UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

INTERNET EMBARCADA E COMUNICAÇÃO IRDA UTILIZANDO O MICROCONTROLADOR 8051

CARLOS EDUARDO COUTINHO NOGUEIRA
TIAGO ALVES DA FONSECA

ORIENTADOR: RICARDO ZELENOVSKY

MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: ENE-1/2005

BRASÍLIA / DF, 05 JULHO/2005

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

INTERNET EMBARCADA E COMUNICAÇÃO IRDA UTILIZANDO O MICROCONTROLADOR 8051

CARLOS EDUARDO COUTINHO NOGUEIRA
TIAGO ALVES DA FONSECA

PROJETO FINAL DE GRADUAÇÃO SUBMETIDO AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRICISTA.

APROVADA POR:

RICARDO ZELENOVSKY, Doutor, PUC-RJ
(ORIENTADOR)

GEOVANY ARAÚJO BORGES, Douteur, Université Montpellier II, FR
(EXAMINADOR INTERNO)

HUMBERTO SEROZA, Engenheiro , Instituto Militar de Engenharia.
(EXAMINADOR EXTERNO)

DATA: BRASÍLIA/DF, 5 JULHO DE 2005

FICHA CATALOGRÁFICA

NOGUEIRA, CARLOS EDUARDO COUTINHO &
FONSECA, TIAGO ALVES DA
Internet Embarcada e Comunicação IrDA utilizando o microcontrolador 8051 [Distrito Federal] 2005.
xv, 145p., 297 mm (ENE/FT/UnB, Bacharel, Engenharia Elétrica, 2005). Monografia de Graduação –
Universidade de Brasília, Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. Comunicação IrDA 2. Internet Embarcada
I. ENE/FT/UnB. II. Título

REFERÊNCIA BIBLIOGRÁFICA

NOGUEIRA, CARLOS EDUARDO COUTINHO e FONSECA, TIAGO ALVES DA (2005). Internet Embarcada e Comunicação IrDA com o Microcontrolador 8051. (Projeto Final de Graduação), Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF,

CESSÃO DE DIREITOS

NOME DO AUTOR: Carlos Eduardo Coutinho Nogueira & Tiago Alves da Fonseca
TÍTULO DA DISSERTAÇÃO: Internet Embarcada e Comunicação IrDA com o Microcontrolador 8051.
GRAU: Engenheiro Eletricista ANO: 2005

Carlos Eduardo Coutinho Nogueira
SMPW Q. 4 Ch. 23 Lote 02
CEP 71734-040 – Águas Claras – DF - Brasil

Tiago Alves da Fonseca
SQN 114, Bloco G, AP 610
CEP 70764-070 – Brasília – DF - Brasil

AGRADECIMENTOS

Os autores agradecem ao orientador, Dr. Ricardo Zelenovsky, por todo conhecimento transmitido e pela prontidão no atendimento das necessidades do projeto. Agradecem também a toda Equipe do GPDS, pela atenção e apoio. Aos Técnicos e funcionários do SG-11 pela presteza e ajuda.

DEDICATÓRIA

Carlos Eduardo dedica esse trabalho a Deus, que sempre esteve presente em sua vida, à minha Tia Lycia de Lorena Coutinho que me ajudou durante todo o meu período acadêmico, aos pais, familiares e amigos sempre presentes em sua vida.

Tiago Alves dedica esse trabalho a Deus, que constantemente mostra a fragilidade do homem, aos pais, que sempre lutaram para ofererecer-lhe uma boa educação, à namorada Thais Poliana, que o apoiou nos momentos difíceis, aos amigos, sempre presentes nos momentos de tristeza e felicidade.

RESUMO

O presente trabalho trata do estudo e implementação de padrões de comunicação amplamente difundidos em um ambiente de desenvolvimento para microcontroladores da família 8051. São abordados os protocolos IrDA e TCP/IP e propostas soluções para sistemas embarcados. Disponibilizar uma interface entre o microcontrolador 8051 e a *internet* motivou este projeto. A interconexão entre a *internet* e o microcontrolador foi bem sucedida e resultou na implementação de um servidor de páginas na *web* microcontrolado enquanto a implementação de uma comunicação sem fios foi realizada usando o método serial infravermelho.

ABSTRACT

This work presents a study of two well known communications standards (TCP/IP and IrDA protocols) and suggests their implementations in a 8051 microcontrolled environment. The motivation of this work is provide an interface between the internet and an 8051 microcontroller. The project succeeded in realising an 8051 microcontrolled web server and wireless communication by SIR – Serial Infrared – proposed by IrDA.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIAÇÕES

Abreviação	Significado
IrDA	*The Infrared Data Association*
IrCOMM	Especificação IrDA para a emulação de comunicação de portas serial e paralela , *IrDA specification for the emulation of serial and parallel port communications*
IrLAN	Especificação IrDA para acesso LAN por médio infravermelho, *IrDA specification for accessing a LAN over an infrared médium.*
IrLAP	Especificação IrDA para o protocolo do acesso da ligação, *IrDA specification for Link Access Protocol. This document specifies an HDLC-based protocol for controlling access to the infrared medium.*
IrLMP	*IrDA specification for Link Management Protocol. This protocol provides the LM-MUX and LM-IAS services.*
IrOBEX	*IrDA specification that defines the protocol for generic object exchange in an IrDA – enabled device.*
IrPHY	A especificação que descreve as propriedades da camada física do padrão IrDA, *The specification that describe the physical layer properties of the IrDA standard.*
LM-IAS	*The link Management Information Access Service allows a pair of IrDA devices to interrogate each other to determine the sevices available on each device.*
LM-MUX	*The link management multiplexr allows any pair of IrDA devices to simultaneously and independently use a single IrDA connection between themselves.*
LSAP	*Link Service Access Ports are address fields that uniquely identify applications on the source and destination devices.*
LSAP-SEL	*Link Service Access Port Selector*
LSAP	*Link Service Access Ports are address fields that uniquely identify applications on the source and destination devices.*
Cell	Um símbolo em PPM
Chip	*A pulse within a symbol (cell) in PPM.*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
HTTPD	*Hypertext Transfer Protocol daemon*
IP	*Internet Protocol*
IPv4	*Internet Protocol - version 4*
IPv6	Protocolo de Internet – versão 6, *Internet Protocol - version 6.*
ISP	Provedor de Serviço de Internet, *Internet Service Provider.*
PC	Computador Pessoal, *Personal Computer.*
QoS	Qualidade de Serviço, *Quality of Service.*
TCP	Protocolo de Controle de Transmissão, *Transmission Control Protocol.*
UDP	*User Datagram Protocol*
VBR	Taxa de bit variável, *Variable Bit Rate.*
ENDEC	**Codificador Decodificador usado na camada física da IrDA, The enconder-decoder used in the IrDA physical layer.**
WWW	Grande Teia Mundial, *World Wide Web.*
Tiny TP	*Lightweight transport protocol specification*

SIR	Infravermelho Serial, *Serial Infrared.*
PPM	Modulação por Posição de Pulso, *Pulse Position Modulation.*
API	**Application Program Interface**
PDA	**Personal Digital Assistant**
ASYNC	Enquadramento Assíncrono
TTP	**Tiny TP**
GAR	**Registrador de Endereço de Gateway**
SMR	**Registrador de Sub-máscara de Rede**
SHAR	**Registrador de Endereço de Hardware de Origem**
SIPR	Registrador de Endereço de IP de Origem
DVR	*Digital Video Recorder*
RF	Rádio Freqüência
BER	Taxa de erro de bit, *Bit Error Rate.*
CI	Circuito Integrado
OWD	*Oscillator Watchdog*
CC	*Compare/Capture registers*
UART	Transmissor Receptor Universal Assíncrono, *Universal Asynchronous Receiver Trasmitter*.
LED	Diodo Emissor de Luz
PCA	*Programmable Counter Array*
ALU	*Aritmetic Logic Unit*
PWM	*Pulse width modulation*
A/D	*Analógic / Digital Converter*
SPI	Comunicação serial síncrona, *Serial Peripheral Interface Bus*
BRG	*Baud Rate Generator*
CAN	*Controller Área Network*
HVAC	*High-voltage alternating current*
IR	Infravermelho
SDU	Unidade de Dados de Serviço, *Service Data Unit*
OSI	*Open Systems Interconnection*
HDLC	*High-Level Data Link Control*
SDLC	*Synchronous Data Link Control*
LSI	Integração em Larga Escala, *Large Scale Integration*

1. INTRODUÇÃO

A evolução da eletrônica nas últimas décadas permitiu ao mundo um salto tecnológico sem precedentes na história, facilitando a vida do homem por meio da automação de processos repetitivos, a redução de tamanho dos componentes e integração em larga escala dos sistemas eletrônicos.

A difusão da eletrônica acabou por baratear os preços dos componentes o que justifica encontrarmos sistemas inteligentes nos mais diversos tipos de equipamentos: de máquinas de lavar a telefones celulares de última geração. A esse fenômeno dá-se o nome de computação invasiva e um dos protagonistas desse fenômeno é o microcontrolador.

Um microcontrolador é uma espécie de processador com baixa capacidade de processamento (se comparado com os processadores usados na arquitetura PC), de instruções compactas e velozes, capaz de realizar com simplicidade manipulações binárias, que realiza operações matemáticas somente com números inteiros, composto por interfaces que permitem fácil integração a outros dispositivos, de fácil programação, enfim, um ambiente eletrônico planejado para facilitar atividades de controle digital.

As telecomunicações foram beneficiadas pela mesma eletrônica que automatiza processos e miniaturiza sistemas: os simples pares trançados que carregavam os sinais de voz foram ultrapassados pelos sistemas de banda larga via satélites e pela fibra óptica. O crescimento das comunicações e sua difusão começaram a justificar a disponibilização em ambientes microcontrolados de protocolos de comunicação, facilitando a vida dos desenvolvedores no embarque de novas capacidades em seus sistemas eletrônicos.

A necessidade de comunicar os sistemas microcontrolados usando novos paradigmas levou ao estudo dos protocolos IrDA, que trata de comunicações em infravermelho, e do protocolo TCP/IP, que trata da interligação dos sistemas com Internet.

Será realizado um estudo do protocolo IrDA, apresentando as suas características e implementando um modo de comunicações simples, o serial sobre infravermelho (SIR). Será proposto um circuito de interfaceamento entre o microcontrolador da família 8051 e a plataforma IrDA através de um circuito integrado que implementa a pilha de protocolos IrDA, o MCP2150.

Um outro circuito integrado, o W3100A, apresenta a pilha de protocolos TCP/IP implementada em *hardware*, facilitando interfaceamento de microcontroladores com a

Internet. Será realizado um estudo do funcionamento desse integrado, de seus métodos de interfaceamento e, a partir dos recursos disponibilizados por esse integrado em um módulo chamado NM7010A, realizar a implementação de um servidor de páginas da Internet microcontrolado por um integrado da família 8051.

2. IRDA

À medida que comunicações em infravermelho, baseadas nos padrões da IrDA (*Infrared Data Association*), tornam-se largamente disponibilizadas em computadores pessoais e periféricos, surge uma oportunidade de realizarmos comunicações sem fio de curtas distâncias de maneira efetiva e barata para sistemas embarcados e dispositivos de todos os tipos. Os padrões IrDA desenvolveram-se rapidamente (comparados com a maioria organizações responsáveis por lançamento de padrões) e as informações a respeito dos protocolos IrDA ainda não atingiram cada esquina do universo de sistemas embarcados.

A IrDA é um grupo "industrial" baseado na união de 150 empresas que vêm desenvolvendo padrões de comunicação voltados especialmente para comunicações baratas, de curtas distâncias, entre diferentes plataformas e entre dois pontos (*point-to-point*) e em diversas taxas de transmissão. Esses padrões têm sido implementados em várias plataformas de computadores e mais recentemente têm-se tornado disponíveis em muitas aplicações em sistemas embarcados. Devido à sua vasta aceitação, as especificações IrDA estão agora numa trajetória acelerada para sua adoção como padrão ISO.

2.1. PILHA DE PROTOCOLOS IRDA

Protocolos de comunicação tratam de muitas questões e são geralmente divididos em camadas que abrangem de um conjunto gerenciável de responsabilidades e fornecem os recursos solicitados pelas camadas superiores e inferiores. Quando empilhamos camadas, temos o que é convencionalmente chamado de pilha de protocolo como uma pilha de pratos. Uma pilha de protocolos IrDA é um conjunto de protocolos particularmente destinado a comunicações ponto a ponto em infravermelho e às aplicações necessárias naquele ambiente. A Figura 2.1 abaixo apresenta as camadas de protocolo IrDA [1], [8] e [9]:

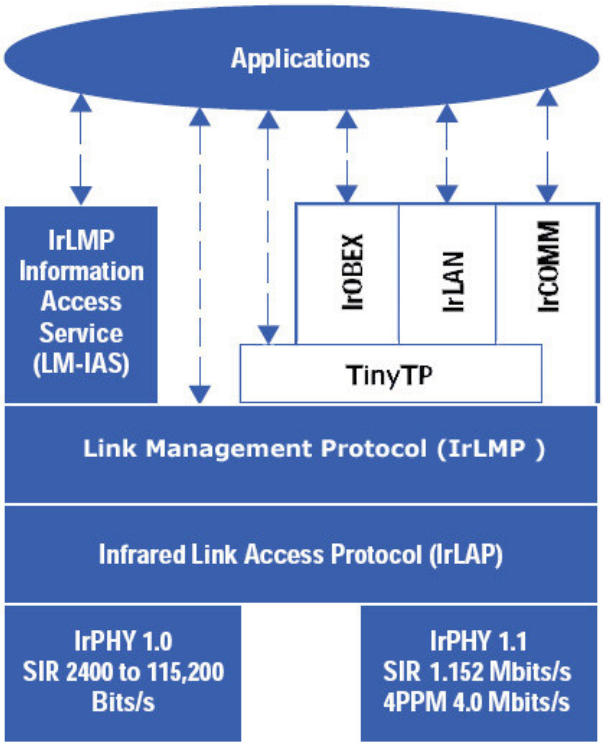


Figura 2.1 Pilha de Protocolos IrDA

As camadas dentro dessa pilha podem ser divididas em dois grupos: necessárias e opcionais.

2.1.1. Protocolos IrDA necessários

As camadas necessárias na pilha de protocolos IrDA estão nas caixas em azul do diagrama da Figura 2.1 e incluem:

- ✓ Camada Física: especifica características ópticas, codificação dos dados e enquadramento para várias velocidades;
- ✓ IrLAP: *Link Access Protocol*. Estabelece a conexão confiável básica.
- ✓ IrLMP: *Link Management Protocol*. Multiplexa serviços e aplicações na conexão LAP.
- ✓ IAS: *Information Access Service*. Disponibiliza as "páginas amarelas" dos serviços em um dispositivo.

2.1.2. Protocolos IrDA Opcionais

Os protocolos opcionais são mostrados nas caixas em branco da Figura 2.1. O uso de camadas opcionais depende da aplicação do cliente. Os protocolos opcionais são:

- ✓ TinyTP: *Tiny Transport Protocol*. Adiciona controle de fluxo por canal para manter a comunicação ocorrendo de forma confiável. Essa é uma função muito importante e é requerida na maioria dos casos.
- ✓ IrOBEX: *Object Exchange Protocol*. Fácil transferência de arquivos e outros objetos.
- ✓ IrCOMM: Emulação das portas paralela e serial, habilitando aplicações existentes que usam comunicações serial e paralela a usar infravermelho sem alterações.
- ✓ IrLAN: Acesso a rede local de computadores, habilitando acessos por meio de LAN infravermelha para laptops e outros dispositivos.

Quando a pilha de camadas mostrada na Figura 2.1 integrada em um sistema embarcado, o sistema pode ser representado da maneira abaixo:

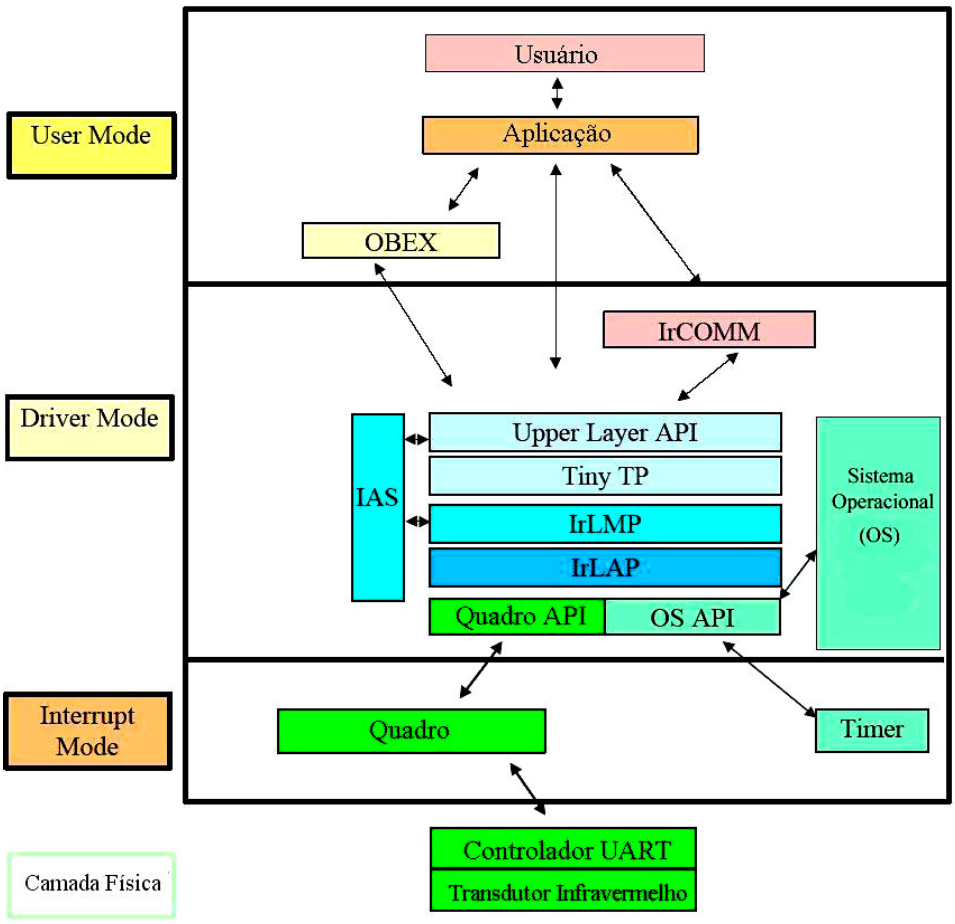


Figura 2.2 Sistema Embarcado.

2.2. CAMADA FÍSICA E ENQUADRAMENTO

A camada física inclui o transmissor-receptor óptico (*transceiver* óptico) e trata da modulação e outras características dos sinais infravermelhos incluindo codificação dos bits de

dados e enquadramento dos dados em quadros adicionando flags de início de término de um quadro (BOF's e EOF's) e checagem redundante cíclica (CRC). Essa camada deve ser pelo menos parcialmente implementada em hardware e encontram-se casos em que ela é totalmente gerenciada pelo hardware.

Para isolar o restante da pilha da camada de hardware, que sempre muda devido à diversidades de ambientes de implementação, uma camada em software chamada enquadrador é criada. Sua responsabilidade primária é aceitar pacotes provenientes do hardware e apresentá-los à camada *Link Access Protocol* (IrLAP). Isso inclui aceitar quadros de saída e fazer o que for necessário para transmiti-los via infravermelho. Além disso, o enquadrador é responsável por mudar as velocidades de transmissão do hardware no processo de rastreamento da camada IrLAP usando quaisquer truques que o projetista de hardware tenha inventado para tal propósito.

2.2.1. IrLAP - *Link Access Protocol*

Imediatamente acima do enquadrador encontramos a camada IrLAP, também conhecida como *Link Access Protocol*, ou LAP abreviadamente. IrLAP é um protocolo IrDA necessário e corresponde à camada 2 OSI (*data link protocol*). É baseado em *High-Level Data Link Control* (HDLC) e *Synchronous Data Link Control* (SDLC) com extensões para algumas características únicas das comunicações infravermelhas. IrLAP garante transferência confiável de dados usando os seguintes mecanismos:

- ✓ Retransmissão;
- ✓ Controle de fluxo em baixo nível. (TinyTP oferece controle de fluxo em alto nível e deve sempre ser usado no lugar do controle de fluxo IrLAP).
- ✓ Detecção de erros.

Em se tratando de transmissão confiável de dados em baixo nível, as camadas superiores estão livres dessa responsabilidade e podem confiar que seus dados irão ser entregues (ou elas pelo menos serão informadas se seus dados não foram entregues). A entrega dos dados podem falhar se a linha de visada for obstruída. Por exemplo, alguém pode por uma xícara de café na linha de visada do feixe infravermelho. Como um exemplo de como o sistema deve reagir nessa situação, uma aplicação operando na pilha pode ser alertada de

uma interrupção no fluxo de dados, permitindo ao sistema alertar o usuário por meio de alguma interface. O usuário pode então resolver o problema (removendo a xícara de café) sem perder a conexão ou os dados transferidos até aquele ponto.

2.2.2. Características Ambientais

Muitos fatores ambientais influenciaram o desenvolvimento da camada IrLAP. Esses incluem:

- ✓ **Ponto a ponto**: As conexões são um a um, como entre câmara e PC ou coletor de dados e impressora. A faixa de operação está tipicamente entre zero e um metro embora uma expansão dessa faixa para dez metros ou mais está em desenvolvimento.

- ✓ **Half-duplex**: A luz infravermelha e, portanto, os dados são enviados em uma direção por vez. Entretanto o link muda de sentido freqüentemente e pode simular comunicações full-duplex em casos onde a temporização não é crítica.

- ✓ **Cone infravermelho estreito**: A transmissão infravermelha é direcional dentro de quinze graus de meio-ângulo de maneira a minimizar a interferência de dispositivos vizinhos.

- ✓ **Nós escondidos**: Outros dispositivos infravermelhos aproximando-se de uma conexão existente podem não estar a par da conexão se ele se aproxima por trás do atual transmissor. Esse dispositivo deve então esperar e ver se o link muda de sentido antes de ele iniciar a comunicação.

- ✓ **Interferência:** IrLAP deve filtrar interferências de lâmpadas fluorescentes, outros dispositivos infravermelhos, luz do sol, raio de lua etc.

- ✓ **Não detecção de colisão:** O projeto de hardware é feito tal que colisões não são detectadas e o software deve cuidar dos casos onde as colisões causam perda de dados com métodos como *back off* randômico.

2.2.3. Papéis numa conexão LAP

As duas partes de uma conexão LAP possuem uma relação mestre-escravo com diferentes responsabilidades (e complexidade de códigos). Os termos IrDA para eles são Primários (mestre) e Secundários (escravo).

Estação Primária:

- ✓ Envia quadros comandos: inicia as conexões e transferências.
- ✓ Responsável pela organização e controle do fluxo dos dados.
- ✓ Trata os erros irrecuperáveis do link de dados.
- ✓ Os dispositivos primários típicos são PC's, PDA's, câmaras e tudo que precisa de uma impressora (impressoras são geralmente dispositivos secundários).

Estação Secundária:

- ✓ Envia quadros de resposta: apenas fala quando solicitado.
- ✓ Dispositivos secundários típicos são impressoras e outros periféricos, bem como dispositivos com recursos limitados (secundários são menores e menos complexos).

Em qualquer conexão um dos dispositivos deve exercer o papel de primário. O outro dispositivo deve, portanto, exercer o papel de secundário. Uma vem iniciada a conexão, os dois lados alternam-se na comunicação com a liderança do primário deposta. Nenhum dos lados pode falar mais que 500ms por vez antes dar ao outro lado a chance de falar (mesmo se for apenas para dizer que não possui informações a serem enviadas no momento).

2.3. MODOS

IrLAP é construído sobre dois modos de operação, correspondendo à existência ou não de conexão.

2.3.1. Modo Normal Desconectado (*Normal Disconnect Mode*)

NDM é também conhecido como estado de contenção e é o estado padrão de dispositivos desconectados. Nesse modo, um dispositivo deve observar um conjunto de regras de acesso ao meio de comunicação infravermelho. De maior importância, um dispositivo em NDM deve checar a ocorrência de outras transmissões (uma condição conhecida como meio ocupado: *media busy*) antes de transmitir. Se nenhuma atividade é detectada dentro de 500ms

(o tempo máximo para a mudança de direção na comunicação), o meio é considerado disponível para o estabelecimento de uma conexão.

Uma característica de fácil uso é fornecida pelas regras de comunicação do modo NDM. Um problema clássico é ter ambos os lados da conexão configurados com os mesmos parâmetros de comunicação - freqüentemente os usuários ficam completamente confusos quando são solicitados a configurar conexões indicando a taxa de transferência, presença e quantidade de stop bits etc. Isso pode ser particularmente difícil em sistemas embarcados que não possuem interface com o usuário para ajustar e verificar os parâmetros de comunicação. Esse problema é completamente ausente em comunicações IrDA - todas as comunicações em NDM usam os seguintes parâmetros: ASYNC, 9600bps, 8bit sem paridade. Durante o processo de conexão, os dois lados trocam informações a respeito de suas capacidades e posteriormente ajustam-se para os melhores parâmetros suportados por ambos os lados.

2.3.2. Modo Normal de Resposta (*Normal Response Mode*)

NRM é o modo de operação para dispositivos conectados. Uma vez iniciada a conversa entre os dispositivos usando os melhores parâmetros de comunicação suportados (estabelecidos no NDM), as camadas altas da pilha usam quadros normais de comando e resposta para trocarem informações.

2.3.3. Formato do Quadro IrLAP

Figura 2.3 O formato básico do formato do quadro IrLAP

Enquanto há muita informação a respeito do formato dos quadros, é interessante observar que os campos de endereço e controle necessitam de apenas dois bytes no total - os protocolos IrDA adicionam pouco *overhead* aos dados do usuário.

2.3.4. Enquadradores IrLAP

Antes de mandar, um quadro é enquadrado com as suas respectivas informações. Três enquadradores diferentes são usados por IrLAP dependendo da velocidade da conexão.

- ✓ Enquadramento Assíncrono ASYNC: 9600bps a 115,2 kbps.
- ✓ Enquadramento Síncrono HDLC: 579 kbps e 1,152 Mbps.
- ✓ Enquadramento Síncrono 4 PPM: 4 Mbps.

2.3.5. Diagrama de Primitivas de Serviço

As operações IrLAP são descritas na especificação usando primitivas de serviço. Podemos imaginar uma primitiva de serviço como um modelo conceitual de uma API para uma operação que o IrLAP realiza (as API's reais para os serviços IrLAP dependem completamente do desenvolvedor). O diagrama da Figura 2.4 ilustra uma operação: ele começa com um requisição de serviço, que atravessa o link como um quadro e é reportada como uma indicação (freqüentemente uma chamada) no lado receptor; o receptor formula a resposta que atravessa novamente o link como um quadro, resultando em uma confirmação (freqüentemente uma chamada) para o requerente.

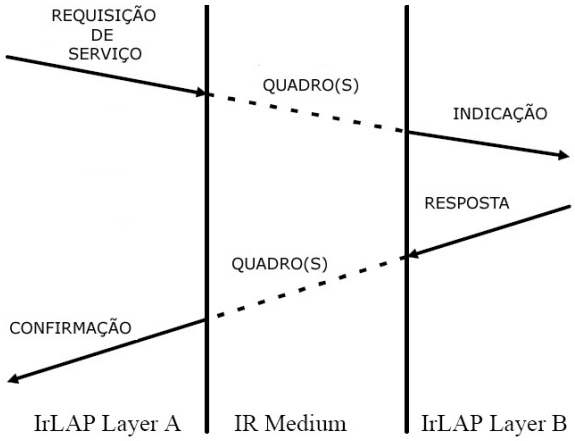


Figura 2.4 Diagrama de conexão IrLAP

2.3.6. Serviços IrLAP

Um número de serviços é definido na especificação IrLAP. Nem todos os serviços são necessários a todos os dispositivos e a especificação IrLAP (juntamente com o padrão IrDA

Light) descreve os requerimentos mínimos. Os serviços mais importantes incluem os seguintes:

- ✓ **Descoberta de Dispositivos:** exploração da vizinhança infravermelha para ver quem está presente e descobrir algumas de suas capacidades.

- ✓ **Conexão:** Escolha de um parceiro específico, negociação dos melhores parâmetros de comunicações suportados por ambos os lados e conexão ao dispositivo.

- ✓ **Envio de dados:** A única razão para todo esse esforço - usado pelas camadas superiores em todos os seus trabalhos.

- ✓ **Desconexão:** Fecha a conexão e retorna ao estado NDM, pronto para uma nova conexão.

2.4. IRLMP: *LINK MANAGEMENT PROTOCOL*

A camada IrLMP depende do quão confiável é a conexão e os parâmetros de conexão fornecidos pela camada IrLAP. IrLMP é uma camada IrDA necessária e disponibiliza as seguintes funcionalidades:

- ✓ **Multiplexação:** LMP permite vários clientes IrLMP funcionarem em um único link IrLAP.

- ✓ **Descoberta de alto nível**, consistindo de:

  - ➢ Resolução de conflitos de endereço da Procura por Dispositivos IrLAP. Trata o caso em que múltiplos dispositivos com o mesmo endereço IrLAP; faz isso por meio da solicitação de geração de novos endereços para os dispositivos conflitantes.

- **Serviço de Informação de Acesso (IAS):** Uma espécie de páginas amarelas descrevendo os serviços disponíveis no dispositivo.

2.4.1. Terminologia IrLMP

De maneira a ter múltiplas conexões IrLMP em uma única conexão IrLAP, deve existir um esquema de endereçamento de alto nível. A seguinte terminologia é usada para descrever esse endereçamento:

- ✓ LSAP (*Logical Service Access Point*): O ponto de acesso ao serviço ou aplicação no IrLMP (por exemplo, um serviço de impressão). É referenciado com um único número armazenável em um byte, o LSAP-SEL.

- ✓ LSAP-SEL (LSAP *Selector*): Um número armazenável em um byte que corresponde a um LSAP. Podemos encará-lo como o endereço de um serviço em um multiplexador LMP. Esse byte é quebrado em duas faixas - 0x00 é o servidor IAS, 0x01 a 0x6F são conexões LMP legais, 0x70 são para serviços sem conexões e o restante é reservado para uso futuro.

Dado o número limitado de valores LSAP-SEL, serviços não são associados a "endereços de portas" fixos como em TCP/IP. Em vez disso, ele possuem nomes comuns fixos e o LMP IAS (páginas amarelas) é usado para verificar o LSAP-SEL para o dispositivo desejado.

2.4.2. Serviços IrLMP

Aqui estão os serviços definidos na especificação IrLMP. Nem todos os serviços são necessários em todos os dispositivos e a especificação (juntamente com o padrão IrDA *Light*) descreve o mínimo de requerimentos. Note que esse é um conjunto idêntico ao conjunto listado na seção a respeito de IrLAP.

- ✓ **Descoberta de Dispositivos**: Procura por informações adicionais sobre os dispositivos na vizinhança infravermelha.
- ✓ **Conexão:** Estabelece uma conexão entre um par de serviços ao nível LMP.

- ✓ **Dados:** Manda e recebe dados
- ✓ **Desconexão:** Fecha a conexão LMP específica. Note que isso não necessariamente fecha a conexão IrLAP.

2.4.3. Formato do Quadro

A camada IrLMP adiciona os seguintes dois bytes de informação aos quadros de maneira a executar as operações básicas:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
C	DLSAP-SEL							r	SLSAP-SEL						

Figura 2.5 Quadro IrLMP

- ✓ **C:** Distingue entre quadros de controle e dados.
- ✓ **r:** Reservado.
- ✓ **DLSAP-SEL:** LSAP-SEL (endereço do dispositivo) para o destino do quadro atual.
- ✓ **SLLSAP-SEL:** LSAP-SEL para o emissor do quadro atual.

2.4.4. IAS - Serviço de Informações de Acesso (*Information Access Service*)

O IAS (páginas amarelas para serviços e aplicações), ou Serviço de Informações de Acesso, atua como as "páginas amarelas" para um dispositivo. Todos os serviços e aplicações disponíveis para conexões iniciantes devem ter entradas no IAS que podem ser usadas para determinar o endereço do serviço (LSAP-SEL). O IAS pode também ser questionado a respeito de informações adicionais sobre os serviços.

Uma implementação completa do IAS consiste dos componentes cliente e servidor. O cliente é o componente que faz perguntas sobre serviços do outro dispositivo usando o Protocolo de Informações de Acesso (*Information Access Protocol*, IAP, usado somente no IAS). O servidor é o componente que sabe como responder às perguntas de um cliente IAS. O servidor usa uma base de informações de objetos fornecidos pelas aplicações e serviços locais. Em sistemas embarcados com propósito especifico, essa base pode ser uma coleção estática de objetos, enquanto que em um PDA podem existir API's para registrar e desregistrar

serviços. Note que dispositivos que nunca iniciam conexões LMP podem ser incluídos somente no servidor IAS.

2.4.5. O modelo de informações IAS

A Base de Informações IAS é uma coleção de objetos que descrevem os serviços disponíveis para conexões iniciantes. A base de informações é usada pelo servidor IAS para responder a perguntas IAS provenientes das conexões iniciantes.

Os objetos da base de informações consistem de um nome de classe e um ou mais atributos. Eles são similares às entradas nas páginas amarelas de um catálogo telefônico. O nome de classe é o equivalente ao tipo de negócio no catálogo telefônico - esse é o nome publicado oficial do serviço ou aplicação. Os clientes IAS irão questionar sobre um serviço usando esse nome. Os atributos contêm informações análogas ao número do telefone, endereço e outras características do negócio. Um atributo essencial para cada entrada é o LSAP-SEL (ou endereço de serviço) que é necessário na realização de uma conexão LMP ao serviço. Um objeto IAS é feito das seguintes peças:

- ✓ Nome da classe (até 60 bytes).
- ✓ Atributos nomeados (nomes de até 60 bytes)
  - Até 256 atributos.
  - Tipos de valores dos atributos:
  - string (até 256 bytes)
  - seqüência de octetos (até 1024 bytes)
  - inteiro com sinal (32 bits)

Obtendo informações usando o IAS:

Há um número de operações IAS pré-definidas no padrão IrLMP, mas a mais usada e a única necessária é chamada de GetValueByClass. A parte solicitante dá o nome da classe (por exemplo, *Printer* - impressora) e o nome do atributo que ela deseja (por exemplo, o LSAP-SEL) e recebe um retorno que consiste de umas ou mais respostas (por exemplo, os LSAP-SEL's de quaisquer serviços de impressão na base de informações da parte solicitada) ou uma indicação de que o serviço ou atributo não existe.

Parâmetros de Perguntas IAS:

- ✓ *Class Name Length*, comprimento do nome da classe (um bytes).
- ✓ *Class Name*, nome da classe (comprimento bytes).
- ✓ *Attribute Name Length*, comprimento do nome do atributo (um bytes).
- ✓ *Attribute Name*, nome do atributo (comprimento bytes).

Resulta em:

- ✓ Código de retorno:
  - 0: sucesso, segue o resultado.
  - 1: não existe tal classe, sem resultado.
  - 2: não existe tal atributo, sem resultado.

Se o código de retorno indica sucesso, a chamada retorna as seguintes informações:

- ✓ *List Length*, comprimento da lista (dois bytes).
- ✓ *List of results*, lista de resultados:
  - Identificador de Objeto (dois bytes).
  - Valor do atributo (baseado no tipo de atributo).

2.5. TINYPT - *TINY TRANSPORT PROTOCOL*

A fim de entender melhor o TinyTP, ajuda rever as camadas apresentadas até então:

- ✓ A Camada Física define os requisitos de hardware e o enquadramento em baixo nível dos dados.
- ✓ IrLAP fornece conexão segura, seqüencial e sem problemas sobre os melhores parâmetros de conexão escolhidos na negociação automática.
- ✓ IrLMP fornece multiplexação de serviços na conexão LAP e as páginas amarelas do IAS para os serviços disponíveis nas conexões iniciantes.

TinyTP (abreviadamente TTP) é uma camada opcional do IrDA embora seja tão importante que deva ser considerado como uma camada necessária (exceto no caso de soluções de serviços de impressão) . TTP fornece duas funções:

- ✓ Controle de fluxo por conexão LMP (por canal).
- ✓ SAR (segmentação e remontagem).

TTP adiciona um byte de informação a cada pacote IrLMP para executar a sua tarefa.

2.5.1. Controle de Fluxo TinyTP

Controle de fluxo por canal é o mais importante uso do TinyTP. O que justificaria outro mecanismo de controle de fluxo já que o IrLAP oferece controle de fluxo? Para ilustrar a necessidade, suponhamos que uma conexão LAP é estabelecida e duas conexões LMP são feitas no topo da conexão LAP usando a capacidade de multiplexação LMP. Se um lado aciona o controle de fluxo de dados, o fluxo de dados da conexão LAP (que carrega todas as conexões LMP) é completamente interrompido naquele sentido e o outro lado não pode receber dados necessários até o controle de fluxo LAP ser desligado. O trabalho do segundo dispositivo será seriamente interrompido (especialmente se temporizadores estiverem envolvidos).

Se o controle de fluxo é aplicado por conexão LMP usado TinyTP (ou outros mecanismos), então o um lado pode parar de processar as informações sem afetar negativamente o outro lado.

2.5.2. Como o controle de fluxo TTP funciona

TTP é um esquema de controle de fluxo baseado em créditos. Ele opera da seguinte maneira:

Na conexão, alguns créditos são oferecidos para cada lado. Um crédito corresponde à permissão para enviar um pacote LMP. Enviado um crédito, o dispositivo deve ser capaz de aceitar um pacote de tamanho máximo. Podemos ver que o número de créditos oferecidos

depende inteiramente do tamanho do espaço em buffer disponível. À medida que você tem *buffers*, é permitido enviar de um a 127 créditos.

O envio de dados causa uso de créditos (uma unidade de crédito por pacote enviado). Periodicamente o receptor emite mais créditos. Essa "política de créditos" está sobre a responsabilidade do receptor, mas a política pode fazer grande diferença no desempenho do *link*. Se o emissor fica sem créditos constantemente e tem que esperar por mais créditos, o *throughput* irá ser prejudicado. Se o emissor não possui mais crédito, nenhuma troca de dados pode ocorrer mas, um pacote de um crédito pode sempre ser enviado (ele não é sujeito a controle de fluxo).

Embora a descrição acima fale sobre o emissor e o receptor como se esses papéis fossem fixos, é comum para ambos os lados de uma conexão LMP enviar e receber, portanto, ambos os lados estarão emitindo créditos e consumindo-os. Note que o byte de créditos normalmente atravessa como parte do pacote de dados LMP, então pacotes LMP não são necessários para enviar créditos à medida que há dados a serem enviados e créditos a serem enviados consigo. Obviamente um pouco de trabalho é necessário de maneira a ajustar eficientemente a política de créditos.

2.5.3. Segmentação e Remontagem

A outra função do TTP é chamada de SAR (segmentação e remontagem). A idéia básica é que o TTP quebre grandes pacotes em pedaços (segmentação) e os ponha juntos do outro lado da conexão (remontagem). O pedaço inteiro de dados sendo fragmentado e reconstituído é chamado SDU, *Service Data Unit* (unidade de dados de serviço), e o tamanho máximo do SDU é negociado quando a conexão TTP/LMP é feita pela primeira vez.

2.5.4. Primitivas de Serviço TinyTP

Da mesma maneira que as camadas IrLAP e IrLMP, as operações TTP são caracterizadas por primitivas de serviços:

- ✓ ***Connect*:** negocia o tamanho máximo do SDU.
- ✓ ***Disconnect.***
- ✓ ***Data*:** seqüências de dados confiáveis.
- ✓ ***Local Flow Control*:** parar entrega de dados.
- ✓ ***Udata*:** dados não seqüenciais e não confiáveis (passa através do IrLMP).

As primitivas de serviços TTP enfatizam-se no núcleo das primitivas LMP - *connect, send* e *disconnect*, adicionando meios de exercer o controle de fluxo.

2.5.5. Formatos de quadros TTP

Os dois formatos de quadros usados pelo TTP são o pacote *connect* (carregado como o pacote *connect* IrLMP, portanto de tamanho limitado) e o pacote de dados, carregados com pacotes de dados IrLMP.

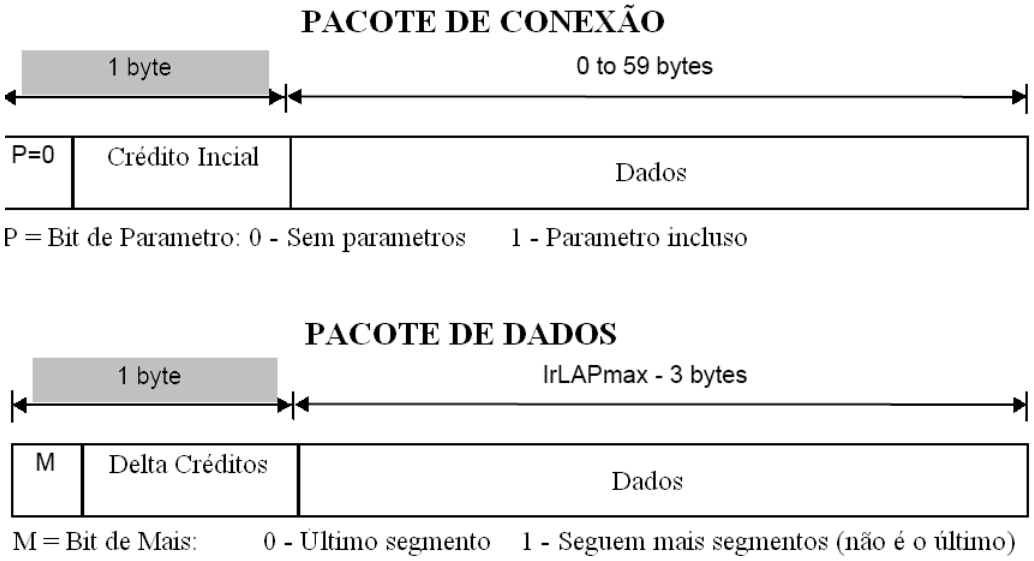


Figura 2.6 Pacote de Conexão

2.6. VISÃO GERAL DO IROBEX

O IrOBEX é um protocolo de camada de aplicação opcional projetado para habilitar sistemas de todos tipos e tamanhos a trocarem uma ampla variedade de dados e comandos de uma maneira padronizada e sensível aos recursos. Ele realiza uma das operações mais comuns em PC's ou sistemas embarcados: pegar um objeto arbitrário de dados (um arquivo, por exemplo), e enviá-lo para o dispositivo infravermelho para onde está apontando. Ele também

oferece algumas ferramentas para habilitar o objeto a ser reconhecido e manipulado inteligentemente no lado receptor.

