UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS
DEPARTAMENTO DE FÍSICA E INFORMÁTICA

"INTERFACE CHEAPERNET PARA IMPLEMENTAÇÃO EM REDES DE INSTRUMENTOS"

LEONARDO BATISTA DE ALMEIDA SCARABELLI

Dissertação apresentada ao Instituto de Física de São Carlos, Universidade de São Paulo, para obtenção do título de Mestre em Ciências "Física Aplicada-opção Física Computacional"

Orientador: Prof. Dr. Valentin Obac Roda

São Carlos – São Paulo
2000

Scarabelli, Leonardo Batista de Almeida
Interface Cheapernet para Implementação de Redes de Instrumento/Leonardo Batista de Almeida Scarabelli. -- São Carlos, 2000.
98 p.

Dissertação (Mestrado) – Instituto de Física de São Carlos, 2000.

Orientador: Prof. Dr. Valentin Obac Roda

1. Instrumentação
I. Título

UNIVERSIDADE
DE SÃO PAULO
Instituto de Física de São Carlos

Av. Dr. Carlos Botelho, 1465
CEP 13560-250 - São Carlos - SP
Brasil

Fone (016) 273-9333
Fax (016) 272-2218

MEMBROS DA COMISSÃO JULGADORA DA DISSERTAÇÃO DE MESTRADO DE LEONARDO BASTISTA DE ALMEIDA SCARABELLI APRESENTADA AO INSTITUTO DE FÍSICA DE SÃO CARLOS, DA UNIVERSIDADE DE SÃO PAULO, EM 29 DE JUNHO DE 2000

COMISSÃO JULGADORA:

Prof. Dr. Valentin Obac Roda/EESC-SEL-USP

Prof. Dr. Jan Frans Willem Slaets/IFSC-FFI-USP

Prof. Dr. Onofre Trindade Júnior/ICMC-SCE-USP

E mail: wladerez@if.sc.usp.br

Dedico esta dissertação aos Meus pais, Nair Batista de Almeida Scarabelli e Nilson Scarabelli que sempre me incentivaram nos estudos, à minha noiva, Keila Fabiana da Silva, pela paciência e compreensão que teve pelos meus dias de ausência e principalmente a minha filha Letícia Silva Scarabelli.

AGRADECIMENTOS:

- Ao Prof. Dr. Valentin Obac Roda, meu orientador, pela sempre disponibilidade na orientação e principalmente pela amizade desenvolvida durante esses anos.
- Ao Bruno Otto Theodoro Rosa, chefe da Seção Técnica de Informática do Instituto de Física de São Carlos, que contribuiu muito para que a interface funcionasse.
- Aos amigos do laboratório Marcelo Marques Simões de Souza e Maximilian Luppe que me auxiliaram para que esta dissertação tivesse o seu final.
- A todos os técnicos que se dispuseram a ajudar a qualquer momento e a todos os amigos do laboratório Pott, Isaura, Bráulio, Alisson e Marcos que de uma forma ou de outra contribuiram com esta dissertação.

Sumário:

Resumo:

Rede de Área Local ("LAN – Local Area Network") é uma rede onde vários terminais e equipamentos estão interconectados por cabos dentro de uma curta distância uns dos outros (a uma distância máxima de 500 m, por exemplo, no mesmo edifício). Nesta Dissertação, desenvolveremos uma interface de rede, tipo Cheapernet, para possibilitar a transmissão de dados de um instrumento de medição a um computador através de uma LAN. O instrumento de medição tomado como base, é uma sonda para medida de variedades físicas e químicas da água.

A interface utiliza um controlador dedicado de rede, o dispositivo DP8390 que faz a administração lógica da rede. As rotinas de inicialização da interface Cheapernet foram desenvolvidas em linguagem montadora do microcontrolador 8051. A interface foi testada utilizando rotinas e subrotinas como se fosse a função "PING", ou seja, no driver montado foi implementado um programa que envia um pacote de dados para o computador e este devolve o mesmo pacote. O contrário também pode ocorrer, ou seja, se enviarmos um pacote de dados do computador utilizando o número IP da interface Cheapernet, este receberá de volta o pacote que foi enviado.

Foi implementado apenas a função "PING" do protocolo ICMP para mostrarmos que a interface de rede funciona. A parte de implementação dos protocolos superiores, tais como TCP ou UDP ficará para trabalhos futuros.

Abstract:

Local Area Networks (LAN) are networks where several terminals and hosts computers are attached via cables, not longer that 500 m, in the same building. In the present thesis, we present the development of a network interface (Cheapernet) which makes possible data transmission between a measuring instrument and a computer. The developed interface was connected to an to measure water physical and chemical variables.

The interface developed was based on a dedicated network controller, the integrated circuit DP8390 which logically manages the network. The initialization routines for the Cheapernet interface were written using the 8051 microcontroller assembly language. The interface was tested using routine and subroutine procedures emulating the "PING" function, in other words, the routines control the interface to send a data packet to a host and wait to receive it back. Also the host can send a data packet using as address the interface's IP and receive it back.

To show that the network interface was operating correctly, only the "PING" function of the ICMP protocol was implemented. Upper protocols such as TCP or UDP, will be left for future works.

Lista de Tabelas:

Lista de Figuras:

Glossário:

- **API** – APPLICATION PROGRAMMING INTERFACE = INTERFACE DE PROGRAMAÇÃO PARA O USUÁRIO.
- **ATM** – ASYNCHRONOUS TRANSFER MODE – MODO DE TRANSFERÊNCIA ASSÍNCRONA; rede de alta velocidade muito utilizada por bancos e grandes empresas.
- **AUI** – ATTACHMENT UNIT INTERFACE = UNIDADE DE CONEXÃO DA INTERFACE; é o cabo transmissor que conecta o MAU com a interface ethernet.
- **BANDA BASE;** é o tipo de transmissão que utiliza toda a banda do canal de comunicação para transmitir, sem precisar utilizar multiplexação.
- **BANDA LARGA;** utiliza modulação por divisão de frequência, permitindo várias comunicações simultâneas. Um exemplo é a TV a cabo.
- **BURST** - rajada, explosão; curta seqüência isolada de sinais transmitidos; "burst mode" = modo intermitente = transmissão de dados em grupos separados de dados feita de modo intermitente.
- **CI** – CIRCUITO INTEGRADO.
- **CODIFICAÇÃO MANCHESTER** – Tipo de codificação utilizada pelo protocolo IEEE 802.3 para evitar ambigüidades dos sinais enviados ou ruídos na rede.
- **COL** – COLLISION DETECT OUTPUT = DETETOR DE COLISÃO NA SAÍDA.

- **CRC** - CYCLIC REDUNDANCY CHECK = VERIFICAÇÃO POR REDUNDÂNCIA CÍCLICA; código de detecção de erro para dados transmitidos.
- **CRS** – CARRIER SENSER = SENSOR DA PORTADORA.
- **CSMA-CD** - CARRIER SENSE MULTIPLE ACCESS-COLLISION DETECTION = MECANISMO DE ACESSO AO MEIO COM DETEÇÃO DE COLISÃO DE COMUNICAÇÕES EM REDE.
- **CTI** – COAXIAL TRANSCEIVER INTERFACE – INTERFACE PARA TRANSMISSÃO COAXIAL; tem a função de transmitir/receber para a/da linha do cabo coaxial para redes locais Cheapernet e Ethernet.
- **DMA** - DIRECT MEMORY ACCESS = ACESSO DIRETO À MEMÓRIA; ligação direta rápida entre um periférico e a memória principal de computador que evita o uso de rotinas de acesso para cada item de dado lido.
- **DTE** – DATA TERMINAL EQUIPAMENT = EQUIPAMENTO TERMINAL DE DADOS.
- **ECL** - EMITTER COUPLED LOGIC = LÓGICA DE EMISSOR ACOPLADO projeto de circuito lógico de alta velocidade usando os emissores dos transistores como conexões de saída para outros estágios.
- **ETHERNET** - forma de rede de área local onde o dado é transmitido em pacotes de comprimento variável e cada dispositivo de rede seleciona somente o pacote enviado para ele.
- **FCS** – FRAME CHECK SEQUENCE = SEQUENCIA DE CHECAGEM DE FRAME; campo de 32 bits de CRC calculado e é acrescentado ao pacote durante a transmissão parra que o computador destino consiga identificar erros durante o tráfego.

- **FDDI** - FIBER DISTRIBUTED DATA INTERFACE = INTERFACE DE DADOS DISTRIBUÍDOS POR FIBRA ÓTICA; rede de área local de alto desempenho por fibra ótica, consistindo de dois anéis de fibras, um transmitindo no sentido horário e o outro no sentido anti-horário.
- **FIFO** - FIRST IN FIRST OUT = PRIMEIRO A ENTRAR, PRIMEIRO A SAIR; método de armazenamento de leitura/escrita no qual o primeiro item armazenado é o primeiro a ser lido.
- **FRAME** - pacote de dados transmitido incluindo informações de controle e rota.
- **HEADER**• = CABEÇALHO ou REGISTRO INICIAL; (a) em uma rede local, um pacote de dados que é enviado antes de uma transmissão para fornecer informação sobre destinos e rotas; (b) informação no início de uma lista de dados referente ao resto dos dados; ± header block = bloco inicial ou cabeçalho = bloco de dados no início de um arquivo contendo dados sobre as características do arquivo;
- **HEARTBEAT** = BATIDA DE CORAÇÃO; na realidade não tem nada a ver com este significado. O gerador "heartbeat" cria uma pequena colisão em um curto período de tempo para se ter certeza de que o circuito de colisão está funcionando corretamente.
- **IEEE** - INSTITUTE OF ELECTRICAL AND ELECTRONIC ENGINEERS = INSTITUTO DOS ENGENHEIROS ELÉTRICOS E ELETRÔNICOS; ± IEEE-802.3 = IEEE-802.3 = padrão de interfaceamento conforme formulado pelo IEEE, onde apenas dados e sinais de estabelecimento de comunicação são utilizados, usado principalmente em laboratórios para conectar computadores a equipamentos de medição.

- **IPG** – INTER PACKET GAP = ESPAÇO ENTRE OS PACOTES; é o tempo mínimo entre a transmissão de 2 pacotes consecutivos.
- **ISO/OSI** – INTERNATIONAL STANDARD ORGANIZATION'S OPEN SYSTEMS INTERCONNECT = ORGANIZAÇÃO INTERNACIONAL PADRÃO DE SISTEMAS ABERTOS INTERCONECTADOS; é o modelo padrão de protocolos de redes e de aplicações distribuídas.
- **JITTER**; este efeito ocorre quando a latência não é constante, ou seja, quando existe duas latências diferentes em um determinado circuito.
- **LSB** - LEAST SIGNIFICANT BIT = BIT MENOS SIGNIFICATIVO dígito binário ocupando a posição mais à direita de uma palavra e que tem a menor potência de dois na palavra (igual a 2 elevado a zero = 1).
- **MAC** – MEDIUM ACCESS CONTROL = CONTROLE DE ACESSO AO MEIO; nada mais é que a interface ethernet.
- **MAU** – MEDIUM ATTACHMENT UNIT = UNIDADE DE CONEXÃO DO MEIO; são os transmissores do padrão original ethernet, fazendo ambos, transmissão e recepção dos sinais no meio.
- **MDI** – MEDIUM DEPENDENT INTERFACE = INTERFACE DEPENDENTE DO MEIO; é um pedaço do hardware usado para manter uma conexão direta do meio Físico e conexão elétrica para o meio (10BASET usa o MDI RJ – 45 estilo telefone).
- **MIRINET** – Rede de Alta Velocidade.
- **MP** – MEDIUM PHYSICAL = MEIO FÍSICO; é o meio físico entre os computadores que carregam os sinais ethernet (10BASE5, 10BASE2, etc.).

- **MSB** - MOST SIGNIFICANT BIT = BIT MAIS SIGNIFICATIVO bit em uma palavra que representa o maior valor ou peso (normalmente o bit que está mais à esquerda).
- **NIC** – NETWORK INTERFACE CONTROLLER = CONTROLADOR DE INTERFACE DA REDE; é o coração dos três CI's que implementam o protocolo completo IEEE - 802.3 responsável pela transmissão e recepção dos pacotes nas camadas de Controle de Acesso ao Meio (MAC).
- **NRZ** - NON RETURN TO ZERO = SEM RETORNO A ZERO.
- **OSI** - OPEN SYSTEM INTERCONNECTION = INTERCONEXÃO DE SISTEMAS ABERTOS; ISO/OSI.
- **PREÂMBULO** – consiste de 1 e 0s (62 bits) alternados com a Codificação de Manchester para conseguir sincronização a nível de bits com o pacote que está chegando.
- **RAM** – RANDOM ACCESS MEMORY = MEMÓRIA DE ACESSO ALEATÓRIO; memória que permite acesso a qualquer posição em qualquer ordem, sem ter que acessar seqüencialmente a partir do primeiro elemento;
- **RXC** – RECEIVE CLOCK = CLOCK RECEBIDO.
- **RXD** – RECEIVE DATA OUTPUT = DADO RECEBIDO NA SAÍDA.
- **SETUP**; configurar, ajustar; inicializar ou definir ou iniciar uma aplicação ou sistema.
- **SFD** – START FRAME DELIMITER = DELIMITADOR DE INÍCIO DE FRAME; consiste de dois 1s consecutivos que avisam ao receptor da mensagem que (após esses dois 1's) a mensagem está chegando.
- **SNI** – SERIAL NETWORK INTERFACE = INTERFACE DE REDE SERIAL; é responsável pela codificação e decodificação dos dados em Código Manchester.

- **SQUELCH CIRCUIT** = CIRCUITO de ALTA IMPEDÂNCIA; na entrada, rejeita sinais com largura de pulso menor que 8 ns (negativo) ou com níveis menores que – 175 MV.
- **STATE IDLE** – ESTADO OSCIOSO; período de tempo durante o qual um dispositivo está ligado mas não executa qualquer ação.
- **TIMER JABBER** = TEMPORIZADOR "JABBER" monitora o Transmissor e inibe a transmissão se o Transmissor estiver ativado em um tempo maior que 20 ms.
- **TRANSCEIVER** – é um elemento do controlador de rede que é responsável, entre outras coisas, em relatar se houve colisão ou não na rede.
- **TCP/IP** - TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL = PROTOCOLO DE CONTROLE DE TRANSMISSÃO/ PROTOCOLO INTERNET.
- **UDP** - USER DATAGRAM PROTOCOL = PROTOCOLO DE DATAGRAMA DO USUÁRIO.

Capítulo 1

1.0 – Introdução:

Nos dias de hoje, a utilização de redes locais vem sendo a cada dia mais e mais comum. No nosso caso específico, a rede local será utilizada para possibilitar a transmissão de dados e o controle de instrumentos de medição através da comunicação destes computadores ligados à rede. O instrumento de medição utilizado como exemplo é uma sonda que tem a finalidade de medir variáveis físicas e químicas da água. Os valores das variáveis medidas pela sonda, serão passados para a interface de rede ficando a disposição de qualquer computador ligado a ela.

A interface de rede construída é baseada no protocolo IEEE 802.3 [20], mais especificamente no "Thin Ethernet" (ou 10BASE2). Esta interface construída foi controlada pelo micro-controlador 8051 por ser o controlador utilizado no instrumento de teste desenvolvido. A linguagem estrutural utilizada, ou seja o "driver" foi desenvolvido em linguagem montadora, para conseguir maior velocidade no processamento.

Existem vários tipos de redes locais além da Ethernet: FDDI, ATM, etc, então porque utilizar logo a rede ethernet?

Para responder esta questão será necessário entendermos algumas coisas antes. O padrão ethernet foi desenvolvido no PARC (Palo Alto Research Center) e depois tornou-se um padrão internacional. Os comitês do padrão IEEE 802.3 foram muito produtivos e deixaram especificada uma ampla gama de opções quanto as técnicas de construção de redes Ethernet em termos de equipamentos utilizados e cabos. O mesmo esquema do cabeçalho ("header") utilizado pelo IEEE 802.3 para o envio de pacotes na rede foi adotado pelo Ethernet.

