Parte 4 - Técnicas de programação (Lógica simples)

INTRODUÇÃO

Programar em ladder é muito simples, desde que ele tenha uma estrutura sob a qual o programa deve ser desenvolvido, ou seja, se deve ter um modelo de comportamento, obviamente antes de programar em ladder. Este modelo pode ser elaborado de varias maneiras, o importante é ter algo em que se basear um modelo impecável resultará em um programa ladder impecável. As técnicas utilizadas neste trabalho são:

4.1. Lógica combinacional simples: São utilizados em lógica simples sem muitas divergências e convergências, são sugeridos aos que tem familiaridade com sistemas digitais, porém se o modelo ficar muito extenso deve-se minimiza-lo.

4.2. Mapas de Veith-Karnaugh: São utilizados na minimização de sistemas de dificuldade média ou em sistemas onde o comportamento de entradas depende de outras entradas. Se as entradas forem superiores a quatro os mapas não são recomendados.

4.3. Máquina de estados: São utilizados em sistemas de complexos, é de fácil transformação para ladder desde que não haja muitas ramificações.

4.1 LÓGICA COMBINACIONAL SIMPLES

O CLP é um equipamento eletrônico que entre suas aplicações mais simples, esta a execução de funções lógicas em um ambiente industrial. E quando se fala em lógica, logo vêm à mente funções lógicas como “E” ou “AND” e “OU” ou “OR”, muito conhecidas na eletrônica digital. Esta mesma lógica, com algumas mudanças nos símbolos, também pode ser usada na estruturação de programas a serem desenvolvidos em ladder.

Principais blocos

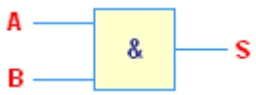
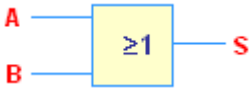
Lógica AND (E)	Expressão lógica	Função executada
	**S = A.B**	Executa função lógica “AND”, ou seja, somente se as entradas A e B estiverem em nível alto a saída S será acionada.

Tabela verdade

A	B	S
0	0	0
0	1	0
1	0	0
1	**1**	**1**

Programa Ladder correspondente

Lógica OR (ou) | **Expressão lógica** | **Função executada**



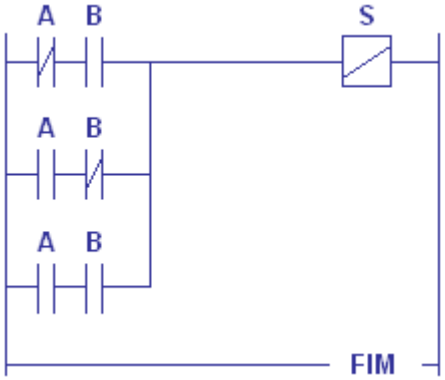
S = A+B

Executa função lógica "OR", ou seja, para que a saída S seja acionada basta que uma das entradas A ou B esteja em nível alto.

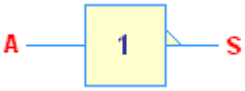
Tabela verdade

A	B	S
0	0	0
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

Programa Ladder correspondente



Lógica NOT (não) | **Expressão lógica** | **Função executada**



S = A'

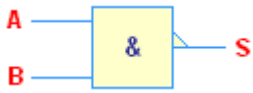
Executa função lógica "NOT", ou seja, nega ou inverte o sinal de entrada.

Tabela verdade

A	S
0	**1**
1	0

Programa Ladder correspondente

Lógica NAND (não e) | **Expressão lógica** | **Função executada**



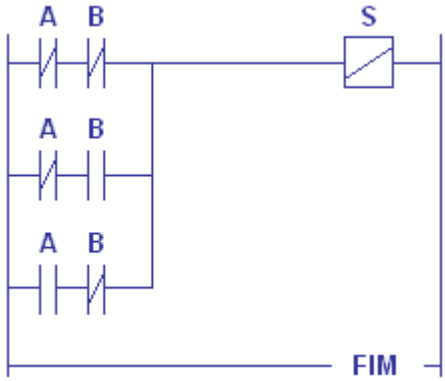
S = (A.B)'

Executa função lógica "NAND", ou seja, nega ou inverte as saídas da função AND.

Tabela verdade

A	B	S
0	**0**	**1**
0	**1**	**1**
1	**0**	**1**
1	1	0

Programa Ladder correspondente



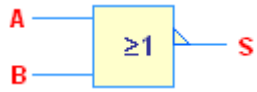
Lógica NOR (não ou)	Expressão lógica	Função executada
	S = (A+B)'	Executa função lógica "NOR", ou seja, nega a função OR, invertendo assim, suas saídas.