O gama de objetos é enorme, envolvendo não somente arquivos tradicionais mas páginas, recados de telefone, imagens digitais, cartões de negócios eletrônicos, registros de banco de dados, resultados de instrumentos portáteis ou diagnósticos e programação. Um ponto importante é que a aplicação não precisa ou não quer estar envolvida com o gerenciamento de conexões ou tratamento do processo de comunicação como um todo. Apenas pegar o objeto e enviar ao outro lado com o mínimo de trabalho possível. É muito similar ao papel que o HTTP serve no conjunto de protocolos da Internet, apesar do HTTP ser muito orientado a requisições em sua concepção fundamental enquanto que o OBEX é mais balanceado.

2.6.1. Características do Protocolo IrOBEX

OBEX foi criado a fim de disponibilizar uma comunicação IrDA o mais completo possível e, portanto, simplificando o desenvolvimento de aplicações que possuem comunicações habilitadas. Foi concebido para satisfazer os seguintes critérios:

- ✓ **Simples:** suporta as operações e aplicações mais necessárias.
- ✓ **Compacto:** código de menos de 1K em sistemas pequenos.
- ✓ **Flexível:** suporta processamento dos dados para padrões industriais e sistemas específicos.
- ✓ **Extensível e depurável.**
- ✓ **Trabalha com IrDA**, mas é independente no transporte.

2.6.2. Componentes do protocolo IrOBEX

O padrão OBEX consiste das seguintes partes:

- ✓ **Modelo de sessão**: as regras de conversação que governam a troca de objetos. Inclui negociação opcional durante a conexão, um conjunto de operações como Put e Get. Permite o término da transferência de um objeto sem fechar a conexão. Suporta fechamento suave de conexão.

- ✓ **Modelo de objeto:** oferece uma representação flexível e extensível de objetos e as informações que os descrevem.
- ✓ **Guias para uso e extensão:**
  - Definição de novas operações de sessão.
  - Definição de novos tipos de objetos.
- ✓ A entrada IAS para um servidor padrão OBEX e sugestões para suas características.

2.7. IRCOMM - EMULAÇÃO DAS PORTAS SERIAL E PARALELA

Quando os padrões IrDA foram desenvolvidos, havia um grande desejo de permitir as aplicações existentes em PC que usavam as portas serial e paralela operarem via infravermelho sem mudanças. Essas aplicações, comumente conhecidas como aplicações legadas, incluem impressão, operações de transferência de arquivos como *LapLink* e comunicações via modem.

Entretanto, as comunicações em infravermelho diferem significativamente das comunicações paralela e serial. Por exemplo, ambos os cabos serial e paralelo possuem circuitos individuais nos quais os sinais podem ser enviados independentemente e concorrentemente. Em contraste, infravermelho possui um simples feixe de luz e toda a informação deve ser posta em LMP ou em pacotes de camadas superiores no fluxo serial.

O padrão IrCOMM foi desenvolvido para resolver esses problemas e permitir aplicações legadas serem usadas com o mínimo de trabalho. A chave do IrCOMM é a definição de um canal de controle para carregar as informações de circuito. Na figura da pilha, IrCOMM fica no topo do IrLMP e do TinyTP.

IrCOMM é um protocolo opcional IrDA adequada apenas a algumas aplicações. Em geral, novas aplicações são melhores servidas se evitarem IrCOMM e usarem outros protocolos de aplicações IrDA como TrOBEX, IrLAN ou TinyTP. Isso é porque IrCOMM mascara algumas das mais úteis características construídas sobre as camadas inferiores. Além disso, é seu trabalho fazer o IrDA parecer-se como os meios serial ou paralelo que não possuem característica úteis como negociação dos melhores parâmetros de conexão em comum e as páginas amarelas dos serviços disponíveis.

2.7.1. Tipos de Serviços IrCOMM

Devido ao fato de diferentes aplicações usarem circuitos de comunicações serial e paralela de maneiras diferentes, quatro tipos de serviços são definidos em IrCOMM:

- ✓ ***3-Wire Raw*** (emulação serial e paralela): envia apenas dados, sem informações de circuito e portanto sem controle de canal. Funciona diretamente sobre a IrLMP.
- ✓ ***3-Wire*** (emulação serial e paralela): uso mínimo do canal de controle. Usa o TinyTP.
- ✓ ***9-Wire*** (emulação serial apenas): usa o canal de controle para o estado dos circuitos RS-232. Usa o TinyTP.
- ✓ ***Centronics*** (emulação paralela apenas): usa o canal de controle para o estado do circuito *Centronics*. Usa TinyTP.

3. O MICROCONTROLADOR 8051

Um microcontrolador é um componente que tem, num único chip, além de uma CPU, elementos tais como memórias ROM e RAM, temporizadores/contadores, PWM, conversor AD, canais de comunicação e conversores analógico-digitais. Esta característica diferencia os sistemas baseados em microcontroladores daqueles baseados em microprocessadores, onde normalmente se utilizam vários componentes para implementar essas funções. Com isso, os microcontroladores permitem a implementação de sistemas mais compactos e baratos do que aqueles baseados em microprocessadores.

Em contrapartida, as CPUs dos microcontroladores são, menos poderosas do que os microprocessadores. Seu conjunto de instruções costuma se limitar às instruções mais simples, sua freqüência de clock é mais baixa e o espaço de memória endereçável costuma ser bem menor. Vê-se daí que o campo de aplicação dos microcontroladores é diferente daquele dos microprocessadores, e que um sistema que possa ser controlado por um microcontrolador tende a ter menor complexidade e menor custo do que um sistema que exija a capacidade de processamento de um microprocessador.

Exemplos de sistemas onde os microcontroladores encontram aplicação incluem controle de semáforos, balanças eletrônicas, microterminais, telefones públicos, controle de carregadores de baterias, inversores eletrônicos, controles de acesso, taxímetros, sistemas de aquisição de dados de manufatura e eletrodomésticos em geral.

A programação dos microcontroladores é mais simples do que a dos microprocessadores. Isto acontece porque os periféricos on-chip dos microcontroladores são acessados de uma forma padronizada e integrada na própria linguagem de programação, dispensando o conhecimento de detalhes externos. Não se deve pensar, porém, que isto simplifique a tarefa do programador em todos os níveis: é necessário que ele conheça bem o hardware conectado ao microcontrolador para poder produzir programas que funcionem corretamente.

Cabe citar ainda uma vantagem particular dos microcontroladores que possuem memória ROM, que é a possibilidade de armazenar programas internamente, dificultando sensivelmente a cópia ilícita do código.

Diversos fabricantes produzem microcontroladores da família 8051 (Intel, AMD, Atmel, Dallas, OKI, Matra, Philips, Siemens, SMC, SSI). A Intel iniciou a produção do 8051 em 1981. Em 1982 foram produzidos 2 milhões de unidades, em 1985 foram 18 milhões e em 1993, 126 milhões [2] e [7].

Além do 8051 propriamente dito, existem variantes como o 8031 (sem memória ROM interna e com apenas 128 bytes de memória RAM), o 8751 (4 kB de memória EPROM) e o 8052 (8 kB de memória ROM, um terceiro timer e 256 bytes de memória RAM) [2] e [7].

3.1. PRINCIPAIS CARACTERÍSTICAS

- ✓ Freqüência de *clock* de 12 MHz, com algumas versões que alcançam os 40 MHz;
- ✓ até 64 kB de memória de dados externa;
- ✓ 128 bytes de RAM interna;
- ✓ até 64 kB de memória de programa configurável de duas formas mutuamente excludentes:
  - 4 kB internos (ROM no 8051 e EPROM no 8751) e mais 60 kB externos;
  - 64 kB externos;
- ✓ 4 portas bidirecionais de I/O, cada uma com 8 bits individualmente endereçáveis; duas dessas portas (P0 e P2) e parte de uma terceira (P3) ficam comprometidas no caso de se utilizar qualquer tipo de memória externa;
- ✓ 2 temporizadores /contadores de 16 bits;
- ✓ 1 canal de comunicação serial;
- ✓ 5 fontes de interrupção (dois *timers*, dois pinos externos e o canal de comunicação serial) com 2 níveis de prioridade selecionáveis por software;
- ✓ oscilador de *clock* interno.

As características citadas são básicas e formam o núcleo da família 8051, que pode ser acrescido de uma ou mais das características especiais mostradas na Figura 3.1.

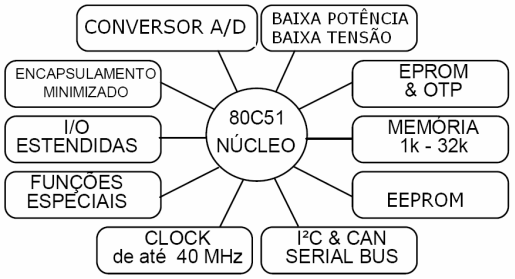


Figura 3.1 Características especiais da família 8051

A título de exemplo, a Tabela 3.1 apresenta as características de alguns componentes da família 8051 com a indicação de seus fabricantes. O número total de variantes é muito maior. A Philips, por exemplo, produz mais de 40 tipos diferentes.

Tipo	**Pinos**	**Fabricante.**	**RAM**	**CODE**	**Notas** (***LV - low voltage***)
MCS251	40	Intel	1K	16K	16 Bit 80x51 FX Preliminar
80C517A	84	Siemens	256	64Kx	ALU; 8 PWM; CC ; 2UART; 10bA/D
80C537A	84	Siemens	256	32K	ALU; 8 PWM; CC ; 2UART; 10bA/D
80C535A	68	Siemens	256	64Kx	515+10bA/D; 1KXRAM; BRG; OWD[1]
80C515A	68	Siemens	256	32K	515+10bA/D;1KXRAM; BRG; OWD
80535	68	Siemens	256	64Kx	Timer 2 Capt Comp 6 ports 8/10bA/D
80515	68	Siemens	256	8K	Timer 2 Capt Comp 4 ports 8b A/D
80C535	68	Siemens	256	64Kx	Timer 2 Capt Comp 5 ports 8b A/D
80C51GB	68	Intel	256	64Kx	8051FA+PCA; 8bA/D; SPI
87C51GB	68	Intel	256	8K	8051FA+PCA; 8bA/D; SPI
80C592	68	Philips	256	64Kx	552-I_2C+CAN+XRAM
87C598	80	Philips	256	32K	552-I_2C+CAN+XRAM
80C552	68	Philips	256	64Kx	10bA/D; I2C; Capt Comp; PWM
87C552	68	Philips	256	8K	10bA/D; I2C; Capt Comp; PWM

Tabela 3.1 Alguns integrantes da família MCS51.

[1] OWD - Oscillator Watchdog [7], demais siglas podem ser explicadas na Lista de Abreviaturas, pag. xii.

3.2. ANÁLISE EXTERNA

O aspecto externo do 8051 quando empregado em encapsulamento DIP40 é o da Figura 3.2. Os pinos com nomes da forma P0.0, P0.1, etc. correspondem às quatro portas de E/S (P0 a P3). Note a dupla utilidade das portas P0 e P2, que ficam comprometidas com o uso de memória externa, assim como os pinos P3.6 e P3.7. O sinal ALE (*Address Latch Enable*) permite fazer a demultiplexação de dados e endereços na porta P0, conforme também é mostrado no esquema da Figura 3.2.

Através do sinal PSEN (*program storage enable*), o controlador informa o mundo externo se a operação em andamento é uma leitura de instrução (acesso à memória de programa) ou um acesso à memória de dados. Este sinal permite que o processador tenha duas regiões distintas de memória externa, uma para armazenar código e outra para dados. Ambas ocupam os endereços de 0 a FFFFH (64 kB), num total de 128 kB [2].

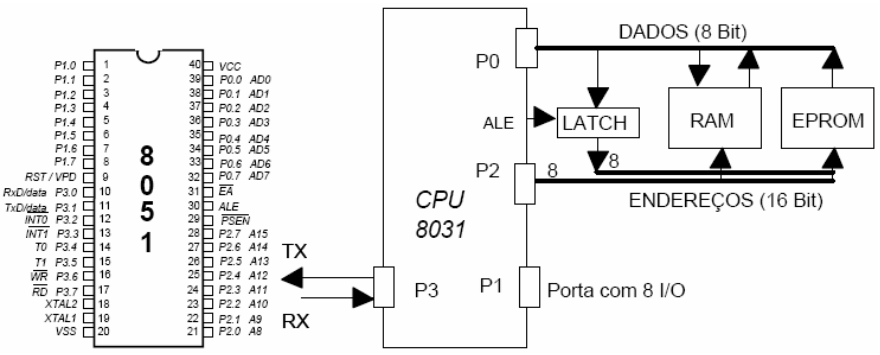


Figura 3.2 Pinagem e diagrama em bloco da expansão da memória.

O pino EA é um sinal de entrada, através do qual o usuário escolhe se será utilizada a memória ROM interna ou se todo o programa será armazenado externamente. Os pinos da porta P3 também são utilizados para realizar certas funções especiais:

- ✓ **P3.0** - RxD/*data* - recepção serial assíncrona ou E/S de dados síncronos;
- ✓ **P3.1** - TxD/*clock* - transmissão porta serial assíncrona ou saída de *clock* p/ dados síncronos;
- ✓ **P3.2** - INT0 - entrada da interrupção 0 ou *bi*t de controle para o temporizador/contador 0;
- ✓ **P3.3** - INT1 - entrada da interrupção 1 ou *bit* de controle para temporizador/contador ;
- ✓ **P3.4** - T0 - entrada de *clock* externo para o temporizador/contador 0;

- ✓ **P3.5** - T1 - entrada de *clock* externo para o temporizador/contador 1;
- ✓ **P3.6** - WR - sinal de escrita na memória de dados externa;
- ✓ **P3.7** - RD - sinal de leitura na memória de dados externa.

A alimentação (5V) é feita pelo pino 40 e o GND é o pino 20; o cristal para o oscilador interno é conectado aos pinos 18 e 19. Finalmente, o pino 9, RST/VPD, é a entrada de reset.

3.3. MODELO DE PROGRAMAÇÃO SIMPLIFICADO

Desenhar um modelo de programação completo para um microcontrolador é mais difícil do que para um microprocessador, porque os microcontroladores têm, em geral, um número bem maior de registradores, que servem para acessar os componentes periféricos *on-chip*. O modelo da Figura 3.3 contempla apenas os registradores de caráter geral.

O registrador A é o acumulador que, como no 8085, é responsável pelas principais operações, sobretudo as lógicas e aritméticas. B é um registrador de caráter geral, assim como os oito registradores R0 a R7. DH e DL também são de uso geral, mas podem ser utilizados como um registrador de 16 bits, que se denomina DPTR. Este registrador é o único que pode conter valores de 16 bits e por isso é freqüentemente utilizado no endereçamento da memória externa, que é sempre indireto. PC contém o endereço da próxima instrução executável e SP aponta para o topo da pilha.

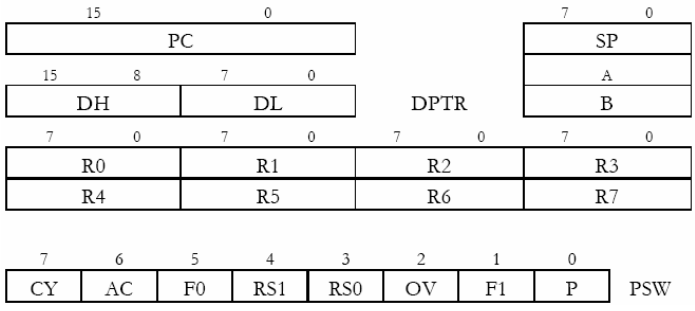


Figura 3.3 Registros internos do 8051

O 8051 conta com quatro *flags* de sistema (*carry, auxiliary carry, overflow* e *parity*) e dois flags de usuário (F0 e F1), que o programador pode utilizar como desejar. Os flags estão

todos reunidos no registrador PSW (*program status word*), que contém ainda os bits de seleção de banco de registradores, RS0 e RS1.

3.3.1. Os registradores de funções especiais

Os registradores de funções especiais incluem posições de acesso às portas de E/S, registradores de interrupção, registradores da porta serial, temporizadores e registradores aritméticos. Aqueles situados em endereços múltiplos de 8 também podem ser endereçados bit a bit. Os registradores são descritos resumidamente a seguir:

- ✓ P0 (80H), P1(90H), P2 (A0H) e P3(B0H) correspondem a posições de RAM contendo os dados das portas de E/S; os bits individuais são endereçados como P0.0, P0.1, etc.;
- ✓ TH1 (8DH), TL1 (8BH), TH0 (8CH) e TL0 (8AH) contêm os valores das contagens dos temporizadores/contadores 1 e 0, respectivamente;
- ✓ TCON (88H) e TMOD (89H) são os registradores de controle e modo de operação dos temporizadores/contadores;
- ✓ PCON (87H) permite adaptar o chip a uma situação na qual não há processamento, mas onde se quer manter os conteúdos das memórias internas (falha de alimentação, por exemplo);
- ✓ SCON (98H) e SBUF (99H) permitem, respectivamente, programar a porta de comunicação serial e armazenar o dado recebido ou a ser transmitido;
- ✓ IE (A8H) e IP (B8H) são registradores associados à gestão de interrupção (habilitação e prioridade);
- ✓ SP (81H), PSW (D0H), A (E0H), B (F0H), DPH (83H) e DPL (82H) .

Endereço	0	1	2	3	4	5	6	7	**Endereço**
F8	B								**FF**
F0	B								**F7**
E8									**EF**
E0	ACC								**E7**
D8									**DF**
D0	PSW								**D7**
C8									**CF**
C0									**C7**
B8	IP								**BF**
B0	P3								**B7**
A8	IE								**AF**
A0	P2								**A7**
98	SCON	SBUF							**9F**
90	P1								**97**
88	TCON	TMOD	TL0	TL1	TH0	TH1			**8F**
80	P0	SP	DPL	DPH				PCON	**87**

Tabela 3.2 Registradores e funções especiais

4. TRANSMISSOR E RECEPTOR ASSÍNCRONO - MAX 3100

A UART MAX3100 (UART - *universal asynchronous receiver trasmitter*, transmissor receptor universal assíncrono) é a primeira UART otimizada especificamente para pequenos sistemas baseados em microcontroladores. Usa uma interface SPI / *Microwire* para se comunicar com o microcontrolador *host*,.A entrada / saída assíncrona é ideal para o uso em RS-232, RS-485, IR e para comunicações isolação optica. Comunicações em infravermelho são facilmente implementadas com o modo de temporização IrDA disponibilizado pelo MAX3100 [4] e [13].

O MAX3100 possui um oscilador a cristal e um gerador de *baud-rate* (taxa de simbolo) com divisor de taxa programado via *software*, resultando em um *baud rate* de 300 *baud* a 230k *baud.*

Uma fila de buffer de 8 palavras minimiza o processamento de *overhead*. O dispositivo também inclui interrupção flexível com quatro fontes mascaráveis, incluindo reconhecimento redes de endreços com 9 bits. Possui também duas linhas de controle de *hardware-handshaking* (uma de entrada e outra de saída).

4.1. MODO SERIAL INFRAVERMELHO IRDA

O modo IrDA do MAX3100 pode ser usado para realizar a comunicação com outros dispositivos IrDA SIR - compatíveis (SIR - serial infravermelho) ou para reduzir o consumo de potência em aplicações opto-isoladas.

No modo IrDA, o período de um bit é reduzido a 3/16 do período do *baud* (1,6μs a 115.200 *baud*):

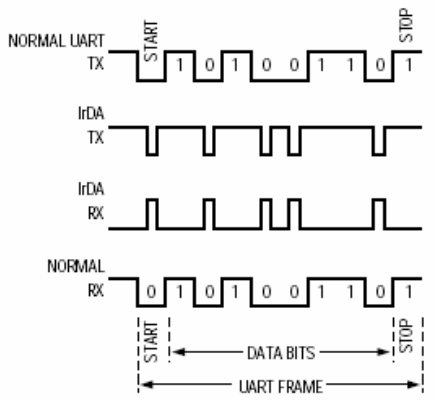


Figura 4.1 Temporização IrDA

Um dado zero é transmitido como um pulso de luz (pino TX = lógica negativa, pino RX = lógica positiva).

No modo de recepção, a amostra do sinal recebido é feita na metade da transmissão de um nível lógico alto. A amostra é feita uma vez apenas, em vez de três vezes, como é feito no modo normal. O MAX3100 ignora pulsos mais curtos que aproximadamente 1/16 do período de um *baud*. O dispositivo IrDA que está se comunicando com o MAX3100 deve estar ajustado para enviar pulsos de largura de 3/16 do período do *baud*. Para compatibilidade com outros dispositivos IrDA devemos ajustar a UART para trabalhar com 8 bits de dados, um bit de parada, sem paridade.

4.2. MÓDULO IRDA

O MAX3100 foi otimizado para acionar optoacopladores enquanto os módulos IrDA possuem *buffers* inversores. É necessária a inversão das saídas TX e RX como mostrado na Figura 4.2 abaixo [4] e [13]:

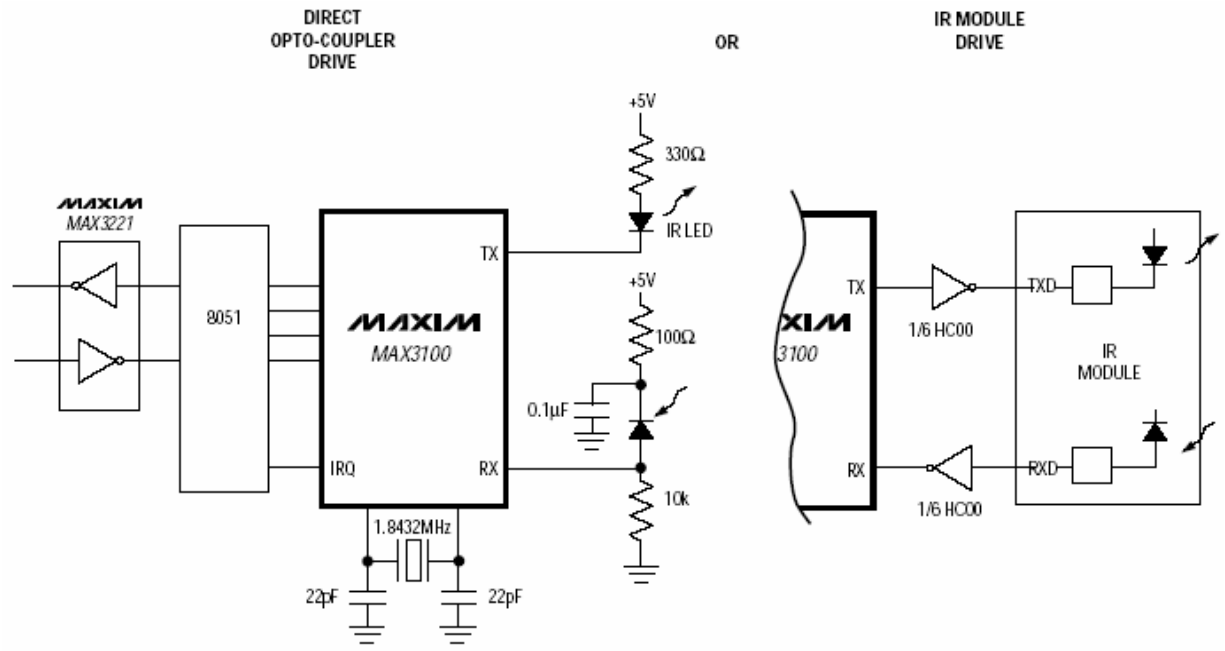


Figura 4.2 RS242 IrDA bidirecional com o 8051

O Diagrama ilustrado na Figura 4.3 abaixo, exemplifica ao funcionamento do MAX3100 em bloco:

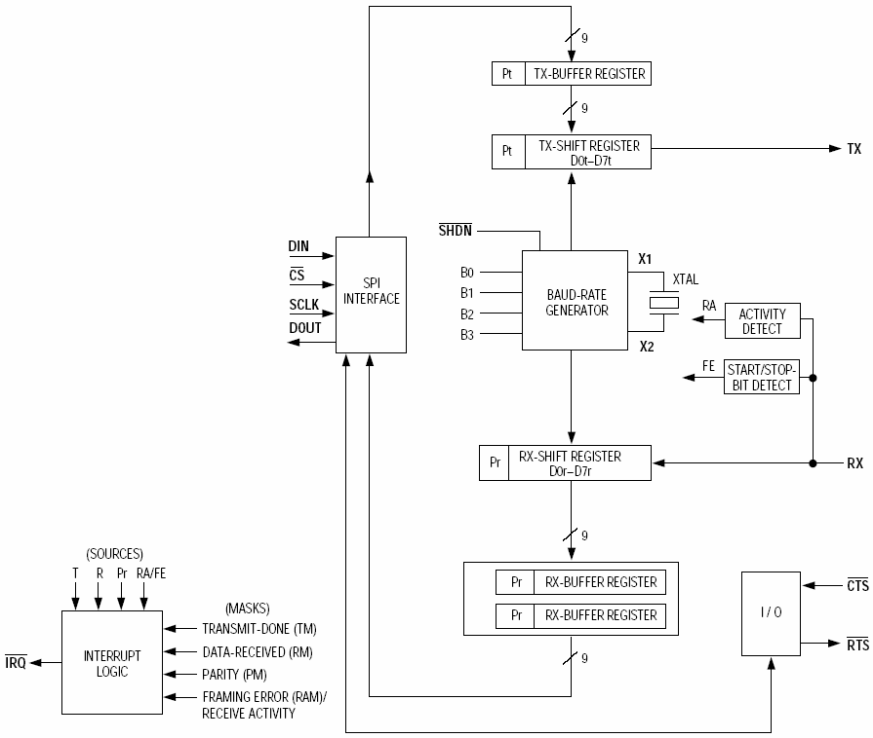


Figura 4.3 Diagrama Funcional MAX3100

4.2.1. Implementação usando o MAX3100

O circuito integrado MAX3100 foi usado para estabelecer um *link* serial infravermelho (SIR) usando a modulação de pulsos proposta pela IrDA. Para tanto, criou-se um circuito eletrônico que habilitava um PC, usando o sistema operacional DOS para acessar os recursos da porta paralela, a controlar o MAX3100; na literatura de comunicações infravermelhas é comum designar tal circuito como *dongle*. Criou-se um *dongle* também para o ambiente de desenvolvimento para o microcontrolador 8051.

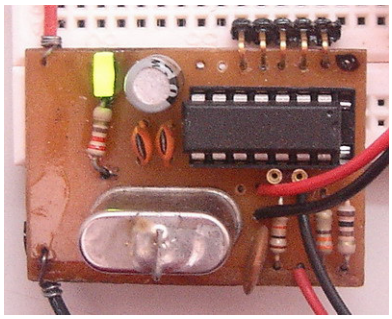


Figura 4.4 Foto dos dongles

A Figura 4.4 acima apresenta o circuito sintetizado para a implementação do *dongle* usando o MAX3100, apêndice A, Código 01 – Comunicação do MAX3100 com o PC encontra-se o código para controle do MAX3100 via PC (max3100pc.cpp) e também no apêndice A temos o Código 02 – Comunicação do MAX3100 com o Microcontrolador 8051 para controle do MAX3100 via 8051 (max31008051.asm).

5. CONTROLADOR DE PILHA DE PROTOCOLOS IRDA - MCP2150

O MCP 2150 é um dispositivo que implementa conectividade sem fio no padrão IrDA. O MCP2150 dá suporte para a pilha de protocolos padrão do IrDA e bit de codificação / decodificação.

A taxa de símbolos (*baud rate*) da interface serial é usado para selecionar uma das quatro taxas de símbolos do padrão IrDA entre 9600 *bauds* a 115,2k *bauds* (9600, 19200, 57600, 115200). A Taxa de símbolos IR é especificada pelo usuário entre uma destas cinco: 9600, 19200, 37400, 57600 ou 115200. A taxa de símbolos serial é especificadas pelos pinos *BAUD*0 e *BAUD*1 ilustrada na Tabela 5.1, enquanto o *baud rate* IR é especificado pelo dispositivo primário (durante a fase descoberta da conexão) isso significa que o *baud rate* não precisa ser o mesmo. O MCP opera como um equipamento terminal de aplicações e se situa entre a UART e um transceptor óptico infravermelho. O relógio (*clock*) do MCP2150 é gerado pela ponte ressonante ilustrada na Figura 5.1 baixo [3] e [12]:

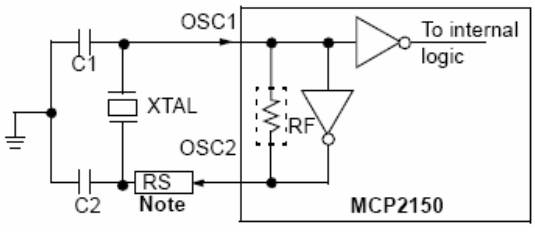


Figura 5.1 Oscilador do MCP2150

BAUD1:BAUD0	Taxa de Símbolo (*Baud Rate*) [11,0592MHz]	Taxa de *Bit* (*Bit Rate*)
00	9600	F_{osc} / 1152
01	19200	F_{osc} / 576
10	57600	F_{osc} / 192
11	115200	F_{osc} / 96

Tabela 5.1 Taxa de transmissão Serial (Seleção vs. Freqüência)

O MCP2150 codifica assincronamente um fluxo serial de dados, convertendo cada bit de dados para seu correspondente em pulso no formato infravermelho (IR). Os pulsos

infravermelhos recebidos são então processados pela máquina de estados que controla os protocolos. O controlador de protocolos envia os bytes de dados apropriados para o *host* de controle da UART no formato serial. A Figura 5.2 abaixo, ilustra a dinâmica de funcionamento do MCP2150:

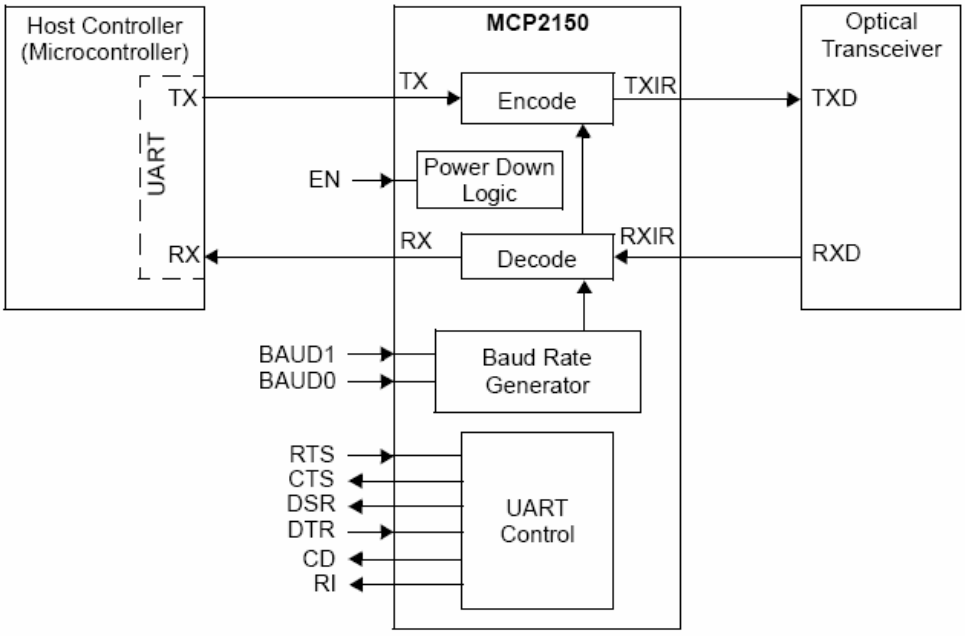


Figura 5.2 Diagrama de Blocos de funcionamento do MCP2150

A seguir temos a Figura 5.3 que mostra a pinagem do MCP2150 e o tipo de encapsulamento (DIP) [3] e [12]:

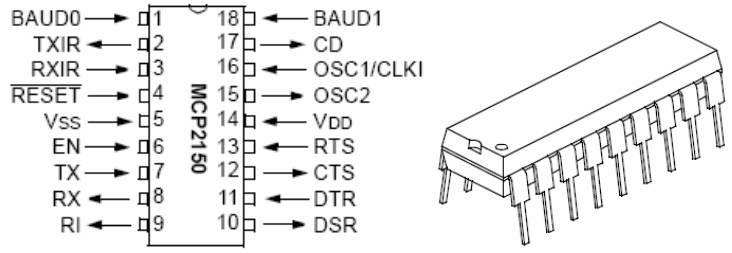


Figura 5.3 Pinagem do MCP2150 (PDIP, SOIC)

Na Tabela 5.2 a seguir temos um breve comentário da função de cada pino deste circuito integrado :

Nº do Pino	Sinal	Direção	Comentário
1	CD	MCP2150→HC	Portadora detectada
2	RX	MCP2150→HC	Recebe o Dado
3	TX	HC → MCP2150	Transmite o Dado
4	DTR	-	
5	GND	-	Terra
6	DSR	MCP2150→HC	Conjunto de Dados Pronto
7	RTS	HC → MCP2150	Pedido para transmitir
8	CTS	MCP2150→HC	Pronto para Transmitir
9	RI(1)	-	Indicador de Sinal
Legenda: HC : Hóspede Controlador **Nota 1: O sinal não é implementado no MCP2150**			

Tabela 5.2 Direção dos Sinais de MCP2150

As Figura 5.4 e Figura 5.5 a seguir ilustra como é feita a codificação e a decodifação do bit para ser transmitido via IR no MCP2150.

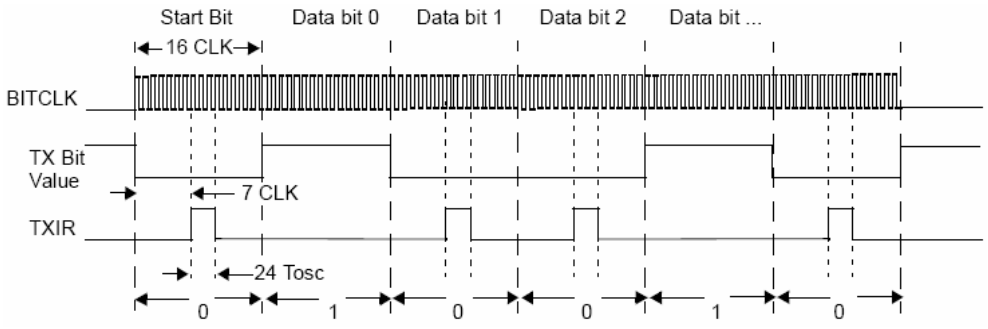


Figura 5.4 Codificação infravermelho do MCP2150.

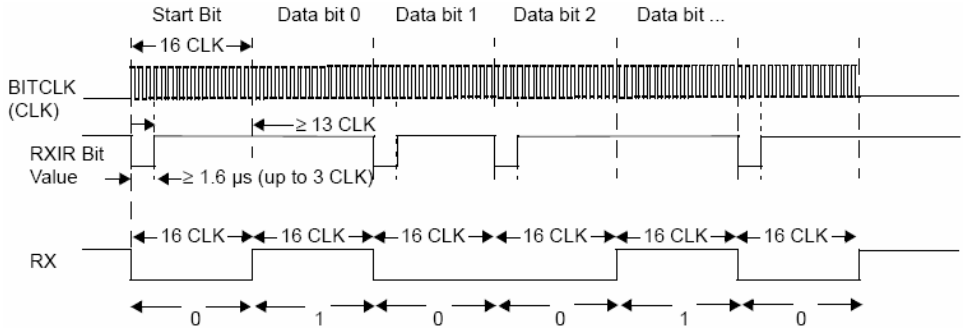
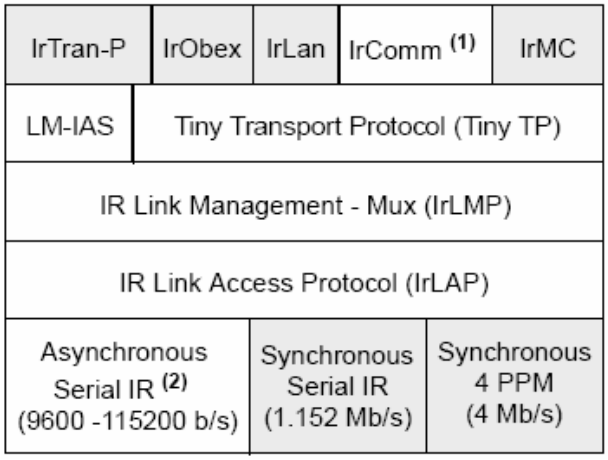


Figura 5.5 Decodificação infravermelho do MCP2150.

A Figura 5.6 abaixo ilustra quais os protocolos do padrão IrDA são implementados pelo MCP2150, sendo que o protocolo IrCOMM não implementa todas as classes:



Legenda:

- Suportado pelo MCP2150
- Protocolos IrDA opcional não suportados pelo MCP2150

Figura 5.6 Pilha de Protocolos de dados do padrão IrDA

A Figura 5.7 ilustra quais as classes do protocolo IrCOMM são implementados pelo MCP2150:

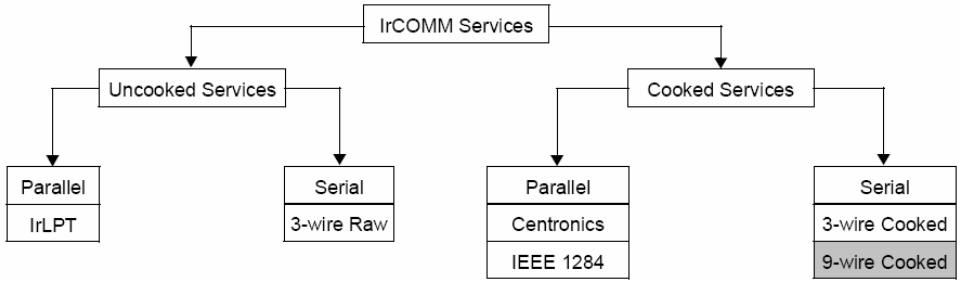


Figura 5.7 Classes suportadas pelo IrCOMM no MCP2150

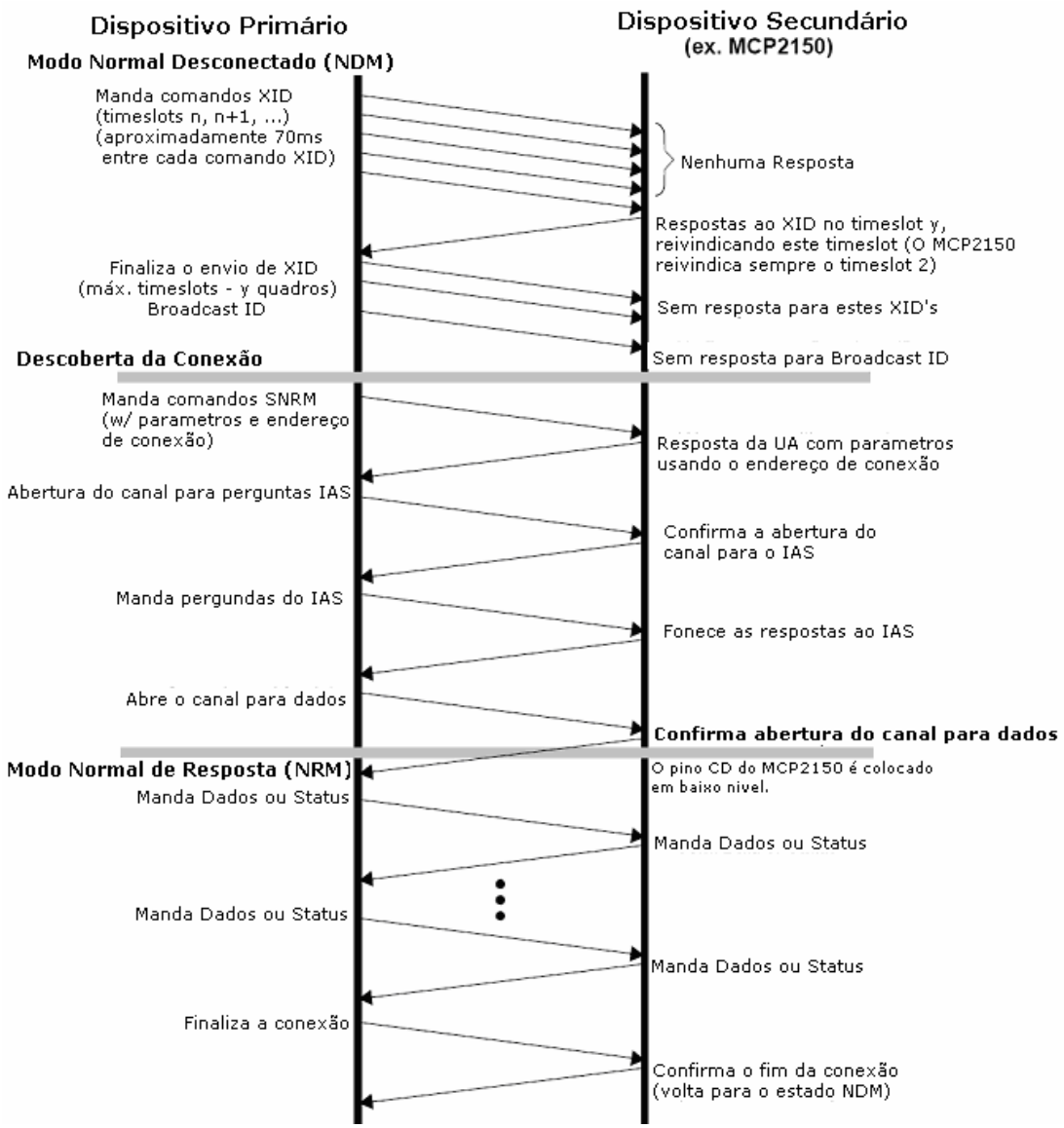


Figura 5.8 Seqüência de Conexão

5.1. TRANSDUTOR SERIAL INFRAVERMELHO (*SERIAL INFRARED TRANSCEIVER SIR*)

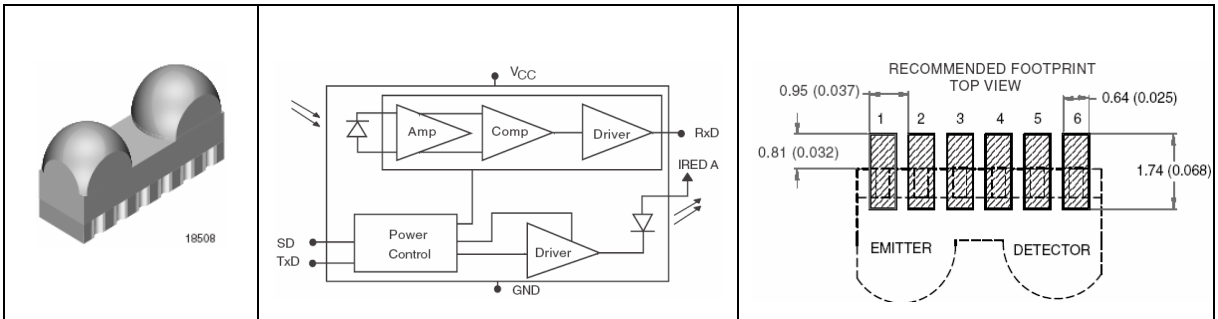
Como ilustrado na Figura 5.2 é necessário a utilização de um bom transceiver de modo a se obter um melhor ganho e qualidade no sinal, possibilitando assim uma conexão mais segura a uma distância maior e a uma taxa de transmissão maior. O transceiver escolhido para este projeto foi o TFBS4711. O TFBS4711 tem um perfil de módulo transdutor de dados infravermelho. Ele suporta taxas de dados em IrDA acima de 115,2 kbit/s (SIR). O Módulo transdutor consiste de um fotodiodo PIN, um emissor de infravermelho (IRED) e um circuito

integrado controlador de baixa potência CMOS que fornece uma solução adiantada em um único pacote.

