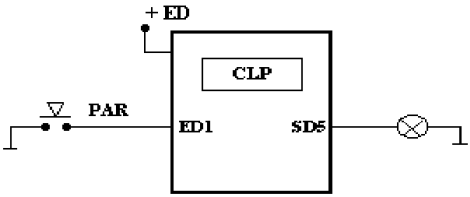
Variável	Significado
HOR	Contador de Horas
MIN	Contador de Minutos
SEC	Contador de Segundos
DEC	Contador de Décimos de Segundo
CEN	Contador de Centésimos de Segundo

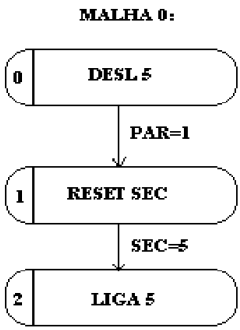
Estas variáveis são inicializadas e incrementadas, na sua unidade de tempo, a partir do instante que o CP é energizado.

A qualquer instante o programador pode resetar ou presetar qualquer uma destas variáveis de forma independente uma das outras.

Deve haver o cuidado que, ao escrever em uma dessas variáveis, a mesma não demorará o tempo característico dela para incrementar pela primeira vez. Por exemplo, ao zerar a variável MIN, o primeiro incremento ocorrerá em um tempo aleatório, que pode variar de zero até 1 minuto.

Exemplo: - Utilização das variáveis de atribuição fixa para Temporização





Listagem do Programa

CLP=BCM1086	; Especifica tipo de CP a ser utilizado
INTERFACE1(9600)=1	; Habilita canal serial RS232 na velocidade de 9600 baud
DISPLAY=DPA	; Habilita funcionamento do display alfanumérico
MALHA 0:	; Especifica número da malha de controle
ESTADO 0:	; Especifica o número do estado corrente
DESL 5	; Mantém saída digital 5 desligada
RESET SEC	; Reseta contador de segundos
SE SEC=5 ENTAO 1	; Aguarda 5 segundos e muda para estado 1
ESTADO 1:	; Especifica o número do estado corrente
LIGA 5	; Mantém saída digital 5 ligada
FIM	; Informa o fim da edição do programa

2.5.6 - Uso de Temporizadores associados ao teste de uma variável

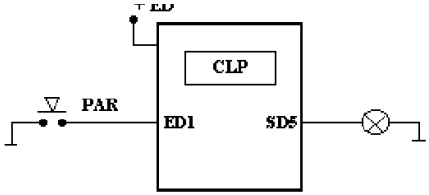
A regra 15 do diagrama de estados diz que “em determinadas situações pode ser necessário combinar uma condição temporizada com um ou mais testes de variáveis”.

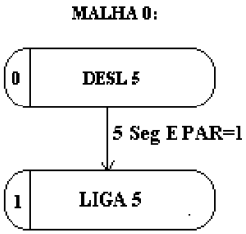
Isto pode ocorrer em duas situações:

1º - Utilizando-se o conectivo E.

2º - Utilizando-se o conectivo OU.

Exemplo: - Ligar uma lâmpada após 5 seg somente se o botão PAR estiver pressionado





Listagem do Programa

```
CLP=BCM1086              ; Especifica tipo de CP a ser utilizado
INTERFACE1(9600)=1       ; Habilita canal serial RS232 na velocidade de
                           9600 baud
DISPLAY=DPA              ; Habilita o display alfanumérico

MALHA 0:                 ;Especifica número da malha de controle
ESTADO 0:                ; Especifica o número do estado corrente
DESL 5                   ; Mantém Entradas digital 5 desligada
SE ATRASO=50 E PAR=1 ENTAO 1 ; Muda para o estado 1 após 5
  seg. se o botão PAR estiver pressionado
```

```
ESTADO 1:          ; Especifica o número do estado corrente
LIGA 5             ; Mantém saída digital 5 ligada
FIM                ; Informa o fim de edição do programa
```

2.5.7 - Leitura de Entradas Analógicas

As entradas analógicas são interfaces que recebem e interpretam sinais elétricos variáveis provenientes de sensores do processo sob controle, onde a grandeza medida é transportada através da intensidade deste sinal elétrico.

De um modo geral os sensores analógicos fornecem sinais elétricos padronizados, proporcional as grandezas medida, tais como 0 a 5 Vdc, 0 a 10 Vdc, 0 a 20mA, 4 a 20mA entre outros.

Através de conversores A/D (analógico/digital) estes sinais são convertidos em sinais digitais cuja resolução depende do conversor A/D utilizado.

Resolução	Valor Decimal
8 bits	0 a 255
12 bits	0 a 4095
16 bits	0 a 65.535

Por exemplo:
Um sensor/transmissor de nível, com escala para até 100 litros e saída 4 a 20mA, conectado a uma placa de entradas analógicas com resolução de 8 bits.

```
0 litros    50 litros   100 litros
4mA         12mA        20mA
    |-----------|-----------|
  0           125         250
```

Significado:
A variável que está associada à entrada analógica, onde estiver conectado este sensor, assumirá valores inteiros na faixa de 0 a 250 proporcional ao nível medido (0 a 100 litros).

Se conectarmos este mesmo sensor a uma placa de entradas analógicas com resolução de 12 bits:

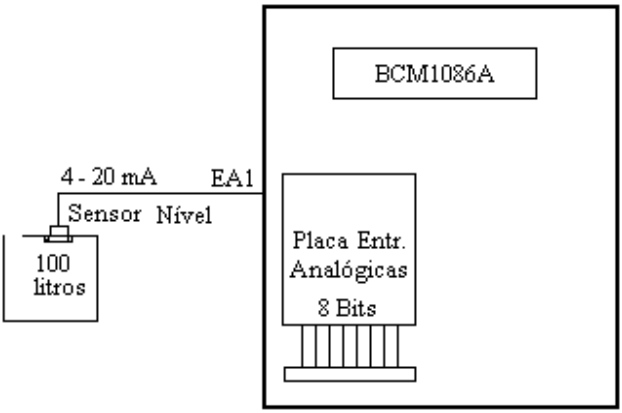
```
 0        50 litros      100 litros
4mA       12mA           20mA
  |-----------|-----------|
 0          2000          4000
```

A variável que está associada à entrada analógica, onde estiver conectado este sensor, assumirá valores inteiros na faixa de 0 a 4000 proporcional ao nível medido (0 a 100 litros).
Comparação:

Escala do Sensor	**Variável Entrada Analógica**	
	8 bits	12 bits
0 a 100 litros	2 unid/litro	40 unid/litro

Para leitura de uma entrada analógica utiliza-se a instrução condicional SE.
Normalmente é necessário fazer-se uma conversão de escala por software, isto é, transformar o conteúdo da variável de entrada analógica (8 ou 12 bits) em um valor correspondente a faixa de operação do sensor utilizado.

Exemplo:
Converter o sinal elétrico de 4-20mA fornecido por um transmissor de nível, conectado a uma placa de entradas analógicas com resolução de 8 bits, em uma escala de 0 a 100 litros.



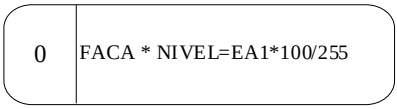
Solução:
Como a placa de entradas analógicas possui resolução de 8 bits, logo, o sinal de 4 a 20mA, corresponde a um valor decimal de 0 a 255.
A equação a ser utilizada é a seguinte:

NIVEL=EA1*F.esc/Resolução

Onde:
NIVEL - é a variável interna do programa que representa o nível instantâneo do tanque.
EA1 - é o valor da entrada analógica, já convertido em valor decimal correspondente a 8 bits (0 a 255).
F.esc - representa o fundo de escala desejado (100 litros).
Resolução - valor decimal correspondente à resolução do conversor A/D utilizado na placa de entradas analógicas.

Diagrama de estados

MALHA 0:



Listagem do Programa

```
CLP=BCM1086              ; Especifica tipo de CP
INTERFACE1(9600)=1       ;Habilita Canal RS232 em 9600 Bauds cód. acesso 1
ESA=1                    ;Habilita tratamento da interface de E/S analógicas
RTC                      ;Habilita tratamento do relógio/calendário
DISPLAY=DPA              ;Habilita tratamento display alfanumérico

MALHA 0:                 ; Especifica número da malha corrente
ESTADO 0:                ; Especifica número do estado corrente
FACA* NIVEL=EA1*100/255; Converte sinal de EA1 em escala de 0 a 100 litros
FIM                      ; Indica fim da edição do programa
```

LEMBRE-SE: A variável **EA1** deve ser associada (declarada) à entrada física do CP em que estiver conectado o sensor, durante a edição do programa no PROCP. NIVEL é uma variável interna do programa e, portanto, não precisa ser declarada.

2.5.8 - Controle de Saídas Analógicas

As saídas analógicas são interfaces que recebem e interpretam valores binários fornecidos pela CPU, e convertem estes valores em sinais elétricos proporcionais ao valor recebido.

De um modo geral os conversores D/A e circuitos associados, fornecem sinais elétricos padronizados, tais como 0 a 5Vdc, 0 a 10Vdc, 0 a 20mA, 4 a 20mA entre outros.

Os valores permitidos dependem do conversor D/A utilizado:

Resolução	Valor Decimal
8 bits	0 a 255
12 bits	0 a 4095
16 bits	0 a 65.535

O conteúdo das variáveis de saídas analógicas é controlado através dos resultados aritméticos na instrução FACA. Ao ser ligado o CP, todas as variáveis desse subgrupo são zeradas.

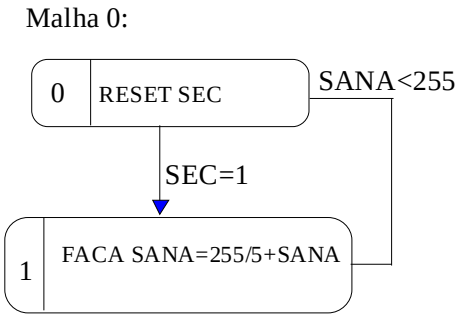
Exemplo:
Incrementar em 20% do fundo de escala a cada segundo o valor de uma saída analógica com resolução de 8 bits.

0	1s	2s	3s	4s	5 segundos
0	20	40	60	80	100%
0	51	102	153	204	255

Neste caso, o valor da saída analógica será atualizado a cada segundo através do resultado da seguinte equação:

SANA=255/5+SANA
Onde:
SANA - é a variável da saída analógica
255 - resolução do conversor D/A
5 - constante que corresponde a 20% do fundo de escala

Diagrama de Estados:



Listagem do Programa

Edição do Programa:

CLP=BCM1086	; Especifica tipo de CP
INTERFACE1(9600)=1	;Habilita Canal RS232 em 9600 Bauds cód. acesso 1
ESA=1	;Habilita tratamento da interface de E/S analógicas
RTC	;Habilita tratamento do relógio/calendário
DISPLAY=DPA	; Habilita tratamento display alfanumérico
MALHA 0:	; Especifica número da malha corrente
ESTADO 0:	; Especifica número do estado corrente
RESET SEC	; Reseta contador de segundos
SE SEC=1 ENTAO 1	;Se decorreu 1 segundo após o reset de SEC salta p/estado 1

ESTADO 1:

FACA SANA=255/5+SANA;Incrementa em 20% o valor da Saída Analógica

SE SANA<255 ENTAO 0	;Se SANA ainda é menor que 255 (100%) retorna ao estado 0
FIM	; Informa fim da edição do programa

LEMBRE-SE:A variável **SANA** deve ser associada (declarada) à uma saída física do CP quando da edição do programa no PROCP.

2.5.9 - Uso de Display

Display Alfanumérico DPA
Este recurso está disponível nos CPs BCM1086A, GP3000 e BCM2085 com módulo Display DPA instalado

O controle de apresentação de mensagens e variáveis no display alfanumérico é feito unicamente através da instrução MOSTRA.

Atributo	Função
T	Limpa a Tela antes de apresentar a nova mensagem
L	limpa a Linha antes de apresentar a nova mensagem
Dn	Alinha o conteúdo da variável à direita, com “n” digitos
Z	Inclui os zeros à esquerda dos valores, se houver “Dn”
Sn	Condiciona a apresentação ao valor da variável de atribuição fixa WWW, igual a “n”. O parâmetro “N” pode estar na faixa de 0 a 99.

Podem ser utilizados vários atributos combinados, se necessário.

Limitações e cuidados especiais no uso da instrução “MOSTRA”

a - Em cada instrução “MOSTRA” poderão ser definidos até três elementos, mas apenas uma variável.

b - O tamanho máximo das mensagens definidas nas instruções “MOSTRA” é 40 caracteres (32 para BCM2085 com CPU4 instalado). Mensagens ou variáveis que iniciem na primeira linha e não caibam inteiramente nela continuarão na segunda linha.

c - Informações (variáveis ou mensagens) carregadas no display por uma instrução “MOSTRA” só desaparecerão do display se forem eliminadas por outra instrução “MOSTRA”.

d - Uma variável que tenha sido carregada no display terá seu valor corrente atualizado automaticamente pelo Sistema Operacional a partir do momento em que houver sido carregada e até o momento em que seja eliminada do display, mesmo que o programa saia do estado que contém a instrução “MOSTRA” que carregou a variável no display.

e - Variáveis carregadas no display por uma instrução “MOSTRA” sem atributo de apresentação condicional só serão eliminadas pelos atributos “T” e “L”.