Tabela verdade

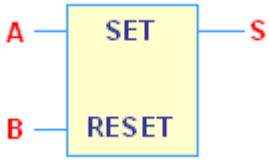
A	B	S
0	**0**	**1**
0	1	0
1	0	0
1	1	0

Programa Ladder correspondente

Blocos de memorização.

São utilizados com muita freqüência, pois tem a função de memorizar um BIT.

SET RESET



Função executada

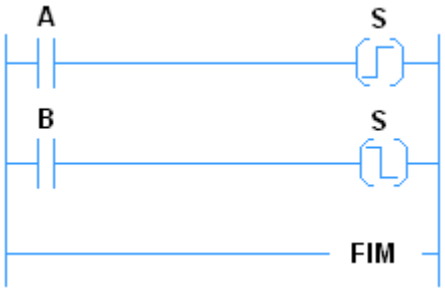
"Set" significa Ligar e "Reset" desligar. Seu funcionamento é simples uma vez setado (nível lógico (1) em A) ele comuta a saída S, ou seja, vai para (1) e somente volta para nível baixo (0) se for resetado.

Tabela verdade

A	B	S
0	**0**	**CA**
0	**1**	**0**
1	**0**	**1**
1	**1**	**CP**

CA= Condição Anterior
CP= Condição Proibida

Programa Ladder correspondente

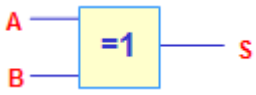


Exercícios com lógica simples

1- Implemente uma lógica XOR (OU exclusivo) em ladder.

I0	I1	O0

Símbolo (FDB)

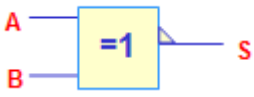


Ladder correspondente

2- Implemente uma lógica XNOR (OU coincidência) em ladder.

I0	I1	O0

Símbolo (FDB)



Ladder correspondente

3- Extraia a expressão lógica, monte o circuito lógico (utilize blocos lógicos funcionais) e construa a lógica ladder a partir da tabela verdade.

a)

I0	I1	O0
0	0	0
0	1	1
1	0	0
1	1	1

Ladder aqui e demais respostas (no verso)

b)

I0	I1	O0
0	0	0
0	1	1
1	0	0
1	1	0

Ladder aqui e demais respostas (no verso)

c)

I0	I1	O0
0	0	1
0	1	1
1	0	0
1	1	1

Ladder aqui e demais respostas (no verso)

d)

I0	I1	I3	O0
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Ladder aqui e demais respostas (no verso)

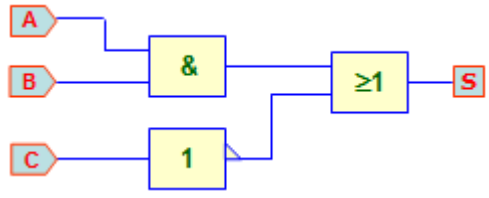
e)

I0	I1	I3	O0
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

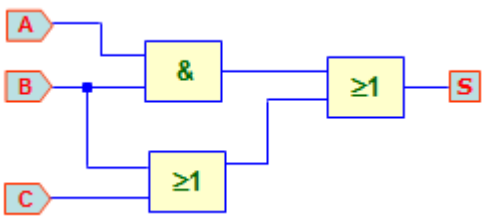
Ladder aqui e demais respostas (no verso)

Converta os diagramas (dado em FBD) para ladder.

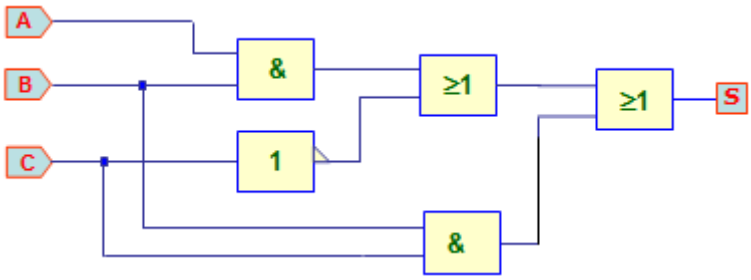
a) Lógica FBD **Ladder correspondente**



b) Lógica FBD **Ladder correspondente**



c) Lógica FBD **Ladder correspondente**



Ladder correspondente

Dado as seguintes expressões lógicas, construa o diagrama correspondente em linguagem ladder e em diagramas de blocos funcionais (FDB).

a) S= (A+B).C

b) S= (A.B)+(C.D)

c) S= (A'+B).(C.D)'

d) S= (A+B).D'

