

# **CONTROLADORES PROGRAMÁVEIS**

**Curso de Automação Industrial**

**DEXTER Ind. e Com. de Equip. Eletrônicos Ltda.**

**Claudio Richter**

**2001**

## Controladores Programáveis

### Introdução

Os controladores programáveis (CP) ou controladores lógico-programáveis (CLP ou PLC, em inglês) surgiram para substituir painéis de controle a relé, na década de 60. A grande vantagem dos controladores programáveis era a possibilidade de reprogramação. Já os painéis de controle a relés necessitavam modificações na fiação, o que muitas vezes era inviável, tornando-se mais barato simplesmente substituir todo painel por um novo. Portanto, os CLPs permitiram transferir as modificações de hardware em modificações no software.

Existe um paralelo do que aconteceu com os painéis de controle a relés x controladores programáveis acontecendo atualmente na indústria eletrônica. Até algum tempo atrás, eram muito utilizados os CIs de lógica TTL (ou seus equivalentes CMOS), como portas NAND, OR, FLIP-FLOPs, etc. Eles começaram a ser substituídos por lógica programável (PAL – Programmable Array Logic) e, recentemente, por CPLD (Complex Programmable Logic Device) e FPGAs (Field Programmable Gate Array). As FPGAs permitem integrar funções extremamente complexas, como CPUs inteiras.

Com o sucesso de uso de CLPs na indústria, a demanda por novas funções e maior capacidade aumentou consideravelmente. Os equipamentos cresceram em poder de processamento, número de entradas e saídas (I/O), e novas funções. Entretanto, estes controladores ainda usavam lógica discreta e só eram utilizados na indústria, pois seus custos tornavam inviáveis outras aplicações (automação predial, por exemplo).

O advento do microprocessador permitiu uma diminuição nos custos e tamanho dos CLPs, com aumento de poder de processamento e confiabilidade. Surgiram as redes locais para comunicação entre CLPs e entre CLPs e computadores.

Atualmente existe uma forte tendência a utilização de pequenos controladores programáveis, controlando processos locais e comunicando-se com outros controladores e com sistemas supervisórios. Assim, é possível descentralizar o controle industrial, evitando que uma pane interrompa toda a planta. Muitas máquinas já possuem pequenos CLPs para controlá-las. Além disso, diversos sensores na indústria já utilizam microprocessadores junto ao sensor, para conformar o sinal de saída (que ainda pode ser 4 a 20mA ou comunicação serial 485, por exemplo). Com a diminuição de custos dos CLPs, estes passaram a ser utilizados em outros campos, como na automação predial (controle de iluminação, alarme, ambiência – ventilação, temperatura e umidade, etc). No Brasil ainda é pequeno o mercado de automação predial, mas provavelmente será um campo promissor em breve (ainda mais com o risco de cortes no fornecimento de energia elétrica). A automação residencial também desponta como uma aplicação para pequenos CLPs. Neste caso o maior problema, além dos custos, é a fiação necessária, pois o projeto civil normalmente não prevê eletrodutos para isso. Então, a comunicação entre diversos sensores e CLPs deve ser implementada via rede 485, rádio ou rede elétrica. As aplicações residenciais e prediais são vastas – alarme contra intrusos, controle de painéis solares para aquecimento de água, controle de iluminação, acionamento remoto de equipamentos, economia de energia elétrica...).

### Conceitos Básicos

#### Ponto de Entrada

Considera-se cada sinal recebido pelo CLP a partir de dispositivos ou componentes externos (sensores) como um ponto de entrada. Os pontos de entrada podem ser digitais ou analógicos. Os pontos de entrada digitais, obviamente, reconhecem apenas dois estados: ligado ou desligado. Já os pontos de entrada analógicos reconhecem mais de dois estados – normalmente um número múltiplo de dois (4, 8, 16, 32, 64, 128, 256, ....). O número de estados depende do número de bits usado pelo conversor A/D da entrada. Assim, um conversor A/D de 12 bits permite 1024 estados de entrada ( $2^{10}$ ).

Como exemplo de entradas digitais, pode-se citar sensores fim-de-curso (microchaves ou sensores indutivos), botoeiras, contatos secos (relés), etc.

Já entradas analógicas podem estar ligadas a termopares, sensores resistivos de posição, sinais 4 a 20mA ou 0 a 10V, tensão, corrente, etc.

### **Ponto de Saída**

Cada sinal produzido pelo CLP para acionar dispositivos ou componentes do sistema de controle (atuadores) constitui um ponto de saída. Novamente, podemos separar em saídas digitais ou analógicas. As saídas digitais possuem apenas dois estados, enquanto saídas analógicas possuem mais de dois estados (normalmente, o número de estados é múltiplo de dois – 4, 8, 16, 32, 64,...). O número de estados depende do número de bits usado pelo conversor D/A da saída. Assim, um conversor D/A de 8 bits permite 256 estados de saída ( $2^8$ ).

Pontos de saída digitais podem ser implementados por relés, transistores, ou ainda por SCRs e TRIACs. São usados para acionar lâmpadas, motores, solenóides, válvulas, etc.

Já pontos de saída analógicos fornecem correntes de 4 a 20mA, ou tensões de 0 a 10V. São usados para atuar válvulas proporcionais, controlar velocidade de motores (via Inversor de Frequência), etc.

Nota: Embora normalmente SCRs e TRIACs sejam usados em saídas digitais (ligado ou desligado), é possível usar estes dispositivos como uma saída analógica (com mais de 2 estados), controlando a fase de disparo do dispositivo em relação a rede elétrica. Este é o princípio de funcionamento dos controles de iluminação residencial (dimmers).

### **Conexão de Sensores a Pontos de Entrada**

A conexão de sensores e sinais externos no controlador programável deve ser feita com certo cuidado, em especial no que tange a interferência elétrica induzida por cabos de força ou acionamento. Como os sinais de entrada, normalmente, têm níveis de tensão e corrente pequenos (mV, no caso de termopares), eles se tornam susceptíveis a interferências de campos elétricos e magnéticos a sua volta, ou ainda a induções provenientes de telefones celulares, rádio transmissores, etc. Assim, cabos de entradas analógicas devem ter malha de blindagem, e os cabos de entradas (tanto analógicas quanto digitais) devem ser conduzidos dos sensores ao CLP via eletroduto ou calha específica, de metal e aterrada. Não deve-se misturar aos cabos de entrada cabos de acionamento e, muito menos, cabos de força. No caso de cruzamento entre cabos de entrada e cabos de força ou acionamento, fazer o cruzamento a 90°, de forma a minimizar a possibilidade de interferências. Deve-se evitar colocar cabos de entrada e cabos de força “correndo” em paralelo em um eletroduto ou calha, pois o acoplamento indutivo e capacitivo entre eles será maximizado.

As entradas analógicas a corrente (4 a 20mA) costumam ser mais imunes a ruídos elétricos do que entradas a tensão (0 a 10V), pois apresentam uma impedância menor. As entradas digitais normalmente são dimensionadas para a tensão de alimentação do controlador (12 ou 24 Vdc), e não devem ser ligadas diretamente a rede elétrica, a não ser que o manual do equipamento indique que isso é permitido.

### **Conexão de Atuadores a Pontos de Saída**

As saídas analógicas (4 a 20mA, 0 a 10V) são pontos de saída de baixa potência e, por isso, devem ser isoladas de cabos de força ou acionamento. Podem ser incluídas no eletroduto ou calha com os cabos de entrada ao CLP.

Já as saídas digitais, que acionam lâmpadas, solenóides, contactoras, etc., devem ser isoladas das entradas do CLP, pelos motivos expostos no item anterior. No caso de atuação de cargas indutivas, há de se considerar ainda a supressão da força contra-eletromotriz gerada na bobina do atuador, ao desligá-lo. Devido a importância deste fenômeno, vamos revisá-lo rapidamente.

Digamos que tenhamos o circuito a seguir, com a chave fechada durante um longo período. Neste caso a corrente já se estabilizou, já que a bobina ideal não oferece resistência a uma corrente constante. Revisando, a bobina oferece resistência a variação de corrente, pois a tensão em seus terminais é dada por:

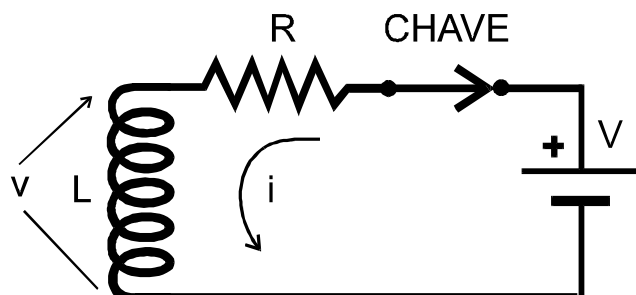
$$v = L \frac{\Delta i}{\Delta t}$$

Sendo:  $v$  = tensão nos terminais da bobina (V)

$L$  = indutância da bobina (Hy)

$i$  = corrente na bobina (A)

$t$  = tempo (s)



Ou seja, com a chave fechada a um longo período a corrente  $i = V/R$ , e a tensão sobre a bobina é nula (já que  $\Delta i = 0$ ). Mas ao abrimos a chave a corrente tende a zero instantaneamente, e com isso o termo  $\Delta i / \Delta t$  tende ao infinito. Resulta que a tensão nos terminais da bobina tende ao infinito. Esta alta tensão gera um arco elétrico na chave, pois a tensão nos terminais chega a tal valor que rompe a rigidez dielétrica do ar (cerca de 1000 V/mm).

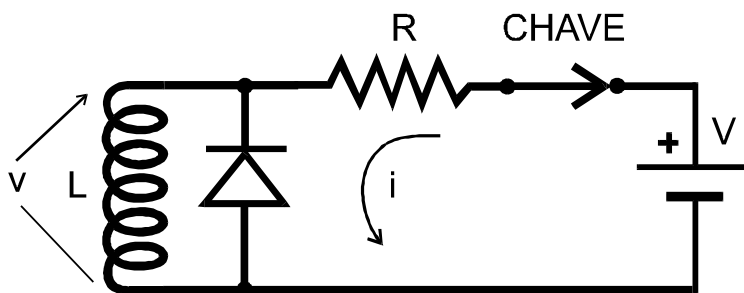
Por exemplo, digamos que a indutância de determinada válvula solenóide é de 10 mHy, a corrente de acionamento da válvula é 200 mA, e a tensão de alimentação da válvula é 24 V. O circuito está ligado quando, repentinamente, a chave é aberta. Vamos supor que a chave leve cerca de 1  $\mu$ s para abrir. Ora, a tensão nos terminais da bobina nesta situação atingiria:

$$v = 10 \times 10^{-3} \left( -0,2 / 10^{-6} \right) = -2000V !!!$$

Ou seja, apesar do circuito ser alimentado com uma tensão de apenas 24V, ao abrir a chave (que pode ser o contato de um relé do CLP) a tensão atinge milhares de volts!

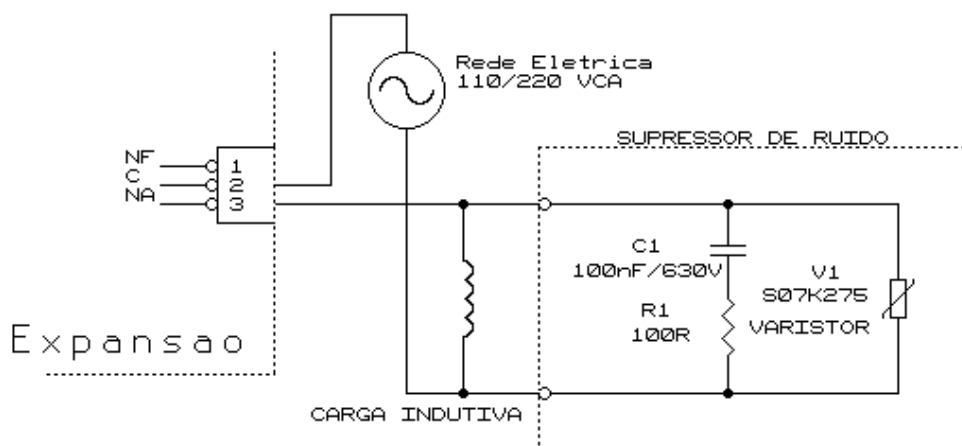
Na verdade, o que ocorre é que existe energia armazenada no campo magnético da bobina, e esta energia é dissipada muito rapidamente no arco elétrico que se forma nos contatos da chave. Obviamente, este arco deve ser evitado, pois diminui muito a vida útil da chave (ou relé do CLP), e a alta tensão gerada pode interferir com sinais de entrada do CLP. A solução é providenciar um caminho para a corrente da bobina, quando a chave é aberta.

No caso de circuitos de corrente contínua, um diodo pode fornecer a solução:



Quando a chave é aberta, a bobina gera a tensão contra-eletromotriz, mas o diodo passa a conduzir quando esta atinge cerca de -0,7V, mantendo a corrente na bobina. A corrente diminui lentamente devido a resistência associada a qualquer bobina (exceto se a bobina fosse feita de material super-condutor), e evita-se o arco na chave.

Já se o circuito for usado em corrente alternada, não é possível colocar um diodo em paralelo com a bobina, pois durante meio-ciclo da rede ele entraria em condução. Neste caso pode-se usar filtros RC (resistor-capacitor), e também varistores (resistores variáveis com a tensão). Abaixo temos um exemplo de filtro RC + Varistor ligado a uma saída da Expansão de Entradas e Saídas do Controlador  $\mu$ DX, da Dexter.



### Programa Aplicativo

A lógica que avalia a condição dos pontos de entrada e dos estados anteriores do CLP, executando as funções desejadas e acionando as saídas, é chamada de programa aplicativo ou simplesmente programa do CLP.

Para isso, o CLP lê ciclicamente as entradas, transferindo-as para uma memória imagem (que recebe em cada endereço correspondente a uma entrada o seu valor – 0 ou 1 no caso de entradas digitais, ou um valor numérico no caso de entradas analógicas).

De posse da memória imagem e dos estados internos gerados pelos ciclos de execução anteriores, o CLP gera uma memória imagem das saídas conforme as operações definidas no programa.

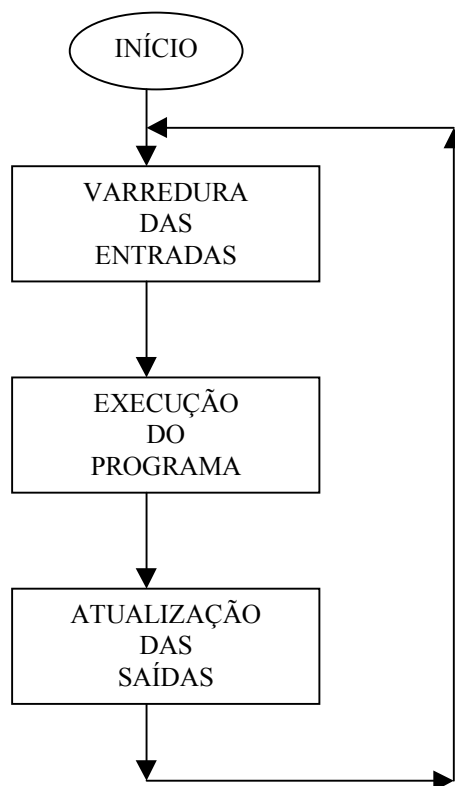
Por fim, a memória imagem das saídas é transferida para as saídas (valor 0 ou 1 causa o desligamento ou acionamento de uma saída digital, ou um valor numérico modifica o valor de corrente ou tensão de uma saída analógica).

Como para qualquer controle ou automatização é necessário o maior grau de paralelismo possível (em qualquer processo sempre pode ocorrer mais de um evento diferente ao mesmo tempo) é empregado nos CLPs um método que simula paralelismo.

Neste método os parâmetros de entrada (estado de ligações e valores de variáveis) são mantidos numa tabela acessível por qualquer um dos blocos de instrução que esteja sendo interpretado (memória imagem das entradas). Uma segunda tabela (memória imagem das saídas), com os resultados produzidos pela interpretação de cada bloco, vai sendo montada a medida que os blocos vão sendo lidos e interpretados.

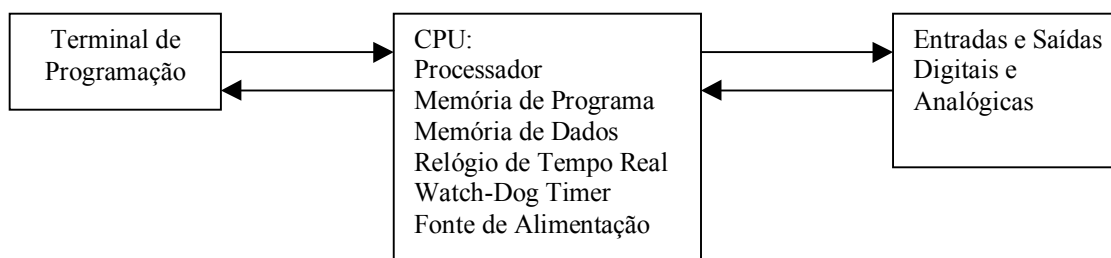
Assim, cada bloco poderá utilizar qualquer um dos parâmetros de entrada sem que estes sejam alterados devido à interpretação de algum outro bloco. Depois, no final do ciclo, a tabela de saída (com os resultados) é movida diretamente para a tabela de entrada para que os novos valores estejam disponíveis igualmente para todos os blocos no próximo ciclo.

É fácil perceber que esta forma de funcionamento faz com que todos os blocos sejam interpretados em paralelo, o que permite a elaboração de programas segmentados, onde cada parte pode controlar um processo independentemente e ao mesmo tempo que as demais.



Este paralelismo, operado em ciclos, faz com que a atualização da saída de um bloco de instrução para a entrada de um ou mais blocos demore o equivalente ao tempo de um ciclo. Esta demora, ou atraso, deve ser considerado no planejamento de um programa pois a conexão "encadeada" de, por exemplo, 10 blocos de instrução terá um atraso de 10 ciclos desde o estímulo na entrada do primeiro bloco até a saída no último. Com um tempo de ciclo de 1/16s do  $\mu$ DX isto resultaria em um atraso de 0,625 segundos.

### Componentes de um CLP



O terminal de programação é um dispositivo que, conectado temporariamente ao CLP, permite introduzir o programa aplicativo, fazendo com que esse se comporte conforme a necessidade de controle de processo do usuário. Além disso, o terminal de programação permite, muitas vezes, monitorar o programa aplicativo, ou seja, visualizar em tempo real o programa sendo executado, ou ainda executá-lo passo a passo. Alguns CLPs permitem, inclusive, a simulação do programa aplicativo (sua execução apenas no terminal de programação, com fins de depuração).