Carregar uma mensagem ou outra variável na posição ocupada por umavariável carregada anteriormente não elimina esta variável, e causa conflito no display.

f - A execução de uma instrução “MOSTRA” com atributo de apresentação condicional eliminará do display todas as informações (mensagens e variável) que houverem sido carregadas pela execução anterior de outra instrução “MOSTRA” com atributo de apresentação condicional, mesmo que esta instrução tenha sido executada em outro estado ou mesmo em outra máquina do programa.

g - A execução simultânea de duas ou mais instruções “MOSTRA” com atributo de apresentação condicional faz com que apenas as informações referentes à última delas sejam apresentadas.

h - Informações que excedam a área do display não serão apresentadas, e o display não apresentará nenhuma indicação da ocorrência deste erro. Assim, se uma variável for programada para começar na posição 2,15 e o seu valor corrente tiver cinco dígitos, apenas os dois mais significativos serão apresentados, e a informação apresentada no display não será válida.

i - Se houver um elemento mensagem após uma variável com alinhamento pela esquerda, o tamanho desta mensagem não poderá exceder 7 caracteres. Um elemento mensagem colocado após uma variável com alinhamento pela esquerda na mesma instrução “MOSTRA” terá sua posição de apresentação no display variável conforme o número de dígitos do valor corrente da variável correspondente.

j - No caso de o espaço previsto para a variável ser menor do que o número de dígitos do seu valor corrente, ocorre a sobreposição do valor da variável sobre a mensagem. Neste caso, se o número de dígitos do valor corrente da variável diminuir, fazendo desaparecer a sobreposição, a mensagem original será recuperada.

k - Não ocorre a apresentação de todos os dígitos do valor corrente de uma variável se estes não couberem inteiramente na área do display. É importante notar que neste caso o display não apresenta nenhuma indicação de erro, ao contrário do que ocorre se uma variável for apresentada com o atributo “Dn” e o número de dígitos do seu valor corrente for maior do que o parâmetro “n”(ocorrência de overflow), caso em que o espaço reservado para a variável é preenchido com asteriscos.

- **Uso dos Atributos da Instrução “MOSTRA**”:

Atributo “T”(Limpa Tela) : A inclusão deste atributo em uma instrução “MOSTRA” faz com que todas as mensagens e variáveis carregadas anteriormente na tela sejam apagadas.

Atributo “L” (Limpa a Linha): A inclusão deste atributo em uma instrução “MOSTRA” faz com que as mensagens ou variáveis que aparecem na linha em questão sejam apagadas do display. É importante notar que no caso de uma mensagem que ocupe um pedaço de cada linha, somente será apagado a parte contida na linha que estiver sendo apagada. No caso de variáveis, serão apagadas todas as variáveis cuja primeira posição se localize na linha em questão.

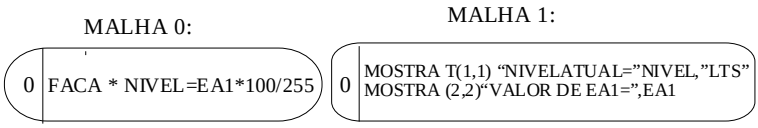
Exemplo - Apresentação de Mensagens

```
CLP=BCM1086              ; Especifica tipo de CP
INTERFACE1(9600)=1       ; Habilita Canal Serial RS232 na velocidade de
                         em 9600 Bauds
DISPLAY=DPA              ; Habilita o display alfanumérico

MALHA 0:                 ; Especifica número da malha de controle
ESTADO 0:                ; Especifica número do estado corrente
MOSTRA T(1,1)” MANUAL PROGRAMAÇÃO”
MOSTRA (2,2)”BCM Engenharia”
SE ATRASO=50 ENTAO 1          ;Aguarda 5 segundos e salta p/estado 1

ESTADO 1:                ; Especifica o número do estado corrente
MOSTRA L(2,1)”USO INSTRUÇÃO MOSTRA”
SE ATRASO=50 ENTAO 0          ;Aguarda 5 segundos e retorna ao
                         ESTADO 0.
```

Observe que no estado 0, do exemplo acima, o atributo **T** limpa a tela antes de apresentar a nova mensagem e, no estado 1 o atributo **L** limpa apenas a linha 2 antes de apresentar a nova mensagem. É necessário utilizar uma instrução MOSTRA para cada linha do display.

Exemplo - Apresentação de Mensagens e Variáveis - Mostrar no display o valor do nível de um tanque.

Diagrama de estados:




```
CLP=BCM1086              ; Especifica tipo de CP
INTERFACE1(9600)=1       ; Habilita Canal serial RS232 na velocidade de
                         9600 Bauds
ESA=1                    ; Habilita E/S analógicas
DISPLAY=DPA              ; Habilita o display alfanumérico

MALHA 0:                 ; Especifica número da malha de controle
ESTADO 0:                ; Especifica número do estado corrente
FACA* NIVEL=EA1*100/255; Converte valor de EA1 em escala 0 a 100
litros

MALHA 1:                 ; Especifica número da malha corrente
ESTADO 0:                ; Especifica número do estado corrente
MOSTRA T(1,1)”NIVEL ATUAL=”,NIVEL,”LTS”
MOSTRA (2,2)”VALOR DE EA1=”,EA1
```

O resultado apresentado no display será:

```
NIVEL ATUAL=100 LTS
VALOR DE EA1=255
```

Atributo “Dn”(Alinha à direita com “n” dígitos): Este atributo modifica a forma de tratamento dos zeros à esquerda da variável a ser apresentada. Se este atributo for omitido, a variável será apresentada com apagamento dos zeros à esquerda e alinhamento pela esquerda. Este atributo fará a variável ser apresentada com apagamento dos zeros à esquerda mas com alinhamento pela direita, sendo reservados “n” dígitos para a variável.

Este atributo influi apenas na apresentação da variável contida na mesma instrução “MOSTRA”, não influenciando na apresentação das demais variáveis da tela.

Atributo “Z” (Apresenta os zeros à esquerda) : Se estiver presente o atributo “Dn”, este atributo faz com que sejam apresentados os zeros à esquerda no valor da variável, até o limite do número de dígitos definido pelo parâmetro “n”. Este atributo influi apenas na apresentação da variável contida na mesma instrução “MOSTRA”, não influenciando na apresentação das demais variáveis da tela.

Se acrescentarmos o atributo Dn para o exemplo acima, com “n” valendo 2:

```
.
.
.
MALHA 1:
ESTADO 0:
MOSTRA TD2(1,1)”NIVEL ATUAL=“,NIVEL,”LTS”
```

Resultado obtido: Será apresentado o valor da variável NIVEL com apenas dois dígitos, alinhados à direita. Quando o valor atingir 3 dígitos (100), irá aparecer no display dois asteriscos.

```
NIVEL ATUAL=**LTS
```

Se acrescentarmos o atributo Dn para o exemplo acima, com “n” valendo 4:

```
.
.
.
MALHA 1:
ESTADO 0:
MOSTRA TD4(1,1)”NIVEL ATUAL=“,NIVEL,”LTS”
```

Resultado obtido: Será reservado um espaço de quatro dígitos, alinhados à direita, para apresentação do valor da variável NIVEL. Como o valor de NIVEL tem no máximo 3 dígitos ficarão em branco os caracteres à esquerda do valor.

```
NIVEL ATUAL=100 LTS
```

Se acrescentarmos o atributo Z para o exemplo acima:

```
.
.
.
MALHA1:
ESTADO0:
MOSTRA TD4Z(1,1)”NIVELATUAL=“,NIVEL,”LTS”
```

Resultado obtido: Será reservado um espaço de quatro dígitos, alinhados à direita, para apresentação do valor da variável NIVEL. Como o valor de NIVEL tem no máximo 3 dígitos, os caracteres à esquerda do valor serão completados com o zeros..

```
NIVEL ATUAL=0100 LTS
```

Atributo
“Sn”(Apresentação condicional): Uma instrução “MOSTRA” que contenha este atributo só será executada se o valor da variável “WWW” for igual ao parâmetro “n”. As mensagens e variáveis que houverem sido carregadas no display por alguma instrução “MOSTRA” com atributo de apresentação condicional serão apagadas quando uma nova instrução “MOSTRA” com atributo de apresentação condicional for executada.

Exemplo - Apresentar no display os níveis médio e crítico para o tanque

```
CLP=BCM1086              ;Especifica tipo de CP
INTERFACE1(9600)=1       ;Habilita Canal Serial RS232 na velocidade de
                         9600 Bauds
ESA=1                          ; Habilita E/S analógicas
DISPLAY=DPA              ; Habilita tratamento display alfanumérico

MALHA 0:                       ; Especifica número da malha de
controle
ESTADO 0:                ; Especifica número do estado corrente
FACA* NIVEL=EA1*100/4095 E WWW=NIVEL/3

MALHA 1:                       ; Especifica o número da malha de
controle
ESTADO 0:                ; Especifica o número do estado corrente
MOSTRA T(1,1)"NIVEL ATUAL=",NIVEL,"lts"
MOSTRA S30(2,4)"NIVEL MEDIO"
MOSTRA S20(2,4)"NIVEL CRITICO"
```

Neste caso a mensagem “NIVEL MEDIO”só será apresentada no display quando a variável WWW for igual a 30 (S30) e a mensagem “NIVEL CRITICO” quando a variável WWW for igual a 20 (S20).

- Uso das Teclas de Função

A interface teclado/display alfanumérico possui quatro teclas de função (F1, F2, F3 e F4) livremente configuráveis pelo usuário.

Estas teclas possuem nomes de variáveis de atribuição fixa já definidas pelo Sistema Operacional, portanto disponíveis para utilização, sem necessidade de declaração.

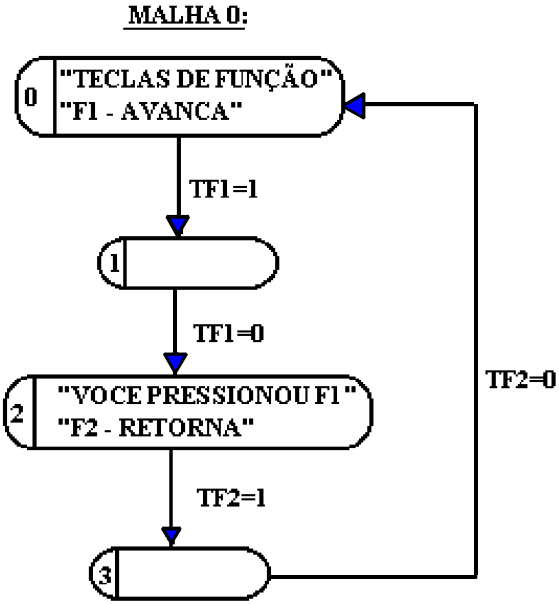
Tecla	Variável
F1	TF1
F2	TF2
F3	TF3
F4	TF4

Do ponto de vista elétrico, estas teclas são contatos NA (Normalmente Aberto) e seu funcionamento é idêntico às entradas digitais.

Situação	Nível Lógico
Tecla em repouso	0 (zero)
Tecla pressionada	1 (Um)

Exemplo - Apresentar mensagens controladas pelas teclas de função.

Diagrama de Estados



Listagem do Programa

```
CLP=BCM1086              ; Especifica tipo de CP
INTERFACE1(9600)=1       ; Habilita Canal RS232 na velocidade de 9600
                         Bauds
DISPLAY=DPA              ; Habilita tratamento display alfanumérico

MALHA 0:                 ; Especifica número da malha de controle
ESTADO 0:                ; Especifica número do estado corrente
MOSTRA T(1,1)"TECLAS DE FUNCAO"
MOSTRA (2,1)"F1 - AVANCA"
SE TF1=1 ENTAO 1         ; Se F1 for pressionada salta p/o estado 1

ESTADO 1:
SE TF1=0 ENTAO 2         ; Se F1 for solta p/o estado 2

ESTADO 2:
MOSTRA T(1,1)"VOCE PRESSIONOU F1"
MOSTRA (2,1)"F2 - RETORNA"
SE TF2=1 ENTAO 3         ; Se F2 for pressionada salta p/o estado 3

ESTADO 3:
SE TF2=0 ENTAO 0         ; Se F2 for solta retorna ao estado 0
```

- Como carregar um valor em uma variável através do teclado/display

O módulo teclado/display alfanumérico é uma importante Interface-Homem- Máquina bidirecional entre o programa e o usuário do CP, isto é, pode ser utilizada para ler ou para escrever valores em variáveis do programa. (Set Point).

As variáveis de display podem ser livremente manejadas no programa, aparecendo em testes e operações aritméticas.

As variáveis desse grupo são sempre retentivas, permitindo que pontos de ajuste, calibrações ou programações, sejam mantidos independente de que o controlador seja mantido ligado.

Para carregar valores em variáveis deve-se utilizar a instrução MOSTRA e a variável na qual se deseja escrever um valor deve ser declarada como canal de display através da janela “Variáveis do Programa” no PROCP.

Para carregar o valor desejado na variável siga os seguintes passos:

1º) Posicione o display de forma a apresentar a variável desejada
2º) Pressione a tecla de cursor até que o valor correspondente a variável fique “piscando”
3º) Digite o novo valor desejado
4º) Pressione a tecla EV para que a variável assuma o novo valor desejado.

Exemplo: - Carregar o valor em uma variável

Listagem do Programa

```
CLP=BCM1086            ; Especifica tipo de CP
INTERFACE1(9600)=1     ; Habilita Canal RS232 numa velocidade de
                       9600 Bauds
DISPLAY=DPA            ; Habilita tratamento display alfanumérico

MALHA 0:               ; Especifica número da malha corrente
ESTADO 0:              ; Especifica número do estado corrente
MOSTRA T(1,1)”VALOR DESEJADO=“,VDES
```

LEMBRE-SE:
A variável **VDES** deve ser associada a um canal de display do CP quando da edição do programa no PROCP.

- Apresentação de valores fracionários no display alfanumérico

Nos modelos de controladores programáveis que trabalham só com variáveis inteiras, é necessária uma técnica adicional para apresentar valores fracionários.