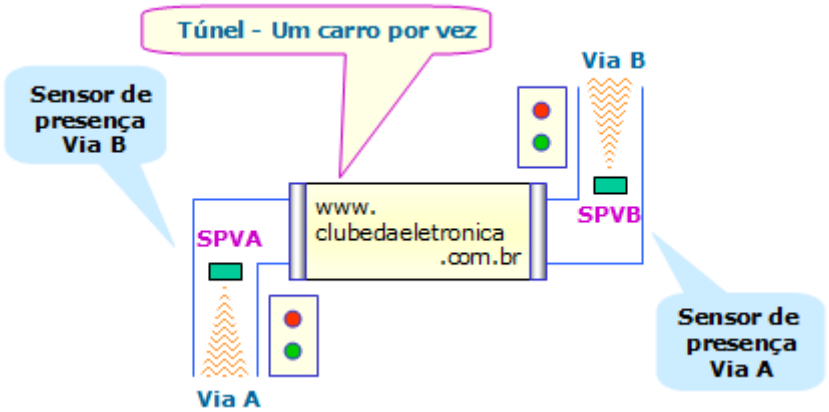
Aplicação da lógica combinacional (Simples)

A aplicação da lógica combinacional é sem dúvidas o que mais interessa nos sistemas digitais, pois pode ser usada em diversas áreas.

Aplicação 01 – Controle de trafego (resolvido)

Deseja-se programar um controle de trafego para um túnel que só permite a passagem de um carro por vez.

Veja ilustração:



A prefeitura que encomendou o projeto tem os seguintes critérios:

Quando os sensores detectarem a presença do carro, um nível lógico alto (ON) será enviado ao seu respectivo dispositivo de atuação.

Situação dos sensores		Critérios de projeto
SPVA (SA)	**SPVB (SB)**	
OFF	**OFF**	Se não houver nenhum carro, a **via B deverá ser liberada (verde)** e a **via A bloqueada (vermelho)**.
OFF	**ON**	**Se o sensor detectar carro na** via B, esta será liberada (sinal verde) **e a** Via A bloqueada (sinal vermelho)**.**
ON	**OFF**	Se o sensor detectar carro na **via A, esta será liberada (sinal verde)** e a **Via B bloqueada (sinal vermelho)**.
ON	**ON**	Se ambos os sensores detectarem carros, a via A deverá ser liberada (sinal verde) e a **via B bloqueada (sinal vermelho)**.

1º Passo – Montar a tabela verdade a partir de todas as condições possíveis

SPVA (SA)	SPVB (SB)
0	0
0	1
1	0
1	1

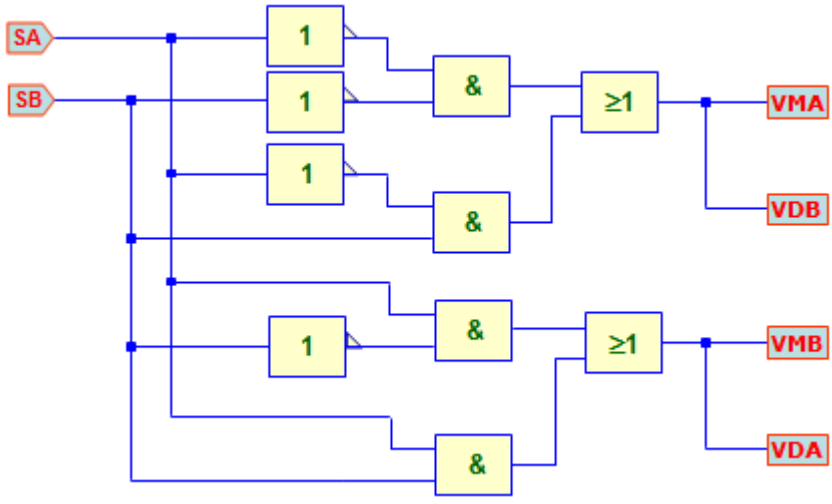
VMA	VDA	VMB	VDB
1	0	0	1
1	0	0	1
0	1	1	0
0	1	1	0

2º Passo – Extrair a tabela verdade das expressões verdadeiras

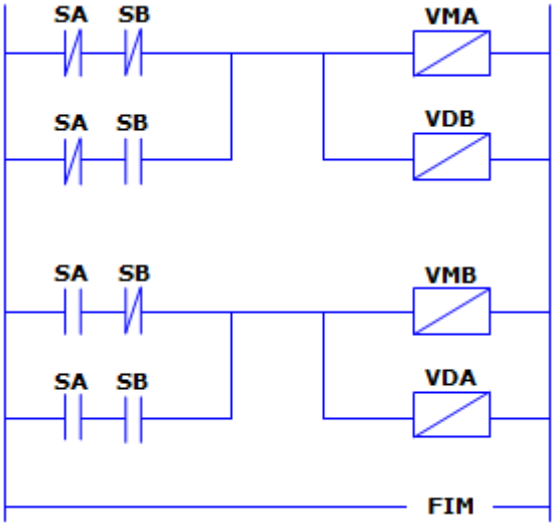
$$VMA = VDB = (SA' . SB') + (SA' . SB)$$

