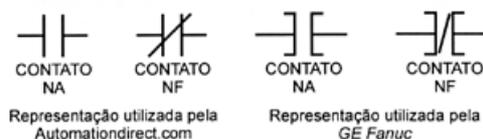


5. Fundamentos da Programação LADDER

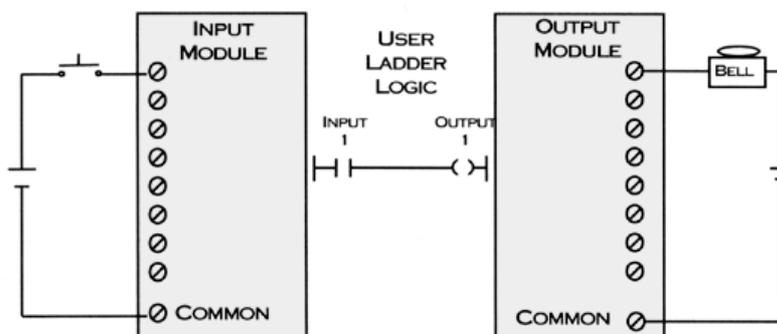
Este capítulo expõe os conceitos básicos da programação ladder. Independente da complexidade do programa de aplicação, há certos fundamentos da linguagem que são imprescindíveis para um desenvolvimento adequado e que são válidos genericamente a todos os PLCs.

DEFINIÇÕES INICIAIS

Mesmo tendo sido a primeira linguagem destinada especificamente à programação de PLCs, a Linguagem Ladder mantém-se ainda como a mais utilizada, estando presente praticamente em todos os PLCs disponíveis no mercado. Por ser uma linguagem gráfica, baseada em símbolos semelhantes aos encontrados nos esquemas elétricos (contatos e bobinas), as possíveis diferenças existentes entre os fabricantes de PLCs, quanto à representação das instruções, são facilmente assimiladas pelos usuários, como exemplificado na figura.

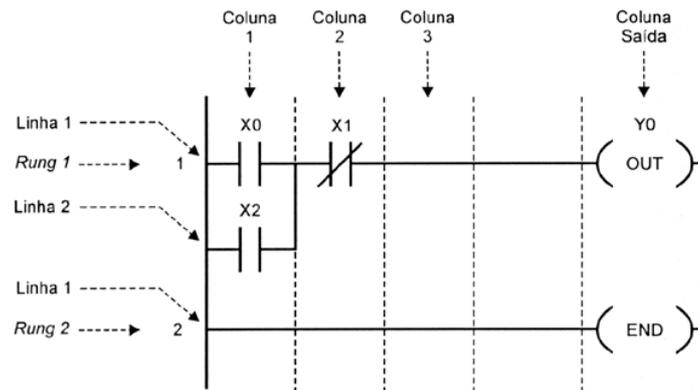


Verifique por exemplo o esquema conceitual de um sistema PLC, mostrado na figura abaixo. As entradas físicas reais estão fixadas a um módulo de entrada (esquerda) enquanto as saídas estão fixadas a um módulo de saída (direita). No centro, vê-se a representação lógica que a CPU deve processar, na linguagem Ladder. Neste caso, se o Input 1 (interruptor normalmente aberto) for fechado, a Output 1 (campainha) é ligada.



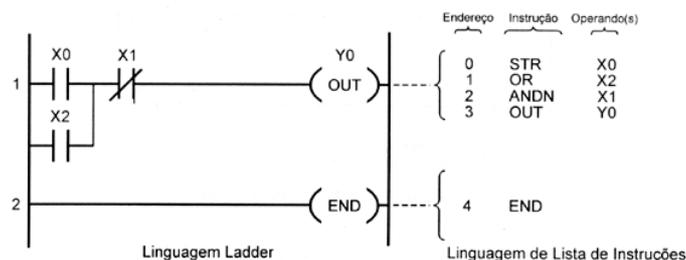
O nome Ladder deve-se à representação da linguagem se parecer com uma escada (ladder), na qual duas barras verticais paralelas são interligadas pela Lógica de Controle, formando os degraus (rungs) da escada. Portanto, a cada Lógica de Controle existente no Programa de Aplicação dá-se o nome de rung, a qual é composta por Colunas e Linhas, conforme apresentado na figura a seguir.

A quantidade de Colunas e Linhas, ou Elementos e Associações, que cada rung pode conter é determinada pelo fabricante do PLC, podendo variar conforme a CPU utilizada. Em geral, este limite não representa uma preocupação ao usuário durante o desenvolvimento do Programa de Aplicação, pois os Softwares de Programação indicam se tal quantidade foi ultrapassada, por meio de erro durante a compilação do Programa de Aplicação.



Cada Elemento (contato ou bobina, por exemplo) da Lógica de Controle representa uma Instrução da Linguagem Ladder sendo alocada em um endereço específico e consumindo uma quantidade determinada de memória (word) disponível para armazenamento do Programa de Aplicação, conforme a CPU utilizada. Um mesmo símbolo gráfico da Linguagem Ladder (Contato Normalmente Aberto, por exemplo) pode representar Instruções diferentes, dependendo da localização na Lógica de Controle.

