



UNIVERSIDADE LUTERANA DO BRASIL
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA



Gonçalo Moacir da Silva Abbad

Controle Lógico Programável Didático

Canoas, 05 de dezembro de 2007



Gonçalo Moacir da Silva Abbad

Controle Lógico Programável Didático

Trabalho de Conclusão de Curso
apresentado ao Departamento de
Engenharia Elétrica da ULBRA como um
dos requisitos obrigatórios para a obtenção
do grau de Engenheiro Eletricista.

Departamento:

Engenharia Elétrica

Professor Orientador:

MSc Eng. Eletr. Dalton Viddor – CREA-RS:
79005-D

Canoas

2007



FOLHA DE APROVAÇÃO

Nome do Autor: Gonçalo Moacir da Silva Abbad

Matrícula: 011100872-7

Título: Controle lógico programável Didático

Trabalho de Conclusão de Curso
apresentado ao Departamento de
Engenharia Elétrica da ULBRA como um
dos requisitos obrigatórios para a obtenção
do grau de Engenheiro Eletricista

Professor Orientador:

Dr Eng. Eletr. Dalton Viddor

CREA-RS: 79005-D

Banca Avaliadora:

MSc Eng. Eletr. Augusto de Mattos

CREA-RS: RS088003

Conceito Atribuído (A-B-C-D):

MSc Eng. Eletr. Paulo César Godoy

CREA-RS: RS116822

Conceito Atribuído (A-B-C-D):

Assinaturas:

Autor

Orientador
Dalton Viddor

Avaliador
Augusto de Mattos

Avaliador
Paulo César Godoy

Relatório Aprovado em: 09/07/2007



DEDICATÓRIA

Dedico aos meus pais...

-



AGRADECIMENTOS

A todos que colaboraram direta ou indiretamente na elaboração deste trabalho, o meu reconhecimento.

Ao Professor Dalton pelo estímulo, dedicação e esforço pessoal proporcionado.

Aos colegas pelas sugestões e observações valiosas.

Aos Professores do curso de engenharia elétrica pelas valiosas contribuições.

À minha mãe, que muito lutou para me educar e criar, e que sempre me deu forças para eu chegar até onde cheguei.

À minha esposa, pelo seu esforço pessoal, ajuda financeira, e motivação nas muitas vezes as quais quis desistir do curso. Sem esse apoio não chegaria a este momento.

Aos meus filhos, por terem suportado minha ausência devido a muitas horas dispensadas a trabalhos e estudo.



EPÍGRAFE

Veritas vos liberabit...

-



RESUMO

AUTOR, Gonçalo Moacir da Silva Abbad. **Controle Lógico Programável Didático: CLPD**. Trabalho de Conclusão de Curso em Engenharia Elétrica - Departamento de Engenharia Elétrica. Universidade Luterana do Brasil. Canoas, RS. 2007.

O trabalho de conclusão consiste em um controlador lógico programável simplificado com fins didáticos. A proposta é construir um controle lógico programável com funções lógicas (or, and, xor, not), operações aritméticas e de comparação. Esta CLP não utilizará a linguagem Ladder para comunicação homem máquina. Mas utilizará valores hexadecimais que estarão associados a funções a serem executadas pelo CLP. Estes valores (códigos em hexadecimal) serão digitados na seqüência que queremos que sejam executadas as operações tal como é feita nas linguagens mnemônicas usadas na programação de alguns modelos de CLP. O programa a ser processado será digitado na própria tela do Hiperterminal ou no bloco de notas do Windows. Para cada valor em hexadecimal corresponderá uma função exercida pelo controlador. Foi desenvolvido um software que fará a interpretação e execução de forma seqüencial dos programas desenvolvidos para o CLP.

O programa é executado no kit ULBEE51 (kit desenvolvido por alunos da ULBRA para desenvolver projetos com o microcontrolador AT89C52 na disciplina de microprocessadores. A linguagem de programação utilizada para programar o kit é a linguagem C e o compilador utilizado é o SDCC (Small Device C Compiler) o programa monitor gravado na memória flash do microcontrolador é o Paulmon2.

Palavras-chave: Controle. Lógico. Programável Didático.



ABSTRACT

Gonçalo Moacir da Silva Abbad. **Logical Controller Programmable Didactic.** Conclusion Work of the Electric Engineering Course - Department of Electric Engineering. Brazilian Lutheran University. Canoas, RS. 2007.

This work consists of a simplified programmable logical controller, with didactical objectives. The purpose is to construct a programmable logical controller with logical functions (or, and, xor, not), arithmetic operations, comparison. This PLC will not use the ladder language for communication between man and machine, but will use hexadecimal values that will be associated to functions to be executed by the PLC. These values (hexadecimal codes) will be typed in the sequence in which is aimed the operations to be executed, just the same way that happens in the mnemonic languages used in the programming of some models of PLC. The program to be processed will be typed in the Hiperterminal screen or in the Windows notepad. For each hexadecimal value will correspond a function exerted by the controller. One software was developed in order to interpret and execute the sequential form of the programs developed to the PLC.

The program is executed in the kit ULBEE51 (developed by students of ULBRA to develop projects with the microcontroller AT89C52, in the discipline of Microprocessors). The programming language used to program the kit is the C language and the compiler used is the SDCC (Small Device C Compiler). The monitor program recorded in the flash memory of the microcontroller is the Paulmon2.

Keywords: **logical. Controller. Programmable. Didactic.**



LISTA DE FIGURAS

Figura 1 Diagrama em blocos de um CLP básico.	07
Figura 2: Identificação dos pinos do fotoacoplador TIL111.....	09
Figura 3: Foto de CLPs Siemens.	10
Figura 4: Foto de um programador manual de CLPM.	11
Figura 5: Representação em Ladder de um circuito motor chave.....	13
Figura 6: Sentido da leitura de um programa qualquer em Ladder.....	14
Figura 7: Símbolos mais comuns utilizados na linguagem Ladder.....	15
Figura 8: Esquema elétrico do circuito de entrada e saída	19
Figura 9: Diagrama interno dos latches (datasheet 74LS373).....	21
Figura 10: Diagrama interno das portas Not e tabela Verdade (datasheet 74LS04).	22
Figura 11: Vista parcial do esquema elétrico do circuito de entrada	22
Figura 12: Vista parcial do esquema elétrico do circuito de saída.....	24
Figura 13: Programa 1 para CLP didática (programa em hexadecimal versus circuito lógico a ser processado).....	38
Figura 14: Programa 2 para CLP didática (programa em hexadecimal versus circuito lógico a ser processado)	39
Figura 15: Programas em Hexadecimal comentados.....	40
Figura 16: Programa com instrução seqüenciador para o CLP didático.....	41
Figura 17: Programa sem instrução seqüenciador para o CLP didático.....	42



LISTA DE TABELAS

Tabela 1 – Descrição das instruções.....	28
--	----



LISTA DE ABREVIATURAS E SIGLAS

Siglas:

ABNT: Associação Brasileira de Normas Técnicas

NEMA: National Electrical Manufacturers Association

Abreviaturas:

CLP: Controlador Lógico Programável

PLC: Programmable Logic Controller

PID: Proporcional Integral Derivativo

LED: Light Emitting Diode

RAM: Random Access Memory

EEPROM: Electrically Erasable Programmable Read Only Memory (E2PROM)

EPROM: Erasable Programmable Read Only Memory

TTL: Transistor Transistor Logic

I/O: Input Output

AD: Analógico Digital

DA: Digital Analógico

SDCC: Small Device C Compiler

USB: Universal Serial Bus



LISTA DE SÍMBOLOS

V – volt (Símbolo de tensão elétrica)

A – Ampere (símbolo de intensidade da corrente elétrica)

W – Watt (símbolo de potência elétrica)

Ω – Ohm (símbolo de resistência elétrica e impedância)



SUMÁRIO

Introdução.....	13
1. Um breve histórico do CLP	15
2. Controlador Lógico Programável.....	18
2.1 Principais blocos que compõem um CLP	19
2.2 Operação Básica do CLP	22
3. Programando o Controlador Lógico Programável.....	24
3.1. Linguagens utilizadas na programação de CLP.....	24
3.2. Linguagem Ladder	25
3.3. Símbolos utilizados na linguagem Ladder	26
3.4. Aplicação dos Controladores Lógicos Programáveis	27
4. O Controlador lógico Programável Didático.....	28
4.1. Hardware	28
4.2 O kit ULBEE51 (características gerais).....	29
4.3. Módulo de entrada e saída.....	31
4.4. Descrição dos circuitos integrados utilizados no projeto	31
4.4.1 O circuito integrado 74LS373.....	31
4.4.2 O circuito integrado LS04	33
4.4.3 Funcionamento módulo de entrada e saída	34
5. Descrição do Software.....	38
5.1. Estrutura do software do controlador lógico programável	39
5.1.1 Módulo Editor	39
5.1.2 Módulo interpretador	39
5.2. Instruções de comparação: EO_{16} , $E1_{16}$, $E2_{16}$	47
5.3. Instruções Aritméticas	48
5.4. Limites a serem respeitados pelo programador	48
5.5. Exemplos de programas a serem executados pelo CLP didático	50
Conclusão.....	55
Referências bibliográficas	56



INTRODUÇÃO

Nos capítulos iniciais (capítulos 1 e 2) desta monografia é feito uma abordagem resumida a respeito do histórico do Controle lógico programável seu funcionamento desde seu surgimento e sua evolução até os dias atuais.

O diagrama de blocos principais de um CLP hipotético é também apresentado no capítulo 1 bem como a função exercida por cada bloco.

No capítulo 2 se faz uma rápida abordagem a respeito das linguagens usadas na programação de CLP's.

A descrição do trabalho de conclusão tem o seu início a partir do capítulo 3 onde é apresentado o hardware que foi em parte projetado (módulo de entrada e saída) e o kit microprocessado Ulbee51 (com as suas principais características). Neste capítulo é detalhado o funcionamento dos módulos de entrada e saída bem como a sua interação com o kit microprocessado.

No capítulo 4 é mostrado a estrutura lógica do CLP Didático onde a lógica utilizada para se desenvolver o programa em C é mostrada com o uso de fluxogramas demonstrando assim o funcionamento do programa editor e do programa interpretador.

No capítulo 5 é mostrada tabela de instruções em hexadecimal que será usada na programação do CLP Didático. A função exercida por cada instrução é



vista em detalhes neste capítulo. Regras que devem ser seguidas para a construção de programas para CLP Didática e diversos exemplos também são encontrados neste capítulo.

Nos apêndices A temos um projeto de fonte desenvolvido para fornecer energia para o CLP Didático e possíveis interfaces de entrada e saída, que possam a vir ser ligados nos terminais de entrada e saída do CLP Didático. No apêndice C tem sugestão de dois circuitos: uma interface óptica de entrada com foto acoplador e uma interface de saída com relé. Permitindo assim o controle cargas com tensões e correntes altas.

No apêndice D encontra-se o esquema elétrico completo do kit Ulbee51.

Para o kit Ulbee51 compilar o programa do CLP Didático (programa em C) deve o ambiente de desenvolvimento (Mide 51) estar configurado para trabalhar com o kit bem como o hiper terminal deve ter seus parâmetros de comunicação ajustados convenientemente para estabelecer comunicação entre o kit e o microcontrolador. Nos apêndices E e F são encontrados dois tutoriais que mostram com efetuar as configurações necessárias.

Finalmente no apêndice G pode ser encontrado o programa em C que edita e interpreta o programa escrito em hexadecimal para a CLP Didática.

No CD ROM que acompanha esta monografia tem o programa em C do clp e programas no bloco de notas em hexadecimal prontos para serem repassados ao clp didático. No arquivo leia-me estão todos os passos para transferir o arquivo do bloco de notas para o CLP Didático.



1. UM BREVE HISTÓRICO DO CLP

A idéia para desenvolver um controle lógico programável surgiu junto ao desenvolvimento dos computadores e inicialmente foi proposta pela GM (General Motors) na década de 1960. O CLP deveria substituir os controles de máquinas industriais compostos de relés que tinham como grande desvantagem o elevado custo e a necessidade de se montar um novo quadro de comando toda vez que fossem alterados os parâmetros de funcionamento da máquina controlada. Nesse procedimento, perdia-se tempo e muito material que, por vezes, não podia ser reaproveitado.

O novo sistema (CLP) deveria agregar preço competitivo em relação aos quadros de comandos com relés, funcionar em ambiente industrial (ambiente propenso à interferência eletromagnética, vibrações, calor, etc.), ser de fácil manutenção, ter dispositivos de entrada e saída de fácil troca e ser durável.

O primeiro dispositivo que atendeu a essas especificações foi desenvolvido pela Gould Modicon em 1969.

Com o surgimento do CLP, a indústria deu um salto em produtividade e qualidade. Para modificar os parâmetros de funcionamento de uma máquina não se necessitava mais montar novos quadros de comandos que satisfizessem as necessidades de controle da máquina. Pois no CLP a mudança se faz através da programação dispensando a montagem de novos quadros de comando. Com o novo



sistema, pode-se programar o funcionamento da máquina e salvar esses programas guardando os mesmos em arquivos virtuais podendo fazer com que a máquina pudesse vir a efetuar o mesmo trabalho em situações futuras necessitando assim somente recarregar a CLP com o programa desejado. Poupando tempo e dinheiro.

Na década de 1970 com o aprimoramento tecnológico dos microprocessadores novas funções foram sendo agregadas ao Controle Lógico Programável. Apresenta-se a seguir um cronograma.

1972 - Funções de temporização e contagem;

1973 - Operações aritméticas, manipulação de dados e comunicação com computadores;

1974 - Comunicação com interfaces homem-máquina;

1975 - Maior capacidade de memória, controles analógicos e controle PID;

1979/80 - Módulos de I/O remotos, módulos inteligentes e controle de posicionamento.

A partir de 1980, os microprocessadores atingiram um alto grau de integração surgiram as micro e miniCLPs (1982). Atualmente, os Controles Lógicos programáveis podem agregar as seguintes características:

-Módulos de I/O de alta densidade (grande número de pontos de I/O por módulo);

-Módulos remotos controlados por uma mesma CPU;

-Módulos inteligentes (coprocessadores que permitem realização de tarefas complexas: controle

-PID, posicionamento de eixos, transmissão via rádio ou modem, leitura de código de barras);



- Softwares de programação em ambiente Windows (facilidade de programação);
- Integração de Aplicativos Windows (Access, Excel, Visual Basic) para comunicação com CLPs;
- Recursos de monitoramento da execução do programa, diagnósticos e detecção de falhas;
- Instruções avançadas que permitem operações complexas (ponto flutuante, funções trigonométricas);
- Scan Time (tempo de varredura) reduzido (maior velocidade de processamento) devido à utilização de processadores dedicados;
- Processamento paralelo (sistema de redundância), proporcionando confiabilidade na utilização em áreas de segurança;
- Pequenos e micros CLPs que oferecem recursos de hardware e de software dos CLPs maiores;
- Conexão de CLPs em rede (conexão de diferentes CLPs na mesma rede, comunicação por meio de Rede Ethernet).
- Sintetizadores de voz.



2. CONTROLADOR LÓGICO PROGRAMÁVEL

Controlador Lógico Programável. Segundo a ABNT (Associação Brasileira de Normas Técnicas), é um equipamento eletrônico digital com hardware e software compatíveis com aplicações industriais.

Segundo a NEMA (National Electrical Manufacturers Association), é um aparelho eletrônico digital que utiliza uma memória programável para armazenar internamente instruções e para implementar funções específicas, tais como lógica, seqüenciamento, temporização, contagem e aritmética, controlando, por meio de módulos de entradas e saídas, vários tipos de máquinas ou processos (Wikipédia, 2007).

2.1. Principais blocos que compõem um CLP:

Na figura 1 é mostrado os principais blocos de um CLP qualquer.