$$VMB = VDA = (SA . SB') + (SA . SB)$$

3º Passo – Montar o circuito lógico



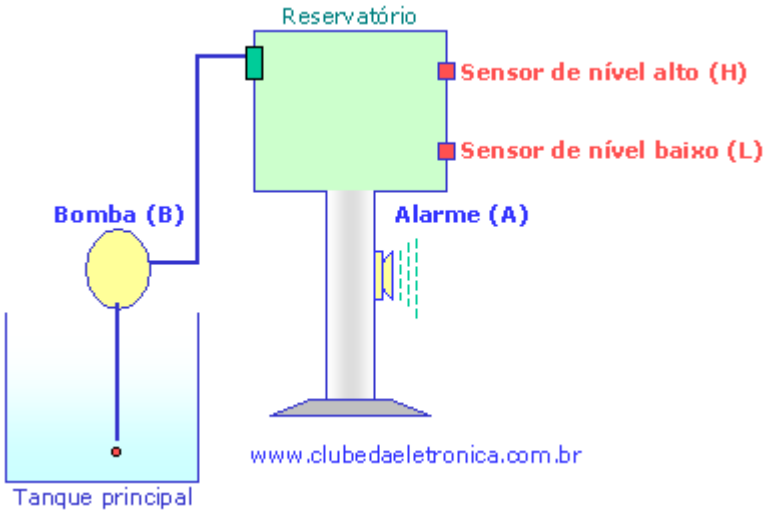
4º Passo – Montar o programa ladder



Praticando...

1- Aplicação 2 – controle de nível

Deseja-se controlar o nível de água de um reservatório, conforme ilustração:

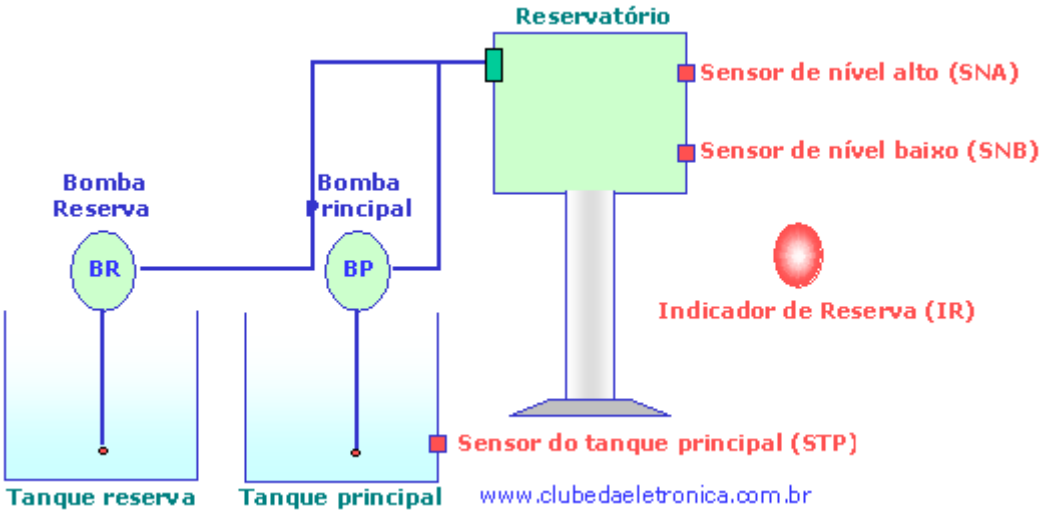


Descrição de funcionamento:

- O reservatório deve estar sempre cheio, ou seja, H=1;
- Se H=0, a bomba deverá ser acionada;
- Se a bomba não atender a demanda e o reservatório esvaziar, ou seja, L=0, um alarme deverá ser acionado.

2- Aplicação 3 – controle de nível com tanque reserva

Deseja-se controlar o nível de água de um reservatório, conforme ilustração:



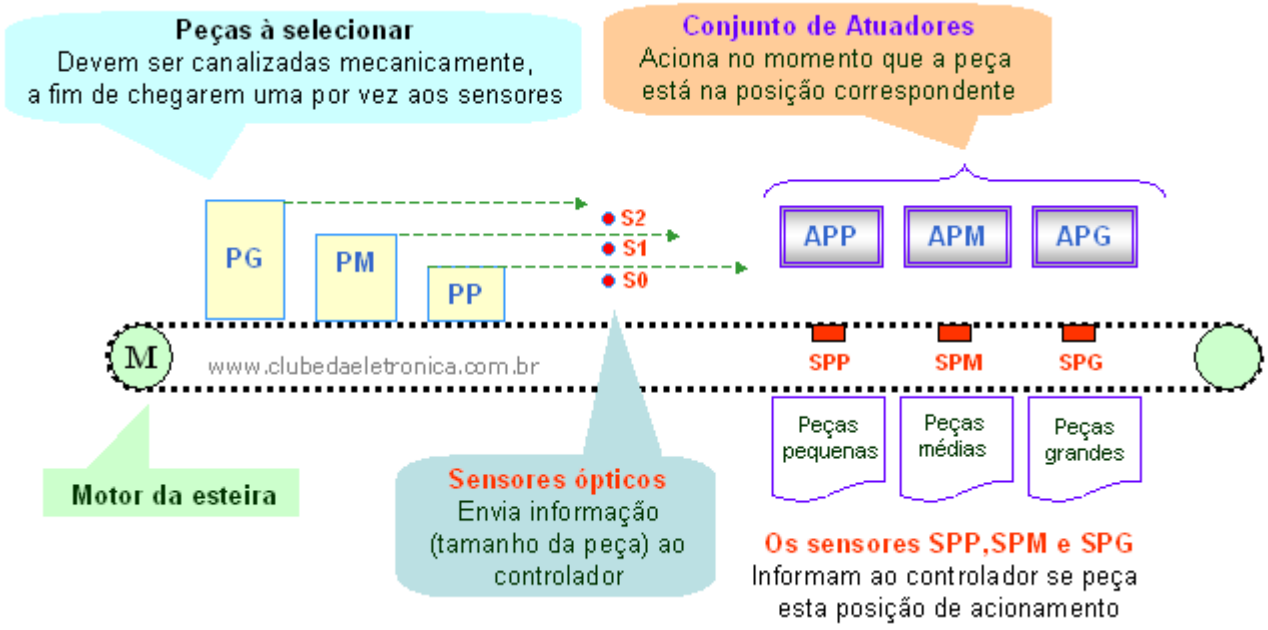
Seu funcionamento deve ser o seguinte:

- O reservatório deve estar sempre cheio, ou seja, SNA=1;
- Se SNA=0, a bomba principal BP deverá ser acionada, mas somente se houver água no tanque principal, ou seja, STP =1, se STP =0, a bomba reserva deve ser acionada;
- Se a bomba reserva BR for acionada, um indicador de reserva (IR) deverá ser acionado.

3- Aplicação 4 – Selecionar de peças (Resolvido)

Deseja-se implementar um selecionador de peças pequenas, médias e grandes. O sistema consiste dois sensores S1 e S2 que selecionarão as peças e três atuadores sendo um para cada tipo de peça que deverão colocar cada peça em seu respectivo compartimento.

Ilustração simplificada:



Descrição:

- Se nenhum sensor for ativado, então a peça é pequena.
- Se somente o sensor S1 for ativado, então a peça é média.
- Se os dois sensores forem ativados então a peça é grande.

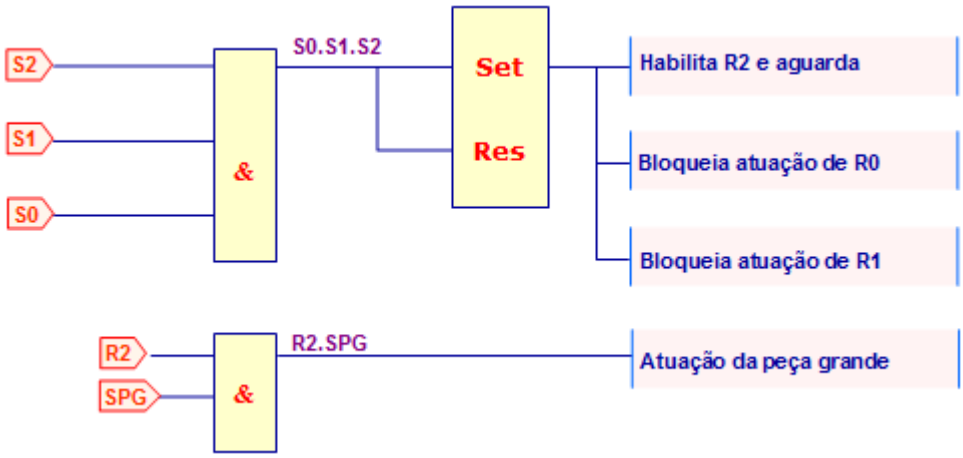
Nota: O processo é continuo e somente haverá uma atuação por vez.
Haverá um alimentador (não incluso) que colocará uma peça por vez com intervalo de tempo pré-definido entre elas.