A figura seguinte apresenta a equivalência entre o Programa de Aplicação em Linguagem Ladder e o mesmo Programa em Linguagem de Lista de Instruções (Linguagem de Máquina - mnemônicos). Como pode ser visto, cada Instrução utilizada na Linguagem Ladder ocupou apenas um endereço de memória, o que é verificado pelo incremento simples de endereço em Linguagem de Lista de Instruções. Porém, há instruções que ocupam mais de um endereço de memória, conforme a CPU utilizada.



A relação entre o símbolo gráfico da Linguagem Ladder e a Instrução a ser executada pode ser verificada nos Endereços 0 e 1 do Programa em Linguagem de Lista de Instruções. Neste caso, a representação em Linguagem Ladder para os Elementos X0 e X2 são Contatos Normalmente Abertos idênticos. Porém, a localização de cada um na Lógica de Controle determina Instruções diferentes, ou seja, o Contato Normalmente Aberto de X0, por iniciar o rung, determina a Instrução 'Store' (STR X0) e o Contato Normalmente Aberto de X2 (com representação gráfica idêntica à de X0), por estar em paralelo com X0, determina a Instrução 'Or' (OR X2). Esta característica da Linguagem Ladder normalmente facilita o desenvolvimento do Programa de Aplicação, uma vez que o usuário precisa certificar-se apenas se a associação desejada é aceita pela CPU utilizada, não se prendendo à Instrução propriamente dita.

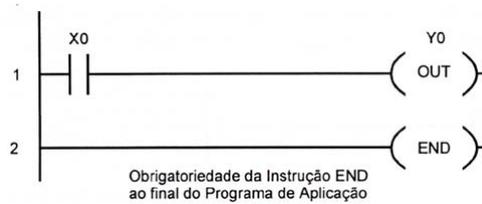
CONCEITOS BÁSICOS DA PROGRAMAÇÃO EM LINGUAGEM LADDER

Os conceitos apresentados em seguida são necessários para o correto desenvolvimento de Programas de Aplicação em Linguagem Ladder Eles são aplicados a todos os PLCs, independente de fabricante e de recursos disponíveis na CPU utilizada.

Instrução END

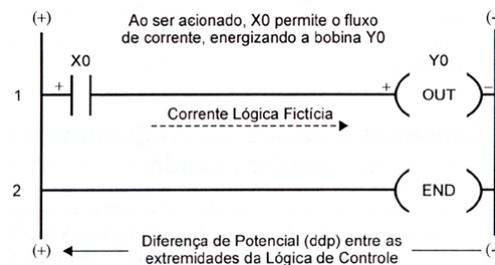
Todo programa em Linguagem Ladder deve ter uma Instrução END, indicando o seu final. Trata-se de uma bobina e é classificada como Instrução de Controle do Programa. É uma Instrução

incondicional, não admitindo qualquer tipo de Elemento em sua Lógica de Controle. Toda Instrução localizada após a Instrução END não será executada pelo Programa de Aplicação, com exceção das Instruções de Interrupção, Sub-Rotinas e Controles Específicos (Mensagens, por exemplo). A não-existência da Instrução END no Programa de Aplicação gera um 'Erro Fatal', fazendo com que a CPU não permaneça em Modo de Execução (RUN).

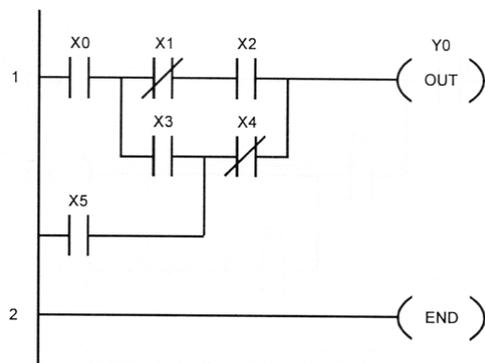


Corrente Lógica Fictícia

Para que uma bobina (ou outro Elemento de Saída - temporizador, contador ou bloco de função, por exemplo) seja acionada (Instrução executada), faz-se necessário “energizá-la logicamente”. Assim, utiliza-se o conceito de Corrente Lógica Fictícia, ou seja, supondo que entre as barras verticais que 'sustentam' toda a Lógica de Controle haja uma diferença de potencial (a barra da esquerda com potencial positivo e a barra da direita com potencial negativo, por exemplo), haverá a circulação de corrente da esquerda para a direita se a Lógica de Controle der condições para tal. A este conceito dá-se o nome de Corrente Lógica Fictícia.