O dispositivo é projetado para trabalhar em baixa potência no padrão IrDA com um distância axial de até 1m. O comprimento do pulso de R_{xd} é independente da duração do pulso de T_{xd} e sempre está ajustado para um valor ótimo encontrado pelo dispositivo para todos os padrões SIR codificador / decodificador e interfaces. A característica da parada programada (*ShutDown* - SD) corta o consumo atual para tipicamente 10 nA [5] e [15].

Características:

- ✓ Encapsulamento reduzido H 1.9 mm x D 3.1 mm x L 6.0 mm
- ✓ Compatível com a última especificação de baixa potência da camada física IrDA (9.6 kbit/s to 115.2 kbit/s).
- ✓ Distância típica de comunicação em visada de até 1m.
- ✓ Bateria e Características de gerenciamento de energia:
  - Corrente ociosa - 75 µA típica
  - Corrente de Parada – 10 nA típica
  - Tensão de operação de 2,4 a 5,5V dentro da especificação da escala de temperatura de -25ºC a +85ºC.
- ✓ Controle Remoto – distância de transmissão de até 8m.



(a) Encapsulamento do transceiver	*(b) diagrama funcional*	*(c) dimensões em mm (inches)*

Figura 5.9 Encapsulamento e diagrama funcional do transceiver.

O circuito implementado para a utilização do TFBS4711 está ilustrado na Figura 5.10 a seguir e os valores dos componentes mais indicados para o circuito estão na Tabela 5.3 [5] e [15]:

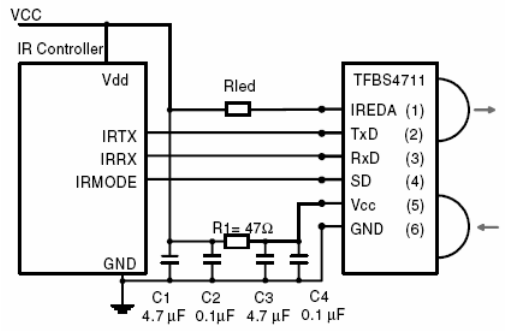


Figura 5.10 Circuito de aplicação recomendado

Componente	Valor Recomendado
C1, C3	4,7µF, 16V
C2, C4	0,1µF, Cerâmico
R1	47Ω, 0.125W

Tabela 5.3 Componetes de Aplicação do Circuito de Aplicação

Nas Tabela 5.4 e Tabela 5.5 abaixo temos alguns parâmetros importantes de operação do transceiver:

Parâmetro	**Condições de Teste**	**Símbolo**	**Mín.**	**Típ.**	**Máx.**	**Unidade**
Radiação angular mínima E_e [(1)] em escala angular	9,6kbits/s a 115,2kbits/s λ = 850 nm – 900nm α = 0º, 15º	E_e		35 (3.5)	80 (8)	mW/m^2 ($\mu W/cm^2$)
Radiação angular máxima E_e em escala angular	λ = 850 nm – 900nm	E_e		5 (500)		kW/m^2 (mW/cm^2)
Tempo de subida do sinal de saída	10% a 90%, CL = 15pF	$t_{r(Rxd)}$	10		100	ns
Tempo de descida do sinal de saída	90% a 10%, CL = 15pF	$t_{f(Rxd)}$	10			
Largura do pulso Rxd	Largura do pulso de entrada > 1,2µs	t_{PW}	1.7	2.0	3.0	µs
Jitter de borda principal	Radiação de entrada = $100mW/m^2$, ≤ 115,2kbit/s				250	ns
Standby / tempo de parada, tempo de Startup do receptor	Depois desligado ou ligado				150	µs
Latência		t_L			150	µs
[(1)] Definições de sensibilidade IrDA: Radiação angular mínima E_e, potencia por unidade de área. O receptor deve manter a a BER especificada enquanto a fonte opera em intensidade mínima na escala angular até o mínima escala de meio ângulo até o máximo comprimento do link. T_{amb} = 25ºC, Vcc = 2.4V a 5.5V						

Tabela 5.4 Parâmetros de recepção

Parâmetro	Condições de Teste	Símbolo	Mín.	Típ.	Máx.	Unidade
Corrente de operação IRED	T_{amb} = -25ºC a 85ºC	I_D	200	300	400	mA
Pico de operação do Transceiver	Durante a operação de pulso de IRED	I_{CC}		0.57		mA
Corrente de fuga IRED	Txd = 0V, 0 < V_{CC} < 5.5V	I_{IRED}	-1		1	µA
Intensidade da radiação de saída	α = 0º, Txd = High, SD = Low	I_e	tbd	60		mW/sr
	α = 0º, 15º Txd = High, SD =	I_e	tbd	35		mW/sr
	V_{CC} = 5.0V, α = 0º, 15º, Txd =	I_e			0.04	mW/sr
Ângulo da intensidade de radiação de saída		α		±22		-
Comprimento de onda do pico de emissão		λ_P	880		900	nm
Largura de banca espectral		Δλ		45		nm
Tempo de subida óptico		t_{opt}	10		100	ns
Tempo de decida óptico		t_{opt}	10		100	ns
Duração do pulso de saída óptico	Pulso de entrada com largura de 1.63 µs	t_{opt}	1.41	1.63	2.23	µs
	Pulso de entrada com largura de t_{Txd} < 20 µs	t_{opt}	t_{Txd}		t_{Txd} + 0.15	µs
	Pulso de entrada com largura de $t_{Txd} \geq$ 20 µs	t_{opt}			300	µs
Overshoot óptico					25	%

Tabela 5.5 Parâmetros de transmissão

5.2. TRANSCEIVER IRDA PARA PC:

O IrDA é um padrão tão comum que já implementado em diversas Placas Mãe nos PC. O não é usual a utização do desta porta de comunicação por parte dos usuários de PC's, tal vez por desconhecimento deste recurso. Afim de utilizarmos o MCP2150 com outro disposistivo de maior acesso (neste caso o PC) que implemente o padrão IrDA temos aqui uma sugestão simples para utilizar a porta serial infravermelha do PC.

A maior parte da lógica de transcepção em IR ou infravermelho fica a cargo da placa-mãe, anexando-se a esta somente a parte da transdução, ou sejam, os módulos mínimos que cuidam da recepção e da transmissão de dados. A Figura 5.11 exemplifica um circuito simples que pode ser montado para um PC:

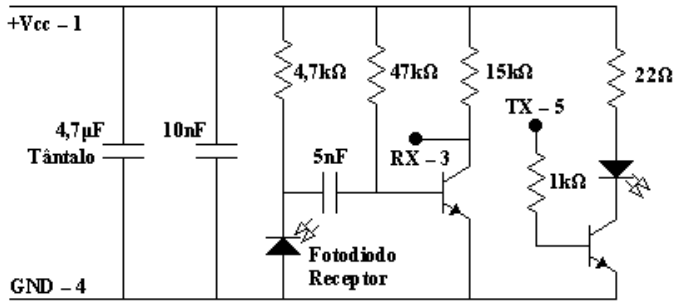


Figura 5.11 Circuito implementado para o PC

O desacoplamento da fonte e do *Ripple* (ruído) desta é fundamental em qualquer circuito e o PC tem uma fonte ruidosa, do ponto de vista eletrônico, justificando-se assim o uso desses dois capacitores.

É fundamental, num circuito que vai trabalhar com luz como meio, boa isolação luminosa dos transdutores (diodo emissor e fotodiodo, receptor). Consegue-se isto com um casulo para led - na falta deste, à guisa de improviso. Coloca-se esta numa posição que a luz só saia pelo orifício de ajuste de freqüência, no topo da caneca. Na falta deste componente, sugere-se colocar uma fita isolante sobre os dois diodos, não encobrindo, claro, o facho de luz na frente dos diodos. Com estes cuidados, o aparelho só vai cuidar da luz que interessa, ou seja, infravermelho, aumentando bastante o seu desempenho.

Convém ressaltar que existem dois tipos de conector IrDA em placas-mãe de fabricantes mais comuns, sendo o modelo da "Asus" quase um padrão, ou seja, vários outros fabricantes o adotam; o outro é a "PCChips", mais específico deste fabricante. Abaixo, vemos os dois modelos de conectores na Tabela 5.6.

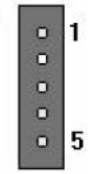
Pinagem Asus e Outros	**Pinagem PCChips**
1 - Power (VCC) 2 - Não Conectado 3 - IRRX 4 - Gnd 5 - IRTX 	*IR1 Header* 

Tabela 5.6 Pinagem do adaptador IrDA em placas mãe.

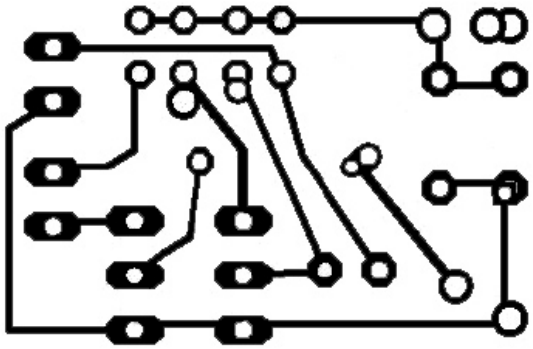


Figura 5.12 Placa de Circuito Impresso do circuito da Figura 5.11

5.2.1. Utilização do Circuito

É necessário que se configurem, no BIOS da Placa-Mãe, os parâmetros de operação do IRDA. Normalmente, é necessário somente habilitá-lo. Coloque o IRDA obrigatoriamente como IRDA, SIR ou HPSir. Não o coloque como **ASKIR**, pois isto poderá danificar irremediavelmente o circuito, pois este foi projetado para trabalhar com Sir ou "Serial Infra Red". O ASKIR é um protocolo que usa uma sinalização contínua, daí este circuito não ser projetado para tal.

No Windows, em qualquer versão, a partir do 9x, o sistema reconheceu automaticamente o novo circuito, depois de habilitado no BIOS.

6. INTERNET EMBARCADA

O iinChip W3100A é um LSI, circuito integrado para integração em larga escala, que implementa em hardware uma pilha de protocolos proporcionando solução fácil e barata para conectividade para Internet de alta velocidade a ser aplicada em dispositivos digitais permitindo simples instalação da pilha de protocolos TCP/IP [6] e [16].

Esse integrado oferece a desenvolvedores uma maneira rápida e fácil de acrescentar as facilidades da interconexão em redes a qualquer sistema digital. Adicionar esse LSI em um sistema disponibiliza acesso à Internet para esse sistema e libera-o de eventuais processamentos necessários para o gerenciamento das conexões e especialmente do próprio protocolo [6] e [16]

O W3100A contém a Pilha de Protocolos do TCP/IP como o TCP, UDP, IP, ARP e o ICMP bem como protocolos Ethernet como o *Data Link Control* e o protocolo MAC. Uma da maiores vantagens na utilização desse integrado está no fato dele oferecer uma API (*Application Programming Interface*) para *sockets* que é similar à API de *sockets* do Windows, além de oferecer interface aos barramentos usados pelos microcontroladores Intel e Motorola (8051, i386, 6811 já testados) e ao barramento I2C.

O W3100A permite o interfaceamento de sistemas simples com a Internet. Aplicações possíveis são inúmeras.

- ✓ Rede Caseira: TV digital, DVD player, Vídeo Games podem ter suas capacidades aumentadas por meio do W3100A atuando como canal de retorno.
- ✓ Automação de Plantas: Controlador Remoto, Sistema de aquisição de dados, Medidores Remotos, Sistema de controle para ar-condicionado e ventilação, Sistema de controle de temperatura e umidade, Sistema de Gerenciamento de Rede, enfim, monitoramento de todo e qualquer tipo de processo pode ser beneficiado por meio do uso da Internet.
- ✓ Aplicações Médicas: Sistema de Diagnóstico Remoto e Sistema de Monitoramento de Paciente serão dinamizados pelo uso do W3100A .

- ✓ Distribuição Automática: Leitores de Códigos de barra, Scanner e Impressora de Sistema podem prescindir de contato humano no transporte de informações por meio da construção de sistemas preparados para a conectividade da Internet.

- ✓ Segurança de Edificações: Câmera em rede, DVR em rede, Sistema de Controle de Acesso, Sistema de Identificação de Impressões Digitais, Sistema de Segurança com Sensores RF podem usar a infra-estrutura lógica para monitoramento dos ambientes.

Suas características principais são as seguintes:

- ✓ Protocolos implementados em hardware: TCP, IPv4, UDP, ICMP, ARP.
- ✓ Protocolos Ethernet: DLC, MAC.
- ✓ Suporta quatro conexões independentes simultâneas.
- ✓ O protocolo ICMP embutido responde a comandos PING.
- ✓ Velocidade de processamento do protocolo: 20 Mbps full-duplex.
- ✓ Interface ao barramento usados pelos microcontroladores Intel/Motorola.
- ✓ Interface I2C.
- ✓ API para *socket* proporciona fácil programação de aplicações.
- ✓ SRAM interna de dupla porta de 16 Kbytes para buffer de dados.
- ✓ Tecnologia CMOS de 0.35 um.
- ✓ Larga faixa de operação em tensões: 3,3 V de operação interna mas é tolerante a IO (entrada e saída) em 5 V.

6.1. INICIALIZAÇÃO DO W3100A

Para usar o W3100A, registradores básicos necessários ao funcionamento mínimo do W3100A devem ser ajustados. Os registradores básicos incluem o GAR (Registrador de Endereço de Gateway), SMR (Registrador de Submáscara de Rede), SHAR (Registrador de Endereço de Hardware de Origem) e SIPR (Registrador de Endereço de IP de Origem).

GAR, SMR e SIPR são as informações de rede sobre as quais o W3100A é operado e esses registradores devem ser ajustados de acordo com o ambiente de operação. SHAR é o

endereço de hardware usado na camada MAC do W3100A e o endereço disponibilizado pelo fabricante é utilizado.

Depois de ajustar apropriadamente os registradores mencionados, W3100A pode ser ativado na rede pela execução do comando sys_init. A ativação pode ser verificada por meio de um comando PING (ICMP *Echo request*).

6.2. PROTOCOLO TCP

TCP é um protocolo orientado a conexão. Usando um método de *handshaking* de três etapas no ajuste da conexão e no processo de conclusão é possível termos transmissão de dados confiáveis e a recepção é garantida.

6.2.1. Processo de Inicialização de TCP

Para usar o TCP do W3100A, o campo de protocolo do Cx_SOPR do canal desejado (onde x é o número do canal) precisa ser ajustado como SOCK_STREAM(0x01). Depois de o canal ser ativado pelo comando sock_init, Cx_TW_PR (Registrador Ponteiro de Escrita TX do Canal x), Cx_TR_PR (Registrador Ponteiro de Leitura do Canal x) e Cx_TA_PR (Registrador Ponteiro de *Acknoledge* TX) precisam ser inicializados com o mesmo valor.

6.2.2. Processo de Ajuste da Conexão TCP

No W3100A, o processo de conexão suscitado pelos comandos *Connect* ou *Listen* é gerenciado internamente. O envio de um pacote SYN resultante de um comando *Connect* é chamado de abertura ativa de conexão enquanto que a espera por um pacote SYN externo resultante do comando *Listen* é chamado de abertura passiva de conexão.

6.2.3. Abertura Ativa

O modo de cliente TCP conhece o endereço IP e o número da porta de destino, e o ajuste da conexão é continuado.

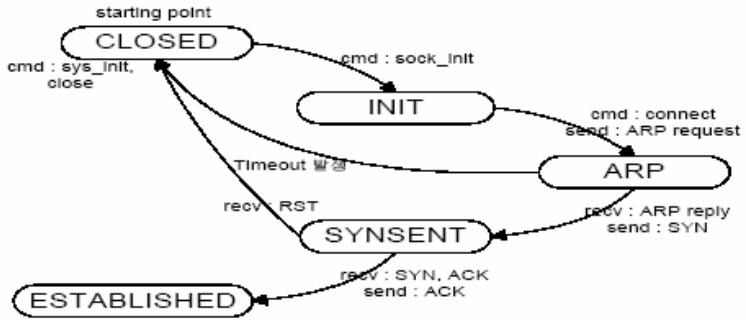


Figura 6.1 Diagrama da abertura ativa da comunicação TCP

O diagrama acima ilustra o processo de ajuste de conexão usando abertura ativa. Cada estado pode ser verificado através do Registrador de Estado de Socket do canal correspondente.

a) Estado CLOSED: o canal é inicializado pela execução do comando sys_init ou do comando close.

b) Estado INIT: ajusta o número da porta (Registrador de Porta de Origem) a ser usada no canal e ativa o canal pela execução do comando sock_init.

c) Estado ARP: de maneira a ajustar a conexão, MCU (Unidade Microcontroladora) ajusta o IP de destino, a porta de destino por meio da escrita nos devidos registradores e executa o comando *conect*. Baseado nesse comando, W3100A muda para esse estado e transmite o pacote de requisição ARP. Quando o pacote ARP de resposta é recebido da outra extremidade da conexão, é efetuada uma retransmissão. Quando nenhuma resposta é recebida (dentro da duração do *timeout* ajustada), ocorre *timeout* e isso leva-nos ao estado CLOSED.

d) Estado SYNSENT: nesse estado, W3100A transmite um pacote SYN e espera receber um pacote SYN, ACK da outra extremidade da conexão. No caso de recepção apropriada do pacote SYN, ACK,

e) W3100A transmite um pacote ACK e completa o ajuste da conexão e muda o estado para ESTABLISHED. No caso de não recebermos o pacote SYN,ACK apropriado da outra extremidade da conexão é feita a retransmissão de um pacote SYN. Quando nenhuma resposta é recebida dentro de uma duração de "timeout" designada, ocorre timeout e muda o estado do sistema para CLOSED. Além disso, se a outra extremidade da conexão não possuir nenhuma aplicação esperando no modo passivo, a

extremidade da conexão recebe um pacote RST e sistema muda seu estado para CLOSED.

6.2.4. Abertura Passiva

No modo Servidor TCP, o sistema espera pelo estabelecimento de conexão do lado cliente por meio do comando *Listen* e o início da conexão é aceito quando requisitado.

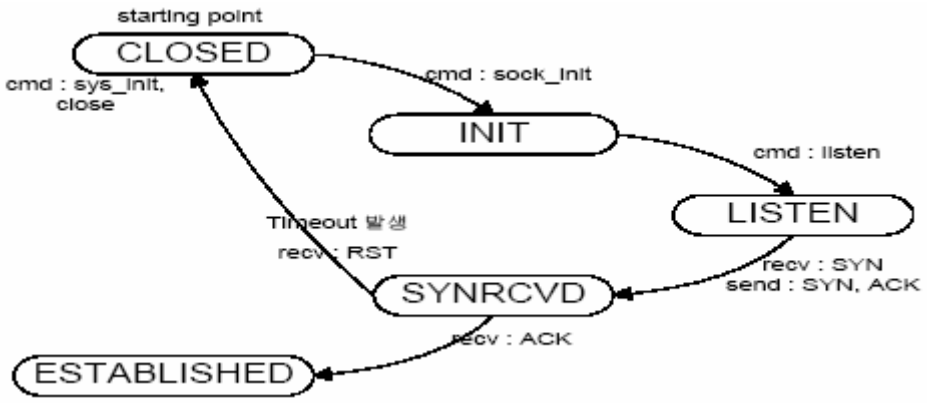


Figura 6.2 Diagrama da abertura passiva da comunicação TCP

O diagrama acima ilustra o processo de ajuste de conexão usando abertura ativa. Cada estado pode ser verificado através do Registrador de Estado de *Socket* do canal correspondente.

a. Estado CLOSED: o canal é inicializado pela execução do comando sys_init ou do comando close.

b. Estado INIT: ajusta o número da porta (Registrador de Porta de Origem) a ser usada no canal e ativa o canal pela execução do comando sock_init.

c. Estado LISTEN: espera por uma conexão do cliente. Quando um pacote SYN da porta correspondente é recebido pelo outro lado da conexão, um pacote SYN, ACK é transmitido, mudando o estado para SYNRCVD.

d. Estado SYNRCVD: um pacote SYN, ACK é transmitido e espera por um ACK do outro lado da conexão. Quando a resposta do outro lado é recebida, isso

muda o estado para ESTABLISHED e, quando nenhuma resposta é recebida, um pacote SYN, ACK é retransmitido e o sistema muda seu estado para CLOSED depois da ocorrência de *timeout* out da recepção de um pacote RST.

6.2.5. Processo de Fechamento de uma conexão TCP

De maneira semelhante ao processo de início de uma conexão TCP, o processo de fechamento também usa um método de *handshake* de 3 etapas.

O envio de FIN depois de receber um comando Close da aplicação é chamado de fechamento ativo e fechar a conexão após receber um FIN da outra extremidade da conexão é chamado de fechamento passivo.

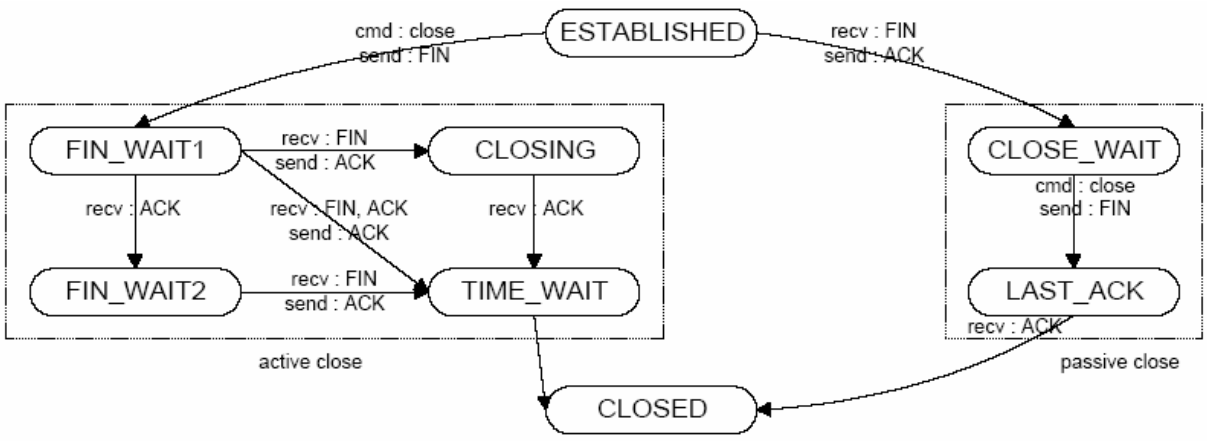


Figura 6.3 Fechamento de Conexão

6.2.6. Fechamento Ativo

Depois de terminar a transmissão e a recepção dos dados, a aplicação usa o comando Close para finalizar a conexão. Quando a conexão é terminada após um comando Close enquanto o sistema está no estado ESTABLISHED, temos um fechamento ativo e o processo é ilustrado no lado esquerdo do diagrama acima,

Estado FIN_WAIT1: proveniente do estado ESTABLISHED, o sistema muda o estado de conexão para esse após a recepção do comando Close e transmite um pacote FIN. Ele muda para o estado FIN_WAIT2 quando um ACK for enviado do outro extremo da conexão em resposta ao FIN. O sistema transmite um ACK e muda para o estado CLOSING quando um pacote FIN é recebido do outro lado da conexão. Depois, o sistema transmite um ACK e muda o estado para TIME_WAIT quando um pacote FIN,ACK for recebido. Em caso de

nenhuma resposta, uma retransmissão é feita e se nenhuma resposta for recebida até a ocorrência de um timeout, o sistema muda para o estado CLOSED.

Estado FIN_WAIT2: espera por um pacote FIN do outro lado da conexão. Nesse estado o W31000A não recebe dados do outro lado da conexão e se algum dado for recebido, a inicialização da conexão é imediatamente abortada por meio de um pacote RST. Isso é justificado pelo fato de W3100A não processar dados adicionais durante o fechamento de uma conexão.

Estado CLOSING: resultado do fechamento simultâneo da aplicação. Muda para o estado TIME_WAIT quando o ACK for recebido do outro lado da conexão.

Estado TIME_WAIT: visto como um estado 2MSL (*Maximum Segment Lifetime*) WAIT pelo TCP. No caso de um pacote FIN ser reenviado porque o outro lado da conexão não pôde receber ACK, há uma função onde o TCP reenvia o último ACK. No caso em que a conexão estiver em um estado de espera 2MSL, há uma outra função onde outro cliente ou servidor está bloqueado para o uso dessa conexão. O W3100A, devido às restrições de recursos e visando o uso eficiente do canal, muda desse estado para CLOSED sem espera.

6.2.7. Fechamento Passivo

No fechamento passivo, um pacote FIN é recebido proveniente do outro lado da conexão para encerrar uma conexão no estado ESTABLISHED conforme ilustrado no lado direito do DIAGRAMA.

Estado CLOSE_WAIT: provindo do estado ESTABLISHED, o sistema muda para esse estado depois de receber um pacote FIN do outro lado da conexão. O sistema transmite um ACK para o FIN e gera uma interrupção no MCU. Pelo atendimento da interrupção, o MCU executa o comando Close para o W3100A e completa a finalização da conexão. Mas, se ainda houver dados a serem enviados, o valor TW_PR que não é igual a TA_PR, portanto não devemos solicitar ao sistema um comando Close mas esperar até que um timeout ocorra ou até mesmo ignorar o procedimento de fechamento e avançar nos processos de comunicações por meio de um comando Sock_init, por exemplo.

Estado LAST_ACK: quando um comando Close é enviado pelo MCU, um pacote FIN é transmitido e espera por um ACK. Se nenhum pacote ACK for recebido, o pacote FIN será

retransmitido. Se nenhuma resposta é recebida até a ocorrência de um timeout, o sistema muda para o estado CLOSED.

6.2.8. Transmissão e Recepção de Dados TCP

Diferentemente do protocolo UDP, a transmissão e recepção de dados TCP somente é possível após o estabelecimento de uma conexão. O W3100A possui memórias exclusivas para transmissão e recepção, 8KB para transmissão e 8KB para recepção. Essa memória pode ser ajustada para 1KB, 2KB, 4KB e 8KB usando RMSR (Rx data Memory Size Register - registrador de tamanho de memória de recepção) e TMSR (Tx data Memory Size Register - registrador de tamanho de memória de transmissão).

6.2.9. Ajuste do tamanho da memória de transmissão

A memória de transmissão do W3100A possui capacidade de armazenamento de 8KB, e uma fração desse tamanho pode ser ajustado para cada canal por meio do registrador TMSR. Um exemplo de TMSR ajustando o tamanho de cada memória é ilustrado no diagrama abaixo

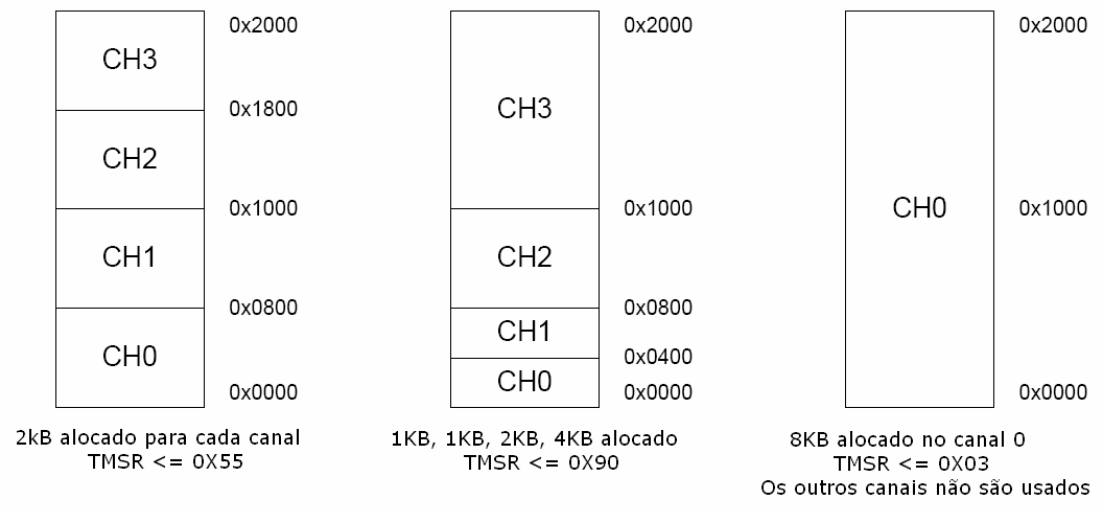


Figura 6.4 Alocação de Memória na Transmissão

Quando o tamanho da memória do canal 0 exceder 8KB, toda chamada à memória é ignorada.

6.2.10. Processo de Transmissão de Dados TCP

Para proceder à recepção dos dados TCP, o W3100A possui um ponteiro de 4 bytes chamado Cx_TW_PR (Tx *Write Pointer Register of Channel* x - registrador ponteiro de escrita para transmissão do canal x) e outro ponteiro de 4 bytes chamado Cx_TA_PR (Tx *Ack Pointer Register of Channel* x - registrador ponteiro Ack para transmissão do canal x). Cx_TW_PR é o ponteiro que escreve dados para serem transmitidos pelo MCU e Cx_TA_PR é o ponteiro que completa a transmissão pelo W3100A. Cx_TW_PR e Cx_TA_PR tornam-se iguais depois do estabelecimento de uma conexão. Numa abertura ativa, eles são inicializados igualmente após o comando Sock_init solicitado pelo MCU. Na abertura passiva, um é inicializado pelo outro. A diferença entre os ponteiros resulta no espaço livre em buffer, FBS (free buffer size). Os dados são armazenados a partir do Cx_TW_PR de acordo com tal tamanho e quando a escrita de dados é terminada, Cx_TW_PR é mudado de acordo com o tamanho dos dados armazenados e executa o comando Send.

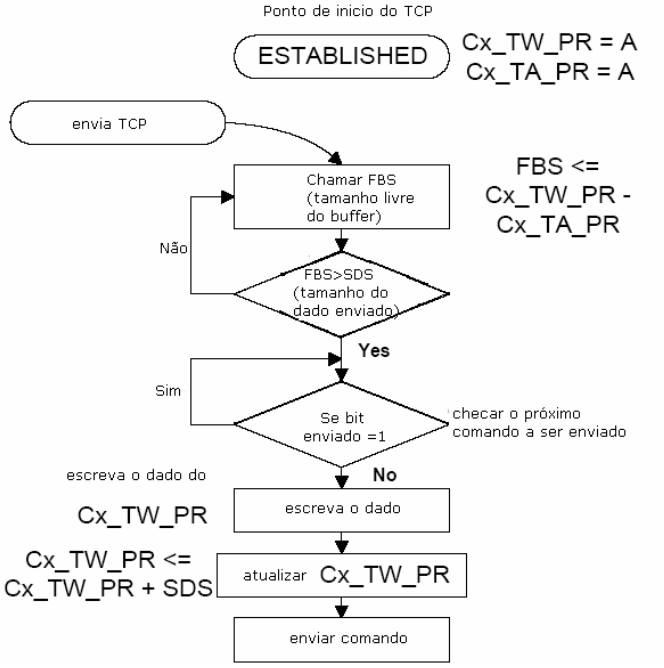


Figura 6.5 Processo de Transmissão de Dados TCP

O diagrama acima ilustra a mudança no Cx_TW_PR e no Cx_TA_PR quando a transmissão de dados é realizada com o sistema ajustado para memória de transmissão de 2KB no canal 0.

6.2.11. Ajuste do tamanho da memória de recepção

A memória de recepção do W3100A tem a mesma estrutura da memória de transmissão e opera da mesma maneira.

A memória tem capacidade de 8KB ao total, e sua alocação por canal pode ser efetuada através de RMSR (Rx data *Memory Size register* - registrados de tamanho de memória de recepção). Um exemplo de RMSR e o tamanho de cada memória é ilustrado no diagrama abaixo.

Quando o tamanho da memória do canal 0 excede 8KB, toda chamada à memória é ignorada.

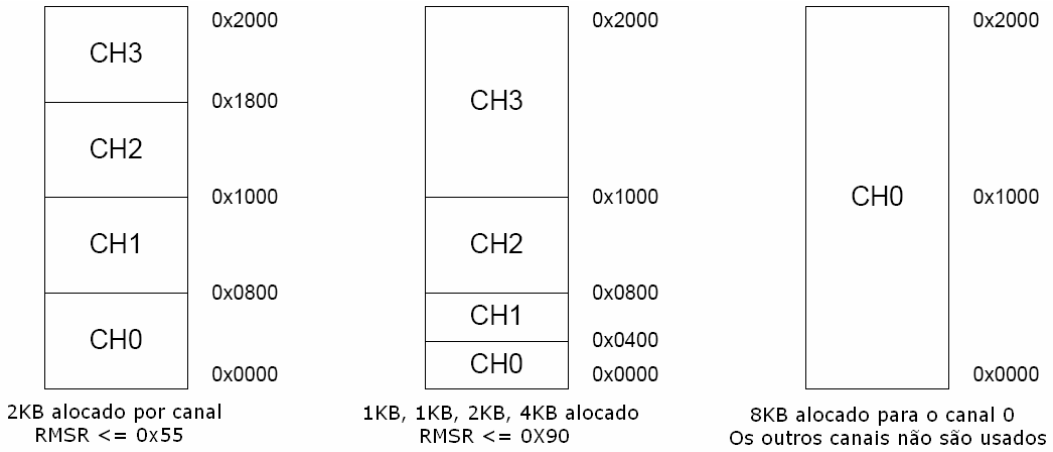


Figura 6.6 Alocação de memória de recepção

Processo de recepção de dados TCP

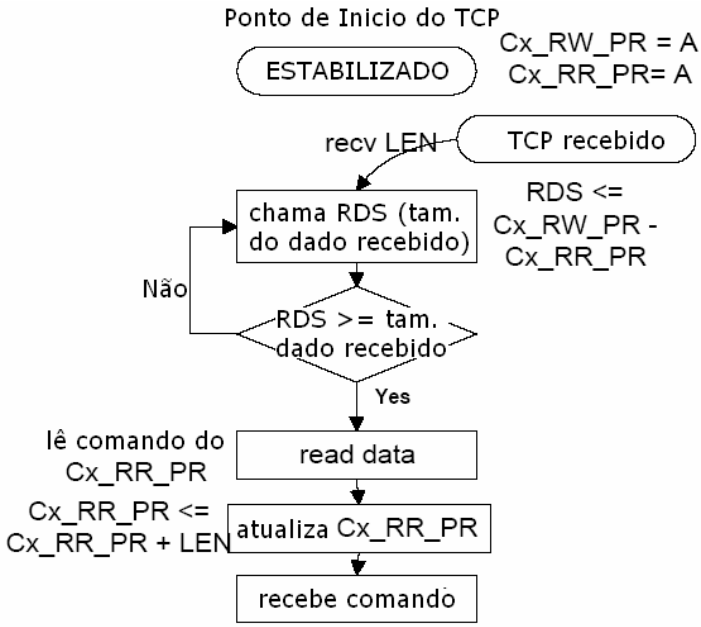


Figura 6.7 Recepção de dados TCP

A recepção de dados TCP pelo W3100A é ilustrado no diagrama acima. No W3100A, quando o dado é recebido da rede, é armazenado na memória de recepção a partir do Cx_RW_PR (Rx *Write Pointer Register of Channel* x - registrador ponteiro de recepção de escrita do canal x), Cx_RW_PR é aumentado de acordo com o tamanho dos dados recebidos quando a recepção é terminada e, então, o MCU é interrompido de maneira a indicar a recepção dos dados. Através da interrupção ou do polling, o MCU compara Cx_RW_PR e Cx_RR_PR (Rx *Read Pointer Register of Channel* x - registrador ponteiro de recepção de leitura do canal x) e quando a recepção de dados é observada, o tamanho dos dados recebidos é calculado e Cx_RR_PR é aumentado depois que os dados forem lidos e processados por Cx_RR_PR. Finalmente, o comando Recv é executado para indicar ao W3100A que o processamento dos dados recebidos já foi finalizado.

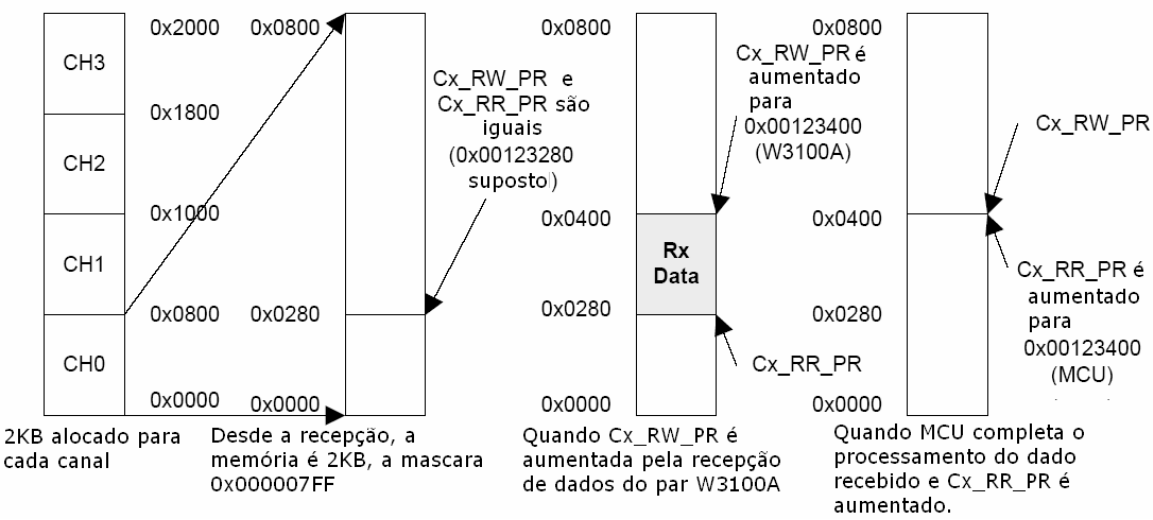


Figura 6.8 Ponteiro de gerenciamento durante a recepção TCP

O diagrama acima ilustra a mudança no Cx_RW_PR e no Cx_RR_PR quando os dados são recebidos e com a memória de recepção ajustada para 2KB no canal 0.

6.2.12. Ajuste do tempo de retransmissão

W3100A usa o IRTR (*Initial Retry Time-value Register* - registrador de tempo de retransmissão) e RCR (*Retry Count Register* - registrador de contagem de tentativas) para ajustar o temporizador usado na retransmissão TCP.

A retransmissão TCP é executado quando o timer inicial de retransmissão expira, e o timer de retransmissão é reajustado para o dobro do valor atual. Tal processo é repetido de acordo com o valor RCR e, na última tentativa, uma interrupção de *timeout* ocorre e o sistema desiste.

