**FUNDAÇÃO EDSON QUEIROZ**

**[](https://github.com/Judahh/RTPhone/blob/master/Images/Unifor.png)**

**UNIVERSIDADE DE FORTALEZA**

**UNIFOR**

Controlador Lógico Programável

Alunos: Judah Holanda Correia Lima

Kairo

David

Sumário

[Histórico 3](#_Toc373347523)

[Funcionamento 5](#_Toc373347524)

[Principio de Funcionamento 5](#_Toc373347525)

[Estrutura Básica de um CLP 6](#_Toc373347526)

[Entradas e Saídas Discretas 7](#_Toc373347527)

[Módulos de Entrada Discreta 7](#_Toc373347528)

[Módulos de Saída Discreta 7](#_Toc373347529)

[Entradas e Saídas Analógicas 8](#_Toc373347530)

[Módulos de Entrada Analógica 8](#_Toc373347531)

[Módulos de Saída Analógica 8](#_Toc373347532)

[Programação 9](#_Toc373347533)

[LADDER 10](#_Toc373347534)

# Histórico

O CLP foi idealizado pela necessidade de poder se alterar uma linha de montagem sem que tenha de fazer grandes modificações mecânicas e elétricas.

O CLP nasceu praticamente dentro da indústria automobilística, especificamente na Hydronic Division da [General Motors](http://pt.wikipedia.org/wiki/General_Motors" \o "General Motors), em [1968](http://pt.wikipedia.org/wiki/1968" \o "1968), sob o comando do engenheiro Richard Morley e seguindo uma especificação que refletia as necessidades de muitas indústrias manufatureiras.

A idéia inicial do CLP foi de um equipamento com seguintes características resumidas:

* 1. Facilidade de programação;
* 2. Facilidade de manutenção com conceito plug-in;
* 3. Alta confiabilidade;
* 4. Dimensões menores que painéis de Relés, para redução de custos;
* 5. Envio de dados para processamento centralizado;
* 6. Preço competitivo;
* 7. Expansão em módulos;
* 8. Mínimo de 4000 palavras na memória.

Podemos didaticamente dividir os CLP's historicamente de acordo com o sistema de programação por ele utilizado:

* **1ª Geração**: Os CLP's de primeira geração se caracterizam pela programação intimamente ligada ao [hardware](http://pt.wikipedia.org/wiki/Hardware" \o "Hardware) do equipamento. A linguagem utilizada era o [Assembly](http://pt.wikipedia.org/wiki/Assembly" \o "Assembly) que variava de acordo com o processador utilizado no projeto do CLP, ou seja, para poder programar era necessário conhecer a eletrônica do projeto do CLP. Assim a tarefa de programação era desenvolvida por uma equipe técnica altamente qualificada, gravando-se o programa em memória [EPROM](http://pt.wikipedia.org/wiki/EPROM), sendo realizada normalmente no laboratório junto com a construção do CLP.
* **2ª Geração**: Aparecem as primeiras “[Linguagens de Programação](http://pt.wikipedia.org/wiki/Linguagens_de_Programa%C3%A7%C3%A3o" \o "Linguagens de Programação)” não tão dependentes do hardware do equipamento, possíveis pela inclusão de um “Programa Monitor “ no CLP, o qual converte (no jargão técnico, “compila”), as instruções do programa, verifica o estado das entradas, compara com as instruções do programa do usuário e altera o estados das saídas. Os Terminais de Programação (ou maletas, como eram conhecidas) eram na verdade Programadores de Memória EPROM. As memórias depois de programadas eram colocadas no CLP para que o programa do usuário fosse executado.
* **3ª Geração**: Os CLP's passam a ter uma Entrada de Programação, onde um Teclado ou Programador Portátil é conectado, podendo alterar, apagar, gravar o programa do usuário, além de realizar testes ([Debug](http://pt.wikipedia.org/wiki/Debug" \o "Debug)) no equipamento e no programa. A estrutura física também sofre alterações sendo a tendência para os Sistemas Modulares com Bastidores ou Racks.
* **4ª Geração**: Com a popularização e a diminuição dos preços dos microcomputadores (normalmente clones do IBM PC), os CLP's passaram a incluir uma entrada para a comunicação serial. Com o auxílio dos microcomputadores a tarefa de programação passou a ser realizada nestes. As vantagens eram a utilização de várias representações das linguagens, possibilidade de simulações e testes, treinamento e ajuda por parte do software de programação, possibilidade de armazenamento de vários programas no micro, etc.
* **5ª Geração**: Atualmente existe uma preocupação em padronizar protocolos de comunicação para os CLP's, de modo a proporcionar que o equipamento de um fabricante “converse” com o equipamento outro fabricante, não só CLP's, como[Controladores de Processos](http://pt.wikipedia.org/wiki/Controladores_de_Processos" \o "Controladores de Processos), [Sistemas Supervisórios](http://pt.wikipedia.org/wiki/Sistemas_Supervis%C3%B3rios), Redes Internas de Comunicação e etc., proporcionando uma integração a fim de facilitar a automação, gerenciamento e desenvolvimento de plantas industriais mais flexíveis e normalizadas, fruto da chamada Globalização. Existem Fundações Mundiais para o estabelecimento de normas e protocolos de comunicação. A grande dificuldade tem sido uma padronização por parte dos fabricantes.