Descrição das etapas:

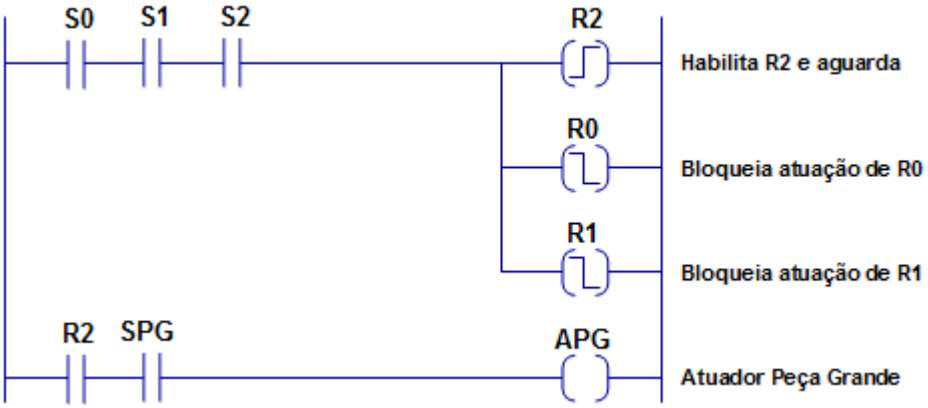
Peça grande ⇒ Se S1, S2 e S3 forem cortados, "setará" um contato auxiliar R0 que fica aguardando a posição atuador de peça grande (SPG) e se esta for alcançada a peça será retirada.

A peça grande só será retirada se as peças pequenas ou médias não estiverem aguardando o sensor de suas posições.

O diagrama de blocos para peça grande



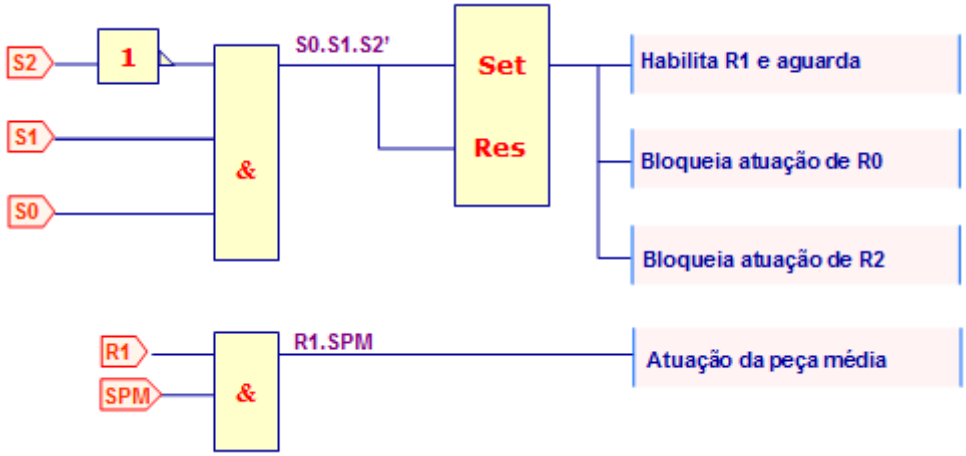
Ladder correspondente para peça grande



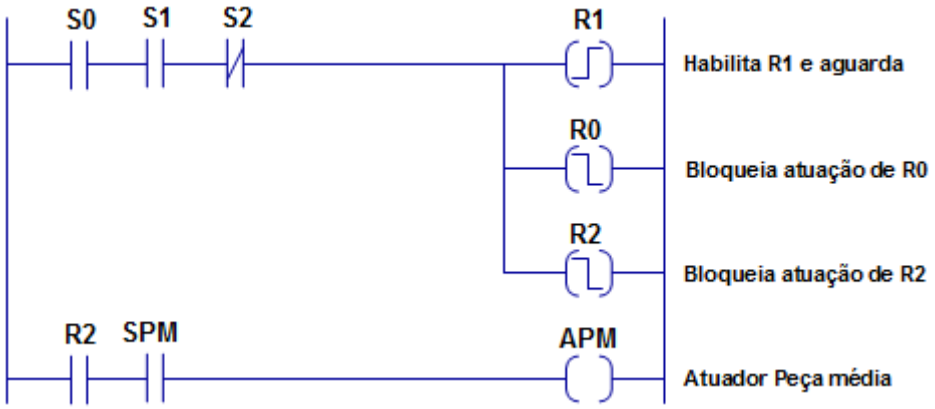
Peça média ⇒ Se S0 e S1 forem cortados e S2 não "setará" um contato auxiliar R1 que fica aguardando a posição atuador de peça média (SPM) se esta for alcançada a peça será retirada.

A peça média só será retirada se as peças pequenas ou grandes não estiverem aguardando o sensor de suas posições.