Atualmente, o mais usual é a utilização de um microcomputador IBM-PC compatível como terminal de programação (na versão desktop ou laptop, para programação em campo). Os fabricantes de CLPs disponibilizam os softwares de programação (que rodam sob DOS ou Windows) e cabos para conexão ao CLP (normalmente, pela porta serial do micro e, mais raramente, pela porta paralela, como é o caso do controlador  $\mu$ DX).

A CPU (unidade central de processamento) é a responsável pelo armazenamento do programa aplicativo e sua execução. Ela recebe os dados de entrada, realiza as operações lógicas baseada no programa armazenado e atualiza as saídas. Consta de um processador, memória de programa (não-volátil), memória de dados, relógio de tempo real (para disparo de eventos em datas e horários determinados), watch-dog timer (reinicializa o processador no caso do programa “pendurar”) e fonte de alimentação.

As Entradas e Saídas são módulos responsáveis pela interface do CLP com o ambiente externo, realizando a adaptação de níveis de tensão e corrente, filtragem dos sinais (ruído elétrico), e conversão de sinais analógicos em digitais e vice-versa. Pequenos CLPs, como o  $\mu$ DX, abordado neste curso, possui os módulos de entradas e saídas integrados a CPU.

## Linguagens de Programação

### Linguagem de Relés e Blocos (Ladder)

Trata-se de uma linguagem gráfica que permite transladar com relativa facilidade os diagramas elétricos baseados em relés para o CLP. Existe uma linha vertical de energização a esquerda e outra linha a direita. Entre estas duas linhas existe a matriz de programação formada por xy células, dispostas em x linhas e y colunas. Abaixo exemplificamos um caso de 32 células, dispostas em 4 linhas e 8 colunas


Barra de  
energia esquerda

Barra de  
energia direita

Cada conjunto de 32 células é chamado de uma lógica do programa aplicativo. As duas linhas laterais da lógica representam barras de energia entre as quais são colocadas as instruções a serem executadas. As instruções podem ser contatos, bobinas, temporizadores, etc.

A lógica deve ser programada de forma que as instruções sejam “energizadas” a partir de um “caminho de corrente” entre as duas barras, através de contatos ou blocos de funções interligados. Entretanto, o fluxo de “corrente elétrica” simulado em uma lógica flui somente no sentido da barra de energia esquerda para a direita, diferentemente dos esquemas elétricos reais. As células são processadas em colunas, iniciando pela célula esquerda superior e terminando pela célula direita inferior.

Cada célula pode ser ocupada por uma conexão (“fio”), por um bloco (relé de tempo, operação aritmética, etc), ou ainda por um contato ou bobina.

Além disso, existem algumas regras impostas na linguagem Ladder. Por exemplo, as bobinas devem ocupar somente a última coluna a direita.

Abaixo temos a ordem de execução das células em uma lógica Ladder. Note que o programa aplicativo pode ser composto de várias lógicas Ladder. Além disso, um módulo de configuração permite especificar parâmetros do CLP, como modelo, velocidade de ciclo, endereço do CLP na rede de comunicação, etc.

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

### Linguagem de Diagrama Esquemático

Também é uma linguagem gráfica, usada pelo controlador programável  $\mu$ DX Série 100, da Dexter. Nesta linguagem, as células estão dispostas em uma área de edição única, com 10 colunas e um número variável de linhas. Os elementos a serem dispostos nas células são blocos ou nodos. Os blocos representam as instruções a serem executadas pelo programa, e os nodos são os “fios” de conexão entre os nodos.

Não existem barras de energização (embora possam ser criadas pelo usuário). Um bloco de ENERGIA permite energizar qualquer ponto do programa. É permitido fazer ligações cruzadas ou energizar nodos da direita para a esquerda (o fluxo de “corrente elétrica” simulado flui em qualquer direção). Apenas os contatos permitem uma única direção da “corrente”.

O programa processa as células da esquerda para direita, e de cima para baixo:

1 Config	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40

```

•      •      •      •      •      •      •      •      •      •
•      •      •      •      •      •      •      •      •      •
•      •      •      •      •      •      •      •      •      •

```

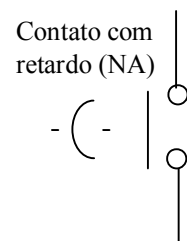
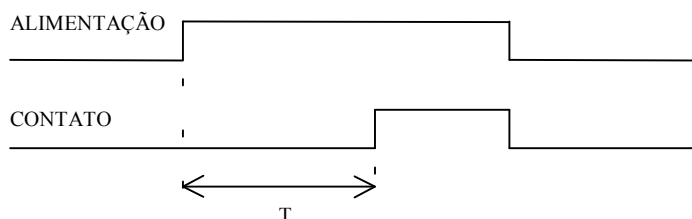
A primeira célula contém o bloco de configuração, onde se define a velocidade de ciclo do programa e o endereço do controlador na rede DXNET (rede de comunicação entre CLPs, periféricos e microcomputador).



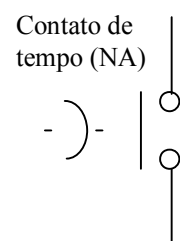
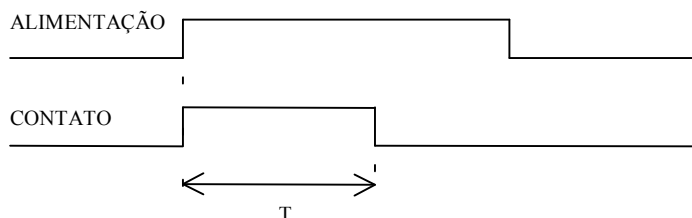
No curso serão abordados exemplos de programação via linguagem de diagrama esquemático em controlador programável  $\mu$ DX Série 100, da Dexter. Serão mostradas implementações em Ladder dos mesmos exemplos, para ilustrar a similaridade entre as linguagens.

## Revisão de Relés Temporizadores e Contactoras

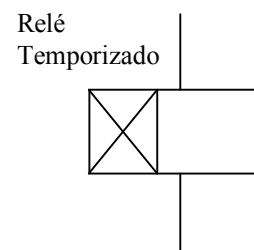
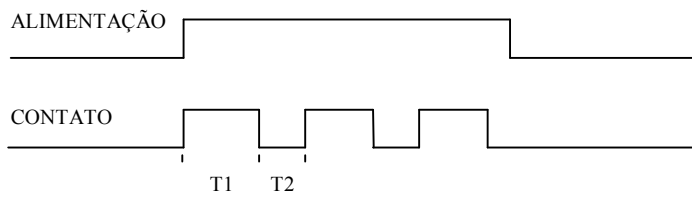
Relé de Retardo  
(Atraso)



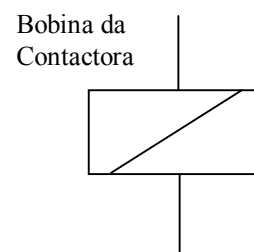
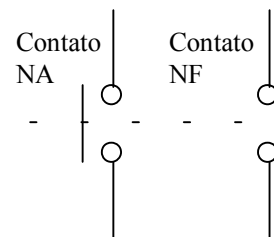
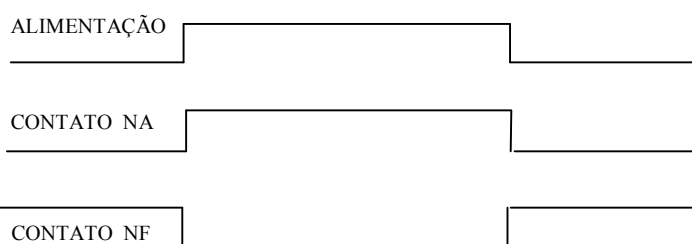
Relé de Tempo  
(Monoestável)



Relé Cíclico  
(Oscilador)



Contactora

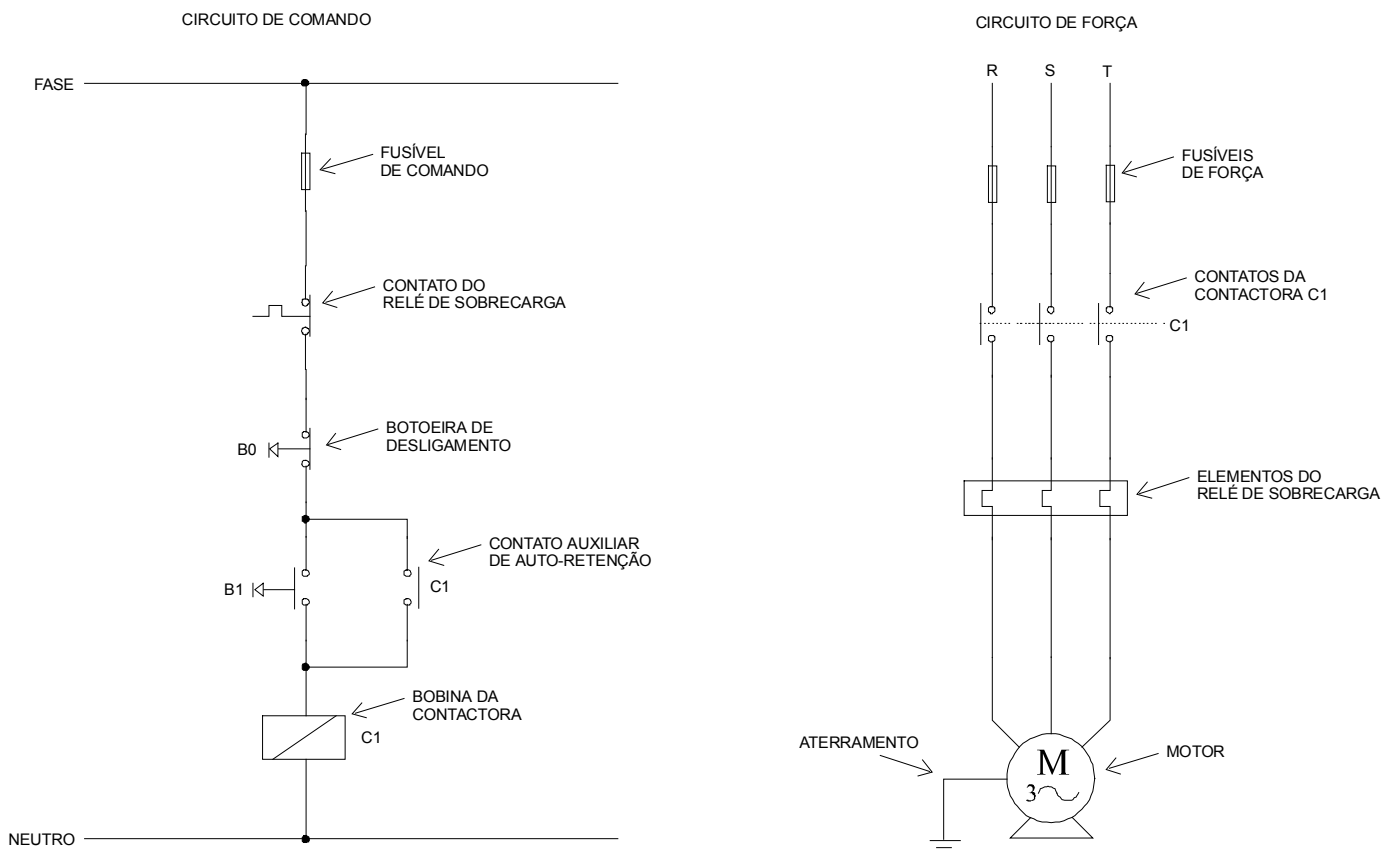


Nota: NA = Normal aberto.  
NF = Normal fechado.

## Revisão de Circuitos de Comando e Força

### Circuito de Partida Direta

O circuito abaixo permite partir ou parar um motor, através de dois botões de contato momentâneo (botoeiras). Note o contato auxiliar da contactora, usado para manter sua energização após o operador soltar o botão de partida (B1). Já o botão de parada (B0) é do tipo normal fechado (NF). Ao ser pressionado ele interrompe o circuito, desenergizando a contactora e, portanto, abrindo também o contato auxiliar de auto-retenção.

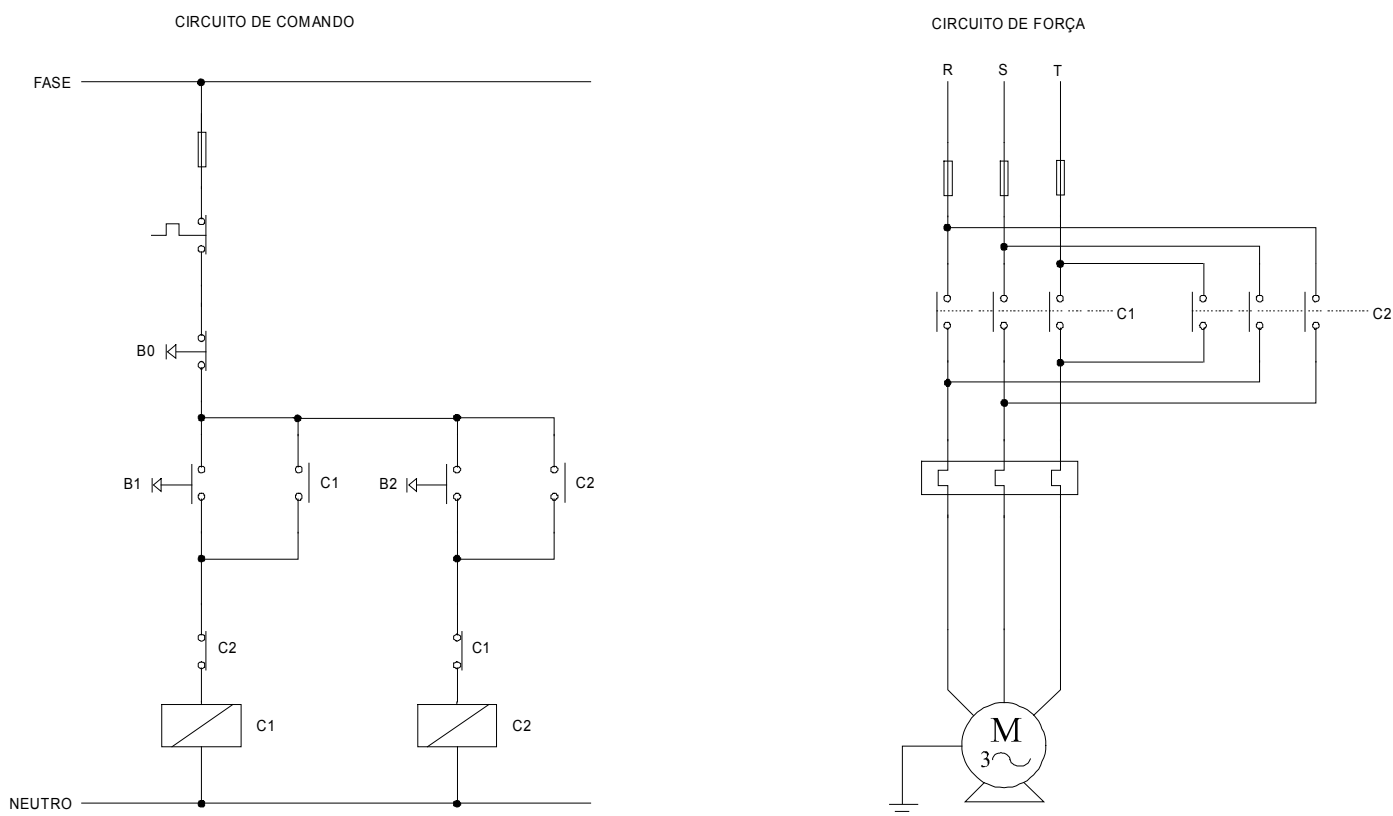


Note que este circuito, no caso de interrupção da rede elétrica, se desarma automaticamente. Isso é importante para segurança. Caso simplesmente fosse utilizada uma chave 1 pólo, 2 posições para acionar a contactora, ao retornar a energia elétrica (no caso de um “apagão”, por exemplo) o motor seria energizado, pois a chave se manteria na posição ligada.

### Circuito de Partida Direta Reversora

Neste caso existem dois botões de contato momentâneo para partir o motor (B1 e B2). Um deles faz o motor girar no sentido horário e o outro no sentido anti-horário. Um terceiro botão desliga o motor (B0), independentemente do sentido de rotação. Note os contatos auxiliares NA das contadoras usados para auto-retenção. Além disso, as contadoras se inibem mutuamente através dos contatos auxiliares NF. Assim, se a contactora C1 estiver energizada, a contactora C2 não pode ser energizada, e vice-versa. Isso impede que o operador, inadvertidamente, acione simultaneamente os dois sentidos de giro do motor. Caso as duas contadoras fossem energizadas simultaneamente, o resultado seria a queima dos fusíveis de força (pois teríamos curto-circuito entre as fases R e S).

Note que para inverter o giro do motor basta inverter duas fases (no caso, são invertidas as fases R e S).