Fórmula do valor de *timeout*

- ✓ IRTR: *Initial Retry Time-value Register* [2]
- ✓ RCR: *Retry Count Register*

[2] $IRTR \times 100\mu s$ = timeout inicial (em segudos), Valor de timeout Total até desistência: $(RTR \times 100\mu s) \times 2^{(RCR-1)}$

Internamente, o valor padrão de IRTR é 0x07D0 e RCR é 0x06, onde a tentativa inicial é realizada 200ms e a freqüência de retransmissão torna-se 6. Portanto, a menos que esses registradores sejam ajustados, as retransmissões são feitas em 200ms, 600ms, 1400ms, 3000ms, 6200ms, 12600ms e o sistema desiste nos 12600ms finais.

6.3. W3100A E PROTOCOLO I2C

O protocolo I2C usa as linhas SDA e SCL para permitir a transmissão e recepção serial de dados entre o MCU e o W3100A.

Como uma linha de *clock*, SCL precisa ser fornecida pelo MCU e SDA é usado como linha para transmitir dados e endereços entre o MCU e o W3100A 70.

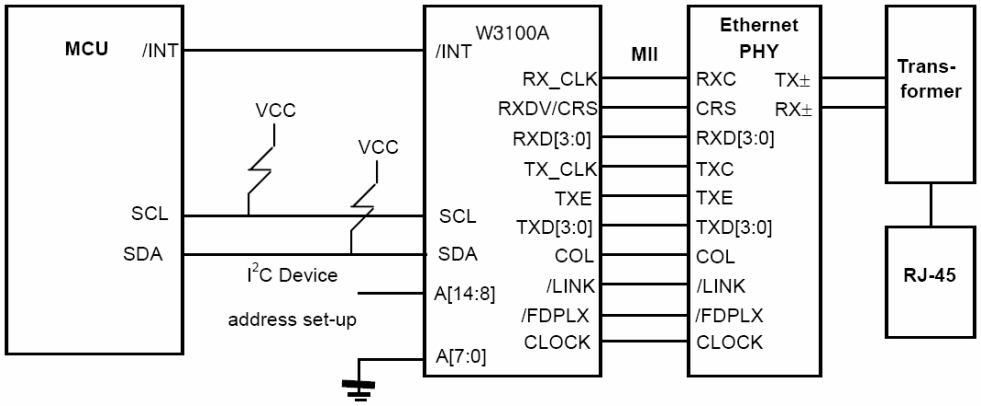


Figura 6.9 Interfaceamento I_2C

Um diagrama de blocos como o mostrado acima ilustra a conexão entre o MCU e o W3100A usando o I_2C. Como ilustrado acima, é recomendado ajustar o endereço para o I2C usando A[14:8] e fornecer resistores de pull-up de 4,7kΩ para as linha SCL e SDA e aterrar os pinos restantes de endereços A[7:0].

Para sincronizar o MCU e o W3100A, o protocolo I_2C exige uma condição de START antes de transmitir e receber dados e uma condição de STOP criada depois de completar a transmissão e a recepção de dados. Quando a linha SDA é posta em nível baixo enquanto a linha SCL está em nível alto, temos a sinalização da condição de START. Quando a linha SDA fica em alto enquanto SCL está em alto, temos a sinalização da condição de STOP.

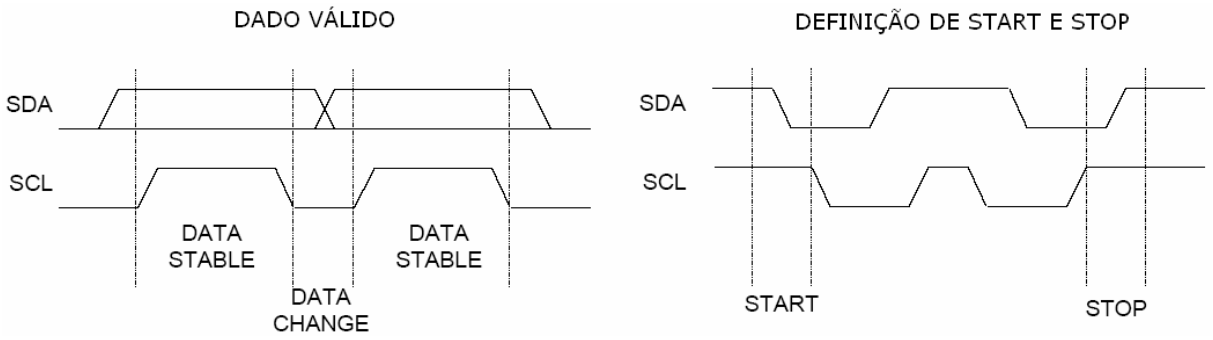


Figura 6.10 Diagramas do estado da comunicação no barramento I2C

Métodos de acesso envolvem acesso aleatório para leitura e escrita para dados de um byte e acesso seqüencial de leitura e escrita.

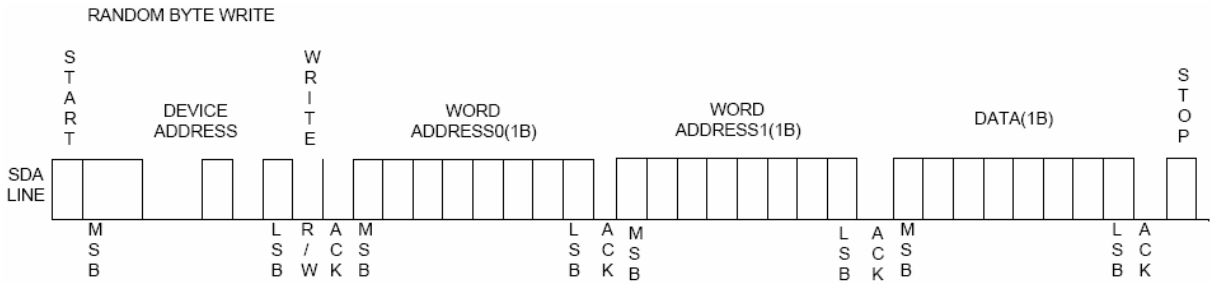


Figura 6.11 Diagrama de acesso aleatório de um byte

Acesso a um byte é ilustrados na Figura 6.11 acima e a escrita aleatória de byte ocorre seguindo a ordem START, DEVICE ADDRESS, endereço de 2 bytes do registrador a ser acessado, dado e STOP.

De maneira a ajustar o endereço do registrador a ser acessado, na leitura aleatória enviamos primeiro START, DEVICE ADDRESS e o endereço de 2 bytes do registrador a ser acessado, seguido de STOP. Depois, novamente, são enviados START e DEVICE ADDRESS seguidos dos dados enviados pelo W3100A e de STOP.

A escrita seqüencial de bytes inicia com START, DEVICE ADDRESS, endereço de 2 bytes do registrador a ser acessado e os dados, seguidos do STOP. Como resultado, o MCU pode designar o endereço dos dados uma única vez, o que permite aos dados serem escritos seqüencialmente e o endereço ser incrementado.

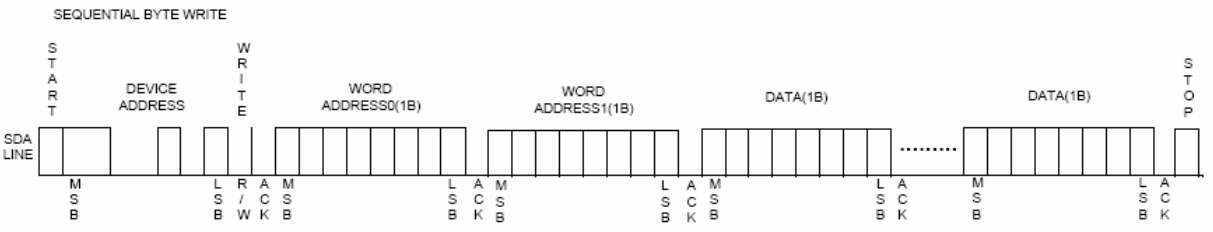


Figura 6.12 Diagrama de tempo para escrita seqüencial de dados

De maneira a ajustar o endereço do registrador a ser acessado, a leitura seqüencial envia primeiro START, DEVICE ADDRESS e o endereço de 2 bytes do registrador a ser acessado e, posteriormente STOP. Depois, novamente, START, DEVICE ADDRESS é enviado e o dado lido é enviado antes do STOP. Nesse caso, o endereço de acesso também é automaticamente incrementado.

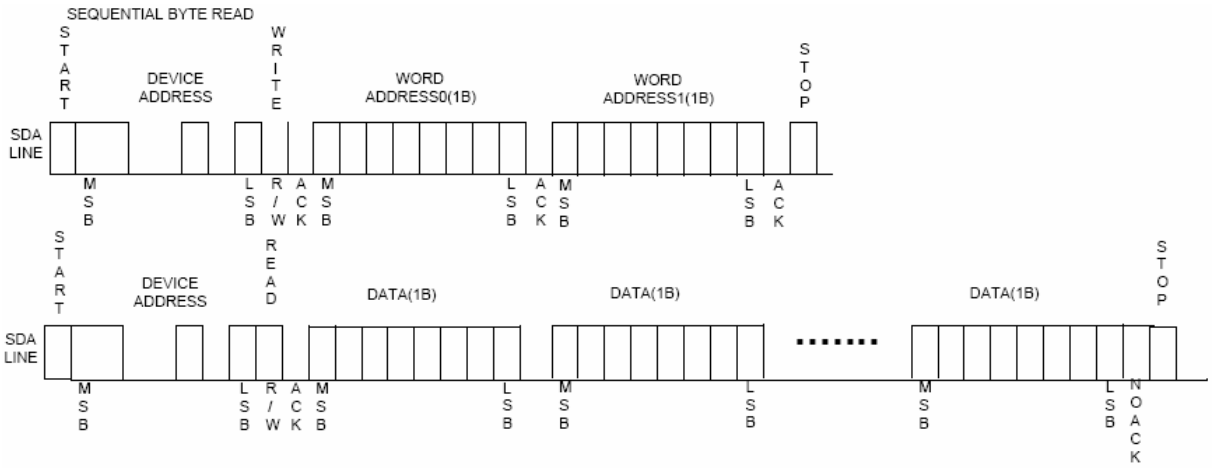


Figura 6.13 Diagrama de tempo para leitura seqüencial de dados

6.4. ESTRATÉGIAS DE IMPLEMENTAÇÃO:

Para a implementação de um servidor de páginas da Web embarcado usando um microcontrolador da família 8051, optou-se por usar o barramento I_2C por sua simplicidade eletrônica pois, para construir a interface, são necessárias somente 3 linhas: a linha SDA (linhas de dados duplex), linha SCLK (linha de sincronismo) e a linha de referência de tensão GND. Existe todo um protocolo que deve ser seguido a fim de conseguirmos efetivar as comunicações em I_2C.

O *datasheet* do circuito integrado W31000A apresenta instruções de utilização do barramento que contradizem implementações realizadas: o esquema de envio de bytes para os

registradores via barramento I_2C segue a convenção *Big Endian* enquanto o *datasheet* ilustra o contrário nos diagramas de tempo para a interface I_2C. Cabe uma ressalva a respeito do registrador que guarda o endereço de hardware ethernet, formado por seis bytes: esse exige que a escrita seja efetuada sob a convenção Little Endian.

O servidor de páginas da Internet usa o microcontrolador da família 8051 para configurar os registradores necessários à implementação e para realizar o controle de todo o sistema: no ambiente de desenvolvimento estão armazenadas rotinas de controle do barramento I_2C e seqüências de comandos necessárias ao correto comportamento do módulo IIM7010A como servidor de internet embarcado.

A fim de criar um servidor de páginas, o desenvolvedor deve conhecer como se comporta o protocolo HTTP e os navegadores de internet e usar esse conhecimento a seu favor para simplificar sua implementação.

Uma característica interessante em relação aos navegadores de internet é que eles interpretam o código HTML de maneira que o criador de uma página pode suprimir toda endentação e quebra de linhas que facilite a leitura do código por um humano. Isso permitiu economizar a memória destinada ao armazenamento da página da internet a ser servida.

O protocolo HTTP possui um método de confirmação de transmissão em que, uma vez que o cliente esteja conectado ao servidor, o cliente deve enviar um comando (GET) indicando que deseja receber a página provida. Os servidores de páginas da internet embarcados usualmente driblam o protocolo HTTP desconsiderando o comando GET e enviando a página provida imediatamente após o estabelecimento da conexão TCP.

O circuito integrado W31000A possui uma máquina de estados bem simplificada para o protocolo TCP/IP mas cujos recursos são suficientes para fornecerem a conectividade à internet a um ambiente minimalista, que é o caso dos microcontroladores. Contudo, a simplicidade implica o sistema ficar extremamente vulnerável a alguns ataques do tipo DoS (*Denial of Service*: recusa de serviço) por meio da execução de simples utilitários de rede como o nmap[3], disponível na maioria das distribuições Linux.

[3] Nmap é um escaneador de hosts que usa recursos avançados para verificar o estado do seu alvo. A ferramenta é gratuita e encontrada nas versões Linux, Windows (95,98,NT, Me, 2K e XP), Mac OS, Solaris, FreeBSD e OpenBSD. As versões para Windows e Linux contam com uma interface gráfica [11].

6.5. IMPLEMENTAÇÃO DO SERVIDOR DE INTERNET EMBARCADO

O ambiente de desenvolvimento para o controlador da família 8051 usado na realização do servidor embarcado age como controlador das transações TCP/IP por meio do ajuste dos registradores do CI W31000A e pelo envio de comandos ao mesmo sobre o barramento I2C.

A implementação do servidor de páginas da Internet embarcado foi usado um fluxograma bem simples, apresentado a seguir Figura 6.14

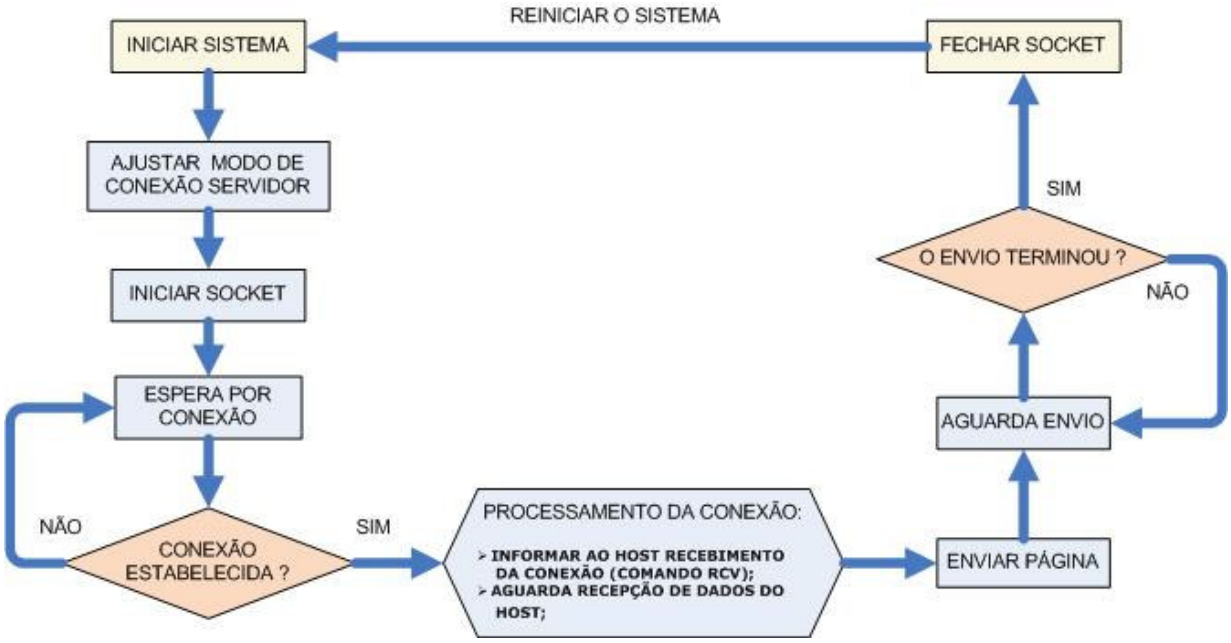


Figura 6.14 Estabelecimento de uma conexão http.

As etapas do processo do estabelecimento de uma conexão HTTP serão detalhadas abaixo.

6.5.1. Configuração Mínima Para Operação do Módulo

A fim de fazer o uso do módulo de Ethernet NM7010A o programador deve realizar a configuração dos seguintes registradores:

- ✓ GAR (*Gateway Address Register*): registrador que armazena o endereço IP do *gateway* a ser usado.

- ✓ SMR (*Subnet Mask Register*): registrador que armazena o número da máscara da sub-rede, usado pelo sistema para rotear os pacotes pela rede.

- ✓ SHAR (*Source Hardware Address Register*): registrador que armazena o endereço de *hardware* do PHY do módulo NM7010A.

- ✓ SIPR (*Source IP Register*): registrador que armazena o endereço IP do sistema.

6.5.2. Iniciar Sistema

O sistema é inicializado somente após o envio do comando SysInit, que é invocado por meio de um bit do registrador C0_CR (*Channel 0 Control Register*). Após esse passo, o programador pode verificar o funcionamento do sistema usando o comando *ping*, que envia pacotes ICMP Echo Request.

Após a inicialização do sistema, o tempo de retransmissão, a alocação de memória de recepção e de transmissão, o tipo de protocolo de comunicação a ser usado no canal e a porta de operação a ser usada devem ser ajustados. Os registradores envolvidos nessas configurações são os seguintes: IRTR (*Initial Time Retry Value*), RMSR (*Rx data Memory Size Register*), TMSR (*Tx data Memory Size Register*), SOPR (*Socket Option and Protocol Register*) e SPR (*Source Port Register*).

A fim de trabalhar com a transmissão de dados, o registrador ponteiro do canal X CX_TW_PR (*Tx Write Pointer Register of Channel X*) deve ser ajustado com um valor coerente com a alocação de memória de cada um dos canais a fim de que o módulo saiba de onde ele deve começar e terminar de ler dados da memória com fins de transmissão.

6.5.3. Iniciar *Socket*

Os *sockets* são inicializados somente após o envio do comando SocketInit, que é invocado por meio de um bit do registrador CX_CR (*Channel X Control Register*), onde X é o número de um dos canais do W3100A, que vão de 0 a 3. Após esse passo, o programador pode verificar o funcionamento do sistema usando o comando *ping*, que envia pacotes ICMP Echo Request.

Em seguida, o comando SocketListen também é invocado por meio da escrita no mesmo registrador citado acima, o que faz o módulo de *ethernet* aguardar por uma conexão.

6.5.4. Espera por Conexão

Enquanto nenhum *host* solicitar conexões, o módulo de *ethernet* aguarda em um laço infinito.

6.5.5. Processamento da Conexão

Se o sistema atingiu esse estado é porque algum *host* solicitou uma conexão TCP/IP. Se o usuário usou um *browser* de *Internet*, após o estabelecimento da conexão TCP/IP, ele deve ter invocado um comando GET, usado no protocolo HTTP; o sistema espera, portanto, a recepção desse comando e envia ao *host* uma confirmação do recebimento do comando. Mesmo que o comando não seja enviado, o módulo de *ethernet* segue carregando a memória de transmissão do sistema com uma página em HTML.

6.5.6. Enviar Página

Finda a transferência da página para a memória interna do W3100A, o microcontrolador atualiza os registradores ponteiros de transmissão e manda o comando SocketSend, invocado por meio de um bit do registrador CX_CR. O W3100A segue esvaziando a memória a transmissão até o valor apontado pelo registrador ponteiro.

6.5.7. Aguarda Envio

O sistema aguarda a confirmação de recebimento da página proveniente do *host*. Caso essa confirmação não seja recebida, o próprio W3100A encarrega-se de retransmitir os pacotes perdidos.

6.5.8. Fechar *Socket*

Os *sockets* são inicializados somente após o envio do comando SocketClose, que é invocado por meio de um bit do registrador CX_CR (*Channel X Control Register*).

6.5.9. Reiniciar Sistema

Nessa etapa, o sistema é novamente iniciado. Os Códigos usados para a implementação da Internet embarcada são os Código 03 – Rotina Principal I_2C para Trabalho com Módulo Ethernet e Código 04 – Internet Embarcada do Apêndice A.

7. RESULTADOS PRÁTICOS

Para a implementação de uma comunicação no padrão IrDA com o microcontrolador 8051 concluímos que a solução mais prática, tanto econômica quanto tecnologicamente era fazer uso de um outro dispositivo que interfaceasse essa comunicação. Assim seria posível reutilizar os dispositivos em outras aplicações e não ocuparíamos o microcontrolador 8051 com a comunicação dando assim mais portabilidade aos recursos utilizados.

Para a implementação de uma comunicação em infravermelho utilizamos a UART MAX3100 que enterfaceia o microcontrolador 8051. E para uma comunicação no padrão IrDA temos o MCP2150 da Microchip que já possui um pacote de protocolos do padrão IrDA emplementada em hardware que resolve toda a parte de codificação, decodificação e modulação de alguns dos protocolos do padrão IrDA. A comunicação com o dispositivo é via uma UART para o infravermelho um transceiver em infravermelho.

E com o módulo NM7010A que implementamos a conexão do 8051 coma internet.

7.1. RESULTADOS OBTIDOS COM O MAX3100

O MAX3100 não implementa o padrão IrDA, mas é uma ótima interface infravermelho. O MAX3100 se comunica serialmente com um dispositivo qualquer[4] através de uma UART e transmite e / ou recebe dados por infravermelho. A Figura 7.1 abaixo ilustra o circuito implementado para a utilização do MAX3100 no PC .

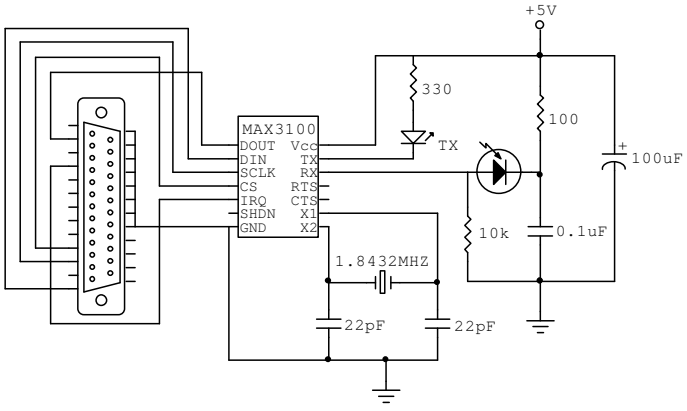


Figura 7.1 Circuito implementado para a utilização do MAX no PC

[4] - Qualquer dispositivo que tenha capacidade implementação de uma UART como por exemplo um Microcontrolador ou um PC.

A seguir na Figura 7.2 temos o layout da placa de circuito impresso implementada para o MAX3100. Para sua utilização com o PC fez-se uso da porta paralela. Utilizou-se a porta paralela e não a serial por causa da maior disponibilidade de pinos controláveis que a porta paralela tem e que são necessários para configurar o MAX3100.

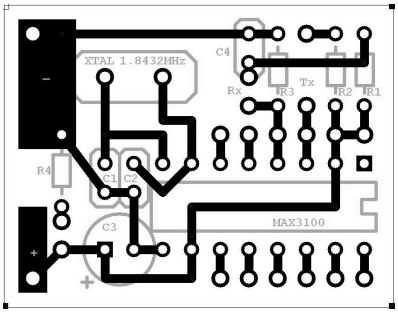
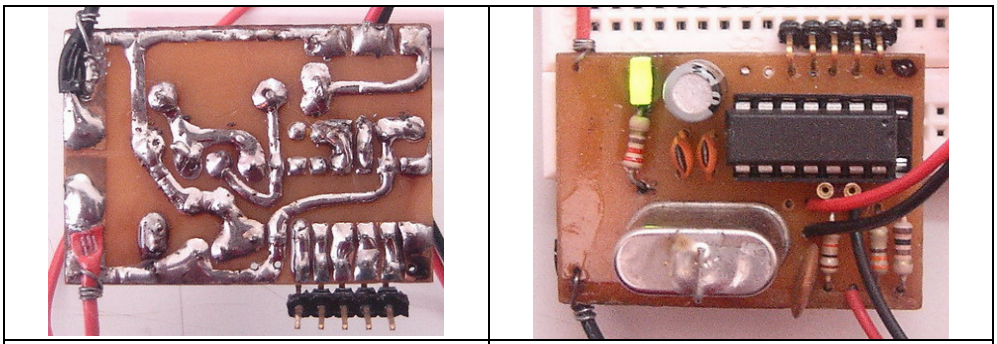


Figura 7.2 Layout da placa de implementação do MAX

Na Figura 7.3 abaixo é possível apreciar o circuito montado para o MAX3100 de modo a possibilitar a utilização essa UART com o microcontrolador 8051 e o Computador.



(a) *(b)*

Figura 7.3 Circuito montado para o MAX3100, (a) trilhas, (b) componente.

A comunicação via porta paralela do PC funciona com o programa: Código 01 – Comunicação do MAX3100 com o PC[5] do apêndice A. O circuito implementado para o PC ilustrado acima também serve para o placa microcontrolada pelo 8051 e o programa em assembly que implementa o seu funcionamento é: Código 02 – Comunicação do MAX3100 com o Microcontrolador 8051 do apêndice A. A fim de obter uma maior alcance colocou-se um *buffer* antes do LED infravermelho visando uma irradiação mais intensa e eficiente. Na Figura 7.4 abaixo temos ilustrado o esquema de montagem do MAX3100 como o computador.

[5] Para sua utilização do programa no Windows 2000 e XP faz se uso do programa: "userport.exe", encontrado na referencia [17], ele habilita no Windows XP a escrita na porta paralela.

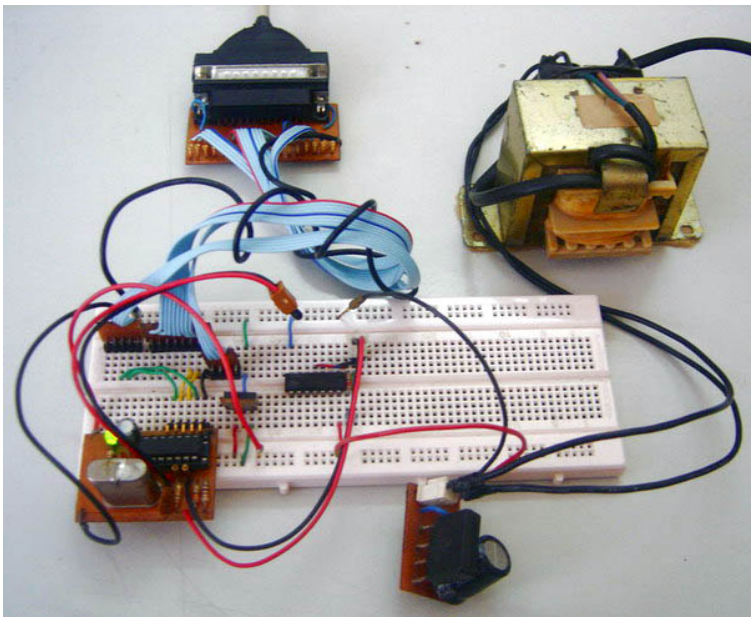


Figura 7.4 Esquema de montagem do MAX (buffer adaptado no Prontoboard)

Antes de se iniciar a comunicação no MAX3100 é preciso configura-lo. A tela abaixo da Figura 7.5 mostra quando a configuração do MAX3100 foi bem sucedida.

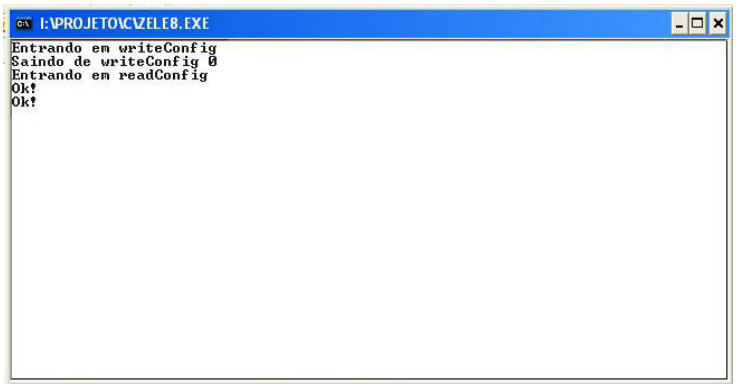


Figura 7.5 Tela de apresentação: Configuração bem sucedida.

E tela seguinte, Figura 7.6 ilustra quando a configuração falha. Tal falha pode ocorrer por algum mal contato ou por danifação do circuito integrado.

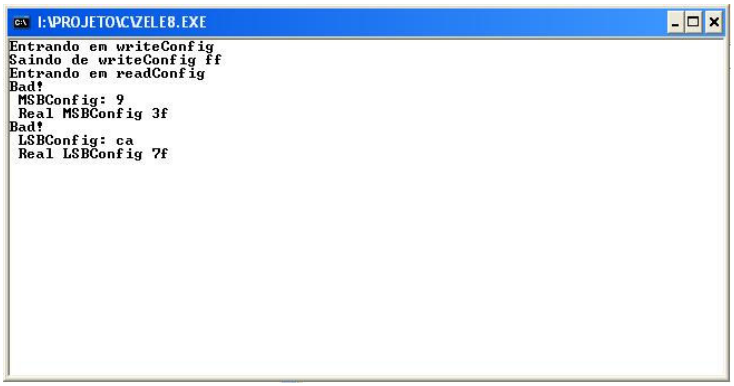


Figura 7.6 Tela de apresentação: Configuração falhou

A tela seguinte, Figura 7.7, exibe os dados (caracteres) sendo transmitidos e recebidos pela interface infravermelho do MAX3100:

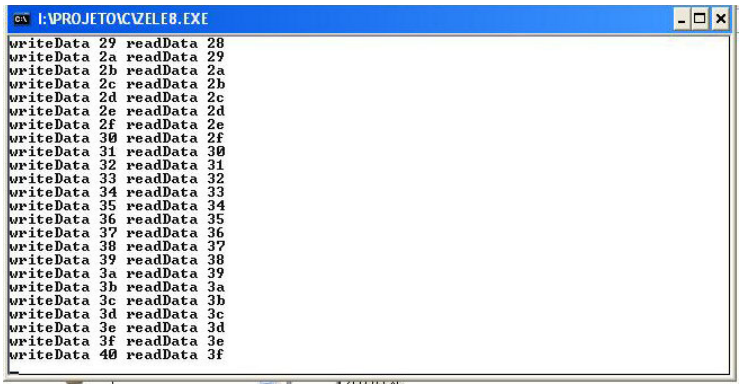


Figura 7.7 Tela de apresentação: Transmissão e Recepção

A Figura 7.8 abaixo mostra a visada direta do receptor e transmissor no MAX3100. Essa distância ilustradada não é máxima obtida.

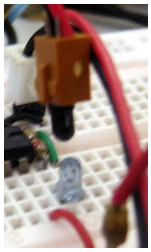
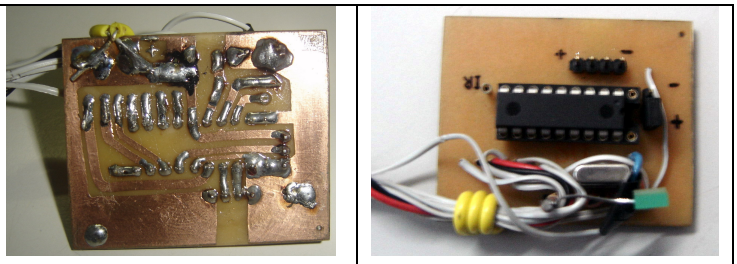


Figura 7.8 Visada entre o emissor (TXIR) Infravermelho e o receptor (RXIR)

7.2. RESULTADOS OBTIDOS COM O MCP2150

Para o MCP2150 fezemos a montamos a seguinte placa ilustrada na Figura 7.9 baixo. E segundo o seu Datasheet seria necessário apenas configura seu ID via software, pois se trata de um dispositivo secundário. O MCP agurada uma requisção de um dispositivo primário para então estabelecer uma conexão.



(a) trilhas do layout	*(b) componentes*

Figura 7.9 Foto do Circuito implementado para a utilização do MCP com o 8051.

7.2.1. Transceiver

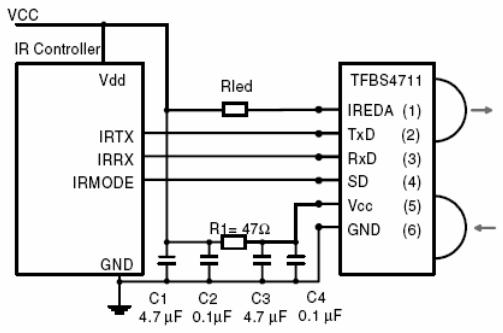
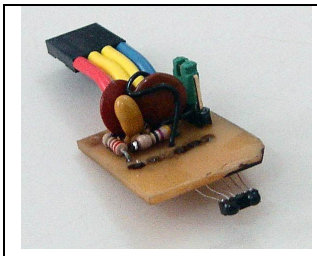
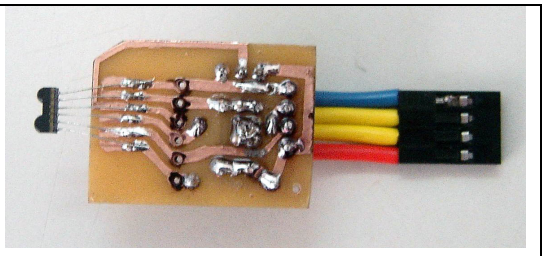


Figura 7.10 Circuito implementado para o transceiver.

Para o transceiver fezemos uma circuito daptável de modo que ele possa ser usado tanto no MAX3100 como no MCP2150 ou em outro dispositivo qualquer que necessite de um transceiver. De acordo com a figura o pino em vermelho ilustra a alimentação ao seu lado temos TX em amarelo e de azul o Terra (GND), ao lado do Terra temos RX. Na Figura 7.11 abaixo temos fotos de um dos modelos construidos.



(a) componetes



(b) trilhas do layout

Figura 7.11 Foto do circuito implementado para o transceiver

E na Figura 7.12 temos o Layout do circuito montado para o transceiver.

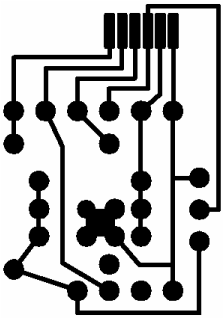


Figura 7.12 Layout da placa de implementação do transceiver

As figuras Figura 7.13 e Figura 7.14 abaixo ilustram o uso do transceiver com o MCP2150:

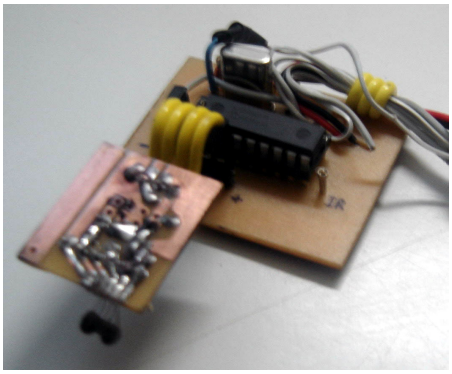


Figura 7.13 Transceiver e MCP2150 acoplados

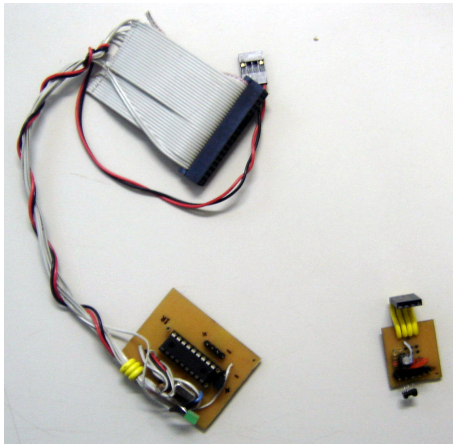


Figura 7.14 MCP2150, Transceiver e Cabo de conexão

7.3. RESULTADOS OBTIDOS COMO O I2C E INTERNET EMBARCADA

Na Figura 7.15 temos uma foto do Módulo IIM7010A que faz a interface do 8051 com a internet.

Figura 7.15 Módulo de Rede NM7010A

Conforme projetado o temos na Figura 7.16 a foto do circuito implementado para o modulo IIM7010A que fornece sua alimentação de 3,3V e faz a conexão via cabo flat com a Placa controladora Penta Valente (microcontrolador 8051):

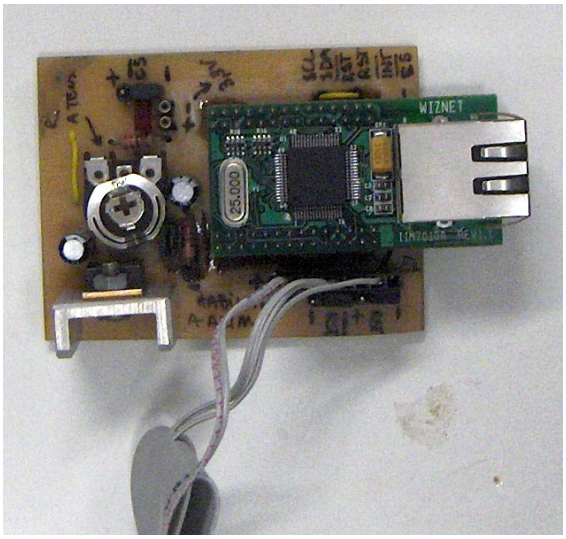


Figura 7.16 Foto Circuito implementado para a utilização do Módulo IIM7010A

A Figura 7.17 ilustra a conexão do módulo IIM7010A com a placa penta valente (microcontrolador 8051). Note que o circuito aproveita a alimentação da placa de 5,0 V para gera a sua alimentação de 3,3V que é feita pele integrado LM317 [14].

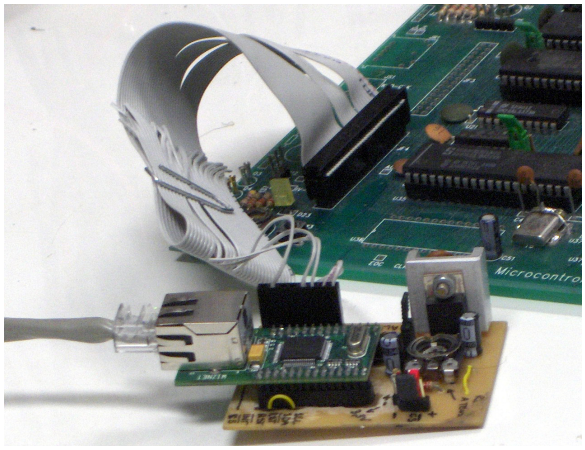


Figura 7.17 Conexão do Módulo à placa Penta Controladora [8051]

E na Figura 7.18 temos o Layout do circuito:

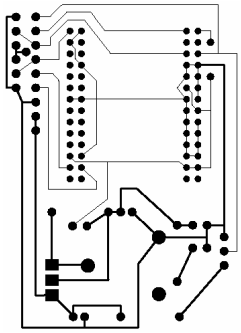


Figura 7.18 Layout do circuito implementado para o Módulo IIM7010A

Ao se acessar o microcontrolador a página gerada por ele é a seguinte ilustrada na figura Figura 7.19:

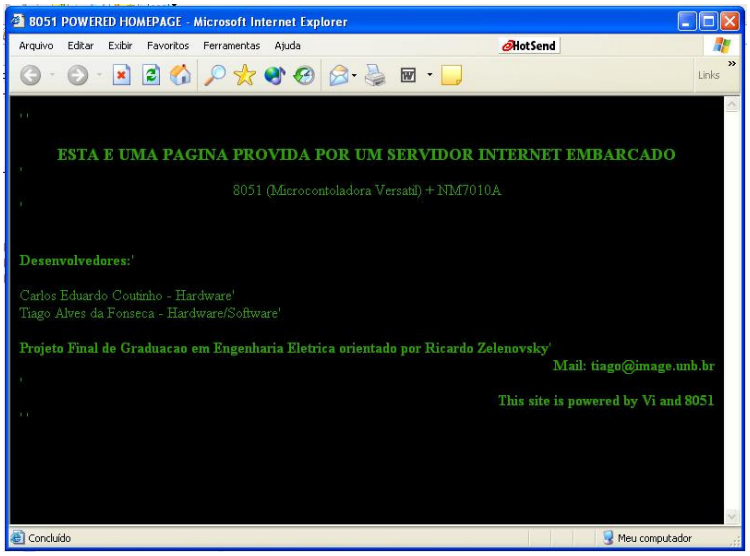


Figura 7.19 Página apresentada pelo 8051 via Módulo IIM7010A

8. CONCLUSÃO

Esse trabalho estudou de meios de comunicação e apresentou soluções da aplicação desses em um ambiente microcontrolado por 8051. Um servidor de páginas da internet foi implementado usando microcontrolador da família 8051 e um módulo de interfaceamento com a Ethernet, o NM7010A. Tal aplicação abrirá portas permitindo o uso de microcontroladores no sensoreamento de sistemas usando aquisição de dados via internet, aproveitando as redes lógicas já existentes nos ambientes. A comunicação sem fio por infravermelho facilitará também o sensoreamento remoto por meio da eliminação de conexões físicas entre o microcontrolador e o terminal de aquisição.

Trabalhos futuros poderão desenvolver sistemas de sensoreamento desenvolvendo aplicações em TCP/IP que enviem comandos para o microcontrolador via Ethernet e recebam relatórios de dados adquiridos.

BIBLIOGRAFIA

[1] IrDA Principles and Protocols" by Dr. Charles Knutson with Jeffrey Brown Now Available.