O diagrama de blocos para peça média



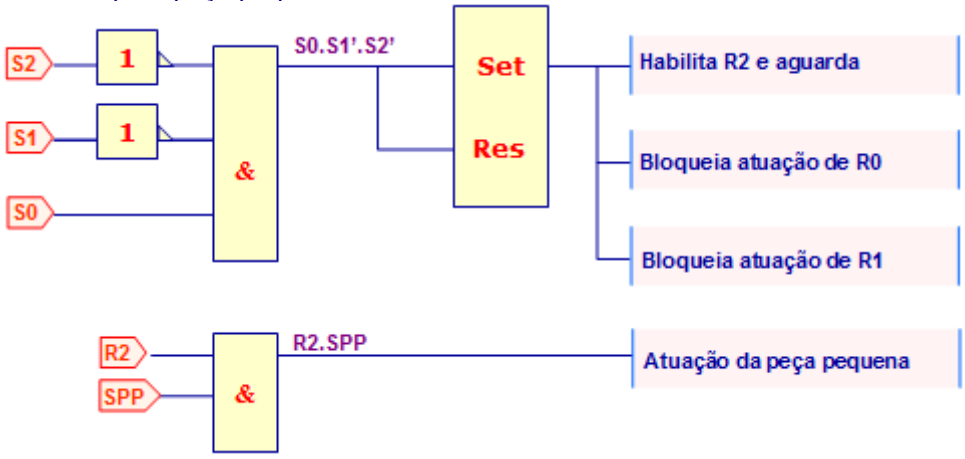
Ladder correspondente para peça média



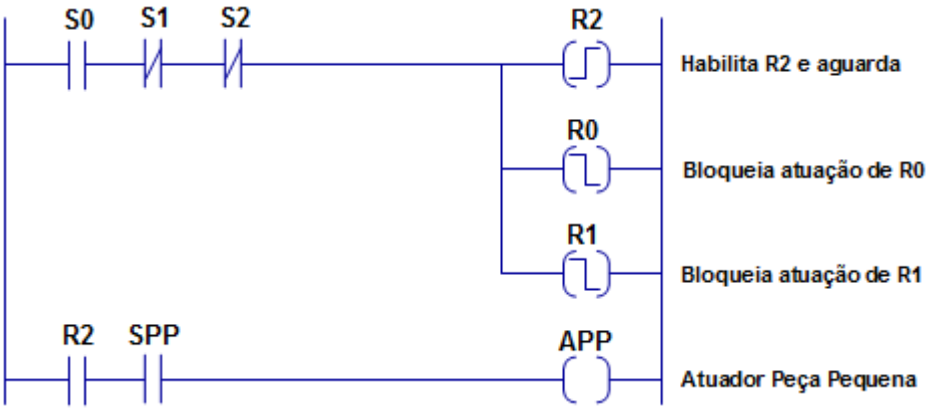
Peça pequena ⇒ Se somente S0 for cortado, "setará" um contato auxiliar R2 que fica aguardando a posição atuador de peça pequena (SPP) e se esta for alcançada a peça será retirada.

A peça pequena só será retirada se as peças médias ou grandes não estiverem aguardando o sensor de suas posições.

O diagrama de blocos para peça pequena

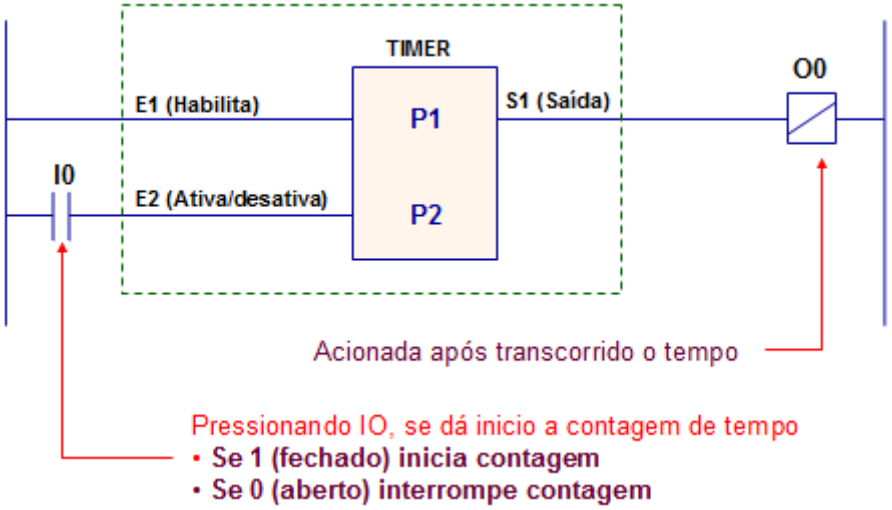


Ladder correspondente para peça pequena



Bloco timer (temporizador)

Este elemento, como o próprio nome diz, tem a finalidade de contar o tempo. Uma vez carregado um determinado período de tempo como parâmetro e tendo a contagem sido habilitada, este valor é decrementado de 10 ms até que chegue a zero, momento em que a saída do bloco é ativada indicando o fim da contagem.



Parâmetros:

P1 – Representa o valor corrente da contagem do temporizador e deve ser obrigatoriamente uma memória inteira (operador M).