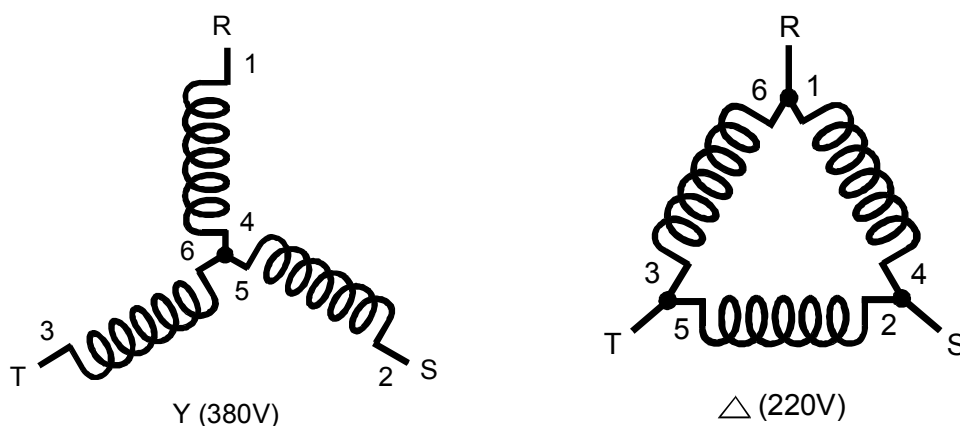
### Circuito de partida Estrela-Triângulo

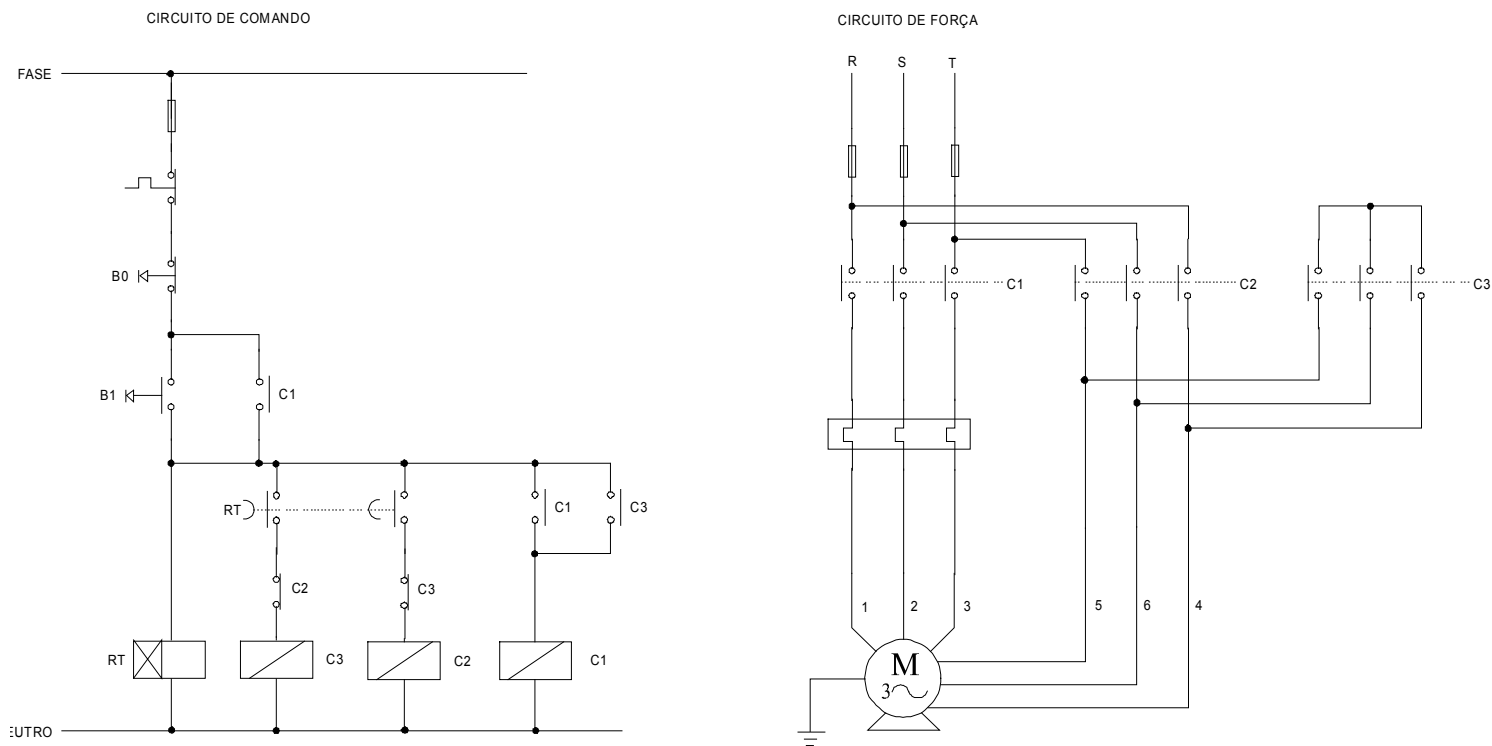
Neste caso, partimos o motor na configuração estrela, de forma a minimizar a corrente de partida e, após determinado tempo especificado no relé temporizado, comuta-se o motor para a configuração triângulo.

Ao pressionar B1, energiza-se a contactora C3, que por sua vez energiza a contactora C1. Isso liga o motor a rede trifásica na configuração estrela. Após o tempo especificado no relé temporizado RT, a contactora C3 é desenergizada e a contactora C2 energizada. C1 continua energizada, pois existe um contato auxiliar de C1 para efetuar sua auto-retenção. Com isso, o motor é conectado a rede trifásica na configuração triângulo.

Note os contatos auxiliares NF que fazem com que jamais as contactoras C1 e C2 possam ser energizadas simultaneamente.

A botoeira B0, quando pressionada, interrompe todo circuito.





## Instruções Básicas Ladder

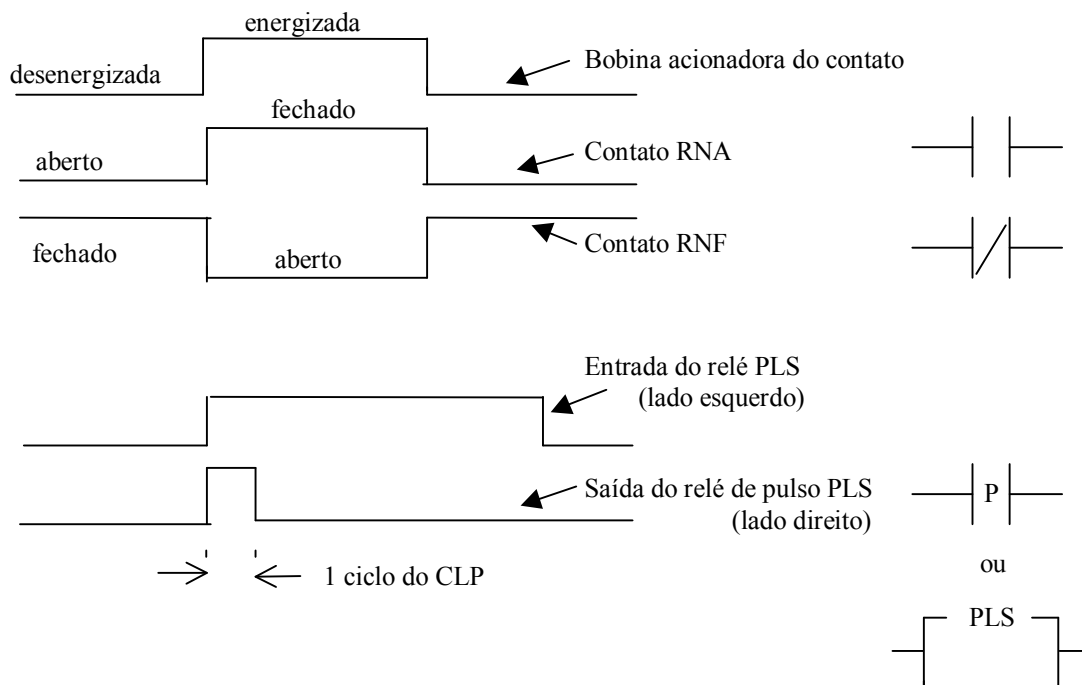
### Contatos

Existem os seguintes tipos de contato:

**RNA** → Contato normalmente aberto.

**RNF** → Contato normalmente fechado.

**PLS** → Relé de pulso: Contato detector de variação.



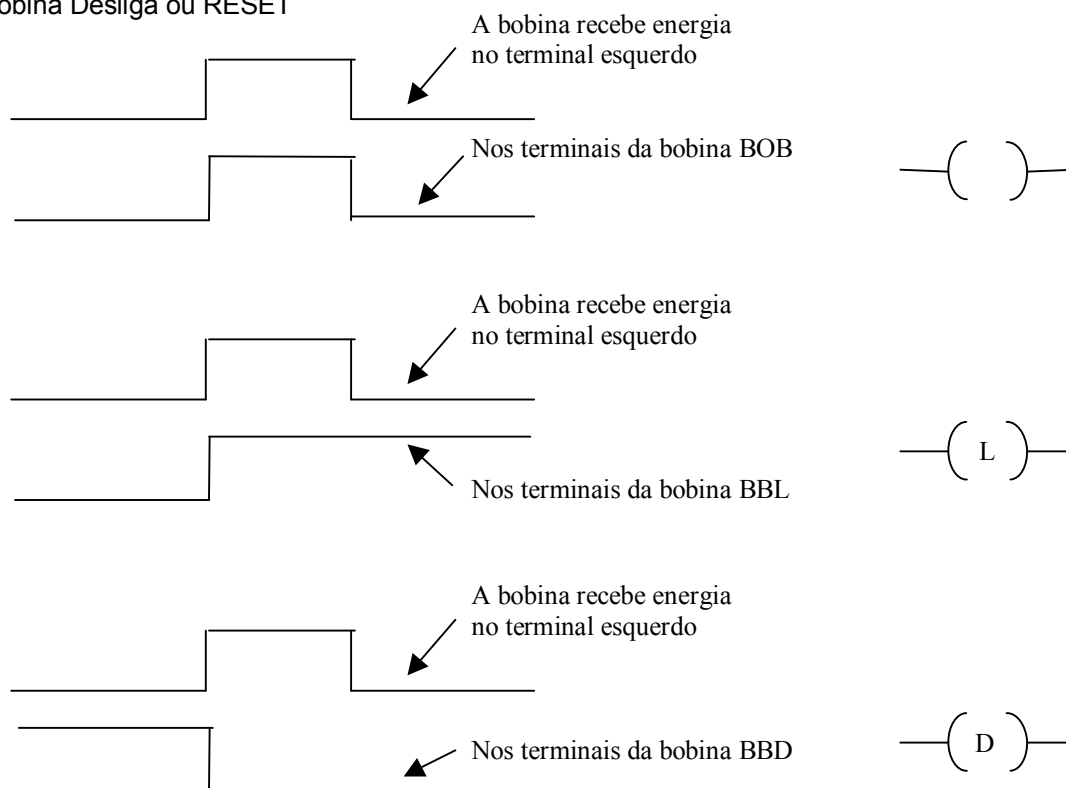
### Bobinas

Existem os seguintes tipos principais de bobinas:

**BOB** → Bobina simples

**BBL** → Bobina Liga ou SET

**BBD** → Bobina Desliga ou RESET

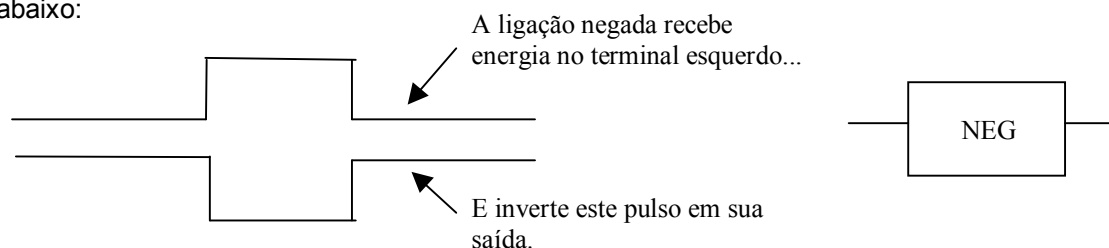


Como podemos perceber, a bobina simples comporta-se como uma contactora comum, ou seja, quando energizada aciona seus contatos. Já a bobina liga e bobina desliga funciona como uma contactora com retenção, ou seja, um pulso em bobina liga aciona a contactora, e um pulso em bobina desliga a desativa (como um flip-flop R-S). Note que todos os contatos associados a uma bobina são acionados quando esta bobina é energizada.

As bobinas podem ser associadas a saídas digitais do CLP, e contatos podem ser associados a entradas digitais. Assim, ao energizar uma entrada o contato associado a ela é acionado (se for um contato NA ele se fechará; se for um contato NF irá abrir). Já ao energizar uma bobina associada a uma saída do CLP fazemos com que esta saída seja ativada (feche o contato do relé de saída, por exemplo).

### Ligações

As ligações são os “fios” de interconexão entre as células da lógica Ladder (contatos, bobinas e blocos de funções). Podemos ter ligações na horizontal, na vertical, e ainda uma ligação negada (inversora). As ligações horizontais e verticais simplesmente conectam saídas de células as entradas de outras células. Já a ligação negada inverte o sinal na sua entrada, como mostrado abaixo:



Tente vislumbrar a diferença entre uma ligação negada (que inverte o valor binário em sua entrada) com a chave NF, que abre a ligação entre sua entrada e sua saída quando a bobina associada a ela é energizada.

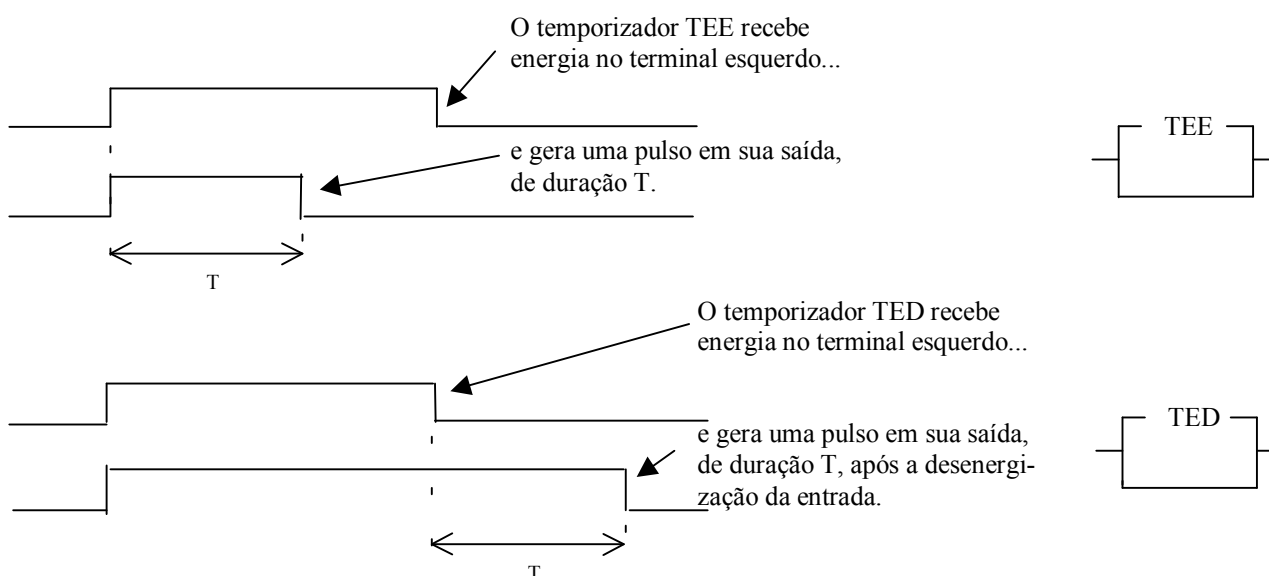
Ainda existem vários blocos para operações e testes aritméticos, temporizadores, etc. Vamos examinar apenas dois temporizadores (outros detalhes podem ser obtidos nos manuais dos fabricantes).

### Temporizadores

Vamos examinar dois tipos de temporizadores:

**TEE** → Temporizador na energização.

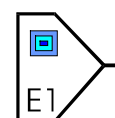
**TED** → Temporizador na desenergização.



### Instruções de Diagrama Esquemático

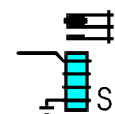
#### Entradas

Trata-se das entradas digitais do CLP. Na verdade, este símbolo apenas designa que o nodo ("fio" de ligação entre os blocos) está conectado a uma entrada digital do CLP.



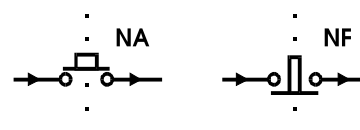
#### Saídas

Trata-se das saídas digitais do CLP. Note o desenho, representando um relé. Este símbolo associa o nodo ligado a sua entrada a uma saída digital do CLP.



#### Chaves NA e NF

A chave NA (normal aberta) fecha o circuito entre sua entrada (terminal a esquerda) e a saída (terminal a direita), caso o nodo de controle (terminal superior ou inferior; linha pontilhada) estiver energizado. É similar a uma chave interruptora comum.

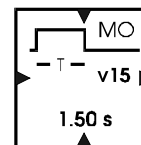
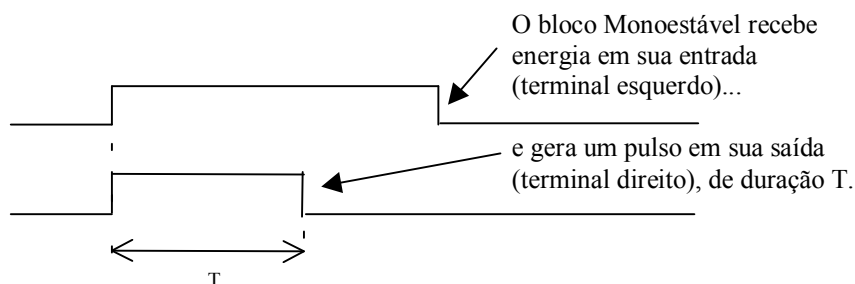


A chave NF (normal fechada) abre o circuito entre sua entrada (terminal a esquerda) e a saída (terminal a direita), caso o nodo de controle (terminal superior ou inferior; linha pontilhada) estiver energizado. É similar a uma chave de porta de geladeira. Ao pressionar a chave, quando fechamos a geladeira, ela desliga a lâmpada interna.

Note as setas indicando sentido da esquerda para direita (da entrada para saída). Elas indicam que as chaves permitem apenas este sentido de energização. Ou seja, uma energização em uma saída de uma chave não irá se propagar para a entrada da chave.

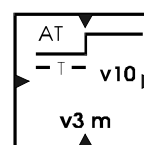
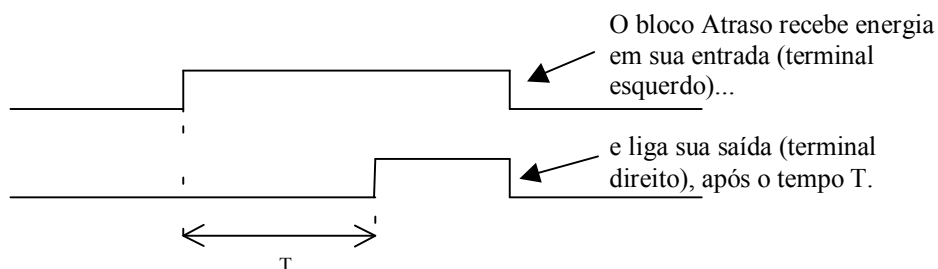
### Monoestável

Trata-se de um relé de tempo (o equivalente ao TEE, da Ladder). Ou seja, ao energizar o nodo de entrada o nodo de saída permanece energizado durante um tempo programado T.



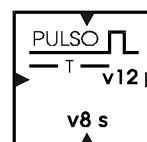
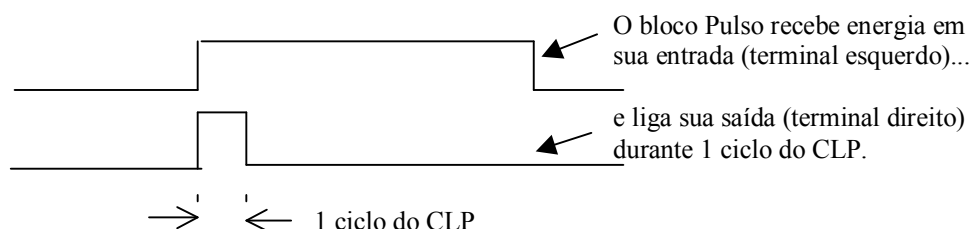
### Atraso

Trata-se de relé de retardo. Ou seja, ao energizar o nodo de entrada o nodo de saída é energizado, após um tempo programável T. Ao desenergizar o nodo de entrada o nodo de saída também é desenergizado.