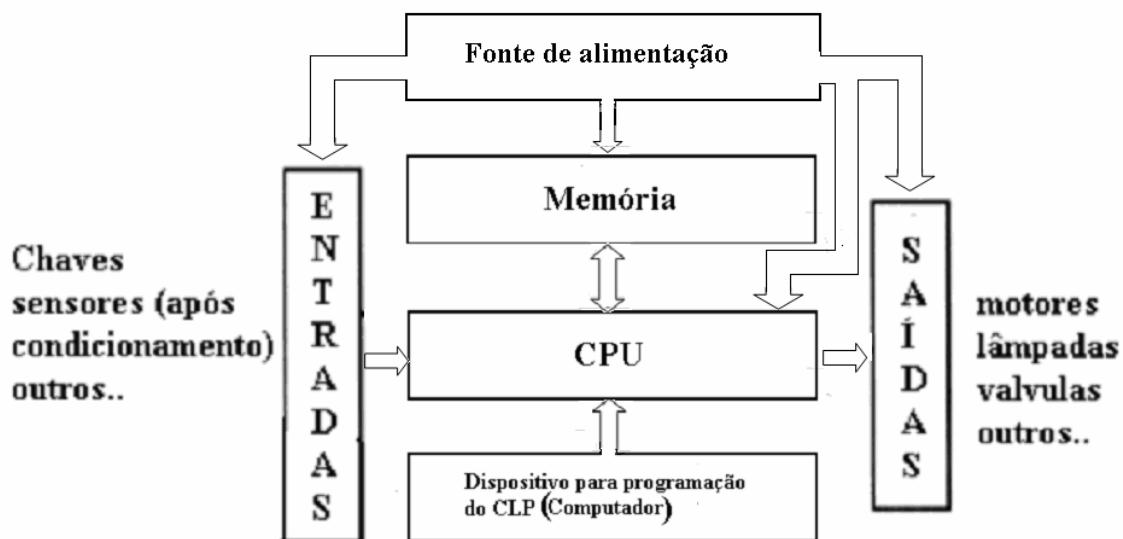


Figura 1: Diagrama em blocos de um CLP básico.

Fonte de Alimentação: responsável pela alimentação fornecida a todos os blocos do Controle lógico programável. A fonte pode ser convencional (com transformador ligado diretamente à rede) ou chaveada. A fonte de alimentação exerce dois papéis fundamentais: dimensiona valores de tensão e corrente para valores adequados aos circuitos alimentados e converte a corrente alternada presente na rede elétrica (corrente alternada é imprópria para alimentar circuitos eletrônicos) em corrente contínua através de um processo chamado retificação (a corrente contínua é o tipo de corrente adequado ao funcionamento de aparelhos eletrônicos digitais e analógicos).

CPU (Central Processing Unit - Unidade Central de Processamento): É o local onde são tomadas as decisões para o controle da máquina ou processo. Na



CPU são processadas todas as operações lógicas e aritméticas do CLP. A CPU é parte integrante do microcontrolador ou microprocessador.

Circuitos/Módulos de I/O (Input/Output - Entrada/Saída): bloco composto pelas entradas e saídas do controle lógico. É por esse bloco que o CLP interage com o meio recebendo informações externas para serem processadas. As informações a serem processadas devem estar presentes em suas entradas e o resultado do processamento será devolvido ao sistema controlado através de suas saídas. As entradas podem ser analógicas ou digitais.

As entradas analógicas são dotadas de conversor Analógico Digital (abreviadamente AD). Nas entradas analógicas podem ser ligados dispositivos que transformem fenômenos físicos (calor, temperatura, pressão, etc.) em sinais elétricos. Como os sinais elétricos provenientes de sensores normalmente são muito tênues e, portanto, são incapazes de gerar os valores elétricos necessários para excitar de forma adequada os AD. É comum o uso de circuitos condicionadores tendo em sua entrada o transdutor que produzirá o sinal a ser condicionado. A saída do condicionador que apresenta já o sinal devidamente condicionado está ligada a entrada do AD.

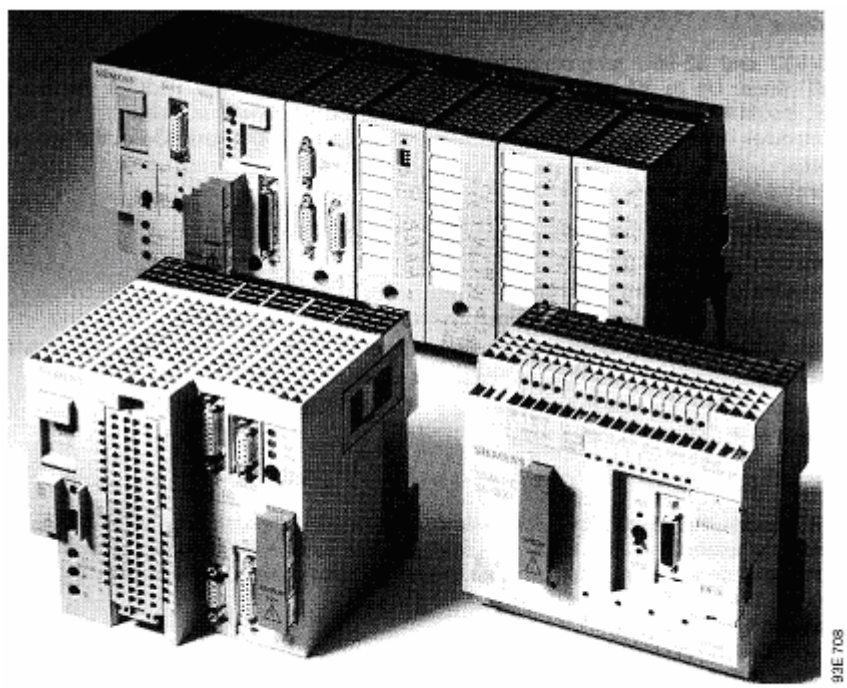
A justificativa do uso do conversor analógico digital nas entradas analógicas justifica-se pelo fato de que a CPU (que é composta por um microcontrolador) é uma máquina digital e, portanto, trabalha exclusivamente com dois níveis lógicos (nível alto ou um e nível baixo ou zero).

As grandezas analógicas apresentam infinitos níveis (valores). Portanto, para o microcontrolador poder trabalhar com esses valores devemos transformar os infinitos valores de uma grandeza analógica em uma série finita de valores digitais. O dispositivo que faz esta conversão é o circuito eletrônico chamado conversor analógico digital.

Entradas digitais estas entradas são destinadas a sinais digitais estes sinais podem ser gerados por dispositivos que tem na sua saída valores digitais ou podem ser produzidos pelo fechamento de uma chave. Por exemplo: chaves fim de curso.

[illegible]

A Figuras 3 a seguir mostra a foto de um CLP comercial:



CLPS Siemens modelos S5-90U, S5-95U e S5-100U
foto retirada do manual destes controladores

Figura 3: Foto de CLPs Siemens²

2.2 Operação Básica do CLP

O CLP é um dispositivo que efetua leituras periódicas (cíclicas) dos *estados*, de suas entradas efetua um processamento dessas informações (programa feito pelo usuário) e após o processamento gera uma saída. Os estados presentes nas entradas do CLP são copiados para uma parte da memória RAM denominada tabela imagem de entrada. Essa informação contida na tabela imagem é processada pela CPU e os valores resultantes do processamento são armazenados em uma parte da memória RAM denominada tabela imagem de saída esses valores são colocados no módulo de saída do CLP acionando os dispositivos de saída. Esse ciclo de busca

² Fonte: Manual do fabricante, Siemens



processamento e saída é repetido enquanto o controle lógico estiver ligado à alimentação e executando um programa.

O programa que processará as informações contidas nas entradas do CLP é feito em um computador ou em um programador manual Figura 4.



Figura 4: Foto de um programador manual de CLPM.³

³ Fonte: www.engprod.ufjf.br/epd_automacao/EPD030_PLCS.pdf



3. PROGRAMANDO O CONTROLADOR LÓGICO PROGRAMÁVEL

O programa a ser executado pela CLP é escrito no computador ou no programador portátil utilizando *software* específico para este fim. Uma vez escrito o programa devemos enviá-lo via porta serial ou USB para o CLP onde deve ser armazenado em uma memória (*RAM, FLASH, E2PROM*) ou, ainda, pode ser enviado a um programador de memória *EPROM* gravando o programa a ser executado pelo CLP nesta memória.

De qualquer forma deve ficar evidente que o computador ou o programador manual (quando usado especificamente para a programação do CLP) pode ser desconectado fisicamente do computador tão logo seja efetuada a transferência do programa para o CLP.

3.1. Linguagens utilizadas na programação de CLP

Uma linguagem de programação é um software no qual escrevemos o programa a ser executado. Nesse software existe um conjunto de regras de sintaxe (que devem ser respeitadas) e instruções que devidamente ordenadas geram um programa se não houver erros de sintaxe o programa será compilado pelo compilador específico e após gravado no CLP pode ser executado.

As linguagens mais utilizadas na programação de CLP são: linguagem Ladder ou linguagem de contatos e linguagens mnemônicas. Outras linguagens que são às vezes usadas para programar CLPs são as Booleana, e linguagem C, entre outras linguagens. A linguagem C é utilizada quando o programa requer cálculos muito complexos para serem programados em linguagem Ladder. A linguagem Ladder é a mais usada das linguagens na programação de CLPs.

3.2. Linguagem Ladder

A linguagem Ladder é uma linguagem gráfica de fácil manuseio, pois foi baseado nos diagramas Ladder elétricos (*diagramas bifilar*). Os diagramas bifilar consistem em duas barras verticais ou barras de alimentação, nas quais são colocados os elementos do circuito (no mínimo um elemento a ser controlado e os elementos de controle).

Na figura 5, são mostrados um diagrama bifilar e seu equivalente programa gráfico em linguagem Ladder e o circuito representado.

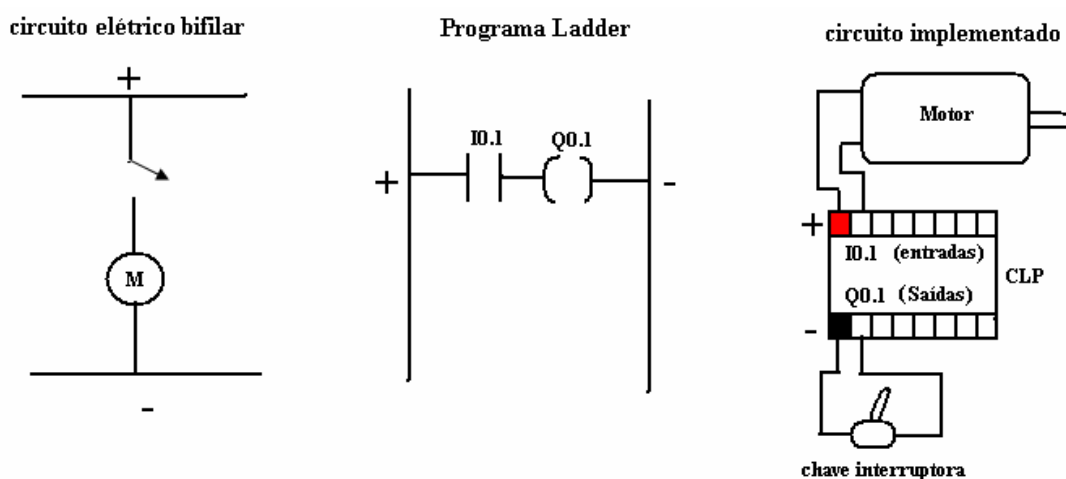


Figura 5: Representação em Ladder de um circuito motor chave

Como pode se ver na Figura 5, a linguagem Ladder utiliza símbolos gráficos para representar entradas e saídas. As entradas e as saídas são identificadas por letras e números (I1.0 e Q1.0) para que dessa forma a CPU do CLP possa identificar as entradas e saídas para processar a informação. As letras e números identificam o endereço (posição de entrada ou saída) e o bit do endereço indicado.

No circuito da Figura 5 ao acionarmos a chave colocamos na entrada I1.0 o *nível lógico* alto o qual conforme programação Ladder escrita em aplicativo específico para este fim fará com que um nível alto apareça na saída Q1.0 ativando o motor elétrico ligado à sua saída. O programa é executado (símbolos são interpretados) da esquerda para direita conforme mostrado na Figura 6 e de cima para baixo quando o programa possuir diversos segmentos.

Programa Ladder

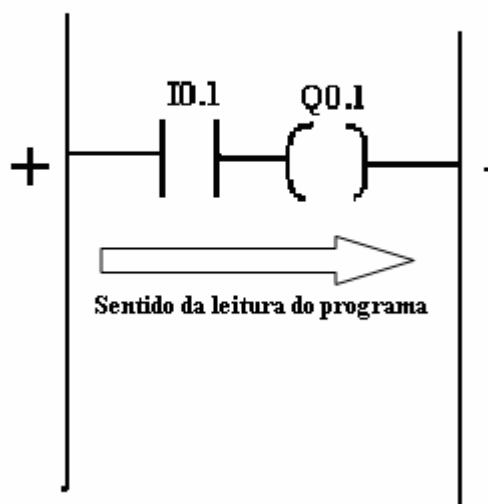


Figura 6: Sentido da leitura de um programa qualquer em Ladder

3.3. Símbolos utilizados na linguagem Ladder

Os símbolos mais comuns usados na linguagem Ladder podem ser vistos na figura 7. Nesta figura pode-se se ver que funções especiais (contadores, timers,

operações matemáticas) são representados em linguagem Ladder por meio de blocos. Na Figura 7, mostra-se como exemplo um bloco representando um contador hipotético.

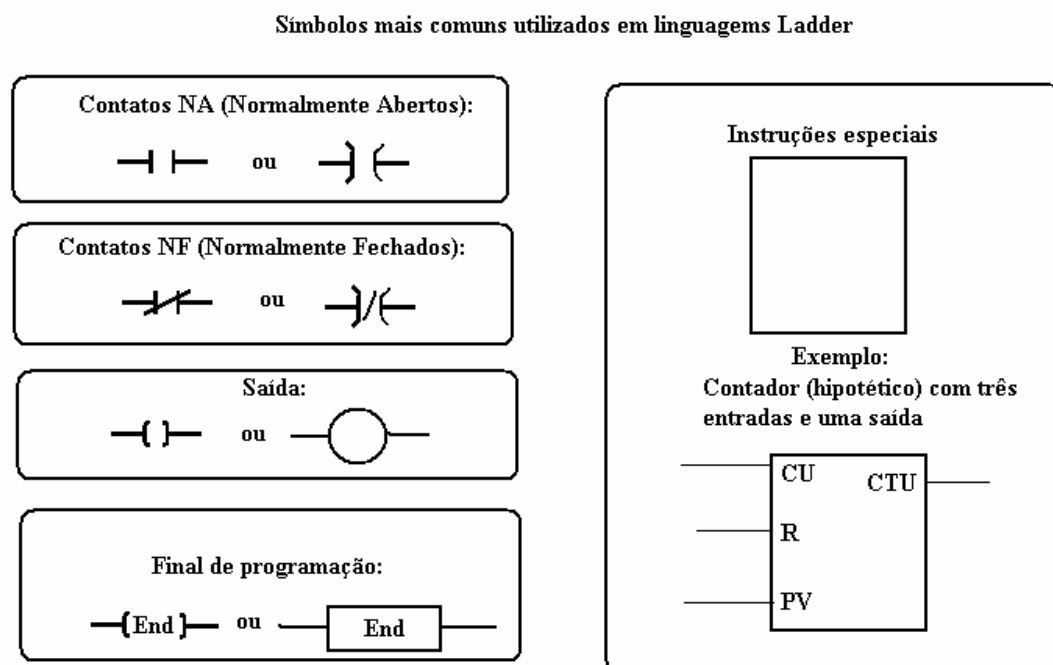


Figura 7: Símbolos mais comuns utilizados na linguagem Ladder.

3.4. Aplicação dos Controladores Lógicos Programáveis

São inúmeras as aplicações para este versátil dispositivo. Ele é usado na indústria controlando processos e máquinas industriais, equipamentos de controle de energia em condomínios inteligentes podendo vir a ter outras aplicações não citadas nesta monografia.