[2] PC: um Guia Prático de Hardware e Interfaceamento", (3ª edição - 2002) de Ricardo Zelenovsky e Alexandre Mendonça – disponível na URL http://www.mzeditora.com.br

[3] Microchip, fabricante do MCP2150 informações sobre o dispositivo e outros similares ou da mesma família – disponível na URL http://www.microchip.com

[4] Dallas Semiconductor MAXIM, fabricante do MAX3100 informações sobre o dispositivo e outros similares ou da mesma família – disponível na URL http://www.maxim-ic.com

[5] Vishay, fabricante do transceiver TFBS4711 informações sobre o dispositivo e outros similares ou da mesma família – disponível na URL http://www.vishay.com/

[6] Wiznet, fabricante do Módulo de Rede IIM7010A informações sobre o dispositivo e outros similares ou da mesma família – disponível na URL http://www.wiznet.co.kr

[7] Manual da família 8051 – disponível na URL http://www.eel.ufsc.br/eel7030/8051.pdf

[8] *Infrared Data Association*, Associação que define as especificações das comunicações sem fio em infravermelho – disponível na URL www.irda.org

[9] *The IrDA Standards for high-speed Infrared Communications*, The Hewlett Packard Journal – Article 2, February 1998. Artigo sobre o padrão IrDA – disponível na URL http://www.findarticles.com/p/articles/mi_m0HPJ/is_n1_v49/ai_20329859

[10] *Two-wire bus*, Tutorial sobre o barramento comunicação I_2C – disponível na URL http://www.geocities.com/SiliconValley/Program/3430/i2c.htm

[11] Sítio de pesquisa sobre redes com dicas de implementação de redes – disponível na URL http://www.invasao.com.br/coluna-andre-07.htm

[12] Datasheet do circuito integrado MCP2150 – disponível na URL http://www.microchip.com/downloads/en/DeviceDoc/21655b.pdf

[13] Datasheet do circuito integrado MAX3100 – disponível na URL http://pdfserv.maxim-ic.com/en/ds/MAX3100.pdf

[14] Datasheet do circuito integrado LM317 – disponível na URL http://www.alldatasheet.com

[15] Datasheet do circuito integrado TFBS4711 – disponível na URL http://www.vishay.com/search?query=tfbs4711&type=

[16] Datasheet do circuito integrado IIM7010A – disponível na URL
http://wiznet.co.kr/wiznet/product_nm7010a.html#software

[17] Utilizando a Porta Paralela nos Windows NT/XP/2000, disponível na URL
http://www.eletronica.org/modules.php?name=News&file=article&sid=142

APÊNDICE

APÊNDICE A – PROGRAMAS PARA O MAX3100

Código 01 – Comunicação do MAX3100 com o PC

```
/* MAX3100PC.CPP [MAX 3100]
 *
 * PROGRAMA QUE USA A PORTA PARALELA DO PC PARA GERAR OS SINAIS
 * NECESSÁRIOS A FIM DE CONTROLAR O MAX3100 VIA INTERFACE SPI. USA A
 * LINGUAGEM C++ PARA O SISTEMA OPERACIONAL DOS.
 */

#include <stdio.h>              //funcoes padrao
#include <conio.h>              //funcoes padrao
#include <ctype.h>              //funcoes padrao
#include <math.h>               //usado em buildOutput()
#include <dos.h>                //necessario para as rotinas de IO

//DEFINIÇÕES INTERESSANTES:

#define portIN 0x378    //porta para DIN
#define portOUT 0x379   //porta para DOUT
#define wMaskUp 0x02    //mascara para ativar !CS e por SCLK em nivel alto
#define wMaskDown 0x01  //mascara para ativar !CS e por SCLK em nivel baixo

/* VARIAVEIS GLOBAIS:
 * Usaram-se inteiros pela facilidade de tratamento, contudo eles podem
 * ser substituidos por chars
 */

int DIN;          //entrada de dados para IR
int DINMSB;       //entrada de dados para IR - MSB
int DINLSB;       //entrada de dados para IR - LSB
int DOUT;         //saida de dados de IR
int DOUTMSB;      //saida de dados de IR - MSB
int DOUTLSB;      //saida de dados de IR - LSB

//FUNÇOES DE BAIXO NIVEL:

void writeConfig();            //funcao de configuracao de escrita
void readConfig();             //funcao que testa configuracao de escrita
void iof(int dataMSB, int dataLSB);       //funcao de io
void buildOutput(int data, int expoente); //funcao de montagem dos bytes de
DOUT
void testConfig();            //funcao que testa a configuracao do MAX3100
void testWrote();             //funcao que testa a escrita no MAX3100
void testRead();              //funcao que testa a leitura no MAX3100
void writeData(int data);     //funcao que envia dados para MAX3100
void readData();              //funcao que recebe dados de MAX3100
void dataReady();             //funcao que verifica a disponibilidade de
dados no buffer do MAX3100

//ROTINA PARA USO FUTURO: tratamento da IRQ7:

void interrupt ISR_IRQ7();               //rotina de tratamento de
interrupcao
```

```
/*Rotina de tratamento de dados seriais: os bits sao invertidos no envio
 *a recepcao ja realiza a montagem dos dados
 */

int invertByte(int data);        //funcao que inverte os bytes para TXRX
void nop();                      //funcao de delay: usada para gerar atraso
entre o flanco de descida e a leitura do estado de 0x379

int main(void) {
      int contador=0x00;
      DOUTLSB = 0x00;
      clrscr();
      printf("Entrando em writeConfig\n");
      writeConfig();
      printf("Saindo de writeConfig %x\n",DOUTLSB);
      printf("Entrando em readConfig\n");
      readConfig();
      printf("Saindo de readConfig %x\n",DOUTLSB);
      getche();
      for(;;) {
            writeData(contador);
            readData();
            printf("writeData %x readData %x\n",contador,DOUTLSB);
            contador++;
      }
      getche();
      return 0;

}

void iof(int dataMSB, int dataLSB) {       //funcao que realiza a leitura e
escrita no MAX3100
      int data;                  //variavel local para tratamento de DIN
      int cache = 0x00;          //variavel local para tratamento de DOUT
      int count;

//    SELECIONAR CHIP

      DIN = 0x00;
      outport(portIN,DIN);              //escrita com SCLK baixo

//    CHIP SELECIONADO

//    inicio de MSB

      DOUT = 0x00;
      data = dataMSB;
      DIN = data&wMaskDown;
      outport(portIN,DIN);              //escrita com SCLK baixo
      DIN = DIN|wMaskUp;
      nop();
      outport(portIN,DIN);              //escrita com SCLK alto


      for(count=7;count;count--) {
            DIN = DIN&wMaskDown;
            nop();
            cache = inportb(portOUT);  //leitura em flanco de descida
            outport(portIN,DIN);       //escrita em borda de descida
            buildOutput(cache,count);
            data = data >> 1;
```

```
            DIN = data&wMaskDown;
            outport(portIN,DIN);       //escrita com SCLK baixo
            nop();
            DIN = DIN|wMaskUp;
            outport(portIN,DIN);       //escrita com SCLK alto
      }

      DIN = data&wMaskDown;
      nop();
      cache = inportb(portOUT);          //leitura em flanco de descida
      outport(portIN,DIN);               //escrita com SCLK baixo
      nop();
      buildOutput(cache,0);
      DOUTMSB = DOUT;

//    inicio de LSB

      DOUT = 0x00;
      data = dataLSB;
      DIN = data&wMaskDown;
      outportb(portIN,DIN);              //escrita com SCLK baixo
      DIN = DIN|wMaskUp;
      nop();
      outport(portIN,DIN);               //escrita com SCLK alto
      for(count=7;count;count--) {
            DIN = data&wMaskDown;
            nop();
            cache = inportb(portOUT);  //leitura em flanco de descida
            outport(portIN,DIN);       //escrita em flanco de descida
            nop();
            buildOutput(cache,count);
            data = data >> 1;
            DIN = data&wMaskDown;
            outportb(portIN,DIN);      //escrita com SCLK baixo
            DIN = DIN|wMaskUp;
            nop();
            outport(portIN,DIN);       //escrita com SCLK alto
      }

      DIN = data&wMaskDown;
      nop();
      cache = inportb(portOUT);          //leitura em flanco de descida
      outport(portIN,DIN);               //escrita com SCLK baixo
      nop();
      buildOutput(cache,0);
      DOUTLSB = DOUT;

//    DESELECIONAR CHIP

      DIN = 0x04;
      nop();
      outport(portIN,DIN);               //escrita com SCLK baixo
      DIN = 0x06;
      nop();

//    CHIP DESELECIONADO

}

void buildOutput(int data, int expoente) {
```

```
      data = data >> 5;                         //montagem DOUT: recuperacao
do bit 379.5
      DOUT += (data&0x01)*pow(2,expoente);    //montagem DOUT: soma de
montagem

      }

void writeConfig () {

      DINMSB = 0x27;       //MSB que sera rotacionado para entregar os
dados em DIN: configuracao de escrita
      DINLSB = 0x53;       //LSB que sera rotacionado para entregar os
dados em DIN
      iof(DINMSB,DINLSB);

      }

void readConfig () {

//AJUSTAR ESSA FUNCAO

      DINMSB = 0x02;       //MSB que sera rotacionado para entregar os
dados em DIN: configuracao de leitura
      DINLSB = 0x00;       //LSB que sera rotacionado para entregar os
dados em DIN: TEST ativado!
      iof(DINMSB,DINLSB);
      testConfig();
      }

void testConfig() {

//AJUSTAR ESSA FUNCAO

      int MSBConfig = 0x24;
      int LSBConfig = 0x53;
      if(MSBConfig==(DOUTMSB&0x3F))
            printf("Ok!\n");
      else
            printf("Bad!\n MSBConfig: %x\n Real MSBConfig
%x\n",MSBConfig,(DOUTMSB&0x3F));

      if( LSBConfig == DOUTLSB)
            printf("Ok!\n");
      else
      printf("Bad!\n LSBConfig: %x\n Real LSBConfig
%x\n",LSBConfig,DOUTLSB/2);
      getche();
}

int invertByte(int data) {

      int count, aux;
      aux = data;
      for(count=7;count>0;count--) {  //laco de inversao
             data = data >> 1;
             aux = aux<<1;
             aux = (data&1)|aux;
      }
      return (aux&255);
}
```

```
void writeData(int data) {

      DINMSB = invertByte(0x80);    //mascara de configuracao de escrita
      DINLSB = invertByte(data);    //dado a ser enviado ja invertido
      iof(DINMSB,DINLSB);
      testWrote();
}

void readData() {

      DINMSB = 0x00;                //mascara de configuracao de leitura:
MSB
      DINLSB = 0x00;                //mascara de configuracao de leitura:
LSB
      iof(DINMSB,DINLSB);
      testRead();
}

void testWrote() {}
void testRead() {}

//ACERTAR OS ENDERECOS

void interrupt ISR_IRQ7() {
      outportb(0x300,1);            //ativa bit 0 para iniciar reset do
pedido
      outportb(0x300,0);            //desativa bit 0 para finalizar reset do
pedido
      readData();
      outportb(0x20,0x20);          //transmissao do comando EOI
}

void dataReady() {                  //verifica se a IRQ7 esta ativada
(solucao simples)

      int aux;
      aux = inport(portOUT);        //usado para verificar ser o byte 0x379
esta com IRQ setado (nivel baixo)
      aux = (aux>>4)&1;               //pega o conteudo do bit somente
0x379.4
      printf("\n%x %x",aux,DOUTLSB);
      if(aux)
            return;
      else
            readData();
}

void nop() {
      delay(0);   // delay para estabilidade
}
```

Código 02 – Comunicação do MAX3100 com o Microcontrolador 8051

```
;  MAX31008051.ASM [MAX3100]
;  PROGRAMA EM ASSEMBLY QUE IMPLEMENTA A COMUNICAÇÃO VIA PORTA P1
;  DO MICROCONTROLADOR 8051 COM O CIRCUITO INTEGRADO MAX3100,
;  E ESTE POR SUA VEZ IMPLEMENTA UMA COMUNICAÇÃO EM IFRAVERMELHO
```

```
;  COM OUTRO DISPOSITIVO.


; E

$MOD51

; ENDERECOS DE IO PARA O MAX3100

DOUT    BIT     P1.6     ;PINO USADO PARA LER MAX3100
DIN     BIT     P1.5     ;PINO USADO PARA ESCREVER MAX3100
SCLK    BIT     P1.2     ;PINO USADO PARA GERAR SCLK MAX3100
CS      BIT     P1.1     ;PINO USADO PARA GERAR CS MAX3100
IRQ     BIT     P1.7     ;PINO USADO PARA LER IRQ MAX3100
FLAG    BIT     20H.0    ;FLAG RX

; POSICOES DA RAM

TX1     EQU     10H      ;BYTE MSB DE TRASMISSAO
TX2     EQU     11H      ;BYTE LSB DE TRASMISSAO
RX1     EQU     12H      ;BYTE MSB DE RECEPCAO
RX2     EQU     13H      ;BYTE LSB DE RECEPCAO
RX3     EQU     14H      ;BUFFER BYTE LSB DE RECEPCAO
                         ;RAZAO DE SER DO RX3: TODA TX RESULTA RX E TODA RX
EXIGE TX
                         ;JUSTIFICANDO GUARDAR RX2 PARA USOS FUTUROS

        ORG     0
        LJMP    MAIN
        ORG     13H
        LJMP    INTRX
        ORG     500H

INTRX:
                         ;MAX3100 TRABALHA POR NIVEL:
        MOV     TX1,#0   ;TESTAR PARA VER SE FUNCIONA BEM ATENDENDO POR
FLANCO
        MOV     TX2,#0
        ACALL   UTLK
        MOV     RX3,RX2  ;GUARDAR BYTE NO BUFFER PARA EVITAR PERDAS
        SETB    FLAG     ;DADO DISPONIVEL
        PUSH    ACC      ;DEBUG
        MOV     A,#'X'
        LCALL   SER_CHAR ;

        MOV     A,RX3
        LCALL   SER_W8
        MOV     A,#'X'
        LCALL   SER_CHAR ;
        POP     ACC      ;DEBUG
        RETI

MAIN:
                             ;CONFIGURACOES DA PORTA SERIAL
        MOV     TMOD,#20H    ;CT1 MODO 2
        MOV     TL1,#255     ;28800
        MOV     TH1,#255
        SETB    TR1          ;LIGAR CT0
        MOV     SCON,#0C8H   ;MODO SERIAL
        MOV     DPTR,#MSG
        LCALL   SER_STRC     ;AVISAR QUE FUNCIONA
```

```
                            ;FIM DE CONFIGURACOES DE PORTA SERIAL

        SETB    DOUT        ;PREPARANDO BITS PARA LEITURA
        SETB    IRQ         ;PREPARANDO BITS PARA LEITURA
        CLR     FLAG        ;INICIALIZANDO FLAG
        MOV     A,#'X'
        LCALL   SER_CHAR    ;
        MOV     A,#0FFH
        MOV     DPTR,#8200H
        MOVX    @DPTR,A
        INC     DPH
        MOVX    @DPTR,A
        INC     DPH
        MOVX    @DPTR,A
        INC     DPH
        MOVX    @DPTR,A
        INC     DPH
        MOVX    @DPTR,A
        INC     DPH
        MOVX    @DPTR,A
        MOV     A,#'x'
        LCALL   SER_CHAR     ;
        SETB    EX1          ;HABILITAR INTERRUPCAO EXTERNA
        SETB    PX1          ;AJUSTAR PRIORIDADE ELEVADA PARA INTERRUPCAO
EXTERNA
        SETB    IT1          ;HABILITAR INTERRUPCAO EXTERNA POR FLANCO
        SETB    EA           ;HABILITAR INTERRUPCOES
                             ;LEMBRAR DE DESABILITAR INTERRUPCOES NA
CHAMADA AS ROTINAS TX
        CLR     SCLK
        MOV     TX1,#0E4H    ;MSB PARA CONFIGURACAO MAX3100: SEM FIFO, SEM
SHUTDOWN, MASCARA DE IRQ
        MOV     TX2,#0CAH    ;LSB PARA CONFIGURACAO MAX3100: MODULACAO
IRDA, UM STOP BIT, SEM PARIDADE, 9600BPS
        CLR     EA
        ACALL   UTLK         ;CHAMADA DE ROTINA TX/RX MAX3100
        SETB    EA
        MOV     R7,#0
LBZZ:   MOV     TX1,#80H
        MOV     TX2,R7
        CLR     EA
        ACALL   UTLK
        MOV     A,#'-'
        LCALL   SER_CHAR
        MOV     A,#'-'
        ACALL   SER_CHAR
        MOV     A,#'-'
        LCALL   SER_CHAR
        MOV     C, IRQ
        CLR     A
        ADDC    A,#0
        LCALL   SER_W8
        MOV     A,#'-'
        LCALL   SER_CHAR
        MOV     TX1,#0
        MOV     TX2,#0
        CLR     EA
        ACALL   UTLK
        SETB    EA
        MOV     A,RX2
        LCALL   SER_W8
```

```
        MOV     A,R7
        LCALL   SER_W8
        ACALL   UTLK
        MOV     A,#'-'
        LCALL   SER_CHAR
        MOV     C, IRQ
        CLR     A
        ADDC    A,#0
        LCALL   SER_W8
        MOV     A,#'-'
        LCALL   SER_CHAR
        MOV     A,#LF
        LCALL   SER_CHAR
        INC     R7
        SJMP    LBZZ
LOOP:   JB      FLAG,URCV     ;VERIFICA DADOS MAX3100
NRECV:  JBC     RI,RCV51      ;VERIFICA DADOS 8051
        SJMP    LOOP
URCV:   CLR     FLAG          ;DADOS JAH CAPTURADOS: SENDO TRABALHADOS
        MOV     A,RX3         ;MANDAR DADOS RECEBIDOS VIA SERIAL
        MOV     SBUF,A        ;TX 8051
        SJMP    LOOP
RCV51:  MOV     A,SBUF        ;RECEBE DADOS 8051
        MOV     TX1,#80
        MOV     TX2,A
        ACALL   UTLK          ;CHAMADA DE ROTINA TX/RX MAX3100
        SJMP    LOOP
UTLK:   CLR     CS            ;ROTINA DE TX/RX MAX3100
        MOV     A,TX1
        ACALL   BYT8          ;ENVIA/RECEBE MSB MAX3100
        MOV     RX1,A
        MOV     A,TX2
        ACALL   BYT8          ;ENVIA/RECEBE LSB MAX3100
        MOV     RX2,A
        SETB    CS
        RET
BYT8:   MOV     R4,#8          ;GERACAO DE SINAIS PARA TX/RX MAX3100
        SETB    DOUT
        CLR     C
B8LP:   RLC     A
        MOV     DIN,C
        SETB    SCLK
        MOV     C,DOUT
        CLR     SCLK
        MOV     ACC.0,C
        DJNZ    R4,B8LP
        RET
MSG:    DB      'TENTANDO MAX3100', 0DH, 0AH, 0

$INCLUDE(rotina~1\B_CABEC.ASM)
$INCLUDE(rotina~1\B_SER.ASM)
$INCLUDE(rotina~1\B_LCD.ASM)
$INCLUDE(rotina~1\B_OUTRAS.ASM)

        END
```

APÊNDICE B – PROGRAMAS PARA O CI W31000A

Código 03 – Rotina Principal I_2C para Trabalho com Módulo Ethernet

```
;Net1.asm       [Internet Embarcada]
$MOD51
;ROTINA PRINCIPAL I2C PARA TRABALHO COM MODULO ETHERNET
;R7: ENDERECO DO REGISTRADOR PARA I2C E ROTINA DE TEMPO
;R6: DADO PARA I2C (ENDERECO DE 8 BITS), ENDERECO MSB DO REGISTRADOR PARA
I2C E ROTINA DE TEMPO
;R5: ROTINA DE TEMPO
;R4: LSB WRT
;R3: DADO I2C (ENDERECO DE 16 BITS)
;R2: LSB ACK
;R1: ENDERECO DO BUFFER PARA TRANSFERENCIA
;R0: QUANTIDADE DE BYTES DE TRANSFERENCIA I2C

;CHANGELOG:

;USO SOMENTE DO CANAL 0
;FLUXO LIMPO
;WEBSERVER OK
;INTERRUPTION POLLED
;PROXIMO PASSO: RESET NO SISTEMA EM CASO DE ERRO E SALTO PARA MAIN OU SALTO
PARA CONNECT SEM RESET

;CHANGELOG

I2C_ETHERNET  EQU   12H    ;ENDERECO I2C DO MODULO ETHERNET (SETAR
FISICAMENTE)

;-----------------------------------------------
;-REGISTRADORES DE SISTEMA---------------------
;-----------------------------------------------

GAR           EQU   80H    ;ENDERECO DE GW (4 BYTES)
SMR           EQU   84H    ;MASCARA DE SUBREDE (4 BYTES)
SHAR          EQU   88H    ;ENDERECO DE HARDWARE (6 BYTES)
SIPR          EQU   8EH    ;ENDERECO DE IP DE ORIGEM (4 BYTES)
IRTR          EQU   92H    ;ENDERECO DE TIME RETRY (2 BYTES)
RMSR          EQU   95H    ;TAMANHO DA MEMORIA DE RX
TMSR          EQU   96H    ;TAMANHO DA MEMORIA DE TX

;-----------------------------------------------
;-REGISTRADORES DE CANAL-----------------------
;-----------------------------------------------

C0SOPR        EQU   0A1H   ;OPCOES DE SOCKET E PROTOCOLO CANAL 0
C0SPR         EQU   0AEH   ;PORTA DE ORIGEM CANAL 0 (2 BYTES)

;-----------------------------------------------
;-REGISTRADORES DE CONTROLE--------------------
;-----------------------------------------------

C0_CR         EQU   0H      ;REGISTRADOR DE COMANDO DO CANAL 0
C0_ISR        EQU   04H     ;REGISTRADOR DE ESTADO DE INTERRUPCAO DO CANAL
0
IR            EQU   08H     ;REGISTRADOR DE INTERRUPCAO
IMR           EQU   09H     ;REGISTRADOR DE MASCARA DE INTERRUPCAO
```

```
;-----------------------------------------------
;-REGISTRADORES PONTEIROS----------------------
;-----------------------------------------------

C0_TW_PR      EQU   40H     ;REGISTRADOR PONTEIRO DE ESCRITA DE TRANSMISSAO
DO CANAL 0
C0_STW_PR     EQU   0F0H    ;REGISTRADOR PONTEIRO DE ESCRITA DE TRANSMISSAO
DO CANAL 0
C0_TR_PR      EQU   44H     ;REGISTRADOR PONTEIRO DE LEITURA DE TRANSMISSAO
DO CANAL 0
C0_STR_PR     EQU   0F1H    ;REGISTRADOR PONTEIRO DE LEITURA DE TRANSMISSAO
DO CANAL 0
C0_TA_PR      EQU   18H     ;REGISTRADOR PONTEIRO DE ACK DE TRANSMISSAO DO
CANAL 0
C0_STA_PR     EQU   0E2H    ;REGISTRADOR PONTEIRO DE ACK DE TRANSMISSAO DO
CANAL 0

;-----------------------------------------------
;-DECLARACOES BY ZELE--------------------------
;-----------------------------------------------

LEDS_ADR      EQU   8000H   ;ENDERECO IO DOS LEDS
LCD_DADO      EQU   8100H   ;ENDERECO IO LDC
SDA           EQU   P1.5    ;DADOS I2C
SCL           EQU   P1.4    ;CLOCK I2C
XMT_DAT       EQU   0H      ;BUFFER DE TRANSMISSAO DE 2KB (PARA BUFFER
INTERNO USAR 0CH)
RCV_DAT       EQU   800H    ;BUFFER DE RECEPCAO DE 2KB
DPH0          EQU   10H     ;DPTR0: ENDERECO DO BUFFER
DPL0          EQU   11H
DPH1          EQU   12H     ;DPTR1: TAMANHO DO BUFFER 16BITS
DPL1          EQU   13H
XMT_DATI      EQU   14H     ;BUFFER DE TRANSMISSAO INTERNO DE 8B + CONTADOR
(SINTAXE)
RCV_DATI      EQU   1DH     ;BUFFER DE RECEPCAO INTERNO DE 8B + CONTADOR
(SINTAXE)
TWMSB         EQU   1DH     ;POSICAO DE MEMORIA USADA PARA ARMAZENAR A
POSICAO DO REGISTRADOR PONTEIRO (MSB)
TWLSB         EQU   1EH     ;POSICAO DE MEMORIA USADA PARA ARMAZENAR A
POSICAO DO REGISTRADOR PONTEIRO (LSB)
IRQ           EQU   P1.7

;-----------------------------------------------
;-FIM DECLARACOES------------------------------
;-----------------------------------------------

              ORG   0
              LJMP  MAIN
              ORG   13H
              LJMP  INTNET
              ORG   100H
INTNET:                       ;FLANCO OU NIVEL?

              PUSH   ACC      ;DEBUG
              MOV    A,#'X'
              LCALL  SER_CHAR ;
              POP    ACC      ;DEBUG
              RETI
MAIN:
              SETB   IRQ
              SETB   SW3
```

```
              SETB   EX1       ;HABILITAR INTERRUPCAO EXTERNA
              SETB   PX1       ;AJUSTAR PRIORIDADE ELEVADA PARA INTERRUPCAO
EXTERNA
              SETB   IT1       ;HABILITAR INTERRUPCAO EXTERNA POR FLANCO
              SETB   EA        ;HABILITAR INTERRUPCOES
                               ;LEMBRAR DE DESABILITAR INTERRUPCOES NA
CHAMADA AS ROTINAS TX
              MOV    SP,#2Fh   ;#2F Set stack to start at 30h.
                               ;PARA NAO INVADIR AREA BIT A BIT
                ;CONFIGURAR PORTA SERIAL
              MOV    TMOD,#20H           ;CT1 MODO 2
              MOV    TL1,#255            ;28800
              MOV    TH1,#255
              SETB   TR1                 ;LIGAR CT0
              MOV    SCON,#0C8H          ;MODO SERIAL
              MOV    DPTR,#MSG
              LCALL  SER_STRC            ;AVISAR QUE FUNCIONA

;FIM DE CONFIGURACOES DE PORTA SERIAL

              SETB   SW3                 ;PREPARAR SW3 PARA LEITURA
              MOV    TWLSB,#0
              MOV    TWMSB,#0
              MOV    A,#'X'
              LCALL  SER_CHAR            ;
              MOV    DPTR,#GATEWAY
              MOV    R1,#XMT_DATI
              LCALL  TRANSFER            ;ESCRITA DA STRING NO BUFFER
              LCALL  DWRITE              ;ESCRITA DE DADOS I2C
              MOV    DPTR,#MASCSUBREDE
              MOV    R1,#XMT_DATI
              LCALL  TRANSFER            ;ESCRITA DA STRING NO BUFFER
              LCALL  DWRITE              ;ESCRITA DE DADOS I2C
              MOV    DPTR,#HARDADDR
              MOV    R1,#XMT_DATI
              LCALL  TRANSFER            ;ESCRITA DA STRING NO BUFFER
              LCALL  DWRITE              ;ESCRITA DE DADOS I2C
              MOV    DPTR,#SOURCEIP
              MOV    R1,#XMT_DATI
              LCALL  TRANSFER            ;ESCRITA DA STRING NO BUFFER
              LCALL  DWRITE              ;ESCRITA DE DADOS I2C
              MOV    DPTR,#INTMASK
              MOV    R1,#XMT_DATI
              LCALL  TRANSFER            ;ESCRITA DA STRING NO BUFFER
              LCALL  DWRITE              ;ESCRITA DE DADOS I2C
              MOV    DPTR,#CTRLINITSYS
              MOV    R1,#XMT_DATI
              LCALL  TRANSFER            ;ESCRITA DA STRING NO BUFFER
              LCALL  DWRITE              ;ESCRITA DE DADOS I2C
              MOV    R7,#0H              ;INT
              MOV    R6,#08H             ;APAGAR OS PEDIDOS!
              LCALL  BREAD2
              LCALL  SER_W8


;COLOCAR AQUI LEITURA DO ESTADO DO CANAL 0

              MOV    R7,#0H
              MOV    R6,#04H
              LCALL  BREAD2
              LCALL  SER_W8
```

```
;FIM LEITURA DO ESTADO

                MOV     A,#'X'
                LCALL   SER_CHAR          ;
                MOV     A,#'Y'
                LCALL   SER_CHAR          ;
                MOV     DPTR,#TIMERETRY
                MOV     R1,#XMT_DATI
                LCALL   TRANSFER          ;ESCRITA DA STRING NO BUFFER
                LCALL   DWRITE            ;ESCRITA DE DADOS I2C
                MOV     DPTR,#RXMEMORY
                MOV     R1,#XMT_DATI
                LCALL   TRANSFER          ;ESCRITA DA STRING NO BUFFER
                LCALL   DWRITE            ;ESCRITA DE DADOS I2C
                MOV     DPTR,#TXMEMORY
                MOV     R1,#XMT_DATI
                LCALL   TRANSFER          ;ESCRITA DA STRING NO BUFFER
                LCALL   DWRITE            ;ESCRITA DE DADOS I2C
                MOV     DPTR,#OPCANAL0
                MOV     R1,#XMT_DATI
                LCALL   TRANSFER          ;ESCRITA DA STRING NO BUFFER
                LCALL   DWRITE            ;ESCRITA DE DADOS I2C
                MOV     DPTR,#PTORGCANAL0
                MOV     R1,#XMT_DATI
                LCALL   TRANSFER          ;ESCRITA DA STRING NO BUFFER
                LCALL   DWRITE            ;ESCRITA DE DADOS I2C
MOV               R7,#0H                            ;INT
                MOV     R6,#08H           ;APAGAR PEDIDOS DE INT!
                LCALL   BREAD2
                LCALL   SER_W8
                MOV     DPTR,#CTRLCANAL0 ;INICIO DE SOCKET
                MOV     R1,#XMT_DATI
                LCALL   TRANSFER          ;ESCRITA DA STRING NO BUFFER
                LCALL   DWRITE            ;ESCRITA DE DADOS I2C
                MOV     DPTR,#SRVCANAL0   ;PREPARAR CANAL 0 PARA ESCUTA DE
CONEXOES
                MOV     R1,#XMT_DATI
                LCALL   TRANSFER          ;ESCRITA DA STRING NO BUFFER
                LCALL   DWRITE            ;ESCRITA DE DADOS I2C
                MOV     DPTR,#TCPC0TWPR
                MOV     R1,#XMT_DATI
                LCALL   TRANSFER          ;ESCRITA DA STRING NO BUFFER
                LCALL   DWRITE            ;ESCRITA DE DADOS I2C

CONNECTION:                                       ;PROCESSAMENTO DA CONEXAO
HTTP

                MOV     A,#'C'
                LCALL   SER_CHAR          ;
                MOV     R7,#0H
                MOV     R6,#C0_ISR
                LCALL   BREAD2            ;LEITURA DE DADOS I2C
                LCALL   SER_W8
                                          ;ATENCAO AQUI
LBSR:           JB      IRQ,$
                MOV     R7,#0H
                MOV     R6,#C0_ISR
                LCALL   BREAD2            ;LEITURA DE DADOS I2C
                JNB     ACC.2,LBS1        ;CONEXAO ESTABALECIDA
                MOV     A,#'S'
```

```
                LCALL   SER_CHAR                ;
                MOV     DPTR,#CTRLCANAL0RECV    ;ENVIAR COMANDO DE RECEIVE PARA
O OUTRO LADO DA CONEXAO
                MOV     R1,#XMT_DATI
                LCALL   TRANSFER          ;ESCRITA DA STRING NO BUFFER
                LCALL   DWRITE            ;ESCRITA DE DADOS I2C
                LJMP    LBSR
LBS1:           JNB     ACC.6,LBS2        ;CONEXAO RECEBEU OK
                MOV     A,#'H'
                LCALL   SER_CHAR          ;
                                          ;LEITURA DO SHADOW
                MOV     R7,#1H
                MOV     R6,#C0_STW_PR
                LCALL   BREAD2            ;LEITURA DE DADOS I2C
                                          ;LEITURA DO TW
                MOV     R7,#0H
                MOV     R6,#C0_TW_PR
                LCALL   BREAD2            ;LEITURA DE DADOS I2C
                INC     R6
                LCALL   BREAD2            ;LEITURA DE DADOS I2C
                INC     R6
                LCALL   BREAD2            ;LEITURA DE DADOS I2C
                MOV     TWMSB,A           ;CUIDADO COM R2
                INC     R6
                LCALL   BREAD2            ;LEITURA DE DADOS I2C
                MOV     TWLSB,A           ;INTERESSA-NOS APENAS LSB
                                          ;MANDAR PAGINA AQUI
                MOV     DPTR,#PAGINA2
                MOV     DPL0,#0
                MOV     DPH0,#0
                LCALL   TRANSFER3
                LCALL   DWRITE3
                                          ;MANDAR PAGINA AQUI
                                          ;ATUALIZACAO DE TW
                                          ;CALCULOS
                CLR     C
                MOV     A,#0F7H
                ADD     A,TWLSB
                MOV     TWLSB,A
                MOV     A,#2
                ADDC    A,TWMSB
                MOV     TWMSB,A
                                          ;ESCRITA NO REGISTRADOR TW
                MOV     R7,#0H
                MOV     R6,#C0_TW_PR
                MOV     R3,#00H
                LCALL   BWRITE2           ;LEITURA DE DADOS I2C
                INC     R6
                MOV     R3,#00H
                LCALL   BWRITE2           ;LEITURA DE DADOS I2C
                INC     R6
                MOV     R3,TWMSB
                LCALL   BWRITE2           ;LEITURA DE DADOS I2C
                INC     R6
                MOV     R3,TWLSB
                INC     R3
                LCALL   BWRITE2           ;LEITURA DE DADOS I2C
                                          ;FIM DA ATUALIZACAO DE TW
                                          ;AGORA MANDAR SEND
                MOV     DPTR,#CTRLCANAL0SEND
                MOV     R1,#XMT_DATI
```

```
              LCALL  TRANSFER         ;ESCRITA DA STRING NO BUFFER
              LCALL  DWRITE           ;ESCRITA DE DADOS I2C
                                      ;OK: A PAGINA DEVE TER SIDO
TRANSMITIDA

              MOV    A,#'s'
              LCALL  SER_CHAR         ;
              LJMP   LBSR
LBS2:         JNB    ACC.5,LBS21
              SJMP   LBS22
LBS21:        LJMP   LBSR
LBS22:
              ;TRANSMISSAO PROVAVELMENTE ACABOU: FECHAR O SOCKET E
REINICIALIZA-LO

              MOV    A,#'L'
              LCALL  SER_CHAR            ;
              MOV    DPTR,#CTRLCANAL0CLOSE
              MOV    R1,#XMT_DATI
              LCALL  TRANSFER            ;ESCRITA DA STRING NO BUFFER
              LCALL  DWRITE              ;ESCRITA DE DADOS I2C

                ;SETUP DA CONEXAO

              MOV    DPTR,#CTRLCANAL0    ;INICIAR SOCKET
              MOV    R1,#XMT_DATI
              LCALL  TRANSFER            ;ESCRITA DA STRING NO BUFFER
              LCALL  DWRITE              ;ESCRITA DE DADOS I2C
              MOV    DPTR,#SRVCANAL0     ;LISTEN CHANNEL 0
              MOV    R1,#XMT_DATI
              LCALL  TRANSFER            ;ESCRITA DA STRING NO BUFFER
              LCALL  DWRITE              ;ESCRITA DE DADOS I2C
              MOV    R7,#0H              ;LEITURA DE STATUS
              MOV    R6,#C0_ISR
              LCALL  BREAD2              ;LEITURA DE DADOS I2C
              LJMP   LBSR
              CJNE   A,#0AH,PANIC
                                         ;CONSERTAR ESSE TRECHO

              LJMP   LBSR
PANIC:        MOV    A,#'P'
              LCALL  SER_CHAR
              SJMP   $

;-----------------------------------------------
;-STRINGS PARA I2C-----------------------------
;-----------------------------------------------

;-STRINGS ESSENCIAIS-SUFICIENTES PARA PING------

;STRING PARA ENDERECO DO GATEWAY
GATEWAY:      DB    6, 00, GAR, 192, 168, 10, 1
;STRING PARA MASCARA DE SUBREDE
MASCSUBREDE:  DB    6, 00, SMR, 255, 255, 255, 0
;STRING DE ENDERECO DE HARDWARE
HARDADDR:     DB    8, 00, SHAR, 00H, 0FAH, 0FAH, 0FAH, 0FAH, 0FAH
;STRING DE IP DE ORIGEM
SOURCEIP:     DB    6, 00, SIPR, 192, 168, 10, 3

;-STRINGS DE REGISTRADORES DE SISTEMA-----------
```

```
;STRING DE TIMERETRY VALUE
TIMERETRY:    DB    4, 00, IRTR, 0E8H, 03H ; 03H, 0E8H
;STRING DE CONFIGURACAO DE MEMORIA RX
RXMEMORY:     DB    3, 00, RMSR, 03H;55H
;STRING DE CONFIGURACAO DE MEMORIA TX
TXMEMORY:     DB    3, 00, TMSR, 03H;55H

;-STRINGS DE REGISTRADORES DE CANAL-------------

;STRING REGISTRADOR DE OPCOES DO CANAL 0
OPCANAL0:     DB    3, 00, C0SOPR, 1
;STRING REGISTRADOR DE PORTA DE ORIGEM CANAL 0
PTORGCANAL0:   B    4, 00, C0SPR, 1FH, 90H; 21

;-STRINGS DE REGISTRADORES PONTEIROS------------

;STRING REGISTRADOR PONTEIRO
TCPC0TWPR:    DB    6, 00, C0_TW_PR, 00H, 40H, 0H, 0
;STRING REGISTRADOR PONTEIRO
TCPC0TRPR:    DB    6, 00, C0_TR_PR, 00H, 40H, 0H, 0
;STRING REGISTRADOR PONTEIRO
TCPC0TAPR:    DB    6, 00, C0_TA_PR, 00H, 40H, 0H, 0

;-STRINGS DE REGISTRADORES DE CONTROLE----------

;STRING REGISTRADOR DE CONTROLE INITSYS
CTRLINITSYS:  DB    3, 00, C0_CR, 1
;STRING REGISTRADOR DE CONTROLE: INITSOCK CANAL 0
CTRLCANAL0:   DB    3, 00, C0_CR, 2
;STRING REGISTRADOR DE CONTROLE: INITSOCK CANAL 0
CTRLCANAL0CLOSE: DB   3, 00, C0_CR, 10H
;STRING REGISTRADOR DE CONTROLE: INITSOCK CANAL 0
CTRLCANAL0RECV:  DB   3, 00, C0_CR, 40H
;STRING REGISTRADOR DE CONTROLE: INITSOCK CANAL 0
CTRLCANAL0SEND:  DB   3, 00, C0_CR, 20H
;STRING REGISTRADOR DE CONTROLE: LISTEN CANAL 0
SRVCANAL0:       DB   3, 00, C0_CR, 08H;8
;STRING REGISTRADOR DE STATUS CANAL 0
STATUSCANAL0:    DB   2, 00, C0_ISR
;STRING REGISTRADOR DE CONTROLE HABILITANDO INTERRUPCOES: CANAL 0

INTMASK:      DB     3, 00, IMR, 1;0;0FFH
PAGINA2:      DB     ' <HTML><HEAD><TITLE>8051 POWERED
HOMEPAGE</TITLE></HEAD>'
              DB     '<BODY BGCOLOR="#000000" LINK="#64EC18" TEXT="#2BA306"
VLINK="#3FEC3C">'
              DB     '<P><CENTER><B><FONT SIZE="4">ESTA E UMA PAGINA
PROVIDA POR UM SERVIDOR INTERNET EMBARCADO</FONT></B></CENTER>'
              DB     '<CENTER><FONT SIZE="3">8051 (Microcontoladora
Versatil) + NM7010A</FONT></CENTER>'
              DB     '<P><BR><FONT SIZE="3"><B>Desenvolvedores:</B>'
              DB     '<BR><BR>Carlos Eduardo Coutinho - Hardware'
              DB     '<BR>Tiago Alves da Fonseca - Hardware/Software'
              DB     '<BR><BR><FONT SIZE="3"><B>Projeto Final de Graduacao
em Engenharia Eletrica orientado por Ricardo Zelenovsky</B>'
              DB     '</FONT></FONT></P>'
              DB     '<DIV ALIGN="RIGHT"><FONT SIZE="3"><B>Mail:
tiago@image.unb.br</B></FONT></DIV>'
              DB     '<DIV ALIGN="RIGHT"><FONT SIZE="3"><B>This site is
powered by Vi and 8051</B></FONT></DIV>'
              DB     '</BODY></HTML> '
```

```
MSG:          DB     'TENTANDO ETHERNET', 0DH, 0AH, 0

;INCLUDES

$INCLUDE(rotina~1\B_CABEC.ASM)
$INCLUDE(rotina~1\B_SER.ASM)
$INCLUDE(rotina~1\B_I2CWEB.ASM)
$INCLUDE(rotina~1\B_OUTRAS.ASM)

              END
```