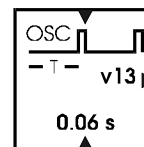
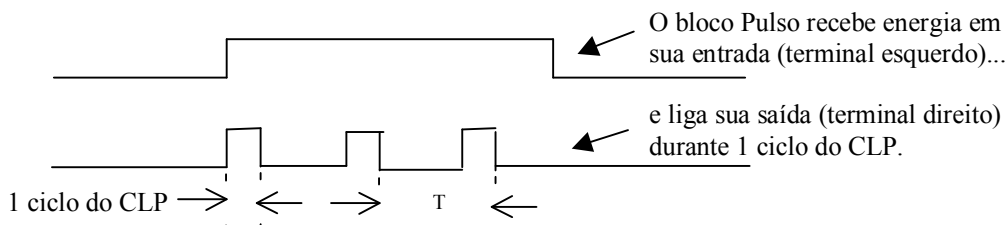
### Pulso

Este bloco opera exatamente da mesma forma que o de atraso, exceto que o atraso mantém a saída ligada (após o tempo de atraso especificado) enquanto a entrada estiver ligada e o pulso produz apenas um pulso na saída, com a duração de 1 ciclo do CLP.



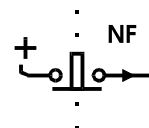
### Oscilador

Este bloco produz pulsos constantemente na saída, espaçados por um intervalo de tempo programável. Permite piscar lâmpadas ou acionar de forma intermitente sirenes de alarme, por exemplo. Os pulsos de saída têm duração de 1 ciclo de CLP, e possuem período programável T.



### Inversor

A chave inversora é, na verdade, uma chave NF com a entrada ligada diretamente à fonte positiva. Assim, se o controle estiver desativo sua saída estará ativa e vice-versa, produzindo a inversão de sinal.



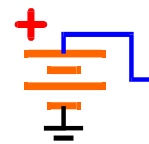
### Relógio

Este bloco permite disparar um processo em um determinado horário. É possível especificar dia da semana, hora e minuto. Note o nodo de controle pontilhado, à esquerda. Este nodo deve estar ativo para que o nodo de saída ligue no horário determinado.



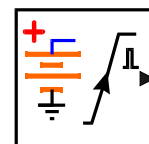
### Energia

Este bloco permite forçar a ativação de determinada ligação (nodo). Assim, ele equivale a ligar esta linha ao positivo do circuito. Por exemplo, o bloco de relógio tem uma linha de controle que deve estar ativa para permitir seu funcionamento. Se quisermos que o relógio funcione constantemente basta conectar à sua linha de controle um bloco de energia.



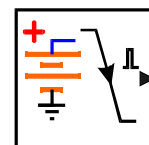
### Nodo EL

Este bloco produz um pulso na saída sempre que o controlador é energizado. Com isso, este bloco permite inicializar parâmetros do programa aplicativo como, por exemplo, valor de variáveis.



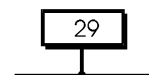
### Nodo ED

Este bloco efetua a mesma função do bloco anterior, só que o pulso é gerado ao faltar energia elétrica. Obviamente, este bloco só irá ser efetivo se o CLP estiver alimentado por pilhas.



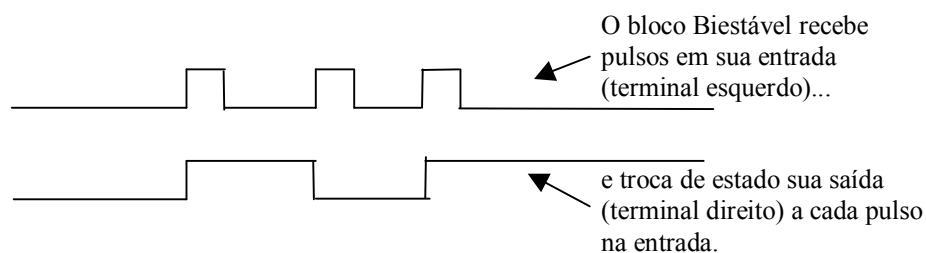
### Rótulo

O rótulo permite conectar dois pontos do programa sem a necessidade de “puxar” uma linha de um ponto ao outro. Isto é útil em programas complexos, em que há dificuldade para efetuar todas as ligações.



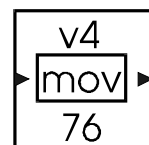
### Biestável

Este bloco, a cada borda de subida do sinal de entrada, troca de estado sua saída. Com ele podemos memorizar algum estado no programa ou dividir a frequência de saída de um oscilador.



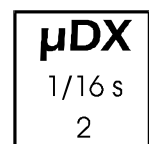
### Função

Este bloco manipula variáveis do programa. As funções permitidas são soma, subtração, mover valor, deslocamento de bit à direita ou à esquerda e operações lógicas como AND, OR e XOR. Além disso, este bloco permite testes como se a variável é maior ou menor que determinado valor e teste de bit.



### Bloco µDX

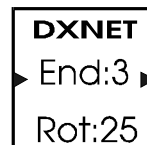
Este bloco sempre está presente no canto esquerdo superior do programa aplicativo. Serve para indicar parâmetros do controlador, como seu endereço na rede local DXNET e a duração do ciclo do CLP.





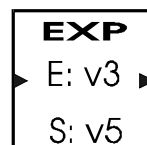
### DXNET

Este é o bloco que permite a intercomunicação com vários  $\mu$ DX utilizando-se a rede local DXNET. Com ele podemos transferir o estado de uma ligação (ativa ou desativa) ou o valor de uma variável de um  $\mu$ DX para qualquer outro  $\mu$ DX. Com este recurso, pode-se fazer programas bastante complexos apenas distribuindo as tarefas entre os vários controladores ligados em rede.



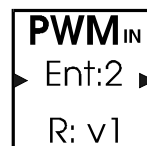
### Expansão

O bloco de expansão acessa o conector de expansão existente no  $\mu$ DX que, através de circuitos opcionais, permite aumentar o número de entradas e saídas de 4 de cada para 12 de cada ou a instalação de teclado/display para entrada de dados.



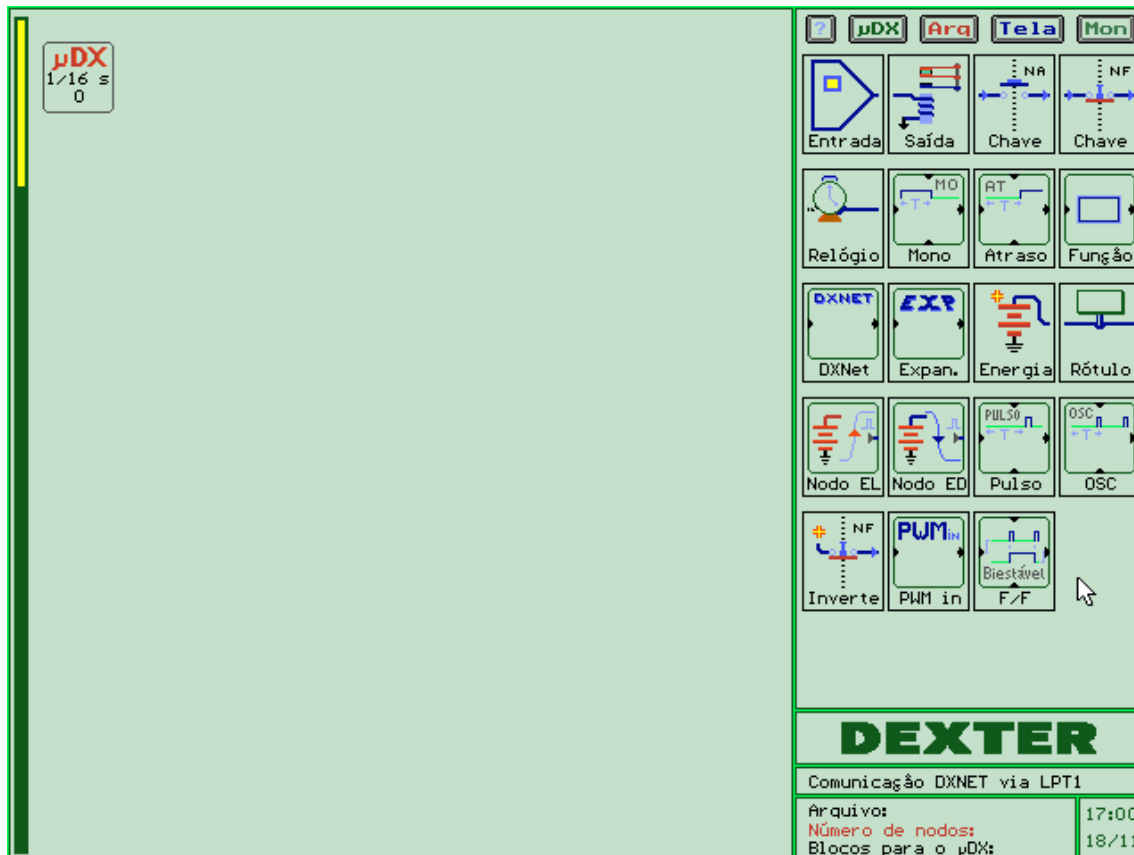
### PWMin

Este bloco de instrução permite que o  $\mu$ DX converta um sinal modulado em largura de pulso para um valor de 8 bits a ser guardado em uma variável do programa. O bloco possibilita a leitura de sinais analógicos, pois efetua a conversão analógica para digital (A/D de 8 bits). Com um circuito usando o tradicional timer 555 e poucos componentes adicionais pode-se monitorar tensões, temperatura, etc.

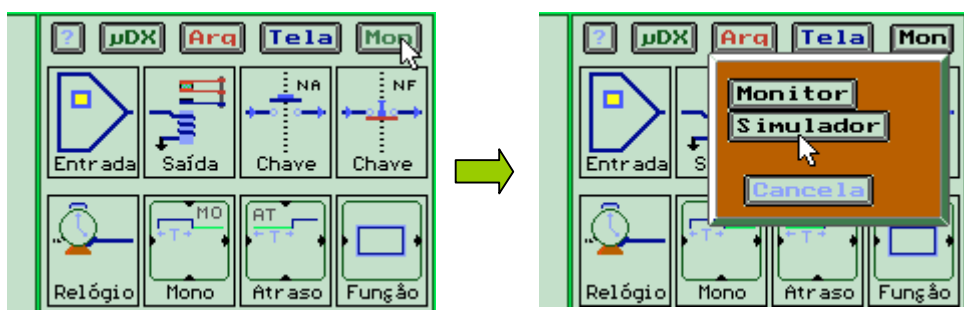
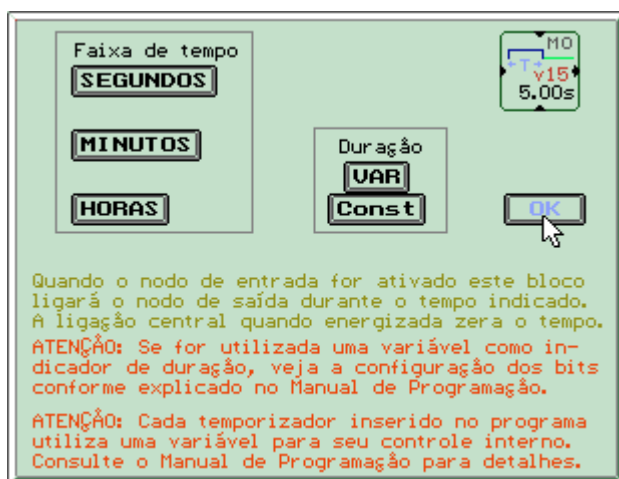
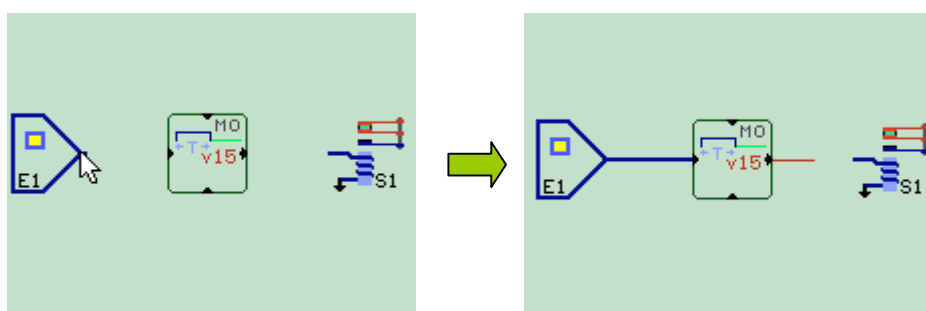
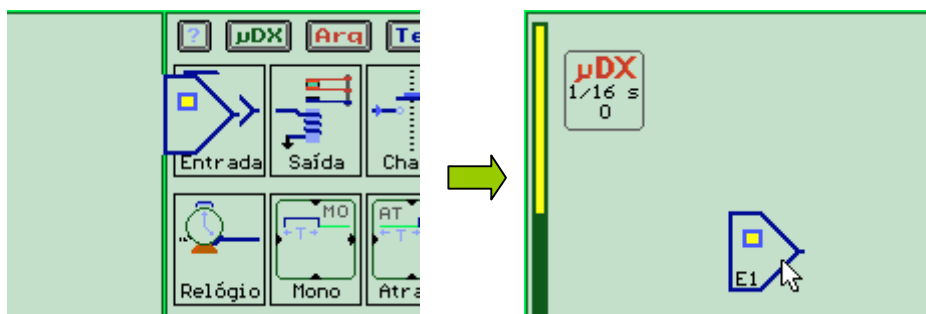


## Como Programar o Controlador $\mu$ DX

Para programar o controlador  $\mu$ DX usa-se o software PG – Programador Gráfico, que acompanha o equipamento e também está disponível na Internet, em [www.dexter.ind.br](http://www.dexter.ind.br). Para instalar o software execute o programa SETUP.EXE e siga as orientações do software de instalação (Install Shield). Ao rodarmos o PG surge a tela de apresentação. Ao pressionar qualquer tecla esta tela é substituída pela tela principal do programa:



Nesta tela notamos, a direita, uma série de desenhos que representam os diferentes "componentes" que irão formar o "circuito" a ser transmitido para o  $\mu$ DX. A área livre, a esquerda, é onde o "circuito" é elaborado. Basta capturar com o mouse os blocos existentes e ir montando o circuito nessa área. Por exemplo, se quisermos fazer um relé temporizado que, uma vez ativado, fique ligado durante 5 segundos, independentemente da entrada, desenhamos:



Primeiro, capturamos uma entrada através do mouse (aponte para as entradas e clique uma vez a tecla esquerda do mouse). A seguir, largamos a entrada na área de edição (clique novamente a tecla esquerda do mouse).

Faça o mesmo com os outros blocos (mono-estável e saída).

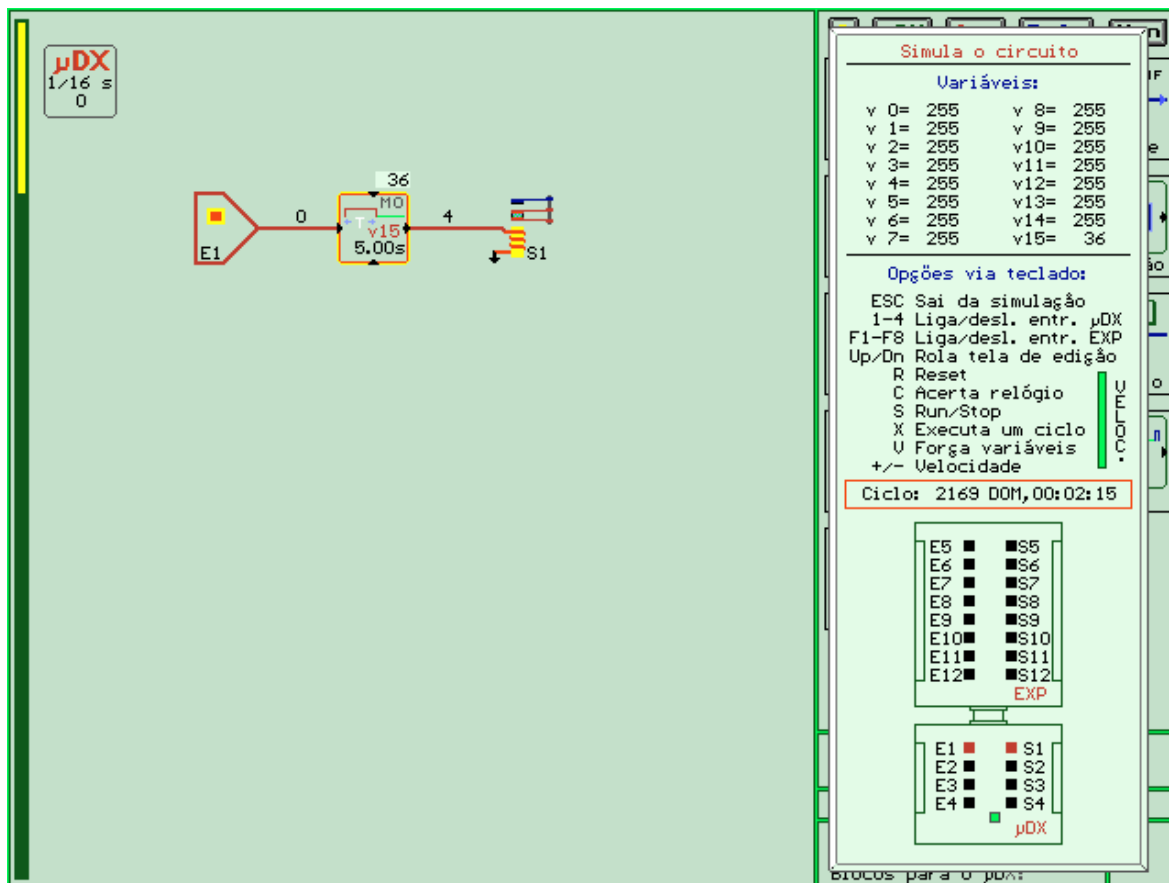
Então, aponte para o vértice do bloco de entrada e pressione o mouse. Com as setas do teclado una a entrada E1 a entrada do mono e sua saída a saída S1.

Por fim, aponte com o mouse para o centro do monoestável e pressione a tecla [E] do teclado do computador.

Programa 5 segundos de duração do pulso do monoestável via tecla [Const]. Pressione [OK] e está pronto o programa!

Para simular este programa selecione a tecla [Mon] (canto superior direito do vídeo) e, a seguir, [Simulador].

Pressione a tecla [S] para iniciar a simulação. A tecla [1] ativa a entrada E1 do  $\mu$ DX simulado, iniciando a temporização. Utilize as teclas [+] e [-] para aumentar ou diminuir a velocidade de simulação, ou ainda a tecla [X] para executar o programa passo a passo.