Com o avanço da tecnologia e consolidação da aplicação dos CLPs no controle de sistemas automatizados, é frequente o desenvolvimento de novos recursos dos mesmos. Com os CLP's temos um aumento na praticidade de processos industriais, não mais necessitando de relés eletromagnéticos, com isso aumentando a velocidade e produtividade de processos industriais

# Funcionamento

## Principio de Funcionamento

Conforme a Figura 1.1 abaixo, o CLP funciona de forma sequencial, fazendo um ciclo de varredura em algumas etapas. É importante observar que quando cada etapa do ciclo é executada, as outras etapas ficam inativas. O tempo total para realizar o ciclo é denominado CLOCK. Isso justifica a exigência de processadores com velocidades cada vez mais altas.

**Início:** Verifica o funcionamento da C.P.U, memórias, circuitos auxiliares, estado das chaves, existência de um programa de usuário, emite aviso de erro em caso de falha. Desativa todas as saídas.

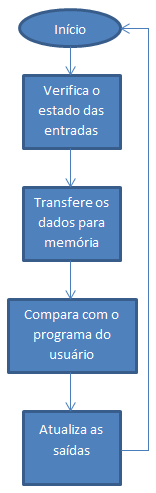


Figura 1.1 – Ciclo de Varredura de um CLP

**Verifica o estado das entradas:** Lê cada uma das entradas, verificando se houve

acionamento. O processo é chamado de ciclo de varredura.

**Campara com o programa do usuário:** Através das instruções do usuário sobre qual ação

tomar em caso de acionamento das entradas o CLP atualiza a memória imagem das saídas.

**Atualiza as saídas:** As saídas são acionadas ou desativadas conforme a determinação da

CPU. Um novo ciclo é iniciado.

## Estrutura Básica de um CLP



Figura 1.2 – Estrutura básica de um CLP

**Fonte de alimentação:** Converte a tensão da rede de 110 ou 220 VCA em +5VCC,

+12VCC ou +24VCC para alimentar os circuitos eletrônicos, as entradas e as as saídas.

**Unidade de processamento:** Também conhecida por CPU, é composta por

microcontroladores ou microprocessadores (Intel 80xx, motorola 68xx, PIC 16xx).

Endereçamento de memória de até 1Mega Byte, velocidades de clock de 4 a 30 MHz,

manipulação de dados decimais, octais e hexadecimais.

**Bateria:** Utilizada para manter o circuito do relógio em tempo real. Normalmente são

utilizadas baterias recarregáveis do tipo Ni - Ca.

**Memória do programa supervisor:** O programa supervisor é responsável pelo

gerenciamento de todas as atividades do CLP. Não pode ser modificado pelo usuário e fica

normalmente em memórias do tipo PROM, EPROM, EEPROM.

**Memória do usuário:** Espaço reservado ao programa do usuário. Constituída por memórias

do tipo RAM, EEPROM ou FLASH-EPROM. Também pode-se utilizar cartuchos de

memória, para proporcionar agilidade e flexibilidade.

**Memória de dados:** Armazena valores do programa do usuário, tais como valores de

temporizadores, contadores, códigos de erros, senhas, etc. Nesta região se encontra

também a memória imagem das entradas – a saídas. Esta funciona como uma tabela virtual

onde a CPU busca informações para o processo decisório.

Os circuitos auxiliares atuam em caso de falha do CLP, são:

POWER ON RESET: desliga todas as saídas assim que o equipamento é ligado,

isso evita que possíveis danos venham a acontecer.