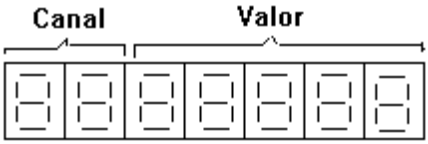
A solução geral consiste em definir uma variável com a grandeza necessária multiplicada por 10, 100 ou 1000, de forma a tornar inteira a parte fracionária. Para apresentação, o valor desta variável será transferido para duas variáveis auxiliares, uma transportando a parte inteira e outra a parte fracionária. Através da instrução Mostra, as duas variáveis são encaeadas e mostradas no display.

Exemplo:
Numa determinada aplicação, a variável TEMP2 deve conter a medida de temperatura com uma casa depois da vírgula. No processo que calcula essa variável irá considerar que a unidade desta será oC*10. Para a apresentação, o trecho de programa deste estado fica:

ESTADO 11:
FACA TMP2I=TEMP2/10 E TMP2F=TMP2I*10 E TMP2F=TEMP2-TMP2F*
MOSTRA D2(1,11)"Tp2:",TMP2I,","
MOSTRA (1,18)TMP2F,"oC"

- Uso do display Numérico

Este recurso está disponível nos CPs BCM1086 e BCM2085 com módulo Display/700 instalado.
A interface display numérico é composto de 7 dígitos divididos em dois campos:



Os dois dígitos à esquerda representam o número do canal corrente de 00 a 99 e os demais dígitos representam o valor da variável do canal corrente.
Os valores admissíveis nessas variáveis estão, em princípio, na faixa de 0 a 65535. O comando DISPLAY(colocado na configuração do CP) pode, no entanto, restringir essa faixa.

As variáveis desse grupo são sempre retentivas, permitindo que pontos de ajuste, calibrações ou programações, sejam mantidos independente de que o controlador seja mantido ligado.
O acesso aos canais do display numérico é feito através da variável de atribuição fixa WWW. Esta variável contém o número do canal do display #1 presente no momento. Quando o usuário muda o canal pelo teclado, a variável transmite essa informação para o programa. No sentido inverso, um valor lançado nessa variável usando a instrução FACA irá fazer com que o display #1 vá para o canal especificado. Aceita valores de 00 a 99.

2.5.10 - Uso de Matrizes

Definição: Matriz é um arranjo de um determinado número de elementos com um significado único. O agrupamento seqüencial desses elementos em linhas e/ou colunas está normalmente associado a uma regra de ocupação determinada.

Uso de Matrizes na Linguagem Descritiva BCM
Com o uso desta função, abre-se todo um espaço de recursos no agrupamento, tratamento e recuperação de informações numéricas.

Aplicações típicas de matrizes:
- Geração de históricos
- Totalização de valores
- etc...

As variáveis utilizadas como matrizes devem ser, obrigatoriamente, variáveis indexadas.

A Linguagem Descritiva BCM possui dois comandos para definição e matrizes:

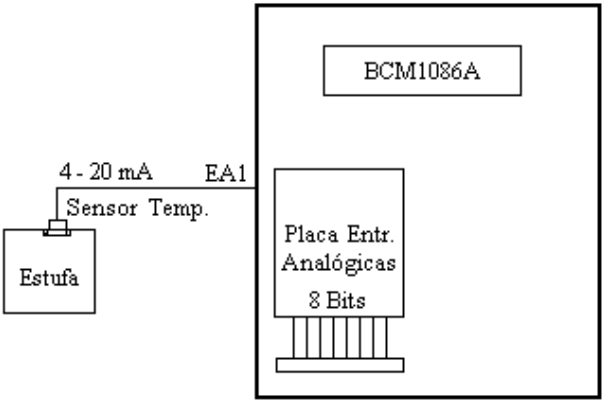
- O comando DIM aaa(xxx) define a dimensão da matriz “aaa”. O número de elementos pode estar na faixa de 0 a 32767.

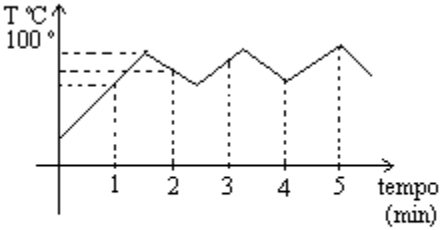
- O comando DIMD aaa(xxx) define que a matriz “aaa” será uma matriz de display, isto é, a cada elemento da matriz corresponderá a um canal do display. O valor “xxx” não terá importância, pois uma matriz de display sempre terá dimensão igual a 99.

O programa do usuário pode trabalhar livremente com uma matriz de display, sem prejuízo das variáveis definidas como variáveis de display na janela “Variáveis do Programa” no PROCP Win ou no menu EDITA na versão DOS.

LEMBRE-SE: O comando DIM deve ser declarado junto dos comandos de configuração do CP, antes da edição do programa. Ver comandos de configuração do CP.

Exemplo: Calcular a temperatura média no interior de uma estufa a cada 5 minutos. O intervalo entre cada leitura deve ser de 1 minuto.





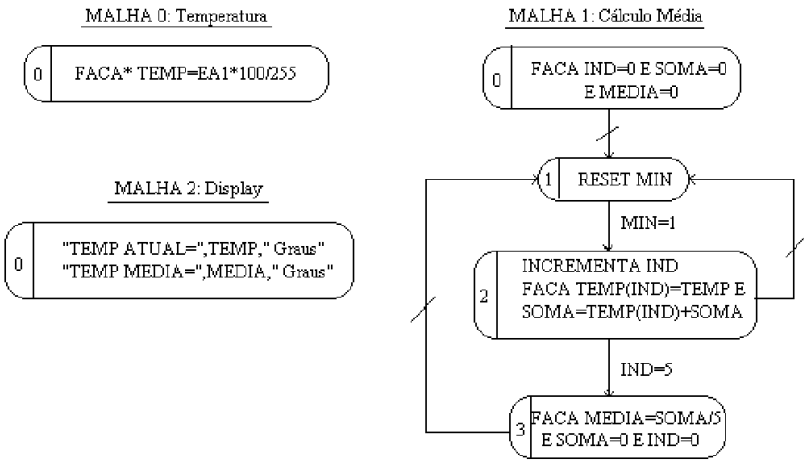
Curva característica da temperatura na estufa.

Solução:
Para armazenar as leituras ao longo do tempo, devemos criar um vetor indexado pelo tempo no qual será armazenado o valor da temperatura a cada minuto.

Para facilitar a compreensão, vamos dividir a solução em malhas de controle.

MALHA	FUNÇÃO
0	Conversão do sinal 4-20mA em escala de temperatura de 0 a 100ºC
1	Leitura, armazenamento dos dados e cálculo da temperatura média.
2	Apresentação dos valores no display

Diagrama de estados:

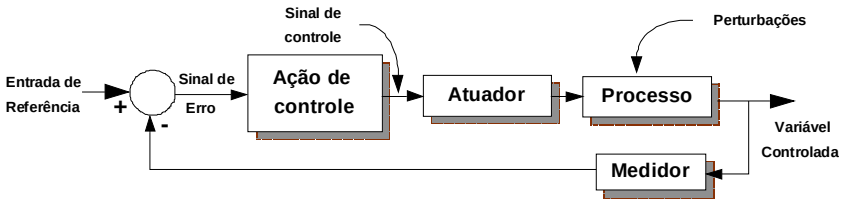


Listagem do Programa

```
CLP=BCM1088              ; Especifica tipo de CP
INTERFACE1(9600)=1       ; Habilita Canal Serial RS232 na velocidade de
                         9600 Bauds
ESA=1                    ;Habilita tratamento da interface de E/S
                         analógicas
RTC                      ; Habilita tratamento do relógio/calendário
DIM TEMP(5)              ; Define a matriz TEMP com 5 elementos
DISPLAY=DPA              ; Habilita tratamento display alfanumérico

MALHA 0:                 ; Especifica número da malha de controle
ESTADO 0:                ; Especifica número do estado corrente
FACA* NIVEL=EA1*100/255; Converte sinal de EA1 em escala de 0 a 100
graus
MALHA 1:                 ; Especifica número da malha de controle
ESTADO 0:                ; Especifica número do estado corrente
FACA IND=0 E SOMA=0 E MEDIA=0
VA PARA 1

ESTADO 1:                ; Especifica número do estado corrente
RESET MIN                ; Reseta Temporizador de Minutos
SE MIN=1 ENTAO 2         ; Se já passou 1 minuto salta p/o estado 2

ESTADO 2:                ; Especifica número do estado corrente
INCREMENTA IND           ; Incrementa Índice do Vetor
FACA TEMP(IND)=TEMP E SOMA=TEMP(IND)+SOMA
SE IND=5 ENTAO 3           ; Se Índice é igual 5 (5 minutos) salta p/estado
                         3
VA PARA 2                ; Retorna ao estado 2, caso IND seja <5

ESTADO 3:                ; Especifica número do estado corrente
FACA MEDIA=SOMA/5 E SOMA=0 E IND=0
VA PARA 1

MALHA 2:                 ; Especifica número da malha de controle
ESTADO 0:                ; Especifica número do estado de controle
MOSTRA T(1,1)”TEMP ATUAL=“,TEMP,” Graus”
MOSTRA (2,1)”TEMP MEDIA=“,MEDIA,” Graus”
FIM                      ;Informa fim da edição do programa
```

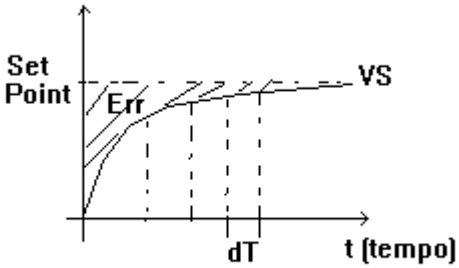
2.5.11 - Controle PID

Controle de processo é a técnica de **MEDIR** uma variável, **COMPARAR** este valor medido com um valor de referência e **AGIR** sobre esta variável de maneira que a diferença entre o valor medido e o valor desejado tenda a zero.



O Controle PID é um método de controle em laço fechado que consiste em aplicar ao atuador o sinal resultante da soma das funções proporcional, integral e derivativa aplicadas ao erro observado na variável medida.

Graficamente temos:



VS=Kp * Err + Ki * Serr * dT + Kd * dErr / dT

Onde: VS é o sinal de saída que será aplicado ao atuador
Err é a diferença entre o valor medido e o valor desejado (Setpoint)
dT é a diferença de tempo entre as amostragens
Serr é a soma dos erros ao longo do tempo
dErr é a diferença de erros - erro atual - erro anterior

A instrução para execução de uma rotina PID na Linguagem Descritiva BCM é a seguinte:

Considerações sobre o uso da instrução PID

1º) Se colocada associada a um estado, a rotina será executada enquanto o estado estiver ativo. Se colocada independente de estado, a rotina ficará operando permanentemente.

2º) Se várias rotinas forem definidas com o mesmo número, a última definição ativada invalidará todas as anteriores. Essa circunstância permite a definição de rotinas PID com parâmetros diferentes para diferentes fases do processo, conforme a necessidade.

A instrução PID executa automaticamente a cada ciclo a seguinte equação:

$$VS=Kp * Err + Ki * SErr.Dt + Kd * dErr/dt$$

Onde: Err é o erro de controle (Referência - Medição)
Serr é a soma de todos os erros acumulados

Em relação a um controlador analógico ideal, a rotina PID executada digitalmente impõe uma série de detalhes a serem observados.

a) O diferencial de tempo, usado para cálculo do erro diferencial e integrado, definido no sistema ideal como infinitamente pequeno, é substituído pelo conceito de Tempo de Amostragem (TA). Esse valor, programável pelo usuário, determina o intervalo sobre o qual o erro é somado no acumulador de erro; sendo também usado para calcular a variação diferencial das leituras tomadas entre duas amostragens.

Definido desta maneira, a equação realmente executada fica sendo a seguinte:

VS=DE + Kp * Err + Ki * J(r) + Kd * (Err-Err')

Onde:

* J(r) é a soma dos erros acumulados a cada período de amostragem
* Err' é o erro calculado no período de amostragem anterior
* DE é uma constante programável somada a variável de saída (em complemento de "2")

b) As variáveis usadas não possuem resolução e capacidade infinita. Deverão ser sempre observados os seguintes limites:

VS, MD, RE, DE- Valores positivos na faixa de 0 a 65535 (16 bits)
Kp, Ki, Kd- Valores positivos na faixa 0,004 a 255 (16 bits)
Err- Valores em complemento de 2 na faixa -32768 a 32767 (16 bits)
J(r) - Valores positivos na faixa 0 a 65535 (16 bits)

Os cálculos internos são todos desenvolvidos em complemento de 2, com resolução de 24 bits.

c) O software é projetado para que, em cada ciclo do CP, uma rotina PID seja atendida. Deve haver cuidado para que o tempo de amostragem escolhido seja SEMPRE maior que o tempo requerido pelo sistema para atendimento de todas as rotinas.

d) Para aperfeiçoar a sequência de controle, é provido um recurso lógico para evitar o disparo do acumulador integral, fenômeno que provoca um atraso muito grande na estabilização do sistema. A atualização do acumulador integral fica associada ao desempenho da saída. Assim, o acumulador não é incrementado se a saída estiver no limite superior (LS). Da mesma forma, o acumulador não é decrementado se a saída estiver no limite inferior (LI). Isso estabelece que a ação integral só ocorre dentro da chamada faixa proporcional.