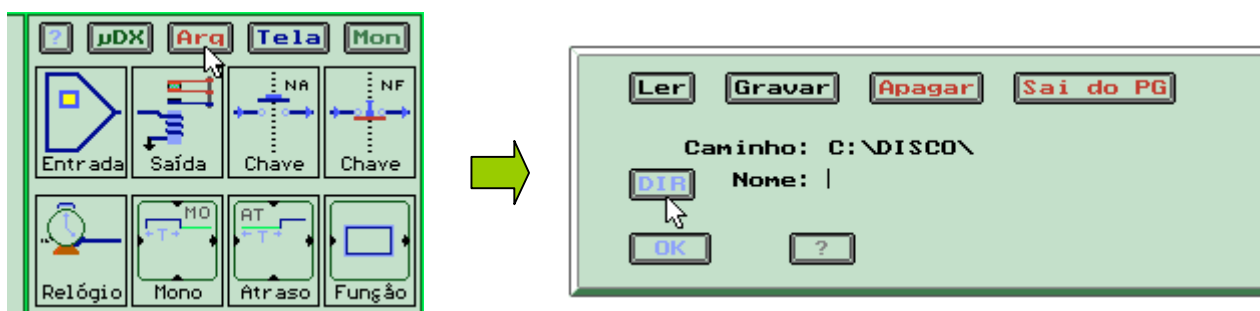


Se transmitirmos este programa ao μDX, cada vez que for energizada a entrada E1 a saída S1 será acionada durante 5 segundos. Para transmitir o programa basta conectar o μDX a porta paralela do microcomputador via cabo próprio (acompanha o μDX) e pressionar a tecla [μDX] existente na tela principal. Esta tecla abre uma janela com várias opções, entre elas compilação do programa e transmissão para o μDX:

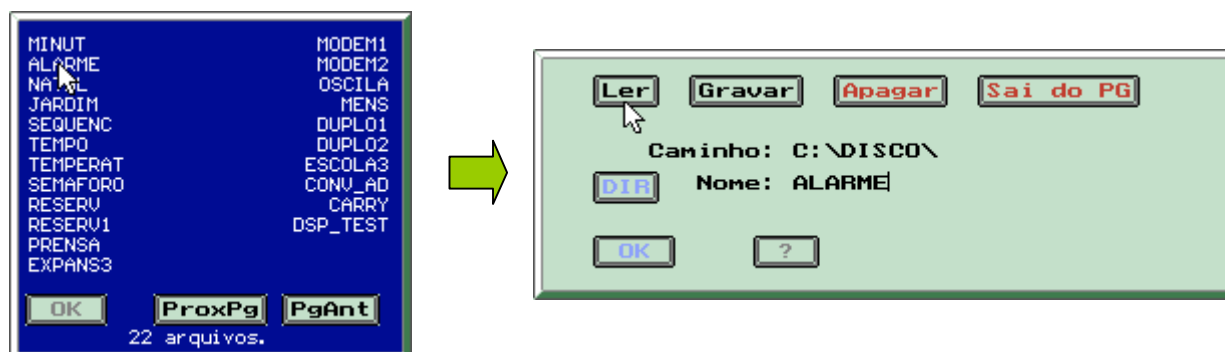


## Programas Aplicativos que acompanham o PG

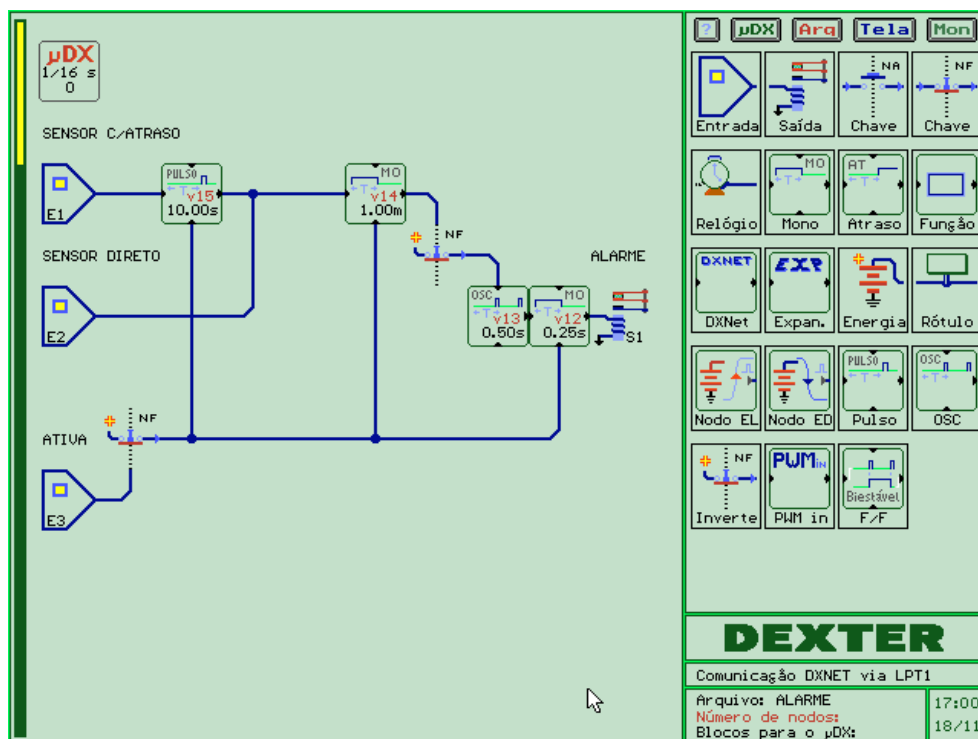
Vamos mostrar algumas aplicações do  $\mu$ DX. O objetivo aqui é dar uma idéia da facilidade de programação e suas particularidades. Note que, devido à programação ser por blocos, é muito fácil ter mais de um processo sendo controlado pelo mesmo  $\mu$ DX. Basta desenhar o diagrama de controle de cada processo independentemente na tela do computador. A tela real de programação é bem maior que a área visível de tela, pois o programa permite "scroll" (mover a tela para cima ou para baixo). Assim, é possível, com o mesmo  $\mu$ DX, por exemplo, controlar a irrigação de jardim, o alarme residencial e ainda um simulador de presença para quando o proprietário se ausenta. Em alguns exemplos é mostrada também a implementação do programa em Ladder, de forma a facilitar a comparação entre as linguagens. Vários programas de exemplo são fornecidos com o PG, e explicados no manual do Controlador  $\mu$ DX. Para carregá-los primeiro clique na tecla [Arq] e, a seguir, em [DIR]:



Deve surgir uma janela azul, com todos os programas de exemplo que acompanham o PG. Selecione o programa desejado apontando-o e pressionando a tecla esquerda do mouse. Por fim, pressione a tecla [Ler]:



Por exemplificar, carregue o programa ALARME. Note os sinais de entrada: E1, E2 e E3. E1 dispara o alarme com retardo de 10 segundos. Esta entrada pode ser ligada a um sensor magnético (reed-switch) ligado na porta de entrada da residência. Assim, o proprietário tem 10 segundos para desativar o alarme após entrar em casa, antes que este dispare. A entrada E2 dispara o alarme imediatamente. Esta entrada pode sensoriar todas as outras aberturas (janelas, outras portas) através de sensores magnéticos colocados em série. Caso qualquer um deles se abrir o alarme será disparado. Por fim, E3 ativa o alarme. Esta entrada deve ser ligada a uma chave, permitindo ligar o alarme. A saída S1 é usada para acionar uma buzina ou sirene de forma intermitente (0,25 s ligado e 0,25 s desligado). Uma vez disparado, o alarme "toca" durante 1 minuto e depois se rearma. Note que foram usados apenas 8 blocos para elaborar este programa (entradas e saídas não são contadas como blocos). Como o limite de blocos para o  $\mu$ DX série 100 é de 127, podemos tornar o alarme muito mais complexo ou colocar outros programas para rodar simultaneamente no mesmo  $\mu$ DX!

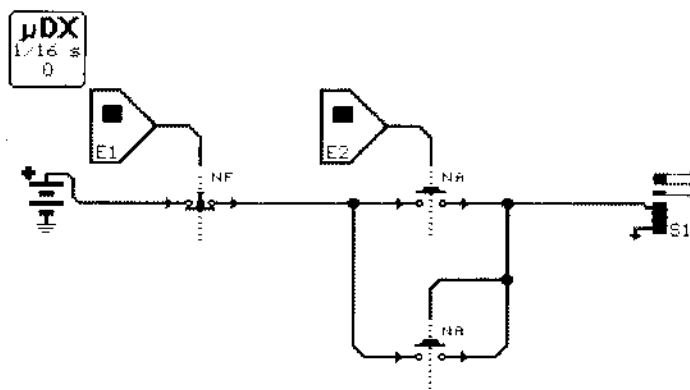


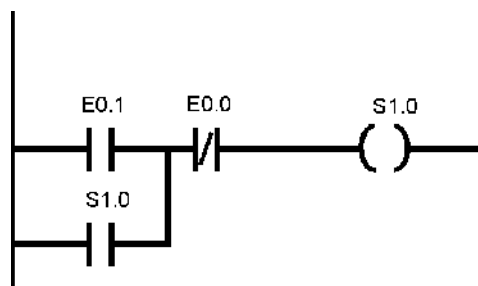
## Exemplos de Implementação

### Chave de partida direta

Abaixo temos a chave de partida direta, implementada em linguagem de diagrama esquemático no controlador  $\mu$ DX Série 100. Note a chave NA de retenção. A entrada E1 desliga a saída S1, e a entrada E2 liga a saída S1.

Controlador Programavel uDX Serie 100  
 Nome do programa: AULA01.UDX  
 Numero de blocos para uDX utilizados: 3



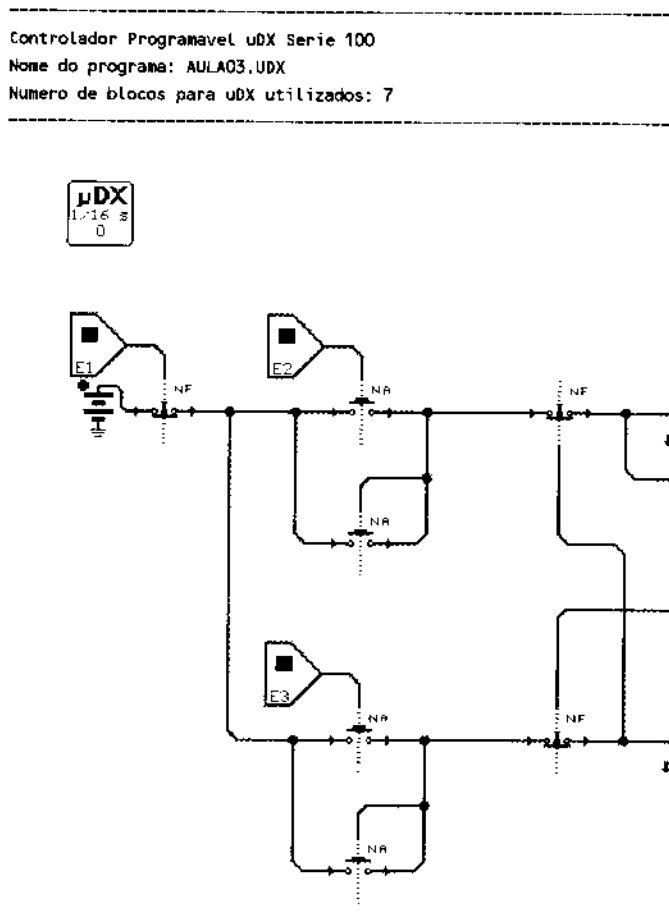


O mesmo circuito implementado em linguagem de relés e blocos (Ladder). Note que a entrada E0.0 desliga a saída S1.0, e a entrada E0.1 liga esta saída.

### Chave de partida direta reversora

Abaixo podemos observar a chave de partida direta com reversão do motor, implementada no controlador  $\mu$ DX Série 100. Note que E1 desliga o motor, E2 liga o motor em um sentido e E3 liga o motor no outro sentido.

As chaves NF nas saídas S1 e S2 impedem que as duas saídas sejam acionadas simultaneamente. Foram desenhadas 3 maneiras de ligar o nodo de controle destas chaves. A primeira maneira é com “fios” de conexão. Outra maneira é via rótulos, e a última é com indicação de rótulo para nodo de controle diretamente na chave NF.



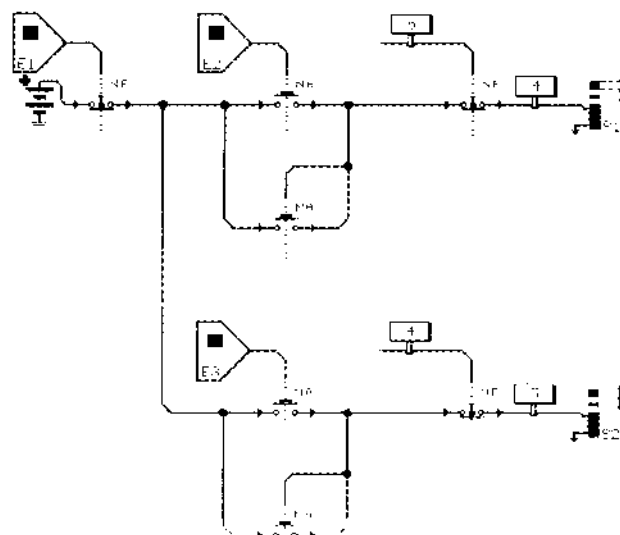
---

Controlador Programável uDX Serie 100

Nome do programa: AULA04.UDX

Numero de blocos para uDX utilizados: 7

---



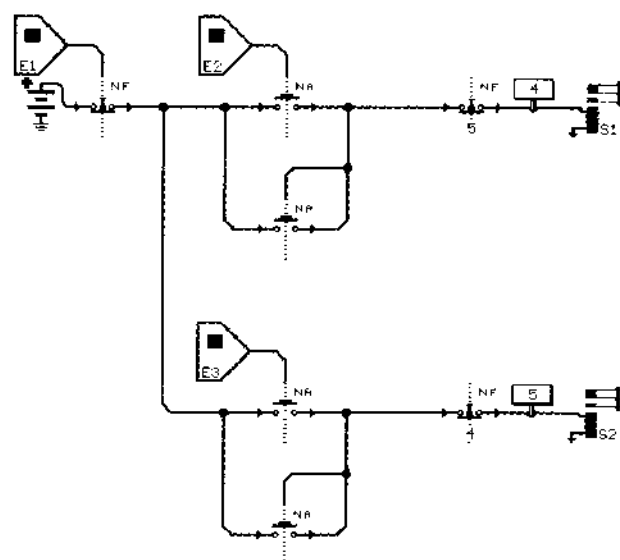
---

Controlador Programável uDX Serie 100

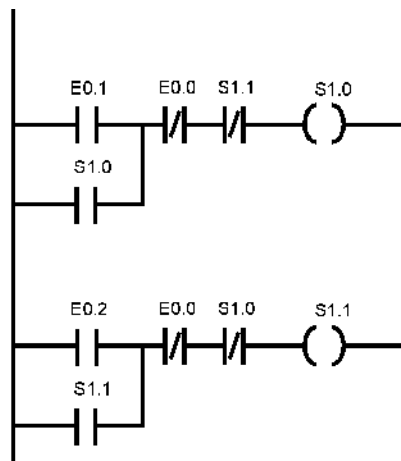
Nome do programa: AULA16.UDX

Numero de blocos para uDX utilizados: 7

---



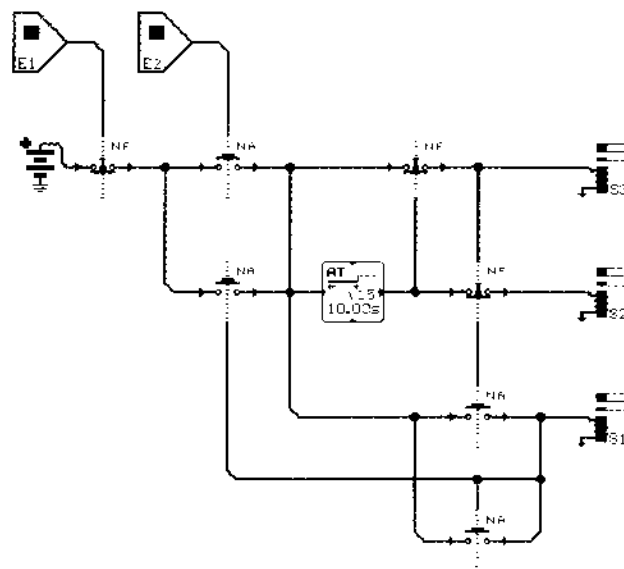
O circuito de partida direta com reversão do motor é mostrado abaixo, implementado em Ladder. As entradas E0.1 e E0.2 ligam o motor e a entrada E0.0 o desliga.



### Chave de partida Estrela-Triângulo

O desenho abaixo representa o problema de partida de motor estrela-triângulo, implementado em controlador  $\mu$ DX. E1 desliga o motor e E2 liga o motor inicialmente em configuração estrela (S1 e S3 fechados) e, após 10 segundos, comuta para configuração triângulo (S1 e S2 fechados).