POWER DOWN: monitora a tensão de alimentação salvando o conteúdo das

memórias antes que alguma queda de energia possa acontecer.

WATCH DOG TIMER: o cão de guarda deve ser acionado em intervalos

periódicos, isso evita que o programa entre em “loop”.

## Entradas e Saídas Discretas

### Módulos de Entrada Discreta

São aquelas que fornecem apenas um pulso ao controlador, ou seja, elas têm apenas um estado ligado ou desligado, nível alto ou nível baixo, remontando a álgebra boolena que trabalha com uns e zeros. Alguns exemplos são mostrados na figura 1.2, dentre elas: as botoeiras (1.2a), válvulas eletro-pneumáticas (1.2b) , os pressostatos (1.2c) e os termostatos (1.2d).

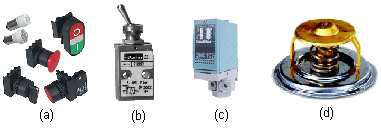
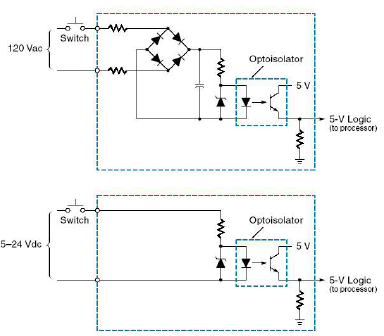


Figura 1.3 – Exemplos de entradas discretas

• 240 Vac, 120 Vac, 24 Vdc ou 5 Vdc.



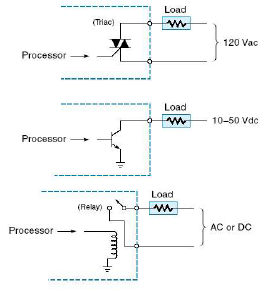
### Módulos de Saída Discreta

São aquelas que exigem do controlador apenas um pulso que determinará o seu acionamento ou desacionamento. Como exemplo têm-se elementos mostrados na figura 1.4: Contadores (1.4a) que acionam os Motores de Indução (1.4b) e as Válvulas Eletropneumáticas(1.4c).

• Alimentam lâmpadas, relés, pequenos motores etc.



Figura 1.4 – Exemplos de saídas discretas



## Entradas e Saídas Analógicas

### Módulos de Entrada Analógica

Como o próprio nome já diz, elas medem as grandezas de forma analógica. Para trabalhar com este tipo de entrada os controladores tem conversores analógico-digitais (A/D). Atualmente no mercado os conversores de 10 bits são os mais populares. As principais medidas feitas de forma analógica são a temperatura e pressão. Na figura 1.3 mostram-se, como exemplo, sensores de pressão ou termopares.

• Contém conversores A/D

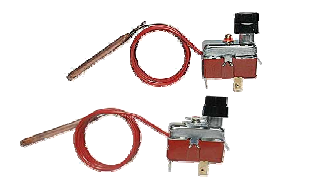
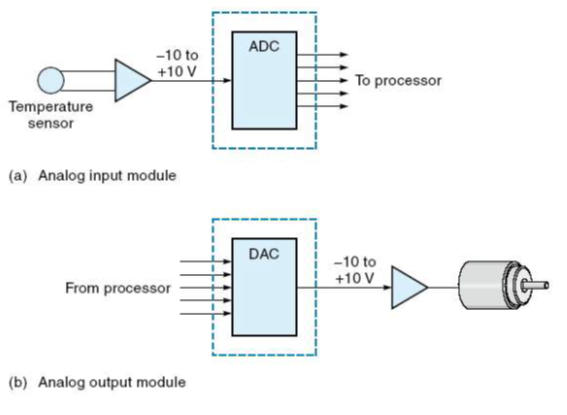


Figura 1.5 – Exemplos de entradas analógicas – Termopares

### Módulos de Saída Analógica

Como dito anteriormente, de forma similar o controlador necessita de um conversor digital para analógico (D/A), para trabalhar com este tipo de saída. Os exemplos mais comuns são: válvula proporcional, acionamento de motores DC, displays gráficos, entre outros.

• Contém conversores D/A



## Programação

De maneira geral, o programa do CLP é um conjunto de expressões booleanas. As expressões são avaliadas uma a uma sequencialmente a cada ciclo de varredura, e o resultado correspondente é armazenado na memória intermediária do CLP. Ao terminar a avaliação, a parte da memória intermediária correspondente às saídas (MIS) é copiada nas saídas.