Para distinguir entre essas diversas implementações, o comitê IEEE desenvolveu uma notação explicativa, na qual a rede é especificada em função dos seus parâmetros. A notação adotada se parece com:

<taxa em Mbps> <método de sinalização> <tamanho máximo do segmento em centenas de metros>

E assim, as alternativas existentes foram:

10BASE5

10BASE2

10BASET

10BROAD36

10BASE-F

As implementações 10BASET e o 10BASE-F não seguem a recomendação de notação original – "T" significa par trançado e "F" significa fibra óptica. Um resumo para essas especificações pode ser vista na tabela I abaixo [21]:

Tabela I *– Especificação das placas de rede ethernet*

	10BASE5	**10BASE2**	**10BASET**	**10BROAD36**	**10BASE-FP**
Meio de Transmissão	Cabo Coaxial (50 Ω)	Cabo Coaxial (50 Ω)	Par trançado não blindado	Cabo Coaxial (75 Ω)	Par de fibra-óptica de 850 nm
Técnica de Sinalização	Banda Base Manchester*	Banda Base Manchester	Banda Base Manchester	Banda Larga (DPSK)	Manchester/ On-Off
Topologia	Barramento	Barramento	Barramento	Barramento/ árvore	Estrela
Tamanho Máximo do Segmento	500 metros	185 metros	100 metros	1800 metros	500 metros
Nós por Segmento	100	30	-	-	33
Diâmetro do cabo	10 mm	5 mm	0.4 – 0.6 (mm)	0.4 – 1.0 (mm)	62.5/125 μm

O sistema ethernet é uma tecnologia de rede local (LAN), que transmite informações entre computadores numa taxa de 10 milhões de bits por segundos (10Mbps). Novos padrões Ethernet já foram desenvolvidos para fornecer dados numa taxa de 100Mbps, 1Gbps e 10Gbps e são chamados de Fast-Ethernet.

* A sincronização de Manchester será explicada no item 2.2.

Escolhemos o sistema Ethernet pelo fato de ser a tecnologia mais popular entre as redes locais, ou seja a grande maioria dos computadores utilizam equipamentos com conexões Ethernet. Pelo fato de seu amplo domínio no mercado de tecnologia, isto ajuda manter os preços competitivos. Aqui entra outro grande aliado, pois a alta utilização de placas Ethernet, faz com que os componentes (os CI's) envolvidos na sua construção também se tornem baratos e assim, torna-se mais acessível a construção da placa (hoje em torno de uns $25.00 conseguimos uma placa com conexão UTP). Mas, se a placa custa tão pouco, é inevitável a seguinte pergunta, porque não compramos uma placa e tentamos apenas fazer o driver da placa "conversando" com esta através do barramento ISA, por exemplo? Aí responderemos que nem sempre é necessário que só entendamos a sua aplicação, mas também é necessário que entendamos o seu funcionamento e a partir daí tirarmos algumas conclusões de como as coisas são planejadas e construídas, tendo um maior domínio da tecnologia. Para a nossa função específica, a de mandar um pacote de dados contendo os parâmetros medidos pela sonda, conseguimos fazer uma interface dentro dos padrões do protocolo IEEE 802.3 com um microcontrolador de 8 bits (no caso, o 8051) que controla todo o funcionamento da interface dando-nos assim, uma opção barata para envio/recepção de pacotes na rede.

De acordo com as 7 camadas do modelo de rede padrão ISO/OSI ("International Standard Organization's Open Systems Interconnect"), o nosso trabalho está inserido principalmente nas seguintes camadas:

- *Em relação à interface:* A interface Cheapernet que construímos estão nas camadas 1 e 2 (física e dados).

- *Em relação ao programa do usuário:* Antes de comentarmos esta relação, é necessário fazermos um parênteses para explicarmos alguns conceitos. Será feito um programa em linguagem montadora que será responsável unicamente em enviar um pacote para o computador hospedeiro ou receber um pacote do mesmo (driver). Alguns protocolos de redes, como por exemplo TCP/IP, UDP ICMP, etc. (que estão normalmente na 3ª e na 4ª camada) utilizam este driver para enviar os pacotes para facilitar o entendimento do usuário. Uma observação importante a falar é que **não será feito neste trabalho nenhum software que utiliza estes protocolos de camadas mais altas,** a não ser até a 3ª camada onde está englobado os protocolos IP e ICMP. Esta parte do software que engloba os protocolos TCP e UDP ficará para futuros trabalhos.

Podemos ver na figura 1 as principais características de cada uma das camadas [37].

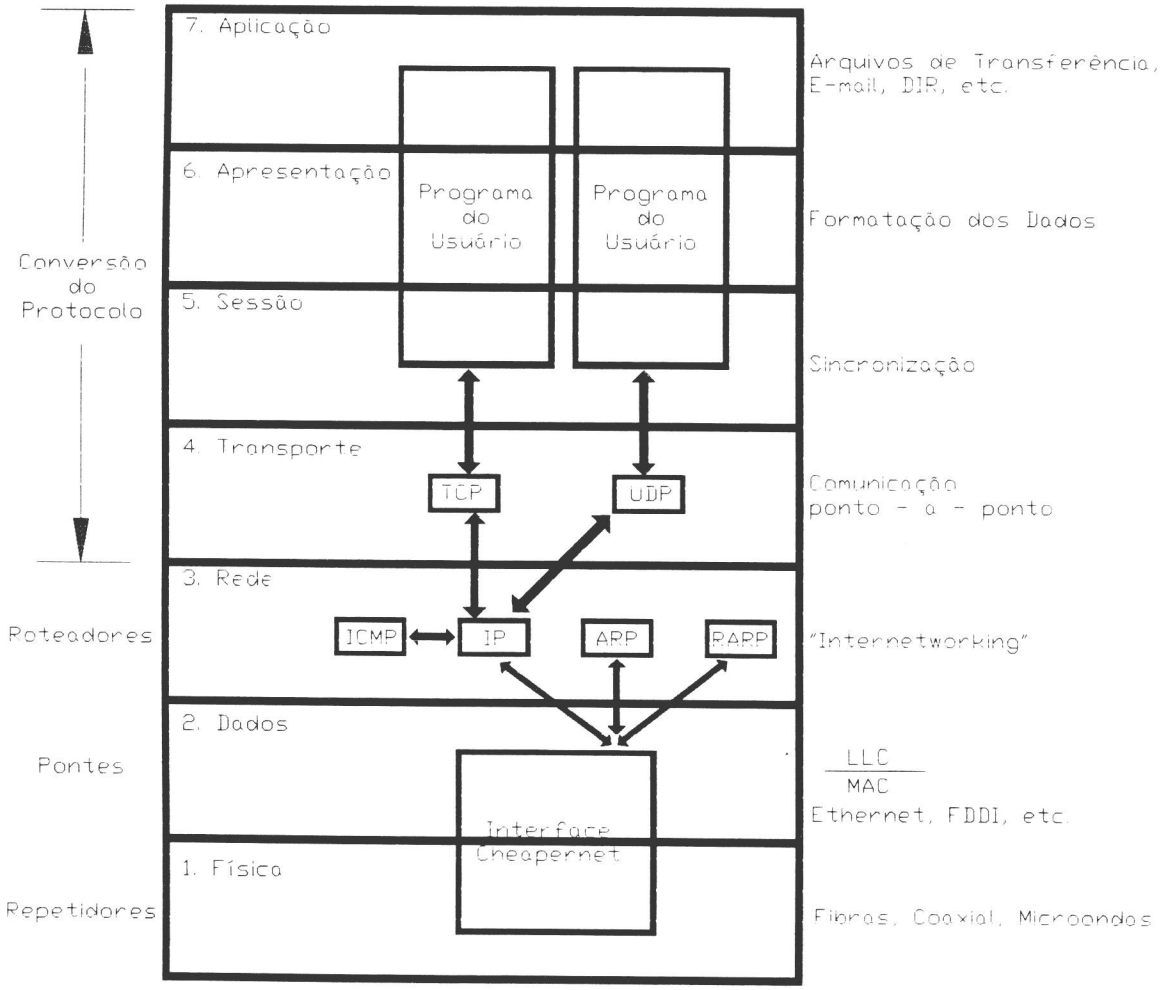


***Figura 1** – Interface Cheapernet relacionada com as Camadas OSI.*

Para entendermos como funciona a transmissão de um pacote, daremos um exemplo. Imagine que um pacote que se quer enviar, possa ser representado da seguinte maneira [32]:

.....

Quando este pacote chega na camada do TCP, ele pega este pacote e quebra-o em pacotes menores chamados de datagramas, da seguinte maneira:

....

O TCP põe um "header" na frente de cada datagrama. Este "header" é normalmente representado por 20 octetos. As partes principais do "header" são os números de portos fonte e destino e o número de sequência. Não comentaremos, nem demonstraremos o header TCP, porque não é esta a função deste exemplo. Só comentaremos os campos mais importantes do "header". Os números de portos são usados para conseguir uma trilha (caminho) para diferentes conexões. O TCP contido no host de origem tem como saber o número do porto utilizado pelo computador destino. Ele põe o valor no campo do porto destino. Cada datagrama tem um número de sequência. Isto é utilizado para que o computador destino possa ter certeza de que conseguiu os datagramas na ordem certa e que nenhum se perdeu. E finalmente o "checksum", que é um número que é calculado de acordo com o valor dos octetos dos datagramas. O resultado é posto no "header". O TCP do computador destino recalcula o "checksum" novamente. Se eles não forem iguais, então alguma coisa aconteceu com o datagrama na transmissão, e ele é jogado fora.

Se nós abreviarmos o "header" TCP como T, o arquivo todo se parecerá com:

T..... T..... T..... T..... T..... T..... T..... T..... T..... T..... T.....

Nível IP: O TCP envia cada um destes datagramas para o IP. É claro que ele tem que dizer para o IP o endereço Internet do computador destino. O emprego do IP é encontrar simplesmente um roteamento para o datagrama e conseguir fazer com que ele chegue ao destino. Para permitir que os "gateways" ou outros sistemas intermediários prossigam com o datagrama, ele adiciona o seu próprio header. Os campos principais neste header são os endereços internet fonte e destino (32 bits de endereço, como 143.107.235.201), o número do protocolo e o checksum.

O endereço internet fonte é simplesmente o endereço de sua máquina. O número do protocolo diz ao IP do destino para enviar o datagrama para o TCP. O checksum permite que o IP do destino verifique que o "header" não foi danificado na transmissão.

Se nós representarmos o header IP por um I, seu arquivo agora se parecerá com:

IT..... IT..... IT..... IT..... IT..... IT..... IT..... IT..... IT..... IT.....

Nível Ethernet: Infelizmente Ethernet têm endereços próprios. Os projetistas do Ethernet queriam ter certeza que nenhuma máquina teriam os mesmos endereços ethernet e assim eles alocaram 48 bits para o endereço ethernet. Pessoas que têm o equipamento ethernet devem registrá-lo com uma autoridade central, para ter certeza de que os números não conflitem com outras manufaturas. Quando você envia um pacote em uma rede Ethernet, todas as máquinas da rede vêem este pacote. Então alguma coisa é preciso ser feita para que a máquina certa receba o pacote para ela destinada, e isto é feito no header ethernet. Cada pacote ethernet tem um header de 14 octetos que incluem os endereços ethernet fonte e destino e um tipo de código. Cada máquina tem que ter uma tabela de endereços ethernet correspondentes ao endereço internet. O tipo de código é utilizado para que se permita que várias diferentes famílias de protocolos utilizem a mesma rede. Finalmente há um checksum. O controlador ethernet calcula um checksum do pacote inteiro. Quando o destino receber o pacote, ele recalcula o checksum e joga-o fora se a resposta for diferente da original. O checksum é posto no final do pacote, não no header.

Se nós representarmos o "header" ethernet com E e o checksum ethernet por C, seu arquivo chegaria no computador destino da seguinte maneira:

EIT.....C EIT.....C EIT.....C EIT.....C EIT.....C EIT.....C EIT.....C

Como não faremos o TCP, em seu lugar pense que teríamos o ICMP. O campo principal do ICMP é o campo que chamamos de código. Neste campo podemos dizer que tipo de informação queremos da rede. No nosso caso, só queremos enviar um pacote para um computador e recebê-lo de volta. Dessa maneira conseguiremos mostrar que a interface funciona. Esta função é muito utilizada computadores conectados por rede para vermos se um computador X está conectado na rede. Se ele estiver, ele responderá com o mesmo pacote para o computador fonte. Esta função é chamada de *PING*.

Assim, se trocarmos a parte do protocolo TCP por ICMP e o header ICMP seria representado pela letra P, teríamos que o arquivo se pareceria com:

EIP.....C EIP.....C EIP.....C EIP.....C EIP.....C EIP.....C EIP.....C

Só para reforçar o que já foi dito acima, só montaremos o protocolo ICMP para mostrarmos a integridade do funcionamento da interface para podermos colocar na parte de resultados e conclusões desta interface. Mas a função desta interface ainda é a de mandar via rede o valor de variáveis medidas por um instrumento desenvolvido no nosso laboratório.

1.1 – O Controle de Acesso ao Meio IEEE 802.3:

O Ethernet é simultaneamente um mecanismo de acesso ao meio ("CSMA/CD – Carrier – Sense Multiple Access with Collision Detection") e a camada física de transmissão em redes de computadores. A operação do CSMA/CD e seus predecessores pode ser resumida a técnicas de acesso aleatório e contenção. Falamos em termos de acesso aleatório porque nunca sabemos quando uma estação conectada à rede vai transmitir – não existe uma seqüência de transmissão.

A contenção se refere ao fato de que todas as estações competem pelo mesmo meio físico.

A técnica de acesso aleatório com detecção de colisão baseia – se no seguinte algoritmo para transmissão dos pacotes:

1. Se o meio estiver livre, transmita, caso contrário vá para 2.
2. Se o canal estiver ocupado, espere até que fique livre e então transmita imediatamente.
3. Se uma colisão for detectada durante a transmissão, transmita um sinal de colisão (jam) breve para certificar-se de que todas as estações percebam que houve uma colisão e parem de transmitir.
4. Depois de transmitir este sinal de colisão, espere um tempo aleatório ("algoritmo de backoff") e então comece a transmitir novamente (vá para passo 1).

Uma observação deve ser feita, os pacotes deverão ser grandes o suficiente para que as estações percebam que houve uma colisão. Esta observação é importante porque o sinal de colisão deve ser detectado por todos os computadores num tempo máximo 512 μs. Se isto não ocorrer, alguma outra estação poderá achar que o canal está livre para a transmissão. Assim, ela poderá transmitir o pacote e provocar mais colisões.

1.2 – Organização do Trabalho:

Esta Dissertação está desenvolvida em 5 capítulos.

No Capítulo 2, explicaremos os aspectos gerais de como funciona a interface Cheapernet e de como é feito o encapsulamento de um pacote a ser transmitido e o desencapsulamento de um pacote recebido.

No Capítulo 3 é explicado como funcionam os três principais CI's (DP8390, DP8391 e DP8392) de uma interface Cheapernet, que faz toda a lógica de administração de pacotes, codificação de Manchester e a codificação do sinal para o cabo coaxial.

No Capítulo 4 explicaremos como funciona o "hardware" da interface Cheapernet e dos seus principais componentes para a administração lógica da rede, e também como será feita a comunicação entre a interface que faz o processamento dos dados medidos pela sonda e a interface Cheapernet.

O Driver (Software) utilizado para a inicialização da interface e para o controle dos pacotes é feito no Capítulo 5. O funcionamento das rotinas e como elas trabalham em conjunto com o hardware, são explicadas neste capítulo.

O Capítulo 6 apresenta os resultados obtidos com a utilização da interface Cheapernet, controlada por um micro-controlador 8051. Fica também no Capítulo 6 as conclusões pertinentes a este trabalho.

Capítulo 2

2.0 – Aspectos Gerais de uma placa de rede:

Recentemente a popularidade de redes tem crescido e como resultado, qualquer tipo de sistema de computador e muitos periféricos são facilmente incorporados à rede. Com a utilização de poucos CI's, o projeto do circuito de uma interface física, torna-se cada vez mais simples. Entretanto o projeto de uma interface pode ser implementado em muitos caminhos diferentes.

Para começarmos a entender os aspectos gerais de uma interface de rede Cheapernet é necessário primeiramente fazer um projeto básico para o interfaceamento do sistema (por exemplo, desempenho da placa, compatibilidade, etc.). Depois de analisado este projeto, é necessário o seu conhecimento técnico para poder aplicá-lo na prática, ou seja, deveremos saber: como será feita a transação da informação na rede? Qual o protocolo a ser utilizado? Como é o formato do frame Cheapernet?

Veremos nos próximos itens, todos essas etapas e assim, nos familiarizaremos com a placa Cheapernet.

O projeto básico para o interfaceamento de um sistema cheapernet inclui a avaliação dos seguintes itens [5]:

Desempenho: Geralmente isto é medido em termos de quantidade de dados transmitidos e recebidos em um dado período de tempo. É melhor quanto maior for o número de dados. Como mostrado na figura 2, a proposta de uma interface da rede é simplesmente: Mover Dados.