P2 - Representa o valor inicial da contagem e deve ser obrigatoriamente uma memória inteira (operador M) ou uma constante inteira (operador K)

Entradas:

E1 – Energizada habilita o bloco temporizador, permitindo a contagem de tempo (se ativado). Se desenergizado o temporizador será desativado.

E2 – Se energizada (1) ativa a contagem do tempo e desenergizada (0) o temporizador fica em seu estado de reset, ou seja, não conta.

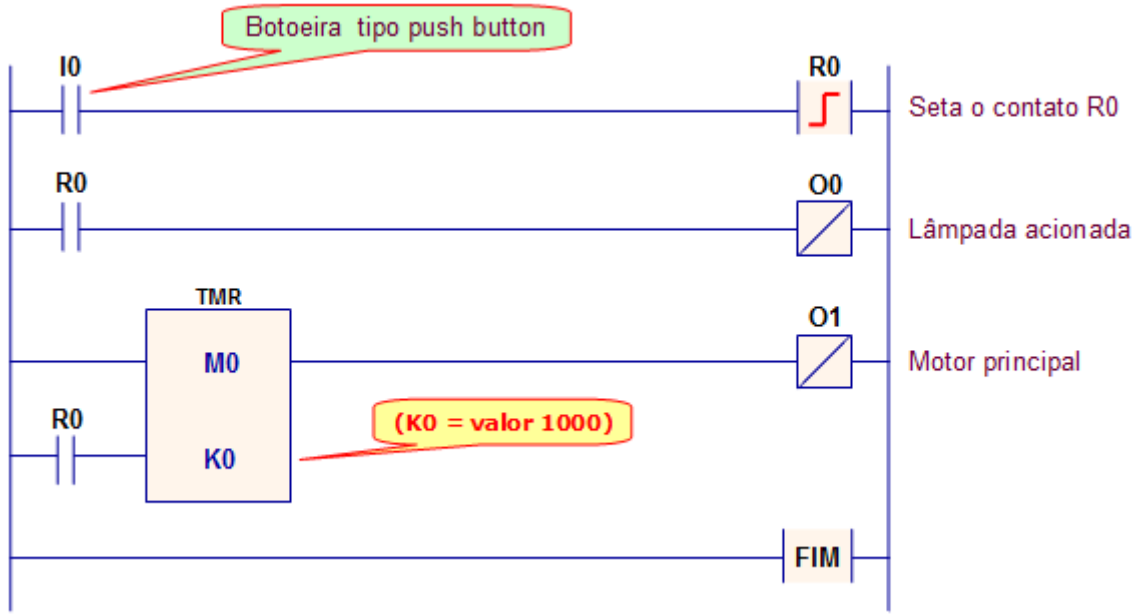
Saída:

S1 – Se ativa (1) indica que o tempo expirou, ou seja, fim da contagem. Se inativa indica que o tempo ainda não terminou ou que o temporizador está desabilitado.

Aplicações práticas do temporizador

1- **(Resolvido)** Deseja-se implementar um sistema em que ao pressionar BL (push button), acenderá imediatamente uma lâmpada indicando que a máquina esta energizada e após 10s ligará o motor principal.

Solução:



2 - Implemente no mesmo programa um botão de desliga, também do tipo push butom.

4- Elabore um programa que ao pressionar BL (push button) ligará uma lâmpada instantaneamente e somente desligará 5 segundos após BD (push button) ter sido pressionado.

5- Elabore um programa que ao pressionar BL ligará 4 motores em seqüência, sendo o primeiro instantaneamente e os demais respeitando um intervalo de 4 segundos. Pressionando BD, todos pararão imediatamente.

6- Construa um programa capaz de energizar 5 motores em sequência M1, M2, M3, M4 e M5 quando BL for pressionado e que desligue também na mesma seqüência, ou seja, M1, M2, M3, M4 e M5. Utilize um intervalo de tempo de 6s.

“Se você fica esperando, tudo o que acontece é que você fica velho.”

(Larry McMurtry)

www.clubedaeletronica.com.br

Referências bibliográficas:

- http://www.plcopen.org/pages/tc1_standards/iec_1131_or_61131/
- http://www.cpdee.ufmg.br/~carmela/NORMA%20IEC%201131.doc
- http://www.software.rockwell.com/corporate/reference/Iec1131/
- http://www.plcopen.org/
- http://www.lme.usp.br/~fonseca/psi2562%20aula%206%20IHM.pdf
- http://www.teses.usp.br/teses/disponiveis/18/18133/tde-11072002-085859/
- http://www.redenet.edu.br/publicacoes/arquivos/20080108_144615_INDU-058.pdf
- http://www.corradi.junior.nom.br/modCLP.pdf
- http://www.cpdee.ufmg.br/~seixas/PaginaII/Download/DownloadFiles/