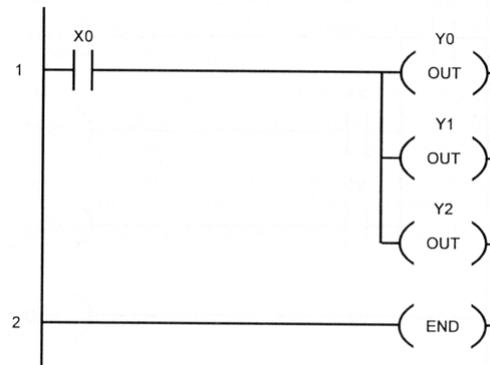
Na figura acima pode-se observar que se o Contato de X0 (Normalmente Aberto) estiver acionado - condição para que haja o fluxo de corrente entre as extremidades da Lógica de Controle, a bobina de Y0 será energizada, atuando tal Saída (por meio da Instrução OUT). Caso contrário, a bobina de Y0 não será energizada (não sendo submetida a uma 'ddp lógica'), mantendo a Saída desligada. O sentido da Corrente Lógica Fictícia é sempre, e tão somente, da esquerda para a direita, não existindo a possibilidade de fluxo em sentido contrário. No exemplo hipotético da figura abaixo, a bobina de Y0 poderia ser acionada apenas por meio de X0-X1-X2, ou X0-X3-X4, ou X5-X4, devidamente acionados. Porém, nunca por meio de X5-X3-X1-X2, mesmo que devidamente acionados.



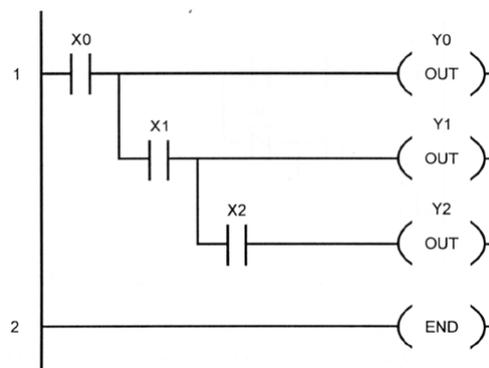
Implementação da Lógica de Controle

A Linguagem Ladder permite o acionamento de vários Elementos de Saída (bobinas, temporizadores, contadores, etc.) simultaneamente, por meio da mesma Lógica de Controle, sem

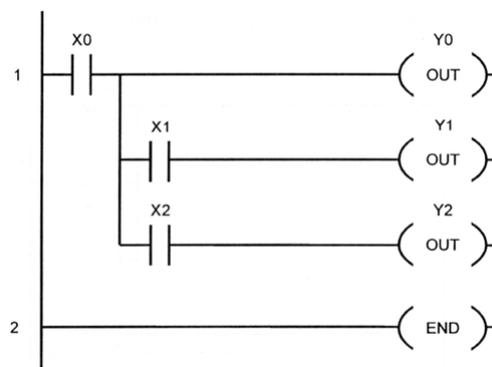
necessidade de construção de rungs similares. A Lógica de Controle implementada apresentada a seguir determina que, ao ser atuada a Entrada XO, as Saídas YO, Y1 e Y2 serão acionadas simultaneamente.



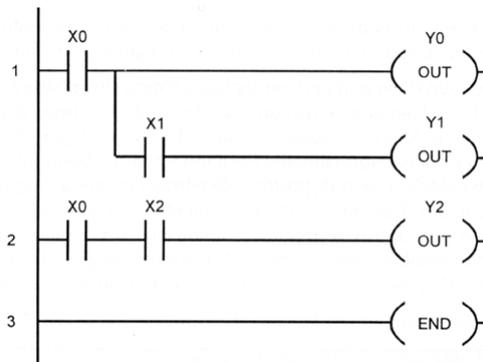
É possível, também, implementar a Lógica de Controle utilizando derivações nas Linhas de um determinado rung. A Lógica de Controle apresentada em seguida determina que a Saída YO é acionada a partir da condição da Entrada XO apenas ($YO = XO$), a Saída Y1 é acionada a partir da condição das Entradas XO e X1 ($Y1 = XO \cdot X1$) e a Saída Y2 é acionada a partir da condição das Entradas XO, X1 e X2 ($Y2 = XO \cdot X1 \cdot X2$). Com este recurso, evita-se a implementação de três Lógicas de Controle, em três rungs distintos.



Algumas CPUs podem apresentar restrições quanto à utilização desta forma de implementação da Lógica de Controle. Por exemplo, as CPUs Automationdirec.com não permitem que após a derivação, em qualquer Linha da Lógica de Controle, haja outro Elemento além da(s) Saída(s) controlada(s). Se isto ocorrer, haverá indicação de “Erro” após a compilação do Programa de Aplicação. A figura abaixo ilustra esta situação, na qual a Saída YO é acionada a partir da condição da Entrada XO apenas ($YO = XO$), a Saída Y1 é acionada a partir da condição das Entradas XO e X1 ($Y1 = XO \cdot X1$) e a Saída Y2 é acionada a partir da condição das Entradas XO e X2 ($Y2 = XO \cdot X2$).



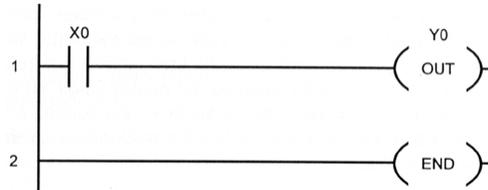
Uma forma de implementar esta mesma Lógica de Controle, nas CPUs que apresentem tal restrição, é ilustrada :



Relação o “Dispositivos de Entrada” x “Lógica de Controle”

A relação entre a condição do dispositivo de entrada (acionado ou não) e o Elemento utilizado na Lógica de Controle (Contato Normalmente Aberto ou Normalmente Fechado) pode causar certa confusão inicial ao usuário durante a implementação de Programas de Aplicação para PLCs.