A seguir são apresentados os parâmetros de controle acessíveis ao usuário. Estes podem ser constantes ou variáveis, conforme as definições:

VS	Variável de saída de controle. Pode ser qualquer variável não indexada. A faixa de saída vai de 0 a 65535 a menos que haja uma limitação via LS ou LI.
RE	Nome da variável ou constante da referência para o controle. Pode ser qualquer variável não indexada. Deve estar na faixa de 0 a 65535.
MD	Nome da variável de medição do processo. Pode ser qualquer variável não indexada. Deve estar na faixa de 0 a 65535.
Kp	Coeficiente proporcional. Pode ser uma constante ou variável não indexada. Se constante, deve estar na faixa de 0,004 a 255. Se variável, o conteúdo da variável será dividido por 256 para entrar no algoritmo de controle. Se omitido, o coeficiente proporcional assume o valor 1.
Ki	Coeficiente integral. Pode ser uma constante ou variável não indexada. Se constante, deve estar na faixa de 0,004 a 255. Se variável, o conteúdo da variável será dividido por 256 para entrar no algoritmo de controle. Se omitido, o coeficiente proporcional assume o valor 0.
Kd	Coeficiente derivativo. Pode ser uma constante ou variável não indexada. Se constante, deve estar na faixa de 0,004 a 255. Se variável, o conteúdo da variável será dividido por 256 para entrar no algoritmo de controle. Se omitido, o coeficiente proporcional assume o valor 0.
TA	Tempo de amostragem em segundos. Constante que define o intervalo entre os incrementos do acumulador integral e entre dois cálculos do erro diferencial. Deve estar na faixa 0,1 a 25,5. Se omitido assume o valor 1. Apenas uma definição de TA é admissível no programa e será válida para todas as rotinas PID existentes.
DE	Deslocamento (offset) de saída. Constante na faixa de 0 a 65535 que desloca a saída em relação ao valor calculado pelo algoritmo básico.
LS	Limite superior de saída. Constante na faixa de 0 a 65535 que limita o valor máximo da variável de saída. Se omitido, a saída pode assumir qualquer valor.
LI	Limite inferior de saída. Constante na faixa de 0 a 65535 que limita o valor mínimo da variável de saída. Se omitido, a saída pode assumir qualquer valor.

OBS.: Ao colocar constantes nos parâmetros PID, o separador decimal deve ser o ponto, e não vírgula (notação inglesa).

Aplicações com a rotina PID

1º) Caso - Atuando somente sobre a constante proporcional

Para constante proporcional igual a 1:

Kp=1
Ki=0
Kd=0

Se aplicarmos estes valores na equação, teremos:

VS=Kp * Err + Ki * Serr + Kd * dErr / dT

VS=1 * Err + 0 * Serr + 0 * dErr / dT

VS=ErrComo: Err = Referência - Valor Medido

VS= (Ref - MD)

Neste caso o PID gera um sinal de saída VS que depende apenas da Referência (Setpoint) e do Valor Medido.

Se Ref > MD=> VS>0(número positivo) aplica tensão na saída

Se Ref < MD=> VS<0(número negativo) desliga saída

VS pode ser uma saída digital (controle PWM) ou analógica (controle proporcional).

Para constante proporcional igual a K . Onde K deve ser um valor maior que 1.

Se aplicarmos estes valores na equação, teremos:

VS=Kp * Err + Ki * Serr + Kd * dErr / dT

VS=K * Err + 0 * Serr + 0 * dErr / dT

VS=ErrComo: Err = Referência - Valor Medido

VS=K (Ref - MD)

Logo:

Se Ref > MD=> VS>>0(número positivo) aplica tensão na saída

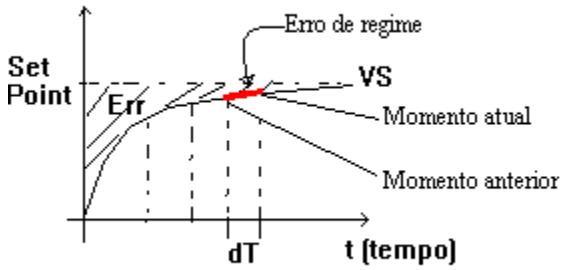
Se Ref < MD=> VS<<0(número negativo) desliga saída

Por Exemplo:

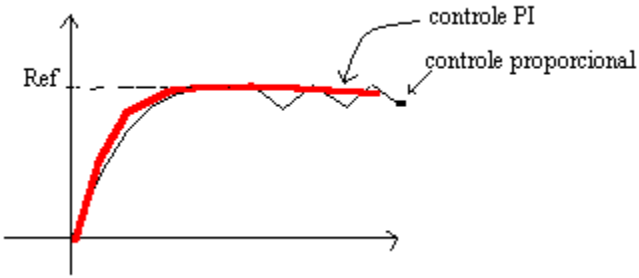
Para K = 1000 e Err = -1 logo a saída assume um valor VS= - 1000

Para corrigir este problema (saída com valor muito menor que zero) foi incluído a constante de integração Ki.

A constante integral compara o valor atual com o valor anterior (determinado pelo tempo de amostragem).



O Controle Proporcional e Integral (PI) leva o regime permanente próximo da referência.



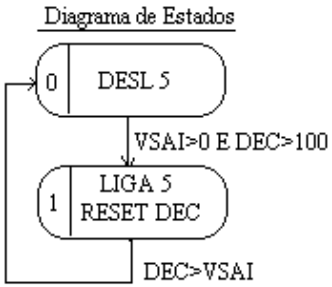
O Controle PI pode levar o sistema a ficar instável se os valores para as constantes Kp ou Ki forem muito grandes.
O controle PI (proporcional e Integral) faz com que o sistema tenha uma resposta lenta quando houver uma variação no valor medido. Para compensar isto deve-se incrementar a constante derivativo (Kd).
O controle derivativo observa qual é a taxa de variação do erro. É geralmente usado quando deseja-se uma resposta muito rápida em relação a uma variação do valor medido.
É importante observar que o uso do controle derivativo pode acontecer uma oscilação em regime permanente. Para diminuir esta variação deve-se atuar sobre a constante proporciona.

Como ajusta um controle PID

Os passos a seguir servem apenas como orientativos para ajustar-se um controle PID.

1. Determinar uma valor para Kp e Zero para Ki e Kd
2. Verificar o gráfico; se VS estiver variando muito, diminuir o valor de Kp até que a variação seja mínima.
3. Quando atingir um valor medido que seja estável em regime permanente, ajustar o valor de Ki. O ajuste de Ki irá diminuir o erro em regime permanente para que o valor medido fique o mais próximo possível do valor de referência. O aumento de Ki deve ser gradativo. Observar se o aumento de Ki não causa uma instabilidade na constante proporcional.

Exemplo: Implementar uma rotina de controle PID com saída PWM, modulando a saída digital 5. As variáveis de referência e os coeficientes podem ser ajustados com outras malhas ou em outros equipamentos através da rede de comunicação.



Listagem do Programa

```
CLP=BCM1086                    ;Especifica tipo de CP
INTERFACE 1(9600)=1            ;Habilita canal serial RS232 na velocidade de 9600
                               bauds
ESA=1                          ;Habilita tratamento de E/S analógicas
DISPLAY=DPA                    ;Habilita tratamento display alfanumérico

MALHA 0:                       ; Especifica o número da malha de controle
PID:1 VSAI=VS RE=TEMP MD=TEMPM KP=COEFP KI=KOEFI TA=0.5 LS=100

ESTADO 0:                      ;Especifica número do estado corrente
DESL 5                         ;Mantém saída digital 5 desligada
SE VSAI>0 E DEC>100 ENTAO 1

ESTADO 1:                      ;Especifica número do estado corrente
LIGA 5                         ;Mantém saída digital 5 ligada
RESET DEC                      ;Reseta contador de décimos de segundos
SE DEC>VSAI ENTAO 0   ;Verifica se DEC é maior que VSAI e retorna ao estado 0
```

2.5.12) Exemplo de uso dos bancos de memória

Este exemplo simples mostra a forma de uso dos bancos de memória. O programa apresenta no display o conteúdo de variáveis matriciais que refletem o conteúdo de dois bancos de memória. A matriz BANKX() pode ler/escrever o conteúdo do banco 1 ou do Banco 2, conforme seja pressionada a tecla F1 ou F2.

```
;       PROGRAMA DE DEMONSTRAÇÃO PARA O USO DE BANCOS DE MEMORIA

;       F1 - Vai para Banco1
;       F2 - Vai para Banco2
;       Definir todas as variáveis não matriciais na área de display.

CLP=BCM3011
INTERFACE1(9600)=1 (PROTOCOLO=BCM2)
NEST=20
RTC
DISPLAY=DPA20
DIM BANK0(2815)                 ;O tamanho deve ser tal que BANKX comece em E000h
DIM BANKX(4000)

MALHA 0:                                ;Apresentação no display
ESTADO 0:
MOSTRA T(1,5)"DEMOBANCO.BCM"
MOSTRA (2,3)"12/abr/2007  17h"
SE ATRASO=50 ENTAO 1

ESTADO 1:
MOSTRA T(1,1)"Banco:",BANCO
MOSTRA (2,1)"INDICE:",IND
MOSTRA (1,11)"BANK0(i):",BANK0(IND)
MOSTRA (2,11)"BANKX(i):",BANKX(IND)

; A matriz BANKX na realidade é múltipla. O valor que se escreve em BANCO permite selecionar um de
até 15 conjuntos de 4000 variáveis. O exemplo mostra uma matriz de 4000 elementos, mas ao invés
disso, conforme o caso, podem ser usadas várias matrizes menores alocadas na área E000h a FFFFh.

;O estado 1 acima apresenta um interface simples para seleção de banco e  entrada/apresentação de
valores.  Observar que a matriz BANK0 (a área de memória que vai de CA00h a DFFFh) é fixa, não
sendo afetada pela seleção de banco.

;É recomendável que as variáveis não matriciais usadas na aplicação seja alocadas na área de display,
de forma a não serem afetadas pela seleção de bancos.
```

MALHA 1:
ESTADO 0:
SE TF1=1 ENTAO 1
SE TF2=1 ENTAO 2

ESTADO 1:
FACA BANCO=1
VA PARA 0

ESTADO 2:
FACA BANCO=2
VA PARA 0

2.6 - TRATAMENTO DE VARIÁVEIS

O sistema adotado para o tratamento de variáveis nos Controladores Programáveis BCM merece uma atenção especial devido a que, uma importante parte do potencial da linguagem BCM está relacionado aos conceitos e regras de operação com variáveis, descritas nesse capítulo.

As variáveis são os elementos que, em última análise transportam as informações das interfaces de entrada para o programa; de uma parte do programa para outra, e do programa para as interfaces de saída. O conteúdo das variáveis, processado de forma adequada, determina o fluxo de todo o programa, controlando as ações a serem executadas e gerando resultados de saída.

Além dessa definição genérica, aplicável a todas as linguagens de programação, alguns tipos de variáveis cumprem funções executivas específicas na linguagem BCM. O uso dessas variáveis, denominadas de atribuição fixa, substituem instruções adicionais, com vantagens de manejo dos procedimentos e legibilidade dos programas. Assim, o estudo da distribuição e funcionamento das variáveis está intimamente ligado ao estudo dos comandos e instruções, uma vez que estas duas estruturas definem os recursos de programação da linguagem.

Uma variável lançada no programa do usuário possui como atributos primários um nome e um endereço definido na memória RAM do controlador, onde o seu conteúdo será armazenado. Esse nome, em alguns casos definido pelo programador, e em outros já padronizado na linguagem, passa a ser a única referência no programa. O endereço de memória onde a variável foi alocada permanece, durante todo o tempo, totalmente invisível para o programador.

Importante: Uma variável não pode assumir duas atribuições ao mesmo tempo. Por exemplo: Não é permitido definir uma variável como entrada analógica e ao mesmo tempo defini-la como variável de display.

Uma característica especial associada as variáveis é a sua **retentividade**. A variável é dita retentiva quando o controlador possui a capacidade de manter o seu valor armazenado mesmo na falta de energia ou no desligamento do controlador. Fisicamente isso é conseguido através do uso de memórias RAM com bateria na placa, para manter os valores na ausência de alimentação externa. Como iremos ver a seguir, alguns grupos de variáveis são retentivas e outras não, conforme a finalidade de cada uma.

É importante observar que, devido a retentividade, as variáveis contém valores aleatórios quando o CP é instalado pela primeira vez ou quando a bateria é substituída. Nos sistemas usados em computadores de uso geral, as variáveis são zeradas na inicialização. Esse fato não pode ser esquecido pelo programador.
Os itens a seguir apresentam de forma detalhada a organização das variáveis na linguagem BCM, dividindo-as em grupos e subgrupos. Serão analisadas as características importantes, restrições e aplicação de cada subdivisão mencionada:

2.6.1 - Variáveis de Atribuição Livre

As variáveis de atribuição livre são as que podem ser criadas pelo programador para atender necessidades próprias da sua lógica. O nome de cada variável pode ser escolhido dentro de qualquer combinação alfanumérica, atendidas as seguintes restrições:

- Os caracteres permitidos em nomes de variáveis são as letras de `A' a `Z', os algarismos de `0' a `9' e o sinal '_' (underscore).
- O primeiro caracter do nome de uma variável deve ser uma letra.
- São permitidos de três a 16 caracteres no nome de uma variável. (Usando o PROCP V5k 4.0 ou superior. Para versões anteriores, é aceito um máximo de 5 caracteres.)

Qualquer variável deste grupo pode participar de operações aritméticas como operando ou resultado. Pode também participar de expressões condicionais na instrução`SE'.Conforme o CP utilizado, as variáveis de atribuição livre podem ser retentivas ou terem os seus valores zerados ao ser ligado o controlador.
Todas as variáveis deste grupo podem assumir valores inteiros e positivos dentro da faixa 0 a 65535 (tipo INT).

Conforme a estruturação, o grupo das variáveis livres podem ser divididas em dois sub-grupos, descritos a seguir:

2.6.1.1 - Variáveis Indexadas

As variáveis indexadas possuem as características já definidas para as variáveis livres, com o recurso adicional de permitir a associação de um índice ao nome. Esse índice, que pode ser uma constante ou um nome de variável, colocado entre parênteses, permite a constituição de matrizes, abrindo todo um espaço de recursos no armazenamento, tratamento e recuperação de informações numéricas.

