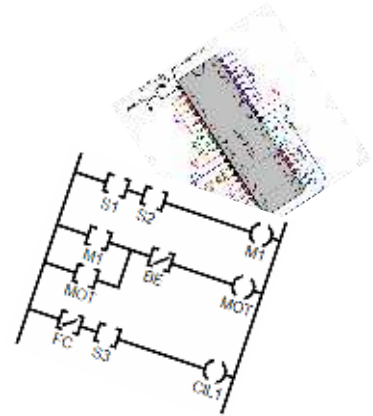


LINGUAGEM LADDER

p/ microcontroladores microchip PIC

Autor: Daniel Corteletti

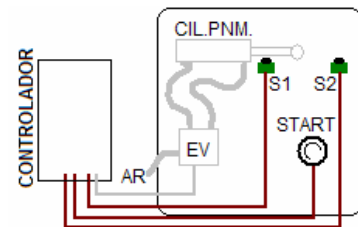
Centro Tecnológico de Mecatrônica SENAI



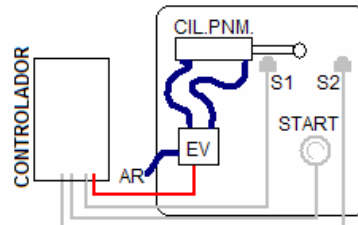
Ladder: É uma linguagem de programação gráfica, em forma de diagrama, que por ser de fácil criação e interpretação e representar ligações físicas entre componentes eletrônicos (sensores e atuadores), acaba sendo bastante utilizada em ambiente industrial.

Em um diagrama LADDER simples, podemos encontrar três tipos de elementos básicos:

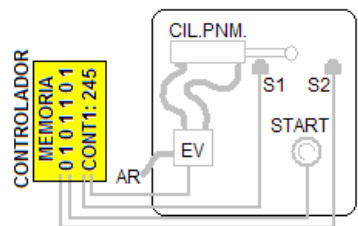
- 1) **CONTATO** (Contact): É o elemento que representa o sensor, ou seja, a entrada de sinal no bloco de controle lógico. Pode ser uma chave, um sensor reflexivo, um final de curso ou até mesmo o contato de algum relé auxiliar.



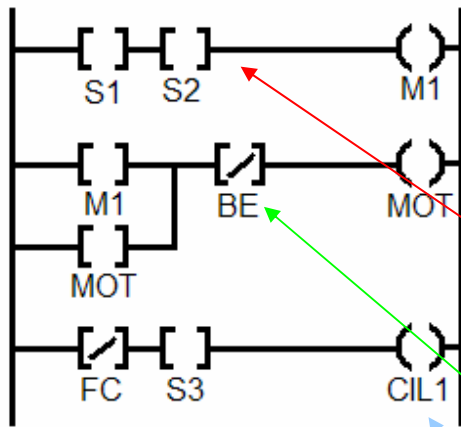
- 2) **BOBINA** (coil): É o elemento atuador, ou seja, o elemento acionado ou desligado pelo bloco de controle lógico. Pode ser uma contactora, um motor, uma lâmpada, um atuador auditivo, etc...



- 3) **MEMÓRIA** ou Relé Interno (Internal Relay): É a representação do estado de um contato ou bobina em memória, sem conexão direta com elementos externos.



Veja o exemplo de um diagrama LADDER:



Para este diagrama, temos o controle de 3 elementos, sendo estes M1, MOT e CIL1. Estes elementos podem ser BOBINAS (ATUADORES) ou MEMÓRIAS (relés internos).

Os elementos S1, S2, BE, VC e S3 só aparecem ao lado esquerdo do diagrama, no formato de colchetes [], o que pressupõe que sejam sensores (entradas).

Na primeira linha, observamos que a regra do programa define que a saída M1 irá ativar somente se os sensores S1 e S2 estiverem AMBOS ligados.

Na segunda linha deste programa, observa-se que a regra determina que a saída MOT irá ligar se BE estiver DESLIGADO (a barra significa inversão) e se M1 ou MOT estiver acionado (ao menos um destes).

Na terceira linha, observa-se que o atuador CIL1 irá ativar caso o sensor FC estiver DESLIGADO (novamente observe a barra), e se o sensor S3 estiver acionado.