Baixo Custo: Obviamente o usuário preferirá pagar o menor custo possível para ter uma conexão à rede.

Compatibilidade: Tanto o "software" quanto o "hardware" deverão ser compatíveis com o padrão estabelecido na indústria. Para o mercado de computação, isto significa a habilidade para trabalhar com um "software" e um "hardware padrões".

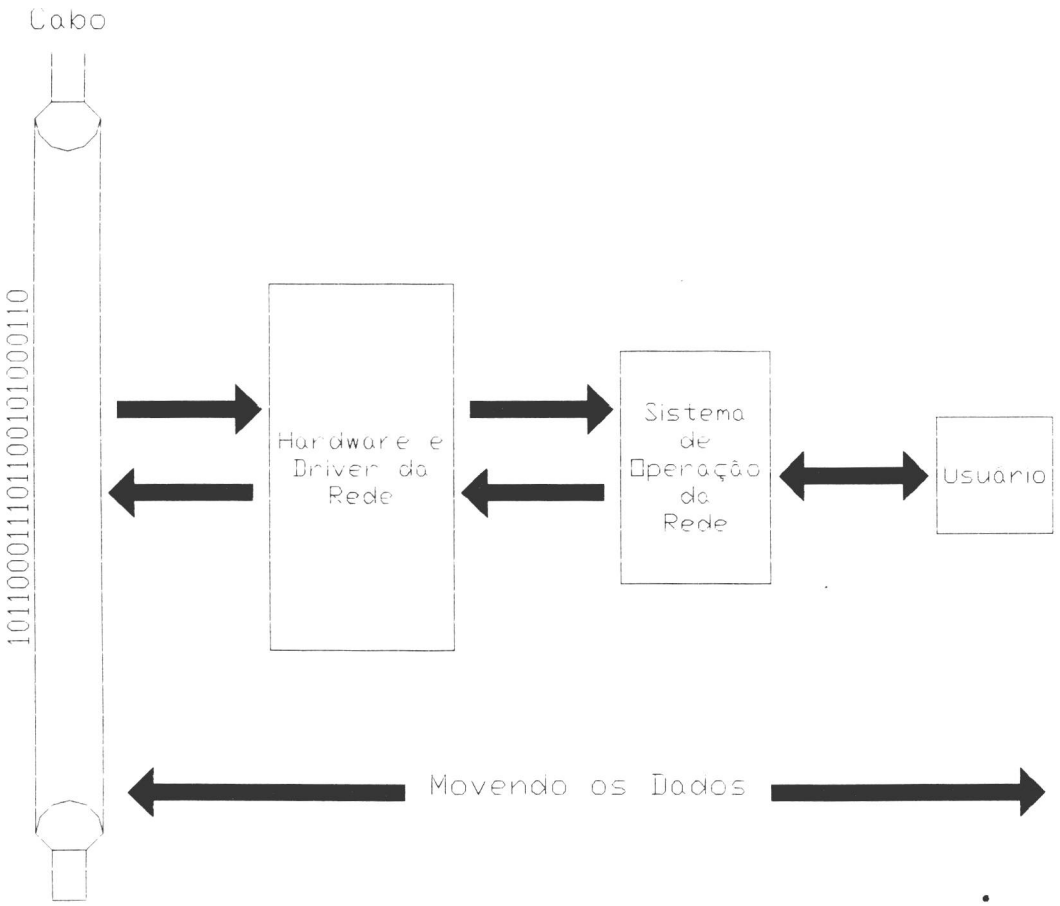


Figura 02 *– Habilidade de mover dados de/para o usuário*

Infelizmente estes simples objetivos acarretam em uma escolha dramática de diferentes projetos para o subsistema de interface Ethernet. A diversidade da arquitetura de computadores (tanto o "software" quanto o "hardware") requerem um único balanço de todos estes critérios.

No nosso caso, os 3 objetivos citados acima, serão atendidos mas o desempenho é que será um pouco prejudicado pois, como já foi dito, existem hoje no mercado, interfaces melhores que a Cheapernet. Entretanto, optamos por desenvolver uma interface Cheapernet por ser mais simples e fácil de implementar.

2.1 – Como é feita transação de uma informação na Rede:

Quando um usuário requer informações ou dados que não residam em seu próprio computador, ele deverá utilizar a rede para conseguir esta informação. Quando uma informação é solicitada na rede, ocorre uma complexa transação de sinais. O sistema de operação do computador do usuário diz ao protocolo da rede (CSMA/CD) para enviar uma mensagem para o servidor perguntando pela informação. O "Software" do protocolo então instrui o driver para enviar o requerimento para a interface do "hardware", que então envia os dados através do cabo para o servidor. Após o pacote ter sido reconhecido, um sinal de reconhecimento é enviado para o servidor indicando que a transferência foi executada com sucesso.

O processo reverso ocorre do servidor até o usuário. Quando o "hardware" recebe o requerimento, o "driver" é instruído para passá-lo para o protocolo da rede. O sistema de operação do computador recebe o requerimento do protocolo e insere uma resposta (os dados atuais) de que está enviando de volta através da rede em um modelo similar.

Todos os componentes do "software" e do "hardware" manipulam o requerimento e a resposta para se ter certeza da entrega dos dados de/para cada destino.

2.2 – Codificação Manchester:

Como já sabemos, o Cheapernet é uma interface de rede local que utiliza como padrão para transmitir um pacote na rede, o protocolo IEEE 802.3. Por sua vez o protocolo 802.3 utiliza a Codificação de Manchester para codificar os seus sinais na rede. Explicaremos neste item, o que vem a ser a codificação de Manchester e como ela funciona.

Nenhuma versão de 802.3 usa uma codificação binária com 0 Volts para bit "0" e 5 Volts para bit "1" porque pode gerar ambigüidades no momento em que recebemos estes sinais. Assim, tentou-se gerar uma não ambigüidade que determinava o início, o final e o meio de cada bit sem referências de um clock externo. Com a Codificação Manchester, cada período de bit é dividido em dois intervalos iguais. Um binário de bit "1" é enviado tendo o complemento do seu valor, ou seja "0", durante o primeiro intervalo e durante o segundo o valor verdadeiro, ou seja "1". Um binário "0" é justamente o reverso [10]. Observe o exemplo abaixo:

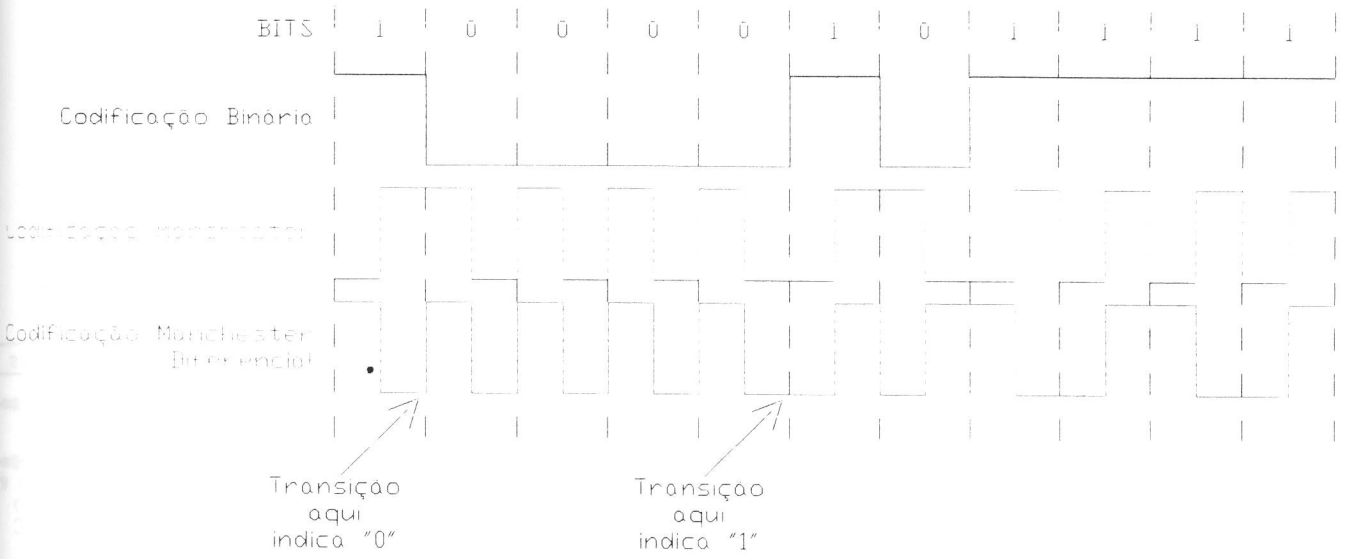


Figura 03 – *Exemplo de Codificação Manchester*

No Código Diferencial de Manchester requer um equipamento mais complexo, mas ao mesmo tempo oferece uma maior imunidade ao ruído. O sinal alto é + 0.85 Volts e o sinal baixo é – 0.85 Volts tendo um valor contínuo médio de 0 Volts.

2.3 – Encapsulamento de uma pacote a ser transmitido/

Desencapsulamento de um pacote recebido:

Um pacote típico do protocolo 802.3 consiste dos seguintes campos:

Preâmbulo, Delimitador de Início de Frame ("SFD – Start Frame Delimiter"), Endereço de Destino, Endereço de Recepção, Tamanho, Dados e Sequência de Checagem de Frame ("FCS – Frame Check Sequence") [1].

Na Figura 04, podemos ver como é o formato do frame Ethernet.

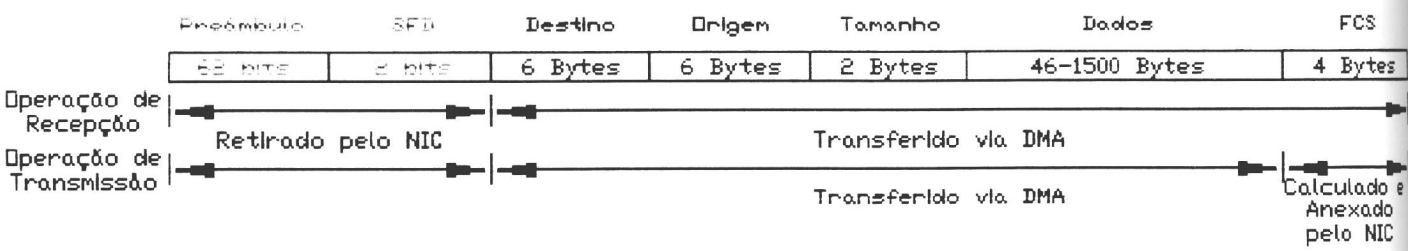


Figura 04 *– Formato do Frame Ethernet*

Durante a operação de recepção o Preâmbulo e o Delimitador de Início de Frame (SFD) – representados em vermelho – são retirados pela placa de rede e as informações restantes são transferidas via DMA – representadas em azul. Se tivermos uma operação de envio, a placa de rede será responsável pela geração do preâmbulo e do Delimitador de Frame (SFD) e também do FCS, que será acrescentado no final do frame a ser enviado.

Detalhamos agora as características de cada campo [1]:

- *Preâmbulo e Delimitador de Início de Frame:*

  O "SNI – Serial Network Interface – Interface de Rede Serial" (DP8391* – Observar Apêndice A) utiliza o campo de Preâmbulo de 1s e 0s alternados com a Codificação de Manchester para conseguir sincronização a nível de bits com o pacote que está chegando. Quando é transmitido, cada pacote possui um preâmbulo de 62 bits de 0s e 1s alternados. Parte desse preâmbulo pode ser perdido quando o pacote trafega pela rede. O campo de preâmbulo é retirado pela placa de rede. O alinhamento do pacote é realizado com o reconhecimento do padrão do Delimitador de Início de Frame (SFD) que consiste de dois 1s consecutivos. O NIC não trata o SFD como um byte, ele detecta apenas o padrão composto de dois 1s. Isto permite que qualquer preâmbulo que preceda o SFD seja usado como sincronizador.

- *Endereço de Destino:*

  O Endereço de Destino indica o destino de um pacote na rede e é utilizado para filtrar pacotes não desejados de entrarem no computador. Há três tipos de endereços suportados pela placa de rede: Físico (unicast), Multicast e Broadcast. O Endereço Físico é o endereço único que corresponde a um único computador. Todos os computadores possuem um MSB de "0". Estes endereços são comparados aos registros de endereços físicos armazenados internamente. Cada bit no endereço de destino deve combinar com o armazenado para que a placa de rede aceite o pacote. Endereços de Multicast começam com um MSB de "1". O

dispositivo DP8390** (NIC – Network Interface Controller – Controlador de Interface da Rede) filtra os endereços de multicast utilizando um algoritmo de mistura ("hashing") padrão que mapeia todos os endereços de multicast num valor de 6 bits. Se o endereço consiste somente de 1s, então ele é um endereço de broadcast indicando que aquele pacote é destinado a todos os nós. Se a placa de rede estiver no modo promíscuo, todos os pacotes serão aceitos: não é necessário que o campo de endereço de destino combine com nenhum dos registros armazenados. Os modos físico, multicast, broadcast e promíscuo podem ser selecionados.

- *Endereço de Origem:*

  O Endereço de Origem é o endereço físico do computador que enviou o pacote. O Endereço de Origem não pode ser nem um endereço de broadcast nem um endereço de multicast. Este campo é simplesmente transferido para a memória do buffer.

- *Campo de Tamanho de Dados:*

  Este campo de dois bytes indica o número de bytes que está dentro do campo de dados. A placa de rede não interpreta esse campo.

- *Campo de Dados:*

  Este campo tem o tamanho variando entre 46 bytes e 1500 bytes. Se uma mensagem tiver um tamanho superior a 1500 bytes seu conteúdo deverá ser dividido em vários pacotes. Se a mensagem for menor que 46 bytes a área de dados deverá ser preenchida até que o campo de dados fique com o tamanho mínimo de 46 bytes. Se houver este preenchimento, o número real de bytes de dados estará indicado no campo de tamanho da área de

---

* Não se preocupe ainda com estes dispositivos, pois eles serão visto com detalhes no capítulo 3.

dados. Não é a placa de rede que retira ou preenche o campo de bytes com o mínimo de bytes necessários. Nem é ela que checa pacotes com excesso de tamanho.

- *Campo de Verificação de Frame:*

A Seqüência de Verificação de Frame ("FCS – Frame Check Sequence") é um campo de 32 bits de CRC ("Cyclic Redundancy Check – Verificação por Redundância Cíclica") calculado e é acrescentado ao pacote durante a transmissão para que o computador destino consiga identificar erros durante o tráfego. Durante a recepção pacotes sem erros resultam num padrão conhecido de CRC. Pacotes com CRC errado serão rejeitados.

- *A Recepção de um pacote:*

Quando existe atividade na rede, os bits são transferidos para o Desserializador de recepção. O desserializador procura o SFD com seu detector de sincronismo depois do preâmbulo e transfere os 6 bytes que vem em seguida para dentro da Lógica de Reconhecimento de Endereços, para certificar-se que aquele pacote pertence ao computador local. O pacote também é posicionado no Verificador/Gerador de CRC para certificação de que é um pacote válido. A cada 8 sinais de clock, um byte é transferido para a FIFO ("First In First Out – Primeiro a entra é o primeiro a sair") de 16 bytes e o Contador de Bytes é incrementado. Se a Lógica de Reconhecimento de Endereços ou o Verificador/Gerador de CRC não reconhecer o pacote como um pacote válido, o pacote é descartado e a FIFO é apagada. A Lógica de Controle de DMA (Direct

Memory Access – Lógica de Acesso Direto à Memória") é responsável pela transferência dos dados da FIFO para a memória do computador.

- *A Transmissão de um pacote:*

  Quando existem dados na FIFO para serem transmitidos, o Serializador de Transmissão é ativado. Ele lê os dados paralelamente da FIFO e os serializa para envio. O pacote também é enviado para o Verificador/Gerador de CRC, para que seja gerado um FCS que será colocado no fim do pacote. Antes do envio o gerador de Padrões de Preâmbulo e Sincronização gera o preâmbulo (62 bits) e a sincronização (2 bits em 1).

2.4 - Conclusão:

Neste capítulo vimos os principais aspectos gerais de uma placa de rede ethernet. Vimos como deve ser o desempenho da placa, partindo do principal foco de eficiência do funcionamento de uma placa que é a de mover dados com alta segurança na transmissão.