Exemplo:

```
DIM MAT(40)
FACA IND=3
FACA MAT(0)=10 E MAT(1)=20 E MAT(2)=30
SE MAT(IND)=30 ENTAO 0
```

A expressão condicional da última linha será válida; a matriz `MAT' contém em suas três primeiras linhas os valores 10, 20 e 30. O comando `DIM' da primeira linha do programa reserva espaço para uma matriz chamada `MAT', composta por quarenta e uma variáveis indexadas.

2.6.1.2 - Variáveis não Indexadas

As variáveis não indexadas não permitem a associação do índice. Cada variável pode conter apenas um valor numérico, mantidas todas as características já detalhadas para as variáveis livres. Neste caso não é necessária nenhuma pré-definição. O controlador reconhece e aloca espaço para a variável automaticamente, quando esta é usada pela primeira vez.

2.6.2 - Variáveis de Entrada e Saída

As variáveis de entrada e saída tem a finalidade específica de proporcionar a ligação entre as operações lógicas do programa e as interfaces físicas de entrada/saída do controlador. Para atender esse objetivo, esse grupo de variáveis apresenta as características básicas já definidas, além de outras que serão analisadas em cada sub-grupo.

Os nomes das variáveis desse grupo são livremente escolhidos pelo usuário, respeitando o número e tipos de caracteres utilizáveis. O programador deve definir todos os nomes de variáveis de entrada/saída dentro da janela de variáveis. Para maiores detalhes, veja o capítulo sobre a operação do PROCP.

2.6.2.1 - Entradas Digitais

Cada variável de entrada digital assume o valor “1" ou “0", de acordo com o estado ligado ou desligado da entrada associada. O sistema operacional se encarrega de ler a porta de entrada a cada vez que uma expressão condicional citando tal entrada aparece no programa.
O número de variáveis permitido fica limitado unicamente pelo hardware utilizado, de acordo com o tipo e configuração de CP adotado.
As variáveis desse sub-grupo podem ser usadas na instrução “SE” não sendo permitido o seu uso em operações aritméticas (instrução “FACA”).

Exemplo:

```
MALHA 0:
ESTADO 0:
SE ENT1=1 OU ENT2=0 DESL 8
```

2.6.2.2 - Entradas Analógicas

As variáveis de entradas analógicas são atualizadas a cada ciclo do controlador através da leitura dos conversores A/D dos módulos específicos. O resultado obtido é transferido pelo sistema operacional automaticamente para as variáveis previamente definidas.

Os valores permitidos dependem do interface A/D usado - No controlador BCM2085, por exemplo, a placa ESA lê entradas analógicas em 8 bits, gerando valores na faixa 0 a 255. A placa ANA lê entradas analógicas em 12 bits, gerando valores na faixa 0 a 4095 (o fundo da escala é calibrado para 4000, sendo os valores 4001 a 4095 considerados como overflow).
O número de variáveis permitido, fica limitado unicamente pelo hardware, de acordo com o tipo e configuração de CP adotado.

As variáveis desse sub-grupo podem ser usadas em testes e operações aritméticas, com as instruções `SE' e `FACA'.

Exemplo:

```
MALHA 0:
ESTADO 1:
SE ENTA=40 ENTAO 40
FACA TOT=ENTA*40+4/SOMA+10
```

2.6.2.3 - Saídas Analógicas

As variáveis de saídas analógicas tem o seu conteúdo determinado unicamente pelo programa, através do resultado de operações aritméticas. O sistema operacional se encarrega de transferir automaticamente os conteúdos das variáveis para os conversores D/A disponíveis.
Aqui novamente, os valores permitidos dependem do interface A/D usado. No controlador BCM2085, por exemplo, a placa ESA converte valores na faixa 0 a 255 para uma tensão ou corrente proporcional ao conteúdo da variável, disponível nos bornes de saída. A placa ANA converte valores na faixa de 0 a 4095. O fundo de escala é calibrado para 4000.

O número de variáveis permitido fica limitado unicamente pelo hardware, de acordo com o tipo e configuração de CP adotado.
O conteúdo das variáveis desse sub-grupo são controladas através dos resultados aritméticos na instrução `FACA'. Ao ser ligado o CP, todas as variáveis desse sub-grupo serão zeradas.

Exemplo:

```
MALHA 0:
ESTADO 0:
FACA SANA=VAR1+40/VAR2*4
```

2.6.2.4 - Display

O sistema de display programador proporciona um interface bidirecional eficiente entre o programa e o usuário do controlador. Concebido originalmente para uso nos CLPs com display numérico, as áreas estão disponíveis em todos os modelos de controladores BCM.

As variáveis de display podem ser livremente manejadas no programa, aparecendo em testes e operações aritméticas. O conteúdo destas variáveis irá aparecer no display programador, uma variável em cada canal disponível. A linguagem BCM permite até três displays com cem variáveis cada, endereçadas pelos canais 00 a 99. (Depende da disponibilidade de hardware, conforme o modelo e os módulos disponíveis). A limitação quanto ao número de unidades de display deve ser ainda conferida com o tipo de controlador usado em cada caso.
A característica de bidirecionalidade é dada pelo fato que através do teclado pode- se alterar o conteúdo das variáveis livremente, modificando situações, pontos de ajuste, etc. O usuário pode assim interagir com o programa através dessas variáveis. Mais detalhes sobre esse assunto serão apresentados no capítulo específico sobre operação com display programador.

Os valores admissíveis nessas variáveis estão, em princípio, na faixa de 0 a 65535. O comando `DISPLAY' pode, no entanto, restringir essa faixa - ver a definição dos comandos operacionais.

As variáveis desse grupo são sempre retentivas, permitindo que pontos de ajuste, calibrações ou programações, sejam mantidos independentemente de que o controlador seja mantido ligado.

O uso das áreas de display nos CLPs sem display númerico oferece os seguintes recursos e vantagens:

- Os endereços fixos das variáveis das áreas de display facilitam o referenciamento das variáveis nos processos de comunicação com outros equipamentos.
- Os endereços fixos das variáveis das áreas de display facilitam o referenciamento das variáveis de interface com Blocos de Função.
- Uma série de funções especiais dos Sistemas Operacionais de vários CLPs são manejadas através de variáveis dessas áreas.

Quadro de utilização e disponibilidade das áreas de display nos principais Controladores Programáveis BCM:

	Display #1	**Display #2**	**Display #3**
GP1500	Não disponível	Não disponível	Não disponível
GP3011	Livre p/usuário	Livre p/usuário	Livre p/usuário
GP3101	Livre p/usuário	Livre p/usuário	Usado p/funções do S.Operacional
GP3210	Livre p/usuário	Livre p/usuário	Usado p/funções do S.Operacional
GP3220	Livre p/usuário	Livre p/usuário	Usado p/funções do S.Operacional
BCM2085B/CPU8	Livre p/usuário	Livre p/usuário	Usado p/funções do S.Operacional
BCM2085B/CPU8A	Livre p/usuário	Livre p/usuário	Usado p/funções do S.Operacional
BCM2085B/CPU9	Livre p/usuário	Livre p/usuário	Usado p/funções do S.Operacional

2.6.3 - Variáveis de Atribuição Fixa

As variáveis de atribuição fixa possuem nomes e atribuições predeterminadas no contexto da linguagem. A sua finalidade é a de executar funções específicas, substituindo instruções e comandos com vantagens de legibilidade dos programas e facilidade de manejo.

O usuário pode incluir as variáveis desse grupo em testes lógicos e operações aritméticas, usando as instruções `SE' e `FACA'. Cada variável possui uma característica operacional própria, com faixa de valores e possibilidades de leitura/escrita definidas para cada caso.
No sentido de agrupar as variáveis de atribuição fixa de acordo com as suas funções, estas foram divididas em sub-grupos. Além disso, cada variável apresenta uma descrição própria do seu funcionamento.

Atenção: Um erro comum no desenvovimento de aplicações é a criação de variáveis pelo programador cujo nome seja o mesmo de uma variável de atribuição fixa do sistema. Se fizer isso, o funcionamento do controlador pode assumir uma situação inesperada, conforme a variável e a forma como essa for afetada.

Abaixo é apresentada uma lista alfabética das variáveis de atribuição fixa, atualizada de acordo com o PROCP V6:

ACELX,Y,Z	CFF1-11	EANER	IMAGE	NEEDD	SEC	TOR1-2
ANAD1-8	CFIC	EBACK	ISIER	NFIC	SHADR	TOUCH
ANANM	CFISI	EDEER	HOR	NPSDD	SIGPS	TVARR
ANO	CFMOT	EEEDD	HOTBK	NTAR	SONAL	UVARR
BANCO	CFSDD	EER1-2	LIBDS	NTMPD	STAGP	VERSO
BATOK	CIC	EMEM	LICLP	PRGPS	TCPAT	V01-16
BLCON	CMOK1-4	ENDR1-9	MATER	QNTOP	TECLA	WROFF
BLFUN	CNCON	ENERE	MES	RATT	TEM1-2	WW2
C3210	CPUAT	ENERZ	MIN	RCEN	TEMCH	WW3
CA1-10	CT1-10	ENGPS	MOTER	RDEC	TF1-TF4	WWW
CEN	CTECL	EPSDD	MULTH	RHOR	TFN0-1	
CFANA	CTH1-10	ETMPD	MVCC	RMIN	TGRAF	
CFCAN	DEC	EUR1-2	MVDD	RODIS	TIPER	
CFEAI	DIA	FUNG1	MVDH	RSEC	TIPOP	
CFEDD	DSEM	GRAV1-2	MVSS	RSSIX	TOC1-2	
CFEDE	E00-E128	J01-32	NBACK	SDCAR	TOF1-2	

2.6.3.1 - Temporização

As variáveis de temporização são incrementadas automaticamente pelo sistema operacional a cada unidade de tempo definida. Podem ser usada para leitura, escrita e teste na faixa 0 a 65535. Se a função de temporização exceder 65535, a variável é zerada e a contagem continua.

É importante observar que as variáveis são totalmente independentes, isto é, o contador de minutos, por exemplo, pode ser zerado ou presetado sem que ocorra nenhuma influência sobre o contador de horas.

HOR - Contador de horas. Incrementa uma unidade por hora.

MIN - Contador de minutos. Incrementa uma unidade por minuto.

SEC - Contador de segundos. Incrementa uma unidade por segundo.

DEC - Contador de décimos. Incrementa uma unidade por décimo de segundo.

CEN - Contador de centésimos. Incrementa a cada por centésimo de segundo.

ETMPD - Endereço do buffer dos temporizadores de décimos de segundo (válido somente para a CPU8). O usuário pode criar até 255 temporizadores, que serão incrementados a cada décimo de segundo. Na variável NTMPD deverá ser programado o número de temporizadores e na variável ETMPD o endereço do buffer dos temporizadores. Este buffer deverá ser definido no programa do usuário e deverá ser compatível com o tamanho configurado em NTMPD. Cada posição do buffer funcionará como um temporizador independente que poderá ser usado livremente pelo usuário. O valor inicial de NTMPD e ETMPD será igual a zero.

NTMPD - Número de temporizadores de décimos de segundo (válido somente para a CPU8). Veja a descrição para **ETMPD**.

2.6.3.2 - Relógio

As variáveis de relógio tornam as informações de data e hora real disponíveis para o programador. Essas variáveis só tem utilidade quando o sistema está configurado com uma placa ou com recursos de relógio de tempo real. Essas variáveis estão disponíveis para inclusão em operações aritméticas (leitura) e operações de teste. A escrita nas variáveis de relógio (acerto do relógio/calendário) deve seguir um roteiro especial, descrito no capítulo específico sobre cada modelo de controlador programável.

ANO Variável associada ao relógio de tempo real (ano atual).

MES Variável associada ao relógio de tempo real (mês atual).

DIA Variável associada ao relógio de tempo real (dia atual)

RHOR Variável associada ao relógio de tempo real (hora atual)

RMIN Variável associada ao relógio de tempo real (minuto atual)

RSEC Variável associada ao relógio de tempo real (segundo atual)

RDEC Variável associada ao relógio de tempo real (décimo de segundo atual)
Obs: esta variável só existe no BCM2085/CPU7

DSEM Variável associada ao relógio de tempo real (dia da semana). No BCM1086A varia de 0(domingo) à 6(sábado); No GP3000 e BCM2085B varia de 1(domingo) à 7(sábado)

RATT Variável de controle para acerto do relógio:
0 - O relógio sendo atualizado, se estiver habilitado
1 - Bloqueia a leitura do relógio, permitindo a atualização das variáveis ANO, MES, DIA, RHOR, RMIN, RSEC para acerto do relógio.
2 - Carrega o valor de ANO, MES, DIA, RHOR, RMIN, RSEC no relógio de tempo real e zera RDEC.
3 - Erro no relógio

Para fazer o acerto do relógio através das variáveis (recurso usado principalmente quando se deseja acertar o relógio remotamente, via canal de comunicação), siga a sequencia:

- Escreva o valor 1 na variável RATT (isso vai interromper a escrita dos valores do RTC nas variáveis)
- Escreva os valores desejados em ANO,MES,DIA,RHOR,RMIN,RSEC
- Escreva o valor 2 na variável RATT (isso vai atualizar o RTC e voltar ao funcionamento normal)

2.6.3.3 - Reconhecimento dos módulos de E/S

As variáveis desse grupo sinalizam o estado de reconhecimento e diagnóstico dos vários módulos de E/S do controlador BCM2085B:

CFANA Configuração dos módulos ANA reconhecidos. Cada bit da variável corresponde a um módulo. O bit 0 corresponde ao módulo 1, o bit 1 corresponde ao módulo 2 e assim por diante. Valor 1 significa que o módulo está presente e OK. Valor 0 significa que o módulo não foi reconhecido ou está em erro. (Não disponível atualmente)

CFEAI Configuração dos módulos EAI reconhecidos. Cada bit da variável corresponde a um módulo. O bit 0 corresponde ao módulo 1, o bit 1 corresponde ao módulo 2 e assim por diante. Valor 1 significa que o módulo está presente e OK. Valor 0 significa que o módulo não foi reconhecido ou está em erro.