4. O CONTROLADOR LÓGICO PROGRAMÁVEL DIDÁTICO

4.1. Hardware

O Controlador lógico Programável Didático é constituído pelo kit ULBEE51 elaborado para estudos da disciplina de microprocessadores. Usado na elaboração de projetos práticos com o microcontrolador AT 89C52.

Para o desenvolvimento deste projeto foi elaborado um circuito eletrônico dotado de oito entradas e oito saídas. Para assim poder se obter um melhor aproveitamento dos recursos do CLP didático.

Foi projetado dois circuitos eletrônicos que ampliavam a porta P1 do kit. Fazendo com que o kit passe a apresentar duas portas uma de entrada e a outra de saída da informação ambas de 8 bits.

Com o hardware projetado podem ser feitas operações lógicas a nível de bit com até oito entradas e oito saídas ambas programáveis bit a bit. Podem ainda ser feitas operações com byte de comparação (maior, menor, igual) sendo que um dos valores a ser comparado é programado pelo operador do CLP didático e o valor a comparar é introduzido por jogo de chaves que será detalhado a seguir.

As operações aritméticas são feitas usando operadores de um byte. Estes valores serão introduzidos no sistema através de dois jogos de chaves e pelo teclado do computador.



A seguir é feita uma descrição geral dos dispositivos de hardware utilizados no projeto.

As operações aritméticas são feitas usando operadores de um byte. Esses valores serão introduzidos no sistema através de dois jogos de chaves e pelo teclado do computador.

4.2 O kit ULBEE51 (características gerais)

O kit ULBEE51 foi desenvolvido pelo professor e alunos da cadeira de microprocessadores com o intuito de facilitar o aprendizado da disciplina bem como para ser usado em desenvolvimento de projetos com o microcontroladores da família 8051.

O kit comunica-se com o computador via porta serial (aplicativo usado na comunicação serial: Hyper terminal) fazendo uso do programa monitor Paulmon2 gravado na memória flash do microcontrolador do kit (microcontrolador AT89C52). O kit utiliza um conversor RS232/TTL e vice versa para comunicação serial entre o computador e o kit (conversão feita pelo circuito integrado da MAXIM MAX232).

O microcontrolador utilizado no kit apresenta 32 pinos de I/O, estes 32 pinos são divididos em quatro conjuntos de 8 bits que recebem o nome de Portas. As Portas são designadas pelos símbolos P0, P1, P2, P3. Cada pino da Porta possui uma identificação. A posição de cada pino vem após a identificação da Porta:

Exemplo P1_3 pino pertencente a porta P1 bit 3 (quarto pino da porta P1).

As portas P0 e P2 não estão disponíveis como portas de uso geral pois estão sendo usadas no endereçamento e troca de dados com a memória RAM do kit. A porta P1 encontra-se totalmente livre e a porta P3 possui somente quatro pinos livres.



O programa é gravado na memória RAM. O kit ainda é dotado de dois multiplexadores que podem gerar pulsos para I/O mapeados. Permitindo a inclusão de até 16 circuitos adicionais, habilitados pelo pulso de strob.

O esquema elétrico completo do kit pode ver visto no apêndice G desta monografia.

Um maior detalhamento dos recursos internos e pinos do microcontrolador utilizado no kit pode ser conseguido nos livros indicados na bibliografia desta monografia ou junto ao datasheet do microcontrolador AT89C52.

O esquema completo do kit encontra-se no apêndice D e a fonte de alimentação é mostrada no apêndice A (a fonte proposta no apêndice A possui saídas de 9V e 12V para alimentar o kit e os periféricos sugeridos no apêndice C).

4.3. Módulo de entrada e saída

Na figura 8 é mostrado o esquema do módulo de entrada e saída.

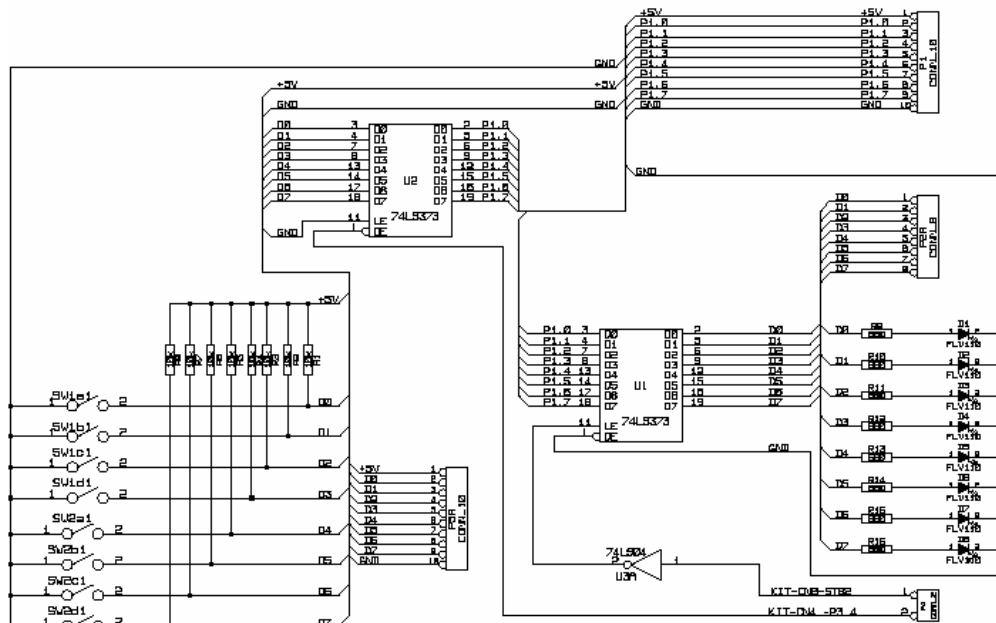


Figura 8: Esquema elétrico do circuito de entrada e saída

4.4. Descrição dos circuitos integrados utilizados no projeto

4.4.1. O circuito integrado 74LS373

O circuito integrado 74LS373 é composto por um conjunto de oito *latches* tipo transparente com saídas tri-state. Os latch recebem o nome de transparente pois, transfere e memoriza os últimos estados (níveis 0 ou 1) presentes nas oito entradas dos latch. Para cada entrada temos um latch gerando uma saída.

Estes circuitos integrados possuem dois pinos de alimentação que devem ser alimentados com uma tensão estável de $+5V \pm 10\%$ (fonte datasheet Fairchild). O valor de alimentação dos circuitos integrados desta tecnologia (tecnologia TTL)



necessitam de alimentação estável aceitando uma pequena tolerância (tolerância de 10%) para poderem funcionar adequadamente.

Possuem dois pinos de controle, o pino 1 coloca todas as saídas em alta impedância basta colocar nível alto no pino 1.

O outro pino de controle é o pino 11 quando este pino está com nível baixo ele desabilita a atualização dos níveis presentes nas saídas dos latch, desta forma ficam presentes nas saídas os níveis anteriores (níveis que estavam presentes nas entradas antes da desabilitação do chip). Na figura 9 (figura extraída do datasheet do fabricante Motorola) temos o desenho identificando a arquitetura interna deste circuito integrado, identificação dos seus pinos e tabela indicando os estados das saídas de acordo com os níveis presentes nos pinos de controle (pinos 1 e 11).

Na tabela encontrada na figura 9 foi usado pelo fabricante as letras L (indicando presença de nível baixo no pino) e a letra H (indicando presença de nível alto no pino) o X é indicação de que pode estar presente um nível alto ou baixo no pino ou seja tanto faz o nível presente no pino marcado com o X que a saída será gerada conforme indicada na tabela.

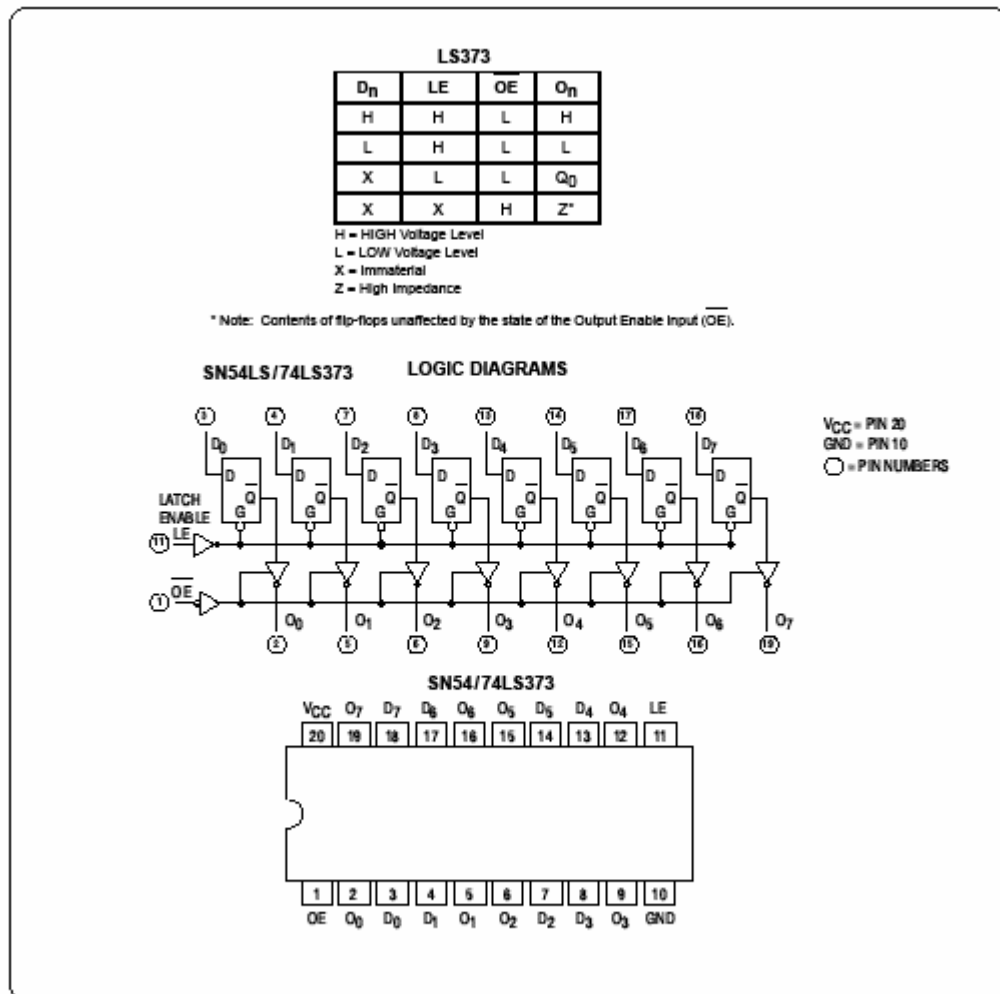


Figura 9: Diagrama interno dos latches.⁴

4.4.2. O circuito integrado 74LS04

Este circuito integrado é constituído por 6 portas inversoras. A sua pinagem e estrutura interna são mostrados na Figura 10 (desenho extraído do datasheet do fabricante Fairchild).

⁴ Vista parcial do Datasheet Fairchild circuito integrado 74LS373

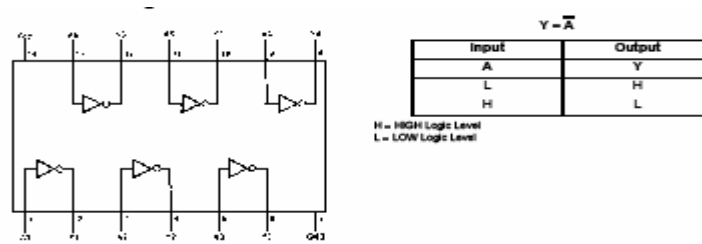


Figura 10: Diagrama interno das portas Not e tabela Verdade.⁵

4.4.3. Funcionamento do módulo de entrada e saída

Circuito de entrada

O circuito de entrada pode ser visto na vista parcial do esquema do módulo de entrada e saída onde temos o circuito de entrada (Figura 11):

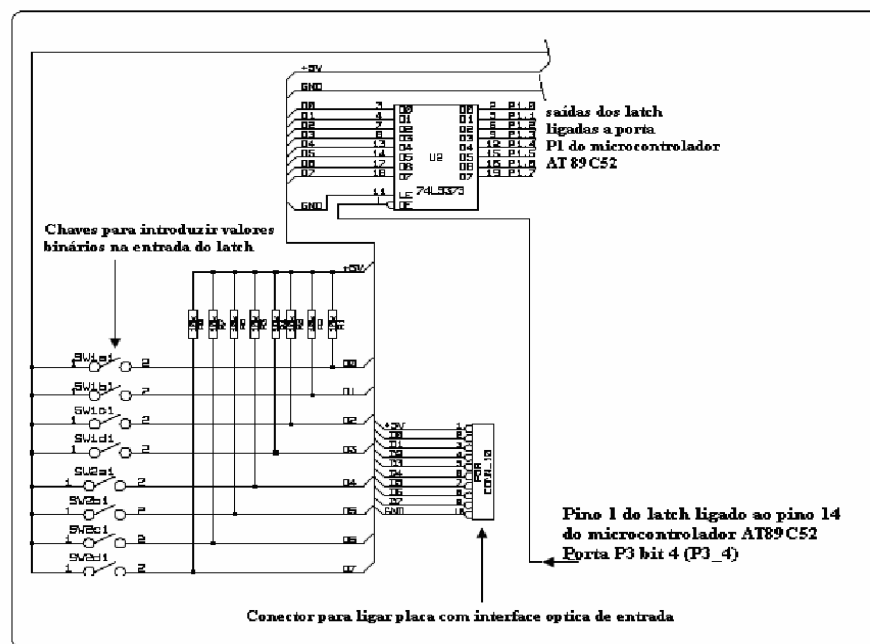


Figura 11: Vista parcial do esquema elétrico do circuito de entrada

⁵ Vista parcial Datasheet Fairchild circuito integrado 74LS04



O circuito de entrada é composto por oito chaves que tem o objetivo inicial de testar as funções do CLP didático.

Na figura 11 podemos ver que este estágio é composto por um conector onde pode ser acoplado a interface de entrada óptica (a interface de entrada óptica tem o seu circuito sugerido no apêndice C) possui também um conjunto de latch's (74LS373).

O pino de controle dos *latch* pino 1 (pino que coloca as saídas em alta impedância) está ligado ao kit ULBEE51 na porta P3 (bit P3_4). O microcontrolador controlado pelo software coloca nível baixo neste pino autorizando a operação de leitura da(s) entrada(s) selecionadas pelo software. Estabelecendo a comunicação da saída dos *latch* com a porta P1 que fará a leitura dos valores presentes na porta P1. Tão logo o processo de leitura seja concluído o software determinará ao microcontrolador a mudança de nível presente no pino 1 dos *latch* colocando suas saídas em alta impedância.