Depois disso vimos como são feitas a transação de informações na rede (através de uma visão geral), ou seja, como são feitas as trocas de sinais de controle entre os computadores até que ambos, servidor e receptor do pacote, estejam prontos para respectivamente enviar e receber o pacote.

Em relação à codificação do sinal na rede, foi visto que é necessário uma codificação melhor que a codificação binária (0 Volts para bit "0" e 5 Volt para bit "1") para minimizar a probabilidade de ocorrência de ambiguidades e ruídos do sinal.

Assim, o protocolo IEEE 802.3 utilizou a Codificação Manchester para tentar atingir aos objetivos citados.

No item anterior vimos como é feito o encapsulamento de um pacote a ser transmitido e o desencapsulamento de um pacote a ser recebido. Na operação de transmissão vimos que o NIC anexa tanto o preâmbulo e o SFD ao pacote a ser enviado para sincronização do receptor quanto o FCS que deve garantir a segurança de que o pacote chegue em perfeitas condições na estação receptora. Na operação de recepção o NIC retira o preâmbulo e o SFD depois de sincronizado o pacote e verifica a sua integridade através do FCS, se por um acaso este pacote estiver com qualquer tipo de erro (pacote anão, ou erro de alinhamento, ou erro de CRC, etc.), ou ainda erro de destino (ou seja, o pacote não é destinado àquele computador), este pacote deverá ser descartado pela estação receptora.

Capítulo 3

3.0 - Descrição Geral do Hardware de um Cheapernet:

O "hardware" desenvolvido na interface Cheapernet utiliza o circuito integrado DP8390D/NS32490D que é um dispositivo de Controlador de Interface da Rede ("NIC – Network Interface Controller") desenvolvido para facilitar a interface, usando o CSMA/CD, com uma rede local (LAN), incluindo Ethernet, Thin Ethernet (Cheapernet) e StarLAN. O NIC implementa todas as funções de transmissão e recepção dos pacotes de acordo com o padrão IEEE 802.3 nas camadas de Controle de Acesso ao Meio ("MAC Medium Access Control"). Uma memória FIFO interna e um canal DMA fornecem uma simples e eficiente administração do pacote.

O NIC é o coração dos três (CI's) que implementam o protocolo completo IEEE 802.3. É mostrado abaixo em sistema de diagrama de blocos uma rede local Thin Ethernet. Os outros CI's envolvidos são o DP8391 – Interface de Rede Serial ("SNI – Serial Network Interface") e o DP8392 – Interface para Transmissão Coaxial ("CTI – Coaxial Transceiver Interface") [6].

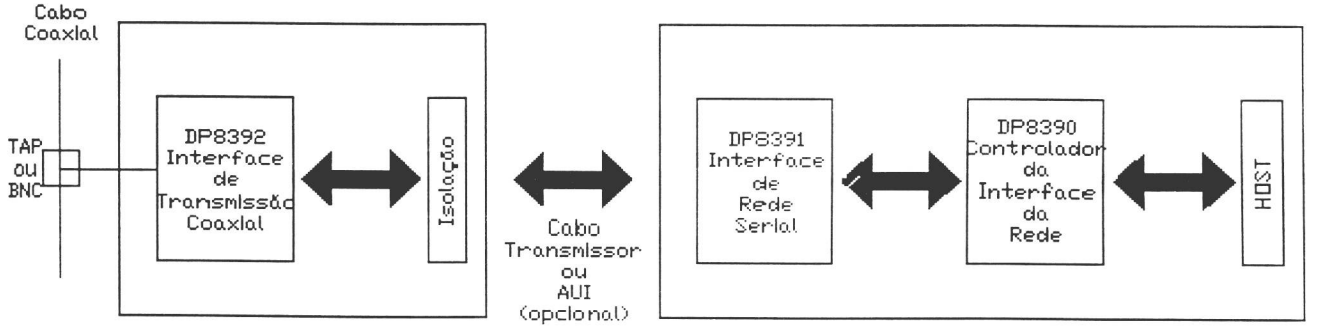


Figura 05 - *Diagrama de Blocos para uma Rede Local Cheapernet.*

As principais características da NIC são [1]:

- Compatibilidade com IEEE 802.3 para Ethernet. Thin Ethernet e StarLAN.
- Interface com microprocessadores de sistemas de 8, 16 e 32 bits.
- Implementação simples e versatilidade na administração do buffer.
- Requer alimentação simples de 5V.
- Inclui:
  - Dois canais DMA de 16 bits.
  - Uma FIFO interna de 16 bytes com tamanho de palavra programável.
  - Um armazenamento de estatísticas da rede.
- Fornece um filtro para endereçamentos físicos, multicast e broadcast.
- Providencia 3 níveis de verificação do funcionamento dos dispositivos da rede ("loopback").
- Utiliza um sistema independente de clocks da rede.

O "hardware" desenvolvido na interface Cheapernet também utiliza o circuito integrado DP8391D/NS32491D, um dispositivo que é uma Interface de Rede Serial ("SNI – Serial Network Interface") e tem a função de codificação e decodificação dos dados em Códigos de Manchester. O SNI providencia uma interface entre a NIC (DP8390) e o CTI (DP8391).

Há uma isolação entre o CTI e o SNI. Isto é feito usando um conversor DC-DC (ver apêndice B) que transforma os 5 Volts de alimentação da placa em –9 Volts de tensão necessário para comunicar a placa com o cabo coaxial.

As principais características do SNI [9] [11] [12] são:

- Compatibilidade com IEEE 802.3 para Ethernet (10BASE5) e Cheapernet (10BASE2).
- 10 Mbps de clock para a codificação e decodificação Manchester.
- Possui um decodificador digital com loop fechado (DPLL), não precisando de componentes externos.
- Capacidade para diagnósticos de "loopback".
- Modos de operações para a transmissão na saída externamente selecionáveis com etapas completas ou com metade das etapas.
- Circuitos "squelch" (pesados, com alta impedância) para as rejeições de ruídos na recepção e de colisões na entrada.
- Conectividade direta com o cabo transmissor ("AUI – Attachment Unit Interface – Interface da Unidade de Conexão").

Um outro dispositivo de "hardware" importante na interface Cheapernet é o DP8392/NS32492 que é um dispositivo de Interface para Transmissão Coaxial ("CTI – Coaxial Transceiver Interface") e tem a função de transmitir/receber para a/da linha do cabo coaxial para redes locais Cheapernet e Ethernet.

Nas aplicações Cheapernet, o CTI localiza-se no DTE ("Data Terminal Equipament = Equipamento Terminal de Dados") e é conectado ao DTE através de um transformador para isolação do sinal.

As principais características do CTI [13] são:

- Compatibilidade com IEEE 802.3 para Ethernet (10BASE5) e Cheapernet (10BASE2).
- Integrar toda a transmissão eletrônica, exceto a isolação dos sinais e da força.
- Operações que permitem a seleção externa do "CD Heartbeat" (será visto com detalhes mais a frente) compatíveis com o IEEE 802.3.
- Precisão do circuito no modo de detecção de colisão.
- Circuito squelch (com alta impedância) que rejeita os ruídos na entrada.
- Usa um padrão de 16 pinos DIP na saída da linha que reduz significantemente a temperatura nos componentes.

3.1 – Controlador da Interface da Rede (NIC):

Para entendermos melhor o funcionamento do NIC, a descrição ficará melhor compreendida através do diagrama de blocos. Neste próximo item será explicado como é o funcionamento de uma placa Ethernet típica. É importante referir-se à figura do hardware no Apêndice A sempre que comentarmos algum elemento deste.

3.1.1 - Descrição Funcional de uma interface Ethernet típica:

De acordo com a figura 06, comentaremos os principais blocos de elementos para o funcionamento da interface [1].

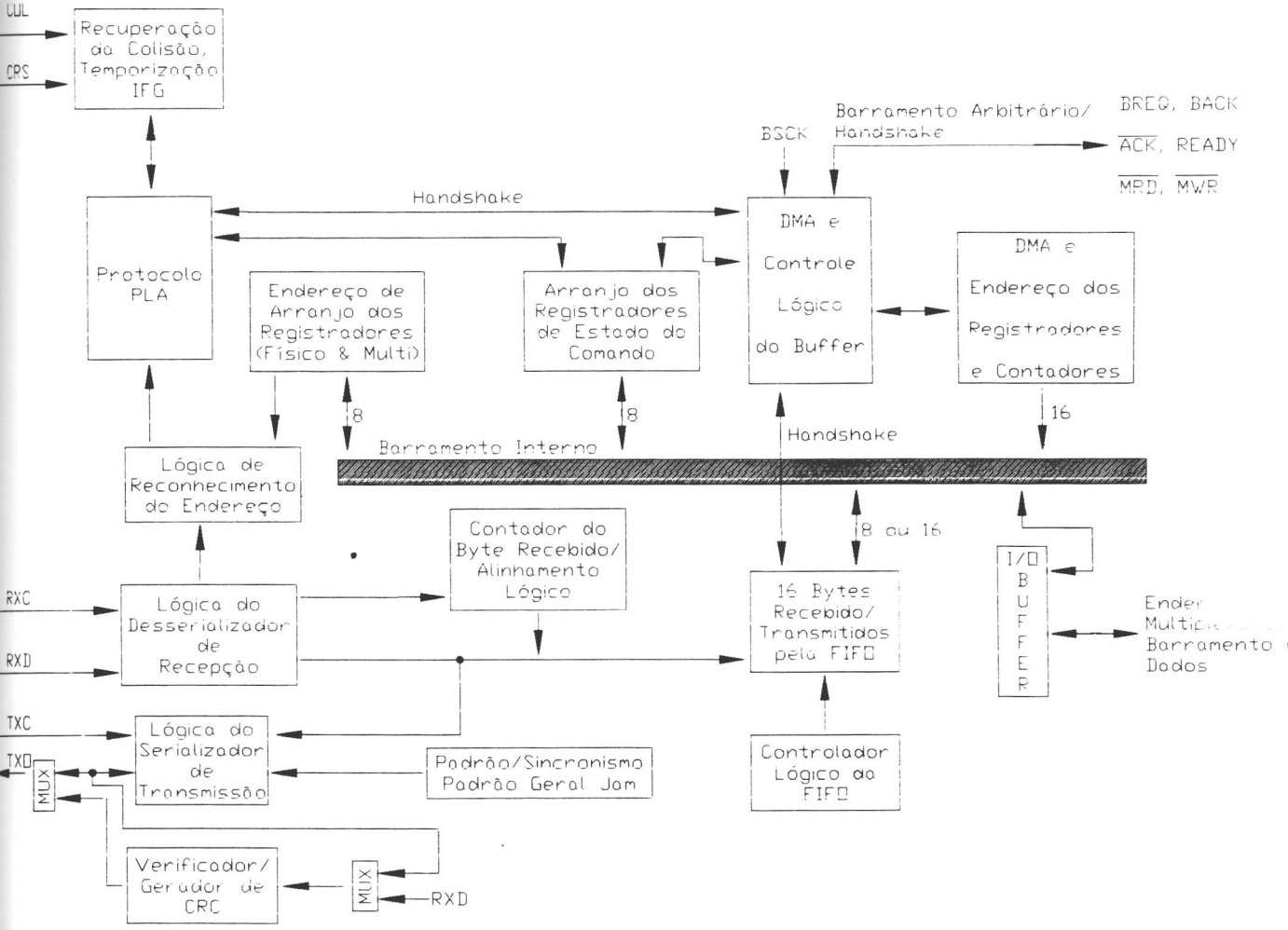


Figura 06 *– Diagrama de Blocos de uma interface Ethernet típica.*

- *Desserializador de Recepção:*

Inicialmente os dados de recepção seriais são colocados no verificador/gerador de CRC. O desserializador de recepção também inclui um detector de sincronismo que detecta o SFD ("Start Frame Delimiter – Delimitador de Início de Frame") para estabelecer onde estão localizados os limites do byte dentro da cadeia de bits seriais que compõe o preâmbulo. Sempre após oito clocks de recepção, os dados no tamanho de um byte são transferidos para a FIFO de 16 bytes e o Contador de

Bytes recebidos é incrementado. Os primeiros 6 bytes depois do SFD são testados comparando-se com a Lógica de Reconhecimento de Endereço. Se a Lógica de Reconhecimento de Endereço não reconhece o pacote como um pacote destinado para aquele computador, a FIFO é apagada.

- *Verificador/Gerador de CRC:*

Durante a transmissão, a lógica de CRC gera um campo local de CRC para a sequência de bits transmitida. O CRC codifica todos os bytes depois de sincronização . O CRC é transmitido para fora do MSB acompanhando o último byte transmitido. Durante a recepção a lógica de CRC gera um campo de CRC para o pacote que está sendo recebido. Este CRC local é serialmente comparado com o CRC colocado no fim do pacote enviado pelo computador de origem. Se ambos os campos de CRC forem iguais, um padrão específico será gerado e decodificado para indicar que nenhum erro nos dados foi encontrado. Erros de transmissão dão como resultado um padrão diferente e assim são identificados, resultando na rejeição do pacote.

- *Serializador de Transmissão:*

O Serializador de Transmissão lê os dados paralelamente da FIFO e os serializa para transmissão. O serializador é sincronizado pelo clock de transmissão gerado pela Interface Serial de Rede (DP8391). Os dados seriais de entrada também são enviados ao verificador/gerador de CRC. No início de cada transmissão o gerador de Sincronização e de Preâmbulo acrescenta 62 bytes de 1s e 0s com função de preâmbulo. Depois que o último byte de dados do pacote foi serializado, o FCS de 32 bits é diretamente retirado do gerador de CRC. No caso de uma colisão o

gerador de Preâmbulo e sincronização é utilizado para gerar um padrão de colisão ("JAM") formado por bits de nível 1.

- *Lógica de Reconhecimento de Endereços:*

  A lógica de Reconhecimento de Endereços compara o campo de Endereço Destino (primeiros 6 bytes do pacote recebido) com os Registros de Endereço Físico armazenados no Arranjo de Registro de Endereço. Se qualquer um dos 6 bytes não combinar com o endereço físico pré – programado, a Lógica de Controle de Protocolo rejeita o pacote. Todos os endereços de destino de "Multicast" são filtrados utilizando uma técnica de "hashing" (mistura). Se o endereço de multicast indexar um bit que foi registrado no filtro de endereços de multicast, o pacote é aceito, caso contrário é rejeitado pela Lógica de Controle de Protocolo. Cada Endereço de destino também é checado com todos os 1's que são reservados para o endereço de broadcast.

- *A FIFO e a Lógica de controle da FIFO:*

  A placa de rede possui uma FIFO de 16 bytes. Durante a transmissão o DMA escreve os dados diretamente na FIFO e o *Serializador de Transmissão* lê dados da FIFO para transmiti-los. Durante a recepção o *Desserializador de Recepção* escreve dados na FIFO e o DMA lê dados da FIFO. A Lógica de Controle da FIFO é usada para contar o número de bytes na FIFO para que depois de um nível pré – ajustado, o DMA possa começar a utilizar um acesso à via de dados ("BUS") e ler/escrever dados da/para FIFO antes que uma sub carga/sobre carga ("underflow/overflow") ocorra.

Devido ao fato da interface de rede precisar guardar o campo de endereços de cada pacote que chegar para determinar se o pacote combina com os Registros de Endereço Físico ou se este mapeia um dos endereços nos registradores de endereços de Multicast, a primeira transferência local de DMA não ocorre até que 8 bytes tenha sido acumulados na FIFO.

Para garantir que não haja nenhuma sobreposição de dados na FIFO, a lógica da FIFO registra uma sobrecarga da FIFO quando o 13° byte for escrito na FIFO. Este procedimento efetivamente reduz a FIFO para 13 bytes. Além disso a lógica da FIFO opera diferentemente no "Modo de Byte" e no "Modo de Palavra" (2 bytes). No "Modo de Byte", o início é indicado quando o byte n+1 entrou na FIFO, assim, para um início de 8 bytes, a interface de rede envia um pedido de Requerimento do Barramento de Dados ("Bus Request" - BREQ) quando o 9° byte entrar na FIFO. Para o "Modo de Palavra", o BREQ não é ativado até que o byte n+2 tenha entrado na FIFO. Assim, com um início de 4 palavra (equivalente a um início de 8 bytes). O BREQ é ativado quando o 10° byte entrar na FIFO.

- *PLA de Protocolo:*

O PLA de protocolo é responsável pela implementação do protocolo 802.3, incluindo a recuperação de estados de colisão com atrasos dinâmicos. O PLA de Protocolo também formata o pacote durante a transmissão e retira o preâmbulo e o sinal de sincronização durante a recepção.