CFEDD Configuração dos módulos EDD reconhecidos. Cada bit da variável corresponde a um módulo. O bit 0 corresponde ao módulo 1, o bit 1 corresponde ao módulo 2 e assim por diante. Valor 1 significa que o módulo está presente e OK. Valor 0 significa que o módulo não foi reconhecido ou está em erro.

CFEDE Configuração dos módulos EDE reconhecidos. Cada bit da variável corresponde a um módulo. O bit 0 corresponde ao módulo 1, o bit 1 corresponde ao módulo 2 e assim por diante. Valor 1 significa que o módulo está presente e OK. Valor 0 significa que o módulo não foi reconhecido ou está em erro.

CFISI Configuração dos módulos ISI reconhecidos. Cada bit da variável corresponde a um módulo. O bit 0 corresponde ao módulo 1, o bit 1 corresponde ao módulo 2 e assim por diante. Valor 1 significa que o módulo está presente e OK. Valor 0 significa que o módulo não foi reconhecido ou está em erro.

CFMOT Configuração dos módulos MOT reconhecidos. Cada bit da variável corresponde a um módulo. O bit 0 corresponde ao módulo 1, o bit 1 corresponde ao módulo 2 e assim por diante. Valor 1 significa que o módulo está presente e OK. Valor 0 significa que o módulo não foi reconhecido ou está em erro. (Não disponível atualmente).

CFSDD Configuração dos módulos SDD reconhecidos. Cada bit da variável corresponde a um módulo. O bit 0 corresponde ao módulo 1, o bit 1 corresponde ao módulo 2 e assim por diante. Valor 1 significa que o módulo está presente e OK. Valor 0 significa que o módulo não foi reconhecido ou está em erro.

CFF1 Indica o número total de módulos SD presentes e OK.

CFF2 Indica o número total de módulos ESA presentes e OK.

CFF3 Indica o número total de módulos CTR presentes e OK.

CFF4 Indica o número total de módulos PAN presentes e OK.

CFF5 Indica o número total de módulos EDD presentes e OK.

CFF6 Indica o número total de módulos SDD presentes e OK.

CFF7 Indica o número total de módulos EDE presentes e OK.

CFF8 Indica o número total de módulos ANA presentes e OK.

CFF9 Indica o número total de módulos ISI presentes e OK.

CFF10 Indica o número total de módulos EAI presentes e OK.

CFF11 Indica o número total de módulos MOT presentes e OK. (Não disponível atualmente)

EANER Apresenta o código de status de erro relativo aos módulos EAN ou EAI. A composição do código segue o seguinte esquema:

```
P P  E
|   |----- Código de erro:
|          1 - Instrução não executada porque a CPU não reconheceu o módulo
|          3 - Erro no sensor de junta fria
|          4 - Erro no conversor A/D ou falta de alimentação
|          5 - Erro na memória Ram
|          6 - Erro na memória Eprom
|          7 - O módulo não responde
|          8 - Módulo indisponível
|          9 - Módulo indisponível
|
|--------- Número da placa com o erro
```

EDEER Apresenta o código de status de erro relativo aos módulos EDE. A composição do código segue o seguinte esquema:

```
P P  E
|   |----- Código de erro:
|          1 - Erro na memória Eprom
|          2 - Erro na memória Ram
|          3 - Erro nas memórias Eprom e Ram
|          4 - Erro em alguma das entradas digitais
|          5 - Erro nas entradas e na Eprom
|          6 - Erro nas entradas e na Ram
|          7 - Erro nas entradas, na Eprom e na Ram
|          8 - Erro no acesso ao módulo
|          9 - Instrução não executada porque a EDE não foi reconhecida na inicialização ou
|              porque o horário está sendo sincronizado pelo GPS
|
|--------- Número da placa com o erro
```

ISIER Apresenta o código de status de erro relativo aos módulos ISI. A composição do código segue o seguinte esquema:

```
P P  E
|   |----- Código de erro:
|          1 a 7 - Erros internos no módulo ISI
|          9 - Instrução não executada porque a CPU não reconheceu o módulo
|
|--------- Número da placa com o erro.
```

MOTER Essa variável apresenta o código de status de erro relativo aos módulos BCM2085/MOT. A composição do código segue o seguinte esquema:

```
P   E
|    |----- Código de erro:
|          1 - Erro de execução em INSMOT
|          5 - Erro na memória RAM
|          6 - Erro na memória EPROM
|          7 - A placa não responde
|          8 - Placa indisponível (Status W)
|          9 - Placa indisponível (Status R)
|
|--------- Número da placa com o erro.
```

Os erros de código 5 a 9 são marcados na inicialização do CP, quando do autoteste dos módulos. Neste momento, o número da placa se refere a primeiro módulo no qual houve erro. O erro código 1 é marcado durante a execução de uma instrução "MOTOR", caso haja alguma anormalidade no interface.

2.6.3.4 - Controle de programa

As variáveis de controle de programa atendem operações internas diversas, relacionadas ao fluxo do programa do usuário:

BANCO Alguns controladores oferecem áreas de memória de variáveis organizadas em segmentos múltiplos denominados bancos de memória. Esta variável permite ao programa do usuário selecionar qual o banco de memória deve ser acessado para leitura ou escrita em determinado momento. Consulte a documentação específica do controlador programável para mais detalhes.

BLFUN Informa o número de blocos de função gravados na Flash do controlador e disponíveis para uso. Esta variável é atualizada na inicialização e após ser concluído a transferência dos blocos de função ou arquivos de imagem. Recurso válido para o GP3210, GP3220 e CPU9.

CIC Contador de ciclos de varredura do CP. Essa variável é incrementada continuamente de 0 a 65535, uma unidade a cada ciclo do controlador.

CPUAT - Indica se a CPU8 está ativa ou em stand-by. Usada em aplicações com redundância usando duas CPUs. Válido para CPU8 (com S.O.2008-2 ou superior).

E00 a E127 Estado de cada uma das 128 malhas disponíveis. Somente a CPU7-11 e CPU8 tem capacidade para 128 malhas; os demais produtos suportam 64 malhas. Este sub-grupo de variáveis permite ao programa controlar em uma máquina o que ocorre nas outras, estabelecendo vínculos entre os diversos ramos lógicos do sistema. Nesta função, estas podem ser testadas e participar de operações aritméticas, se necessário. Pode-se ainda forçar mudanças de estado imediatas em qualquer máquina, colocando valores nessa variáveis através da instrução `FACA'.
Essa operação equivale a mudança de estado normal, a menos do rearme dos flags aritmético e de impressão, os quais não são rearmados quando se executa uma mudança de estado por esse método. As variáveis desse sub-grupo admitem valores na faixa 0 a 254.

ENDR1 a ENDR9 - Endereços de referência para passagem de parâmetros para Blocos de Função usados no controlador BCM2085B/CPU8, CPU8A e CPU9. O significado de cada variável irá depender do Bloco de Função usado e estará descrito na sua documentação.

GRAV1 Controla a gravação e leitura de variáveis para a memória flash. Válido para o GP3101, GP3210 e GP3220. Sob comando dessa variável, o conteúdo da área de display 2 (variáveis do usuário) pode ser escrito ou lido em memória flash de acordo com a seguinte tabela:

Valor	Descrição
0	Aguardando comando; operação concluída
1	Leitura da flash para a Ram, com checagem de CRC
5	Leitura da flash para a Ram, sem checagem de CRC
10	Gravação de variáveis na flash
11	Confirmação da gravação de variáveis na flash
100	Indicação de erro de CRC na gravação da flash
101	Indicação de erro de gravação
102	Gravação não confirmada

GRAV2 Controla a gravação e leitura de variáveis para a memória flash. Válido para o GP3101, GP3210 e GP3220. Sob comando dessa variável, o conteúdo da área de display 3 (variáveis do usuário) pode ser escrito ou lido em memória flash de acordo com a mesma lógica de comando usada em **GRAV1**.

HOTBK Contador de inserções ou retiradas de módulos a quente. Este recurso é valido para o BCM2085B/CPU7-11, CPU8, CPU8A e CPU9.

IMAGE Informa o número de arquivos de imagem gravados na Flash do controlador. Esta variável é atualizada na inicialização e após ser concluída a transferência de arquivos de imagem ou blocos de função. Recurso válido para GP3210, GP3220 e CPU9.

J01 a J32 Acumulador de erros, usado para cálculos do termo integral nas rotinas de controle PID Para cada uma das 32 rotinas PID disponíveis, essa variável contém o valor atual do acumulador integral. Essa área de variáveis pode ser incluída em testes, bem como em operações de leitura e escrita. Admite valores na faixa de 0 a 65535.

LIBDS Esta variável controla o acesso ao display de serviço pela tecla N/S. Com LIBDS=0, o acesso ao display de serviço fica bloqueado. Com LIBDS=1, o acesso é liberado. Com LIBDS=2, o acesso ao display de serviço fica limitado às telas de E/S digitais (Recurso somente disponível para o GP3101). Quando o CP é ligado, esta variável é sempre inicializada em 1. O uso de LIBDS permite ao programador impedir que o usuário do sistema entre no display de serviço e possa causar danos ao processo.

LICLP Controle do fluxo de programa do usuário:

0 - Bloqueado
1 - Liberado
2 a 255 - Executa (n-2) ciclos de varredura
65534 - Desativa CPU, liberando barramento externo (soft-reset) (somente para CPU8)
65535 - Força reset via watchdog (hard-reset)(válido para GP1500, GP3101, GP3210, GP3220, CPU8 e CPU9)

MATER Esta variável (quando em 1) indica que existe uma instrução FACA ou SET tentando realizar uma escrita em área de variáveis de RAM interna do S.O., ou seja, fora da faixa de 1200H a FFFFH. A variável é inicializada em zero. Esta instrução está disponível no PROCP V301.0 e PROCP V3K podendo ser usada somente com o BCM2085/CPU7.

MVDD Informa a tensão Vdd da placa CPU do controlador (V *100). O valor nominal é 1200 (12V). Estão previstas para implementação futura as variáveis **MVCC, MVSS, MVDH**, medindo outras tensões da placa. Recurso válido para o GP3210 e GP3220.

MULTH Esta variável recebe a parte alta (bits 16 a 31) do resultado de uma operação de multiplicação realizada pela instrução FACA.

TIPER Tipo de erro detectado pela CPU, conforme a codificação:
0 - Não há erro
1 - Erro na Ram da CPU
32 - Erro de checksum da CPU
64 - Conflito de enderaçamento entre os módulos 2085B/ESA e /ANA.

TIPOP Tipo de produto - Mapa de bits descrevendo a configuração e recursos instalados em determinado produto. Recurso válido para o controlador GP1500. Ver detalhes da codificação dos bits no manual de Instalação e Características GP1500.

VERSO Esta variável indica a versão do Sistema Operacional do CLP. Este recurso é válido para o PROCP V5k, revisão 4.0 ou posterior.

2.6.3.5 - Contadores Rápidos

O sistema lê o conteúdo dos contadores do GP3000 ou da placa BCM2085/CTR e atualiza um grupo composto por 2 variáveis em cada contador (contagem alta e contagem baixa). Esse formato é necessário porque os contadores podem assumir valores de 0 até 1048575 (20 bits). Como todas as variáveis do sistema são de 16 bits, cada contador é dividido em duas variáveis. As variáveis definidas para contagem baixa variam na faixa 0 a 65535; as variáveis definidas para contagem alta variam na faixa 0 a 15. O sistema prevê até um máximo de 10 grupos de variáveis (5 placas CTR). No GP 3000, estão disponíveis até 4 contadores, todos de 16 bits, usando as variáveis CT1, CT2, CT3, CT4.

CT1 a CT10 Valor da parte baixa dos registros de contagem rápida nas placas CTR 1 a 5.

CTH1 a CTH10 Valor da parte alta (bits 16 a 19) dos contadores rápidos nas placas CTR 1 a 5.

CA1 a CA10 Resultado da operação dos comparadores analógicos das placas CTR 1 a 5. Essas variáveis assumem os valores 0 ou 1, conforme os níveis de tensão nas entradas analógicas. Se o valor da tensão da entrada positiva (X+ ou Y+) for maior que o valor da entrada negativa (X- ou Y-), a variável CAx assume o valor zero. Caso contrário, assume o valor 1.

2.6.3.6 - Interfaces com periféricos

Esse sub-grupo de variáveis de atribuição fixa tem a finalidade específica de proporcionar a ligação entre as operações lógicas do programa e alguns interfaces físicos de entrada/saída de uso menos comum, não abrangidos pelas variáveis de entrada/saída. Como nos outros tipos, a aplicabilidade das informações depende da existência ou não do suporte de hardware no modelo/versão de controlador programável utilizado:

ACELX,Y,Z Aceleração mecânica medida ao longo dos eixos X, Y e Z do controlador GP1500. Ver detalhes da codificação e unidade de medida no manual de Instalação e Características GP1500.

ANAD1 a ANAD8 Estas variáveis são usadas para acionar as saídas digitais das placas ANA nº 1 até a nº 8. No programa BCM, estas variáveis podem ser usadas como as demais, mas apenas os 4 bits menos significativos é que irão acionar as saídas digitais da respectiva placa. Por exemplo, se ANAD1 = 1, só a 1ª saida digital da 1ª placa será acionada. Se ANAD1 = 3, a 1ª e a 2ª saidas digitais serão acionadas. Se ANAD1 = 15, todas as 4 saídas serão acionadas.