Circuito de Saída

O circuito de saída pode ser visto na vista parcial do esquema do módulo de entrada e saída (Figura 12):

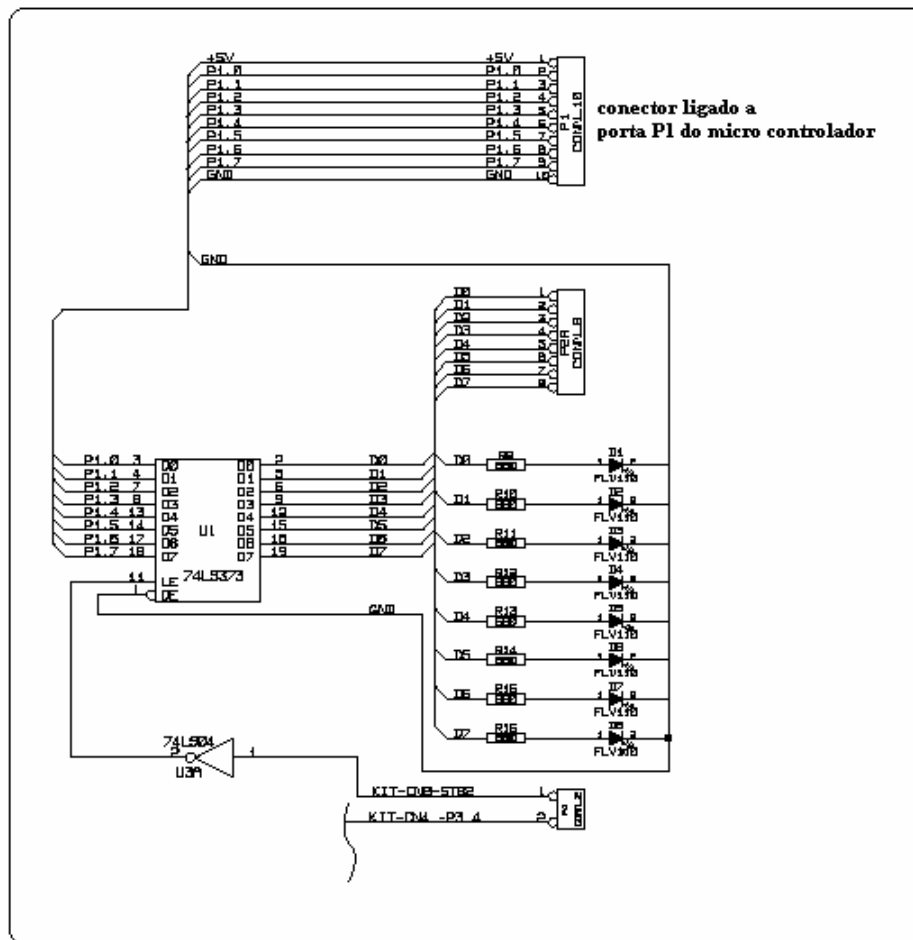


Figura 12: Vista parcial do esquema elétrico do circuito de saída.

Neste circuito de saída é usado o *latch* 74LS373 onde temos oito leds ligados nas oito saídas para assim podermos monitorar os níveis de saída do Controlador Lógico Programável didático.

Neste circuito temos ligado ao pino 11 uma porta inversora com o objetivo de inverter o pulso proveniente do kit ULBEE51 (pulso de *strob*), pois o nível presente na entrada da porta inversora (sem a interferência do microcontrolador) é sempre nível alto pois os pulsos de *strob* não são produzidos por portas I/O do microcontrolador mas sim por uma saída de um multiplexador que em repouso apresenta nível lógico alto, mudando para nível lógico baixo quando a saída



selecionada por três pinos de seleção (pinos ligados ao microcontrolador) habilitarem a referida saída no kit.

Na saída do multiplexador do kit surgirá um pulso de *strobe* que é habilitado via software e produz um breve pulso com nível baixo.

Como os *latch* somente atualizam as suas saídas quando o seu pino 11 estiver em nível alto devemos efetuar a inversão do pulso de *strobe* permitindo assim que o *latch* atualize as suas saídas somente quando for dado um novo pulso de *strobe* pulso com nível alto. Após o pulso de *strobe* ser dado no pino 11 os níveis presentes nas entradas dos *latch* são transferidos para as saídas. Estes níveis são mantidos nas saídas até novo pulso de *strobe*. Com o uso da porta inversora o nível de repouso da saída do multiplexador passará a ser nível baixo desautorizando os *latch* de atualizarem as suas saídas na ausência de um pulso de *strobe*.



5. DESCRIÇÃO DO SOFTWARE

O software desenvolvido para fazer com que o kit ULBEE51 opere como uma CLP didática foi desenvolvido em linguagem C, sendo efetuada a chamada de algumas macros contidas no programa monitor (Paulmon2) em assembler.

As macros utilizadas no programa são:

pm2_pstr(); Escreve na tela do Hiper Terminal mensagens colocadas entre aspas duplas escritas dentro dos parênteses.

pm2_newline; Muda de linha e coloca o cursor no início da linha na tela do Hiper Terminal.

ghex; Coloca o cursor no Hiper Terminal no início da linha e aguarda a digitação de valores com dois dígitos em hexadecimal. O valor digitado é copiado para o acumulador (registrador de uso geral ACC do microcontrolador).



5.1. Estrutura do software do controlador lógico programável

O software é dividido basicamente em dois módulos: módulo editor, módulo interpretador.

5.1.1. Módulo editor

O módulo editor é a parte do software onde ocorre a comunicação serial do kit com o Hiper Terminal. Nesse módulo é feita a escrita do programa através de código em hexadecimal. Este módulo é constituído basicamente por um vetor que é carregado com as informações digitadas pelo operador do CLP didático. Este módulo foi previsto para carregar programas com no máximo de 255 linhas. O fluxograma do módulo editor pode ser visto no Apêndice F.

5.1.2. Módulo interpretador

O módulo interpretador é o módulo responsável pelo processamento seqüencial dos códigos (instruções) digitados no módulo editor. Nesse módulo é efetuada a leitura das entradas e são geradas também as saídas do sistema. Executa um loop infinito refazendo leituras do programa enquanto kit não for resetado. O fluxograma do módulo interpretador pode ser visto no Apêndice G. E o programa completo em C (ambos os módulos) pode ser visto na integra no Apêndice H.



INSTRUÇÕES DO CONTROLADOR LÓGICO PROGRAMÁVEL DIDÁTICO

Na Tabela 1 é mostrado o código em hexadecimal e a sua respectiva instrução que será utilizada pelo módulo interpretador (instruções).

Tabela 1: Descrição das instruções

Código em hexadecimal	Instrução
00	Reset
01	Lê a entrada 1
02	Lê a entrada 2
03	Lê a entrada 3
04	Lê a entrada 4
05	Lê a entrada 5
06	Lê a entrada 6
07	Lê a entrada 7
08	Lê a entrada 8
09	Leitura Byte entradas
10	Escrita (0 ou 1) na saída 1
11	Escrita (0 ou 1) na saída 2
12	Escrita (0 ou 1) na saída 3
13	Escrita (0 ou 1) na saída 4
14	Escrita (0 ou 1) na saída 5
15	Escrita (0 ou 1) na saída 6
16	Escrita (0 ou 1) na saída 7
17	Escrita (0 ou 1) na saída 8
18	Escrita (Byte de saída)
38	Seqüenciador (saída imediata)
45	Encerra programa início da execução
A0	And (bit)
A1	And (byte)
B0	Or (bit)
B1	Or (byte)
C0	Exor (bit)
C1	Exor (byte)
D0	Not (bit)
E0	Maior
E1	Menor
E2	Igualdade
E3	Soma
E4	Subtração
E5	Multiplicação
E6	Divisão
E7	Resto da divisão



00 – Reset se o interpretador encontrar alguma linha do programa sem nenhum valor ele reinicia a leitura do programa este comando foi criado para caso o programa se perder ou varrer alguma linha sem programação (falha na programação).

01₁₆ – Leitura somente da entrada 1 as demais entradas ficam desabilitadas

02₁₆ – Leitura somente da entrada 2 o mesmo ocorrendo com as instruções 03₁₆ a 08₁₆ cada uma destas instruções habilitará a leitura de sua respectiva entrada.

09₁₆ – Leitura do Byte de entrada (funções lógicas de byte, aritméticas e de comparação).

10₁₆ – Habilita a escrita somente da saída 1 apresentando o resultado de uma operação lógica (0 ou 1) que esteja sendo processado no CLP

11₁₆ – Habilita a escrita somente da saída 2 apresentando o resultado de uma operação lógica (0 ou 1) que esteja sendo processado no CLP

12₁₆ – Habilita a escrita somente da saída 3 apresentando o resultado de uma operação lógica (0 ou 1) que esteja sendo processado no CLP

13₁₆ – Habilita a escrita somente da saída 4 apresentando o resultado de uma operação lógica (0 ou 1) que esteja sendo processado no CLP

14₁₆ – Habilita somente a saída 5 apresentando o resultado de uma operação lógica (0 ou 1) que esteja sendo processado no CLP

15₁₆ – Habilita a escrita somente da saída 6 apresentando o resultado de uma operação lógica (0 ou 1) que esteja sendo rodado no CLP

16₁₆ – Habilita a escrita somente da saída 7 apresentando o resultado de uma operação lógica (0 ou 1) que esteja sendo processado no CLP



17₁₆ – Habilita a escrita somente da saída 8 apresentando o resultado de uma operação lógica (0 ou 1) que esteja sendo processado no CLP

18₁₆ – Habilita a escrita em todas as saídas apresentando o resultado de uma operação lógica (0 ou 1) que esteja sendo processado no CLP. É utilizada em operações que necessita-se de uma saída de byte.

38₁₆ – Seqüenciador esta instrução habilita as saídas na seqüência programada. Sem esta instrução o programa só atualizará as saídas quando encontrar a instrução (45₁₆). A função da instrução (38₁₆) é de renovar o nível da(s) saída(s) que estiver após esta instrução. Ficará mais clara a função desta instrução após os exemplos de programas apresentados nesta monografia.

45₁₆ – Esta instrução indica o final da programação para o programa editor indicando também o início da execução do programa. Para o programa interpretador a ocorrência desta instrução é interpretada como final do programa reiniciando a leitura do programa (realiza o reset do programa interpretador)

A0₁₆ – Esta instrução realiza a operação lógica And a nível de bit. Quando esta for a primeira instrução do programa. O programa interpretador solicitará que o programador defina duas entradas para operar a lógica. Quando não for a primeira instrução do programa. O programa interpretador solicitará que o programador defina apenas uma entrada para efetuar a operação lógica entre a entrada programada e o valor da operação anterior.

A1₁₆ – Esta instrução realiza a operação lógica And a nível de byte. Nesta instrução deve o programador inserir um valor para efetuar a operação lógica com a entrada de byte.

B0₁₆ – Esta instrução realiza a operação lógica OR a nível de bit. Quando esta for a primeira instrução do programa. O programa interpretador solicitará que o programador defina duas entradas para operar a lógica. Quando não for a primeira instrução do programa. O programa interpretador solicitará que o programador defina apenas uma entrada para efetuar a operação lógica entre a entrada programada e o valor da operação anterior.



B1₁₆ – Esta instrução realiza a operação lógica Or em nível de byte. Nesta instrução deve o programador inserir um valor para efetuar a operação lógica com a entrada de byte.

C0₁₆ – Esta instrução realiza a operação lógica Exor em nível de bit, quando esta for a primeira instrução do programa. O programa interpretador solicitará que o programador defina duas entradas para operar a lógica. Quando não for a primeira instrução do programa, o programa interpretador solicitará que o programador defina apenas uma entrada para efetuar a operação lógica entre a entrada programada e o valor da operação anterior.

C1₁₆ – Esta instrução realiza a operação lógica Exor em nível de byte. Nesta instrução deve o programador inserir um valor para efetuar a operação lógica com a entrada de byte.

D0₁₆ – Esta instrução é a instrução not em nível de bit e opera uma entrada por vez.

E0₁₆ – Esta instrução realiza a comparação entre um valor digitado pelo programador (byte) com outro valor presente nas entradas. Verifica se o valor presente na entrada é maior que o valor programado gerando uma saída (esta instrução gera somente dois valores 0 se não for igual ou 1 se for igual) o valor (0 ou 1) é depositado em uma variável de uso geral.

E1₁₆ – Esta instrução realiza a comparação entre um valor digitado pelo programador (byte) com outro valor presente nas entradas. Verifica se o valor presente na entrada é menor que o valor programado gerando uma saída (esta instrução gera somente dois valores 0 se não for igual ou 1 se for igual) o valor (0 ou 1) é depositado em uma variável de uso geral.

E2₁₆ – Esta instrução realiza a comparação entre um valor digitado pelo programador (byte) com outro valor presente nas entradas. Verifica se o valor presente na entrada é igual ao valor programado gerando uma saída (esta instrução gera somente dois valores 0 se não for igual ou 1 se for igual) o valor (0 ou 1) é depositado em uma variável de uso geral.



E3₁₆ – Esta é uma instrução de soma, quando for esta a primeira instrução. O programa editor solicitará a instrução de entrada para byte e logo a seguir solicita um valor ao programador (valor entre 00₁₆ a FF₁₆). Se esta não for a primeira instrução, o programa editor solicitará um valor ao programador este valor será somado ao valor resultante de uma operação anterior o resultado será depositado em uma variável de uso geral.

E4₁₆ - Esta é uma instrução de subtração, quando for esta a primeira instrução do programa. O programa editor solicitará a instrução de entrada para byte e logo a seguir solicita um valor ao programador (valor entre 00₁₆ a FF₁₆). Se esta não for a primeira instrução, o programa editor solicitará um valor ao programador este valor será subtraído do valor resultante de uma operação anterior o resultado desta operação será depositado em uma variável de uso geral.

E5₁₆ – Esta é uma instrução de multiplicação, quando for esta a primeira instrução do programa. O programa editor solicitará a instrução de entrada para byte e logo a seguir solicita um valor ao programador (valor entre 00₁₆ a FF₁₆). Se esta não for a primeira instrução, o programa editor solicitará um valor ao programador este valor será multiplicado pelo valor resultante de uma operação anterior o resultado desta operação será depositado em uma variável de uso geral.

E6₁₆ – Esta é uma instrução de divisão, quando for esta a primeira instrução do programa. O programa editor solicitará a instrução de entrada para byte e logo a seguir solicita um valor ao programador (valor entre 00₁₆ a FF₁₆). Se esta não for a primeira instrução, o programa editor solicitará um valor ao programador este valor será o divisor do valor resultante de uma operação anterior o resultado desta operação será depositado em uma variável de uso geral.

E7₁₆ – Com esta instrução podemos resgatar o valor do resto da divisão efetua por E6₁₆ (esta instrução na verdade aponta para uma variável que armazena o valor do resto de uma divisão)

IMPORTANTE: Se após a ocorrência de uma saída (habilitação de uma saída) o programa continuar sendo digitado, a próxima instrução será uma



operação que solicitará novamente duas entradas ou uma entrada e um valor (depende da operação que segue após a saída), pois volta a ser o primeiro operador de uma nova operação.

Precedência de operadores

Para o CLP didático operar de forma adequada deve-se observar a ordem em que devem ser colocadas as operações e os operadores para assim obterem-se os resultados desejados.

- As operações devem sempre sempre sempre anteceder a(s) entrada(s) e valores que serão processadas pela operação.

Instruções de operações: $A0_{16}$, $A1_{16}$, $B0_{16}$, $B1_{16}$, $C0_{16}$, $C1_{16}$, $E0_{16}$, $E1_{16}$, $E2_{16}$, $E3_{16}$, $E4_{16}$, $E5_{16}$, $E6_{16}$, $E7_{16}$;

Instruções de I/O (operadores): 01_{16} , 02_{16} , 03_{16} , 04_{16} , 05_{16} , 06_{16} , 07_{16} , 08_{16} , 09_{16} , 10_{16} , 11_{16} , 12_{16} , 13_{16} , 14_{16} , 15_{16} , 16_{16} , 17_{16} , 18_{16}

Instruções especiais:

38_{16} seqüenciador deve anteceder a saída que queremos apresentar o valor imediato.

45_{16} Esta instrução deve ser usada somente no final da programação, pois, uma vez digitada, ela sai da programação (programa editor) para a execução (programa interpretador) de forma automática.

Instruções de operações com byte: $A1_{16}$, $B1_{16}$, $C1_{16}$, $E0_{16}$, $E1_{16}$, $E2_{16}$

$A1_{16}$ – operação and byte;

09_{16} – entrada de um byte (operador 1);

$XX-XX$ pode ser qualquer valor entre 00 a FF_{16} (operador 2);



18₁₆ - habilita todas as saídas (byte);

45₁₆ - encerra programação passa para a execução.

A ordem mostrada acima (1ª operação, 2ª entrada, 3º valor) é válida para os operadores A1₁₆, B1₁₆, C1₁₆, E0₁₆, E1₁₆, E2₁₆, E3₁₆, E4₁₆, E5₁₆, E6₁₆. Quando essas operações vierem no início ou após uma instrução de saída a ordem dos fatores é a mostrada acima. Porém, se essas operações forem colocadas após um conjunto de instruções, essas operações somente solicitaram um valor. O segundo valor a operar ou comparar será o valor da operação anterior (depositado em variável de uso geral). Exemplo:

B1₁₆ - operação or byte;

09₁₆ - entrada de um byte;

XX - XX pode ser qualquer valor entre 00 a FF₁₆.