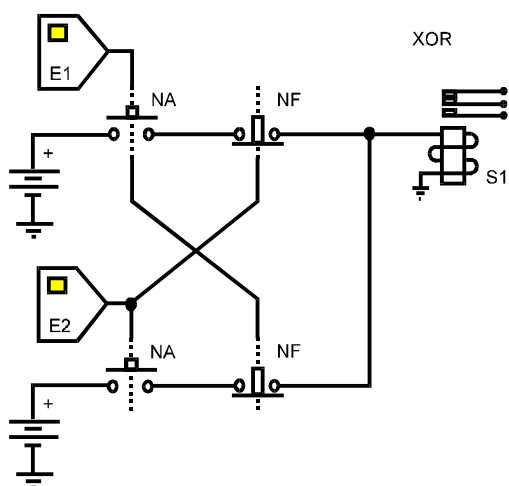
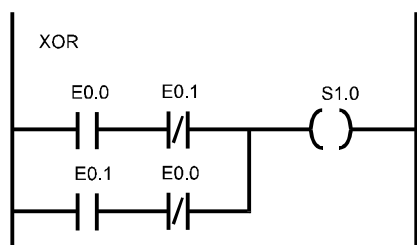
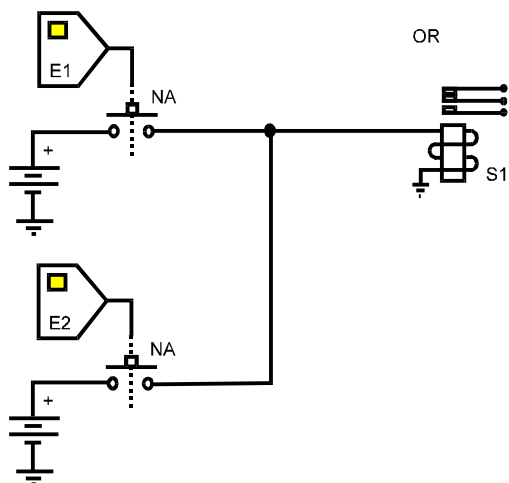
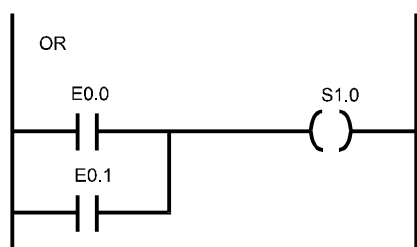
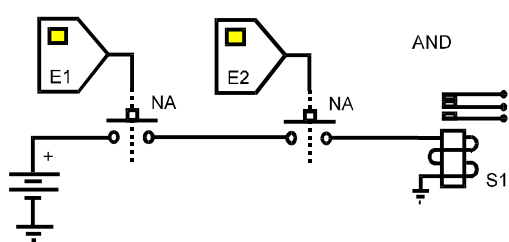
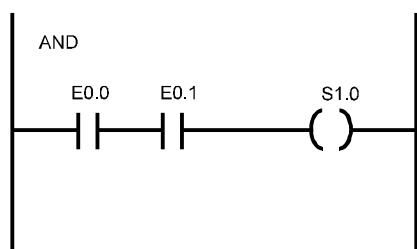
Controlador Programavel  $\mu$ DX Serie 100  
 Nome do programa: AULA10.UDX  
 Numero de blocos para  $\mu$ DX utilizados: 8





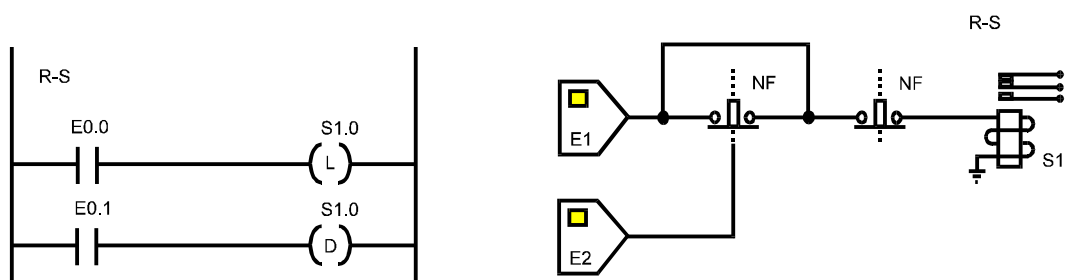
## Revisão de Instruções Booleanas

Entradas	AND	NAND	OR	NOR	XOR	NXOR
0 0	0	1	0	1	0	1
0 1	0	1	1	0	1	0
1 0	0	1	1	0	1	0
1 1	1	0	1	0	0	1



### Revisão de flip-flop tipo R-S

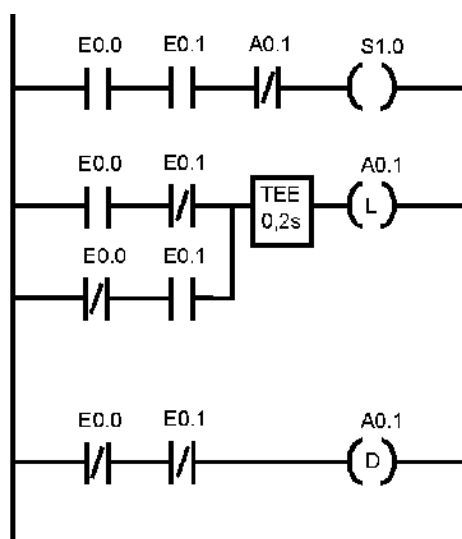
Entrada R	Entrada S	Saída
0	0	Inalterada
0	1	1
1	0	0
1	1	Indefinida



**Nota:** Na verdade, existem muitas formas de implementar um flip-flop tipo R-S em um CLP. Por exemplo, a chave de partida direta já estudada comporta-se como um flip-flop tipo R-S. Os circuitos mostrados acima são os utilizados no exemplo a seguir (segurança em prensa). No caso do flip-flop R-S implementado em diagrama esquemático, foi inserida uma chave NF com nodo de controle aberto para isolar entrada e saída do controlador  $\mu$ DX (entradas e saídas não podem ser conectadas diretamente). Lembre-se que as chaves são unidirecionais!

### Segurança em prensa

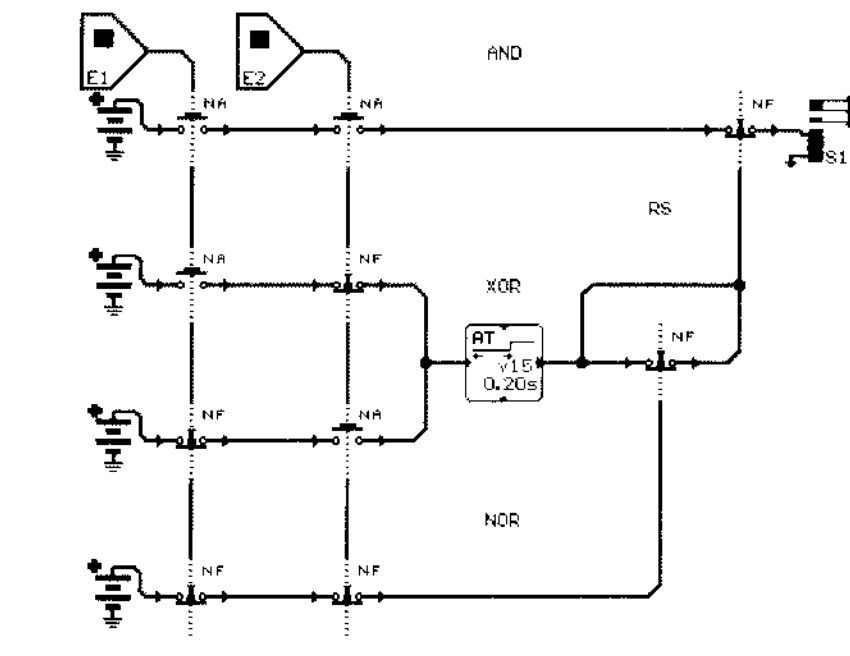
Uma prensa automática deve ativar somente se dois botões, devidamente afastados, forem pressionados praticamente ao mesmo tempo (dentro de 0,2 segundos). Se um dos botões for acionado e o outro depois deste tempo estipulado, o cilindro da prensa não deve avançar. Isso impede que o operador fixe um dos comandos manuais na posição acionada e trabalhe daí em diante apenas com o outro. Desacionando qualquer um dos botões, o cilindro deve recuar imediatamente. Abaixo, o programa implementado em Ladder.



Controlador Programável uDX Serie 100

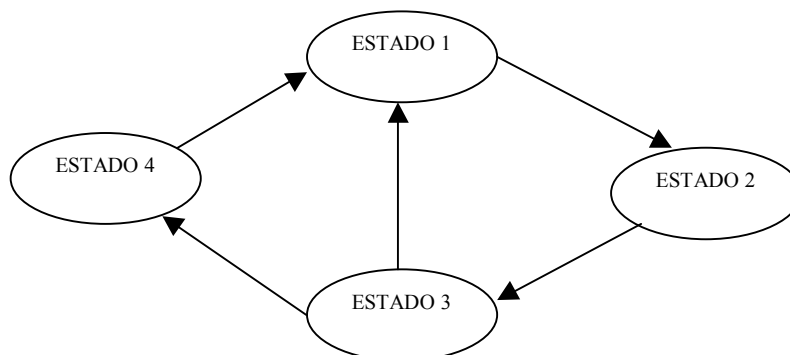
Nome do programa: AULA11.UDX

Numero de blocos para uDX utilizados: 11



### Revisão de Diagrama de Estados

Uma forma muito poderosa de representação de um processo cíclico é através de diagrama de estados ou “máquina” de estados. Deve-se identificar os estados do processo a ser automatizado e, a seguir, determinar quais as condições para que o controlador passe de um estado para outro. Com isso, gera-se um diagrama no qual as elipses representam os estados e as setas indicam as condições para passar de um estado para outro. Os exemplos a seguir irão esclarecer melhor o uso desta técnica.



### Controle de reservatório para processo industrial

Digamos que devemos controlar um reservatório, no qual temos sensor de nível inferior, sensor de nível superior, válvula para entrada de água, válvula de drenagem, botoeira para operador, botoeira de pausa, sinal luminoso para operador, bomba para drenagem e agitador mecânico. O processo é o seguinte: Inicialmente o reservatório está vazio e, portanto, o sensor de nível alto está desativado e o sensor de nível baixo ativado. Caso a botoeira de pausa não esteja acionada devemos permitir a entrada de água (acionando a válvula de entrada de água), até que o sensor de nível alto energize. Desliga-se a válvula de entrada de água e liga-se o aviso luminoso para o operador. O operador irá diluir algum produto na água do tanque (hipoteticamente) e pressionará uma botoeira que avisa ao  $\mu$ DX que a diluição está completa. A seguir o  $\mu$ DX irá acionar o agitador mecânico durante 4 horas, de forma a homogeneizar a mistura. Após estas 4 horas o  $\mu$ DX desliga o agitador e espera outras 4 horas para decantar o material não solúvel. Por fim, é aberta a válvula de drenagem e acionada a bomba, de forma a esvaziar o reservatório. Ao atingir o nível inferior a bomba é desligada, a válvula de drenagem fechada e retorna-se ao estado inicial, ou seja, se a botoeira de pausa não estiver acionada o processo se reinicia, com a entrada de água no tanque. Este exemplo é interessante para realçar a importância de, em processos mais complexos, fazer um diagrama de estados. Assim, neste processo pode-se identificar 6 estados:

<i>Estado 0</i>	<i>Tanque Parado</i>	<i>Nenhum acionamento ligado.</i>
<i>Estado 1</i>	<i>Entrada de água</i>	<i>Apenas válvula de água ligada.</i>
<i>Estado 2</i>	<i>Avisa operador</i>	<i>Aviso luminoso acionado.</i>
<i>Estado 3</i>	<i>Agitação</i>	<i>Agitador acionado.</i>
<i>Estado 4</i>	<i>Decantação</i>	<i>Nenhum acionamento ligado.</i>
<i>Estado 5</i>	<i>Descarga</i>	<i>Válvula de saída e bomba ligadas.</i>

As condições para trocar de estado são as seguintes:

<i>Estado 0 para Estado 1</i>	<i>Chave nível baixo tanque acionada.</i> <i>Botoeira de pausa não acionada.</i>
<i>Estado 1 para Estado 2</i>	<i>Chave nível alto tanque acionada.</i>
<i>Estado 2 para Estado 3</i>	<i>Botoeira de operador ciente acionada.</i>
<i>Estado 3 para Estado 4</i>	<i>Passaram 4 horas desde início estado 3.</i>
<i>Estado 4 para Estado 5</i>	<i>Passaram 4 horas desde início estado 4.</i>
<i>Estado 5 para Estado 0</i>	<i>Chave nível baixo tanque acionada.</i>

Ou seja, temos uma máquina de estados com todos os estados definidos. Com base nisso é relativamente fácil elaborar o programa para o  $\mu$ DX. No programa RESERV.UDX (listado a seguir), usei a variável *v0* para criar a máquina de estados. Basicamente esta variável vai assumindo os valores de 0 a 5, conforme o estado atual. A mudança de estado é determinada pelas condições descritas acima. Esta maneira de resolver problemas de automação (via máquina de estados) é muito poderosa, pois mesmo processos muito complexos podem ser tratados de forma sistemática. Além disso, o translado para o  $\mu$ DX é praticamente direto. Basta eleger uma variável que definirá o estado corrente e colocar as condições para a mudança de seu valor. Agora falta definir as entradas e saídas necessárias. Os sinais de entrada e saída são:

<i>Entradas:</i>	<i>Sensor Inferior</i>	<i>E1</i>
	<i>Sensor Superior</i>	<i>E2</i>
	<i>Botoeira Operador</i>	<i>E3</i>
	<i>Botoeira Pausa</i>	<i>E4</i>
<i>Saídas:</i>	<i>Válvula d'água</i>	<i>S1</i>
	<i>Sinal Luminoso</i>	<i>S2</i>
	<i>Válvula Saída e Bomba de Drenagem</i>	<i>S3</i>
	<i>Agitador Mecânico</i>	<i>S4</i>

Note que a válvula de saída e a bomba de drenagem são acionadas simultaneamente pela saída S3. Examine cuidadosamente o programa RESERV.UDX. Foram inseridos comentários para ajudar...

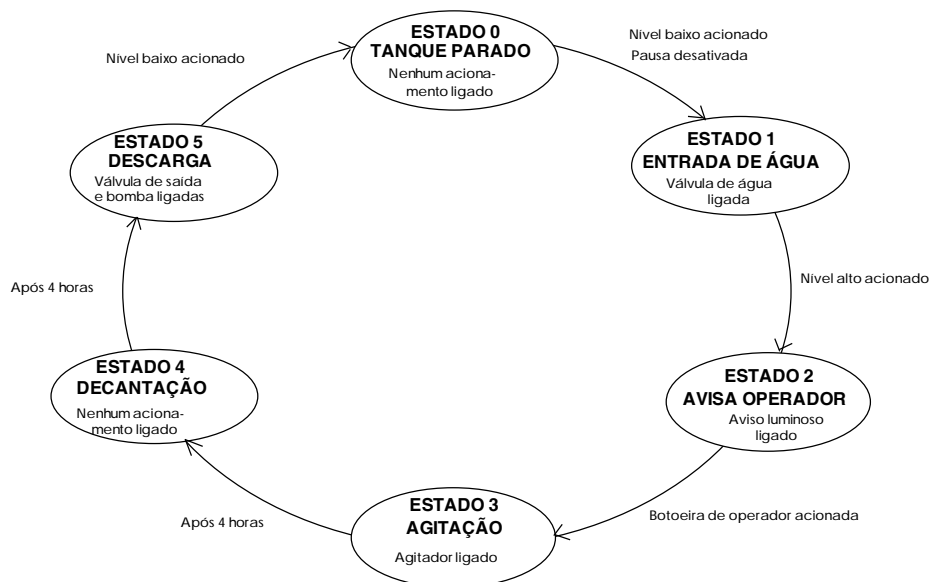


DIAGRAMA DE ESTADOS PARA RESERV.UDX

Agora digamos que, ao instalar o sistema, surgiu a necessidade de abortar o ciclo de decantação via comando do operador. Ou seja, se o operador quiser, ele pode encerrar o ciclo de decantação antes de concluir as 4 horas programadas. Para isso vou utilizar a própria botoeira de operador (E3). Se ela for acionada durante o estado 4 (decantação) o controlador pula para o estado 5 (descarga). O diagrama de estados fica:

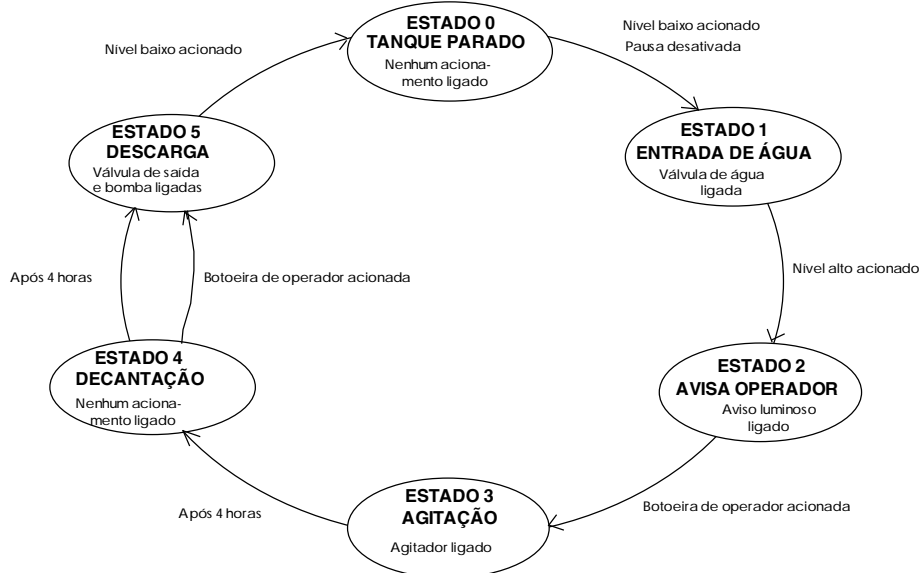
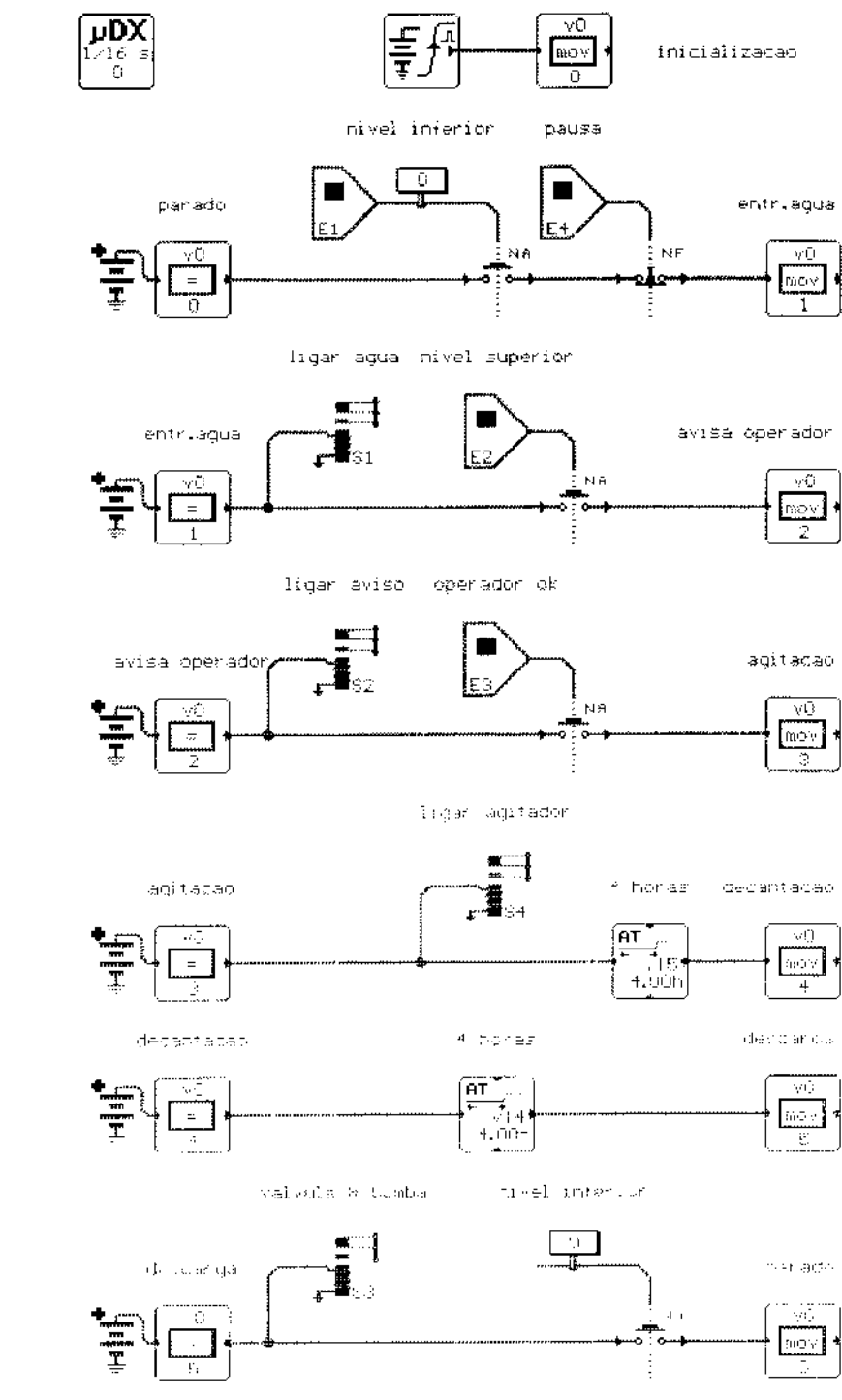