Linguagens de programação do CLP (Norma IEC 1131-3):

• *Linguagens Gráficas*

• Diagramas de Funções Seqüenciais (*Sequential Function Chart* – SFC)

• Equivalente ao Grafcet

• Diagramas de Contatos (*Ladder Diagram* – LD)

• Diagramas de Blocos de Funções (*Function Block Diagram* – FBD)

• *Linguagens Textuais*

• Lista de Instruções (*Instruction List* – IL)

• Texto Estruturado (*Structured Text* – ST)

### LADDER

#### Elementos Básicos de um programa em Ladder

A linguagem Ladder foi a primeira que surgiu na programação dos Controladores Lógico Programáveis (CLPs), pois sua funcionalidade procurava imitar os antigos diagramas elétricos, utilizados pelos Técnicos e Engenheiros da época. O objetivo era o de evitar uma quebra de paradigmas muito grande, permitindo assim a melhor aceitação do produto no mercado.

Um programa escrito em *Ladder* é constituído por um conjunto de sequências (*rungs*) que são executados sequencialmente pelo autómato. Uma sequência é composta por um conjunto de elementos gráficos limitados à esquerda e à direita por linhas de energia (*power rails*). Os elementos gráficos representam:

- I/O do autómato (interruptores, sensores, indicadores, relés, etc.).

- Blocos funcionais (temporizadores, contadores, etc.).

- Operações aritméticas e lógicas.

- Variáveis internas do autómato.

Cada sequência contém no máximo 7 linhas e 11 colunas que se encontram dividas em na zona de teste, onde se encontram as condições necessárias para a execução das ações, e na zona de atuação, onde se encontram as ações que são executadas dependendo do resultado da zona de teste.



figura 2.1 – Estrutura de uma sequência

O diagrama de contatos (Ladder) consiste em um desenho formado por duas linhas verticais, que representam os polos positivo e negativo de uma bateria, ou fonte de alimentação genérica. Entre as duas linhas verticais são desenhados ramais horizontais que possuem chaves. Estas podem ser normalmente abertas, ou fechadas e representam os estados das entradas do CLP. Dessa forma fica muito fácil passar um diagrama elétrico para linguagem Ladder. Basta transformar as colunas em linhas, como se mostra nas figuras 2.2 e 2.3, para o caso de uma simples partida direta.



Figura 2.1 – Diagrama elétrico de uma partida direta



Figura 2.2 – Diagrama elétrico de uma partida direta

Não se deve esquecer de ligar as botoeiras e contatores, que são os elementos de comando, externamente ao CLP. Para o caso deste comando as ligações elétricas são mostradas na figura 2.3. É importante observar que o relé foi colocado para permitir a existência de dois circuitos diferentes, o de comando composto por uma tensão contínua de 24 V, e o circuito de potência, composto por uma tensão alternada de 220 V.

Ainda no CLP a letra “**I**” significa entrada (Input) e a letra ”**O**” significa saída (Output).

Deve-se lembrar sempre que em painéis elétricos o CLP está inserido na parte de comando do mesmo.

Deve-se lembrar sempre que em painéis elétricos o CLP está inserido na parte de comando do mesmo.



Figura 2.3 – Exemplo de ligação para acionamento de um contator, como no caso do comando direto de um motor de indução trifásico

O mesmo procedimento de conversão pode ser feito com para uma partida de motores com reversão, como mostram as figuras 2.4 e 2.5 a seguir.



Figura 2.4 – Diagrama de comando para uma partida com reversão

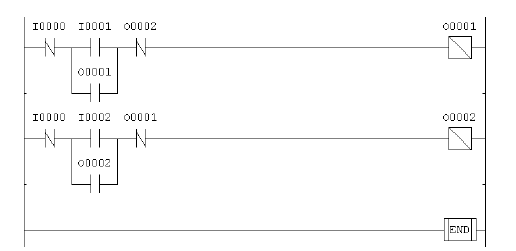


Figura 2.5 – Programa em Ladder para uma partida com reversão

Na figuras 2.4 e 2.5 podem-se observar os elementos básicos de comando, que são os selos dados pelos contatos abertos de O1 e O2, e também os intertravamentos dados pelos contatos fechados de O1 e O2.

Observando os dois exemplos dados, pode-se definir agora os elementos essenciais em uma programação Ladder:



Tabela 2.1 – Elementos Básicos em Ladder



Tabela 2.2 – Elementos Básicos em Ladder (continuação)