A1₁₆ - operação and byte (note que antes deste and há linhas de programa e não é antecedido por uma instrução de saída);

XX- XX pode ser qualquer valor entre 00 a FF₁₆;

18₁₆ - habilita as saídas (byte);

45₁₆ - encerra programação passa para a execução.

Nota-se que foi suprimido no programa acima a entrada 09₁₆ solicitando somente um valor programado pelo programador (XX). Esse valor será processado com o resultado da operação anterior.



5.2. Instruções de comparação: $E0_{16}$, $E1_{16}$, $E2_{16}$

$E0_{16}$ – compara se um valor programado é maior que o valor presentes nas entradas ou se é maior que o resultado de uma operação anterior. Se o resultado for verdadeiro, o valor é 1, se for falso, o valor é zero.

$E0_{16}$ – a comparação verifica se o valor programado é maior do que o valor presente na entrada byte.

09_{16} – entrada de um byte.

XX - XX pode ser qualquer valor entre 00 a FF_{16} . Este valor será comparado ao valor das chaves.

10_{16} – habilita a saída 1, se o valor programado for maior que o valor introduzido pelas chaves. Nesta saída aparecerá o nível lógico alto, caso contrário, é nível lógico baixo.

45_{16} – encerra programação passa para a execução.

$B1_{16}$ – operação or byte.

09_{16} – entrada de um byte.

XX – XX pode ser qualquer valor entre 00 a FF_{16} .

$E0_{16}$ – Operação maior (note que antes desta instrução de comparação tem linhas de programa e não é antecedido por uma instrução de saída).

XX – O próximo valor que sucede o operador não será uma instrução, mas um valor a comparar.

10_{16} – habilita a saída 1, se o valor programado (XX) for maior que o resultado da operação anterior colocará a saída 1 o nível lógico alto, caso contrário, o nível lógico nesta saída será baixo.



45₁₆ – Encerra a programação passa para a execução.

As mesmas regras se aplicam para as instruções E1 (menor) e E2 (igual).

5.3. Instruções Aritméticas

Instruções Aritméticas: E3₁₆, E4₁₆, E5₁₆, E6₁₆, E7₁₆.

5.4. Limites a serem respeitados pelo programador

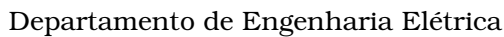
As instruções aritméticas são operações que operam com valores de um byte a resposta da operação (saída) deve ser representada por valores que possam ser representados por oito bits, ou seja, a resposta da operação não pode ter valor superior a 255₁₀. Deve-se ter o cuidado com operações aritméticas encadeadas, pois o valor máximo que pode ser representado pelo CLP didático não pode superar um byte.

Cálculos internos não devem superar os seguintes limites -32768₁₀ a +32767₁₀ impostos a variáveis tipo inteira (int).

Valores negativos não foram definidos para efeito de introdução de valores de entradas, porém nada impede que se obtenham resultados de operações internas valores negativos.

Resultados de operações aritméticas com valor negativo colocados na saída resultaram valores errados o mesmo teste foi feito com a calculadora científica virtual do Windows, obtendo-se os seguintes resultados. Cálculo testado:

-



1111101,



instrução de subtração o valor presente na entrada será subtraído pelo valor programado. Se não vier como primeira instrução, o valor do resultado da operação anterior será subtraído pelo valor programado.

5.5. Exemplos de programas a serem executados pelo CLP didático

Na Figura 13 temos um programa em nível de bit que implementa os circuitos lógicos mostrados na mesma Figura.

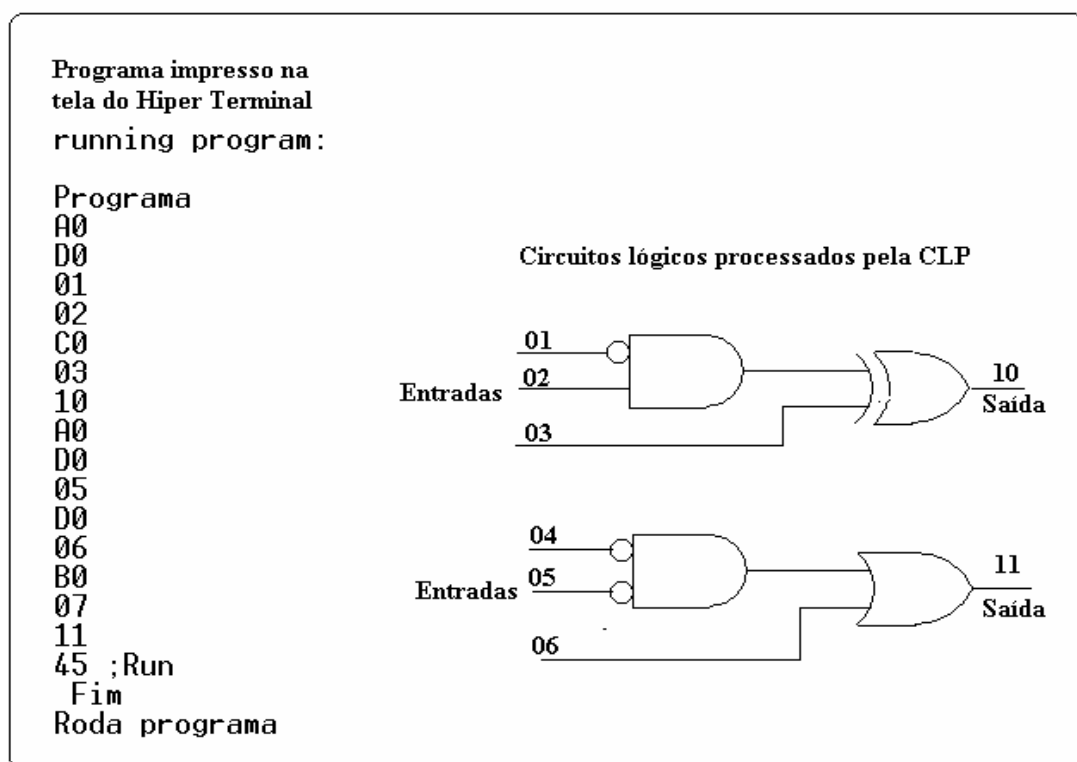


Figura 13: Programa 1 para CLP didática (programa em hexadecimal versus circuito lógico a ser processado).



Na figura 14 temos um segundo programa em nível de bit que implementa o circuito lógico mostrado.

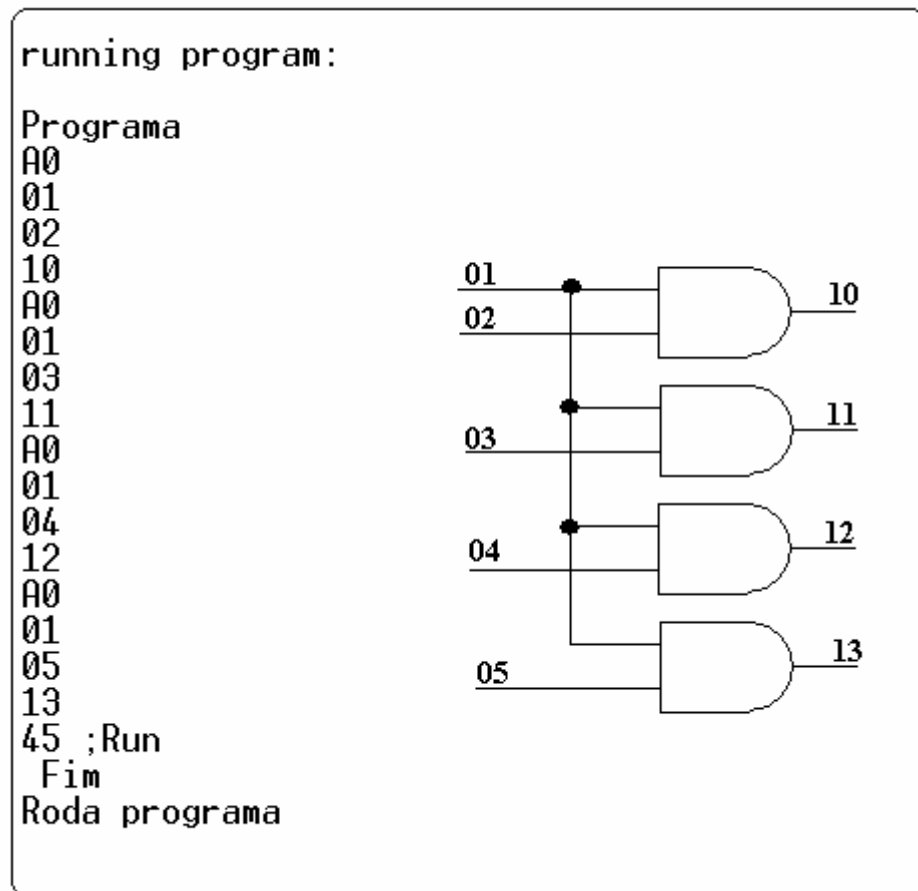


Figura 14: Programa 2 para CLP didática (programa em hexadecimal versus circuito lógico a ser processado).

Na Figura 15 temos dois pequenos programas que demonstram o uso da instrução maior e menor.

<pre>running program: Programa E0 → comparação (maior) 09 → Habilita a entrada de byte 0A → Valor programado = 10 12 → Habilita saída 2 45 ;Run Fim Roda programa</pre>	<pre>running program: Programa E1 → comparação (menor) 09 → Habilita a entrada de byte 0A → Valor programado = 10 12 → Habilita saída 2 45 ;Run Fim Roda programa</pre>
--	--

Figura 15: Programas em Hexadecimal comentados.

A seguir é dado um exemplo de um programa que faz uso da instrução seqüenciador (38₁₆).

Na Figura 16, temos o programa usando o seqüenciador. Deve-se colocar nível lógico alto nas entradas 01₁₆ até a entrada 05₁₆ para se obter a seqüência mostrada no desenho que representa as oito saídas com leds. Na Figura 16 são mostrados quatro momentos das saídas. O desenho das saídas com leds foi repetido quatro vezes uma vez para cada momento. Os leds em vermelho claro representam que o led está ligado indicando nível alto nesta saída, os demais leds em marrom representam leds desligados ou saída em nível lógico baixo.

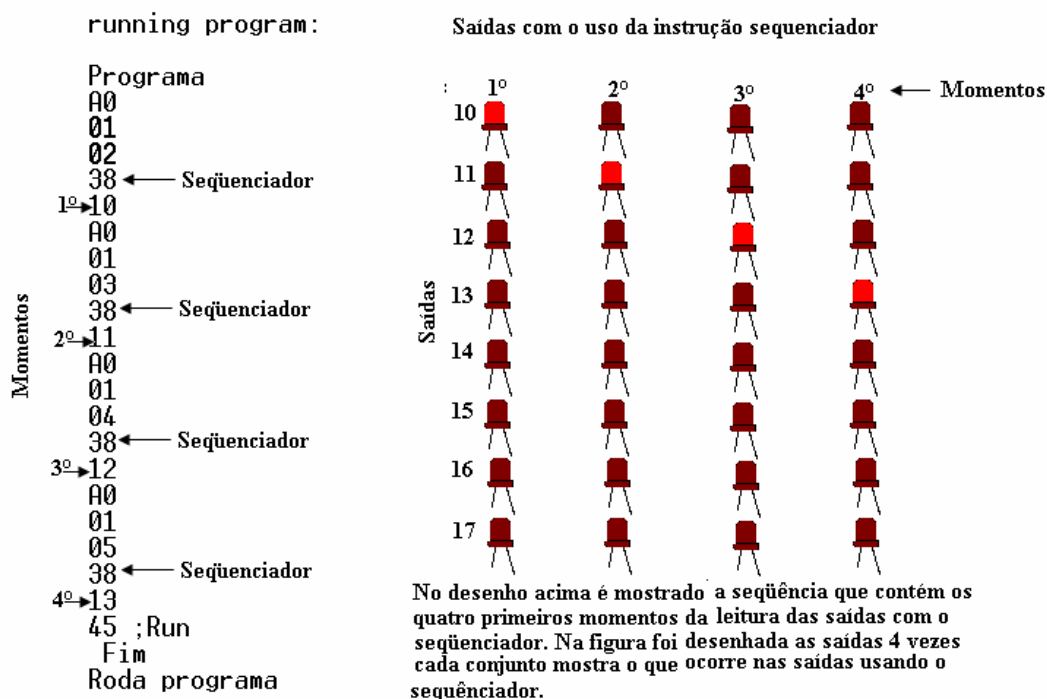


Figura 16: Programa com instrução seqüenciador para o CLP didático.

No exemplo a seguir repetimos o programa agora sem a instrução do seqüenciador nota-se na Figura 17 que os níveis nas saídas somente são atualizados no final do programa quando o programa interpretador encontra a instrução 45₁₆. As condições dos níveis de entrada indicadas no exemplo anterior continuam sendo válidas para este exemplo.

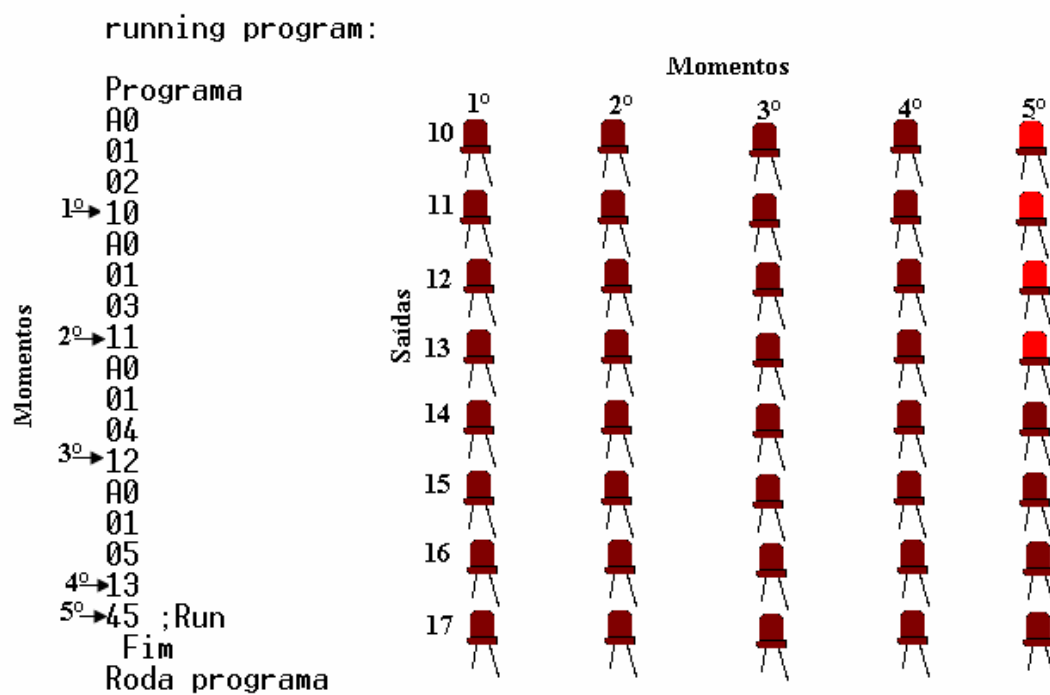


Figura 17: Programa sem instrução seqüenciador para o CLP didático.



CONCLUSÃO

O protótipo do Controlador Lógico Programável Didático após diversos testes mostrou-se fiel aos resultados esperados. Foram feitos inúmeros ensaios onde foi programado diversos circuitos lógicos combinacionais, programas que efetuavam também comparações. As funções aritméticas necessitam ser mais trabalhadas para assim poderem trabalhar com valores negativos e ponto flutuante. Melhorias devem ser agregadas ao protótipo para assim poder transformá-lo em um produto (aumento do número de funções, interface gráfica).

A proposta inicial que era de desenvolver um pequeno controlador lógico foi concluída com êxito.