DIAGRAMA DE ESTADOS PARA RESERV1.UDX

Controlador Programável uDX Serie 100

Nome do programa: RESERV.UDX

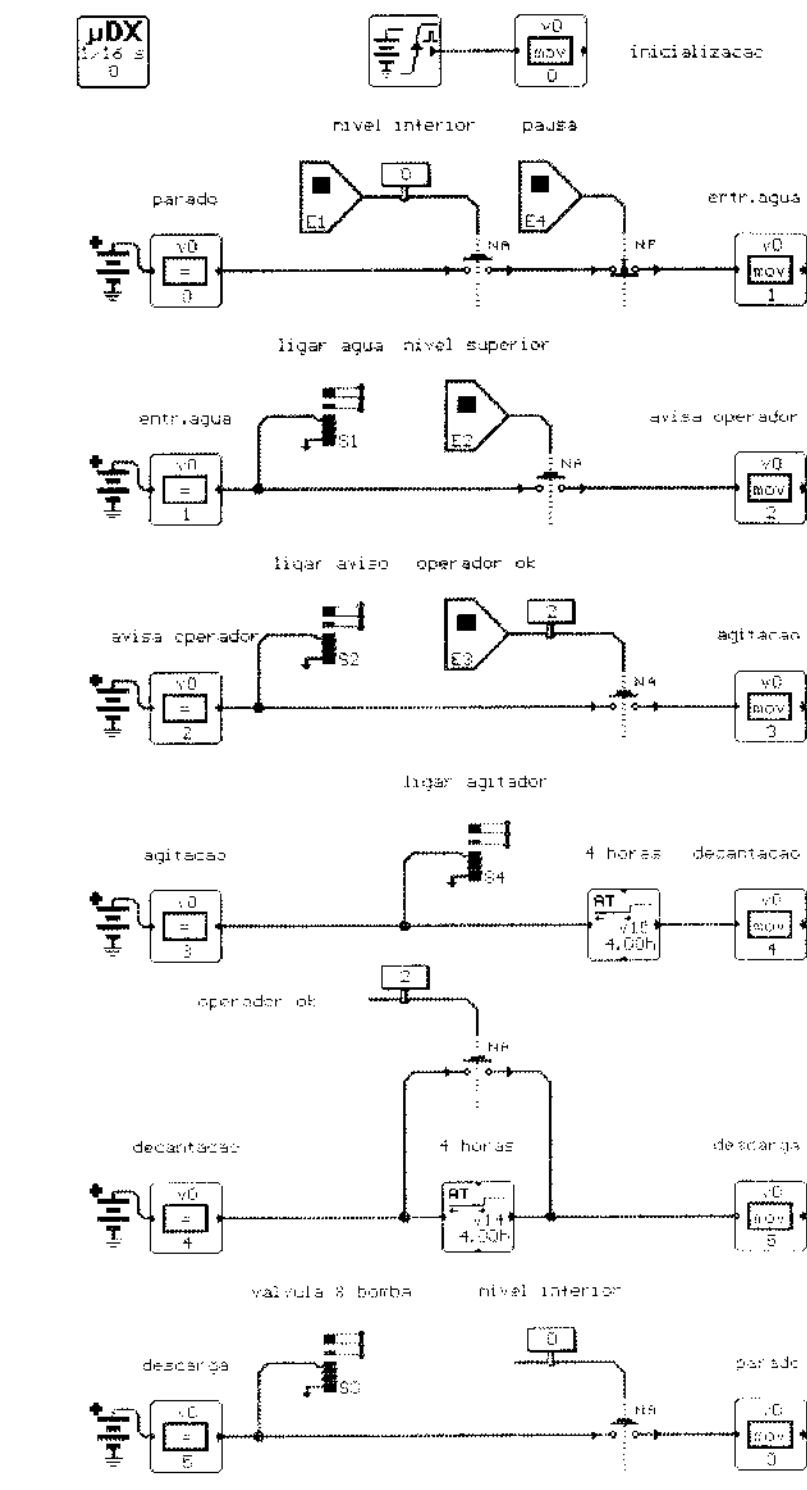
Numero de blocos para uDX utilizados: 20



Controlador Programável uDX Serie 100

Nome do programa: RESERV1.UDX

Numero de blocos para uDX utilizados: 21



### Prensa Industrial

Outro exemplo de programação via máquina de estados. Ao pressionar uma tecla ligada à entrada E4, a saída S1 (responsável pela pressurização do sistema hidráulico ou pneumático) é acionada. Esta entrada E4 serve como chave liga/desliga para a prensa. O pistão da prensa deve avançar (S2 - rótulo 5) até o sensor fim de curso ligado à E2 (rótulo 1). A seguir o pistão deve recuar (S3 - rótulo 6) até o fim de curso E1 (rótulo 0). Este movimento deve ser repetido 5 vezes (estado 1 e estado 2 da máquina de estados).

A seguir o  $\mu$ DX pula para o estado 3, que continua recuando o pistão via saída S3, mas temporiza 1 segundo antes de passar para o estado 4, que aciona a saída S4, que abre a tampa do compartimento da prensa (para expulsão do material prensado). Após dois segundos a tampa já está aberta e o pistão é acionado até o fim de curso E3. Note que a tampa permanece aberta via saída S4 (rótulo 7).

Depois de atingir E3, o  $\mu$ DX pula para estado 6 ( $v1=6$ ), que faz o pistão recuar até o fim de curso E2 (rótulo 1), mantendo a porta aberta. Uma vez atingido E2, a porta é fechada e o pistão continua recuando até atingir o fim de curso E1 (rótulo 0). Quando o pistão atinge E1, o sistema retorna ao estado 0 e, caso a entrada E4 esteja acionada, inicia novo ciclo de prensagem. Assim, neste processo pode-se identificar 7 estados:

<i>Estado 0</i>	<i>Prensa Parada</i>	<i>Nenhum acionamento ligado.</i>
<i>Estado 1</i>	<i>Prensa Avançando</i>	<i>Apenas avanço acionado.</i>
<i>Estado 2</i>	<i>Prensa Recuando</i>	<i>Apenas recuo acionado.</i>
<i>Estado 3</i>	<i>Recuo p/abrir Porta</i>	<i>Recuo acionado durante 1 seg.</i>
<i>Estado 4</i>	<i>Abertura de Porta</i>	<i>Abertura de porta acionado.</i>
<i>Estado 5</i>	<i>Descarga</i>	<i>Avanço e abertura de porta acionados.</i>
<i>Estado 6</i>	<i>Recuo p/fechar Porta</i>	<i>Abertura de porta e recuo acionados.</i>
<i>Estado 7</i>	<i>Fechamento de Porta</i>	<i>Recuo acionado.</i>

As condições para trocar de estado são as seguintes:

<i>Estado 0 para Estado 1</i>	<i>Chave liga/desliga E4 acionada.</i>
<i>Estado 1 para Estado 2</i>	<i>Fim de curso E2 acionado.</i>
<i>Estado 2 para Estado 3</i>	<i>Contador de ciclos (<math>v0</math>) &lt; 5.</i>
<i>Estado 3 para Estado 4</i>	<i>Fim de curso E2 acionado.</i>
<i>Estado 4 para Estado 5</i>	<i>Contador de ciclos (<math>v0</math>) = 5.</i>
<i>Estado 5 para Estado 6</i>	<i>Fim de curso E1 acionado.</i>
<i>Estado 6 para Estado 7</i>	<i>Passou 1 segundo desde início do estado 3.</i>
<i>Estado 7 para Estado 0</i>	<i>Passou 2 segundos desde início do estado 4.</i>
<i>Estado 0 para Estado 1</i>	<i>Fim de curso E3 acionado.</i>
<i>Estado 1 para Estado 2</i>	<i>Fim de curso E2 acionado.</i>
<i>Estado 2 para Estado 3</i>	<i>Fim de curso E1 acionado.</i>

Agora falta definir as entradas e saídas necessárias. Os sinais de entrada e saída são:

<i>Entradas:</i>	<i>Fim de curso pistão recolhido</i>	<i>E1</i>
	<i>Fim de curso pistão prensando</i>	<i>E2</i>
	<i>Fim de curso pistão expulsando material</i>	<i>E3</i>
	<i>Chave liga/desliga</i>	<i>E4</i>
<i>Saídas:</i>	<i>Pressurização do sistema</i>	<i>S1</i>
	<i>Avança pistão</i>	<i>S2</i>
	<i>Recua pistão</i>	<i>S3</i>
	<i>Abertura de tampa</i>	<i>S4</i>



Note que foram utilizadas duas variáveis neste programa (além das variáveis utilizadas nos blocos de temporização):

*v0*                      Contador de número de ciclos de prensagem.  
*v1*                      Indica estado da máquina de estados da prensa.

A variável *v0* é incrementada cada vez que a prensa entra no estado 1. Para isso foi utilizado um bloco de Pulso, pois este gera um pulso com duração de apenas um ciclo de execução do  $\mu$ DX, e assim a variável é incrementada somente uma vez.

O diagrama de estados para este exemplo fica:

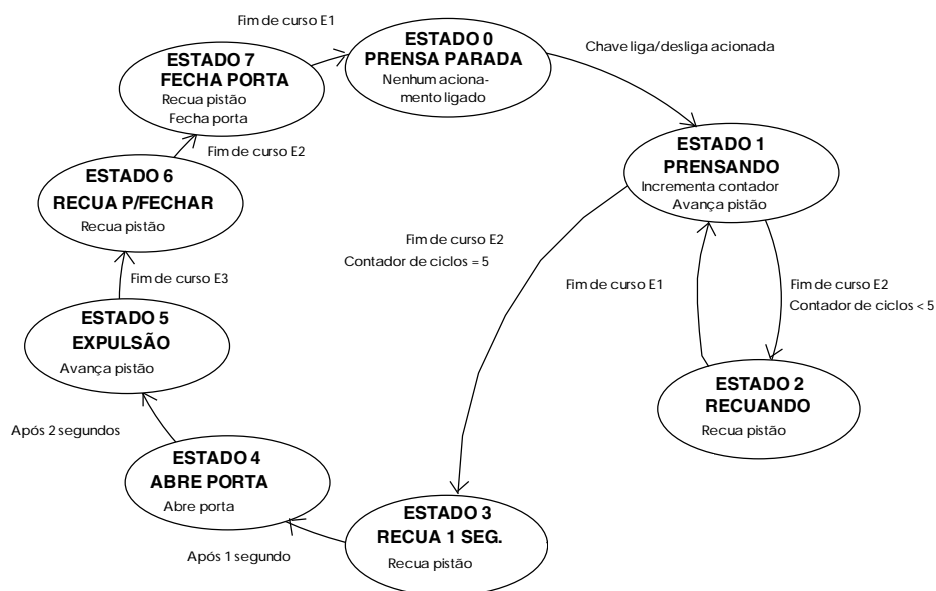
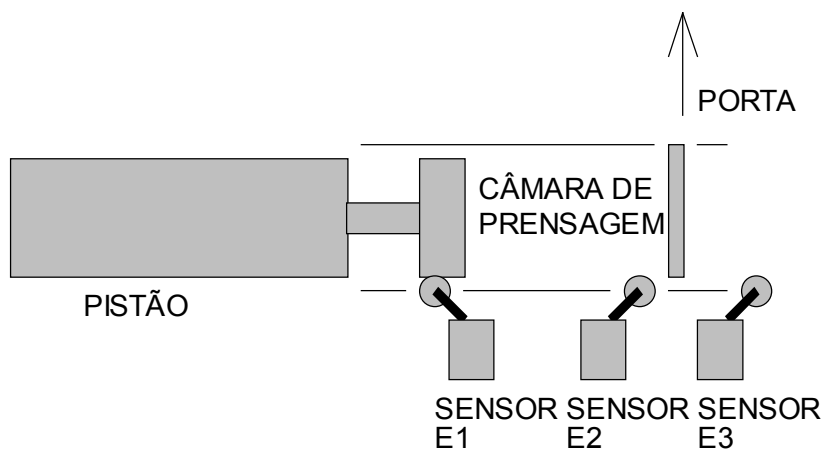


DIAGRAMA DE ESTADOS PARA PRENSA.UDX

O esquema de sensores e acionamentos da prensa é o seguinte:

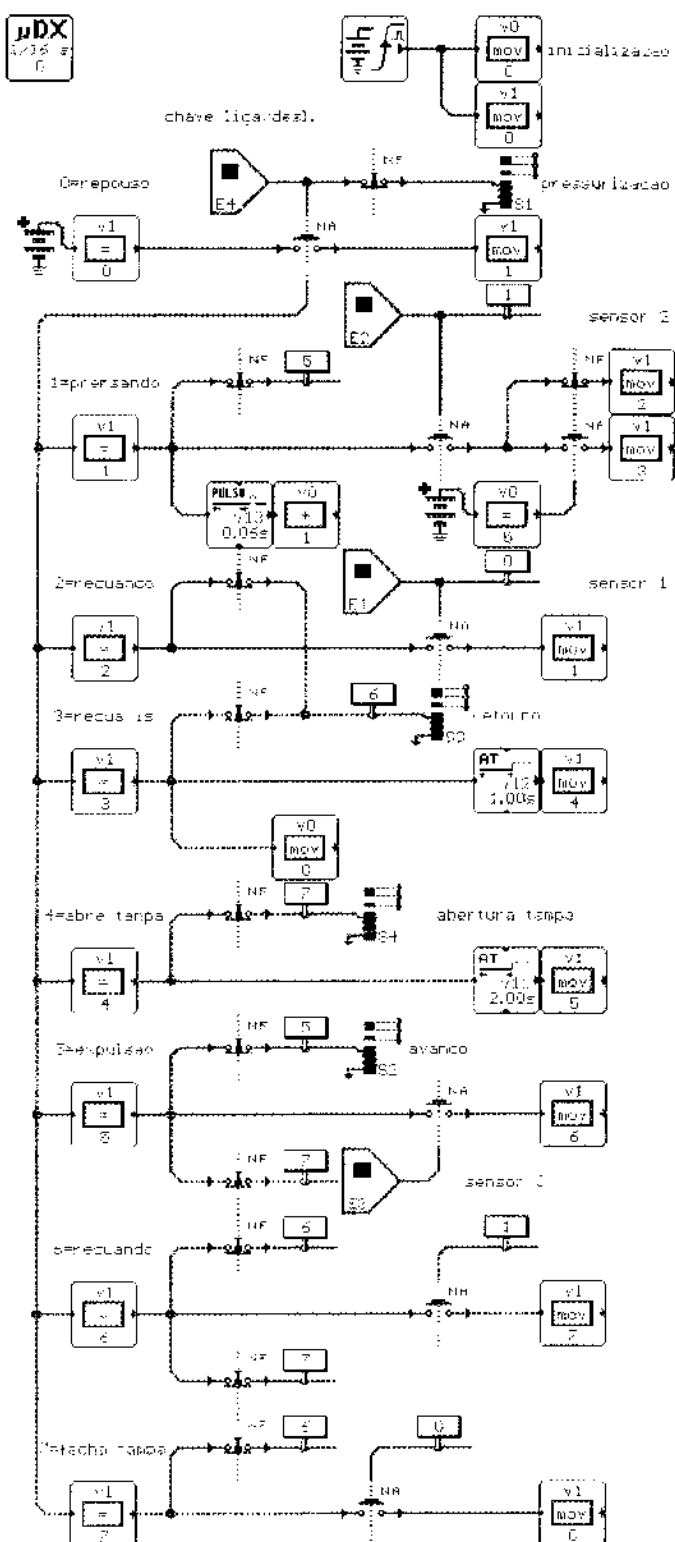


Tanto a variável *v0* (contador de ciclos de prensagem) como *v1* (variável de estado da máquina de estados) são inicializados pelo bloco NodoEL no início do programa. Além disso, são utilizadas chaves NF em vários pontos do programa para isolar nodos, evitando que um estado interfira no outro.

Controlador Programável UDX Serie 100

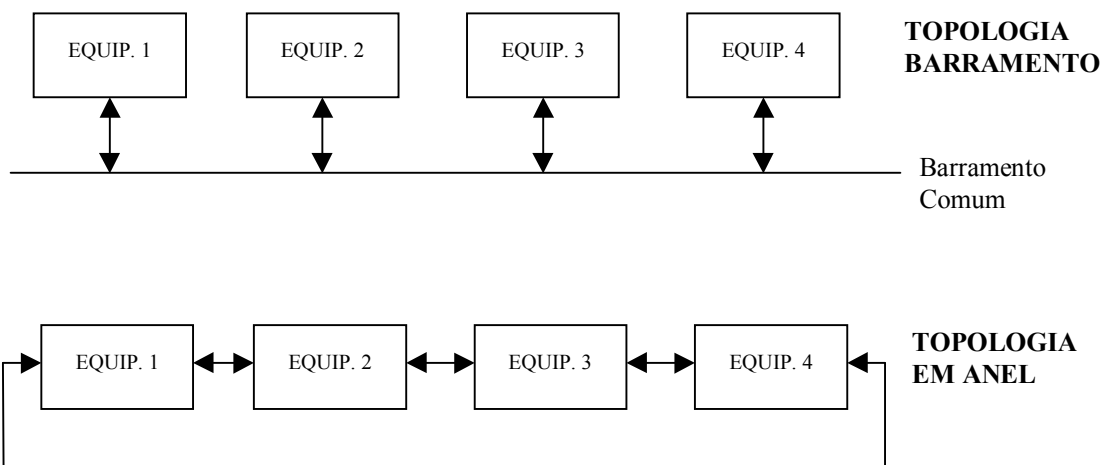
Nome do programa: PRENSA.UDX

Numero de blocos para uDX utilizados: 43



## Redes de Comunicação

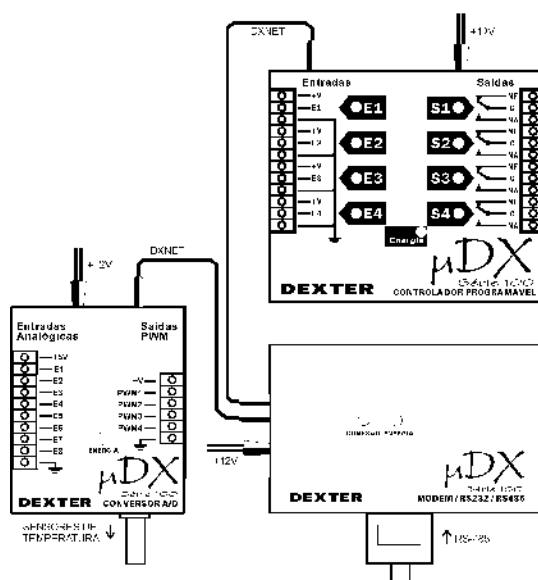
As redes de comunicação surgiram para permitir troca de dados entre controladores programáveis e também para sua conexão a computadores. Podem ser do tipo mestre-escravo - apenas um equipamento pode iniciar comunicação ou multi-mestre - qualquer dos equipamentos ligados a rede pode iniciar comunicação. Neste caso, é necessário prever estratégias para evitar conflitos (dois dispositivos usando simultaneamente a rede). Em termos de topologia, podem ser tipo barramento – todos os equipamentos ligados a mesma linha física, ou tipo anel – a conexão entre equipamentos é feita um a um. No caso de anel, este pode ser fechado (o último dispositivo liga no primeiro) ou aberto. Existem ainda outras topologias, não abordadas neste documento.