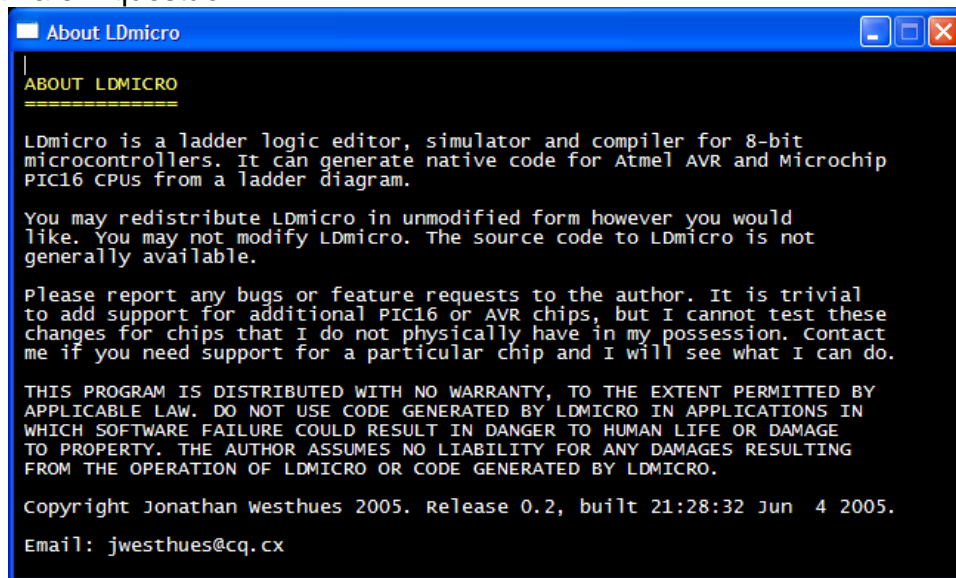
Existem ainda algumas outras regras importantes sobre programação LADDER:

- 1) Não é permitido (ou pelo menos recomendado) o uso de uma mesma bobina em mais de uma linha, pois as regras irão conflitar. Por exemplo, não poderíamos inserir no diagrama anteriormente representado mais uma linha que acionasse o atuador CIL1.
- 2) Existe a possibilidade, em algumas variações da linguagem, do uso do comando SET e RESET (liga e desliga) que determina em que momento um determinado atuador irá ligar ou desligar.
- 3) Existem blocos especiais que permitem temporizar, detectar pulso, borda, contagem e outros recursos. Isso pode variar conforme a linguagem utilizada.

LADDER PARA MICROCONTROLADOR PIC – O LDMICRO

A linguagem LADDER nasceu na necessidade de facilitar a programação em ambientes industriais, remetendo para uma linguagem de alto nível e fácil de ser utilizada. No entanto existe um programa, (LDMICRO) de Jonathan Westhues, que permite a programação LADDER de microcontroladores, que viabiliza o estudo e implementação de controles de baixíssimo custo.

Este software é muito versátil, não requer instalação (basta executar o arquivo ldmicro.exe em ambiente windows ou emulador compatível), e é de livre distribuição, como podemos ver na janela abaixo, extraída do próprio HELP do programa em questão:



```

About LdMicro

ABOUT LDMICRO

LdMicro is a ladder logic editor, simulator and compiler for 8-bit
microcontrollers. It can generate native code for Atmel AVR and Microchip
PIC16 CPUs from a ladder diagram.

You may redistribute LdMicro in unmodified form however you would
like. You may not modify LdMicro. The source code to LdMicro is not
generally available.

Please report any bugs or feature requests to the author. It is trivial
to add support for additional PIC16 or AVR chips, but I cannot test these
changes for chips that I do not physically have in my possession. Contact
me if you need support for a particular chip and I will see what I can do.

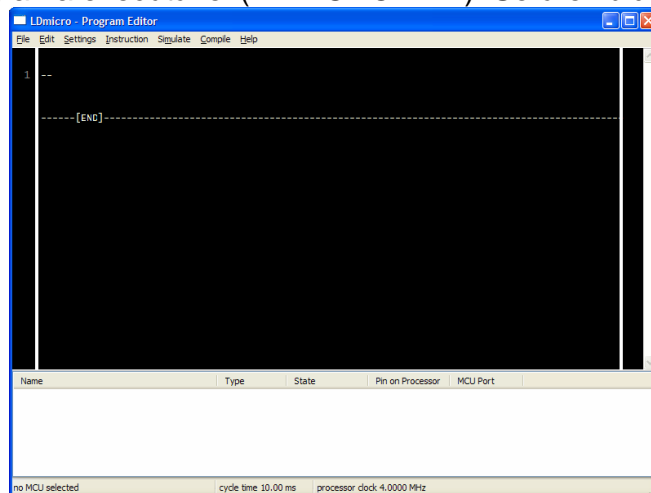
THIS PROGRAM IS DISTRIBUTED WITH NO WARRANTY, TO THE EXTENT PERMITTED BY
APPLICABLE LAW. DO NOT USE CODE GENERATED BY LDMICRO IN APPLICATIONS IN
WHICH SOFTWARE FAILURE COULD RESULT IN DANGER TO HUMAN LIFE OR DAMAGE
TO PROPERTY. THE AUTHOR ASSUMES NO LIABILITY FOR ANY DAMAGES RESULTING
FROM THE OPERATION OF LDMICRO OR CODE GENERATED BY LDMICRO.

Copyright Jonathan westhues 2005. Release 0.2, built 21:28:32 Jun 4 2005.
Email: jwesthues@cq.cx