Normalmente, faz-se a associação direta entre o Elemento utilizado na Lógica de Controle e a condição do dispositivo de entrada, o que gera tal confusão. Ou seja, ao se deparar com um programa que tenha Lógica de Controle semelhante à apresentada abaixo, acredita-se inicialmente que a Saída YO estará acionada quando a Entrada XO estiver aberta, tal qual indicado na Linguagem Ladder. A verdade é exatamente oposta a esta idéia, ou seja, a Saída YO só estará acionada quando a Entrada XO estiver fechada.



A relação existente entre a condição dos dispositivos de entrada e o Elemento utilizado na Lógica de Controle pode ser definida da seguinte maneira:

“Se o dispositivo de entrada estiver fechado (Ponto de Entrada / Tabela de Imagem das Entradas = 1), o Elemento utilizado na Lógica de Controle é atuado, ou seja o Contato Normalmente Aberto torna-se fechado (dando condição ao fluxo da Corrente Lógica Fictícia) e o Contato Normalmente Fechado torna-se aberto (impedindo o fluxo de tal corrente). Caso contrário, se o dispositivo de entrada estiver aberto (Ponto de Entrada / Tabela de imagem das Entradas = 0), o Elemento utilizado na Lógica de Controle mantém seu estado natural (ou de repouso), sendo que o Contato Normalmente Aberto permanece aberto (impedindo o fluxo da Corrente Lógica Fictícia) e o Contato Normalmente Fechado permanece fechado (dando condição ao fluxo desta corrente).”

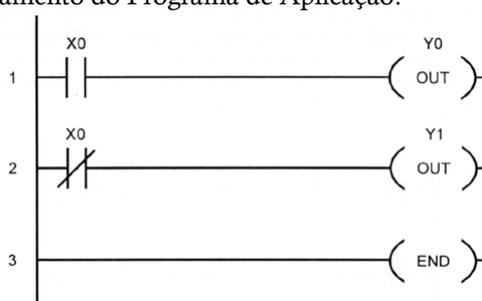
Esta definição é resumida na tabela abaixo.

Dispositivo de Entrada	Tabela de Imagem das Entradas	Elemento da Lógica de controle	Atuação do Contato Lógico	Fluxo de Corrente Lógica
	0		Não	Não
	1		Sim	Sim
	0		Não	Sim
	1		Sim	Não

Conforme visto, independente das características do dispositivo conectado ao Módulo de Entrada (Contato Normalmente Aberto - NA, ou Normalmente Fechado NF), a Lógica de Controle pode ser implementada com Contatos NA ou NF referenciados ao mesmo Ponto de Entrada.

Por exemplo, a Lógica de Controle implementada a seguir determina que se $XO = 1$ (entrada atuada), será acionada a Saída YO . Caso contrário, se $XO = 0$ (entrada não atuada), será acionada a Saída $Y1$. Embora a cada Ponto de Entrada, no caso XO , possa ser conectado apenas um tipo de contato do dispositivo de entrada (NA ou NF), a Lógica de Controle pode ser implementada de tal forma que realize operações distintas, conforme a atuação ou não do dispositivo de entrada, no caso acionando YO ou $Y1$.

Cada Ponto de Entrada tem apenas um único endereço a ele relacionado ($XO, X1, \dots$), porém pode ser utilizado tantas vezes quantas forem necessárias para a implementação da Lógica de Controle, ora como Contato NA, ora como Contato NF, tendo como único limite a quantidade de memória disponível ao armazenamento do Programa de Aplicação.



TIPOS DE DADOS

Além dos Pontos de Entrada e Saída Discretas, há outros Elementos (ou Tipos de Dados) utilizados na implementação da Lógica de Controle. Embora cada PLC utilize nomenclatura, representação gráfica (Linguagem Ladder) e forma de endereçamento próprias, a equivalência entre os Tipos de Dados disponíveis em CPUs distintas proporciona rápida adaptação ao usuário.

Os Tipos de Dados apresentados em seguida referem-se às CPUs Automationdirect.com. Porém, eles estão presentes, com as possíveis diferenças citadas acima, na maioria dos PLCs disponíveis no mercado.

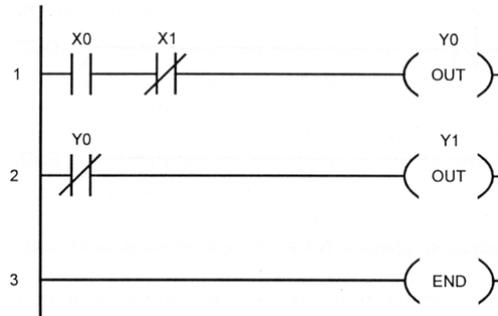
Entradas Discretas – Tipo de Dado: X

As Entradas Discretas são identificadas por X (Dado Tipo X), e cada Ponto é endereçado em base octal ($XO, X1, X2, \dots, X7, X10, X11, \dots, X77, X100, X101, \dots$). Normalmente, estão associadas às Instruções Booleanas de Entrada (Contatos NA ou NF).