Código 04 – Internet Embarcada

```
;B_i2cweb.asm [INTERNT EMBARCADA]


;MUDAMOS CONTEUDO DE R2 E R4 EM DWRITE3

SL_ADR_RD:      MOV     A,#I2C_ETHERNET     ;ROTINA DE ESCRITA DO ENDERECO
I2C DE LEITURA 8583
                INC     A
                LCALL   VAI_BYTE
       RET
      ;
SL_ADR_WR:      MOV     A,#I2C_ETHERNET     ;ROTINA DE ESCRITA DO ENDERECO
I2C DE ESCRITA 8583
                LCALL   VAI_BYTE
       RET
VAI_BYTE:MOV    R2,#8                       ;ROTINA DE ENVIO DE BYTE I2C
LBX:     RLC    A
       MOV  SDA,C
         LCALL  TMP
       SETB SCL
         LCALL  TMP
       CLR  SCL
         LCALL  TMP
         DJNZ   R2,LBX
       RET
VEM_BYTE:SETB   SDA             ;ROTINA DE RECEPCAO DE BYTE I2C
         LCALL  TMP
         MOV      R2,#8
LBY:     SETB   SCL
         LCALL  TMP
       MOV  C,SDA
       RLC  A
       CLR  SCL
         LCALL  TMP
       DJNZ R2,LBY
       RET

STARTI2C:SETB   SCL             ;ROTINA DE START I2C
         LCALL  TMP
         CLR    SDA
         LCALL  TMP
       CLR  SCL
         LCALL  TMP
       RET
STOPI2C: SETB   SCL             ;ROTINA DE STOP I2C
         LCALL  TMP
```

```
       SETB SDA
         LCALL  TMP
       RET
ACKI2C:  SETB   SDA             ;ROTINA DE ACKNOLEDGE I2C
         LCALL  TMP
       SETB SCL
         LCALL  TMP
       MOV  C,SDA
       JC   ERRO1
       CLR  SCL
         LCALL  TMP
       RET
NOACKI2C:SETB   SCL             ;ROTINA PARA NOACKNOLEDGE I2C
         LCALL  TMP
       MOV  C,SDA
       JNC  ERRO2
       CLR  SCL
         LCALL   TMP
       RET
ERRO1:   MOV    DPTR,#LEDS_ADR   ;ROTINA PARA INDICACAO DE ERRO I2C TX
       MOV  A,#05H
       MOVX @DPTR,A
         MOV    A, #'E'
         LCALL  SER_CHAR
         LCALL  MAIN
         SJMP     $
ERRO2:   MOV    DPTR,#LEDS_ADR   ;ROTINA PARA INDICACAO DE ERRO I2C RX
       MOV  A,#03H
       MOVX @DPTR,A
         MOV    A, #'e'
         LCALL  SER_CHAR
         SJMP     $
TEMPO:   MOV    R7,#1       ;ROTINA DE DELAY
TP1:     MOV    R6,#0
TP2:     MOV    R5,#0
TP3:     DJNZ   R5,TP3
       DJNZ R6,TP2
       DJNZ R7,TP1
       RET
TMP:     MOV    R5,#20
TP4:     DJNZ   R5,TP4
       RET
BWRITE:                  ;ROTINA DE ESCRITA DE BYTE I2C
         LCALL  STARTI2C
         LCALL  SL_ADR_WR
         LCALL  ACKI2C
         MOV    A,R7
         LCALL  VAI_BYTE
         LCALL  ACKI2C
         MOV    A,R6
         LCALL  VAI_BYTE
         LCALL  ACKI2C
         LCALL  STOPI2C
         RET
BWRITE2:                 ;ROTINA DE ESCRITA DE BYTE I2C (ENDERECO 16
BITS)
         LCALL   STARTI2C
         LCALL   SL_ADR_WR
         LCALL   ACKI2C
         MOV     A,R7
         LCALL   VAI_BYTE
```

```
         LCALL   ACKI2C
         MOV     A,R6
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         MOV     A,R3
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         LCALL   STOPI2C
         RET
BREAD:                      ;ROTINA DE LEITURA DE BYTE I2C
         LCALL   STARTI2C
         LCALL   SL_ADR_WR
         LCALL   ACKI2C
         MOV     A,R7
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         LCALL   STARTI2C
         LCALL   SL_ADR_RD
         LCALL   ACKI2C
         LCALL   VEM_BYTE
         LCALL   NOACKI2C
         LCALL   STOPI2C
         RET
BREAD2:                      ;ROTINA DE LEITURA DE BYTE I2C (ENDERECO 16
BITS)
         LCALL   STARTI2C
         LCALL   SL_ADR_WR
         LCALL   ACKI2C
         MOV     A,R7
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         MOV     A,R6
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         LCALL   STOPI2C
         LCALL   STARTI2C
         LCALL   SL_ADR_RD
         LCALL   ACKI2C
         LCALL   VEM_BYTE
         LCALL   NOACKI2C
         LCALL   STOPI2C
         RET
DWRITE:                      ;ROTINA DE ESCRITA DE DADOS COM BUFFER INTERNO
I2C
         LCALL   STARTI2C
         LCALL   SL_ADR_WR
         LCALL   ACKI2C
         MOV     R1,#XMT_DATI
         MOV     A,@R1
         MOV     R0,A
LACOTX:  INC     R1
         MOV     A,@R1
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         DJNZ    R0,LACOTX
         LCALL   STOPI2C
         RET
DWRITE2:                      ;ROTINA DE ESCRITA DE DADOS (ATE 256B) I2C A
PARTIR DE BUFFER EXTERNO
         LCALL   STARTI2C
         LCALL   SL_ADR_WR
```

```
         LCALL   ACKI2C
         MOV     DPTR,#XMT_DAT
         MOVX    A,@DPTR
         MOV     R0,A
LACOTX2: INC     DPTR
         MOVX    A,@DPTR
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         DJNZ    R0,LACOTX2
         LCALL   STOPI2C
         RET
DWRITE3:                          ;ROTINA DE ESCRITA DE DADOS (ATE 2KB)
I2C A PARTIR DE BUFFER EXTERNO
         LCALL   STARTI2C         ;USADA PARA MANDAR A PAGINA
         LCALL   SL_ADR_WR
         LCALL   ACKI2C
         MOV     A,#40H
         ADD     A,TWMSB
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         MOV     A,TWLSB
         LCALL   VAI_BYTE
         LCALL   CKI2C
         MOV     DPH1,#1          ;CARREGAR TAMANHO DO BUFFER
         MOV     DPL1,#0A8H       ;MSB E LSB
         MOV     DPH1,#2          ;CARREGAR TAMANHO DO BUFFER
         MOV     DPL1,#0F7H       ;MSB E LSB
         MOV     DPL1,#0F8H       ;MSB E LSB
         MOV     DPTR,#XMT_DAT
LACOTX32:
         CLR     C               ;DECREMENTO 16BITS
         MOV     A,DPL1
         SUBB    A,#1
         MOV     DPL1,A
         MOV     A,DPH1
         SUBB    A,#0
         MOV     DPH1,A
         MOVX    A,@DPTR
         INC     DPTR
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         DJNZ    R0,LACOTX32
         MOV     A,DPL1
         JNZ     LACOTX32        ;Compare # of bytes remaining
         MOV     A,DPH1
         JNZ     LACOTX32
         LCALL   STOPI2C
         RET
DREADBAD:                        ;ROTINA DE LEITURA DE DADOS I2C COM BUFFER
INTERNO
                                  ;ROTINA BURRA: IMPLEMENTADA SEM DUMP ->
BAD!
         MOV     R1,#XMT_DAT     ;NA PRIMEIRA POSICAO DE XMT_DAT DEVE ESTAR
A QUANTIDADE DE BYTES
         MOV     A,@R1           ;NA SEGUNDA, O ENDERECO INICIAL
         MOV     R0,A            ;ENDERECOS DE 8 BITS
         INC     R1
         MOV     A,@R1
         MOV     R7,A
         DEC     R0              ;ENDERECO EH DADO PARA O BUFFER MAIS NAUM
EH DADO A SER RECEBIDO
```

```
LACORX:  LCALL   STARTI2C
         LCALL   SL_ADR_WR
         LCALL   ACKI2C
         MOV     A,R7
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         LCALL   STARTI2C
         LCALL   SL_ADR_RD
         LCALL   ACKI2C
         LCALL   VEM_BYTE
         INC     R7
         INC     R1
         MOV     @R1,A
         LCALL   NOACKI2C
         LCALL   STOPI2C
         DJNZ    R0,LACORX
         RET

DREADGOOD:                       ;ROTINA DE LEITURA DE DADOS (ATE 2KB) I2C
                                 ;IMPLEMENTADA COM DUMP -> GOOD!
         MOV     DPTR,#RCV_DAT   ;NA PRIMEIRA POSICAO DE RCV_DAT DEVE ESTAR
A QUANTIDADE DE BYTES
         MOVX    A,@DPTR         ;NA SEGUNDA, O ENDERECO INICIAL
         MOV     R0,A
         INC     DPTR
         MOVX    A,@DPTR
         MOV     R7,A

;++++++++++++++++++++++++++++++++++
                DEC             R0      ;ENDERECO EH DADO PARA O BUFFER
MAIS NAUM EH DADO A SER RECEBIDO
;++++++++++++++++++++++++++++++++++

         LCALL   STARTI2C
         LCALL   SL_ADR_WR
         LCALL   ACKI2C
         MOV     A,R7
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         LCALL   STARTI2C
         LCALL   SL_ADR_RD
         PUSH    ACC
         MOV     A,#'I'
         LCALL   SER_CHAR
         POP     ACC
         LCALL   ACKI2C
         DEC     R0              ;CONDICAO DE NOACKNOLEDGE FORA DE LACO:
FORCAR R0 PARA FAZER ULTIMA ITERACAO FORA DO LACO
LACORX2: LCALL   VEM_BYTE
         LCALL   ACKI2C
         INC     DPTR
         MOVX    @DPTR,A
         DJNZ    R0,LACORX2
         LCALL   VEM_BYTE
         INC     DPTR
         MOVX    @DPTR,A
         LCALL   NOACKI2C
         LCALL   STOPI2C
         RET
```

```
DREADBAD2:                      ;ROTINA DE LEITURA DE DADOS (ATE
??B/MAX216B) I2C
                                ;IMPLEMENTADA SEM DUMP -> BAD2!
         MOV     DPTR,#RCV_DAT  ;NA PRIMEIRA POSICAO DE RCV_DAT DEVE ESTAR
A QUANTIDADE DE BYTES
         MOVX    A,@DPTR        ;NA SEGUNDA, O ENDERECO INICIAL
         MOV     R0,A
         INC     DPTR
         MOVX    A,@DPTR
         MOV     R7,A
         INC     DPTR
         MOVX    A,@DPTR
         MOV     R6,A
         DEC      R0            ;ENDERECO EH DADO PARA O BUFFER MAIS NAUM
EH DADO A SER RECEBIDO

LACORX3: LCALL   STARTI2C
         LCALL   SL_ADR_WR
         LCALL   ACKI2C
         MOV     A,R7
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         MOV     A,R6
         LCALL   VAI_BYTE
         LCALL   ACKI2C
         LCALL   STARTI2C
         LCALL   SL_ADR_RD
         LCALL   ACKI2C
         LCALL   VEM_BYTE
         INC     R6
         INC     DPTR
         MOVX    @DPTR,A
         LCALL   NOACKI2C
         LCALL   STOPI2C
         DJNZ    R0,LACORX3
         RET
RDEBUG:
         MOV     A,#LF
         LCALL   SER_CHAR
         MOV     R1,#XMT_DAT
         MOV     A,@R1
         MOV     R0,A
         INC     R1
         DEC     R0
LPSTX:   INC     R1
         MOV     A,@R1
         LCALL   SER_W8
         MOV     A,#' '
         LCALL   SER_CHAR
         DJNZ    R0,LPSTX
         MOV     A,#LF
         LCALL   SER_CHAR
         RET
RDEBUG2:
         MOV     A,#LF
         LCALL   SER_CHAR
         MOV     R1,#RCV_DAT
         MOV     A,@R1
         MOV     R0,A
         INC     R1
         DEC     R0
```

```
LPSTX2:  INC     R1
         MOV     A,@R1
         LCALL   SER_W8
         MOV     A,#' '
         LCALL   SER_CHAR
         DJNZ    R0,LPSTX
         MOV     A,#LF
         LCALL   SER_CHAR
         RET

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;
;     TRANSFER Subroutine
;
;     This subroutine copies data from the EPROM referenced by DPTR into a
;     Buffer referenced by R1.
;
;     DPTR = String stored into PROGRAM MEMORY
;     R1 = Buffer in which data shall be stored
;
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

TRANSFER:
         CLR      A               ;Clears ACC
         MOVC     A, @A+DPTR      ;Moves contents of DPTR into A
         MOV      @R1, A          ;Copies A into Buffer
         MOV      R0, A           ;Copies A into R0 (# of bytes)
         INC      R1              ;Next address
         INC      DPTR            ;Next location
         CLR      A               ;Clears A
NEXT:
         MOVC     A, @A+DPTR      ;Moves contents of DPTR into A
         MOV      @R1, A          ;Copies A  into Buffer
         INC      R1              ;Next address
         INC      DPTR            ;Next location
         CLR      A               ;Clears A
         DJNZ     R0, NEXT        ;Compare # of bytes remaining
         RET                      ;If all bytes copied, return

;TRANSFERENCIA PARA MEMORIA DE DADOS EXTERNA (255B): OBSERVE SINTAXE DA
STRING

TRANSFER2:
         CLR      A               ;Clears ACC
         MOVC     A,@A+DPTR       ;POE NO ACC
         PUSH     DPH
         PUSH     DPL             ;GUARDA DPTR
         MOV      DPH,DPH0        ;PEGA ENDERECO DO BUFFER
         MOV      DPL,DPL0
         MOVX     @DPTR,A         ;COLOCA DADO NO BUFFER
         MOV      R0,A
         POP      DPL
         POP      DPH             ;RECUPERA DPTR
         INC      DPTR
                                  ;SOMA DE 16 BITS
         CLR      C
         MOV      A,#1
         ADD      A,DPL0
         MOV      DPL0,A
         CLR      A
         ADDC     A,DPH0
```

```
          MOV       DPH0,A
NEXT2:
          MOVC      A,@A+DPTR         ;POE NO ACC
          PUSH      DPH
          PUSH      DPL               ;GUARDA DPTR
          MOV       DPH,DPH0          ;PEGA ENDERECO DO BUFFER
          MOV       DPL,DPL0
          MOVX      @DPTR,A           ;COLOCA DADO NO BUFFER
          POP       DPL
          POP       DPH               ;RECUPERA DPTR
          INC       DPTR              ;PROXIMA POSICAO DA STRING
                                      ;SOMA DE 16 BITS
          CLR       C
          MOV       A,#1
          ADD       A,DPL0
          MOV       DPL0,A
          CLR       A
          ADDC      A,DPH0
          MOV       DPH0,A
          DJNZ      R0, NEXT2         ;Compare # of bytes remaining

;TRANSFERENCIA PARA MEMORIA DE DADOS EXTERNA (2KB)

TRANSFER3:
          MOV       DPH1,#3H
          MOV       DPL1,#15H
          CLR       A                  ;Clears ACC
          MOVC      A,@A+DPTR          ;POE NO ACC
          PUSH      DPH
          PUSH      DPL                ;GUARDA DPTR
          MOV       DPH,DPH0           ;PEGA ENDERECO DO BUFFER
          MOV       DPL,DPL0
          MOVX      @DPTR,A            ;COLOCA DADO NO BUFFER
          MOV       R0,A
          POP       DPL
          POP       DPH                ;RECUPERA DPTR
          INC       DPTR
                                       ;SOMA DE 16 BITS
          CLR       C
          MOV       A,#1
          ADD       A,DPL0
          MOV       DPL0,A
          CLR       A
          ADDC      A,DPH0
          MOV       DPH0,A
NEXT3:
          CLR       C
          MOV       A,DPL1
          SUBB      A,#1
          MOV       DPL1,A
          MOV       A,DPH1
          SUBB      A,#0
          MOV       DPH1,A
          CLR       A
          MOVC      A,@A+DPTR           ;POE NO ACC
          PUSH      DPH
          PUSH      DPL                 ;GUARDA DPTR
          MOV       DPH,DPH0            ;PEGA ENDERECO DO BUFFER
          MOV       DPL,DPL0
          MOVX      @DPTR,A             ;COLOCA DADO NO BUFFER
          POP       DPL
```

```
        POP      DPH                 ;RECUPERA DPTR
        INC      DPTR                ;PROXIMA POSICAO DA STRING
                                     ;SOMA DE 16 BITS
        CLR      C
        MOV      A,#1
        ADD      A,DPL0
        MOV      DPL0,A
        CLR      A
        ADDC     A,DPH0
        MOV      DPH0,A
        MOV      A,DPL1
        JNZ      NEXT3                ;Compare # of bytes remaining
        MOV      A,DPH1
        JNZ      NEXT3
        RET                           ;If all bytes copied, return
```

APÊNDICE C– PROGRAMAS PARA PLACA CONTROLADORA [8051]

Código 05 – Rotina de Boot da Placa "Penta-Controladora"

```
;B_BOOTT.ASM    V1.0
;
; Rotina para fazer o BOOT da PLACA "PENTA-CONTROLADORA"
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                     MODO_BOOT                                       ;
; Programa para receber arquivo INTEL.HEX e gravar na memoria (RAM1) ;
; R6=Contador dos bytes em cada Record                                ;
; R5=Fazer o Check Sum dos Records                                    ;
;
;              LEDS REDONDOS (so durante boot serial)
;  (P --> LED piscando, 1 --> LED aceso, 0 --> LED apagado)
;  VM      AM   VD     indicacao
;  0       P    0  ==> pronto para receber boot serial
;  0       P    P  ==> executando boot serial
;  0       0    P  ==> terminado boot serial, pronto para executar
;  P       0    0  ==> ERRO 0, nao recebeu ":" ou 0DH ou 0AH
;  P       0    P  ==> ERRO 1, chegou tipo diferente de 0 ou 1
;  P       P    0  ==> ERRO 2, check sum nao conferiu
;  P       P    P  ==> ERRO 3, encheu buffer circular
;  1       0    0  ==> ERRO 4, nao pode converter de asc para hexa
;  1       1    0  ==> ERRO 4, nao pode converter de asc para hexa
;
; Faz as inicializacoes                                               ;
;                                                                     ;
; Colocar TIMER1 no modo 2 para gerar baud rate                       ;
;         TIMER0 NO MODO 1 para gerar base de tempo (5 ms)            ;
; TMOD: GATE  C/*T  M1  M0    GATE  C/*T  M1 M0                       ;
;         0     0    1    0     0     0    0   1                      ;
;                                                                     ;
; Ligar os dois timers                                                ;
; TCON: TF1  TR1  TF0  TR0  IE1  IT1  IE0  IT0                        ;
;        0    1    0    1    0    0    0    0                         ;
;                                                                     ;
; Porta serial para 8 bits de dados, sem paridade, 1 start, 2 stop   ;
; SCON: SM0  SM1  SM2  REN  TB8  RB8  TI  RI                          ;
;        1    1    0    0    1    0   0    0                          ;
```

```
;                                                                              ;
; Cristal = 11.059.200 Hz                                                      ;
; Para baud rate de 9600 (com SMOD=0 de PCON)                                  ;
;         11.059.200                                                           ;
;   n = -------------- = 3  logo 256-3=253  ----> TH1=TL1=253                  ;
;         12*32*9600                                                           ;
;                                                                              ;
; Para base de tempo de 5 ms (200 Hz)                                          ;
;      11.059.200                                                              ;
; n = ------------- = 4.608 ---> recarga = 65536 - 4.608 = 60.928              ;
;        12*200                                                                ;
;                                                                              ;
; Sera usado um byte enderecavel bit a bit como contador                       ;
; bits:    7      6      5     4     3     2     1     0                       ;
;        640ms  320ms  160ms  80ms  40ms  20ms  10ms  5ms                     ;
; Ao zerar esse byte, todas flags de tempo serao zeradas                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MODO_BOOT:
MOV   SP,#SP_INIC                   ;INCIALIZAR PILHA
MOV   TMOD,#21H                     ;TIMER1 MODO2, TIMER0 MODO 1
MOV   TH1,#BAUD                     ;BAUD = 9600
MOV   TL1,#BAUD
MOV   TL0,#LOW  REC5MS  ;GERAR OS 5 mSEG
MOV   TH0,#HIGH REC5MS  ;GERAR OS 5 mSEG
MOV   SCON,#0C8H        ;8 DADOS,1 START,2 STOP
MOV   TCON,#50H                     ;TR0=1 e TR1=1
MOV   CT_PISCA,#REC_PISCA           ;DEFINE FREQ PISCA
CLR   A
MOV   P_LEDS,A          ;GARANTIR LEDS APAGADOS
MOV   AC_LEDS,A
MOV   CONTL,A           ;ZERAR CONTADOR DE BYTES
MOV   CONTM,A
MOV   CONTH,A
MOV   IE,A
SETB  ET0   ;HABILITAR INTERRUPCAO TIMER0
SETB  EA
MOV   DPTR,#MSG2
LCALL LCD_STRC          ;IMPRIMIR MSG 2 NO LCD
MOV   DPTR,#MSG1
LCALL SER_STRC          ;ENVIAR A MENSAGEM 1
CLR   FIM
MOV   PIN,#FILA         ;INICIALIZAR PONTEIROS
MOV   POUT,#(FILA+TAM-1)
LCALL IMP_CONT          ;IMPRIMIR CONTADOR DE BYTES
SETB  P_AM
SETB  PS                ;PRIORIDADE PARA SERIAL
SETB  IT0
SETB  EX0
SETB  ES
SETB  REN   ;HABILITA REC SERIAL

;
; Aqui inicia a recepcao INTEL.HEX
;

INTEL_HEX:
LCALL  VEM_BYTE        ;ESPERAR BYTE
TP6:
SETB   P_VD            ;PISCAR VERDE
XRL    A,#':'          ;CHEGOU ':'
JZ     LB2
```

```
LJMP   ERRO0           ;FALTOU ':'
LB2:
MOV    R5,#0           ;R5=CHECK SUM
LCALL  VEM_BIN_XEQ
MOV    R6,A            ;R6 CONTA OS BYTES
LCALL  VEM_BIN_XEQ     ;ADR HIGH
MOV    DPH,A
LCALL  VEM_BIN_XEQ     ;ADR LOW
MOV    DPL,A           ;DPTR=ADR DE CARGA
LCALL  VEM_BIN_XEQ     ;RECEBER TIPO
JZ     REC_DADO        ;TIPO=0
CJNE   A,#1,LB4
SETB   FIM             ;AVISAR CHEGOU ULTIMO RECORD
SJMP   XEQ_SUM
LB4:   LJMP   ERRO1    ;TIPO DESCONHECIDO
REC_DADO:
LCALL  VEM_BIN_XEQ     ;RECEBER DADO A SER CARREGADO
MOVX   @DPTR,A         ;ESCREVER DADO NA RAM
INC    DPTR
DJNZ   R6,REC_DADO     ;FOI O ULTIMO ?
XEQ_SUM:
LCALL  VEM_BIN         ;RECEBER O CHECK SUM
ADD    A,R5 ;CONFERIR O CHECK SUM
JZ     LB5
LJMP   ERRO2           ;NAO CONFERIU CHECK SUM
LB5:   LCALL  VEM_BYTE ;RECEBER 0DH
SWAP   A
MOV    B,A
LCALL  VEM_BYTE        ;RECEBER 0AH
ADD    A,B             ;MONTAR 0DAH
XRL    A,#CRLF
JZ     LB6
LJMP   ERRO0
LB6:   JNB    FIM,INTEL_HEX
CLR    REN
CLR    P_AM            ;LED AMARELO NAO PISCA
CLR    AC_AM
MOV    A,#0
LCALL  LCD_DDRAM
MOV    DPTR,#MSG3
LCALL  LCD_STRC
CALL   IMP_CONT
SJMP   $            ;TERMINOU

;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          VEM_BIN_XEQ                                        ;
; Retirar 2 bytes da fila e montar um byte                                    ;
; Recebe : nada                                                               ;
; Retorna: Acc=byte montado e ja faz o check sum em R5                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
VEM_BIN_XEQ:    LCALL           TIRA
      JNC   VEM_BIN_XEQ ;PEGAR O MSNibble
      MOV   B,A   ;GUARDAR O MSNibble
VEM1:           LCALL           TIRA            ;PEGAR O LSNibble
      JNC   VEM1
              LCALL           ASC_HEXT         ;CONVERTER PARA HEXA
      MOV   B,A
      ADD   A,R5  ;CHECK SUM
      MOV   R5,A
```

```
      MOV   A,B
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                         VEM_BIN                                     ;
; Retirar 2 bytes da fila e montar um byte                            ;
; Recebe : nada                                                       ;
; Retorna: Acc=byte montado                                           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
VEM_BIN:        LCALL           TIRA
      JNC   VEM_BIN     ;PEGAR O MSNibble
      MOV   B,A   ;GUARDAR O MSNibble
VEM2:           LCALL           TIRA            ;PEGAR O LSNibble
      JNC   VEM2
      LCALL          ASC_HEXT         ;CONVERTER PARA HEXA
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                         VEM_BYTE                                    ;
; Retirar 1 byte da fila                                              ;
; Recebe : nada                                                       ;
; Retorna: Acc=byte                                                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
VEM_BYTE:       LCALL           TIRA
      JNC   VEM_BYTE    ;ESPERAR CHEGAR DADO
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           ASC_HEXT (com teste)                      ;
; Converter dois caracteres HEXA ASCII no hexadecimal binario         ;
; Recebe  : Acc=LSN  B=MSN                                            ;
; Retorna : Acc=byte                                                  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ASC_HEXT:       LCALL           ASC_NIBT         ;CONVERTER LSNibble
      XCH   A,B
                LCALL           ASC_NIBT         ;CONVERTER MSNibble
      SWAP  A
      ADD   A,B
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           ASC_NIBT                                  ;
; Converter um nibble que esta em ASCII no binario (Faz teste)        ;
; Recebe  : Acc=nibble(ASCII)                                         ;
; Retorna : Acc=nibble(binario)                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ASC_NIBT:       CLR             C
      SUBB  A,#30H      ;A-30H SE CY=1 --> A<30H
      JC    ERRO  ;      SE CY=0 --> A>=30H
      PUSH  ACC
      SUBB  A,#10 ;A-10 SE CY=1 --> A<10
      JNC   ASCNIB1     ;     SE CY=0 --> A>=10
      POP   ACC
      RET
ASCNIB1:        POP             ACC
      SUBB  A,#17H      ;A-17H SE CY=1 --> A<17H
      JNC   ERRO  ;      SE CY=0 --> A>=17H(ERRO)
```

```
      ADD   A,#10H
      RET
ERRO:           LJMP            ERRO4
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                               POE                                  ;
; Rotina para colocar um byte na fila circular                       ;
; Recebe Acc=byte                                                    ;
; Retorna: CY=1 se conseguiu colocar na fila                         ;
;          CY=0 se nao foi possivel colocar na fila (cheia)          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
POE:            PUSH            ACC
      MOV   A,PIN
      CJNE  A,POUT,POE1
      POP   ACC   ;PIN=POUT --> FILA CHEIA
      CLR   C
      RET
POE1:           MOV             R0,PIN          ;TEM ESPACO
      POP   ACC
      MOV   @R0,A ;COLOCAR NA FILA
      INC   R0
      CJNE  R0,#(FILA+TAM),POE2
      MOV   R0,#FILA    ;RESTAURAR PARA INICIO
POE2:           MOV             PIN,R0
      SETB  C
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                             TIRA                                   ;
; Rotina para tirar um byte na fila circular                         ;
; Recebe : nada                                                      ;
; Retorna: CY=1 e Acc=byte se conseguiu retirar da fila              ;
;          CY=0 se a fila estava vazia                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
TIRA:           JNB             F160MS,TIRA3
                MOV             CDT,#0          ;ZERAR CONTADOR DE TEMPO
                MOV             A,#40H
                LCALL           LCD_DDRAM       ;CURSOR POSICAO 40H
                MOV             A,CONTH
                LCALL           LCD_BCD2
                MOV             A,CONTM
                LCALL           LCD_BCD2
                MOV             A,CONTL
                LCALL           LCD_BCD2
TIRA3:          MOV             A,POUT
INC   A
CJNE  A,#(FILA+TAM),TIRA1     ;FIM DA FILA
MOV   A,#FILA     ;RESTAURAR INICIO
TIRA1:          CJNE            A,PIN,TIRA2     ;POUT=PIN ?
      CLR   C     ;FILA VAZIA
      RET
TIRA2:          MOV             R0,A
      MOV   A,@R0
      MOV   POUT,R0
                SETB            C
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                            IMP_CONT                                ;
```

```
; Imprimir o contador de bytes                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
IMP_CONT:       MOV             A,#40H
                LCALL           LCD_DDRAM       ;CURSOR POSICAO 40H
                MOV             A,CONTH
                LCALL           LCD_BCD2
                MOV             A,CONTM
                LCALL           LCD_BCD2
                MOV             A,CONTL
                LCALL           LCD_BCD2
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                     ROTINAS INTERRUPCAO                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           SERIAL - Interrupcao                        ;
; Chama rotina POE e coloca byte recebido na fila                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
SERIAL:         PUSH            PSW             ;GUARDAR CONTEXTO
      PUSH  ACC
      PUSH  0     ;R0 E USADO ROTINA POE
      CLR   RI
                MOV             A,CONTL         ;INC CONTADOR (BCD) DE
BYTES
                ADD             A,#1            ; [CONTH:CONTM:CONTL]
                DA              A               ;
                MOV             CONTL,A
                MOV             A,CONTM
                ADDC            A,#0
                DA              A
                MOV             CONTM,A
                MOV             A,CONTH
                ADDC            A,#0
                DA              A
                MOV             CONTH,A
                MOV     A,SBUF      ;LER DADO RECEBIDO
                LCALL           POE             ;POR NA FILA
      JC    SER3
                LJMP            ERRO3           ;FILA CHEIA
SER3:           POP             0               ;RESTAURAR R0
      POP   ACC   ;RESTAURAR CONTEXTO
      POP   PSW
                RETI
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           TIM0 - Interrupcao                          ;
; Tratar interrupcao do TIMER0, faz os leds piscarem                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
TIM0:           MOV             TL0,#LOW  REC5MS
      MOV   TH0,#HIGH REC5MS
      PUSH  ACC
      PUSH  PSW
                PUSH            DPH
      INC   CDT               ;INC CONTADOR DE TEMPO
      DJNZ  CT_PISCA,TIM01
      MOV   CT_PISCA,#REC_PISCA
```

```
      MOV   A,P_LEDS
      ANL   A,#0FH            ;SEPARAR OS 4 LEDS
      XRL   A,AC_LEDS
      MOV   AC_LEDS,A
      XRL   A,#0FH            ;COMPLENTAR BIT DOS LEDS
                MOV             DPH,#END_LEDS_H
      MOVX  @DPTR,A
TIM01:          POP             DPH
      POP   PSW
      POP   ACC
      RETI

;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          ERRO 0                                           ;
; Nao recebeu ":" ou 0DH ou 0AH                                             ;
; VM=P  AM=0  VD=0                                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ERRO0:          CLR             REN
CLR   P_VD
                CLR             AC_VD
                CLR     P_AM
                CLR             AC_AM
SETB  P_VM
                MOV             A,#0
                LCALL           LCD_DDRAM
                MOV             DPTR,#MSG_ERRO_0
                LCALL           LCD_STRC
                MOV             A,#40H
                LCALL           LCD_DDRAM
                MOV             DPTR,#MSG_ERRO_0A
                LCALL           LCD_STRC
      SJMP  $
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          ERRO 1                                           ;
; Recebeu um tipo diferente de 0 ou 1                                       ;
; VM=P  AM=0  VD=P                                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ERRO1:          CLR             REN
      SETB  P_VD
      CLR   P_AM
                CLR             AC_AM
      SETB  P_VM
                MOV             A,#0
                LCALL           LCD_DDRAM
                MOV             DPTR,#MSG_ERRO_1
                LCALL           LCD_STRC
                MOV             A,#40H
                LCALL           LCD_DDRAM
                MOV             DPTR,#MSG_ERRO_1A
                LCALL           LCD_STRC
      SJMP  $
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          ERRO 2                                           ;
; Check sum nao conferiu                                                    ;
; VM=P  AM=P  VD=0                                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;
ERRO2:          CLR             REN
      CLR   P_VD
                CLR             AC_VD
      SETB  P_AM
      SETB  P_VM
                MOV             A,#0
                LCALL           LCD_DDRAM
                MOV             DPTR,#MSG_ERRO_2
                LCALL           LCD_STRC
                MOV             A,#40H
                LCALL           LCD_DDRAM
                MOV             DPTR,#MSG_ERRO_2A
                LCALL           LCD_STRC
      SJMP  $
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                         ERRO 3                                              ;
; Fila circular estourou                                                      ;
; VM=P  AM=P  VD=P                                                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ERRO3:          CLR             REN
      SETB  P_VD
      SETB  P_AM
      SETB  P_VM
                MOV             A,#0
                LCALL           LCD_DDRAM
                MOV             DPTR,#MSG_ERRO_3
                LCALL           LCD_STRC
                MOV             A,#40H
                LCALL           LCD_DDRAM
                MOV             DPTR,#MSG_ERRO_3A
                LCALL           LCD_STRC
      SJMP  $
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                         ERRO 4                                              ;
; Nao conseguiu converter de ASCII para binario (ASC_HEXT)                    ;
; VM=1  AM=0  VD=0                                                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ERRO4:          CLR             REN
      CLR   P_VD
                CLR             AC_VD
      CLR   P_AM
                CLR             AC_AM
      CLR   P_VM
                SETB            AC_VM
                MOV             A,#0
                LCALL           LCD_DDRAM
                MOV             DPTR,#MSG_ERRO_4
                LCALL           LCD_STRC
                MOV             A,#40H
                LCALL           LCD_DDRAM
                MOV             DPTR,#MSG_ERRO_4A
                LCALL           LCD_STRC
      SJMP  $
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                         ERRO 5                                              ;
```

```
; Recebeu um comando diferente de CMD_RECORD                          ;
; VM=1  AM=1  VD=0                                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ERRO5:          CLR             REN
      CLR   P_VD
                CLR             AC_VD
      CLR   P_AM
                SETB            AC_AM
      CLR   P_VM
                SETB            AC_VM
                MOV             A,#0
                LCALL           LCD_DDRAM
                MOV             DPTR,#MSG_ERRO_5
                LCALL           LCD_STRC
                MOV             A,#40H
                LCALL           LCD_DDRAM
                MOV             DPTR,#MSG_ERRO_5A
                LCALL           LCD_STRC
      SJMP  $
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Mensagens que serao usadas pelo programa BOOT                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
MSG1:           DB              'PRONTO PARA BOOT SERIAL ...',CR,LF,0
;SERIAL
MSG2:           DB              'BOOT...',0
MSG3:           DB              'PRONTO!',0
MSG_ERRO_0:     DB              'ERRO 0:',0
MSG_ERRO_0A:    DB              'CR LF : FALTANDO',0
MSG_ERRO_1:     DB              'ERRO 1:',0
MSG_ERRO_1A:    DB              '? TIPO ? DESCON.',0
MSG_ERRO_2:     DB              'ERRO 2:',0
MSG_ERRO_2A:    DB              'CHECK SUM ERRADO',0
MSG_ERRO_3:     DB              'ERRO 3:',0
MSG_ERRO_3A:    DB              'FILA CHEIA',0
MSG_ERRO_4:     DB              'ERRO 4:',0
MSG_ERRO_4A:    DB              'ASC->HEX ERRADO',0
MSG_ERRO_5:     DB              'ERRO 5:',0
MSG_ERRO_5A:    DB              '? CMD DESCONHEC.',0
```

Código 06 – Sub Rotina da Rotina de BOOT

```
;B_CABEC.ASM                    V1.0
;
; É usado pelo BOOTT.ASM da PLACA "PENTA-CONTROLADORA"
;
; Endereços para os Vetores de Interrupção
RESET           EQU             0
EXTI0           EQU             3
TIMER0          EQU             0BH
EXTI1           EQU             13H
TIMER1          EQU             1BH
SINT            EQU             23H

; Constantes
```

```
TRUE  EQU   1
FALSE EQU   0
CR    EQU   0DH
LF    EQU   0AH
CRLF        EQU            0DAH
V_32K       EQU            32768
V_8K        EQU            8192
REC5MS      EQU   60928 ;RECARGA GERAR 5mSEG (TIMER 0)
P10HZ EQU   10    ;PISCAR EM 10 Hz
P5HZ  EQU   20    ;PISCAR EM  5 Hz
P1HZ  EQU   100   ;PISCAR EM  1 HZ
REC_PISCA   EQU            P10HZ      ;RECARGA PARA CONTADOR PISCA
SP_INIC     EQU            5FH        ;PILHA COM 32 BYTES (60->7F)
B9600       EQU            253        ;BAUD RATE 9.600
B14400      EQU            254        ;BAUD RATE 14.400
B28800      EQU            255        ;BAUD RATE 28.800
BAUD        EQU            B28800     ;DEFINIR BR A SER USADO
LCD_1X16    EQU            10H        ;1 LINHA  X 16 COLUNAS
LCD_2X8     EQU            88H        ;2 LINHAS X  8 COLUNAS
LCD_2X16    EQU   90H   ;2 LINHAS X 16 COLUNAS
LCD_1X20    EQU            14H        ;1 LINHA  X 20 COLUNAS
LCD_2X20    EQU            94H        ;2 LINHAS X 20 COLUNAS
LCD_1X40    EQU            28H        ;1 LINHA  X 40 COLUNAS
LCD_2X40    EQU            0A8H       ;2 LINHAS X 40 COLUNAS
TAM         EQU            20H        ;TAMANHO DA FILA CIRCULAR
REBOTE      EQU            100        ;ATRASO PARA EVITAR REBOTES
LCD_L1_INI  EQU            0          ;LCD INICIO LINHA 1
LCD_L1_FIM  EQU            28H        ;LCD FIM    LINHA 1
LCD_L2_INI  EQU            40H        ;LCD INICIO LINHA 2
LCD_L2_FIM  EQU            68H         ;LCD FIM    LINHA 2

; Enderecos de I/O em 16 bits
END_LEDS    EQU   8000H ;LATCH COM OS LEDS
AP_IV01     EQU   8200H ;APAGA PEDIDOS INTERRUPCAO IV0 E IV1
AP_IV0      EQU   8201H           ;APAGA PEDIDO INTERRUPCAO IV0
AP_IV1      EQU   8202H           ;APAGA PEDIDO INTERRUPCAO IV1
END_BUSZW   EQU   8300H ;ENDERECO DOS BARRAMENTOS ZD E WD
AP_IKBD     EQU   8400H ;APAGA PEDIDO INTERRUPCAO TECLADO
END_DA      EQU   8501H           ;ENDERECO DAC
END_AD      EQU   8502H           ;ENDERECO ADC
END_9603    EQU   8600H ;ENDERECO DO CONTROLADOR USB
PED_INT     EQU   8700H ;LER OS PEDIDOS DE INTERRUPCAO

; Enderecos de I/O, byte mais significativo
END_LEDS_H  EQU    80H             ;LATCH COM OS LEDS
END_LCD_H   EQU    81H             ;BUS XD QUE SERVE AO LCD
AP_IKBDH    EQU    84H             ;APAGA PEDIDO INTERRUPCAO TECLADO
PED_INTH    EQU    87H             ;LER PEDIDOS DE INTERRUPCAO
; Enderecos da RAM INTERNA
AC_LEDS     EQU   20H   ;ESTADO DO LATCH COM LEDS (*CS0)
P_LEDS      EQU   21H   ;INDICAR QUAL LED DEVE PISCAR
CDT   EQU   22H   ;CONTADOR INCR A CADA 5 MSEG
LCD_CONF    EQU   23H   ;GUARDAR CONFIGURACOES
CT_PISCA    EQU   30H   ;CONTADOR PARA PISCAR LEDS
LCD_DIM     EQU   31H   ;DIMENSOES DO DISPLAY LCCC CCCC
PIN         EQU   32H             ;PONTEIRO PARA COLOCAR NA FILA
POUT        EQU   33H             ;PONTEIRO PARA RETIRAR DA FILA
CONTL       EQU   34H             ;LSD CONTADOR DE BYTES (BCD)
CONTM       EQU   35H             ;[CONTH:CONTM:CONTL]   (BCD)
CONTH       EQU   36H             ;MSD CONTADOR DE BYTES (BCD)
OPTESTE     EQU   37H             ;OPCAO PARA MODO TESTE
```

```
FILA        EQU   40H             ;INICIO FILA CIRCULAR

; Declaracao de BITS
;;;;;;;;; Acedender e Piscar os LEDS
AC_VD BIT   AC_LEDS.0
AC_AM BIT   AC_LEDS.1
AC_VM BIT   AC_LEDS.2
AC_IV BIT   AC_LEDS.3   ;LED INFRAVERMELHO
P_VD  BIT   P_LEDS.0
P_AM  BIT   P_LEDS.1
P_VM  BIT   P_LEDS.2
P_IV  BIT   P_LEDS.3    ;LED INFRAVERMELHO
;;;;;;;;; Flags de tempo
F5MS  BIT   CDT.0 ;PARA USAR O FLAGS
F10MS BIT   CDT.1 ;ZERAR O CDT E
F20MS BIT   CDT.2 ;ESPERAR O FLAG=1
F40MS BIT   CDT.3 ;SEMPRE HAVERA ERRO
F80MS BIT   CDT.4 ;DE +/- 5 mSEG
F160MS      BIT   CDT.5
F320MS      BIT   CDT.6
F640MS      BIT   CDT.7
;;;;;;;;; LCD Controle
LCD_RS      BIT   AC_LEDS.5   ;LINHA RS DO LCD
LCD_RW      BIT   AC_LEDS.6   ;LINHA RW DO LCD
LCD_E BIT   AC_LEDS.7   ;LINHA E DO LCD
;;;;;;;;; LCD Configuracao (0=desativado e 1=ativado)
LCD_S BIT   LCD_CONF.0  ;DISPLAY SHIFT (ON/OFF)
LCD_ID      BIT   LCD_CONF.1  ;0=DECR    1=INCR
LCD_B BIT   LCD_CONF.2  ;CURSOR BLINK (ON/OFF)
LCD_C           BIT             LCD_CONF.3      ;CURSOR UNDERLINE (ON/OFF)
LCD_D BIT   LCD_CONF.4  ;DISPLAY (ON/OFF)
LCD_RL      BIT   LCD_CONF.5  ;SHIFT: 0=LEFT 1=RIGHT
LCD_SC      BIT   LCD_CONF.6      ;MOVE: 0=CURSOR 1=DISPLAY
LCD_F       BIT   LCD_CONF.7      ;FORMAT: 0=5X7 1=5X10
LCD_ROLA    BIT   P_LEDS.5        ;ROLAGEM VERTICAL DO LCD
LCD_JANF    BIT   P_LEDS.6        ;FIXA JANELA DO LCD
LCD_FORM    BIT   P_LEDS.7        ;HABILITA FORMATACAO LCD
LCD_JADESQ  BIT   24H.1           ;MARCAR Q JA FEZ LCD_DESQ
;;;;;;;;; Chaves
SW3         BIT     P1.0            ;CHAVE SW3
SW1         BIT     P3.2            ;CHAVE SW1 (PASSO A PASSO)
TEC_DT      BIT     P1.7            ;DADOS TECLADO
TEC_CLK     BIT     P1.6            ;RELOGIO TECLADO
;;;;;;;;; Outros
FIM             EQU             P_LEDS.4        ;FIM ARQUIVO INTEL.HEX

;;;;;;;;; Mapa para as teclas especiais do Teclado do PC
T_F0            EQU     080H            ;Tecla F0
T_F1            EQU     081H            ;Tecla F1
T_F2            EQU     082H            ;Tecla F2
T_F3            EQU     083H            ;Tecla F3
T_F4            EQU     084H            ;Tecla F4
T_F5            EQU     085H            ;Tecla F5
T_F6            EQU     086H            ;Tecla F6
T_F7            EQU     087H            ;Tecla F7
T_F8            EQU     088H            ;Tecla F8
T_F9            EQU     089H            ;Tecla F9
T_F10           EQU     08AH            ;Tecla F10
T_F11           EQU     08BH            ;Tecla F11
T_F12           EQU     08CH            ;Tecla F12
NUM_LOCK        EQU     08DH            ;Tecla Num Lock
```

```
SCROLL_LOCK     EQU     08EH            ;Tecla Scroll Lock
CAPS_LOCK       EQU     08FH            ;Tecla Caps Lock
SHIFT_L         EQU     090H            ;Tecla Shift lado esquerdo
SHIFT_R         EQU     091H            ;Tecla Shift lado direito
ALT_L           EQU     092H            ;Tecla ALT lado esquerdo
ALT_R           EQU     093H            ;Tecla ALT lado direito
CTRL_L          EQU     094H            ;Tecla Control lado esquerdo
CTRL_R          EQU     095H            ;Tecla Control lado direito
PRT_SCR         EQU     096H            ;Tecla Print Screen
PAUSE           EQU     097H            ;Tecla Pause
INSERT          EQU     098H            ;Tecla Insert
DELETE          EQU     099H            ;Tecla Delete
HOME            EQU     09AH            ;Tecla Home
TEND            EQU     09BH            ;Tecla End
PAGE_UP         EQU     09CH            ;Tecla Page Up
PAGE_DW         EQU     09DH            ;Tecla Page Down
ENTER_1         EQU     09EH            ;Tecla Enter principal
ENTER_2         EQU     09FH            ;Tecla Enter teclado numerico
SETA_CIMA       EQU     0A0H            ;Tecla Seta para cima
SETA_BAIXO      EQU     0A1H            ;Tecla Seta para baixo
SETA_ESQ        EQU     0A2H            ;Tecla Seta para esquerda
SETA_DIR        EQU     0A3H            ;Tecla Seta para direita
NUM_0           EQU     0B0H            ;Tecla Numerica  0
NUM_1           EQU     0B1H            ;Tecla Numerica  1
NUM_2           EQU     0B2H            ;Tecla Numerica  2
NUM_3           EQU     0B3H            ;Tecla Numerica  3
NUM_4           EQU     0B4H            ;Tecla Numerica  4
NUM_5           EQU     0B5H            ;Tecla Numerica  5
NUM_6           EQU     0B6H            ;Tecla Numerica  6
NUM_7           EQU     0B7H            ;Tecla Numerica  7
NUM_8           EQU     0B8H            ;Tecla Numerica  8
NUM_9           EQU     0B9H            ;Tecla Numerica  9
NUM_MAIS        EQU     0BAH            ;Tecla Numerica  +
NUM_MENOS       EQU     0BBH            ;Tecla Numerica  -
NUM_VEZES       EQU     0BCH            ;Tecla Numerica  *
NUM_DIV         EQU     0BDH            ;Tecla Numerica  /
NUM_PT          EQU     0BEH            ;Tecla Numerica  .
```