```

O LDMICRO funciona da seguinte forma:

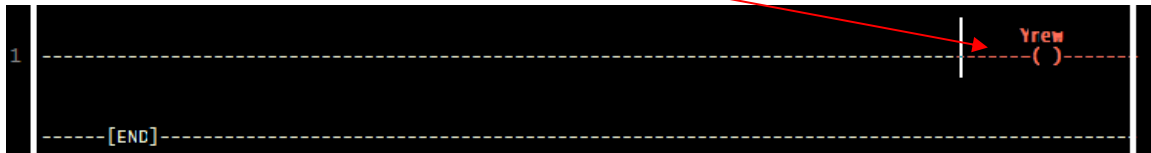
- 1) Inicie o programa executável (LDMICRO.EXE). Será exibida a seguinte tela:



É neste ambiente que você pode gerar o programa LADDER para microcontrolador.

Para inserir uma bobina, pressione L.

Você notará que será construída (ou complementada) a linha editada com a bobina indicada. É permitido inserir mais de uma bobina para a mesma linha.



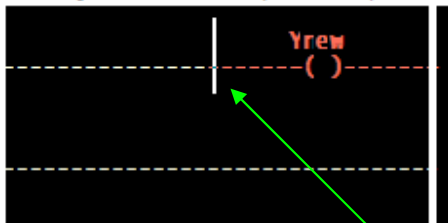
Clicando duas vezes sobre a bobina criada, será aberta a caixa de propriedade da bobina:

Não é bobina externa, é um relé interno (memória)
A bobina é um atuador, uma saída física.

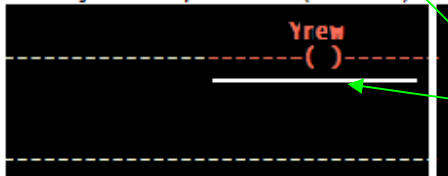
Bobina Normal
Bobina Invertida (negada)
Set somente (ligar somente)
Reset somente (desligar somente)

Nome da bobina

Inserção em série (ao lado)



Inserção em paralelo (abaixo)



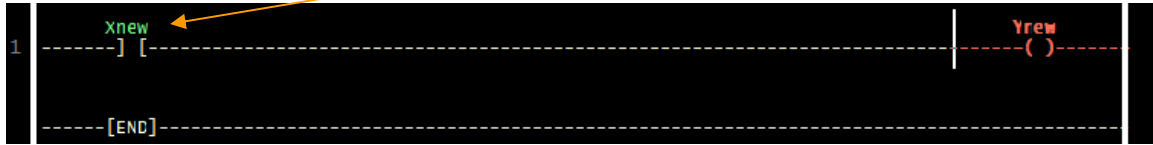
Se a bobina for definida (no campo Source) como INTERNAL RELAY, **o nome da bobina no diagrama ladder será precedido pela letra R**. Exemplo: Se o nome da bobina for new (como no exemplo acima), e se esta for definida como Internal Relay, será exibida como Rnew.

Se a bobina for definida como PIN ON MCU, o nome da bobina **será precedido pela letra Y** (no caso do exemplo, Ynew).

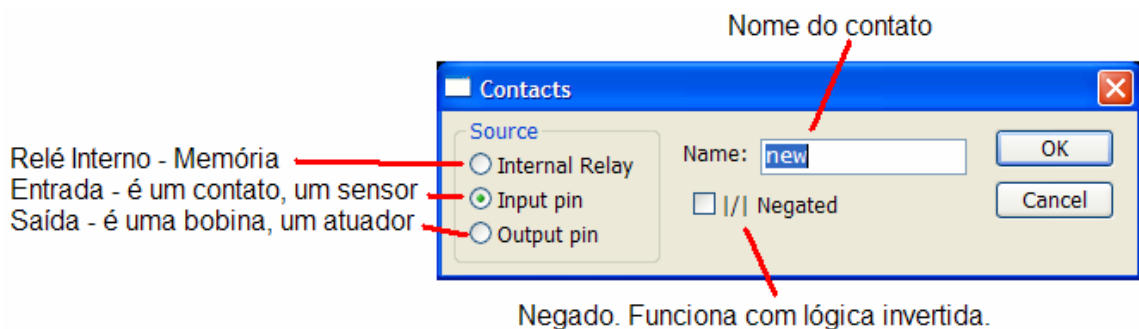
Observe que ao se inserir um contato ou bobina, será respeitada a posição do cursor (barra piscante) para definir o local da inserção. Ou seja, para inserir uma bobina ou contato abaixo de outra, posicione primeiro o cursor na posição horizontal.