Saídas Discretas - Tipo de Dado: Y

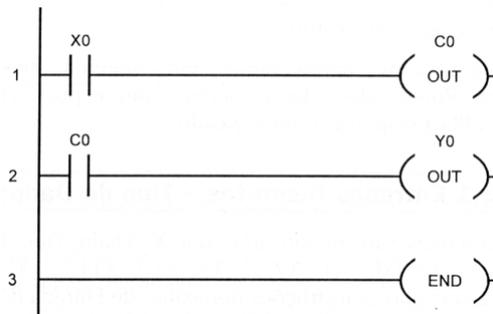
As Saídas Discretas são identificadas por Y (Dado Tipo Y), e cada Ponto é endereçado em base octal ($YO, Y1, Y2, \dots, Y7, Y10, \dots$). Embora estejam, normalmente, associadas às Instruções Booleanas de Saída (Bobinas de diversas funções), podem ser utilizadas também em Instruções Booleanas de Entrada (Contatos NA ou NF), conforme a necessidade.

Na Lógica de Controle implementada em seguida, pode-se observar a utilização do Dado Tipo Y (YO) associado a uma Instrução de Entrada (Contato NF). Neste caso, a Saída YO é acionada a partir da condição das Entradas XO e X1 ($XO = 1$ e $X1 = 0$). Caso esta condição não seja satisfeita, a Saída YO não é acionada (mantendo-se desligada), ocasionando o acionamento da Saída Y1 ($YO = 0$).



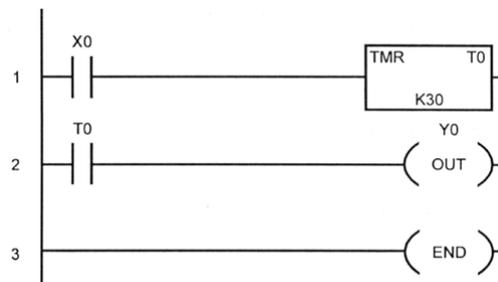
Relés de Controle - Tipo de Dado: C

Trata-se de bits internos à CPU, utilizados como Relés de Controle (ou Auxiliares), não tendo conexão a dispositivos externos de entrada ou saída. São identificados por C (Dado Tipo C), e cada Relé de Controle é endereçado em base octal (C0, C1, C2, ..., C7, C10, ...). São associados às Instruções Booleanas de Entrada ou de Saída. A Lógica de Controle implementada exemplifica a utilização dos Relés de Controle:



Temporizadores (Timers) e Bits de Status - Tipo de Dado: T

Normalmente, são utilizados para temporização de condições e/ou eventos controlados pelo Programa de Aplicação. Cada Temporizador é identificado por T (Dado Tipo T), e endereçado em base octal (T0, T1, T2, ..., T7, T10, ...). Há um Bit de Status (Bit de Condição) relacionado a cada Temporizador (com mesmo endereço), o qual é ativado quando o Valor Atual do Temporizador for igual ou superior ao Valor de Preset (Valor Pré-Configurado). A razão do incremento de tempo depende do Tipo de Temporizador utilizado.



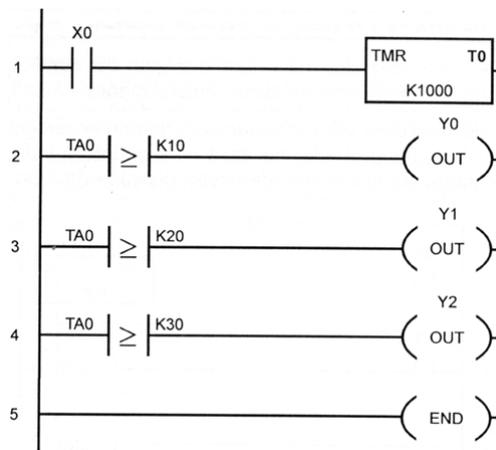
Na Lógica de Controle implementada na figura anterior, ao ser atuada a Entrada XO, é iniciada a temporização de TO, o qual tem Valor de Preset fixo em 3,0 segundos (o Temporizador apresentado possui incremento de tempo de 0,1s, portanto K30 equivale a 30 x 0,1 s). Ao ser atingido o Valor de Preset, é ativado o Bit de Status de TO, acionando a Saída YO. O Bit de Status

de To permanece ativado até que o Temporizador seja desativado ($XO = 0$, Valor Atual de $TO = 0$). Este bit pode ser associado a um Contato NA ou NF, conforme a necessidade.

Valor Atual dos Temporizadores - Tipo de Dado: V (TA)

Além de monitorar se o Temporizador atingiu o Valor de Preset, por meio do Bit de Status, é possível ter acesso ao Valor Atual de cada Temporizador durante a execução do Programa de Aplicação, por meio de um endereço específico da Tabela de Dados. Por exemplo, o endereço 0 (ou VO) contém o Valor Atual de TO, o endereço 1 (ou V1) contém o Valor Atual de T1, e assim sucessivamente. Estes endereços podem ser utilizados na implementação da Lógica de Controle, proporcionando flexibilidade no desenvolvimento do Programa de Aplicação. O Software de Programação da Autormationdirect.com (DirectSOFT) permite que, em vez do endereço (VO, V1, V2, ...), seja utilizada uma representação mais apropriada (TA0 para TO, TA1 para T1, TA2 para T2, ...) como indicação do Valor Atual de cada Temporizador.