Código 07 – Sub Rotina do Código 3

```
;MUDAMOS CONTEUDO DE R2 E R4 EM DWRITE3

SL_ADR_RD:
MOV             A,#I2C_ETHERNET                 ;ROTINA DE ESCRITA DO
ENDERECO I2C DE LEITURA 8583
INC             A
LCALL           VAI_BYTE
RET

SL_ADR_WR:
MOV             A,#I2C_ETHERNET    ;ROTINA DE ESCRITA DO ENDERECO I2C DE
ESCRITA 8583
LCALL           VAI_BYTE
RET

VAI_BYTE:
MOV              R2,#8                      ;ROTINA DE ENVIO DE BYTE I2C
LBX:
RLC             A
```

```
MOV   SDA,C
LCALL           TMP
SETB  SCL
LCALL           TMP
CLR   SCL
LCALL           TMP
DJNZ            R2,LBX
RET

VEM_BYTE:
SETB             SDA                          ;ROTINA DE RECEPCAO DE BYTE
I2C
LCALL           TMP
MOV   R2,#8
LBY:
SETB             SCL
LCALL           TMP
MOV   C,SDA
RLC   A
CLR   SCL
LCALL           TMP
DJNZ  R2,LBY
RET

STARTI2C:
SETB            SCL                          ;ROTINA DE START I2C
LCALL           TMP
CLR             SDA
LCALL           TMP
CLR   SCL
LCALL           TMP
RET

STOPI2C:
SETB            SCL                          ;ROTINA DE STOP I2C
LCALL           TMP
SETB  SDA
LCALL           TMP
RET

ACKI2C:
SETB            SDA                          ;ROTINA DE ACKNOLEDGE I2C
LCALL           TMP
SETB  SCL
LCALL           TMP
MOV   C,SDA
JC    ERRO1
CLR   SCL
LCALL           TMP
RET
NOACKI2C:
SETB            SCL                      ;ROTINA PARA NOACKNOLEDGE I2C
LCALL           TMP
MOV   C,SDA
JNC   ERRO2
CLR   SCL
LCALL           TMP
RET
ERRO1:
MOV             DPTR,#LEDS_ADR           ;ROTINA PARA INDICACAO DE ERRO
I2C TX
```

```
MOV   A,#05H
MOVX  @DPTR,A
MOV             A, #'E'
LCALL           SER_CHAR
SJMP  $
ERRO2:
MOV             DPTR,#LEDS_ADR            ;ROTINA PARA INDICACAO DE ERRO
I2C RX
MOV   A,#03H
MOVX  @DPTR,A
MOV             A, #'e'
LCALL           SER_CHAR
SJMP  $
TEMPO:  MOV             R7,#1              ;ROTINA DE DELAY
TP1:    MOV             R6,#0
TP2:    MOV              R5,#0
TP3:    DJNZ            R5,TP3
DJNZ  R6,TP2
DJNZ  R7,TP1
RET
TMP:    MOV              R5,#20
TP4:    DJNZ            R5,TP4
RET

BWRITE:                                   ;ROTINA DE ESCRITA DE BYTE I2C
        LCALL           STARTI2C
        LCALL           SL_ADR_WR
        LCALL           ACKI2C
        MOV             A,R7
        LCALL           VAI_BYTE
        LCALL           ACKI2C
        MOV             A,R6
        LCALL           VAI_BYTE
        CALL           ACKI2C
        LCALL           STOPI2C
        RET

BWRITE2:                                                        ;ROTINA DE
ESCRITA DE BYTE I2C (ENDERECO 16 BITS)
                LCALL           STARTI2C
                LCALL           SL_ADR_WR
                LCALL           ACKI2C
                MOV             A,R7
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                MOV             A,R6
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                MOV             A,R3
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                LCALL           STOPI2C
                RET
BREAD:                                    ;ROTINA DE LEITURA DE BYTE I2C
                LCALL           STARTI2C
                LCALL           SL_ADR_WR
                LCALL           ACKI2C
                MOV             A,R7
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                LCALL           STARTI2C
```

```
                LCALL           SL_ADR_RD
                LCALL           ACKI2C
                LCALL           VEM_BYTE
                LCALL           NOACKI2C
                LCALL           STOPI2C
                RET
BREAD2:                 ;ROTINA DE LEITURA DE BYTE I2C (ENDERECO 16 BITS)
                LCALL           STARTI2C
                LCALL           SL_ADR_WR
                LCALL           ACKI2C
                MOV             A,R7
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                MOV             A,R6
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                LCALL           STOPI2C
                LCALL           STARTI2C
                LCALL           SL_ADR_RD
                LCALL           ACKI2C
                LCALL           VEM_BYTE
                LCALL           NOACKI2C
                LCALL           STOPI2C
                RET
DWRITE:              ;ROTINA DE ESCRITA DE DADOS COM BUFFER INTERNO I2C
                LCALL           STARTI2C
                LCALL           SL_ADR_WR
                LCALL           ACKI2C
                MOV             R1,#XMT_DATI
                MOV             A,@R1
                MOV             R0,A
LACOTX:         INC             R1
                MOV             A,@R1
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                DJNZ            R0,LACOTX
                LCALL           STOPI2C
                RET

DWRITE2:     ;ROTINA DE ESCRITA DE DADOS (ATE 256B) I2C A PARTIR DE BUFFER
EXTERNO
                LCALL           STARTI2C
                LCALL           SL_ADR_WR
                LCALL           ACKI2C
                MOV             DPTR,#XMT_DAT
                MOVX            A,@DPTR
                MOV             R0,A
LACOTX2:        INC             DPTR
                MOVX            A,@DPTR
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                DJNZ            R0,LACOTX2
                LCALL           STOPI2C
                RET

DWRITE3:                                                        ;ROTINA DE
ESCRITA DE DADOS (ATE 2KB) I2C A PARTIR DE BUFFER EXTERNO
                LCALL           STARTI2C                        ;USADA PARA
MANDAR A PAGINA
                LCALL           SL_ADR_WR
                LCALL           ACKI2C
```

```
                MOV             A,#40H
                ADD             A,TWMSB
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                MOV             A,TWLSB
                LCALL           VAI_BYTE
                LCALL           ACKI2C
TAMANHO DO BUFFER
                MOV             DPH1,#2                         ;CARREGAR
TAMANHO DO BUFFER
                MOV             DPL1,#0F7H                      ;MSB E LSB
                MOV             DPTR,#XMT_DAT
LACOTX32:
                CLR             C                       ;DECREMENTO 16BITS
                MOV             A,DPL1
                SUBB            A,#1
                MOV             DPL1,A
                MOV             A,DPH1
                SUBB            A,#0
                MOV             DPH1,A
                MOVX            A,@DPTR
                INC             DPTR
                LCALL           VAI_BYTE
                LCALL           ACKI2C
;                DJNZ            R0,LACOTX32
                MOV            A,DPL1
                JNZ            LACOTX32      ;Compare # of bytes remaining
                MOV            A,DPH1
                JNZ            LACOTX32
                LCALL           STOPI2C
                RET


DREADBAD:                                                       ;ROTINA DE
LEITURA DE DADOS I2C COM BUFFER INTERNO
                                                                ;ROTINA
BURRA: IMPLEMENTADA SEM DUMP -> BAD!
                MOV             R1,#XMT_DAT                     ;NA
PRIMEIRA POSICAO DE XMT_DAT DEVE ESTAR A QUANTIDADE DE BYTES
                MOV             A,@R1                           ;NA
SEGUNDA, O ENDERECO INICIAL
                MOV             R0,A                            ;ENDERECOS
DE 8 BITS
                INC             R1
                MOV             A,@R1
                MOV             R7,A
                DEC             R0                              ;ENDERECO
EH DADO PARA O BUFFER MAIS NAUM EH DADO A SER RECEBIDO

LACORX:         LCALL           STARTI2C
                LCALL           SL_ADR_WR
                LCALL           ACKI2C
                MOV             A,R7
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                LCALL           STARTI2C
                LCALL           SL_ADR_RD
                LCALL           ACKI2C
                LCALL           VEM_BYTE
                INC             R7
                INC             R1
```

```
                MOV             @R1,A
                LCALL           NOACKI2C
                LCALL           STOPI2C
                DJNZ            R0,LACORX
                RET

DREADGOOD:                                                      ;ROTINA DE
LEITURA DE DADOS (ATE 2KB) I2C

;IMPLEMENTADA COM DUMP -> GOOD!
                MOV             DPTR,#RCV_DAT                   ;NA
PRIMEIRA POSICAO DE RCV_DAT DEVE ESTAR A QUANTIDADE DE BYTES
                MOVX            A,@DPTR                         ;NA
SEGUNDA, O ENDERECO INICIAL
                MOV             R0,A
                INC             DPTR
                MOVX            A,@DPTR
                MOV             R7,A

;++++++++++++++++++++++++++++++++++
                DEC             R0                              ;ENDERECO
EH DADO PARA O BUFFER MAIS NAUM EH DADO A SER RECEBIDO
;++++++++++++++++++++++++++++++++++
                LCALL           STARTI2C
                LCALL           SL_ADR_WR
                LCALL           ACKI2C
                MOV             A,R7
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                LCALL           STARTI2C
                LCALL           SL_ADR_RD
                PUSH            ACC
                MOV             A,#'I'
                LCALL           SER_CHAR
                POP             ACC
                LCALL           ACKI2C
                DEC             R0                              ;CONDICAO
DE NOACKNOLEDGE FORA DE LACO: FORCAR R0 PARA FAZER ULTIMA ITERACAO FORA DO
LACO

LACORX2:        LCALL           VEM_BYTE
                LCALL           ACKI2C
                INC             DPTR
                MOVX            @DPTR,A
                DJNZ            R0,LACORX2

                LCALL           VEM_BYTE
                INC             DPTR
                MOVX            @DPTR,A
                LCALL           NOACKI2C
                LCALL           STOPI2C
                RET

DREADBAD2:                                                      ;ROTINA DE
LEITURA DE DADOS (ATE ??B/MAX216B) I2C

;IMPLEMENTADA SEM DUMP -> BAD2!
                MOV             DPTR,#RCV_DAT                   ;NA
PRIMEIRA POSICAO DE RCV_DAT DEVE ESTAR A QUANTIDADE DE BYTES
                MOVX            A,@DPTR                         ;NA
SEGUNDA, O ENDERECO INICIAL
```

```
                MOV             R0,A
                INC             DPTR
                MOVX            A,@DPTR
                MOV             R7,A
                INC             DPTR
                MOVX            A,@DPTR
                MOV             R6,A
                DEC             R0                                      ;ENDERECO
EH DADO PARA O BUFFER MAIS NAUM EH DADO A SER RECEBIDO

LACORX3:        LCALL           STARTI2C
                LCALL           SL_ADR_WR
                LCALL           ACKI2C
                MOV             A,R7
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                MOV             A,R6
                LCALL           VAI_BYTE
                LCALL           ACKI2C
                LCALL           STARTI2C
                LCALL           SL_ADR_RD
                LCALL           ACKI2C
                LCALL           VEM_BYTE
                INC             R6
                INC             DPTR
                MOVX            @DPTR,A
                LCALL           NOACKI2C
                LCALL           STOPI2C
                DJNZ            R0,LACORX3
                RET
RDEBUG:         MOV             A,#LF
                LCALL           SER_CHAR
                MOV             R1,#XMT_DAT
                MOV             A,@R1
                MOV             R0,A
                INC             R1
                DEC             R0
LPSTX:          INC             R1
                MOV             A,@R1
                LCALL           SER_W8
                MOV             A,#' '
                LCALL           SER_CHAR
                DJNZ            R0,LPSTX
                MOV             A,#LF
                LCALL           SER_CHAR
                RET
RDEBUG2:        MOV             A,#LF
                LCALL           SER_CHAR
                MOV             R1,#RCV_DAT
                MOV             A,@R1
                MOV             R0,A
                INC             R1
                DEC             R0
LPSTX2:         INC             R1
                MOV             A,@R1
                LCALL           SER_W8
                MOV             A,#' '
                LCALL           SER_CHAR
                DJNZ            R0,LPSTX
                MOV             A,#LF
                LCALL           SER_CHAR
```

```
                RET

;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;
;     TRANSFER Subroutine
;
;     This subroutine copies data from the EPROM referenced by DPTR into a
;     Buffer referenced by R1.
;
;     DPTR = String stored into PROGRAM MEMORY
;     R1 = Buffer in which data shall be stored
;
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

TRANSFER:
                ;MOV             DPH,
                CLR             A                                ;Clears ACC
                MOVC            A, @A+DPTR   ;Moves contents of DPTR into A
                MOV             @R1, A       ;Copies A into Buffer
                MOV             R0, A        ;Copies A into R0 (# of bytes)
                INC             R1           ;Next address
                INC             DPTR         ;Next location
                CLR             A            ;Clears A
NEXT:
                MOVC            A, @A+DPTR   ;Moves contents of DPTR into A
                MOV             @R1, A       ;Copies A  into Buffer
                INC             R1           ;Next address
                INC             DPTR         ;Next location
                CLR             A            ;Clears A
                DJNZ            R0, NEXT     ;Compare # of bytes remaining
                RET
;If all bytes copied, return

;TRANSFERENCIA PARA MEMORIA DE DADOS EXTERNA (255B): OBSERVE SINTAXE DA
STRING

TRANSFER2:
                CLR             A                                ;Clears ACC
                MOVC            A,@A+DPTR                        ;POE NO ACC
                PUSH            DPH
                PUSH            DPL             ;GUARDA DPTR
                MOV             DPH,DPH0        ;PEGA ENDERECO DO BUFFER
                MOV             DPL,DPL0
                MOVX            @DPTR,A         ;COLOCA DADO NO BUFFER
                MOV             R0,A
                POP             DPL
                POP             DPH              ;RECUPERA DPTR
                INC DPTR
                                                 ;SOMA DE 16 BITS
                CLR             C
                MOV             A,#1
                ADD             A,DPL0
                MOV             DPL0,A
                CLR             A
                ADDC            A,DPH0
                MOV             DPH0,A
NEXT2:          MOVC            A,@A+DPTR       ;POE NO ACC
                PUSH            DPH
                PUSH            DPL             ;GUARDA DPTR
                MOV             DPH,DPH0        ;PEGA ENDERECO DO BUFFER
                MOV             DPL,DPL0
```

```
                MOVX            @DPTR,A         ;COLOCA DADO NO BUFFER
                POP             DPL
                POP             DPH             ;RECUPERA DPTR
                INC DPTR                        ;PROXIMA POSICAO DA STRING
                                                ;SOMA DE 16 BITS
                CLR             C
                MOV             A,#1
                ADD             A,DPL0
                MOV             DPL0,A
                CLR             A
                ADDC            A,DPH0
                MOV             DPH0,A
                DJNZ            R0, NEXT2       ;Compare # of bytes remaining

;TRANSFERENCIA PARA MEMORIA DE DADOS EXTERNA (2KB)

TRANSFER3:
                ;MOV            DPH1,#1         ;CARREGAR TAMANHO DO BUFFER
                MOV             DPH1,#3H
                MOV             DPL1,#15H
                CLR             A                               ;Clears ACC
                MOVC            A,@A+DPTR                       ;POE NO ACC
                PUSH            DPH
                PUSH            DPL             ;GUARDA DPTR
                MOV             DPH,DPH0        ;PEGA ENDERECO DO BUFFER
                MOV             DPL,DPL0
                MOVX            @DPTR,A         ;COLOCA DADO NO BUFFER
                MOV             R0,A
                POP             DPL
                POP             DPH             ;RECUPERA DPTR
                INC DPTR
                                                ;SOMA DE 16 BITS
                CLR             C
                MOV             A,#1
                ADD             A,DPL0
                MOV             DPL0,A
                CLR             A
                ADDC            A,DPH0
                MOV             DPH0,A
NEXT3:
                CLR             C
                MOV             A,DPL1
                SUBB            A,#1
                MOV             DPL1,A
                MOV             A,DPH1
                SUBB            A,#0
                MOV             DPH1,A
                CLR             A
                MOVC            A,@A+DPTR               ;POE NO ACC
                PUSH            DPH
                PUSH            DPL                     ;GUARDA DPTR
                MOV             DPH,DPH0        ;PEGA ENDERECO DO BUFFER
                MOV             DPL,DPL0
                MOVX            @DPTR,A         ;COLOCA DADO NO BUFFER
                POP             DPL
                POP             DPH             ;RECUPERA DPTR
                INC DPTR                ;PROXIMA POSICAO DA STRING
                                        ;SOMA DE 16 BITS
                CLR             C
                MOV             A,#1
                ADD             A,DPL0
```

```
                MOV             DPL0,A
                CLR             A
                ADDC            A,DPH0
                MOV             DPH0,A
                MOV            A,DPL1
                JNZ            NEXT3                      ;Compare # of
bytes remaining
                MOV            A,DPH1
                JNZ            NEXT3
                RET                                              ;If all
bytes copied, return
```

Código 08 – Rotinas para Acessar o LCD

```
;B_LCD.ASM       V1.0
;
; Usado pelo BOOTT.ASM da PLACA "PENTA-CONTROLADORA"
; Contem todas as rotinas para acessar o LCD
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                         ROTINAS LCD                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_CRLF                                ;
; FAZ O CARRIAGE RETURN E O LINE FEED (LEVA EM CONTA O BIT LCD_ROLA) ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_CRLF:       LCALL           LCD_LF
                LCALL           LCD_CR
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_LF                                  ;
; FAZ O LINE FEED (LEVA EM CONTA O BIT LCD_ROLA)                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_LF:         MOV             A,LCD_DIM
                JNB             ACC.7,LCDX82
                LCALL           LCD_RD_ESTADO
                JNB             LCD_ROLA,LCDX80
                JNB             ACC.6,LCDX80
                PUSH            ACC
                LCALL           LCD_COPIA
                LCALL           LCD_AP_2L2
                POP             ACC
                SJMP            LCDX81
LCDX80:         XRL             A,#40H
LCDX81:         LCALL           LCD_DDRAM
LCDX82:         RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_CR                                  ;
; FAZ O CARRIAGE RETURN                                             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_CR:         LCALL           LCD_RD_ESTADO
                ANL             A,#40H
                PUSH            ACC
```

```
                LCALL           LCD_HOME
                CLR             LCD_JADESQ
                POP             ACC
                LCALL           LCD_DDRAM
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_SOBE                                    ;
; FAZ CURSOR SUBIR UMA LINHA                                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_SOBE:       MOV             A,LCD_DIM
                JNB             ACC.7,LCDX78
                LCALL           LCD_RD_ESTADO
                JNB             ACC.6,LCDX78
                CLR             ACC.6
                LCALL           LCD_DDRAM
LCDX78:         RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_DESCE                                   ;
; FAZ CURSOR DESCER UMA LINHA                                           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_DESCE:      MOV             A,LCD_DIM
                JNB             ACC.7,LCDX79
                LCALL           LCD_RD_ESTADO
                JB              ACC.6,LCDX79
                SETB            ACC.6
                LCALL           LCD_DDRAM
LCDX79:         RET

;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_AVP                                     ;
; FAZ CURSOR AVANCAR UM ESPACO                                          ;
; LEVA EM CONTA A FORMATACAO SE LCD_FORM=1                              ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_AVP:        JNB             LCD_FORM,LCDX77
                LCALL           LCD_RD_ESTADO
                ANL             A,#3FH
                PUSH            ACC
                MOV             B,LCD_DIM
                ANL             B,#3FH
                DEC             B
                SUBB            A,B
                POP             ACC
                INC             A
                JC              LCDX77          ;CY=1->  A<TAM
                JB              LCD_JANF,LCDX75
                CJNE            A,#LCD_L1_FIM,LCDX76
                RET
LCDX76:         LCALL           LCD_DESQ
                SETB            LCD_JADESQ
LCDX77:         LCALL           LCD_CDIR
LCDX75:         RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_BSP                                     ;
; FAZ O BACKSPACE (RETROCEDE CURSOR UMA POSICAO)                        ;
```

```
; LEVA EM CONTA A FORMATACAO SE LCD_FORM=1                              ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_BSP:        JNB             LCD_FORM,LCDX73
                LCALL           LCD_RD_ESTADO   ;LER POSICAO CURSOR
                ANL             A,#3FH
                CJNE            A,#LCD_L1_INI,LCDX71  ;E' PRIMEIRA COLUNA ?
                RET
LCDX71:         MOV             B,LCD_DIM
                ANL             B,#3FH
                CJNE            A,B,LCDX74      ;A=TAM ?
                CLR             LCD_JADESQ
LCDX72:         LCALL           LCD_DDIR
LCDX73:         LCALL           LCD_CESQ
                RET
LCDX74:         CLR             C
                SUBB            A,B
                JC              LCDX73          ;CY=1-> A<TAM
                SJMP            LCDX72          ;CY=0-> A>TAM
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_STRC                                    ;
; IMPRIME UMA STRING (TERMINAR COM 0) NO LCD                            ;
; DPTR = INICIO DA STRING NA MEMORIA DE PROGRAMA                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_STRC:       CLR              A
      MOVC  A,@A+DPTR   ;LER UM CHAR
      JZ    LCDX4       ;SE ZERO, TERMINAR
      INC   DPTR
                LCALL           LCD_CHARI
      SJMP  LCD_STRC
LCDX4:          RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_STRX                                    ;
; IMPRIME UMA STRING (TERMINAR COM 0) NO LCD                            ;
; DPTR = INICIO DA STRING NA MEMORIA DE DADOS EXTERNA                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_STRX:       CLR              A
      MOVX  A,@DPTR     ;LER UM CHAR
      JZ    LCDX5       ;SE ZERO, TERMINAR
      INC   DPTR
      LCALL LCD_CHARI
      SJMP  LCD_STRX
LCDX5:       RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_W16                                     ;
; IMPRIMIR EM HEXADECIMAL UMA PALAVRA DE 16 BITS                        ;
; A=MSB e B=LSB DA PALAVRA A SER IMPRESSA                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_W16:        LCALL           LCD_W8
                MOV             A,B
                LCALL           LCD_W8
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_W8                                      ;
```

```
; IMPRIMIR EM HEXADECIMAL UMA PALAVRA DE 8 BITS                    ;
; Acc = BYTE A SER IMPRESSO                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_W8:         PUSH            B
                LCALL           HEX_ASC
                LCALL           LCD_CHARI
      MOV   A,B
                LCALL           LCD_CHARI
                POP             B
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_D16                                ;
; IMPRIMIR EM DECIMAL UMA PALAVRA DE 16 BITS                       ;
; Acc = MSB E B=LSB                                                ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_D16:        PUSH            DPL
                MOV             DPL,A
                MOV             A,R7            ;
                PUSH            ACC             ;
                MOV             A,R6            ;
                PUSH            ACC             ;GUARDAR
                MOV             A,R5            ;OS
                PUSH            ACC             ;REGISTRADORES
                MOV             A,R4            ;USADOS
                PUSH            ACC             ;
                MOV             A,R3            ;
                PUSH            ACC             ;
                MOV             A,DPL
                LCALL           BIN16_BCD
                MOV             A,R7
                JNZ             LCDX66
                MOV             A,R6
                ANL             A,#0F0H
                JNZ             LCDX67
                MOV             A,R6
                ANL             A,#0FH
                JNZ             LCDX68
                MOV             A,R5
                ANL             A,#0F0H
                JNZ             LCDX69
                SJMP            LCDX70
LCDX66:         ADD             A,#30H
                LCALL           LCD_CHARI
                MOV             A,R6
                ANL             A,#0F0H
LCDX67:         SWAP            A
                ADD             A,#30H
                LCALL           LCD_CHARI
                MOV             A,R6
                ANL             A,#0FH
LCDX68:         ADD             A,#30H
                LCALL           LCD_CHARI
                MOV             A,R3
                ANL             A,#0F0H
LCDX69:         SWAP            A
                ADD             A,#30H
                LCALL           LCD_CHARI
LCDX70:         MOV             A,R5
```

```
                ANL             A,#0FH
                ADD             A,#30H
                LCALL           LCD_CHARI
                POP             ACC             ;
                MOV             R3,A            ;
                POP             ACC             ;
                MOV             R4,A            ;RECUPERAR
                POP             ACC             ;OS
                MOV             R5,A            ;REGISTRADORES
                POP             ACC             ;USADOS
                MOV             R6,A            ;
                POP             ACC             ;
                MOV             R7,A            ;
                POP             DPL
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_D8                                              ;
; IMPRIMIR EM DECIMAL UMA PALAVRA DE 8 BITS                                     ;
; Acc = BYTE A SER IMPRESSO                                                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_D8:         PUSH            DPL
                PUSH            B
                MOV             DPL,A
                MOV             A,R7            ;
                PUSH            ACC             ;SALVAR
                MOV             A,R6            ;REGISTRADORES
                PUSH            ACC             ;USADOS
                MOV             A,DPL
                LCALL           BIN8_BCD
                MOV             A,R7
                JNZ             LCDX63
                MOV             A,R6
                ANL             A,#0F0H
                JNZ             LCDX64
                SJMP            LCDX65
LCDX63:         ADD             A,#30H
                LCALL           LCD_CHARI
                MOV             A,R6
                ANL             A,#0F0H
LCDX64:         SWAP            A
                ADD             A,#30H
                LCALL           LCD_CHARI
LCDX65:         MOV             A,R6
                ANL             A,#0FH
                ADD             A,#30H
                LCALL           LCD_CHARI
                POP             ACC             ;
                MOV             R6,A            ;RECUPERAR
                POP             ACC             ;REGISTRADORES
                MOV             R7,A            ;
                POP             B
                POP             DPL
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_B16                                             ;
; IMPRIMIR EM BINARIO UMA PALAVRA DE 16 BITS                                     ;
; Acc = MSB e B = LSB                                                           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;
LCD_B16:        LCALL           LCD_B8
                MOV             A,B
                LCALL           LCD_B8
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_B8                                              ;
; IMPRIMIR EM BINARIO UMA PALAVRA DE 8 BITS                                     ;
; Acc = BYTE A SER IMPRESSO                                                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_B8:         PUSH            B
                MOV             B,#8
LCDX57:         RLC             A
                PUSH            ACC
                MOV             A,#'1'
                JC              LCDX58
                MOV             A,#'0'
LCDX58:         LCALL           LCD_CHARI
                POP             ACC
                DJNZ            B,LCDX57
                POP             B
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_BCD4                                            ;
; IMPRIMIR EM DECIMAL QUATRO DIGITOS BCD                                        ;
; A=MSD e B=LSD DOS DIGITOS A SEREM IMPRESSOS                                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_BCD4:       LCALL           LCD_BCD2
                MOV             A,B
                LCALL           LCD_BCD2
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_BCD2                                            ;
; IMPRIMIR EM DECIMAL DOIS DIGITOS BCD                                          ;
; Acc = BCD A SER IMPRESSO                                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_BCD2:       PUSH            ACC
      SWAP  A
      ANL   A,#0FH
      ADD   A,#30H
                LCALL           LCD_CHARI
      POP   ACC
      ANL   A,#0FH
      ADD   A,#30H
                LCALL           LCD_CHARI
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_CHAR                                            ;
; IMPRIME UM CARACTER NA POSICAO ATUAL DO CURSOR E AVANCA CURSOR                ;
; Acc = CARACTER A SER IMPRESSO                                                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_CHAR:       PUSH            DPH
                LCALL           LCD_WR_DADO
```

```
      POP   DPH
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          LCD_CHARI (inteligente)                          ;
; IMPRIME UM CARACTER NA POSICAO ATUAL DO CURSOR E AVANCA CURSOR             ;
; FAZ CONTROLE SEGUNDO FLAGS "LCD_ROLA" E "LCD_JANF"                          ;
; Acc = CARACTER A SER IMPRESSO                                              ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_CHARI:      PUSH            ACC
                PUSH            B
                PUSH            DPH
                JB              LCD_FORM,LCDX54 ;LCD FORMATADO ?
                LCALL           LCD_WR_DADO
LCDX54C:        POP             DPH
                POP             B
                POP             ACC
                RET
LCDX54:         CJNE            A,#CR,LCDX54A   ;E' CARRIAGE RETURN ?
                LCALL           LCD_CRLF        ;GERAR CR E LF
                SJMP            LCDX54C
LCDX54A:        CJNE            A,#LF,LCDX54B   ;E' LINE FEED ?
                LCALL           LCD_LF
                SJMP            LCDX54C
LCDX54B:        XCH             A,B
                LCALL           LCD_RD_ESTADO   ;VER POSICAO CURSOR
                INC             A               ;INC POSICAO CURSOR
                XCH             A,B
                LCALL           LCD_WR_DADO     ;ESCREVE CARACTER
                JB              LCD_JANF,LCDX10
;nao_rola nao_janela
;    rola nao_janela
R_NJ:
NR_NJ:          MOV             A,B             ;2xC
                JB              ACC.6,LCDX19
                JB              LCD_JADESQ,LCDX21
                MOV             A,LCD_DIM       ;<40H
                JB              ACC.7,LCDX55
                RR              A
LCDX55:         ANL             A,#7FH
                DEC             A
                SUBB            A,B
                JNC             LCDX17
LCDX21:         MOV             A,B
                XRL             A,#LCD_L1_FIM
                JNZ             LCDX18
                JB              LCD_ROLA,LCDX60 ;APROVEITAR PARA R_NJ
                MOV             A,#LCD_L1_INI   ;
LCDX20:         LCALL           LCD_DDRAM
LCDX18:         SETB            LCD_JADESQ        ;MARCAR COMECOU DESC ESQ
                LCALL           LCD_DESQ
LCDX17:         POP             DPH
                POP             B
                POP             ACC
      RET
LCDX19:         JB              LCD_JADESQ,LCDX22;>40
                CLR             ACC.6
                MOV             B,A
                MOV             A,LCD_DIM
                JB              ACC.7,LCDX56
```

```
                RR              A
LCDX56:         ANL             A,#7FH
                DEC             A
                SUBB            A,B
                JNC             LCDX17
LCDX22:         MOV             A,B
                XRL             A,#LCD_L2_FIM
                JNZ             LCDX18
                JB              LCD_ROLA,LCDX61 ;APROVEITAR PARA R_NJ
                MOV             A,#LCD_L2_INI   ;
                SJMP            LCDX20
LCDX60:         LCALL           LCD_HOME
                MOV             A,#LCD_L2_INI
                LCALL           LCD_DDRAM
                CLR             LCD_JADESQ
                SJMP            LCDX17
LCDX61:         LCALL           LCD_COPIA
                LCALL           LCD_AP_2L2
                SJMP            LCDX60

LCDX10:         JB              LCD_ROLA,R_J

;nao_rola JANELA
NR_J:           MOV             A,LCD_DIM
                JB              ACC.7,LCDX13
                RR              A               ;1 LINHA (QUEBRA NO MEIO)
                XCH             A,B
                JB              ACC.6,LCDX14
LCDX16:         XRL             A,B             ;<40H
                ANL             A,#3FH
                JNZ             LCDX15A
                MOV             A,#LCD_L2_INI   ;
                SJMP            LCDX15
LCDX14:         XRL             A,B             ;>40H
                ANL             A,#3FH
                JNZ             LCDX15A
                MOV             A,#LCD_L1_INI   ;
LCDX15:         LCALL           LCD_DDRAM
LCDX15A:        POP             DPH
                POP             B
                POP             ACC
                RET
LCDX13:         XCH             A,B             ;2 LINHAS
                JB              ACC.6,LCDX16
                SJMP            LCDX14
;rola janela
R_J:            MOV             A,LCD_DIM
                JB              ACC.7,LCDX41
                RR              A               ;1 LINHA
                XCH             A,B
                JB              ACC.6,LCDX42
LCDX44:         XRL             A,B             ;<40
                ANL             A,#3FH
                JNZ             LCDX43
LCDX45:         MOV             A,#LCD_L2_INI
                LCALL           LCD_DDRAM
LCDX43:         POP             DPH
                POP             B
                POP             ACC
                RET
LCDX42:         XRL             A,B             ;>40
```

```
                ANL             A,#3FH
                JNZ             LCDX43
                LCALL           LCD_AP_1L1
                LCALL           LCD_HOME
                SJMP            LCDX43
LCDX41:         XCH             A,B             ;2 LINHAS
                JNB             ACC.6,LCDX44
                XRL             A,B             ;>40
                ANL             A,#3FH
                JNZ             LCDX43
                LCALL           LCD_COPIA
                LCALL           LCD_AP_2L2
                LCALL           LCD_HOME
                SJMP            LCDX45
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_DEFC                                        ;
; DEFINIR UM CARACTER GRAFICO  DO USUARIO                                   ;
; RECEBE: A=CODIGO DO CARACTER A SER DEFINIDO (0, 1, ..., 7)                ;
;         R0 A R7 CONTEM OS CODIGOS PARA CADA LINHA  DO CARACTER           ;
;         R0=TOPO e R7=INFERIOR                                           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_DEFC:       RL              A               ;MONTAR
                RL              A               ;ENDERECO
                RL              A               ;DA
                ADD             A,#40H          ;CGRAM
                LCALL           LCD_CGRAM       ;ATUALIZAR END. CGRAM
                MOV             A,R0            ;0
                LCALL           LCD_WR_DADO
                MOV             A,R1            ;1
                LCALL           LCD_WR_DADO
                MOV             A,R2            ;2
                LCALL           LCD_WR_DADO
                MOV             A,R3            ;3
                LCALL           LCD_WR_DADO
                MOV             A,R4            ;4
                LCALL           LCD_WR_DADO
                MOV             A,R5            ;5
                LCALL           LCD_WR_DADO
                MOV             A,R6            ;6
                LCALL           LCD_WR_DADO
                MOV             A,R7            ;7
                LCALL           LCD_WR_DADO
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_LERC                                        ;
; LER UM CARACTER GRAFICO DO USUARIO                                        ;
; RECEBE: A=CODIGO DO CARACTER A SER LIDO                                   ;
;         R0 A R7 RETORNAM OS CODIGOS PARA CADA LINHA DO CARACTER           ;
;         R0=TOPO e R7=INFERIOR                                           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_LERC:       RL              A               ;MONTAR
                RL              A               ;ENDERECO
                RL              A               ;DA
                ADD             A,#40H          ;CGRAM
                LCALL           LCD_CGRAM       ;ATUALIZAR END. CGRAM
                LCALL           LCD_RD_DADO
                MOV             R0,A            ;0
```

```
                LCALL           LCD_RD_DADO
                MOV             R1,A            ;1
                LCALL           LCD_RD_DADO
                MOV             R2,A            ;2
                LCALL           LCD_RD_DADO
                MOV             R3,A            ;3
                LCALL           LCD_RD_DADO
                MOV             R4,A            ;4
                LCALL           LCD_RD_DADO
                MOV             R5,A            ;5
                LCALL           LCD_RD_DADO
                MOV             R6,A            ;6
                LCALL           LCD_RD_DADO
                MOV             R7,A            ;7
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       LCD_AP_2L1       LCD_AP_2L2      LCD_AP_1L1                  ;
; APAGA UMA LINHA DO LCD, LEVA EM CONTA O FLAG LCD_JANF               ;
; LCD_AP_2L1: APAGA LINHA 1 DE UM LCD DE 2 LINHAS                     ;
; LCD_AP_2L2: APAGA LINHA 2 DE UM LCD DE 2 LINHAS                     ;
; LCD_AP_1L1: APAGA LINHA 1 DE UM LCD DE 1 LINHA                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_AP_2L1:     MOV             A,#LCD_L1_INI
                SJMP            LCDX30
LCD_AP_2L2:     MOV             A,#LCD_L2_INI
LCDX30:         LCALL           LCD_DDRAM
                MOV             B,#40
                JNB             LCD_JANF,LCDX31
                MOV             A,LCD_DIM
                CLR             ACC.7
                MOV             B,A
LCDX31:         MOV             A,#' '
                LCALL           LCD_CHAR
                DJNZ            B,LCDX31
                RET
LCD_AP_1L1:     MOV             A,#LCD_L1_INI
                LCALL           LCD_DDRAM
                MOV             B,#80
                JNB             LCD_JANF,LCDX31
                MOV             A,LCD_DIM
                RR              A
                MOV             B,A
LCDX32:         MOV             A,#' '
                LCALL           LCD_CHAR
                DJNZ            B,LCDX32
                MOV             A,#LCD_L2_INI
                LCALL           LCD_DDRAM
                MOV             A,LCD_DIM
                RR              A
                MOV             B,A
                SJMP            LCDX31
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                               LCD_COPIA                           ;
; COPIA LINHA 2 PARA A LINHA 1, LEVA EM CONTA " LCD_JANF"             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_COPIA:      MOV             A,R7
                PUSH            ACC
```

```
                MOV             A,R6
                PUSH            ACC
                MOV             B,#40
                JNB             LCD_JANF,LCDX39
                MOV             A,LCD_DIM
                CLR             ACC.7
                MOV             B,A
LCDX39:         MOV             R6,#LCD_L1_INI
                MOV             R7,#LCD_L2_INI
LCDX40:         MOV             A,R7
                LCALL           LCD_DDRAM
                LCALL           LCD_RD_DADO
                PUSH            ACC
                MOV             A,R6
                LCALL           LCD_DDRAM
                POP             ACC
                LCALL           LCD_WR_DADO
                INC             R7
                INC             R6
                DJNZ            B,LCDX40
                POP             ACC
                MOV             R6,A
                POP             ACC
                MOV             R7,A
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                  LCD_CDIR     LCD_CESQ                                      ;
; AVANCAR CURSOR UMA POSICAO PARA DIREITA OU ESQUERDA                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_CDIR:       SETB            LCD_RL
      SJMP  LCDX6
LCD_CESQ:       CLR             LCD_RL
LCDX6:          CLR             LCD_SC
                LCALL           LCD_CDS
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                  LCD_DDIR    LCD_DESQ                                       ;
; DESLOCAR DISPLAY UMA POSICAO PARA DIREITA OU ESQUERDA                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_DDIR:       SETB            LCD_RL
                SJMP            LCDX7
LCD_DESQ:       CLR             LCD_RL
LCDX7:          SETB            LCD_SC
                LCALL           LCD_CDS
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_DDRAM  (Display Data RAM)                     ;
; DEFINE NO CONTADOR UM ENDERECO DA DDRAM (MOVE CURSOR)                       ;
; RECEBE: Acc=ENDERECO (0 ATE 7FH)                                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_DDRAM:      ORL             A,#80H
                LCALL           LCD_WR_INSTR
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;                          LCD_CGRAM (Character Generator RAM)       ;
; DEFINE NO CONTADOR UM ENDERECO DA CGRAM                             ;
; RECEBE: Acc=ENDERECO (0 ATE 3FH)                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_CGRAM:      ORL             A,#40H
                LCALL           LCD_WR_INSTR
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          LCD_INIC                                  ;
; INICIALIZAR LCD SEGUNDO DEFINICOES FEITAS EM LCD_CONFIG             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_INIC:       LCALL           LCD_FS
                LCALL           LCD_DEC
                LCALL           LCD_EMS
                LCALL           LCD_CLR
                LCALL           LCD_HOME
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          LCD_CLR                                   ;
; REALIZAR A OPERACAO DE CLEAR NO DISPLAY LCD                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_CLR:        MOV             A,#1
;CMDO CLEAR
                LCALL           LCD_WR_INSTR
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          LCD_HOME                                  ;
; REALIZAR HOME PARA CURSOR E DISPLAY                                 ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_HOME:       MOV             A,#2
;CMDO HOME
                LCALL           LCD_WR_INSTR
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          LCD_EMS                                   ;
; DEFINIR O CHARACTER ENTRY MODE, SEGUE  O DEFINIDO EM LCD_CONF       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_EMS:        MOV             A,LCD_CONF
      ANL   A,#3              ;SEPARAR OS BITS DE INTERESS
      ORL   A,#4              ;INDICAR CHAR ENTRY MODE
                LCALL           LCD_WR_INSTR
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          LCD_DEC                                   ;
; DEFINIR O DISPLAY ON/OFF E CURSOR, SEGUE  O DEFINIDO EM LCD_CONF    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_DEC:        MOV             A,LCD_CONF
      RR    A
      RR    A
      ANL   A,#7              ;SEPARAR OS BITS DE INTERESS
```

```
      ORL   A,#8              ;INDICAR DISPLAY E CURSOR
                LCALL           LCD_WR_INSTR
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_CDS                                         ;
; DEFINIR O DISPLAY/CURSOR SHIFT, SEGUE  O DEFINIDO EM LCD_CONF             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_CDS:        MOV             A,LCD_CONF
      RR    A
      RR    A
      RR    A
      ANL   A,#0CH            ;SEPARAR OS BITS DE INTERESS
      ORL   A,#10H            ;INDICAR CHAR ENTRY MODE
                LCALL           LCD_WR_INSTR
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_FS                                          ;
; DEFINIR O FUNCTION SET, SEGUE  O DEFINIDO EM LCD_CONF E LCD_DIM           ;
; "TODOS LCDs TEM 2 LINHAS" ==> ACC.3=1                                     ;
; LCD 1x16 E FORMADO POR DUAS LINHAS DE 8 CARACTERES                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_FS:         MOV             A,#30H          ;INTERF DE 8 BITS
                MOV             C,LCD_F         ;DEFINIR 5x10 OU 5x7
      MOV   ACC.2,C
                SETB            ACC.3           ;SEMPRE 2 LINHAS
                LCALL           LCD_WR_INSTR
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_WR_INSTR                                    ;
; ESCREVER UM COMANDO NO LCD (REGISTRADOR INTERNO)                          ;
; RECEBE: Acc=CMDO A SER ESCRITO                                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_WR_INSTR:   PUSH            ACC
                CALL            LCD_BUSY?
                LCALL           LCD_RS0_RW0      ;SELECIONAR REG INTERNO
      POP   ACC
                LCALL           LCD_WR_BXD
                LCALL           LCD_E1                          ;FAZER E=1
                LCALL           LCD_E0                          ;VOLTAR E=0
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_WR_DADO                                     ;
; ESCREVER UM DADO NO LCD (DDRAM OU CGRAM)                                  ;
; RECEBE: Acc=DADO A SER ESCRITO                                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_WR_DADO:    PUSH            ACC
                CALL            LCD_BUSY?
                LCALL           LCD_RS1_RW0                     ;SELECIONAR
REG DADOS
      POP   ACC
                LCALL           LCD_WR_BXD
                LCALL           LCD_E1                          ;FAZER E=1
                LCALL           LCD_E0                          ;VOLTAR E=0
```

```
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          LCD_RD_DADO                                        ;
; LER UM DADO DO LCD (DDRAM OU CGRAM)                                         ;
; RETORNA: Acc=DADO LIDO                                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_RD_DADO:    LCALL           LCD_RD_ESTADO
                JC              LCD_RD_DADO
;ESPERAR BUSY=0
                LCALL           LCD_RS1_RW1           ;SELECIONAR REG DADOS
                LCALL           LCD_E1                          ;FAZER E=1
                LCALL           LCD_RD_BXD
                PUSH            ACC
                LCALL           LCD_E0                        ;VOLTAR E=0
                POP             ACC
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          LCD_RD_ESTADO                                      ;
; LER O ESTADO DO LCD, PERMITE VERIFICAR O BUSY                               ;
; RETORNA: Acc=ENDERECO, C=BUSY (0=NAO BUSY  E  1=BUSY)                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_RD_ESTADO:  LCALL           LCD_RS0_RW1                     ;SELECIONAR
MODO ESTADO
                LCALL           LCD_E1                          ;FAZER E=1
                LCALL           LCD_RD_BXD
      PUSH  ACC
                LCALL           LCD_E0
      POP   ACC
      MOV   C,ACC.7
      CLR   ACC.7
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          LCD_BUSY?                                          ;
; ESPERANDO LCD FICAR DESOCUPADO (FLAG BUSY=0)                                ;
; RECEBE: NADA           RETORNA: SE FLAG BUSY = 0                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_BUSY?:      PUSH            DPH
                MOV             DPH,#END_LEDS_H
                SETB            AC_IV           ;DESABILITA LATCH BXD
                CLR             LCD_RS          ;RS=0
                SETB            LCD_RW          ;RW=1 (LEITURA)
                MOV             A,AC_LEDS
                XRL             A,#7
                MOVX            @DPTR,A         ;FAZER RS=0, RW=1, E=0
LCDX9:          SETB            LCD_E           ;E=1
                MOV             A,AC_LEDS
                XRL             A,#7
                MOVX            @DPTR,A         ;FAZER RS=0, RW=1, E=1
                MOV             DPH,#END_LCD_H
                MOVX            A,@DPTR         ;LER BXD
                PUSH            ACC
                MOV             DPH,#END_LEDS_H
                CLR             LCD_E           ;FAZER RS=0, RW=1, E=0
                MOV             A,AC_LEDS
                XRL             A,#7
```

```
                MOVX            @DPTR,A         ;FAZER E=0
                POP             ACC
                JB              ACC.7,LCDX9     ;BUSY = 1?
                POP             DPH
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_WR_BXD                                      ;
; ESCREVE UM DADO NO LATCH DO LCD (BARRAMENTO XD) - LATCH OE=0              ;
; O PINO OE DO LATCH DO LCD JA DEVE ESTAR HABILITADO                        ;
; RECEBE: Acc=DADO A SER ESCRITO                                           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_WR_BXD:     PUSH            DPH
                MOV             DPH,#END_LCD_H
      MOVX  @DPTR,A
                POP             DPH
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           LCD_RD_BXD                                      ;
; LE UM DADO PELO BARRAMENTO XD                                             ;
; O PINO OE DO LATCH DO LCD JA DEVE ESTAR DESABILITADO                      ;
; RETORNA: Acc=DADO LIDO                                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_RD_BXD:     PUSH            DPH
                MOV             DPH,#END_LCD_H
      MOVX  A,@DPTR
                POP             DPH
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; LCD_RS0_RW0  LCD_RS0_RW1  LCD_RS1_RW0  LCD_RS1_RW1  LCD_E0 LCD_E1  ;
; CONJUNTO DE ROTINAS PARA SELECIONAR MODO DE ACESSO AO LCD                 ;
; RS | RW = SELECAO                                                        ;
;  0    0 = ESCRITA NO REGISTRADOR INTERNO (COMANDOS)                      ;
;  0    1 = LEITURA, VEM O BUSY E ENDERECO                                 ;
;  1    0 = ESCRITA NO REGISTRADOR DE DADOS (DDRAM OU CGRAM)               ;
;  1    1 = LEITURA NO REGISTRADOR DE DADOS (DDRAM OU CGRAM)               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
LCD_RS0_RW0:    CLR             AC_IV                           ;ESCRITA
REG INTERNO
      CLR   LCD_RS
      CLR   LCD_RW
      SJMP  LCDX1
LCD_RS0_RW1:    SETB            AC_IV                           ;LEITURA
ESTADO
      CLR   LCD_RS
      SETB  LCD_RW
      SJMP  LCDX1
LCD_RS1_RW0:    CLR             AC_IV                           ;ESCRITA
REG DADOS
      SETB  LCD_RS
      CLR   LCD_RW
      SJMP  LCDX1
LCD_RS1_RW1:    SETB            AC_IV                           ;LEITURA
REG DADOS
      SETB  LCD_RS
      SETB  LCD_RW
```

```
      SJMP  LCDX1
LCD_E0:         CLR             LCD_E                           ;FAZER E=0
      SJMP  LCDX1
LCD_E1:         SETB            LCD_E           ;FAZER E=1
LCDX1:          PUSH            DPH
                MOV             DPH,#END_LEDS_H ;LATCH DOS LEDS
                MOV             A,AC_LEDS       ;ESTADO DOS LEDS E
CONTROLES
                XRL             A,#7            ;INVERTER BITS DOS LEDS
                MOVX            @DPTR,A         ;ENVIAR (RS/RW/E)
                POP             DPH
                RET
```