ANANM Esta variável contém o número de entradas analógicas que são lidas de todas as placas ANA instaladas. Seu valor pode variar de 1 a 8. Se ANANM for igual a zero, será realizado o autoteste do conversor A/D de todas as placas ANA instaladas e o resultado será colocado na 1ª variável de entrada analógica de cada placa. Se o teste estiver OK, o valor lido deve estar entre 3140 e 3150.

BATOK No GP3210 e GP3220, informa a tensão da bateria (em Volts*100). Nos demais controladores, assume o valor 1 quando a bateria esta OK ou zero para bateria gasta.

BLCON Variável de controle para o backlight do display. Permite ao programa do usuário controlar o acendimento ou não do backlight. Setando 1 na variável, acende; setando zero, apaga o backlight. Setando outro valor entre 2 e 255, o display fica aceso por (n/10) segundos após uma tecla ter sido pressionada.

C3210 Variável para configuração das entradas para termopar do GP3210. (Esta variável é válida somente para esse modelo de CLP). Com o valor zero, faz a leitura padrão das entradas analógicas. Com o valor 1, inclui as entradas para termopar no conjunto de canais lidos. Ver o manual do GP3210 para mais detalhes.

CFCAN Usada para configurar o módulo Ethernet TCP-IP (BCM560) quando aplicado no BCM2085B/CPU8. Ver o manual CPU8 para os detalhes de utilização.

CNCON Variável de controle para o contraste do display. Disponível para alguns modelos de controladores. Permite ao programa do usuário regular o contraste do display LCD. Consulte o manual específico do controlador para mais detalhes.

CTECL Configura a forma de reação das variáveis *TF1* a *TF4, TECLA, TOUCH* e botões do display gráfico, em função do toque no display do GP3210 ou GP3220. Ver a descrição de *TF1* a *TF4,* das variáveis *TECLA* e *TOUCH* para as opções disponíveis. Seu valor default é zero.

EBACK - Controla a função de leitura do readback das saídas digitais das placas SDD. Na variável NBACK deve ser configurado o número de placas SDD com readback (a partir da primeira) que o usuário precisa ler e na variável EBACK deve ser programado o endereço do buffer de carga do readback das saídas. Este buffer deverá ser definido no programa do usuário e seu tamanho deverá ser compatível com o número de placas ([NBACK*24-1]/16+1 posições). Em cada posição do buffer será carregado o estado de readback de 16 saídas digitais, sendo que a informação de cada bit pode ser:

0 -> readback da saida digital Ok
1 -> readback da saída digital c/ falha ou SDD não reconhecida

EEEDD Endereço da matriz para carga do estado dos pontos de entradas digitais. Cada posição desta matriz informa o estado de uma entrada digital, a partir da primeira. Válido para GP3101, GP3210, GP3220, CPU8 e CPU9.

Valor = 0 -> indica entrada desligada (OFF)
Valor = 1 -> indica entrada ligada (ON)

Esta variável é usada em combinação com NEEDD (ver descrição abaixo).

EMEM Variável de entrada para uso geral, disponível no módulo BCM2085/MEM. O Sistema Operacional carrega automaticamente o valor binário da porta na variável, para ser usada em testes e operações aritméticas. Assume valores na faixa 0 a 255.

ENERE Fator de escala aplicado ao valor de energia ativa e reativa medida pelo controlador GP3210. Ver detalhes no manual de Instalação e Características GP3210.

ENERZ Força o zeramento dos contadores de energia do controlador GP3210. Ver detalhes no manual de Instalação e Características GP3210.

ENGPS Variável para configurar o endereço inicial da matriz que irá receber as leituras do GPS no programa do usuário do controlador GP1500. Ver detalhes no manual de Instalação e Características GP1500.

EPSDD Endereço do buffer associado às saídas digitais. Cada posição deste buffer força o estado de uma saída digital, a partir da primeira. Válido para GP3101 (com S.O.2008-1). Esta variável é usada em combinação com NPSDD (ver descrição abaixo).

Valor = 0 -> desliga a saída digital (OFF)
Valor >= 1 -> liga a saída digital (ON)

Atenção: Quando este recurso é usado para controlar as saídas digitais, as instruções LIGA e DESL não devem ser usadas.

NBACK Controla a função de leitura do readback das saídas digitais das placas SDD. * Veja a descrição acima para EBACK *

NEEDD O Sistema Operacional oferece o recurso de cópia do estado dos pontos digitais numa matriz do programa do usuário. Para usar esse serviço no GP3101, GP3210 ou GP3220 (S.Operacional 2008-1 ou superior), o programa do usuário deve carregar nesta variável o número de pontos de entrada digital, a partir da primeira, que deseja que sejam carregados na matriz associada.

Para usar esse serviço na CPU8 (S.Operacional 2008-1 ou superior), o programa do usuário deve carregar nesta variável o número de módulos EDD, a partir do primeiro, que deseja que sejam carregados na matriz associada.

Esta variável é usada em combinação com a variável EEEDD (ver descrição acima).

NPSDD Número de pontos de saída digital, a partir da primeira, que tem buffer (ou matriz) associado. Válido para GP3101 (com S.O.2008-1). Esta variável é usada em combinação com EPSDD (ver descrição acima).

PRGPS Variável utilizada para disparar a sincronização do relógio da BCM2085/CPU7, CPU8, CPU8A, CPU9 e das placas BCM2085/EDE através do pulso PPS de um GPS. A descrição dos valores que podem ser atribuídos e seus significados se encontra no manual *BCM2085B - Instalação e Características Técnicas*, capítulo CPU8 ou CPU9.

RODIS Variável de controle para a rolagem das mensagens no display. Disponível para alguns modelos de controladores. Consulte o manual específico do controlador para mais detalhes.

SDCAR Variável de controle para gravação de dados no cartão de memória disponível nos controladores GP3210 e GP3220. Consulte o manual específico do controlador para mais detalhes.

SIGPS Variável utilizada para indicar o estado da sincronização do relógio da BCM2085/CPU7 pelo GPS.

SONAL Variável de controle para o sonalarme (sinalizador sonoro). Disponível para alguns modelos de controladores. Permite ao programa do usuário controlar o acionamento. Setando 1 na variável, o sonalarme toca; setando zero, não toca.

TF1 a TF4 No GP3011, GP3101 e BCM 2085B/DPA, assume o valor 1 quando a respectiva tecla de função é pressionada, senão mantém o valor zero.
No GP3210 e GP3220, depende do valor programado em CTECL:
- Se zero (default): A variável só assume 1 na primeira varredura após o toque, depois volta para zero.
- Se 1, funciona como nos demais produtos (GP3011, GP3101, etc.)

TECLA Assume um valor direto em função de qualquer tecla que esteja pressionada, conforme a seguinte codificação:

Tecla pressionada	Valor em TECLA		Tecla pressionada	Valor em TECLA
0	0		9	9
1	1		N/S	10
2	2		EV - Entra valor	11
3	3		Avança cursor	12
4	4		Recua cursor	13
5	5		Avança valor	14
6	6		Recua valor	15
7	7			
8	8		Nenhuma tecla pressionada	128

Observações:

- No GP3011, o valor em *TECLA* permanece enquanto a tecla estiver pressionada, voltando para 128 quando a tecla é liberada (válido para S.Operacional 2002-3 ou superior)

- No GP3101, o valor em *TECLA* permanece só por um ciclo de varredura do CLP, voltando para 128 no ciclo seguinte, independentemente da tecla ser liberada ou não (válido para S.Operacional 2007-1 ou superior)

- No GP3210 e GP3220, o funcionamento depende da variável *CTECL*. Com *CTECL*=0, o valor da tecla pressionada fica armazenado na variável até ser pressionada outra tecla. Esta funcionalidade somente está disponível no modo normal alfanumérico. No modo de serviço a variável fica congelada em 128 e no modo gráfico o valor da variável não é alterado.
Com *CTECL*=1, o valor em *TECLA* permanece enquanto a tecla estiver pressionada, voltando para 128 quando a tecla é liberada (válido para S.Operacional 2014-1 ou superior)

- No BCM1086A não há suporte para a variável *TECLA*.

TEMCH Temperatura interna do chip (processador), em ºC. Recurso válido para o controlador GP1500.

TGRAF Endereço inicial da matriz para configuração dos objetos gráficos no controlador GP3220. Este valor deve ser presetado pelo usuário, o que irá ativar o tratamento dos objetos gráficos.

TOUCH Posição tocada pelo usuário no display touchscreen do controlador GP3220. O S.Operacional coloca um valor de 1 a 48 nesta variável, considerando que o display é mapeado em 6 linhas e 8 colunas, contadas de baixo para cima e da esquerda para a direita. Se *CTECL*=0, o valor lançado em *TOUCH* permanece; se *CTECL*=1, o valor de *TOUCH* volta para zero quando o toque termina.

V01 a V16 Variáveis associadas ao painel de leds (BCM2085/PAN ou BCM1086/EX1). Cada variável aqui irá controlar uma coluna de 16 leds. No BCM2085, cada uma das duas placas admissíveis possui 8 colunas, totalizando 16 variáveis. O valor binário positivo de cada variável, lançado pelo programa através de uma operação aritmética, é transferido para o padrão de acendimento dos leds. Assim, o valor 0 na variável apaga toda a coluna; o valor 1 acende o primeiro led, e assim por diante até que 65535 acende todos os leds da coluna.

WWW Essa variável contém o canal do display #1 presente no momento. Quando o usuário muda o canal pelo teclado do BCM 2085B/DPA, a variável transmite essa informação para o programa. No sentido inverso, um valor lançado nessa variável usando a instrução `FACA' irá fazer com que o display #1 vá para o canal especificado. Aceita valores na faixa 0 a 99.

WW2 Essa variável contém o canal do display #2 presente no momento. Quando o usuário muda o canal pelo teclado do BCM 2085B/DPA, a variável transmite essa informação para o programa. No sentido inverso, um valor lançado nessa variável usando a instrução `FACA' irá fazer com que o display #2 vá para o canal especificado. Aceita valores na faixa 0 a 99.
Essa variável, em combinação com SDCAR, também é usada para controlar as operações com o cartão de memória. Ver detalhes no manual do CLP específico.

WW3 Essa variável contém o canal do display #3 presente no momento. Quando o usuário muda o canal pelo teclado do BCM 2085B/DPA, a variável transmite essa informação para o programa. No sentido inverso, um valor lançado nessa variável usando a instrução `FACA' irá fazer com que o display #3 vá para o canal especificado. Aceita valores na faixa 0 a 99.

2.6.3.7 - Canais de comunicação

Esse sub-grupo de variáveis fornece informações dinâmicas sobre os processos de comunicação.

CFIC Contador do número de vezes que a ficha foi passada para o CP. Esta variável é usada para avaliar o desempenho da rede, alarmes, etc. Pode ser zerada ou presetada pelo programa do usuário, sendo incrementada cada vez que a ficha é recebida pelo CP. Assume valores na faixa 0 a 65535.

CMOK1-4 Contador de operações de comunicação (mensagens Ok recebidas) em cada um dos 4 canais de comunicação do controlador BCM2085B/CPU8A.

EER1/EER2 Endereço do equipamento na rede BCM2. Disponíveis para os canais seriais 1 e 2.

EUR1/EUR2 Endereço do último equipamento na rede BCM2. Disponíveis para os canais seriais 1 e 2.

FUNG1 Dispara a configuração do modem GPRS. Recurso válido para os controladores GP1500, GP3101, GP3210 e GP3220. Ver detalhes da operação nos manuais específicos desses produtos. Ver também a descrição para STAGP.

NFIC Contador do número de vezes que a ficha foi criada pelo CP. Em conjunto com CFIC, esta variável é usada para avaliar o desempenho da rede, alarmes, etc. Pode ser zerada ou presetada pelo programa do usuário, sendo incrementada cada vez que o CP cria uma ficha na rede. Assume valores na faixa 0 a 65535.

NTAR Número de tarefas de comunicação agendadas e ainda não concluídas. O sistema possui capacidade para suportar até quatro operações pendentes, agendadas simultaneamente. Assume valores na faixa 0 a 4.

RSSIX Status da conexão Zigbee e nível de RF (em -dBm) na conexão de rádio do controlador GP1500. Ver detalhes da codificação do status no manual de Instalação e Características GP1500.

SHADR No protocolo Zigbee, este é o endereço local atribuido pela rede (Short Address) para o controlador GP1500. Ver detalhes no manual de Instalação e Características GP1500.

STAGP Status da comunicação com o modem GPRS. Recurso válido para os controladores GP1500, GP3101, GP3210 e GP3220. Ver detalhes da operação nos manuais específicos desses produtos. Ver também a descrição para FUNG1.

QNTOP Quantidade de operações ModbusTCP através do modem GPRS. Recurso válido para os controladores GP1500, GP3101, GP3210 e GP3220. Ver detalhes da operação nos manuais específicos desses produtos. Ver também a descrição para FUNG1.

TCPAT Disponível para o GP3101, GP3210 e GP3220, controla o disparo da conexão TCP ativa. Consulte o manual do controlador para o detalhamento da forma de operação.

TEM1/TEM2 Tempo de espera gerado antes do envio de cada mensagem na rede BCM2. Em centésimos de segundo.

TOC1/TOC2 - Tempo de time-out entre caracteres para descartar uma mensagem em curso na rede BCM 2para considerar que o destino não respondeu a uma solicitação na rede BCM 2. Em centésimos de segundo.

TOR1/TOR2 Tempo de time-out para considerar que o destino não respondeu a uma solicitação na rede BCM 2. Em centésimos de segundo.

TOF1/TOF2 Tempo de time-out para recriar a ficha na rede BCM2. Em centésimos de segundo.