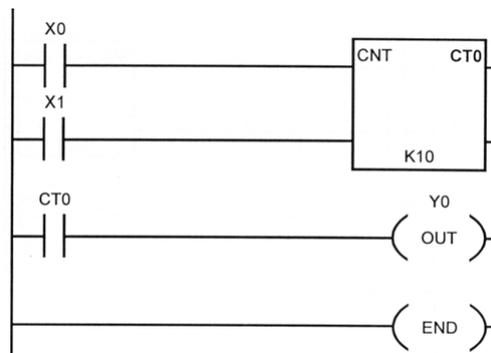
Na Lógica de Controle implementada adiante, ao ser atuada a Entrada XO, é iniciada a temporização de TO. Ao atingir 1s, a Saída YO é acionada (Valor Atual de TO \geq K10); ao atingir 2s, a Saída Y1 é acionada (Valor Atual de TO \geq k(20)); e ao atingir 3s, a Saída Y2 é acionada (Valor Atual de TO \geq K30).



Contadores (Counters) e Bits de Status - Tipo de Dado: CT

Normalmente, são utilizados para contagem de condições e/ou eventos controlados pelo Programa de Aplicação. Cada Contador é identificado por CT (Dado Tipo CT), e endereçado em base octal (CT0, CT1, CT2, ..., CT7, CT10, ...). Há um Bit de Status (Bit de Condição) relacionado a cada Contador (com mesmo endereço), o qual é ativado quando o Valor Atual do Contador for igual ou superior ao Valor de Preset (Valor Pré-Configurado).

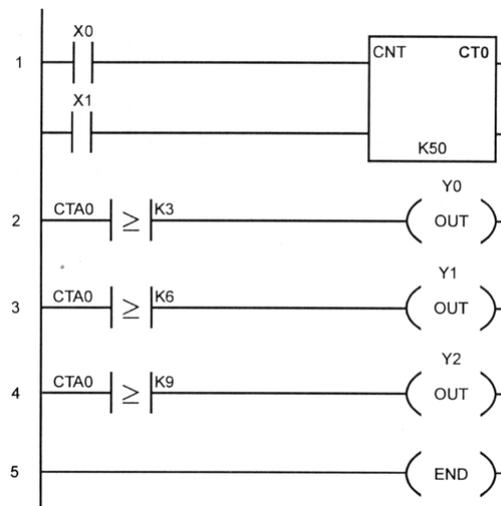
Na Lógica de Controle implementada abaixo, a cada transição de 0 para 1 (off -> on) da Entrada XO, o Valor Atual de CT0 é incrementado em uma unidade, o qual tem Valor de Preset fixo em 10. Ao ser atingido o Valor de Preset, é ativado o Bit de Status de CT0, acionando a Saída YO. O Bit de Status de CT0 permanece ativado até que o Contador seja reinicializado (resetado) por meio da atuação da Entrada X1 (Valor Atual de CT0 = 0). Este bit pode ser associado a um Contato NA ou NF, conforme a necessidade.



Valor Atual dos Contadores - Tipo de Dado: V (CTA)

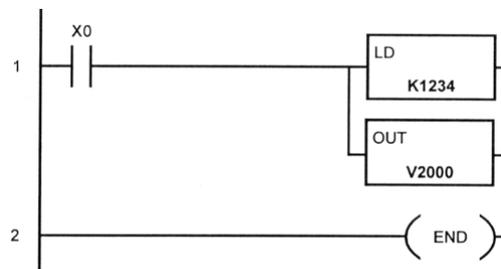
Além de monitorar se o Contador atingiu o Valor de Preset, por meio do Bit de Status, é possível ter acesso ao Valor Atual de cada Contador durante a execução do Programa de Aplicação, por meio de um endereço específico da Tabela de Dados. Por exemplo, o endereço 1000 (ou VI 000) contém o Valor Atual de CTO, o endereço 1001 (ou V1001) contém o Valor Atual de CT1, e assim sucessivamente. Estes endereços podem ser utilizados na implementação da Lógica de Controle, proporcionando flexibilidade no desenvolvimento do Programa de Aplicação. O Software de Programação da Automationdirect.com (DirectSOFT) permite que, em vez do endereço (VI 000, VI 001, VI 002, ...), seja utilizada uma representação mais apropriada (CTAO para CTO, CTA1 para CT1, CTA2 para CT2, ...) como indicação do Valor Atual de cada Contador.

Na Lógica de Controle implementada em seguida, a cada transição de 0 para 1 da Entrada XO, o Valor Atual de CTO é incrementado em uma unidade. Ao atingir o valor de contagem igual a 3, a Saída YO é acionada (Valor Atual de CTO \geq K3); ao atingir o valor de contagem igual a 6, a Saída Y1 é acionada (Valor Atual de CTO \geq K6); e ao atingir o valor de contagem igual a 9, a Saída Y2 é acionada (Valor Atual de CTO \geq K9).