Código 09 – Sub Rotinas de Conversão para BCD

```
;B_OUTRAS.ASM   V1.0
;
; Rotinas de apoio para o BOOTT.ASM da PLACA "PENTA-CONTROLADORA"
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           BIN16_BCD                                          ;
; Converter um palavra de 16 bits em BCD                                       ;
; Recebe  : Acc=MSB B=LSB                                                      ;
; Retorna : R7,R6,R5,R4,R3 os cinco possiveis digitos bcd                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
BIN16_BCD:      MOV             R5,#0
                MOV             R4,#0
                MOV             R3,#0
                PUSH            B
                LCALL           BIN8_BCD
                MOV             B,#6            ;MULTIPLICAR POR 6
BCD1:           MOV             A,R3
                ADD             A,R6
                DA              A
                MOV             R3,A
                MOV             A,R4
                ADDC            A,R7
                DA              A
                MOV             R4,A
                DJNZ            B,BCD1
                MOV             B,#2            ;MULTIPLICAR POR 200
BCD2:           MOV             A,R4
                ADD             A,R6
                DA              A
                MOV             R4,A
                MOV             A,R5
                ADDC            A,R7
                DA              A
                MOV             R5,A
                DJNZ            B,BCD2
;FAZER ARRUMACAO PARA MULTIPLICAR POR 50
                MOV             A,R6
                ANL             A,#0F0H
                ADD             A,R7
                SWAP            A
                MOV             R7,A
                MOV             A,R6
```

```
                ANL             A,#0FH
                SWAP            A
                MOV             R6,A
                MOV             B,#5            ;MULTIPLICAR POR 50
BCD3:           MOV             A,R3
                ADD             A,R6
                DA              A
                MOV             R3,A
                MOV             A,R4
                ADDC            A,R7
                DA              A
                MOV             R4,A
                MOV             A,R5
                ADDC            A,#0            ;CUIDAR DO CARRY
                DA              A
                MOV             R5,A
                DJNZ            B,BCD3
;SE CONSEGUIU 256*BCD(MSB), FALTA ADICIONAR O LSB [256*BCD(MSB)+BCD(LSB)]
                POP             ACC
                LCALL           BIN8_BCD
                MOV             A,R3
                ADD             A,R6
                DA              A
                MOV             R3,A
                MOV             A,R4
                ADDC            A,R7
                DA              A
                MOV             R4,A
                MOV             A,R5
                ADDC            A,#0
                DA              A
                MOV             R5,A
;TERMINOU CONVERSAO, ARRUMAR OS DIGITOS NA POSICAO CORRETA
                MOV             A,R5
                MOV             R7,A
                MOV             A,R4
                MOV             R6,A
                MOV             A,R3
                MOV             R5,A
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                             BIN8_BCD                                       ;
; Converter um byte em BCD                                                   ;
; Recebe  : Acc=Byte                                                         ;
; Usa B:                                                                     ;
; Retorna : R7,R6  (R7=0000aaaa e R6=ccccdddd)                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
BIN8_BCD:       MOV             B,#100
                DIV             AB
                MOV             R7,A
                MOV             A,B
                MOV             B,#10
                DIV             AB
                SWAP            A
                ADD             A,B
                MOV             R6,A
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;                          ASC_HEX (sem teste)                          ;
; Converter dois caracteres HEXA ASCII no hexadecimal binario           ;
; Recebe  : Acc=LSN  B=MSN                                              ;
; Retorna : Acc=byte                                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ASC_HEX:        LCALL           ASC_NIB          ;CONVERTER LSNibble
      XCH   A,B
                LCALL           ASC_NIB          ;CONVERTER MSNibble
      SWAP  A
      ADD   A,B
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                          ASC_NIB                                      ;
; Converter um nibble que esta em ASCII no binario (sem teste)          ;
; Caso nao seja um caracter valido, retorna ZERO (A<30) ou F (A>4F)     ;
; Recebe  : Acc=nibble(ASCII)                                           ;
; Retorna : Acc=nibble(binario)                                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
ASC_NIB:        CLR             C
      SUBB  A,#30H      ;A-30H SE CY=1 --> A<30H
                JC              ASCNIB3         ;       SE CY=0 --> A>=30H
      PUSH  ACC
      SUBB  A,#10 ;A-10 SE CY=1 --> A<10
                JNC             ASCNIB2         ;     SE CY=0 --> A>=10
      POP   ACC
      RET
ASCNIB2:        POP             ACC
      SUBB  A,#17H      ;A-17H SE CY=1 --> A<17H
                JNC             ASCNIB4         ;       SE CY=0 -->
A>=17H(ERRO)
      ADD   A,#10H
      RET
ASCNIB3:        MOV             A,#0            ;A<30H ==> 0
                RET
ASCNIB4:        MOV             A,#0FH          ;A>4FH ==> F
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                       HEX_ASC                                         ;
; Converter um byte em dois caracteres ASCII                            ;
; Recebe  : Acc=byte(binario)nibble(ASCII)                              ;
; Retorna : Acc=MSN e B=LSN                                             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
HEX_ASC:        MOV             B,A
      ANL   A,#0FH
                LCALL           NIB_ASC
      XCH   A,B
      SWAP  A
      ANL   A,#0FH
                LCALL           NIB_ASC
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                       NIB_ASC                                         ;
; Converter um nibble em ASCII                                          ;
; Recebe  : Acc=nibble (binario)                                        ;
; Retorna : Acc=nibble (ASCII)                                          ;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
NIB_ASC:        PUSH            ACC
      CLR   C
      SUBB  A,#10 ;A-10--> CY=1 ==> A<10
      POP   ACC   ;    --> CY=0 ==> A>=10
      JC    NA1
      ADD   A,#7
NA1:            ADD             A,#30H
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                     ROTINAS INTERRUPCAO                             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           EXT1 - Interrupcao                        ;
; Rotina de servico para a interrupcao Externa 1                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
EXT1:           PUSH            ACC
                PUSH            PSW
                PUSH            DPH
                PUSH            DPL
                SETB            RS0             ;BANCO 1
                CLR             RS1
                MOV             DPH,#PED_INTH
                MOVX            A,@DPTR         ;LER PEDIDOS
                JB              ACC.0,EXT1_01   ;FOI IV0 ?
                MOV             DPTR,#AP_IV0
                MOVX            @DPTR,A         ;APAGAR PEDIDO IV0
EXT1_01:        JB              ACC.1,EXT1_02   ;FOI IV1 ?
                MOV             DPTR,#AP_IV1
                MOVX            @DPTR,A         ;APAGAR PEDIDO IV1
EXT1_02:        JB              ACC.2,EXT1_03   ;FOI TECLADO ?
                ;LCALL           REC_TEC
                MOV             DPH,#AP_IKBDH
                MOV             DPH,#84H
                MOVX            @DPTR,A         ;APAGAR PEDIDO TECLADO
EXT1_03:        JB              ACC.3,EXT1_04   ;FOI RP ?
                NOP
EXT1_04:        JB              ACC.4,EXT1_05   ;FOI AD ?
                MOV             DPTR,#END_AD
                PUSH            ACC
                MOVX            A,@DPTR
                POP             ACC
EXT1_05:        JB              ACC.5,EXT1_06   ;FOI USB
                NOP
EXT1_06:        JB              ACC.6,EXT1_07   ;FOI RESERVA ?
                NOP
EXT1_07:        POP             DPL
                POP             DPH
                POP             PSW
                POP             ACC
                RETI
```

Código 10 – Rotina de Debugação do 8031 via porta serial

```
;PASSO.ASM V1.0
;
;Programa para fazer o passo a passo do 8031 via porta serial
;
; COMANDOS DISPONIVEIS:
; P (Passo) --> executa um passo
; D (Dump)  --> faz dump da RAM INTERNA
; R (Reset) --> simula RESET da CPU
; T (Troca) --> troca valor de um dado qualquer ( T ADR DATA )
;
; RECURSOS UTILIZADOS:
; --> 16 bytes da pilha
; --> porta serial
; --> interrupcao externa 0
;
; OBSERVACOES:
; --> nao alterar registrador IP
; --> nao desabilitar interrupcao externa 0 (manter EA=1 e EX0=1)
; --> enquanto estiver sob controle do passo a passo, o timer1 e usado
;     para baud rate da porta serial, seu valor antigo e guardado
; --> LED VERMELHO QUADRADO deve estar aceso
; --> quando passo a passo assume, a pilha fica:
;         0 -- 0     <==== SP
;         1 -- PCON
;         2 -- SCON
;         3 -- TCON
;         4 -- TMOD
;         5 -- TH1
;         6 -- TL1
;         7 -- DPH
;         8 -- DPL
;         9 -- B
;        10 -- A
;        11 -- PSW
;        12 -- PCH
;        13 -- PCL
;
;
; SEQUENCIA DOS REGISTRADORES:
;   1    2       3     4      5      6     7     8    9    10  11   12
; +---+------+------+------+------+-----+-----+-----+-----+---+---+-----+
; | 0 | PCON | SCON | TCON | TMOD | TH1 | TL1 | DPH | DPL | B | A | PSW |
; +---+------+------+------+------+-----+-----+-----+-----+---+---+-----+
;
;    13    14      15       16        17        18        19        20
; +-----+-----+---------+---------+---------+---------+---------+---------+
; | PCH | PCL | @(PC+0) | @(PC+1) | @(PC+2) | @(PC+3) | @(PC+4) | @(PC+5) |
; +-----+-----+---------+---------+---------+---------+---------+---------+
;
;     21        22      23   24   25    26    27    28    29   30   31
; +---------+---------+----+----+----+------+-----+-----+----+----+----+
; | @(PC+6) | @(PC+7) | SP | IP | IE | SBUF | TH0 | TL0 | R1 | R2 | R3 |
; +---------+---------+----+----+----+------+-----+-----+----+----+----+
;
;   32   33    34  35
; +----+----+----+----+
; | R4 | R5 | R6 | R7 |
; +----+----+----+----+
;
;
; SEQUENCIA DO DUMP DA RAM INTERNA:
```

```
;
; +-----+-----+-----+-----+---+---+---+
; | 127 | 126 | 125 | ... | 2 | 1 | 0 |
; +-----+-----+-----+-----+---+---+---+
;
; manter habilitado a interrupcao externa 0
; IE: EA   -   -   ES  ET1  EX1  ET0  EX0
;      1   X   X    X   X    X    X    1
;
; interrupcao externa 0 deve sempre ter prioridade mais alta
; IP:  -  -  -  PS  PT1  PX1  PT0  PX0
;      X  X  X   0   0    0    0    1
;
; colocar TIMER1 no modo 2 para gerar baud rate
; TMOD: GATE  C/*T  M1  M0    GATE  C/*T  M1 M0
;         0    0    1    0     M     M    M   M
;
; ligar timer1
; TCON: TF1  TR1  TF0  TR0  IE1  IT1  IE0  IT0
;        0    1    M    M    M    M    M    M
;
; porta serial para 8 bits de dados, sem paridade, 1 start, 2 stop
; SCON: SM0  SM1  SM2  REN  TB8  RB8  TI  RI
;        1    1    1    0    1    0   0    0
;
;
; Cristal = 10 MHz
; Para baud rate de 9600 (com ..... de PCON zerado)
;         10000000
;   n = ------------     -----> n = 2,71 --> n = 3
;        12*32*9600
; logo 256-3=253 (0FDH) ----> TH1=TL1=0FDH
;

;               PUBLIC          STEP_HAB, STEP_IN, STEP_ON, STEP_OFF
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                       STEP_HAB                                              ;
; Habilitar o passo a passo                                                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
STEP_HAB:       SETB            PX0
;                MOV             IP,#1
      SETB  EA
      SETB  EX0
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                       STEP_ON                                               ;
; Prende el o passo a passo                                                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
STEP_ON:        SETB            EX0
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                       STEP_OFF                                              ;
; Apaga el passo a passo                                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
STEP_OFF:       CLR             EX0
```

```
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           STEP_IN                                      ;
; Ponto de entrada do programa monitor de passo                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
STEP_IN:        PUSH            PSW             ;SALVAR TODOS OS
REGISTRADORES
      PUSH  ACC
      PUSH  B
      PUSH  DPL
      PUSH  DPH
      PUSH  TL1
      PUSH  TH1
      PUSH  TMOD
      PUSH  TCON
      PUSH  SCON
      PUSH  PCON
      PUSH  0     ;R0
      MOV   TMOD,#20H   ;TIMER1 NO MODO 2
                MOV             TH1,#BAUD
                MOV             TL1,#BAUD
      MOV   SCON,#0C8H  ;8 DADOS, 2 STOP, NO PARID.
      MOV   TCON,#40H   ;LIGAR TIMER1
      MOV   PCON,#0     ;SMOD=0
      CLR   ES
      SETB  REN
      LJMP  SAI_REG     ;ENVIAR OS REGISTRADORES
LOOP: LCALL RX_SERIAL   ;ESPERAR COMANDO
                CJNE            A,#'P',LP1
      LJMP  PASSO
LP1:            CJNE            A,#'D',LP2
      LJMP  DUMP
LP2:            CJNE            A,#'R',LP3
      LJMP  RST
LP3:            CJNE            A,#'T',LP4
      LJMP  TROCA
LP4:          SJMP              LOOP
;     MOV   DPTR,#MSG_NSEI
;     LCALL SAI_MSG
;     SJMP  LOOP
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                       PASSO                                            ;
; Permitir a execucao de um passo                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
PASSO:          POP             0               ;R0
      POP   PCON  ;RESTAURAR O QUE HAVIA
      POP   SCON
      POP   TCON
      POP   TMOD
      POP   TH1
      POP   TL1
      POP   DPH
      POP   DPL
      POP   B
      POP   ACC
      POP   PSW
      RETI
```

```
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           DUMP                                          ;
; Faz o dump da RAM INTERNA                                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
DUMP:           CLR             RS0
      CLR   RS1
      MOV   R0,#127     ;TOPO DA RAM
DP1:            MOV             SBUF,@R0
      LCALL TX_WAIT     ;ENVIAR DADO
      DJNZ  R0,DP1
      MOV   R0,SP ;VALOR ORIGINAL DE R0
      MOV   SBUF,@R0
      LCALL TX_WAIT
      LJMP            LOOP
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                         RST                                             ;
; Simular o RESET da CPU                                                  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
RST:            MOV             DPTR,#MSG_RST   ;AVISAR DO RESET
      LCALL SAI_MSG
      MOV   A,#0FFH
      MOV   P0,A
      MOV   P1,A
      MOV   P2,A
      MOV   P3,A
      CLR   A
      MOV   B,A
      MOV   PSW,A
      MOV   DPH,A
      MOV   DPL,A
      MOV   IP,A
      MOV   IE,A
      MOV   TMOD,A
      MOV   TCON,A
      MOV   TH0,A
      MOV   TL0,A
      MOV   TH1,A
      MOV   TL1,A
      MOV   SCON,A
      MOV   PCON,A
      MOV   SP,#9 ;APOS RETI,SP=7
      MOV   9,#0  ;PREPARAR ENDERECO PARA
      MOV   8,#0
      RETI
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           TROCA                                         ;
; Alterar uma posicao qualquer do 8031                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
TROCA:          LCALL           RX_SERIAL       ;ESPERAR ENDERECO
      MOV   R0,A
      LCALL RX_SERIAL   ;ESPERAR DADO
      MOV   @R0,A
      LJMP  LOOP
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;                          SAI_REG                                          ;
; Enviar registradores pela porta serial                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
SAI_REG:        CLR             RS1
      CLR   RS0   ;BANCO 0
      MOV   R0,SP
      MOV   B,#14 ;CONTADOR
SR1:            MOV             SBUF,@R0        ;ENVIAR O QUE ESTA
      DEC   R0          ;NA PILHA
      LCALL TX_WAIT
      DJNZ  B,SR1
;
      INC   R0
      MOV   DPL,@R0     ;R0 AGORA APONTA PARA PCL
      INC   R0
      MOV   DPH,@R0
      MOV   B,#8  ;8 BYTES DE PROG SEGUINTES
SR2:            CLR             A
      MOVC  A,@A+DPTR
      MOV   SBUF,A
      INC   DPTR
      LCALL TX_WAIT
      DJNZ  B,SR2
;
      MOV   SBUF,SP
      LCALL TX_WAIT
      MOV   SBUF,IP
      LCALL TX_WAIT
      MOV   SBUF,IE
      LCALL TX_WAIT
      MOV   SBUF,SBUF
      LCALL TX_WAIT
      MOV   SBUF,TH0
      LCALL TX_WAIT
      MOV   SBUF,TL0
      LCALL TX_WAIT

      INC   R0
      MOV   PSW,@R0     ;RESTAURAR BANCO
      MOV   SBUF,R1     ;ENVIAR R1 ... R7
      LCALL TX_WAIT
      MOV   SBUF,R2
      LCALL TX_WAIT
      MOV   SBUF,R3
      LCALL TX_WAIT
      MOV   SBUF,R4
      LCALL TX_WAIT
      MOV   SBUF,R5
      LCALL TX_WAIT
      MOV   SBUF,R6
      LCALL TX_WAIT
      MOV   SBUF,R7
      LCALL TX_WAIT
      LJMP  LOOP
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                       TX_WAIT                                             ;
; Esperar que a porta serial transmita o byte                               ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
TX_WAIT:        JNB             TI,TX_WAIT
      CLR   TI
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                     RX_SERIAL                                          ;
; Esperar que chegue um byte pela porta serial                           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
RX_SERIAL:      JNB             RI,RX_SERIAL
      CLR   RI
      MOV   A,SBUF
      RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                       SAI_MSG                                          ;
; Enviar uma mensagem ASCII pela porta serial                            ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
SAI_MSG:        CLR             A
      MOVC  A,@A+DPTR
      MOV   SBUF,A
      INC   DPTR
SMSG1:          JNB             TI,SMSG1
      CLR   TI
      CJNE  A,#0AH,SAI_MSG
      RET
MSG_RST:        DB              'RESET DA CPU',0DH,0AH
MSG_NSEI:       DB              'NAO ENTENDI O COMANDO',0DH,0AH
```

Código 11 – Sub Rotina para Imprimir na Porta Serial

```
;B_SER.ASM      V1.0
;
; Para ser usado com o programa BOOTT.ASM da PLACA "PENTA-CONTROLADORA"
; Contem as rotinas para imprimir via porta serial
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;            ROTINAS PARA IMPRIMIR NA PORTA SERIAL                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           SER_STRC                                     ;
; IMPRIME UMA STRING (TERMINAR COM 0) NA PORTA SERIAL                    ;
; DPTR = INICIO DA STRING NA MEMORIA DE PROGRAMA                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
SER_STRC:       CLR              A
      MOVC  A,@A+DPTR   ;LER UM CHAR
                JZ              SERX4           ;SE ZERO, TERMINAR
      INC   DPTR
                LCALL           SER_CHAR
                SJMP            SER_STRC
SERX4:          RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           SER_STRX                                     ;
```

```
; IMPRIME UMA STRING (TERMINAR COM 0) NA PORTA SERIAL                ;
; DPTR = INICIO DA STRING NA MEMORIA DE DADOS EXTERNA                ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
SER_STRX:       CLR              A
      MOVX  A,@DPTR     ;LER UM CHAR
                JZ               SERX5           ;SE ZERO, TERMINAR
      INC   DPTR
                LCALL            SER_CHAR
                SJMP             SER_STRX
SERX5:          RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           SER_W16                                  ;
; IMPRIMIR EM HEXADECIMAL UMA PALAVRA DE 16 BITS NA PORTA SERIAL     ;
; DPTR = PALAVRA A SER IMPRESSA                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

SER_W16:        PUSH             DPL
                MOV              A,DPH
                LCALL            SER_W8
                POP              ACC
                LCALL            SER_W8
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           SER_W8                                   ;
; IMPRIMIR EM HEXADECIMAL UMA PALAVRA DE 8 BITS NA PORTA SERIAL      ;
; Acc = BYTE A SER IMPRESSO                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
SER_W8:         LCALL            HEX_ASC
                LCALL            SER_CHAR
                MOV              A,B
                LCALL            SER_CHAR
                RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           SER_BCD4                                 ;
; IMPRIMIR EM DECIMAL QUATRO DIGITOS BCD NA PORTA SERIAL              ;
; DPTR = DIGITOS A SEREM IMPRESSOS                                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

SER_BCD4:       PUSH             DPL
                MOV              A,DPH
                LCALL            SER_BCD2
                POP              ACC
                LCALL            SER_BCD2
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           SER_BCD2                                 ;
; IMPRIMIR EM DECIMAL DOIS DIGITOS BCD NA PORTA SERIAL                ;
; Acc = BCD A SER IMPRESSO                                           ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
SER_BCD2:       PUSH             ACC
                SWAP             A
                ANL              A,#0FH
                ADD              A,#30H
                LCALL            SER_CHAR
```

```
                POP             ACC
                ANL             A,#0FH
                ADD             A,#30H
                LCALL           SER_CHAR
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           SER_CRLF                                  ;
; IMPRIME CR E LF NA PORTA SERIAL,                                ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
SER_CRLF:       MOV             A,#CR
                LCALL           SER_CHAR
                MOV             A,#LF
                LCALL           SER_CHAR
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                           SER_CHAR                                  ;
; IMPRIME UM CARACTER NA PORTA SERIAL,                                ;
; ESPERA O CARACTER SER ENVIADO, MONITORANDO O FLAG TI                ;
; A = CARACTER A SER ENVIADO                                          ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
SER_CHAR:       CLR             TI
                MOV             SBUF,A
                JNB             TI,$            ;AGUARDAR CARACTER SER
ENVIADO
                CLR             TI
                RET
```

Código 12 – Sub Rotina para Ler Teclas do PC, Contar e Montar o Byte

```
;B_TEC.ASM  V1.0
;LER TECLAS DO PC E RETORNAR CARACTERES ASCII
;R7 = CONTADOR (11, 10, ..., 0)
;R6 = MONTAR O BYTE

REC_TEC:        PUSH            ACC
                DEC             R7
                MOV             A,R7
                CJNE            A,#10,RT01
                SJMP            RT20
RT01:           CJNE            A,#1,RT02
                SJMP            RT20
RT02:           CJNE            A,#0,RT03
                MOV             R7,#11
                MOV             A,OPTESTE
                CJNE            A,#24H,RT04     ;TESTE 24
                MOV             A,R6
                LCALL           POE
                SJMP            RT20
RT04:           MOV             A,R6            ;TESTE 25
                LCALL           TEC_IDENT
                SJMP            RT20
RT03:           MOV             A,R6
                MOV             C,P1.7
                RRC             A
                MOV             R6,A
RT20:           POP             ACC
```

```
                RET
;
; Rotina para identificar as teclas que sao acionadas
; R5 (BK1) DESIGNA O ESTADO PRESENTES DO DECODIFICADOR
;
TEC_IDENT:      CJNE            R5,#0,QE01
                CJNE            A,#0F0H,TECI_01
                MOV             R5,#1
                RET
TECI_01:        MOV             DPTR,#TEC_TAB
                MOVC            A,@A+DPTR
                LCALL           POE
                RET
QE01:           CJNE            R5,#1,QE02
                MOV             R5,#0
                RET
QE02:           RET

; CODIGOS A SEREM PROCURADOS NA TABELA ASCII
TAB             EQU             0
PLIK            EQU             0
BSP             EQU             0               ;BACK SPACE
ESC             EQU             1CH             ;ESCAPE




TEC_TAB:        ;TECLAS NORMAIS SEM SHIFT
DB 0,   T_F9,   0,   T_F5,   T_F3,  T_F1, T_F2,     T_F12     ;00 - 07
DB 0,  T_F10,  T_F8,  T_F6,   T_F4,   TAB,  '~',       0      ;08 - 0F
DB 0,  ALT_L, HIFT_L,   0,  CTRL_L,  'q',   '1',       0      ;10 - 17
DB 0,     0,    z',   's',   'a',    'w',   '2',       0      ;18 - 1F
DB 0,   'c',   'x',   'd',   'e',    '4',   '3',       0      ;20 - 27
DB 0,   ' ',   'v',   'f',   't',    'r',   '5',       0      ;28 - 2F
DB 0,   'n',   'b',   'h',   'g',    'y',   '6',       0      ;30 - 37
DB 0,     0,   'm',   'j',   'u',    '7',   '8',       0      ;38 - 3F
DB 0,   ',',   'k',   'i',   'o',    '0',   '9',       0      ;40 - 47
DB 0,   '.',   '/',   '|',   ';',    'p',   '-',       0      ;48 - 4F
DB 0,     0,  PLIK,     0,   '[',    '=',     0,       0      ;50 - 57
DB CAPS_LOCK, SHIFT_R,  ENTER_1, ']',  0,    '\', 0,   0      ;58 - 5F
DB 0,     0,     0,     0,     0,      0,    BSP,      0      ;60 - 67
DB 0, NUM_1, 0, NUM_4,      NUM_7,     0,      0,      0      ;68 - 6F
DB NUM_0, NUM_PT, NUM_2, NUM_5, NUM_6, NUM_8, ESC, NUM_LOCK   ;70 - 77
DB T_F11, NUM_MAIS, NUM_3, NUM_MENOS, NUM_VEZES, NUM_9, SCROLL_LOCK,0;78-7F
DB 0,     0,     0,     T_F7,  0,      0,      0,      0      ;80 - 87
DB 0,     0,     0,     0,     0,      0,      0,      0      ;88 - 8F
DB 0,     0,     0,     0,     0,      0,      0,      0      ;90 - 97
DB 0,     0,     0,     0,     0,      0,      0,      0      ;98 - 9F
DB 0,     0,     0,     0,     0,      0,      0,      0      ;A0 - A7
DB 0,     0,     0,     0,     0,      0,      0,      0      ;A8 - AF
DB 0,     0,     0,     0,     0,      0,      0,      0      ;B0 - B7
DB 0,     0,     0,     0,     0,      0,      0,      0      ;B8 - BF
DB 0,     0,     0,     0,     0,      0,      0,      0      ;C0 - C7
DB 0,     0,     0,     0,     0,      0,      0,      0      ;C8 - CF
DB 0,     0,     0,     0,     0,      0,      0,      0      ;D0 - D7
DB 0,     0,     0,     0,     0,      0,      0,      0      ;D8 - DF
DB 0,     0,     0,     0,     0,      0,      0,      0      ;E0 - E7
DB 0,     0,     0,     0,     0,      0,      0,      0      ;E8 - EF
DB 0,     0,     0,     0,     0,      0,      0,      0      ;F0 - F7
```

```
DB 0,     0,     0,     0,     0,      0,      0,      0      ;F8 - FF
```

Código 13 – Faz o Boot via Porta Serial

```
;BOOTT.ASM
;
; Fazer o BOOT da PLACA "PENTA-CONTROLADORA" via porta serial
; Inclui rotinas de TESTE para auxiliar na montagem e teste da placa
; Ao fazer o RESET com SW3 acionada a placa entra no MODO TESTE
;
;             LEDS QUADRADOS
;AMARELO  ==> botao de reset acionado
;VERDE    ==> pronto para receber dados pela porta serial
;VERMELHO ==> modo single step
;

$MOD51
$NOPRINT

; Incluir arquivo com declaracao de constantes e variaveis
$INCLUDE (B_CABEC.ASM)




;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                    Inicio dos Programas                                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ORG   RESET
                LJMP            INICIO

      ORG   EXTI0
                LJMP            EMERG               ;

      ORG   TIMER0
                LJMP            TIM0

      ORG   EXTI1
                LJMP            EXT1

      ORG   TIMER1

      ORG   SINT
                LJMP            SERIAL

;Mensagens usadas pelo sistema
      ORG   50H
MSG4:           DB              'SELEC...',0
MSG5:           DB              'TESTE ',0
MSG6:           DB              CR,LF,'MODO TESTE: OPCAO = ',0
MSG7:           DB              'TESTE',0
MSG8:           DB              '---> SELECIONADO MODO TESTE = ',0
MSG9:           DB              'INEXISTE',0
MSG10:          DB              '*** MODO NAO EXISTE,ESCOLHA OUTRO
!',CR,LF,0
M_ERRO0:        DB              'ERRO=0',0
M_ERRO1:        DB              'ERRO=1',0
M_ERRO2:        DB              'ERRO=2',0
```

```
M_ERRO3:        DB              'ERRO=3',0
M_ERRO4:        DB              'ERRO=4',0
M_ERRO5:        DB              'ERRO=5',0

; Trecho usado pelo TESTE 00
                ORG             1FAH
T_00_1:         SJMP            $

                ORG             200H
INICIO:         MOV             R0,#127         ;ZERAR RAM INTERNA
                CLR             A
IN1:            MOV             @R0,A
                DJNZ            R0,IN1
                MOV             LCD_DIM,#LCD_2X16
      MOV   LCD_CONF,#3EH
                LCALL           LCD_INIC
                MOV             R7,#REBOTE
                CLR             C
IN2:            ORL             C,SW3        ;LER CHAVE SW3 DIVERSAS VEZES
                DJNZ            R7,IN2
                JNC             IN3
                LJMP            MODO_BOOT
IN3:            LJMP            MODO_TESTE
;
;ROTINA DE EMERGENCIA EXTERNAL 0
EMERG:          PUSH            ACC
                PUSH            PSW
                PUSH            B
                PUSH            DPH
                PUSH            DPL
                LCALL           IMP_CONT
                MOV             A,#'*'
                LCALL           LCD_CHAR
                POP             DPL
                POP             DPH
                POP             B
                POP             PSW
                POP             ACC
                RETI
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                       Incluir as Rotinas                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
$INCLUDE (B_TESTE.ASM)
$INCLUDE (B_BOOT.ASM)
$INCLUDE (B_LCD.ASM)
$INCLUDE (B_SER.ASM)
$INCLUDE (B_OUTRAS.ASM)
;$INCLUDE (B_PAP.ASM)
$INCLUDE (B_TEC.ASM)
      END
```

Código 14 – Inicia Acesso ao LCD

```
;COLA_LCD.ASM
; PROGRAMA PARA INICIAR ACESSO AO LCD
```

```
$MOD51
$NOPRINT

; Incluir arquivo com declaracao de constantes e variaveis
$INCLUDE (B_CABEC.ASM)

;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                     Inicio dos Programas                                   ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
                ORG             RESET
                LJMP            INICIO

;MENSAGEM PARA O DISPLAY
                ORG             50H
MSG1:           DB              'TESTE LCD',0

                ORG             100H
INICIO:         MOV             LCD_DIM,#LCD_2X16       ;LCD 2 LINHAS X 16
COLUNAS
                MOV             LCD_CONF,#3EH           ;DIVERSOS
PARAMETROS DO LCD
                LCALL           LCD_INIC                ;INICIALIZA LCD
                MOV             DPTR,#MSG1
                LCALL           LCD_STRC                ;IMPRIME MSG
                JB              P1.0,$                  ;ESPERA CHAVE
                MOV             R7,#'A'
LBX:            MOV             A,R7
                ACALL           LCD_CHAR
                INC             A
                CJNE            A,#'Z'+1,LB
                MOV             A,#'A'
LB:             MOV             R7,A
                ACALL           TEMPO
            SJMP        LBX
;ROTINA PARA GERAR RETARDO
TEMPO:          MOV             R6,#0
TP1:            MOV             R5,#0
                DJNZ            R5,$
                DJNZ            R6,TP1
                RET
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                     Incluir as SUBRotinas                                  ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
$INCLUDE (B_LCD.ASM)
$INCLUDE (B_OUTRAS.ASM)
END
```