WROFF Configura o endereço local para acesso ao primeiro registro da comunicação Modbus slave. É válida para os controladores GP3210 e GP3220. Consulte o manual específico do controlador para mais detalhes.

2.6.3.8 - Medição de Tempo de Varredura

TVARR - A medição do tempo de varredura do Sistema operacional do controlador programável está disponível nessa variável. O valor desta variável é igual ao maior tempo de varredura ocorrido (em centésimos de segundo).

UVARR - O tempo de varredura do Sistema operacional do controlador programável, atualizado a cada ciclo está disponível nessa variável. É expresso em ms/5. (Válido para o GP3101, GP3210, GP3220, CPU8 e CPU9)

2.7 - REGRAS ESPECIAIS DE PROCESSAMENTO ARITMÉTICO

Como foi visto nos capítulos anteriores, as variáveis do sistema operacional podem assumir valores inteiros, positivos e dentro da faixa 0 a 65535 (A menos das limitações nas variáveis de atribuição fixa). Quando o resultado de uma operação extrapola esses limites, as seguintes regras são aplicáveis:

Operação resulta em valor negativo

==> A variável de saída assume o valor algébrico do resultado mais 65536.

EXEMPLO:
FACA ARE=1
FACA ASA=ARE-3 ;=> RESULTADO: ASA=(-2)+65536=**65534**

O resultado excede a faixa permitida

==> A variável de saída assume o resultado algébrico menos 65536.

EXEMPLO:
FACA ARE=65000
FACA ASA=ARE+600 ;=>RESULTADO: ASA=65600-65536=**64**

A operação prevista produz um resultado fracionário

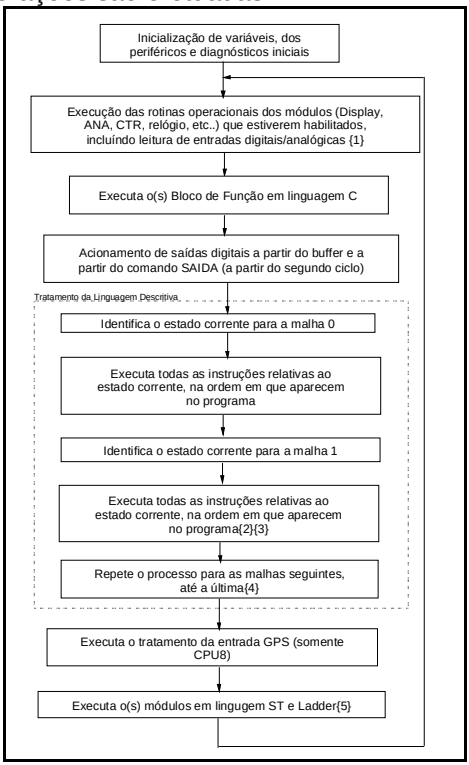
==> A variável de saída assume a parte inteira do resultado, desprezando a parte fracionária.

EXEMPLO:
FACA ARE=20
FACA ASA=ARE/3 =======> RESULTADO: ASA=**6**

2.8 - COMO O PROGRAMA É EXECUTADO / CICLO DE VARREDURA

A execução do programa do usuário no Controlador Programável BCM está relacionada diretamente ao conceito de Ciclo de Varredura. Em cada Ciclo de Varredura, o sistema operacional acompanha o estado dos dispositivos de entrada e saída, executa rotinas operacionais específicas e atende as operações solicitadas pelo programa. Ao final de um Ciclo, o microprocessador envia um pulso de controle ao circuito de Watchdog, voltando ao início e repetindo-o indefinidamente.

A composição do ciclo de varredura apresentada é válida para os Controladores Programáveis BCM1086A, BCM1088, BCM2085B e GP3000. O diagrama a seguir apresenta as tarefas típicas de um Ciclo de Varredura, mostrando a ordem na qual as operações são efetuadas:



Observações:

{1} - O tratamento de cada tipo de periférico tem características próprias relacionadas a velocidade e conveniência do processo:

DISPLAY BCM700: O sistema atende quatro unidades de display/teclado. Em cada passagem pela rotina operacional, uma unidade é atendida. Portanto, a renovação de valores e o reconhecimento do teclado por unidade é feita uma vez a cada quatro Ciclos de Varredura.

DISPLAY DPA: Em cada ciclo de varredura o display alfanumérico é atualizado e o teclado é lido.

ESA: A cada passagem pela rotina, uma entrada e uma saída analógica são processadas em cada placa. (O número de placas usado é definido no programa). Assim, a leitura de uma entrada analógica específica é feita sempre uma vez a cada oito Ciclos de Varredura.

IFT: A cada passagem pela rotina, um módulo é atendido. O atendimento consiste em verificar o final de conversão de um canal, recolher o valor e partir a conversão do canal seguinte. Como o tempo de aquisição por canal é longo (aprox. 300 ms), o método de varredura não influi no processo.

BCD: A cada passagem pela rotina, uma chave BCD é processada. Como é possível instalar até 16 grupos de duas chaves no controlador, a renovação da leitura de cada grupo é feita a cada 32 Ciclos de Varredura.

CTR: A forma de processamento depende do comando de configuração usado. Com "CTR=x", uma placa é atendida a cada passagem, exigindo 'x` Ciclos para renovação do valor de contagem. Com "CTR=x", todas as placas são atendidas em cada passagem, fazendo com que todos os valores sejam renovados no tempo de um Ciclo de Varredura.

PAN : A cada passagem, uma variável 'Vxx` é atualizada, correspondendo a 16 leds. Conforme o comando de configuração, "PAN=1" ou "PAN=2", a renovação completa do painel de leds é feita a cada 8 ou 16 Ciclos de Varredura.

{2} - Devido aos flags específicos usados pelas instruções "FACA" e "IMPRIME", essas são executadas apenas uma vez por entrada no estado corrente, permanecendo inibidas nos Ciclos de Varredura seguintes.

{3} - A instrução "SE" com transição válida, provoca a mudança do estado corrente. Essa mudança, no entanto, só afetará a execução das instruções no Ciclo de Varredura seguinte. As instruções contidas nas linhas subsequentes a linha "SE" que provocou a transição serão executadas. Essa característica será melhor detalhada no item 2.9 - Prioridades nas operações.

{4} - O número "N" de malhas que serão atendidas corresponde ao número de malhas citadas no programa do usuário.

{5} - Somente os controladores GP3101, GP3210, GP3220, CPU8 e CPU9 apresentam esse recurso.

2.9 - PRIORIDADE NAS OPERAÇÕES

Devido ao fato que a linguagem BCM está apoiada nos conceitos básicos de ESTADO e TRANSIÇÃO, não sendo similar as linguagens tradicionais de execução seqüencial, a análise de prioridade nas operações possui características próprias na linguagem BCM, conforme será apresentado.
Nos Controladores Programáveis, a seqüência de operações é determinada basicamente pelo fluxo dos ESTADOS. As condições de transição definem qual será o estado seguinte, cujas operações serão executadas a seguir, independentemente de seu posicionamento no programa fonte. Desta forma, a numeração dos estados e a distribuição destes no programa não tem qualquer influência na seqüência de processamento. Por exemplo, as sequências:

```
ESTADO 7:
LIGA 2,3
SE INI= 1 ENTAO 8
SE ATRASO = 20 ENTAO 10

ESTADO 8:
DESL 1
VA PARA 0

ESTADO 10:
LIGA 4,5,6
VA PARA 0
```

```
ESTADO 8:
DESL 1
VA PARA 0

ESTADO 10:
LIGA 4,5,6
VA PARA 0

ESTADO 7:
LIGA 2,3
SE INI=1 ENTAO 8
SE ATRASO=20 ENTAO 10
```

São executadas exatamente da mesma maneira. Do ESTADO corrente 7, pode-se passar ao ESTADO 8 ou 10 conforme as condições. Nesses ESTADOS serão efetuadas as operações especificadas e o programa seguirá conforme as transições.

Quando são consideradas as operações dentro do estado, o comportamento de programa pode ser dependente da ordem em que as linhas são editadas. Os casos típicos são descritos a seguir:

CASO A: Instrução `SE' concorrentes

Exemplo de referência :

```
100 ESTADO 5:
110 SE INI=1 ENTAO 2
120 SE ROT=1 ENTAO 3
```

No caso das condições "INI=1" e "ROT=1" estarem válidas , quando a máquina entra no estado 5, o sistema analisa a linha 110 e muda para o ESTADO 2. No entanto o sistema continua com a linha 120. Como a condição também será válida, o ESTADO 3 será ativado.
Por isso pode-se definir a seguinte regra de prioridade para o caso de instruções "SE" concorrentes:

- No caso de duas ou mais linhas "SE" com condições de transição simultaneamente válidas, o próximo ESTADO corrente será aquele indicado pela última linha na edição.

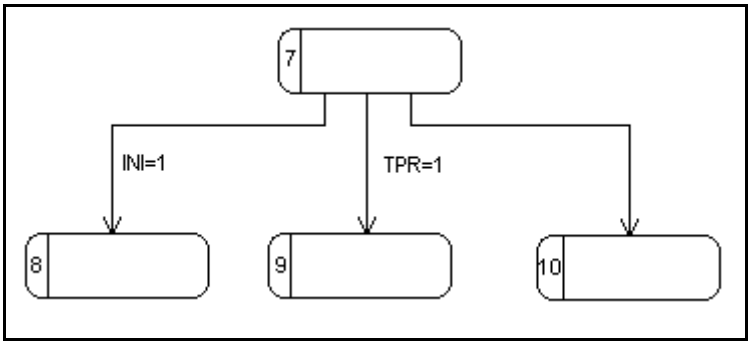
CASO B: Instrução 'SE' concorrente com 'VA PARA'

Exemplo de referência

```
100 ESTADO 5:
110 SE INI=1 ENTAO 2
120 VA PARA 7
```

Nesse caso, ao entrar no ESTADO 5, o sistema testa a condição da linha 110. Se for válida, a máquina irá para o estado 2 e a operação `VA PARA' da linha 120 será inibida por um flag específico. Se `INI' for diferente de 1, a condição 110 não será satisfeita e a máquina irá para o ESTADO 7 imediatamente.
Essa regra permite a implantação eficiente da condição "caso contrário": Um grupo de linhas `SE' executa as transições caso as condições específicas estejam satisfeitas; no caso de não ocorrer nenhuma transição, a instrução `VA PARA' envia a máquina para um estado alternativo.

Exemplo:



Solução 1:

ESTADO 7:
SE INI = 1 ENTAO 8
SE TPR = 1 ENTAO 9
SE INI = 0 OU TPR=0 ENTAO 10

Solução 2:

ESTADO 7:
SE INI= 1 ENTAO 8
SE TPR= 1 ENTAO 9
VA PARA 10

As duas soluções são funcionalmente idênticas em qualquer programa. A solução `2' no entanto, aproveitando a propriedade "caso contrário" da instrução `VA PARA' implementa a lógica com mais clareza, menor ocupação de memória e mais velocidade.

No exemplo a seguir, a lógica exige que, conforme a combinação de duas variáveis digitais, a máquina vá para um entre três estados:

Solução 1:

```
ESTADO 7:
SE INI=0 E TPR=0 ENTAO 8
SE INI=0 E TPR=1 ENTAO 9
SE INI=1 E TPR=0 ENTAO 10
SE INI=1 E TPR=1 ENTAO 10
```

Solução 2:

```
ESTADO 7:
SE INI=0 ENTAO 9
SE INI=0 E TPR=0 ENTAO 8
VA PARA 10
```

Combinando as regras vistas nos casos `a' e `b', usando uma eficiente análise de propriedades, a solução 2 permite diminuir o número de testes a serem feitos, melhorando o desempenho em ocupação de memória e velocidade.

Caso C: Prioridade na instrução 'FACA'

Exemplo de referência:

```
100 ESTADO 5:
110 FACA* ABB = 3 E ASD = 7 E TOTAL= ABB + ASD
120 FACA* TXD = 4 E TXD= 5
130 VA PARA 6
```

No caso de várias linhas com operações aritméticas, as operações serão executadas rigorosamente na ordem em que forem editadas. Da mesma forma, várias operações na mesma linha são completadas sempre na ordem de edição.

Na linha 110 do exemplo, a variável 'TOTAL' assume o valor 10, reconhecendo as operações anteriormente realizadas sobre `ABB' e `ASD'.
Na linha 120, provavelmente devido a um erro de edição, dois valores diferentes foram atribuídos a uma mesma variável. Nesse caso, a variável `TXD' assumirá o valor 5 (última operação).

Caso D: Atuação das instruções ***LIGA*** e ***DESL***

Exemplo de referência:

```
100 MAQUINA 0:
110 ESTADO 1:
120 LIGA 3 DESL 5
130 SE INI=1 ENTAO 2

140 ESTADO 2
150 LIGA 4
.   .
.   .
.   .

300 MAQUINA 1:

310 ESTADO 0:
320 DESL 4
```

Ao encontrar uma instrução ***LIGA/DESL*** no estado corrente, o sistema aciona ou desaciona imediatamente todas as saídas digitais envolvidas, independentemente da ordem de edição ou do número de linhas. A atuação sobre todas as saídas é quase simultânea (diferenças de microsegundos); esse é caso das saídas 3 e 5 na linha 120.

Na máquina 0, ao se cumprir a condição `INI=1', o sistema passa ao estado 2, no qual a saída digital 4 deverá ser acionada (linha 150). A máquina 1, no entanto, está mandando desligar a saída 4 ao mesmo tempo.

Configura-se nesse caso um conflito de saídas que deve ser sempre evitado no desenvolvimento da lógica. A saída ou saídas em conflito assumem uma situação imprevisível, podendo ficar ligadas, desligadas ou oscilar aleatoriamente, ocasionando o funcionamento deficiente do equipamento.