Variáveis (Words) - Tipo de Dado: V

São posições de memória de 16 bits geralmente utilizadas para armazenamento ou manipulação de dados e valores. Cada Word é identificada por V (Dado Tipo V), e endereçada em base octal (V2000, V2001, ..., V2007, V2010, ...). Alguns endereços têm funções predefinidas, como, por exemplo, VO (armazena o Valor Atual de TO) ou V1000 (armazena o Valor Atual de CTO), sendo que a área livre disponível para o usuário é determinada pela CPU utilizada.



Todas as CPUs Automationdirect.com possuem acumulador de 32 bits. Assim, quando necessário, podem-se utilizar Instruções Duplas (que manipulam 32 bits) para operar com duas Words consecutivas de memória (V2000 e V2001, por exemplo).

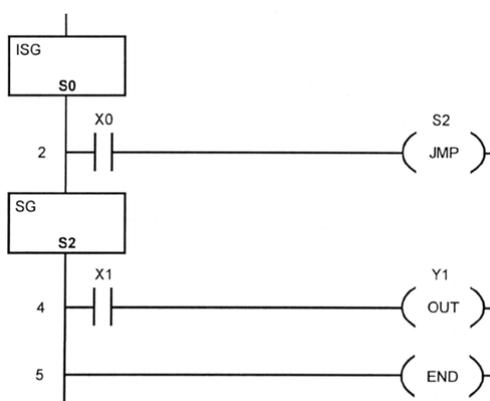
Na Lógica de Controle implementada acima, ao ser atuada a Entrada XO, o valor 1234 (em formato BCD/HEX) é carregado no acumulador (LD / Load - Carregar) e então enviado para a Word V2000 (OUT).

Estágios e Bit de Status - Tipo de Dado: S

Os Estágios são usados no desenvolvimento de Programas de Aplicação estruturados, permitindo a tradução da Descrição realizada em SFC (Capítulo 1) para Linguagem ladder por meio de Instruções RLL/plus (Relay Ladder Logic Plus) disponível na maioria das CPUs Automationdirect. com.

São identificados por S (Dado Tipo 5), e endereçados em base octal (S0, S1, ..., S7, S10, ...). Cada Estágio representa um segmento do Programa de Aplicação e contém a Lógica de Controle referente a este, que será executada apenas se o Estágio estiver ativo. Caso contrário, a CPU 'salta' para o próximo Estágio que atenda a esta condição. Há também um Bit de Status (Bit de Condição) relativo a cada Estágio (com mesmo endereço), que pode ser associado às Instruções Booleanas de Entrada (para indicar se o Estágio está ativo ou não - Contatos NA ou NF) ou de Saída (para determinar o acionamento ou não do Estágio - Instruções de Jump, Set ou Reset).

A Lógica de Controle implementada nesta última figura determina que, ao iniciar a execução do Programa de Aplicação, mesmo que a Entrada X1 seja atuada ($X1 = 1$), a Saída Y1 não será acionada, pois o Estágio que contém a Lógica de Controle (S2) não estará ativo. Neste momento, apenas o Estágio Inicial (S0) estará e permanecerá ativo, até que a Entrada XO seja atuada ($XO = 1$), dando condição de Jump para o Estágio S2, o que ocasiona a desativação de S0 e a ativação de S2 ($S0 = 0$ e $S2 = 1$), permitindo que a Entrada X1 acione a Saída Y1.

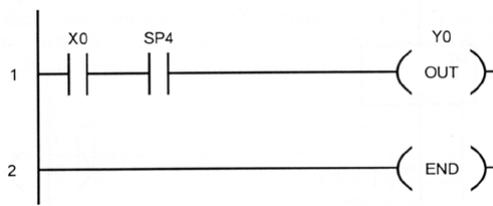


Relés Especiais (Special Relays) - Tipo de Dado: SP

Trata-se de bits internos à CPU com funções predefinidas pelo fabricante, como Indicação de Status do Acumulador da CPU, Monitoramento do Sistema, Indicação de Erros e Base de Tempo-Real, por exemplo. São identificados por SP (Dado Tipo SP), endereçados em base octal e só podem ser associados às Instruções Booleanas de Entrada (Contatos NA ou NF). A tabela a seguir apresenta alguns exemplos de Relés Especiais.

RELÉ	FUNÇÃO	DESCRIÇÃO
SP0	Primeiro Scan	Ativado apenas no primeiro scan da CPU; desativado nos demais scans
SP1	Sempre ON	Ativado em todos os scans da CPU
SP2	Sempre OFF	Desativado em todos os scans da CPU
SP3	Clock 1 Minuto	30 segundos on e 30 segundos off
SP4	Clock 1 Segundo	0,5 segundo on e 0,5 segundo off
SP5	Clock 100 ms	50 ms on e 50 ms off
SP6	Clock 50 ms	25 ms on e 25ms off
SP7	Scans Alternados	Ativo em um scan e desativado no scan seguinte, alternadamente

Na Lógica de Controle implementada, ao ser atuada a Entrada XO, a Saída YO permanecerá acionada por 0,5s e desligada por 0,5s (tempo determinado por SP4).