REFERÊNCIAS BIBLIOGRÁFICAS

ALLEN-BRADLEY COMPANY, INC. **MicroMentor (entendendo e utilizando os microcontroladores programáveis)**. Catálogo 1996.

COCIAN, Luis Fernando Espinosa. **Manual da linguagem C**. Canoas: Ulbra, 2004.

NICOLOSI, Denys E.C. **Microcontrolador 8051 detalhado**. 3. ed. São Paulo: Érica, 2004.

NICOLOSI, Denys E.C.; BRONZERI, Rodrigo B. **Microcontrolador 8051 com Linguagem C**. São Paulo: Érica, 2005.

SÁ, Maurício Cardoso de. **Programação C para microcontroladores 8051**. São Paulo: Érica, 2005.

ZELENOVSKY, Ricardo; MENDONÇA, Alexandre. **Microcontroladores programação e projeto com a família 8051**. Rio de Janeiro: MZ, 2005.

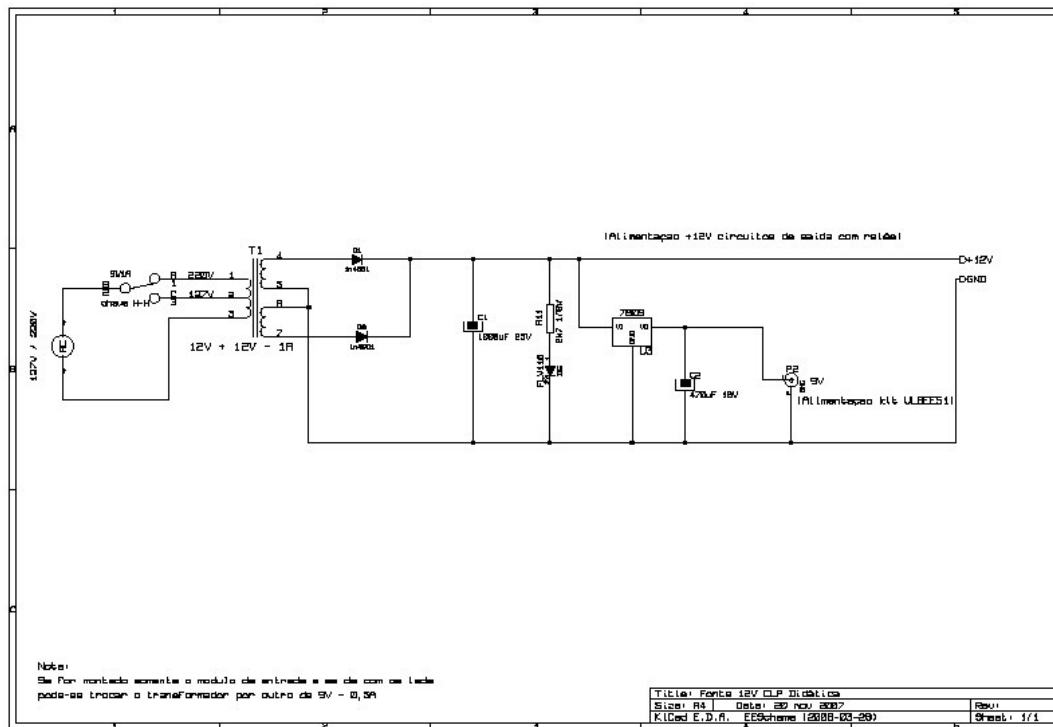
Sites da internet consultados:

www.engprod.ufjf.br/epd_automacao/EPD030_PLCS.pdf - Acesso em nov. 2007.

www.wikipedia.com.br – Acesso em nov. 2007.



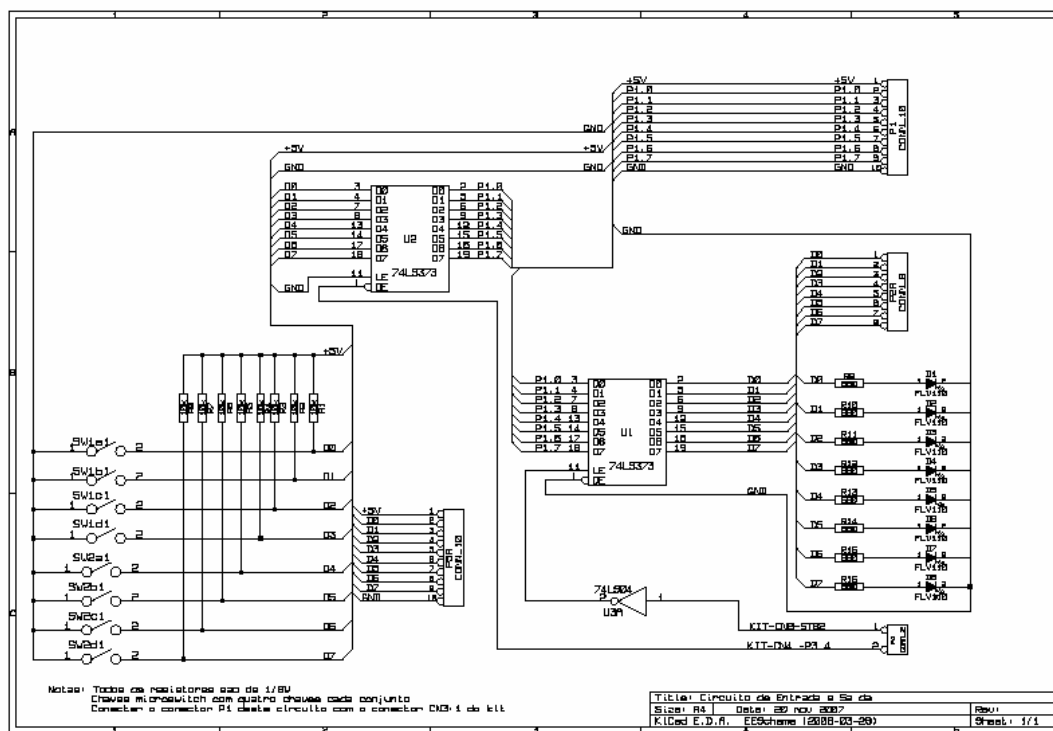
Esquema elétrico da Fonte de Alimentação





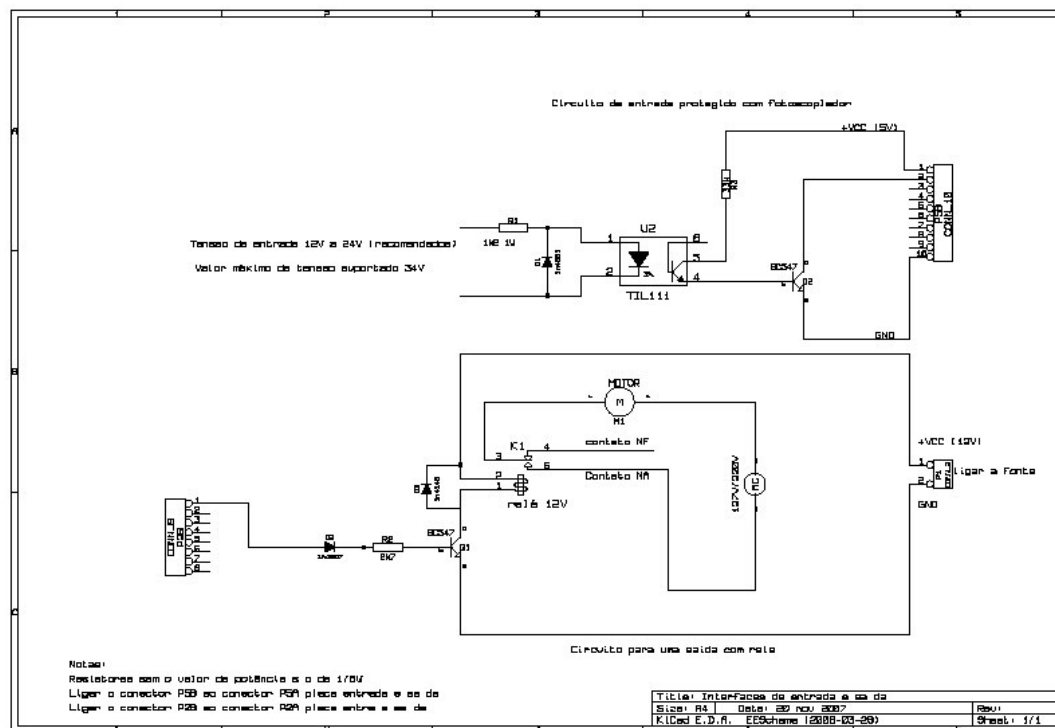
Apêndice B

Esquema elétrico de entrada e saídas padrão (com chaves e leds)



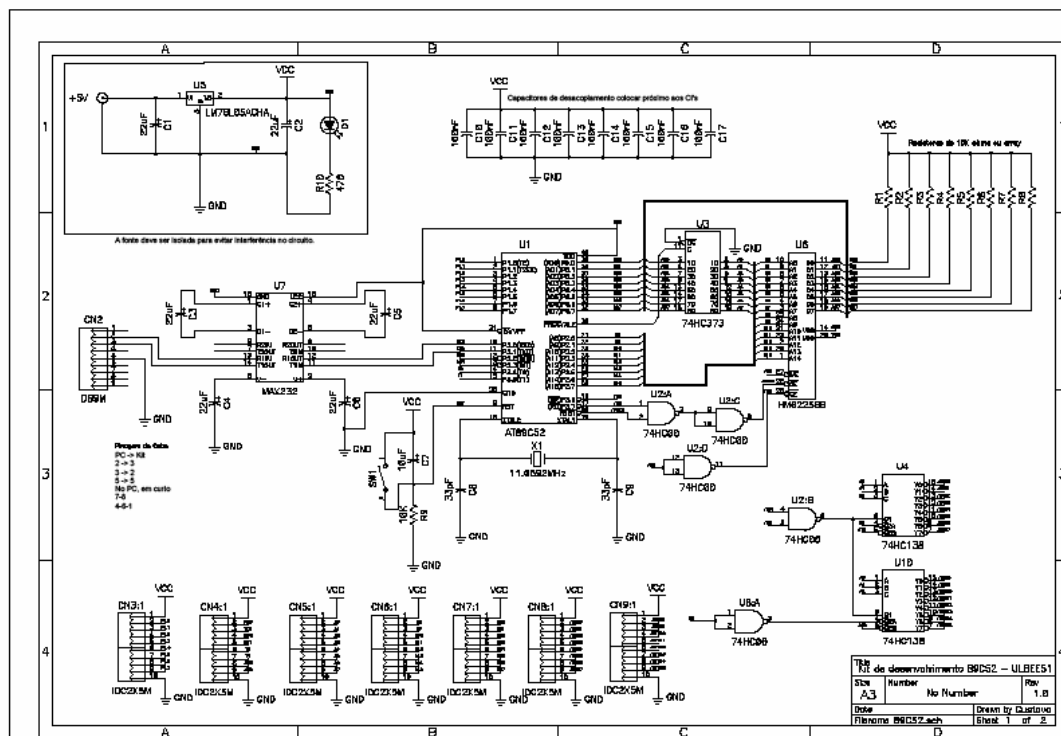
Apêndice C

Esquema elétrico com interfaces ópticas de entrada e de saída com relé



Apêndice D

Esquema elétrico do Kit Ulbee51





Apêndice E

Configuração do mide51 para trabalhar em conjunto com o Kit ULBEE51

1º Fazer o download do compilador SDCC. O download do compilador SDCC é conseguido gratuitamente pelo site: <http://sdcc.sourceforge.net/>. Colocar a pasta SDCC no diretório raiz C:\ abrimos a pasta SDCC e copiamos um atalho do mide para a área de trabalho ícone do mide é mostrado na Figura 1E. Basta clicarmos duas vezes sobre este ícone que se abrirá o editor (Figura 1E). O MIDE 51 pode ser baixado pelo site: <http://www.opcube.com/home.html>.



Figura 1E: Ícone do editor Mide51.

2º Após abirmos o mide 51 vamos configurar o compilador C do SDCC para trabalhar com o kit da Ulbra. Para tanto, devemos selecionar edit\preference Figura 2E.

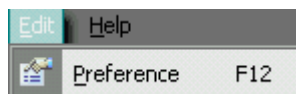


Figura 2E: Vista parcial opção edit, preference do editor Mide51.



3º Após selecionar C compilador e alterar o parâmetro tal como é mostrado na Figura 3E.

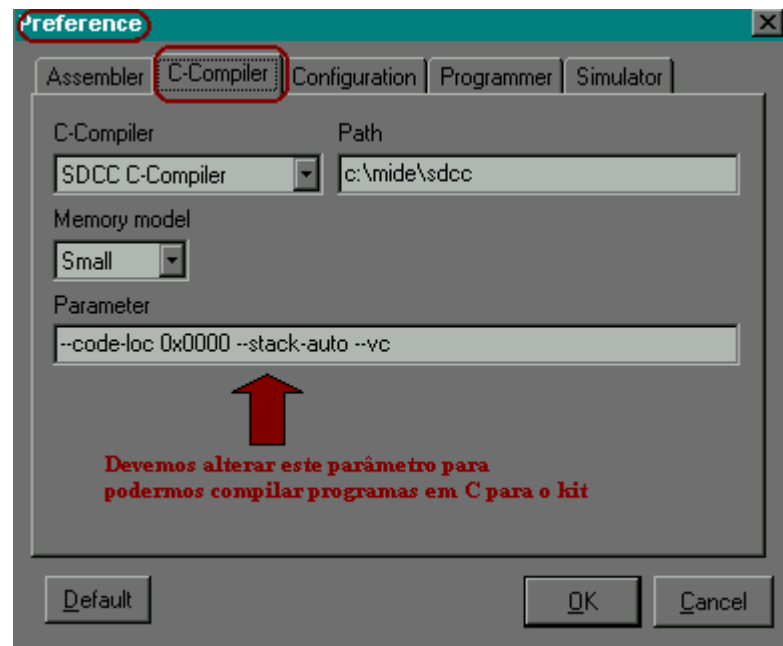


Figura 3E: Opção Preference (quadro com opções) selecionando opção.

4º Devemos colocar o seguinte parâmetro: (basta copiar e colar)

`--code-loc 0x8000 --xram-loc 0xe000--verbose c:\mide\sdcc\lib\small\paulmon2.lib`

Para funcionar devidamente o kit devem-se abrir as pastas no diretório C: as pastas mide/ sdcc/lib/small e, por fim, deve-se colocar o arquivo paulmon2.lib dentro desta última pasta. Este arquivo faz parte do programa monitor (Paulmon2) que deve ser baixado da internet.



5º Ficará da seguinte forma (Figura 4E):

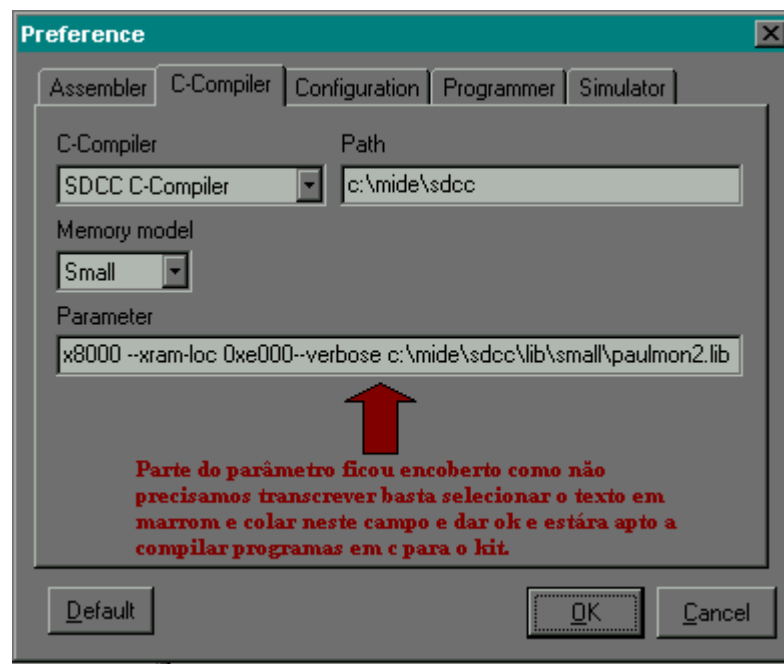


Figura 4E: Opção Preference, C compiler configurando o compilador C.