- *Lógica de Controle de Buffer e DMA:*

A lógica de controle de buffer e de DMA é utilizada para controlar dois canais de 16 bits DMA. Durante a recepção, o DMA local armazena os pacotes num buffer de recepção em forma de anel, localizado na memória de "buffers". Durante a transmissão o DMA local usa o ponteiro programado e os registradores de tamanho para transferir um pacote da memória do buffer local para a FIFO. O segundo canal de DMA é utilizado como um DMA escravo para transferir os dados da memória do buffer local para o computador. O DMA local e o DMA Remoto são arbitradas internamente, com o canal de DMA local tendo a maior prioridade. Ambos os canais de DMA utilizam um clock externo para gerar toda a temporização da via de dados. A arbitragem interna é realizada com uma chamada de barramento comum, o protocolo de comunicação de via de dados.

3.1.2 – Registradores Internos:

Mostraremos aqui, unicamente como são acessados os registradores. O funcionamento bit a bit de cada registrador está descrito na folha de dados do Controlador de Rede (NIC) da National [1].

Todos os registradores tem 8 bits de largura e são mapeados em duas páginas, que são selecionadas no registrador de comando (PS0 e PS1). Os pinos de RA0 – RA3 são usados para endereçar os registradores em cada página (observar estes pinos do 8390 (NIC) no apêndice A). Os registradores na Página 0 são aqueles registradores que são normalmente acessados durante a operação do NIC enquanto os

registradores da Página 1 são usados primeiramente para inicialização. Estes registradores são partidos para permitir uma execução em dois ciclos de leitura/escrita para acessar normalmente os registradores usados.

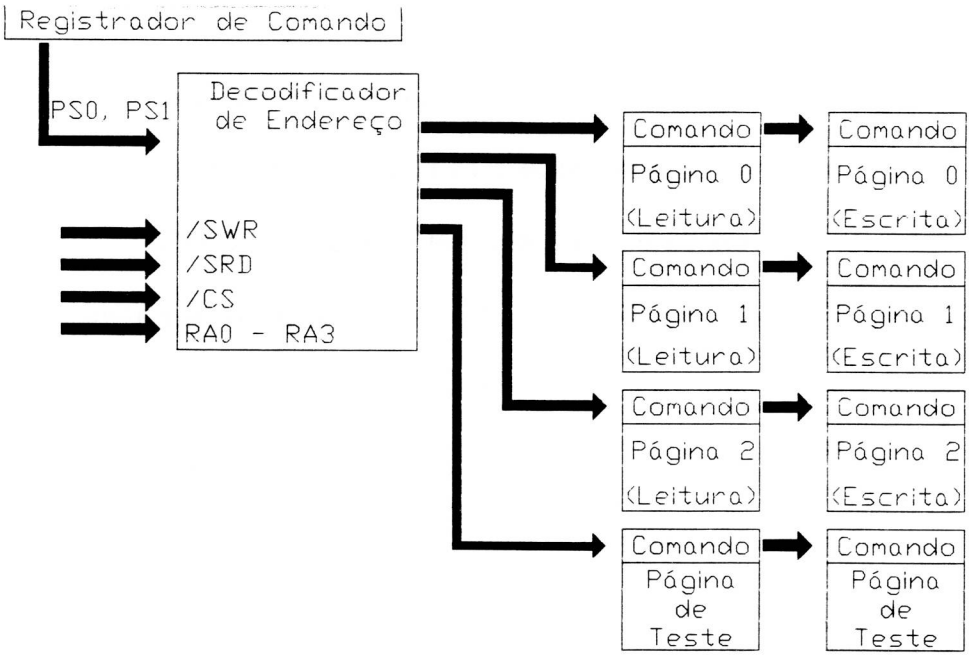


***Figura 07** – Mapa do Endereçamentos dos registradores.*

Nas tabelas abaixo, veremos o endereçamento dos registradores de acordo com o valor de PS0 e PS1. Na tabela II, veremos o endereçamento para Página 0. Na tabela III o endereçamento para Página 1 e na tabela IV o endereçamento para o Página 2.

Tabela II *– Endereços Designados na Página 0 (PS1 = 0, PS0 = 0)*

RA0 – RA3	RD	WR
00H	Comando (CR)	Comando (CR)
01H	Endereço Corrente de DMA Local 0 (CLDA0)	Registrador de Mensagem Inicial (PSTART)
02H	Endereço Corrente de DMA Local 1 (CLDA1)	Registrador de Mensagem Final (PSTOP)
03H	Ponteiro Limite (BNRY)	Ponteiro Limite (BNRY)
04H	Registrador de Estado de Transmissão (TSR)	Endereço Inicial da Mensagem para a Transmissão (TPSR)
05H	Registrador de Número de Colisões (NCR)	Registrador Contador de Byte Transmitido 0 (TBCR0)
06H	FIFO (FIFO)	Registrador Contador de Byte Transmitido 1 (TBCR1)
07H	Registrador de Estado de Interrupção (ISR)	Registrador de Estado de Interrupção (ISR)
08H	Endereço de DMA Remoto Corrente 0 (CRDA0)	Registrador de Endereço Inicial Remoto 0 (RSAR0)
09H	Endereço de DMA Remoto Corrente 1 (CRDA1)	Registrador de Endereço Inicial Remoto 1 (RSAR1)
0AH	Reservado	Registrador Contador de Byte Remoto 0 (RBCR0)
0BH	Reservado	Registrador Contador de Byte Remoto 1 (RBCR1)
0CH	Registrador de Estado de Recepção (RSR)	Registrador de Configuração de Recepção (RCR)
0DH	Contador de Registro 0 (Erros de Alinhamento do Frame) (CNTR0)	Registrador de Configuração de Transmissão (TCR)
0EH	Contador de Registro 1 (Erros de CRC) (CNTR1)	Registrador de Configuração de Dados (DCR)
0FH	Contador de Registro 2 (Erros de Pacotes Perdidos) (CNTR2)	Registrador de Máscara de Interrupção (IMR)

Tabela III *– Endereços Designados no Página 1 (PS1 = 0, PS0 = 1)*

RA0 – RA3	RD	WR
00H	Comando (CR)	Comando (CR)
01H	Registrador de Endereço Físico 0 (PAR 0)	Registrador de Endereço Físico 0 (PAR 0)
02H	Registrador de Endereço Físico 1 (PAR 1)	Registrador de Endereço Físico 1 (PAR 1)
03H	Registrador de Endereço Físico 2 (PAR 2)	Registrador de Endereço Físico 2 (PAR 2)
04H	Registrador de Endereço Físico 3 (PAR 3)	Registrador de Endereço Físico 3 (PAR 3)
05H	Registrador de Endereço Físico 4 (PAR 4)	Registrador de Endereço Físico 4 (PAR 4)
06H	Registrador de Endereço Físico 5 (PAR 5)	Registrador de Endereço Físico 5 (PAR 5)
07H	Registrador de Mensagem Corrente (CURR)	Registrador de Mensagem Corrente (CURR)
08H	Registrador de Endereço "Multicast" 0 (MAR 0)	Registrador de Endereço "Multicast" 0 (MAR 0)
09H	Registrador de Endereço "Multicast" 1 (MAR 1)	Registrador de Endereço "Multicast" 1 (MAR 1)
0AH	Registrador de Endereço "Multicast" 2 (MAR 2)	Registrador de Endereço "Multicast" 2 (MAR 2)
0BH	Registrador de Endereço "Multicast" 3 (MAR 3)	Registrador de Endereço "Multicast" 3 (MAR 3)
0CH	Registrador de Endereço "Multicast" 4 (MAR 4)	Registrador de Endereço "Multicast" 4 (MAR 4)
0DH	Registrador de Endereço "Multicast" 5 (MAR 5)	Registrador de Endereço "Multicast" 5 (MAR 5)
0EH	Registrador de Endereço "Multicast" 6 (MAR 6)	Registrador de Endereço "Multicast" 6 (MAR 6)
0FH	Registrador de Endereço "Multicast" 7 (MAR 7)	Registrador de Endereço "Multicast" 7 (MAR 7)

Tabela IV *– Endereços Designados no Página 2 (PS1 = 1, PS0 = 0)*

RA0 – RA3	RD	WR
00H	Comando (CR)	Comando (CR)
01H	Registrador de Mensagem Inicial (PSTART)	Endereço Corrente de DMA Local 0 (CLDA 0)
02H	Registrador de Mensagem Final (PSTOP)	Endereço Corrente de DMA Local 1 (CLDA 1)
03H	Ponteiro para o Próximo Pacote Remoto	Ponteiro para o Próximo Pacote Remoto
04H	Endereço Inicial da Mensagem para a Transmissão (TPSR)	Reservado
05H	Ponteiro para o Próximo Pacote Local	Ponteiro para o Próximo Pacote Local
06H	Contador de Endereço (Superior)	Contador de Endereço (Superior)
07H	Contador de Endereço (Inferior)	Contador de Endereço (Inferior)
08H	Reservado	Reservado
09H	Reservado	Reservado
0AH	Reservado	Reservado
0BH	Reservado	Reservado
0CH	Registrador de Configuração de Recepção (RCR)	Reservado
0DH	Registrador de Configuração de Transmissão (TCR)	Reservado
0EH	Registrador de Configuração de Dados (DCR)	Reservado
0FH	Registrador de Máscara de Interrupção (IMR)	Reservado

Duas observações são importantes aqui. Primeiramente, os Registradores da Página 2, unicamente deverão ser acessados para propósitos de diagnóstico. Eles não deverão ser modificados durante operação normal. E também, não deveremos nunca modificar a Página 3.

3.2 - Interface Serial da Rede (SNI):

O SNI consiste principalmente dos seguintes blocos de lógica [9] [11] [12]:

1. *Oscilador:* gera os 10 MHz de sinal de clock para transmitir no Sistema.
2. *O Codificador Manchester e o driver de saída diferencial:* tem as funções de: aceitar os dados NRZ ("NON RETURN TO ZERO = SEM RETORNO A ZERO") do controlador; realizar uma codificação Manchester e transmiti-los diferencialmente para o transmissor.
3. *Decodificador Manchester:* recebe dados Manchester do transmissor, converte-os em dados NRZ e pulsa o clock enviando-os para o controlador.
4. *Tradutor de Colisão:* indica para o controlador a presença de um sinal de 10 MHz válido na sua entrada.
5. *O Circuito de "Loopback":* quando for inserido, é chaveado a codificação dos dados ao invés de receber sinais de entrada.

Neste próximo item será explicado como é o funcionamento do SNI.

Também é importante a observação do "hardware" no Apêndice A, sempre que se achar necessário para o melhor entendimento.

3.2.1 – Descrição Funcional:

Como já foi dito anteriormente, o SNI consiste de 5 blocos de lógica principais: *oscilador, codificador e decodificador Manchester, tradutor de colisões e "loopback"* [9]. Veremos com um pouco mais de detalhes agora, observando sempre a figura 08.

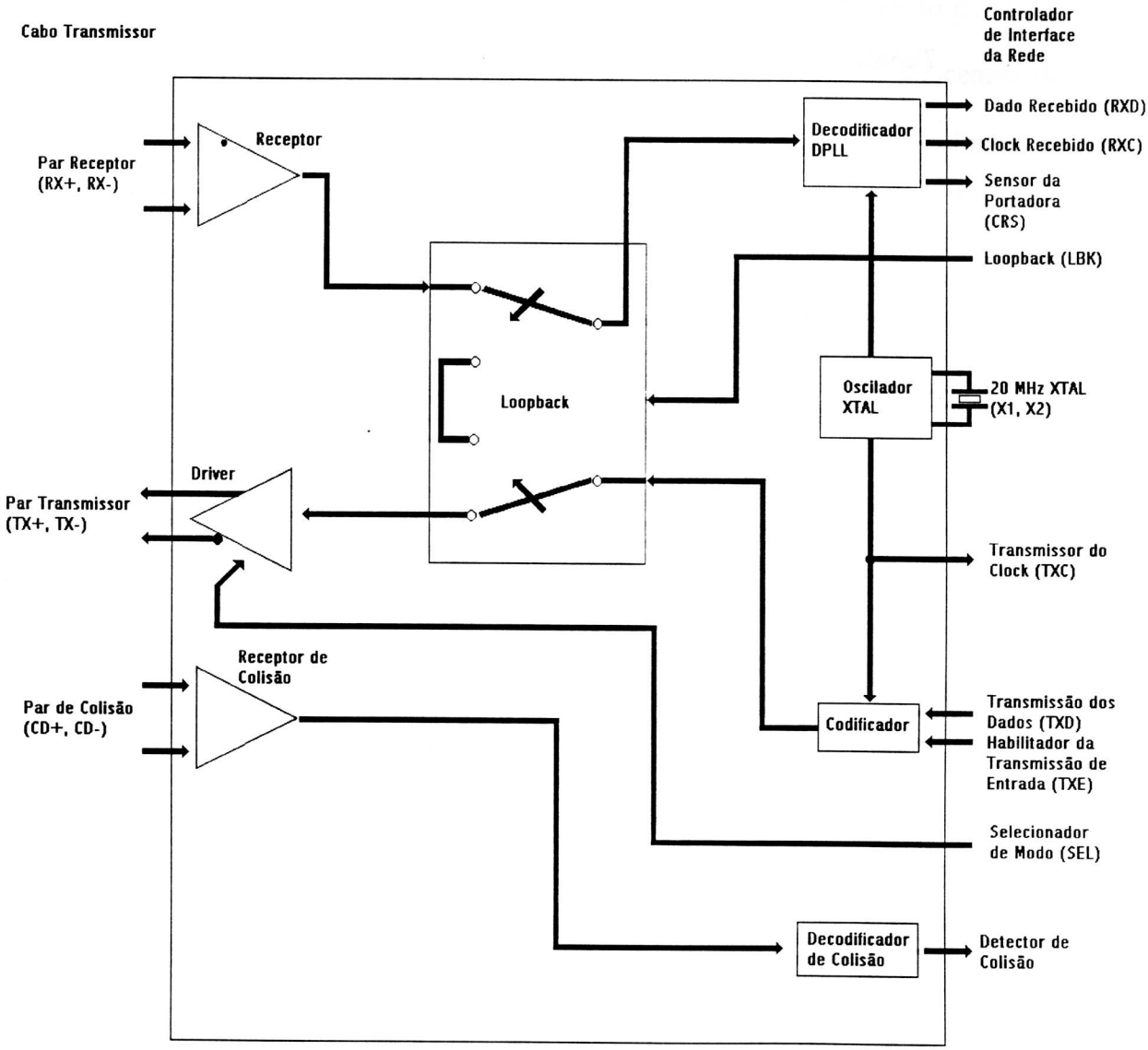


Figura 08 *– Diagrama de Blocos do funcionamento interno do SNI.*

- *Oscilador:* é controlado por um cristal de ressonância paralela conectado entre X1 e X2 ou por um clock externo em X1. O oscilador de 20 MHz é dividido por 2 para gerar os 10 MHz. O oscilador também gera um sinal de clock interno para a codificação e decodificação dos circuitos. Os 20 MHz de conexão com um cristal no SNI requer um cuidado especial, citados na tabela V.

Tabela V *– Especificações do Cristal*

Variável	**Especificação Desejada**
Freqüência de Ressonância	20 MHz
Tolerância	± 0.001% à 25°C
Estabilidade	± 0.005% 0 - 70°C
Circuito	Ressonância Paralela

No nosso caso, não utilizaremos um Cristal como clock e sim um clock externo, portanto, não nos preocuparemos com estas especificações.

- *Codificador Manchester e Driver Diferencial:* o codificador combina as informações do clock e dos dados para o transmissor. Os dados codificados e a transmissão começam com a habilitação do transmissor de entrada (TXE) indo para alto (Observar Figura 09). Enquanto o TXE permanecer alto o dado transmitido TXD é codificado na saída do par transmissor-driver (TX ±). A transmissão é finalizada com a habilitação do transmissor de entrada indo para baixo. A última transição é sempre positiva no par do transmissor de saída. Ela ocorrerá no centro da posição do bit se o último bit for um (Figura 10(b)) ou no limite da posição do bit se o último bit for zero (Figura 10(a)).