MAPEAMENTO DE MEMÓRIA

Todos os Tipos de Dados apresentados têm uma área de memória (Word ou VMemory) reservada para este fim. A quantidade de cada um dos Tipos de Dados disponíveis depende da CPU utilizada, porém o endereço inicial dessas áreas (Mapeamento) é sempre o mesmo, conforme apresentado nas tabelas seguintes.

Endereçamento das Entradas Discretas – Dado Tipo X (Endereços Iniciais).

Relação Word.bit x Dado Tipo X															MSB	LSB	Word
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
X17	X16	X15	X14	X13	X12	X11	X10	X7	X6	X5	X4	X3	X2	X1	X0		V40400
X37	X36	X35	X34	X33	X32	X31	X30	X27	X26	X25	X24	X23	X22	X21	X20		V40401
X57	X56	X55	X54	X53	X52	X51	X50	X47	X46	X45	X44	X43	X42	X41	X40		V40402
X77	X76	X75	X74	X73	X72	X71	X70	X67	X66	X65	X64	X63	X62	X61	X60		V40403

Endereçamento das Saídas Discretas – Dado Tipo Y (Endereços Iniciais).

Relação Word.bit x Dado Tipo Y															MSB	LSB	Word
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Y17	Y16	Y15	Y14	Y13	Y12	Y11	Y10	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0		V40500
Y37	Y36	Y35	Y34	Y33	Y32	Y31	Y30	Y27	Y26	Y25	Y24	Y23	Y22	Y21	Y20		V40501
Y57	Y56	Y55	Y54	Y53	Y52	Y51	Y50	Y47	Y46	Y45	Y44	Y43	Y42	Y41	Y40		V40502
Y77	Y76	Y75	Y74	Y73	Y72	Y71	Y70	Y67	Y66	Y65	Y64	Y63	Y62	Y61	Y60		V40503

Endereçamento dos Relés de Controle – Dado Tipo C (Endereços Iniciais).

Relação Word.bit x Dado Tipo C															MSB	LSB	Word
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
C17	C16	C15	C14	C13	C12	C11	C10	C7	C6	C5	C4	C3	C2	C1	C0		V40600
C37	C36	C35	C34	C33	C32	C31	C30	C27	C26	C25	C24	C23	C22	C21	C20		V40601
C57	C56	C55	C54	C53	C52	C51	C50	C47	C46	C45	C44	C43	C42	C41	C40		V40602
C77	C76	C75	C74	C73	C72	C71	C70	C67	C66	C65	C64	C63	C62	C61	C60		V40603

Endereçamento dos Bits de Status dos Temporizadores – Dado Tipo T (Endereços Iniciais).

Relação Word.bit x Dado Tipo T															MSB	LSB	Word
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
T17	T16	T15	T14	T13	T12	T11	T10	T7	T6	T5	T4	T3	T2	T1	T0		V41100
T37	T36	T35	T34	T33	T32	T31	T30	T27	T26	T25	T24	T23	T22	T21	T20		V41101
T57	T56	T55	T54	T53	T52	T51	T50	T47	T46	T45	T44	T43	T42	T41	T40		V41102
T77	T76	T75	T74	T73	T72	T71	T70	T67	T66	T65	T64	T63	T62	T61	T60		V41103

Endereçamento dos Bits de Status dos Contadores – Dado Tipo CT (Endereços Iniciais).

Relação Word.bit x Dado Tipo CT															MSB	LSB	Word
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
CT17	CT16	CT15	CT14	CT13	CT12	CT11	CT10	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0		V41400
CT37	CT36	CT35	CT34	CT33	CT32	CT31	CT30	CT27	CT26	CT25	CT24	CT23	CT22	CT21	CT20		V41401
CT57	CT56	CT55	CT54	CT53	CT52	CT51	CT50	CT47	CT46	CT45	CT44	CT43	CT42	CT41	CT40		V41402
CT77	CT76	CT75	CT74	CT73	CT72	CT71	CT70	CT67	CT66	CT65	CT64	CT63	CT62	CT61	CT60		V41403

Endereçamento dos Bits de Status dos Estágios – Dado Tipo S (Endereços Iniciais).

Relação Word.bit x Dado Tipo S															MSB	LSB	Word
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
S17	S16	S15	S14	S13	S12	S11	S10	S7	S6	S5	S4	S3	S2	S1	S0		V41000
S37	S36	S35	S34	S33	S32	S31	S30	S27	S26	S25	S24	S23	S22	S21	S20		V41001
S57	S56	S55	S54	S53	S52	S51	S50	S47	S46	S45	S44	S43	S42	S41	S40		V41002
S77	S76	S75	S74	S73	S72	S71	S70	S67	S66	S65	S64	S63	S62	S61	S60		V41003

Os Dados Tipo TA (Valor Atual dos Temporizadores) têm endereço inicial em V0 e os Dados Tipo CTA (Valor Atual dos Contadores) têm endereço inicial em V1000, conforme apresentado anteriormente. Já os Dados Tipo V (Word) têm endereço inicial dependente da CPU utilizada. Porém, o endereço V2000 é referência para a maioria dos modelos.