Configuração do Hiper terminal

Na Figura 5E é demonstrado como é feito para acessar o Hiper Terminal: basta seguir os passos indicados na Figura 5E.

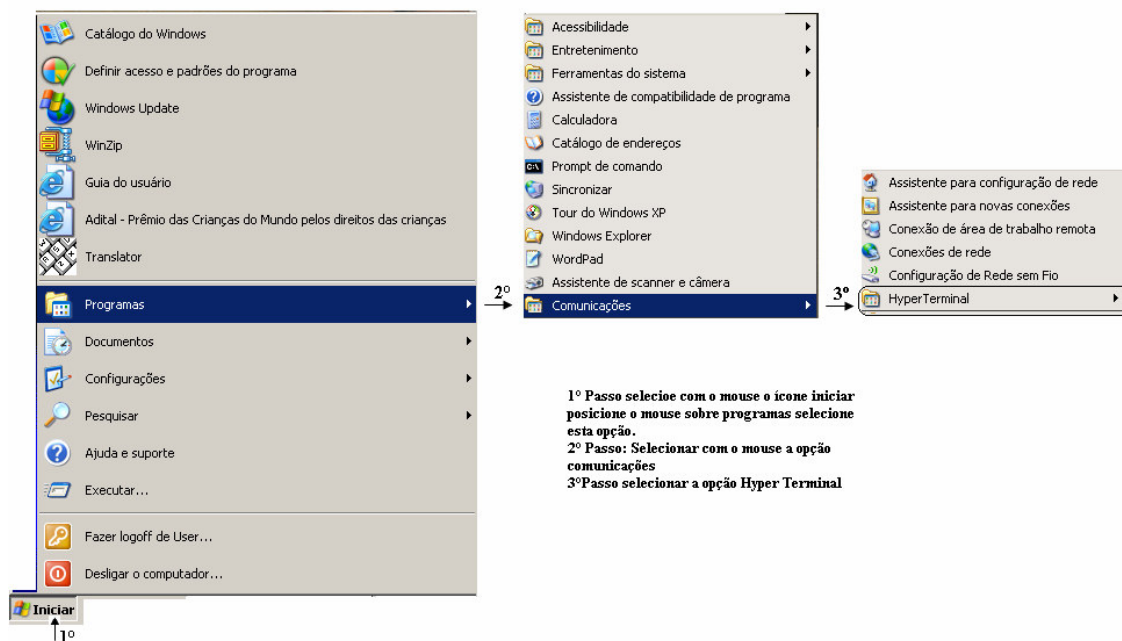


Figura 5E: Passos para acessar o Hiper Terminal (no Windows).

Na Figura 5E está indicado que devemos selecionar com o mouse o ícone iniciar e no quadro que se apresenta a seguir selecionamos programas e logo após selecionamos na seqüência Acessórios, comunicações e Hiper Terminal. Após selecionarmos com o mouse Hiper Terminal aparecerá o ícone mostrado na figura 6E. Deve-se habilitá-lo também. Feito este procedimento abre-se a tela do Hiper Terminal devemos configurar o Hiper Terminal.



Figura 6E: Opção Hiper Terminal.

O próximo passo é selecionar arquivo a opção nova conexão (Figura7E)

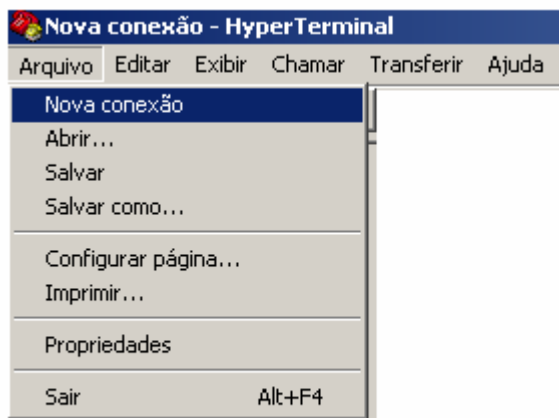


Figura 7E: Vista parcial régua de ferramentas hyperterminal.

Surge o quadro a seguir pedindo um nome para a nova conexão foi dado o nome de teste após selecionamos ok (Figura 8E)



Figura 8E: Nomeando uma nova conexão no hyperterminal.

No próximo quadro devemos selecionar a porta de comunicação serial que será usada. No caso foi a COM1 (Figura 9E).

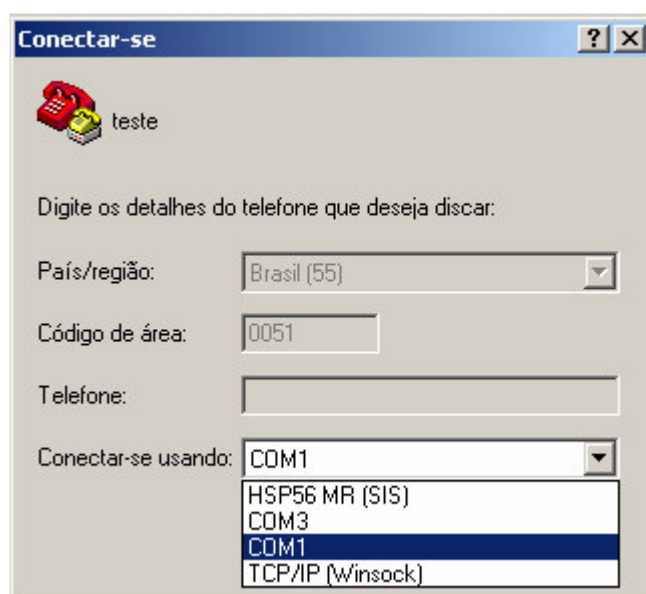


Figura 9E: Escolha da porta de comunicação serial do computador pessoal na qual será ligada o kit.



Após, deve-se configurar a porta COM1. Para tanto, basta copiar os valores indicados abaixo e alterá-los na sua conexão (Figura 10E).

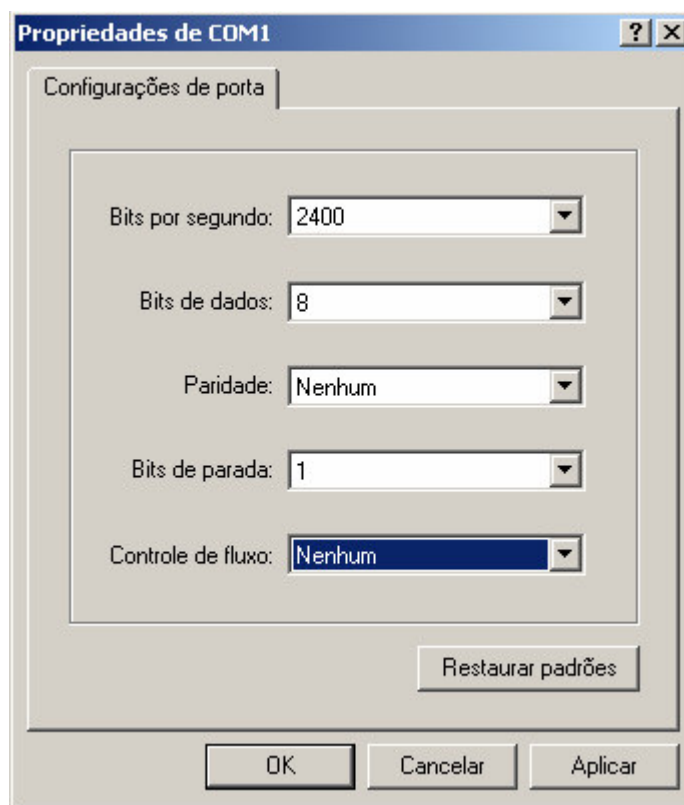


Figura 10E: Configurando a porta serial escolhida para fazer a comunicação.

Acesse propriedades e altere o item emulação (Figura12E). Abra arquivo e salve as configurações feitas.

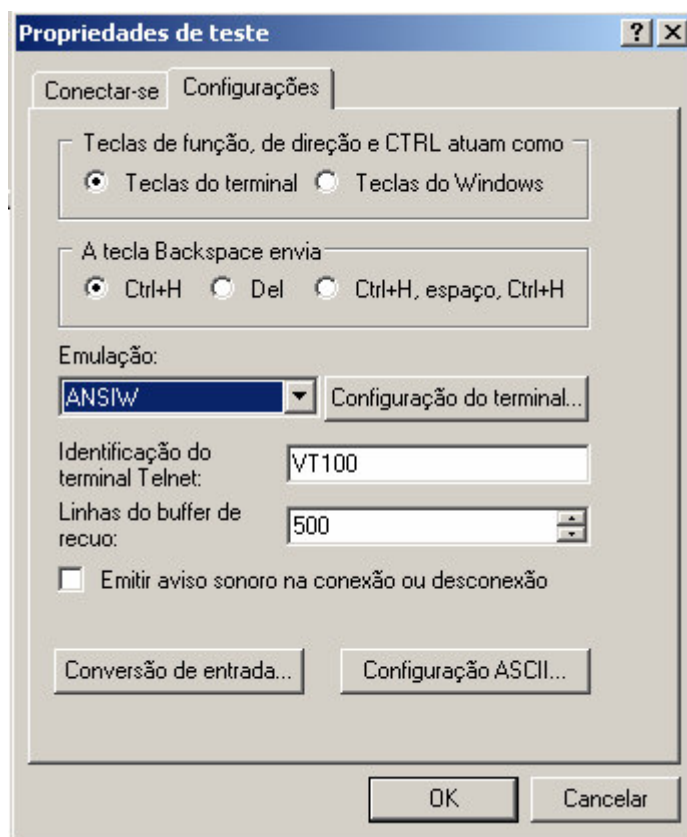


Figura 11E: Configurando as propriedades de teste.

Se tudo foi feito como indicado ligue o kit a porta COM1 via cabo apropriado para esta ligação e tecle a tecla Enter. Se tudo estiver funcionando a contento aparecerão as mensagens mostradas na Figura 12E.

Indicando que o Hiper Terminal está comunicando-se perfeitamente com o kit.



```

Welcome to PAULMON2 v2.1, by Paul Stoffregen

See PAULMON2.DOC, PAULMON2.EQU and PAULMON2.HDR for more information.

Program Name          Location      Type
List                  1000      External command
Single-Step           1400      External command
Memory Editor (VT100) 1800      External command

PAULMON2 Loc:8000 > _
```

Figura 12E: Tela Hiper Terminal expondo tela com inscrições iniciais do programa monitor Paulmon2.



Apêndice G

Fluxograma Programa Editor

[Acesse o arquivo do fluxograma em pdf \(drive d\)](#)



Apêndice G

Fluxograma Programa Interpretador

[Acesse o arquivo do fluxograma em pdf \(drive d\)](#)



Apêndice H

Software do Controlador lógico Programável Didático

```
/*
HABILITANDO BIBLIOTECAS
*/

#include <8052.h>
#include <paulmon2.h>

/*
DEFINIÇÕES
*/

#define ghex                0x03A
#define and_bit             0xa0
#define or_bit              0xb0
#define exor_bit            0xc0
#define not_bit             0xd0
#define and_byte            0xa1
#define or_byte             0xb1
#define exor_byte           0xc1
#define maior               0xe0
#define menor               0xe1
#define igualdade           0xe2
#define soma                0xe3
#define subtracao           0xe4
#define multiplicacao       0xe5
#define divisao             0xe6
#define resto_divisao       0xe7
#define fim                 0x45
#define prepara_leitura     0xff
#define habilita_latch_leitura 0
#define desabilita_latch_leitura 1
#define sequenciador        0x38
```



```

/*****
DECLARAÇÃO DE FUNÇÕES
*****/

unsigned char entrada (unsigned char caso);

/*****
DECLARAÇÃO DE VARIÁVEIS GLOBAIS
*****/

unsigned char leitura_entrada;
xdata unsigned char vetor[255];
unsigned int i;

void main(void)

/*****
DECLARAÇÃO DE VARIÁVEIS LOCAIS
*****/

{
unsigned int acumulador, operando_1, operando_2;

unsigned char flag_saida_1, flag_saida_2, flag_sequenciador, caso, espelho_saida,
acumulador1, acumulador2, resto;

xdata at 0x02 int escreve_valor_na_saida;

/*****
PROGRAMA EDITOR
VETOR QUE CARREGA O PROGRAMA A SER EXECUTADO PELO CLP DIDÁTICO
*****/
pm2_pstr ("Programa");
pm2_newline();

for (i=0; i<255; i++)
{ //for início
_asm;
lcall ghex;
_endasm;
caso = ACC;
vetor[i] = caso;

if (caso == fim)
{ //if início
pm2_pstr (" ;Run");
pm2_newline();
}
}
}

```



```
pm2_pstr (" Fim");
break;
} //if fim
P1 = 0;
pm2_newline();
} //for fim
```

```
/*
PROGRAMA INTERPRETADOR
"EXECUTA O PROGRAMA EM HEXADECIMAL CARREGADO ACIMA"
*/
```

```
pm2_newline();
pm2_pstr ("Roda programa");
pm2_newline();
pm2_newline();
flag_saida_1 = i = caso = acumulador = espelho_saida = flag_sequenciador = 0;
```

```
while (1)
```

```
{ //início while b
```

```
caso = vetor[i];
operando_1 = acumulador;
```

```
if (i==0) flag_saida_2 = 1;
if (flag_saida_1 == 1) flag_saida_2 = 1;
if (flag_saida_2 == 1)
{ //início if
acumulador1 = caso;
i++;
caso = vetor[i];
entrada (caso);
operando_1 = leitura_entrada;
flag_saida_2 = flag_saida_1 = 0;
```

```
if(caso == not_bit)
{
i++;
caso = vetor[i];
entrada (caso);
operando_1 = leitura_entrada;
operando_1 = ~ operando_1;
operando_1 = operando_1 & 1;
}
caso = acumulador1;
} //final if
```

-



```
switch (caso)
{ //casos início

case and_bit:
i++;
caso = vetor[i];
entrada (caso);
operando_2 = leitura_entrada;

if(caso == not_bit)
{
i++;
caso = vetor[i];
entrada (caso);
operando_2 = leitura_entrada;
operando_2 = ~ operando_2;
operando_2 = operando_2 & 1;
}

acumulador = operando_1 & operando_2;

break;

case or_bit://or

i++;
caso = vetor[i];
entrada (caso);
operando_2 = leitura_entrada;

if(caso == not_bit)
{
i++;
caso = vetor[i];
entrada (caso);
operando_2 = leitura_entrada;
operando_2 = ~ operando_2;
operando_2 = operando_2 & 1;
}

acumulador = operando_1 | operando_2;

break;

case exor_bit://xor

i++;
```



```
caso = vetor[i];
entrada (caso);
operando_2 = leitura_entrada;

if(caso == not_bit)
{ //if inicio
i++;
caso = vetor[i];
entrada (caso);
operando_2 = leitura_entrada;
operando_2 = ~operando_2;
operando_2 = operando_2 & 1;
} //if final

acumulador = operando_1 ^ operando_2;
break;

case and_byte : //and byte
i++;
operando_2 = vetor[i];
acumulador = operando_1 & operando_2;
break;

case or_byte : //or byte
i++;
operando_2 = vetor[i];
acumulador = operando_1 | operando_2;
break;

case exor_byte : //xor byte
i++;
operando_2 = vetor[i];
acumulador = operando_1 ^ operando_2;
break;

case maior: //maior

i++;
operando_2 = vetor[i];

if (operando_2 > operando_1)
{
i++;
caso = vetor[i];
acumulador = 1;
}
else acumulador = 0;
```



break;

case menor://menor

```
i++;
operando_2 = vetor[i];
if (operando_2 < operando_1)
{
i++;
caso = vetor[i];
acumulador = 1;
}
else acumulador = 0;
```