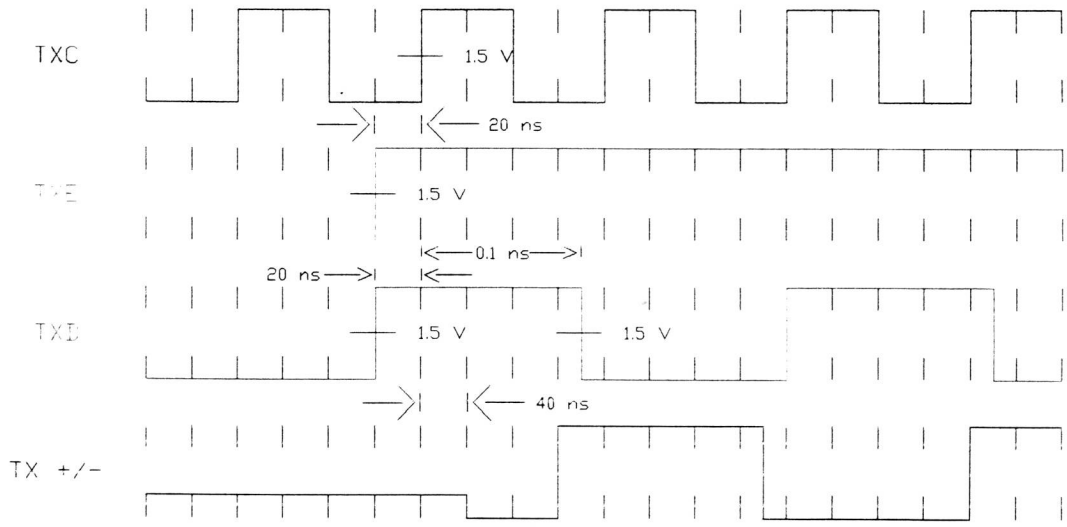


***Figura 09** – Temporização da transmissão – Início da Transmissão*

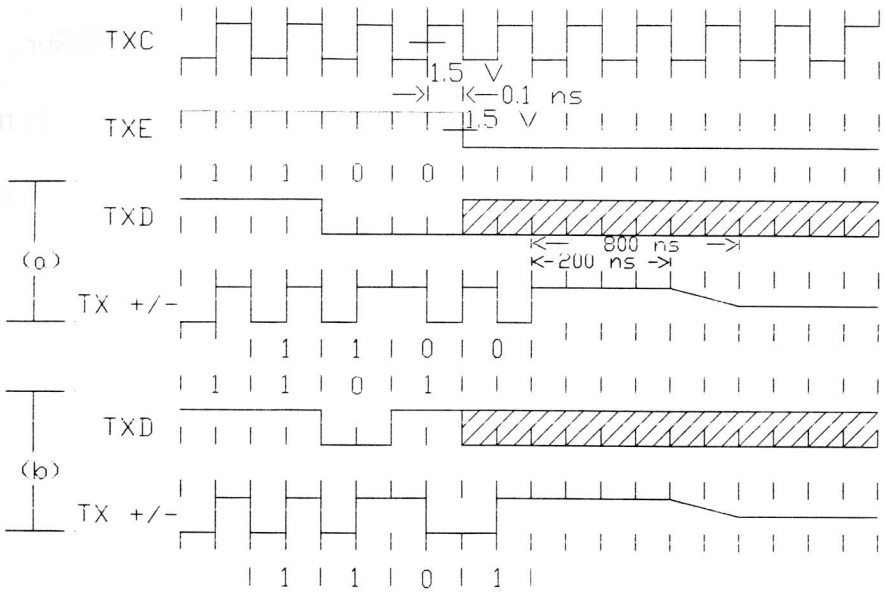


Figura 10(a) – *Temporização da transmissão – Fim da Transmissão (último bit 0)*

Figura 10(b) – *Temporização da transmissão – Fim da Transmissão (último bit 1)*

- *Decodificador Manchester:* o decodificador consiste de um circuito de entrada diferencial e um bloqueador de fase digital ("digital phase-locked loop") para separar a codificação Manchester do fluxo de dados introduzindo sinais de "clock" e dados NRZ. Um circuito de alta impedância (chamado de "circuit squelch") na entrada rejeita sinais com largura de pulso menor que 8 ns (negativo) ou com níveis menores que – 175 mV. Sinais mais negativos que – 300mV e com uma duração maior que 30 ns são sempre decodificados. Isto previne ruídos na entrada de um falso gatilho ("triggering") no decodificador na ausência de um sinal válido. Uma vez que for excedido o valor da entrada do "circuito

squelch", o sensor da portadora ("CRS - carrier senser") é sensibilizado e ligado. O dado recebido (RXD) e o clock recebido (RXC), estão disponíveis a cada 6 bits de clock. Neste ponto, o bloqueador de fase digital interrompe os sinais de chegada. O DP8391 decodifica um pacote ("frame") de dado em torno de 20 ns. O decodificador detecta o final do pacote ("frame") quando cessa-se a transição normal do bit médio na entrada diferencial. Com o tempo de um bit e uma metade após o último bit o sensor da portadora é sensibilizado e desligado. O clock recebido fica ativo por mais 5 bits de tempo antes dele cair e permanecer baixo até o próximo frame. As figuras 11, 12 e 13 ilustram a temporização da recepção.

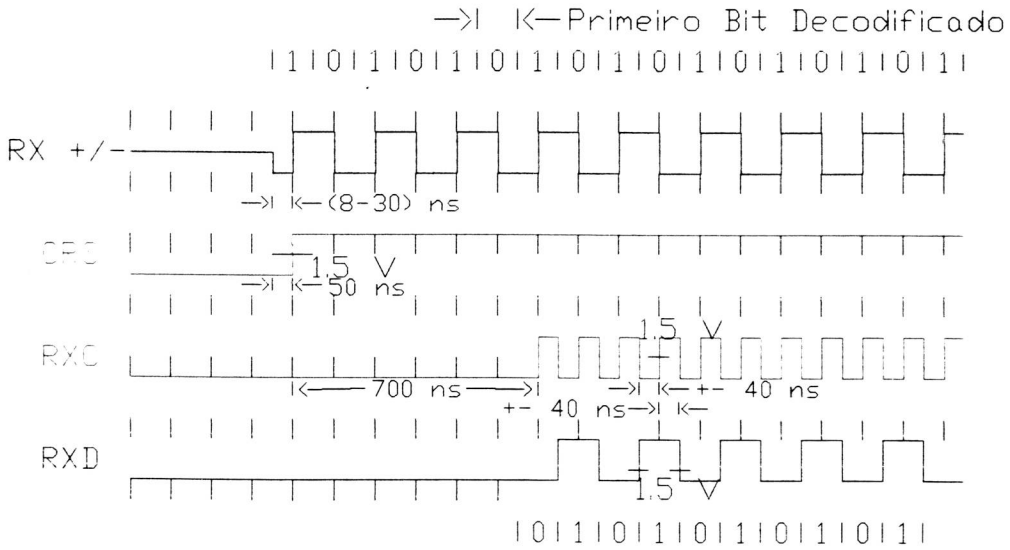


***Figura 11** – Temporização da recepção – Início do Pacote.*

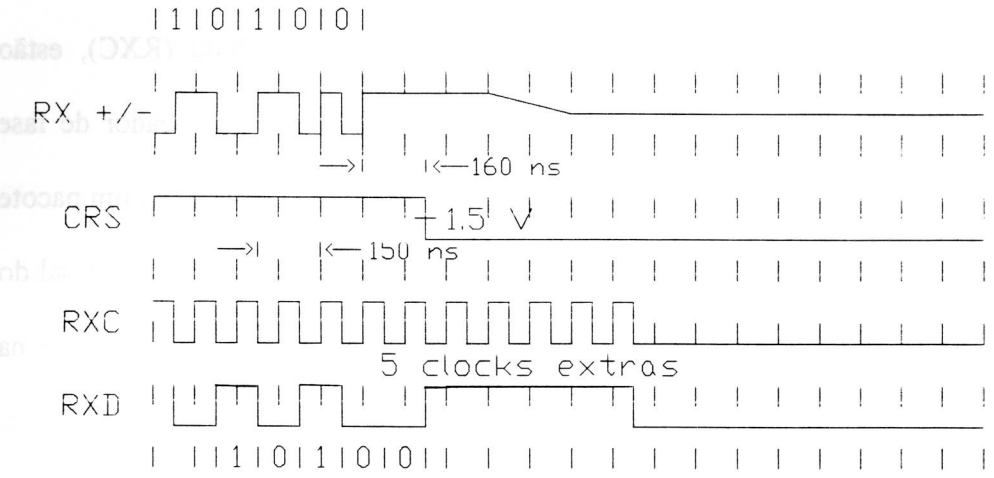


***Figura 12** –Temporização da recepção – Final do Pacote (último bit = 0).*

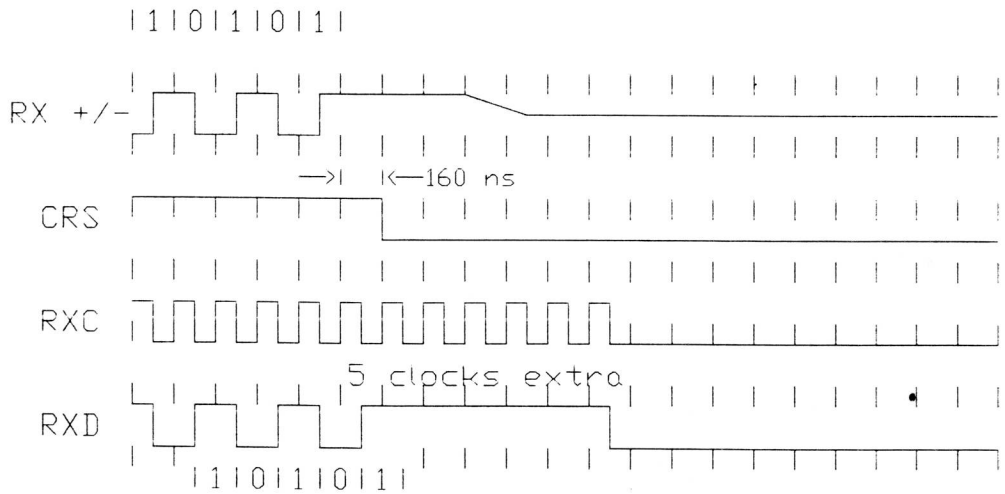


***Figura 13** –Temporização da recepção – Final do Pacote (último bit = 1).*

- *Tradutor de Colisão:* o transmissor Ethernet detecta colisões no cabo coaxial e gera 10 MHz de sinal no cabo transmissor. O tradutor de colisão SNI declara, na saída do detector de colisão (COL), para o

controlador DP8390 quando está presente uma colisão na entrada do sinal de 10MHz. O controlador usa este sinal para auxiliar na transmissão e "reciclá-lo". O detector de colisão na saída é sensibilizado e "desligado" em torno de 350 ns após os 10MHz do sinal da entrada parar. A colisão diferencial da entrada (+ e -) deverá ser finalizada do mesmo jeito que o receptor de entrada. A colisão de entrada também tem um circuito de alta impedância ("circuit squelch") que rejeita sinais com uma largura de pulso menor que 8 ns (negativo) ou com níveis menores que ± 175 mV. A figura 14 ilustra a temporização da colisão.

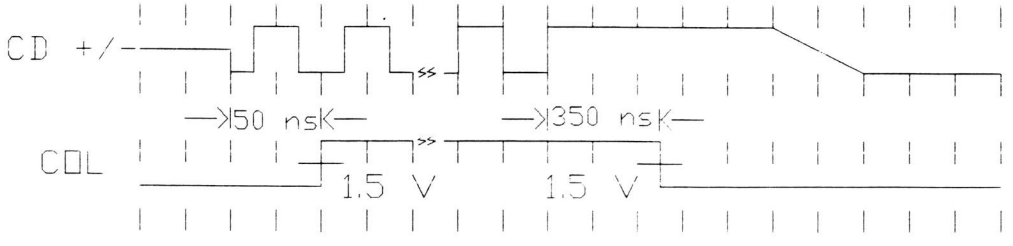


Figura 14 *– Diagrama de temporização da colisão*

- *Funções de "Loopback":* Uma lógica alta na entrada do loopback (LBK) causa um roteamento do dado serial do SNI na entrada do transmissor do dado, através do seu codificador, retornando-o através do decodificador do bloqueador de fase para receber dados na saída. No

modo loopback, o driver de transmissão está no estado ocioso ("idle") e a entrada do circuito do receptor é desabilitado. Observar figura 15.

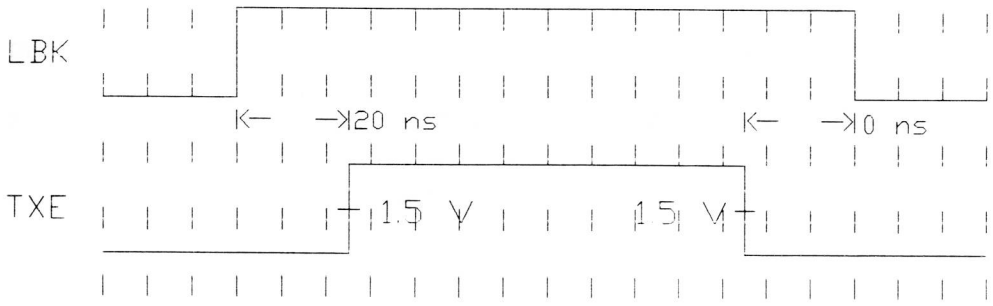


Figura 15 *– Diagrama de temporização do loopback.*

3.3 - Interface para Transmissão Coaxial (CTI):

O CTI consiste de 4 blocos lógicos principais [13]:

- *Receptor:* recebe dados do coaxial e envia-os para o DTE.
- *O Transmissor:* aceita dados do DTE e transmite-os para o cabo coaxial.
- *O Circuito Detetor de Colisão:* indica para o DTE qualquer colisão no coaxial.
- *O Temporizador Jabber:* desabilita o transmissor nos casos em que os pacotes sejam maiores que o tamanho máximo legal.

Neste próximo item será explicado o funcionamento do CTI. Assim como nos itens anteriores, explicaremos rapidamente através de diagrama de blocos e através de temporizações.

3.3.1 – Descrição Funcional:

Como já foi dito anteriormente, o CTI [13] consiste de 4 blocos de lógica principais: *receptor, transmissor, circuito detector de colisões e temporizador "Jabber"*. Veremos com um pouco mais de detalhes agora, observando sempre a figura 16.

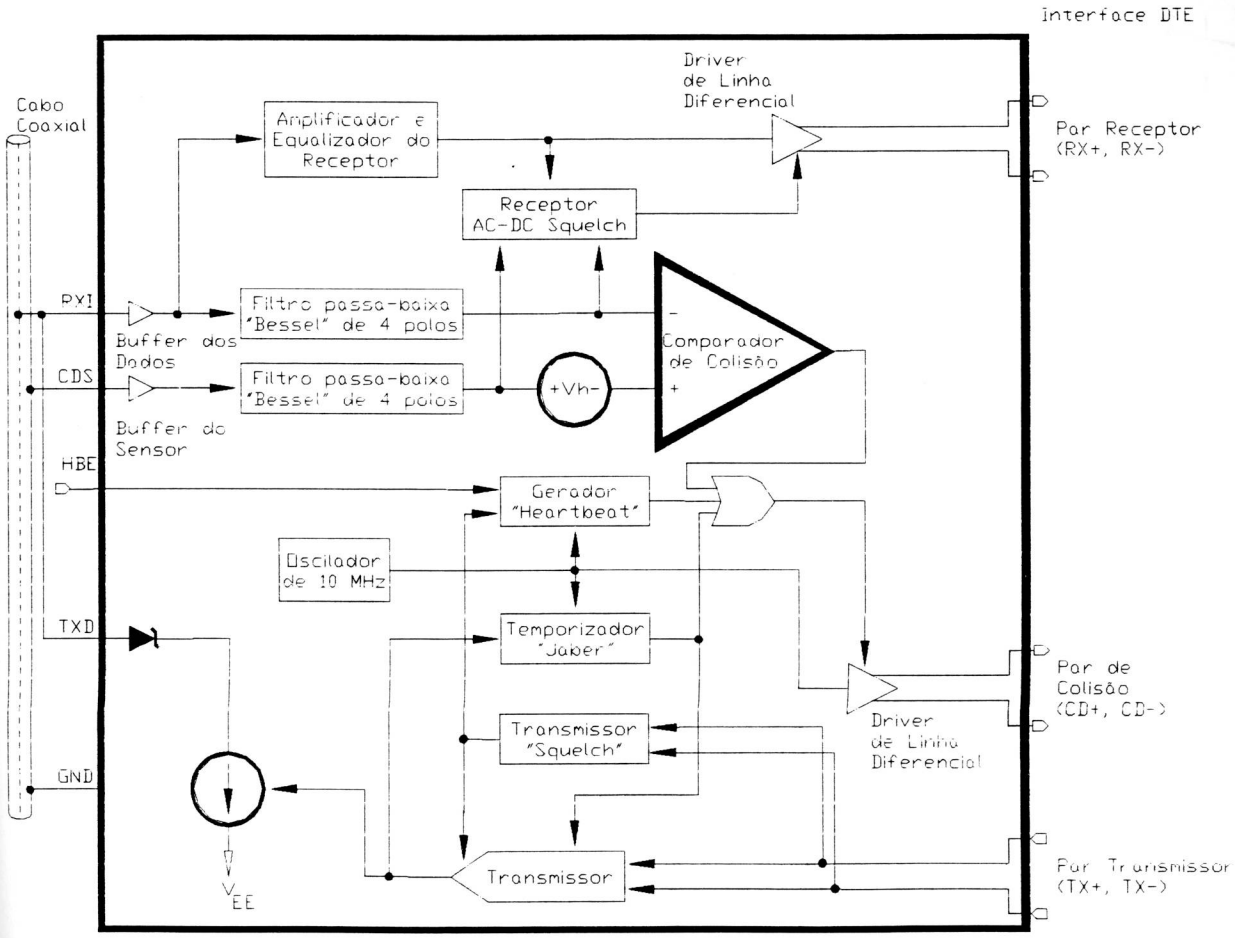


Figura 16 *– Diagrama de Blocos do funcionamento interno do CTI.*

- *Receptor:* inclui um circuito isolador de alta impedância ("buffer") na entrada do cabo equalizador, um filtro passa-baixa Bessel de 4-polos, um "circuito squelch" (circuito que diminui o ruído no cabo coaxial) e um "driver diferencial de linha".