## Rede Local DXNET

No caso do Controlador  $\mu$ DX, a rede local DXNET permite a conexão de 15 controladores  $\mu$ DX ou periféricos e ainda um microcomputador IBM-PC compatível. A rede local é multi-mestre com topologia tipo barramento, ou seja, todos os dispositivos ligados à rede podem iniciar comunicação e recebem simultaneamente as mensagens. Além de controladores, a rede DXNET aceita periféricos para o  $\mu$ DX, como Conversor A/D, Modem e Interface Homem/Máquina (IHM). O protocolo de comunicação utiliza apenas um sinal + referência (clock e dados no mesmo fio). A rede pode ser estendida até 100 metros. Abaixo, um exemplo de conexão de Controlador  $\mu$ DX, Conversor A/D e Modem.

CONEXÃO DE  $\mu$ DX E PERIFÉRICOS À REDE DXNET



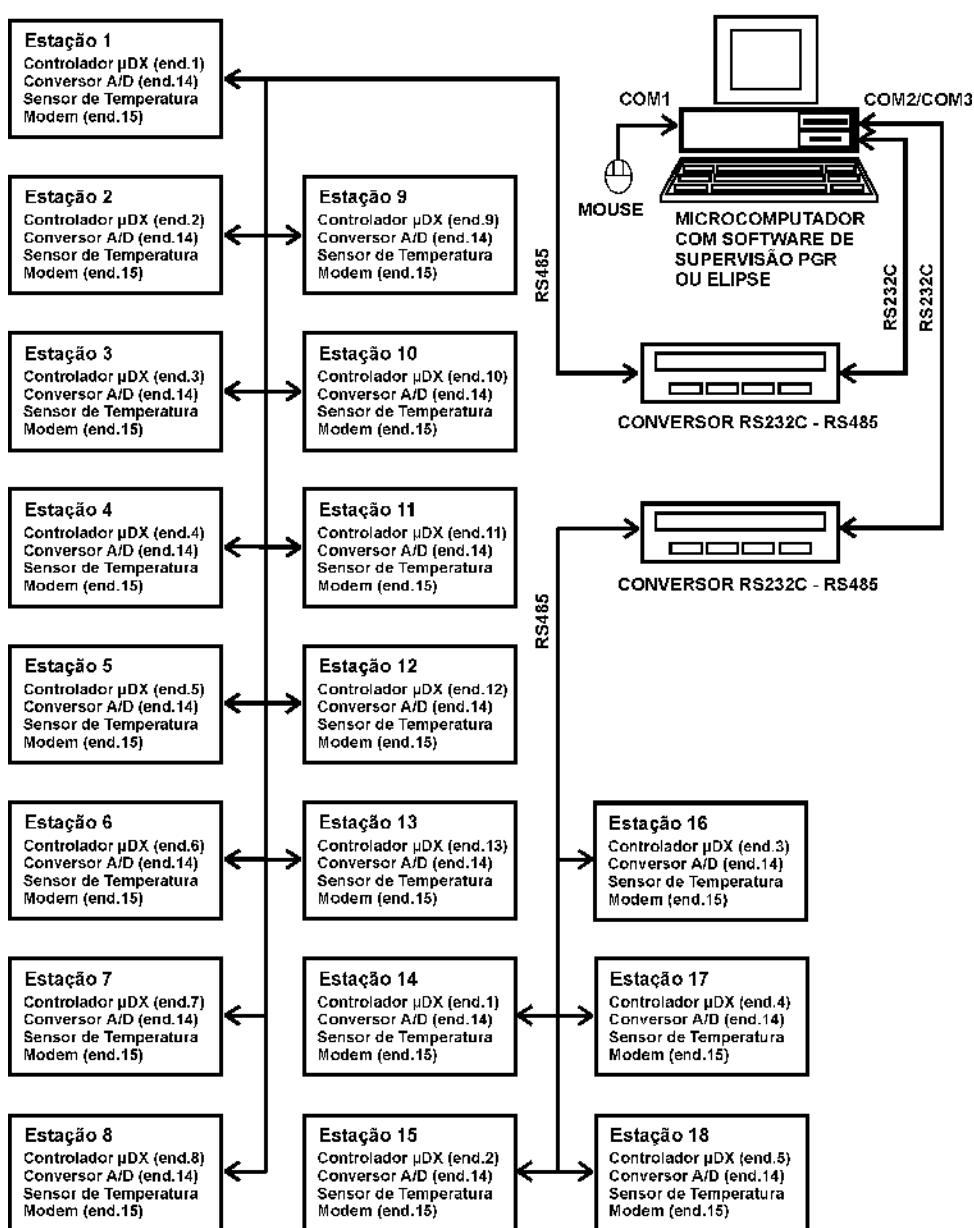
### Rede RS-485

Caso sejam necessárias maiores distâncias, pode-se usar o periférico Modem para Controlador  $\mu$ DX. Este equipamento permite conectar os controladores via conexão serial RS-232C ou RS-485 (ou ainda rádio-transmissor e rede telefônica, opções abordadas a seguir).

A conexão serial RS-232C é muito utilizada em microcomputadores IBM-PC compatíveis, para conexão entre computadores ou instalação de periféricos, como o mouse. Trata-se de conexão ponto a ponto, ou seja, permite a conexão de apenas dois equipamentos entre si. Permite conexões confiáveis até cerca de 500 metros, em taxas de transmissão lentas (300 bps).

A conexão serial RS-485, ao contrário da RS-232C, permite ligar diversos equipamentos entre si (topologia barramento), além de ser confiável para distâncias superiores a 5 Km, em taxas de transmissão lentas (300 bps).

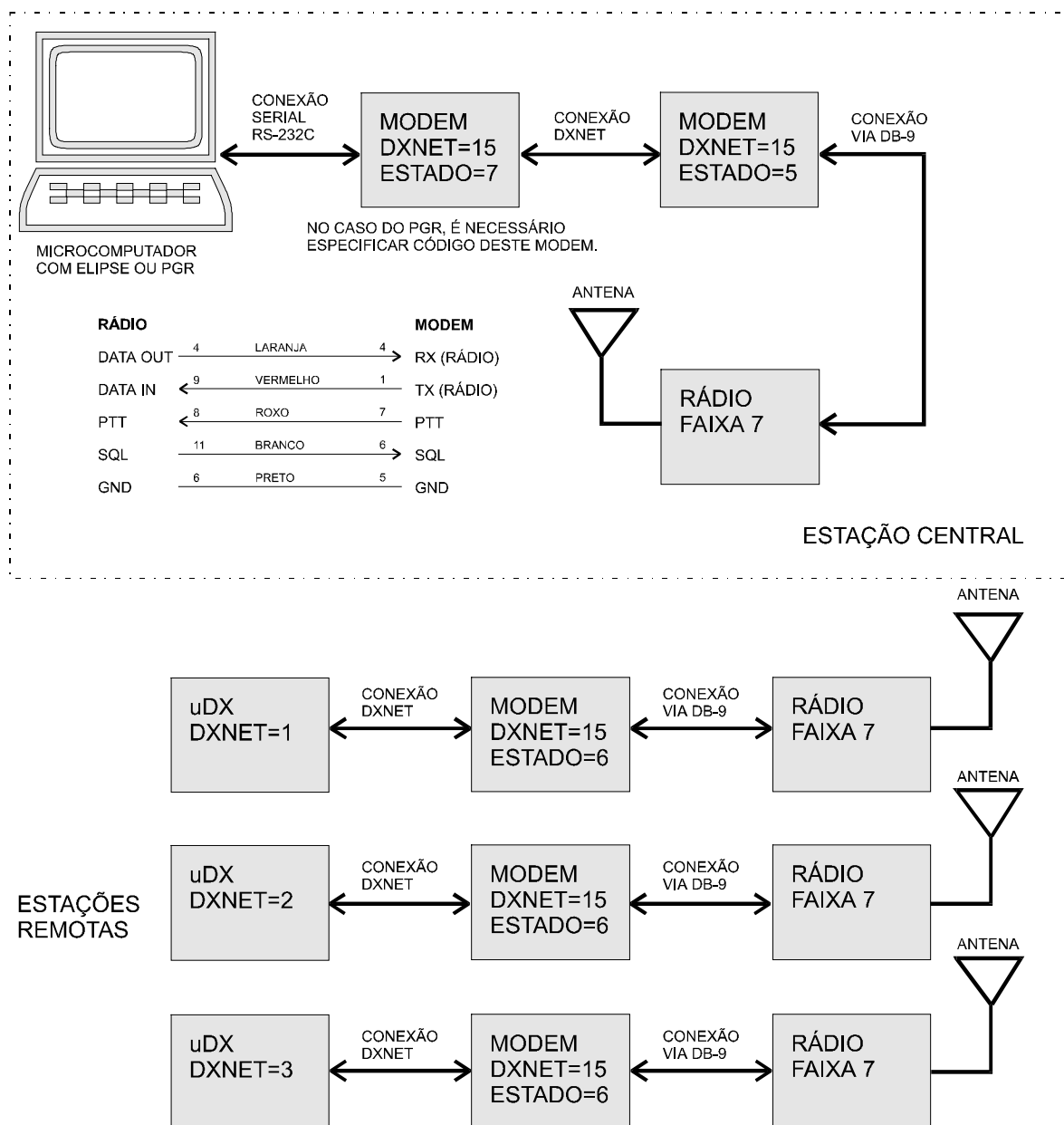
#### CONEXÃO DAS ESTAÇÕES + MICROCOMPUTADOR Projeto p/supervisão predial via rede RS-485



### Rádio Transmissão

E caso sejam necessárias distâncias ainda maiores, ou não existam linhas físicas para conexão dos equipamentos? Neste caso, uma solução pode ser a rádio-transmissão dos dados. O controlador  $\mu$ DX pode ser conectado a outros controladores via rádio-transmissor, controlado pelo Modem para Controlador  $\mu$ DX. As distâncias, neste caso, podem ultrapassar 50 Km. Na figura abaixo é ilustrada a conexão entre três estações remotas e uma estação central.

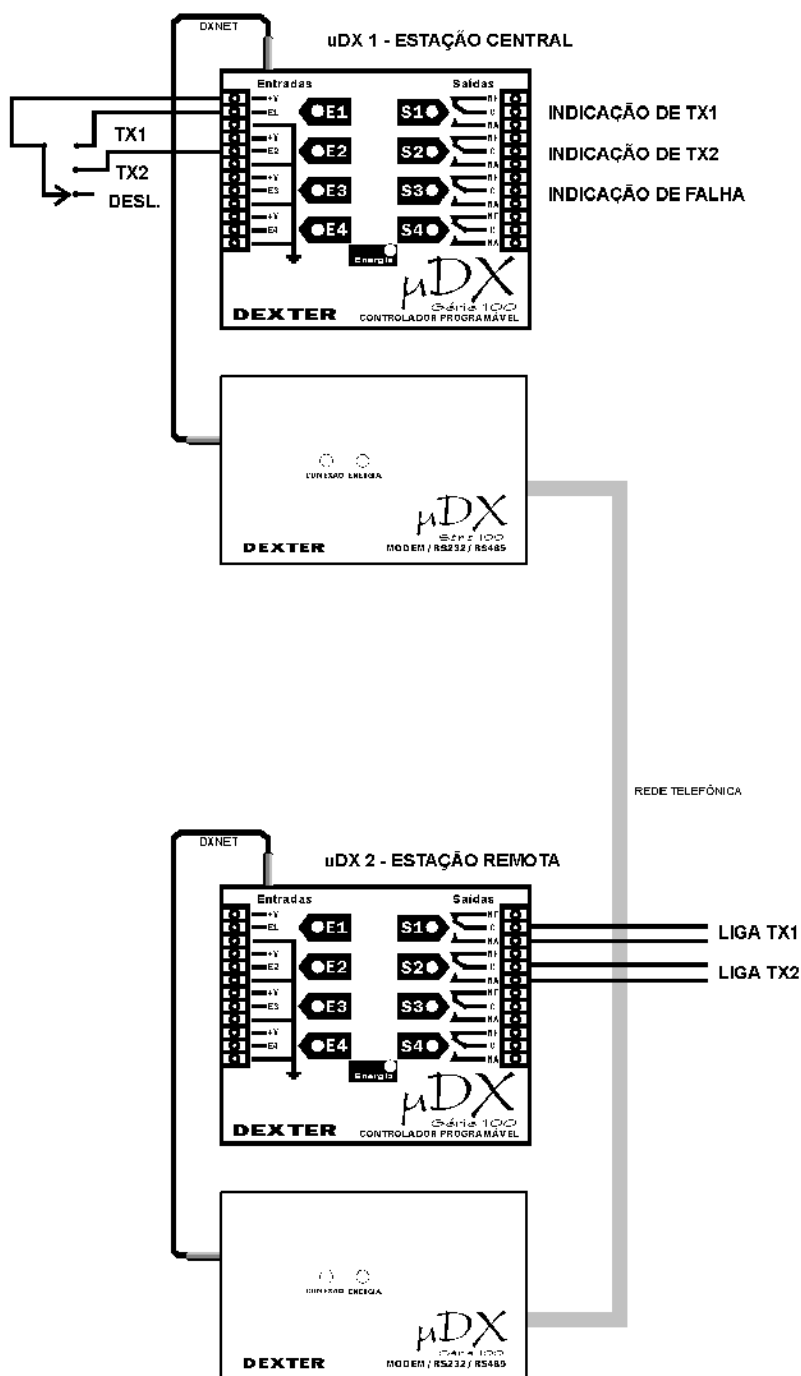
## CONEXÃO DOS MODEMS REMOTOS VIA RÁDIO



### Rede Telefônica

Por fim, a conexão entre os controladores ou entre controladores e uma estação central pode ser efetuada pela rede telefônica. Neste caso, a abrangência da rede é mundial! O Modem para Controlador  $\mu$ DX permite armazenar 16 números telefônicos, cada um com até 16 dígitos, o que permite inserir código de país e de área. Abaixo um exemplo de conexão de dois controladores  $\mu$ DX via rede telefônica. No exemplo abaixo, o controlador  $\mu$ DX da estação central comanda dois relés da estação remota, e indica o estado dos relés da estação remota nas saídas S1 e S2. A saída S3 indica erro na comunicação.

#### SISTEMA DE TELECOMANDO E TELESSINALIZAÇÃO VIA REDE TELEFÔNICA



## Programas Supervisórios

O programa PG - Programador Gráfico – permite programar o controlador  $\mu$ DX e efetuar monitoração e simulação de programas, sendo uma eficiente ferramenta para desenvolvimento e teste dos programas aplicativos elaborados para o  $\mu$ DX. Permite inclusive descarregar programas e monitorá-los remotamente, via rede telefônica.

Entretanto, quando se trata de obter dados do processo controlado o programa PG não é a ferramenta mais adequada. Isso porque o enfoque do PG é no programa aplicativo, e não no processo. Assim, variáveis de processo não podem ser convertidas de seu valor binário para a respectiva grandeza que representam via PG. Isso dificulta muito a visualização destas grandezas. Não é possível associar a determinadas variáveis níveis de alarme, nem visualizar o processo em si, independentemente do programa aplicativo desenvolvido para controlá-lo. Para isso, é necessário o uso de um programa supervisório. A Dexter comercializa o programa PGR – Programa de Gerenciamento Remoto, capaz de monitorar o processo via rede telefônica, indicando estados de nodos (estados binários), variáveis (grandezas analógicas) e constantes (set points do programa). Permite ainda a associação de níveis de alarme as grandezas, indicando visual e auditivamente condições de perigo.

O software PGR roda em microcomputadores IBM-PC compatíveis, sob Windows 95 ou 98. Basicamente, ele permite criar um arquivo de estações remotas, com código e nome da estação, número do telefone remoto, senha de acesso, disposição e parâmetros dos dados monitorados e tela de fundo. Para cada estação remota estes itens são configuráveis e armazenados no arquivo de estações. Ao conectar uma estação remota (o PGR usa o modem interno do microcomputador), a tela principal do PGR apresenta a tela configurada para a estação escolhida. É possível monitorar e/ou modificar o valor de variáveis, o valor de constantes e o estado de nodos do controlador  $\mu$ DX. Com isso, pode-se atribuir, por exemplo, a uma variável do controlador a temperatura ambiente e, no PGR, monitorar esta temperatura a distância; ligar um nodo, que pode modificar o comportamento do controlador ou acionar diretamente um dispositivo, como um motor; pode-se ainda trocar o valor de uma constante de processo (set point), como a temperatura ambiente desejada em uma instalação.

Abaixo dos nodos, constantes e variáveis, existe a possibilidade de desenhar um esquemático do processo, facilitando a visualização da correspondência entre os nodos e variáveis e os respectivos dispositivos reais da instalação remota. O PGR aceita arquivos para tela de fundo no formato bit map (.BMP), windows metafile (.WMF) ou enhance metafile (.EMF). A figura apresenta a tela do PGR com alguns nodos e variáveis sendo monitorados. No caso de variáveis (grandezas analógicas), além do nome atribuído à variável, é visualizado seu valor (já convertido para a unidade correta, segundo fórmula programada no PGR), tanto numérico quanto por representação em gráfico de barra (bargraph).