break;

case igualdade://igualdade

```
i++;
operando_2 = vetor[i];

if (operando_2 == operando_1)
{
i++;
caso = vetor[i];
acumulador = 1;
}

else acumulador = 0;
```

break;

case soma://soma

```
i++;
operando_2 = vetor[i];
acumulador = operando_2 + operando_1;
```

break;

case subtracao://subtração

```
i++;
operando_2 = vetor[i];
acumulador = operando_1 - operando_2;
```



```
break;

case multiplicacao://multiplicação

i++;
operando_2 = vetor[i];
acumulador = operando_2 * operando_1;

break;

case divisao://divisão

i++;
operando_2 = vetor[i];
acumulador = operando_1 / operando_2;
resto = operando_1 % operando_2;

break;

case resto_divisao://resto
acumulador = resto;
espelho_saida = P1 = 0;
break;

} //casos final

i++;
P1 = 0;
acumulador2 = vetor[i];

if (acumulador2 == sequenciador)
{
flag_sequenciador = 1;
i++;
}

switch (caso)//saída

{

case 0x18: P1 = acumulador; flag_saida_1 = 1;
break;

case 0x10: P1_0 = acumulador; flag_saida_1 = 1;
break;
```



```
case 0x11: P1_1 = acumulador; flag_saida_1 = 1;
break;

case 0x12: P1_2 = acumulador; flag_saida_1 = 1;
break;

case 0x13: P1_3 = acumulador; flag_saida_1 = 1;
break;

case 0x14: P1_4 = acumulador; flag_saida_1 = 1;
break;

case 0x15: P1_5 = acumulador; flag_saida_1 = 1;
break;

case 0x16: P1_6 = acumulador; flag_saida_1 = 1;
break;

case 0x17: P1_7 = acumulador; flag_saida_1 = 1;
break;

}

if (flag_sequenciador == 1)
{
if (flag_saida_1 == 1)
{
escreve_valor_na_saida; //dados enviados de forma sequencial
}
}

if (flag_sequenciador == 0)
{
if (flag_saida_1 == 1)
{
espelho_saida = espelho_saida | P1;
}}

//teste
if (vetor[i] == fim)

{
if(flag_sequenciador == 0)
{
P1 = espelho_saida;
escreve_valor_na_saida;
}
}
```




```
espelho_saida = i = operando_2 = 0; //sai fora do loop quando encontra o código
45h no caso recomeça a leitura
}
```

```
if (vetor[i] == 0) i = operando_2 = 0; //reinicia a contagem do vetor
```

```
}//fim while b
```

```
}//fim main
```

```
/*
*****
FUNÇÃO LEITURA DAS ENTRADAS
*****
*/
```

```
unsigned char entrada (unsigned char caso)//testado ok
```

```
{
```

```
P1 = prepara_leitura;
```

```
P3_4 = habilita_latch_leitura ;
```

```
switch (caso)
```

```
{
```

```
case 0x01: leitura_entrada = P1_0;
```

```
break;
```

```
case 0x02: leitura_entrada = P1_1;
```

```
break;
```

```
case 0x03: leitura_entrada = P1_2;
```

```
break;
```

```
case 0x04: leitura_entrada = P1_3;
```

```
break;
```

```
case 0x05: leitura_entrada = P1_4;
```

```
break;
```

```
case 0x06: leitura_entrada = P1_5;
```

```
break;
```

```
case 0x07: leitura_entrada = P1_6;
```

```
break;
```

```
case 0x08: leitura_entrada = P1_7;
```

```
break;
```

```
case 0x09: leitura_entrada = P1;
```

```
break;
```

```
}
```



```
P3_4 = desabilita_latch_leitura;
```

```
return (caso);  
}
```



GLOSSÁRIO

A

Analogia – Uma analogia é uma relação de equivalência entre duas outras relações.

Alta impedância – Alta impedância é a característica que apresenta um material que dificulta enormemente a passagem de um sinal elétrico qualquer por este elemento. No texto equivale a dizer que o pino de um circuito integrado que está em alta impedância não afetará de forma alguma valores de tensão e correntes presentes neste circuito são como se o pino estivesse desligado do circuito elétrico.

Alimentação – A palavra alimentação no texto refere-se ao fornecimento de valores de tensão adequados para o bom funcionamento dos aparelhos eletrônicos

B

Bit – Simplificação para dígito binário, “*Binary digiT*” em inglês) é a menor unidade de medida de transmissão de dados usada na Computação e na Teoria da Informação, embora muitas pesquisas estejam sendo feitas em computação quântica com qubits. Um bit tem um único valor, 0 ou 1, ou verdadeiro ou falso, ou neste contexto quaisquer dois valores mutuamente exclusivos.

Byte – É o conjunto formado por oito Bits

Bifilar – No texto está relacionado representações de plantas de circuitos elétricos que apresentam os circuitos da forma como são montados.



C

Conversor analógico digital – Circuito eletrônico que tem a finalidade de converter um sinal analógico presente em sua entrada em um sinal digital que pode ser processado por um microcontrolador.

Conversor digital analógico – Circuito eletrônico que tem a finalidade de converter um sinal digital presente na sua entrada em um sinal analógico que estará presente em sua saída.

Corrente elétrica – É o movimento ordenado de elétrons em um meio condutor que ocorre quando este meio condutor está sobre a influência de um potencial elétrico.

Corrente contínua – É o nome dado a corrente gerada por um dispositivo químico (pilhas, baterias) ou gerador eletromecânico que provoque a circulação da corrente em um único sentido. A outra característica de uma corrente contínua diz respeito ao valor de tensão fornecido pelo gerador que é constante no tempo. O tipo de gerador é que define se a corrente gerada será contínua pulsante ou alternada.

Corrente alternada – A corrente alternada é caracterizada por alternar ciclicamente o sentido em que se movimentam os elétrons livres e por apresentar variações periódicas nos valores de tensão e intensidade da corrente. A corrente alternada é gerada por máquinas eletromecânicas chamadas de alternadores.

Circuitos condicionadores – Condicionador são circuitos eletrônicos desenvolvidos com a finalidade de modificar valores de tensão ou corrente (normalmente ampliar valores) de sinais elétricos provenientes de sensores e transdutores que devam ser ligados a entradas de um AD,

Chave interruptora – Dispositivo seccionador da corrente em um circuito elétrico. Este dispositivo tem a função de ligar e desligar um aparelho elétrico qualquer.



Capacitor – Componente eletrônico composto basicamente por duas placas de material condutor colocadas uma na frente da outra, separadas por um material isolante denominado dielétrico (o tipo de dielétrico utilizado normalmente dá o nome ao capacitor). A sua função é a de acumular cargas elétricas.

Circuito integrado – Sistema eletrônico integrado a uma pastilha de silício encapsulada em plástico ou outro material (por vezes metal), dotada de terminais que dão acesso externo ao circuito encapsulado.

Circuito elétrico – O circuito elétrico pode ser definido como o percurso completo por onde os elétrons ou os portadores de carga podem entrar de um terminal de uma fonte de tensão, passando através de condutores e componentes, até chegar ao terminal oposto da mesma fonte.

Conector – Diz-se de qualquer dispositivo que facilite a ligação entre dispositivos elétricos distintos de forma provisória.

D

Diodo Retificador - Componente eletrônico que é constituído de material semicondutor que tem a função de permitir a passagem da corrente em um único sentido. É usado no processo de retificação de uma corrente alternada.

Diodo de sinal – É um diodo retificador, porém consegue interromper a corrente circulante por ele de forma bem mais rápida do que um diodo retificador comum. Ele é usado em aplicações onde se opere com correntes de baixa intensidade e tensões pequenas, mas que necessitem de diodos que interrompam a corrente circulante muito rapidamente. É conveniente salientar que ele também entra em condução de maneira muito mais rápida do que o diodo retificador comum.

Diretamente polarizado – Diretamente polarizado é o termo técnico utilizado no texto para definir quando um diodo está conduzindo.



Datasheet – Catálogo fornecido pelos fabricantes de componentes eletrônicos que trazem as principais características elétricas do componente.

E

EPROM – EPROM (Erasable Programmable Read Only Memory). Memória somente de leitura, apagável e programável. Utilizam-se raios ultravioleta para apagar seu conteúdo, podendo ser reprogramada sempre que necessário. Não perde seu conteúdo quando desenergizada.

E2PROM-E2PROM (Electrically Erasable Programmable Read Only Memory). Memória não volátil apagável eletricamente

F

Flash EPROM – Memória não volátil apagável eletricamente.

Fonte de alimentação – Circuito eletrônico responsável pela retificação da corrente e dimensionamento de valores de tensão. Adequando estes valores as necessidades de cada etapa de um circuito eletrônico.

Fotoacopladores – Componente eletrônico que tem a finalidade de isolar etapas de um circuito contra descargas elétricas acidentais. A comunicação das etapas isoladas se dá por um feixe de luz. A variação da intensidade luminosa produzida por um dispositivo eletrônico chamado led produz variações na corrente que circula em um outro dispositivo onde incide a luz denominado fototransistor que converte a variação de luminosidade em variação de corrente. Estes dispositivos normalmente oferecem isolamento para tensões que possam chegar a alguns kilo Volt.

H

Hardware – Equipamentos físicos usados em processamento de dados, onde normalmente são executados programas (software).



I

Inversamente polarizado – No texto está sendo usado para indicar que um diodo está com sua junção semicondutora de tal forma polarizada que o diodo não permitirá a passagem de uma corrente elétrica por ele.

Impedância – É a relação entre o valor eficaz da diferença de potencial entre dois pontos de circuito em consideração, e o valor eficaz da corrente resultante no circuito. É expresso por um fasor, que da forma de número complexo possui uma parte real, equivalente a resistência R , e uma parte imaginária, dada pela reatância X . A impedância também é expressa em ohms, e designada pelo símbolo Z . Indica a oposição total que um circuito oferece ao fluxo de uma corrente elétrica variável no tempo.

Intensidade da corrente elétrica – A intensidade da corrente elétrica é definida pela fórmula, $I=Q/t$ sendo que " Q " é a carga que passa pelo fio e " t " é o tempo em que ela passa. O amperê é sua unidade de medida.

L

Ligação Série – Temos uma ligação série quando os elementos estão a saída de corrente do primeiro elemento está ligado a entrada de corrente do próximo elemento e assim sucessivamente

Ligação Paralelo – A ligação entre elementos é dita paralela quando os terminais de entrada de corrente de todos os elementos estão ligados a um mesmo fio e todos os terminais de saída de corrente dos elementos associados estão ligados a outro fio.

Ligação Mista – Temos uma ligação mista quando temos no mesmo circuito elementos ligados em série e em paralelo.

Led (Light Emitting Diode) – Tipo de diodo semicondutor que emite luz quando estimulado por eletricidade. Utilizado como indicador luminoso.



Linguagem – Um conjunto de regras, de convenções e de sintaxe. Um conjunto de símbolos utilizados para representação e comunicação de informações ou dados entre pessoas e máquinas

M

Microprocessador – É o elemento principal de um computador é um circuito integrado onde internamente encontramos a CPU e a unidade lógica aritmética é onde se processa todas as informações e é a onde é tomada as decisões. Funciona em conjunto com o software.

Microcontrolador – Também é um microprocessador só que inclui internamente outros dispositivos que o microprocessador não possui. O microcontrolador assemelha-se a um pequeno computador.

Módulo (quando se referir a hardware) – Elemento básico na configuração de um sistema que possui funções e sinais de entrada e saída bem definidos. Cada módulo é fabricado, testado e comercializado separadamente.

Mnemônicos – Um mnemônico é um nome reservado de uma família de códigos operacionais que realizam tarefas semelhantes no processador. Os códigos operacionais atuais diferem quanto ao tamanho e tipo de operandos que sejam utilizados.

N

Nível lógico – Nível lógico refere-se aos valores que podem ser assumidos por um bit de informação os níveis assumidos por um bit são 0(0V) e 1 (normalmente 5V).

O

Optoacopladores – Sinônimo de fotoacoplador.



P

Programa aplicativo – Programa projetado para uma aplicação em área de atividade específica.

Protocolo – Regras de procedimentos e formatos convencionais que, mediante sinais de controle, permitem o estabelecimento de uma transmissão de dados e a recuperação de erros entre equipamentos.

Pulso – Em eletrônica e processamento de sinais pulso é um sinal de duração finita, que assume um nível lógico constante por certo intervalo de tempo, assumindo outro nível lógico fora deste intervalo.

Potência dissipada – É parte da energia elétrica fornecida a um resistor que está sendo perdida na forma de calor.

Plug and Play – A tecnologia “Plug and Play (PnP)”, que significa “ligar e usar”, foi criada com o objetivo de fazer com que o computador reconheça e configure automaticamente qualquer dispositivo que seja instalado, facilitando a expansão segura dos computadores e eliminando a configuração manual.

R

Resistor – Resistor é um componente eletrônico que tem a função de limitar a corrente e dissipar parte da energia fornecida em calor

Resistência elétrica – É a oposição natural que todos os materiais apresentam contra a passagem de uma corrente elétrica a sua unidade de medida é o OHM o seu símbolo é a letra grega ômega Ω .

Ram (Random Access Memory) – Memória onde todos os endereços podem ser acessados diretamente de forma aleatória e a mesma velocidade. É volátil, ou seja, seu conteúdo é perdido quando desenergizada. Região de memória onde é feito o armazenamento de dados para o processamento do usuário



Relé – O relé pode ser comparado a uma chave interruptora (dispositivo que liga e desliga por exemplo a lâmpada de um cômodo de uma casa conhecido também pelo nome de chave) a diferença está em como é feito o acionamento. Em uma chave interruptora a pessoa interessada em ligar a lâmpada deve acionar o botão ou tecla da chave interruptora. No relé este acionamento é feito pela circulação de uma corrente elétrica em um rolo de fio (bobina). Quando a corrente passa pela bobina magnetiza um núcleo de material ferromagnético fechando os contatos do relé acionando o dispositivo que será controlado.

Retificação – Retificação é processo que transforma uma corrente alternada em uma corrente contínua pulsante este processo é efetuado por um componente eletrônico chamado diodo retificador.

S

Sintetizador de voz – Voz gerada artificialmente por meio de softwares em dispositivos digitais (computadores, etc.)

Software – Programas de computador, procedimentos e regras relacionadas à operação de um sistema de processamento de dados.

Sinal elétrico – Sinal elétrico é definido como sendo variações elétricas de pequena intensidade, mas que carregam consigo alguma informação que será processada por circuitos eletrônicos.

Sensor – Dispositivo que transforma algum fenômeno de origem não elétrica em um sinal elétrico que possa ser processado por circuitos eletrônicos (exemplo de sensor: sensores de temperatura transformam variações de temperatura em variações de tensão).

T

Tensão elétrica – A tensão elétrica é a grandeza elétrica que provoca a circulação de uma corrente elétrica por um dispositivo qualquer sem tensão não temos a corrente elétrica. Os aparelhos elétricos só funcionam quando por eles



passam uma corrente elétrica provocada por uma tensão elétrica de valor adequada a sua unidade é o volt e o seu símbolo é a letra V.

Transistor – O transistor é um componente eletrônico de material semicondutor que pode exercer as seguintes funções em um circuito eletrônico: amplificação ou chaveamento.

Tiristor – Tiristores são dispositivos utilizados quando queremos ligar e desligar dispositivos que operam com tensões e correntes elevadas e necessitamos de dispositivos mais rápidos do que relés. Tiristor na verdade é o nome comum dado a vários dispositivos semicondutores que efetuam chaveamento. Os tiristores mais comuns são: SCR e o TRIAC.

Tabela verdade – Tabela verdade é uma tabela que nos indica o comportamento de uma porta lógica ou o comportamento de um circuito lógico relacionando os níveis presentes nas suas entradas com as suas saídas.

Transformador – É uma máquina elétrica que pode alterar valores de tensão corrente e impedância em circuitos elétricos.

U

USB (Universal Serial Bus) – É um tipo de conexão *Plug and Play* que permite a conexão de periféricos sem a necessidade de desligar o computador.