  O "buffer" fornece uma alta impedância de entrada e uma baixa capacitância na entrada para minimizar as cargas e as reflexões no cabo coaxial. O equalizador é um filtro passa alta que compensa o efeito passa baixa do cabo. O resultado combinado pelo tamanho máximo do cabo e pelo equalizador é uma resposta comprimida para os sinais de frequência para minimizar o "jitter*".

  O filtro passa-baixa Bessel de 4-polos extrai a média do nível contínuo no cabo coaxial, que é usado por ambos os circuitos, o Receptor "squelch" e o detector de colisão.

  O circuito do Receptor "squelch" previne ruídos no cabo coaxial de gatilhos ("triggering") falsos no Receptor na ausência do sinal. No início do pacote o receptor é ligado quando o nível contínuo do filtro passa baixa for menor que o limiar contínuo "squelch". Entretanto no final do pacote um desligamento rápido do receptor é necessário para rejeitar os bits restantes. Isto é realizado por um circuito de temporização alternado (AC) que reage para sinais de nível alto superiores a 200 ns de duração. O receptor então permanece desligado em torno de 1 μs se o nível contínuo (CC) do filtro passa-baixa ficar acima da subida do limiar contínuo "squelch". A figura 17 ilustra a temporização do receptor.

---

* é um efeito que ocorre quando a latência na rede não é constante, ou seja, quando existe duas latências diferentes em uma determinada rede.

O driver de linha diferencial fornece sinais compatíveis do ECL (Emitter Coupled Logic = Lógica do Emissor Acoplado) para o DTE com tempos de caída e subida em torno de 3 ns. No estado ocioso, sua saída diferencial vai para zero para prevenir uma parada de corrente contínua no transformador isolador.

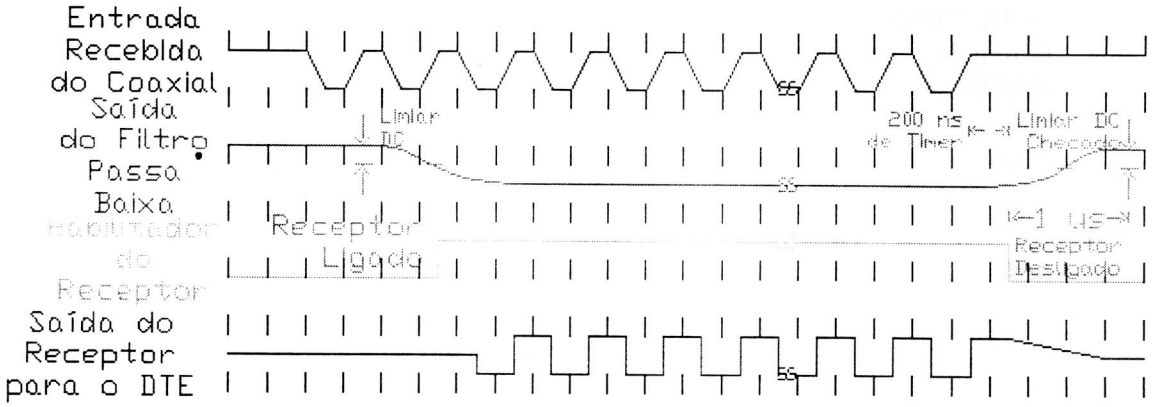


Figura 17 *– Diagrama de Temporização do Receptor.*

- *Transmissor:* contém uma entrada diferencial em coletor aberto na saída do driver corrente. A voltagem de modo comum é estabilizada pelo CTI e não deverá ser alterada por um circuito interno. O transformador conectado com TX ± satisfará esta condição. O driver satisfaz todas as especificações do IEEE 802.3/Ethernet de acordo com os níveis dos sinais.

  O circuito "squelch" do transmissor rejeita sinais com largura de pulso menor que 20 ns (negativo) ou com níveis menores que – 175 mV. O Transmissor desliga-se do final do pacote se o sinal permanecer maior

que – 175 mV e por mais que 300 ns. A Figura 18 ilustra a temporização do transmissor.

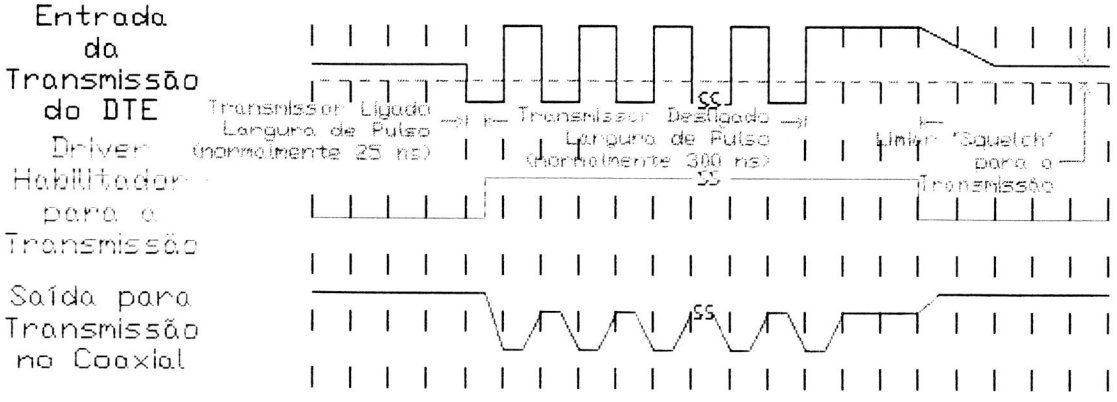


Figura 18 *– Diagrama de Temporização do Transmissor.*

- *Circuito Detector de Colisão:* o circuito de colisão consiste de dois "buffers", dois filtros passa-baixa Bessel de 4-polos, um comparador, um "gerador heartbeat*", um oscilador de 10 MHz e um "driver diferencial de linha".

  Dois buffers idênticos e um filtro passa-baixa Bessel de 4-polos extraem o nível contínuo do centro do condutor (dados) e da blindagem do cabo coaxial (sensor). Estes níveis são monitorados pelo comparador. Se o nível do dado for mais negativo que o nível do sensor pelo menos no limiar da colisão (Vth), a colisão na saída é habilitada.

  No final de cada transmissão, o "gerador de heartbeat" cria uma pseudo colisão em um curto período de tempo para se ter certeza de que o circuito de colisão está funcionando corretamente. Esta explosão ("burst") de colisão na saída ocorre tipicamente 1.1 μs após a

transmissão e tem uma duração em torno de 1 μs. Esta função pode ser desabilitada externamente com o pino HBE ("HEARTBEAT ENABLE") para permitir operações com repetidores.

O oscilador de 10 MHz gera o sinal para a colisão e para funções "heartbeat". Ele é também usado como um tempo de base para todas as funções "jabber" ("Jabber" é um nome dado a um temporizador que tem a finalidade de monitorar o transmissor. Se por um acaso o transmissor estiver ativado em um tempo maior que 20 ms, é inibida a transmissão, pois isto é considerado um defeito). Ele não requer nenhum componente externo.

A colisão no driver de linha diferencial transfere os sinais de 10 MHz para o par CD ± em eventos de colisão. "jabber" ou condições de "heartbeat". Este driver de linha também se caracteriza por um zero diferencial no estado ocioso.

- *Temporizador "Jabber":* o temporizador "jabber" monitora o Transmissor e inibe a transmissão se o Transmissor estiver ativado em um tempo maior que 20 ms (defeito). Ela também habilita a colisão na saída por um efeito duradouro. Após o defeito ser removido, o Temporizador "Jabber" espera em torno de 500 ms antes de reabilitar o Transmissor. A entrada do transmissor deve ficar inativo durante este tempo de 500 ms.

---

* Gerador Heartbeat seria mais ou menos um gerador de pulsos falsos de colisão, no circuito de colisão, com a finalidade de assegurar a integridade deste circuito.

3.4 – Conclusões:

Descrevemos os principais componentes de uma interface de rede (NIC - DP8390, SNI - DP8391 e CTI – DP8392) e como são feitas as trocas de sinais para administrar o controle do pacote a ser enviado ou recebido.

O NIC tem a função de controlar a camada de acesso do meio ("MAC") e a camada física de uma rede de área local ("LAN") de acordo com as especificações padrões do IEEE 802.3. Este padrão é baseado no método de acesso conhecido como CSMA-CD. Para transmitir um pacote o processador hospedeiro (8031) insere um comando de transmissão para o NIC, que normalmente transfere os dados para uma memória de buffer local. O NIC então automaticamente manobra a transmissão do pacote (do buffer local através da FIFO para o SNI) de acordo com o protocolo CSMA/CD. O pacote tem o formato da figura 04.

O SNI combina dados NRZ do pacote recebido do controlador com um sinal de clock e codifica-os numa corrente de bits serial usando uma codificação padrão chamada de codificação Manchester (observar figura 03). O sinal codificado aparece em uma forma diferencial na saída do SNI. Tanto o pacote transmitido do SNI quanto os outros sinais (recepção, colisão, etc.) devem estar eletricamente isolados do cabo coaxial. A isolação significa que deveremos suportar 500 V_{AC} rms durante 1 minuto. A isolação não deve ser feita no cabo coaxial, ao invés disso, ela deve ser feita na unidade da interface de conexão. A isolação dessas três linhas de sinais, foi feita usando um transformador. Observe nos Apêndices A e B, como foi feita o "hardware" e a isolação.

O dado codificado Manchester do SNI chegou agora na entrada da transmissão do CTI depois de passar através do transformador. Um filtro de ruído

nesta entrada fornece uma margem de ruído entre – 175 mV e – 300 mV. Estes limiares, asseguram que os sinais dos dados na transmissão diferencial (TX ±) menores que – 175 mV ou mais reduzidos que 10 ns sejam sempre rejeitados, enquanto os sinais maiores que – 300 mV e mais largo que 30 ns sejam sempre aceitados.

O CTI consiste de um receptor, um transmissor, um detetor de colisão e um temporizador "jabber". O transmissor conecta-se diretamente a um cabo coaxial de 50 Ω, onde ele será usado para direcionar no coaxial quando ele for transmitir. Durante a transmissão, o temporizador "jabber" é iniciado e é desabilitado o transmissor do CTI, caso o pacote de dados exceda o tamanho máximo permitido. O circuito do detetor de colisão monitora os sinais no coaxial para determinar a presença da colisão de pacotes.

É importante comentar que para se conseguir uma capacitância convincente no coaxial, é imperativo que o CTI seja soldado na placa sem um soquete. Na nossa placa, o CTI não foi soldado, pois utilizamos a técnica de "wire – wrap", que só pode ser concebida com soquetes. Se tivéssemos soldado o CTI na placa, seria permitido uma condução direta do aquecimento do dispositivo através da placa, reduzindo assim, significantemente a temperatura de operação do molde.

Capítulo 4

4.0 – Visão Geral sobre o funcionamento da interface:

O diagrama de blocos da Figura 19 ilustra o funcionamento da interface Cheapernet implementada.

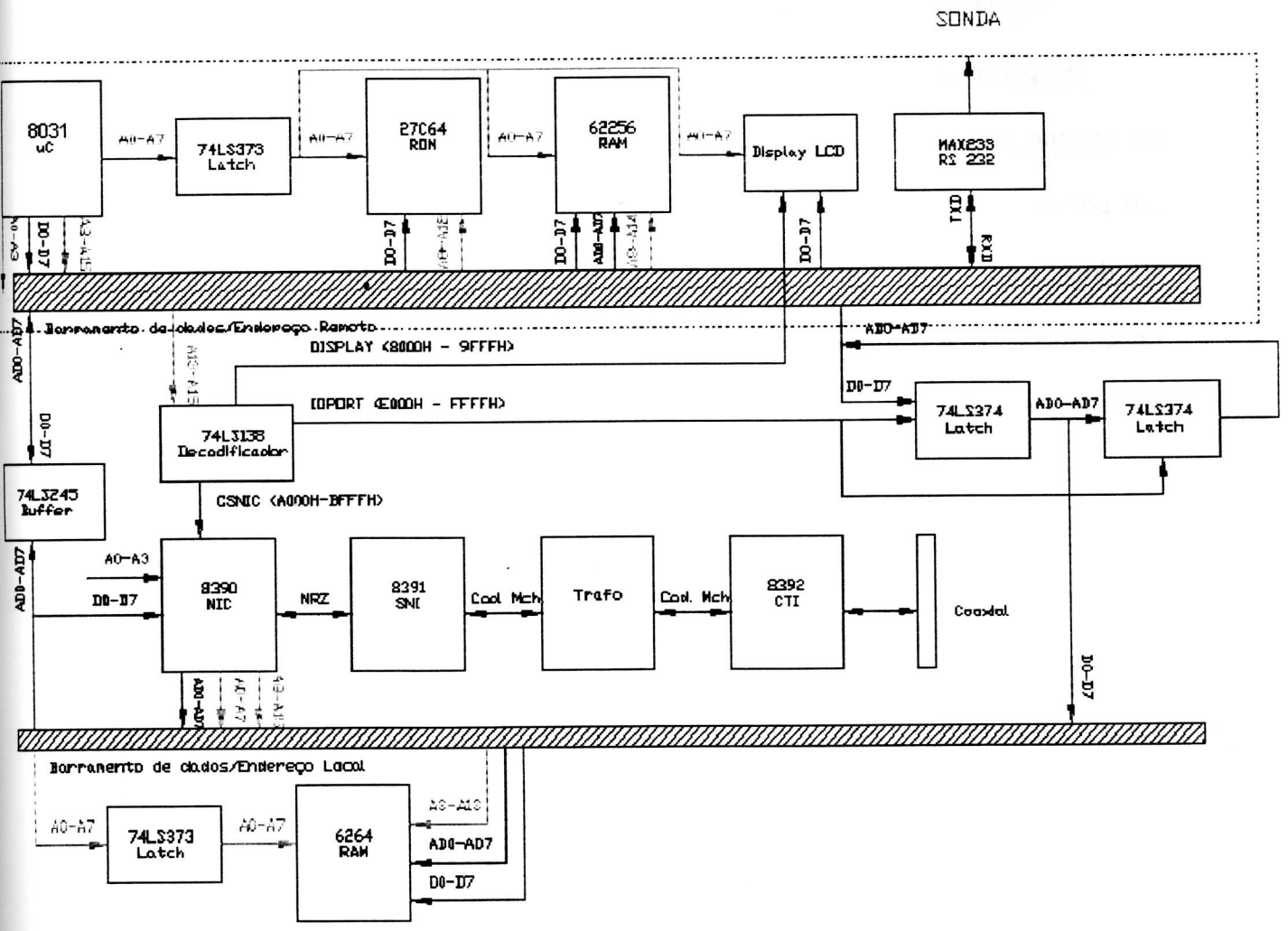


Figura 19 *– Diagrama de Blocos do barramento do sistema (8 bits) para Cheapernet*

O DP8390 (NIC), tem uma interface que permite uma conexão com o fio do Cheapernet que é um fio coaxial. O DMA dual (local e remoto) do NIC tem a capacidade de 8 kbytes de buffer RAM (Random Access Memory = Memória de

Acesso Aleatório). A arquitetura usada no porto de I/O (Entrada e Saída) neste dispositivo isola a CPU (Central Processing Unit – Unidade Central de Processamento) do tráfego da rede, fornecendo um método simples de interface entre o DP8390 e o sistema de barramento do microcontrolador (8031).