Para inserir um contato:

Posicione o cursor no local desejado, e pressione C.



Note que surgirá um campo definido por colchetes --] [--- com o nome Xnew. Clique duas vezes sobre este item para abrir a caixa de propriedades do contato.



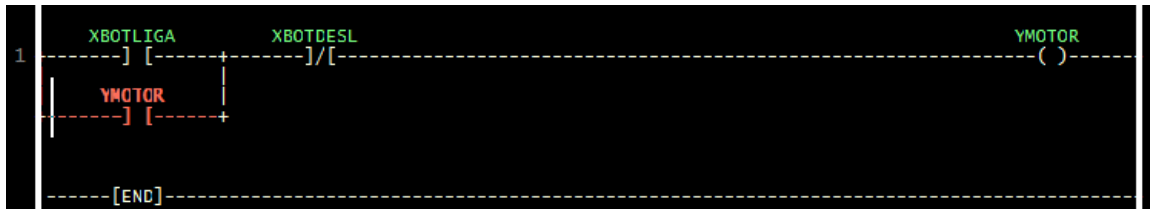
No campo source, você pode definir se o contato é um relé interno (memória). Para este caso, note que **o nome do contato será precedido pela letra R**. Se for definido como INPUT PIN (padrão), o contato é um sensor, uma entrada de sinal digital. Neste caso, **o nome do contato será precedido pela letra X** (como no exemplo acima: Xnew).

Se você desejar usar uma bobina como contato (isso é possível em ladder), basta ativar a opção **OUTPUT PIN**. Neste caso **o nome do elemento inserido será precedido pela letra Y**.

A caixa [/] define que a entrada funcionará negada (com **lógica invertida**), ou seja, aciona zerando o contato, e desativa ligando o contato.

Prática:

Tente agora montar o seguinte diagrama LADDER usando os recursos acima citados:



Depois de editar este programa (**observe que os elementos usados são somente e exatamente XBOTLIGA, XBOTDESL, YMOTOR**). Não deve haver nenhum outro elemento no programa.

SALVANDO

Após escrever seu programa, salve-o clicando em FILE -> SAVE AS... Salve como um arquivo com extensão LD.

SIMULANDO

Com o programa salvo, para simular o programa, clique em **SIMULATE** → **SIMULATION MODE**, e posteriormente em **SIMULATE** → **START REAL TIME SIMULATION**.

A partir deste momento, observe no painel da parte inferior da janela o estado dos contatos e das bobinas. **Basta dar um DUPLO CLICK** sobre o item para mudar seu estado.

Teste alterando o estado dos sensores, e veja se o programa funciona.

CONTATO ABERTO CONTATO FECHADO ATUADOR ATIVADO

Name	Type	State	Pin on Processor	MCU Port
XBOTDESL	digital in	0	(not assigned)	
XBOTLIGA	digital in	0	(not assigned)	
YMOTOR	digital out	1	(not assigned)	

no MCU selected cycle time 10.00 ms processor clock 4.0000 MHz

Isso significa que o contato (digital in) BOTDESL está em off.
Isso significa que o contato (digital in) BOTLIGA está em off.
Isso significa que a bobina MOTOR está acionada

COMPILANDO

Para gerar um arquivo HEX a partir deste programa, basta seguir estes passos:

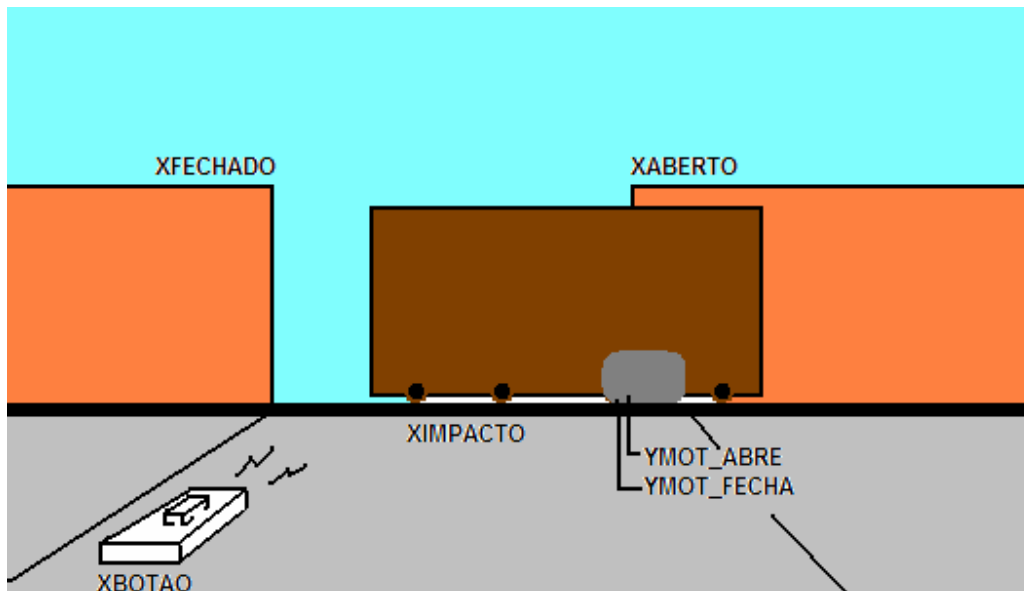
- 1) Clique em **SETTINGS** → **MICROCONTROLLER** e defina qual o microcontrolador a ser utilizado. Para melhor funcionamento, clique em **SETTINGS** → **MCU PARAMETERS** e defina o valor do cristal de clock utilizado. O padrão é 4MHz.
- 2) Agora de um duplo clique sobre cada elemento DIGITAL IN ou DIGITAL OUT da parte inferior da janela, **associando** cada CONTATO ou BOBINA a um pino do microcontrolador.
- 3) Agora clique em **COMPILE** → **COMPILE AS..** e indique o nome do arquivo a ser gerado. **IMPORTANTE:** Não esqueça de colocar a extensão HEX. Ex: **PROG.HEX**. Caso você não informe a extensão, ficará mais difícil achá-la depois com o programa de gravação (EPIC, ICPROG, etc...)

Comandos mais usados:

Inserir nova linha	shift V ou shift 6
Inserir um comentário	ponto e vírgula
Detecta borda subida	/
Detecta borda descida	\
Temporizar para desligar	F
Temporizar para ligar	O
Temporizar para ligar retentivo	T
Contador incremental	U
Contador decremental	I
Contador circular	J
Compara igualdade	=
Compara se é maior	>
Compara se é menor	<
Compara se é maior ou igual	.
Compara se é menor ou igual	,
Inserir BOBINA	L
Inserir Contato	C
Inserir reset de contador	E
Carrega variável c/ valor	M
Inserir operação soma	+
Inserir operação subtração	-
Inserir operação multiplic.	*
Inserir operação de divisão	D
Leitura de analógico	P

Insert Comment	;
Insert Contacts	C
Insert OSR (One Shot Rising)	/
Insert OSF (One Shot Falling)	\
Insert TON (Delayed Turn On)	O
Insert TOF (Delayed Turn Off)	F
Insert RTO (Retentive Delayed Turn On)	T
Insert CTU (Count Up)	U
Insert CTD (Count Down)	I
Insert CTC (Count Circular)	J
Insert EQU (Compare for Equals)	=
Insert NEQ (Compare for Not Equals)	
Insert GRT (Compare for Greater Than)	>
Insert GEQ (Compare for Greater Than or Equal)	.
Insert LES (Compare for Less Than)	<
Insert LEQ (Compare for Less Than or Equal)	,
Insert Open-Circuit	
Insert Short-Circuit	
Insert Coil	L
Insert RES (Counter/RTO Reset)	E
Insert MOV (Move)	M
Insert ADD (16-bit Integer Add)	+
Insert SUB (16-bit Integer Subtract)	-
Insert MUL (16-bit Integer Multiply)	*
Insert DIV (16-bit Integer Divide)	D
Insert UART Send	
Insert UART Receive	
Insert A/D Converter Read	P

2) Tente criar o esquema ladder para um portão de garagem. Usem os seguintes elementos:



XBOTAO : Botão do controle remoto.

XABERTO: Sensor de final de curso que determina que o portão está aberto

XFECHADO: Sensor de final de curso que determina que o portão está fechado

XIMPACTO: Sensor de impacto. Detecta que o portão colidiu em algo.

YMOT_ABRE: Motor que move o portão no sentido de abrir.

YMOT_FECHA: Motor que move o portão no sentido de fechar.

Use sua criatividade. Simule o programa no ambiente ladder, e na estação c/ microcontrolador PIC. Bom trabalho.