No diagrama de blocos da figura 19, o NIC é visto pelo sistema como um bloco de portos de Entrada/Saída. Com esta arquitetura o NIC tem seu próprio barramento local para acessar a memória da placa. O sistema nunca se intrometerá nos portos de Entrada/Saída de qualquer pacote de dados que estiver em operação. Há dois mapas de memórias que descrevem a placa. O primeiro é o mapa de Endereço da RAM do sistema que descreve como o processador acessa a placa. O segundo mapa descreve como o NIC acessa a memória na placa.

4.1 – Mapa de Memória da RAM do sistema:

O mapa de memória da RAM (esta RAM é chamada de RAM Remota) do sistema ficou dividida de acordo com a tabela VI.

Como podemos observar no apêndice A, temos uma RAM com 32kB de memória disponível (62256), no qual codificamos para a utilização da memória, entre os endereços 0000h à 7FFFh. O restante dos endereços 8000h à FFFFh é utilizado para acessar alguns dispositivos periféricos, como por exemplo display, o NIC e o os portos de I/O (74LS374), que comentaremos mais a frente. Utilizamos os endereços 4000h à 45FFh para transmissão do pacote. Este espaço é mais que suficiente, porque o tamanho máximo de um pacote ethernet é de 1518 bytes (contando o tamanho do header ethernet). Como temos 05FFh bytes de memória disponível, isto equivale no sistema decimal a 1535 bytes. Para a recepção,

disponibilizamos os endereços 4600h à 5FFFh. Para recepção é sempre importante ter um espaço maior, pois podemos receber mais que um pacote. Os endereços 6000h até 7FFFh serão utilizados como uma fila para transmissão. Ou seja, se por um acaso quisermos transmitir um pacote em um determinado momento e não pudermos transmiti-los porque neste exato momento ocorre uma recepção, é necessário que enfileiremos o pacote para futura transmissão para desocupar o endereço de memória (4000h). Depois que fizermos a recepção daquele pacote, o NIC fica disponível para transmissão, podendo transmitir o pacote que agora está na fila (6000h).

Tabela VI *– Mapa da Memória da RAM Remota (do 8031)*

Endereços da RAM	**Funções do Endereçamento**
0000h – 3FFFh	Espaço de dados Disponíveis na RAM
4000h – 45FFh	Espaço de dados utilizados para a transmissão de um pacote
4600h – 5FFFh	Espaço de dados utilizados para a recepção de um pacote
6000h – 7FFFh	Espaço de dados utilizados para enfileirar os pacotes para transmissões futuras
8000h – 9FFFh	Espaço de Memória utilizados para acessar o display
A000h – BFFFh	Espaço de memória utilizado para acessar o os registradores do NIC
C000h – DFFFh	Espaço de Memória disponível para acessar periféricos na RAM
E000h – FFFFh	Espaço de memória utilizado para acessar os barramentos de I/O entre RAM Remota e Local

4.2 – Mapa de Memória do NIC:

Para acessarmos o NIC, nós precisamos habilitar o sinal CS (Chip Select) e consequentemente mandar o comando para o NIC através do "Software". Mas

somente o NIC terá acesso a sua RAM, ou seja, depois de mandarmos para o NIC (através de software) onde ele deverá pegar o pacote ou onde ele deverá enviar o pacote, o próprio NIC se encarregará de fazer o resto. O buffer RAM ocupa o espaço de memória (chamado também de memória local) compreendido entre os valores 2000h à 3FFFh (observar tabela VII). Isto porque o buffer RAM contém 8kB de memória disponível (6264) e é usado para armazenar temporariamente na rede pacotes de dados que serão transmitidos ou recebidos. Assim como na RAM remota, a RAM local dividiu-se em duas partes: uma parte para a transmissão (2000h à 25FFh) e a outra para a recepção (2600h à 3FFFh). Quando um pacote for recebido, devemos utilizar o canal de DMA remoto para passar o pacote localizado no endereço 2600h da RAM local para a RAM remota (4600h), para que possamos ler o conteúdo da recepção. No caso da transmissão de um pacote, devemos copiar o pacote localizado no endereço de memória (4000h) da RAM remota para o endereço 2000h da RAM local utilizando o mesmo barramento de DMA remoto.

Tabela VII *– Mapa de Memória da RAM local (NIC – 8390)*

Endereços da RAM	**Funções do Endereçamento**
2000h – 25FFh	Espaço de dados utilizados para a transmissão de um pacote
4600h – 5FFFh	Espaço de dados utilizados para a recepção de um pacote

4.3 – Operação de acesso aos registradores do NIC:

As operações de acesso aos registradores do NIC são feitas para configurar o DP8390 para recepção e transmissão e também para controlar as operações dos canais de DMA. Neste tipo de acesso o NIC trabalhará de forma escrava, ou seja, ele

trabalhará de acordo com a CPU (8031) e não terá autonomia do barramento. Isto somente acontecerá quando o NIC for fazer uma transferência de DMA remota.

4.3.1 – Leitura dos registradores:

Antes de começarmos a explicar o funcionamento do acesso aos registradores do NIC, é importante salientar que tanto neste item quanto no próximo, estaremos nos referenciando ao apêndice A para dar uma olhada no hardware e acompanhar os comentários.

Para começar a leitura nos registradores, a CPU (8031) dirigirá as 4 linhas de endereço (P1.4 – P1.7) para o NIC e as linhas de endereço dos portos P0 e P2 para o decodificador 74LS138. Estas linhas de endereço são decodificados pelo circuito integrado 74LS138 para gerar o "chip select" (CSNIC) do NIC. A CPU também direciona a linha RD, através do comando MOVX, que o NIC vê como SRD ("slave read" – leitura escrava). Assim que o NIC recebe este SRD, ele envia então uma asserção alta no ACK, reconhecendo que está no modo escravo mas ainda não está pronto para completar a leitura.

O NIC então direciona os dados dos seus registradores internos para o buffer 74LS245. O buffer 74LS245 é então habilitado pela asserção do ACK baixo (isto significa que o NIC reconheceu a leitura) e os dados são direcionados para o barramento do 8031 (barramento remoto). Assim, o RD é ativado em alto pela CPU, desabilitando assim o SRD. É neste tempo de subida do RD que o dado que está no barramento do 8031 é buscado ("latched") pelo sistema. No mesmo tempo, os

endereços são removidos, ocasionando na desabilitação do "chip select" do NIC, finalizando o ciclo da leitura.

4.3.2 – Escrita dos registradores:

Para começar a escrita nos registradores, a CPU (8031) direcionará as 4 linhas de endereço (P1.4 – P1.7) para o NIC e as linhas de endereço dos portos P0 e P2 para o decodificador 74LS138. Estas linhas de endereço são decodificados pelo 74LS138 (através do endereço A000H) para gerar o "chip select" (CSNIC) do NIC. A CPU também direciona o "strobe" WR, através do comando MOVX, que o NIC vê como SWR ("slave write" – escrita escrava). Assim que o NIC recebe este SWR, ele envia então uma baixa asserção no ACK, reconhecendo que está no modo escravo está pronta para executar a escrita. Quando a CPU recebe este sinal, ela coloca os dados no barramento do 8031 (remoto) onde ele vai até o buffer 74LS245. O buffer 74LS245 então direciona os dados para o NIC (barramento local), mas os dados não entram no NIC enquanto não acontecer o tempo de subida do sinal WR. O sistema direciona o WR alto, desabilitando assim o SWR e buscando os dados. Simultaneamente os endereços são removidos, ocasionando na desabilitação do "chip select" do NIC, finalizando o ciclo da leitura. Os endereços são retirados e o "chip select" é desabilitado. Assim, a escrita aos registradores NIC é finalizada.

4.4 – Operação Geral da Interface do Barramento:

Todos os caminhos de dados passarão através dos portos de Entrada/Saída (os 74LS374). Cada 74LS374 é unidirecional e pode unicamente direcionar 8 bits,

por esse motivo é necessário ter dois 74LS374. Os dados são direcionados do porto para a memória buffer da interface utilizando um dos 74LS374 e o outro 74LS374 direciona os dado do porto para o barramento do sistema. O buffer RAM só pode ser acessada pelo NIC.

Para receber os pacotes, O NIC primeiro inicia a recepção checando o endereço deste pacote. Se o endereço corresponder ao endereço de sua interface (Endereço Físico colocado nos registradores físicos; rever tabela III) então o dado é recebido pelo NIC. O NIC utiliza o canal de DMA Local para armazenar o pacote na próxima área disponível do "buffer RAM de 8 Kbytes". Quando todo o pacote for armazenado, o NIC gerará um interrupção para a CPU (8031). Se houver um erro no pacote que está armazenado dentro da RAM, o NIC rejeitará o pacote e exigirá que o local da memória que este pacote ocupou seja esvaziado (desocupado). Isto será feito automaticamente por uma função que já existe dentro do NIC chamada de "send packet".

Após o reconhecimento da interrupção, a CPU deverá então programar o NIC para uma leitura remota do pacote via DMA. Quando for feito o DMA Remoto do NIC, a CPU trocará sinais (handshake) com o DMA para ler cada byte através do porto de dados de Entrada/Saída e armazená-lo na memória principal do processador 8031. Observar figura 19.

Para a transmissão do pacote o sistema da CPU primeiro programa o DMA Remoto do NIC para receber um pacote do sistema em uma área predeterminada do buffer RAM na placa. A CPU e o NIC então fazem uma troca de sinais (handshake) enquanto os dados são enviados através dos portos de Entrada/Saída dos dados.

Uma vez que o pacote transmitido estiver por completo na RAM local da placa, o NIC é então programado para transmitir o pacote para fora na rede. A NIC

então lê o dado transmitido usando este canal de DMA local e então segue o protocolo CSMA/CD para transmitir os dados. Quando a transmissão estiver completa uma interrupção é gerada e a CPU pode checar o estado da transmissão para se ter certeza se a própria transmissão ocorreu.

Aqui foi comentado de uma maneira geral como ocorre a leitura e a escrita remota. A seguir será descrito com mais detalhes a leitura e a escrita remota.

4.5 – Transferência do pacote via DMA Remoto pelo NIC:

As transferências DMA Remotas são operações executadas pelo NIC na interface. Estas operações ocorrem quando o NIC é programado para transferir pacotes de dados entre o sistema microcontrolado pelo 8031 e a RAM da interface. Estas transferências são feitas através do interfaceamento dos portos Entrada/Saída.

O canal DMA Remoto é usado para levar os pacotes para a transmissão e para remover os pacotes recebidos no buffer em anel do receptor. Ele também pode ser usado com o propósito de um canal DMA escravo para mover blocos de dados ou comandos entre a memória do microcontrolador 8031 (U4 no Apêndice A) e a memória do buffer local (U14). Existem três modos de operação: Escrita Remota, Leitura Remota e utilizando o comando "Send Packet".

Dois pares de registradores são usados para controlar o DMA Remoto, um par de registradores de Endereço Inicial Remoto (RSAR0, RSAR1) e um par de registradores de Contador de Byte Remoto (RBCR0, RBCR1). O par de registradores de Endereço Inicial, aponta para o começo do bloco a ser movido enquanto o par de registradores de Contador de Byte é usado para indicar o número de bytes a ser

transferido. Toda a lógica de "handshake" (troca de sinais), é providenciada para mover dados entre a memória de buffer local e o porto bidirecional de I/O.

Como pode-se observar, temos que acessar dois registradores para escrevermos tanto o endereço inicial de onde o pacote será transmitido/recebido quanto para o tamanho do pacote que será transmitido/recebido. Isto acontece porque apesar destes registradores serem vistos pelo NIC como registradores de 16 bits, o nosso barramento de dados contém apenas 8 bits, então será necessário acessarmos primeiro o bit menos significativo (LSB) de cada um dos registradores e depois o bit mais significativo (MSB).

4.5.1 – Processo de Leitura Remota:

Para programar o NIC para uma leitura remota, a CPU deve fazer 5 acessos escravos inicialmente. A CPU deve escrever nos registradores de Endereço Inicial Remoto (A000H – BFFFH; que são 2 bytes), nos registradores Contadores de Byte Remoto (que são 2 bytes) e inserir um comando de leitura de DMA Remoto (que é mais 1 byte). Os endereços e o contador de bytes requerem 2 transferências porque eles são de 16 bits e somente oito bits podem ser transferidos pelos portos (74LS374).

Uma vez que o NIC recebeu todos os dados acima ele dirige o sinal BREQ e recebe o BACK imediatamente, já que nenhum outro dispositivo utilizará o barramento local e assim, BREQ e BACK são conectados juntos. Após receber o BACK, o NIC direciona o endereço dos dados que estão sendo requeridos para a leitura. Estes endereços fluem através do 74LS373 e é armazenado pelo ADS0. Daqui os endereços fluem para a RAM. A RAM espera até receber o MRD do NIC e

então dirige os dados para os portos 74LS374. Os portos 74LS374 então, buscam ("latch") os dados no tempo de subida do "strobe" PWR do NIC. O PRQ é então enviado pelo NIC para deixar que o sistema saiba que há dados esperando nos portos.

Depois que os dados estão nos portos, o sistema deve então ler os dados dos portos 74LS374. Isto começa com o 8031 direcionando o endereço que é decodificado pelo 74LS138 (E000H) para o porto de dados de I/O. Assim, o sinal RACK é dirigido para o NIC, indicando que a CPU está pronta para aceitar os dados. Este sinal RACK então, espera os dados dos portos 74LS374 entrarem no barramento do PC (remoto). O sistema desabilita o RD que finaliza o ciclo.

4.5.2 – Processo de Escrita Remota:

Como na leitura remota, a escrita remota também começa com 5 acessos escravos aos registradores internos do NIC. A CPU (8031) também escreve nos registradores de Endereço Inicial Remoto (C000H – DFFFH; que são 2 bytes), nos registradores Contadores de Byte Remoto (que são 2 bytes) e insere um comando de escrita de DMA Remoto.

O NIC então insere um PRQ. A CPU responde enviando um sinal de escrita WR, indicando que ela está pronta para escrever nos portos de I/O. A CPU também envia os endereços que correspondem aos portos I/O (E000H). Este endereço também ajuda a decodificar o sinal WACK. Este sinal WACK busca ("latch") os dados nos portos 74LS374. O NIC inseri um BREQ e imediatamente recebe um BREQ, já que estão conectados juntos pelo fato de não haver outros dispositivos que utilizam o barramento local. O NIC após o recebimento do BACK, direciona as linhas de endereços relativos aos dados no 74LS373. Estas linhas de endereços são

armazenadas pelo ADS0 e então são dirigidas para a RAM. O NIC então envia um PRD e um MWR que dirige os dados dos portos 74LS374 para o endereço que já havia sido especificado na memória RAM. O PRD e o MWR são desabilitados e o ciclo é finalizado.

4.6 – Transferências para/da rede:

Transferências de e para a rede são controladas pelos canais de DMA local do DP8390 que transfere pacotes de dados de/para a FIFO interna do NIC para/de a RAM da interface.

4.6.1 – Recepção:

Os dados que vêm da rede são "desserializados" e armazenados na FIFO dentro do NIC. O NIC então insere um BREQ e imediatamente recebe um BACK desde que os pinos sejam conectados juntos. Após receber o BACK, o NIC direciona as linhas de endereço para o 74LS373. O 74LS373 é armazenado pelo ADS0 e é permitido que o endereço flua para a RAM. Então o NIC direciona o MWR junto com os dados da FIFO. Os dados fluem para RAM no endereço dado anteriormente. O "strobe" MWR é então desabilitado, finalizando assim o ciclo.

A figura 20 [5] descreve o formato como os pacotes recebidos estão localizados na memória pelo canal de DMA local. Quando o primeiro pacote começar a chegar, o NIC começa a armazená-lo na localização apontada pelo registrador de página corrente ("Page Current"). Um deslocamento de 4 bytes (um header) é armazenado no primeiro buffer com informações do estado da recepção do

pacote. Este deslocamento nada mais é que o estado da recepção, um ponteiro para onde o próximo pacote será armazenado (neste exemplo é o buffer 4) e o número de bytes recebido (2 bytes).

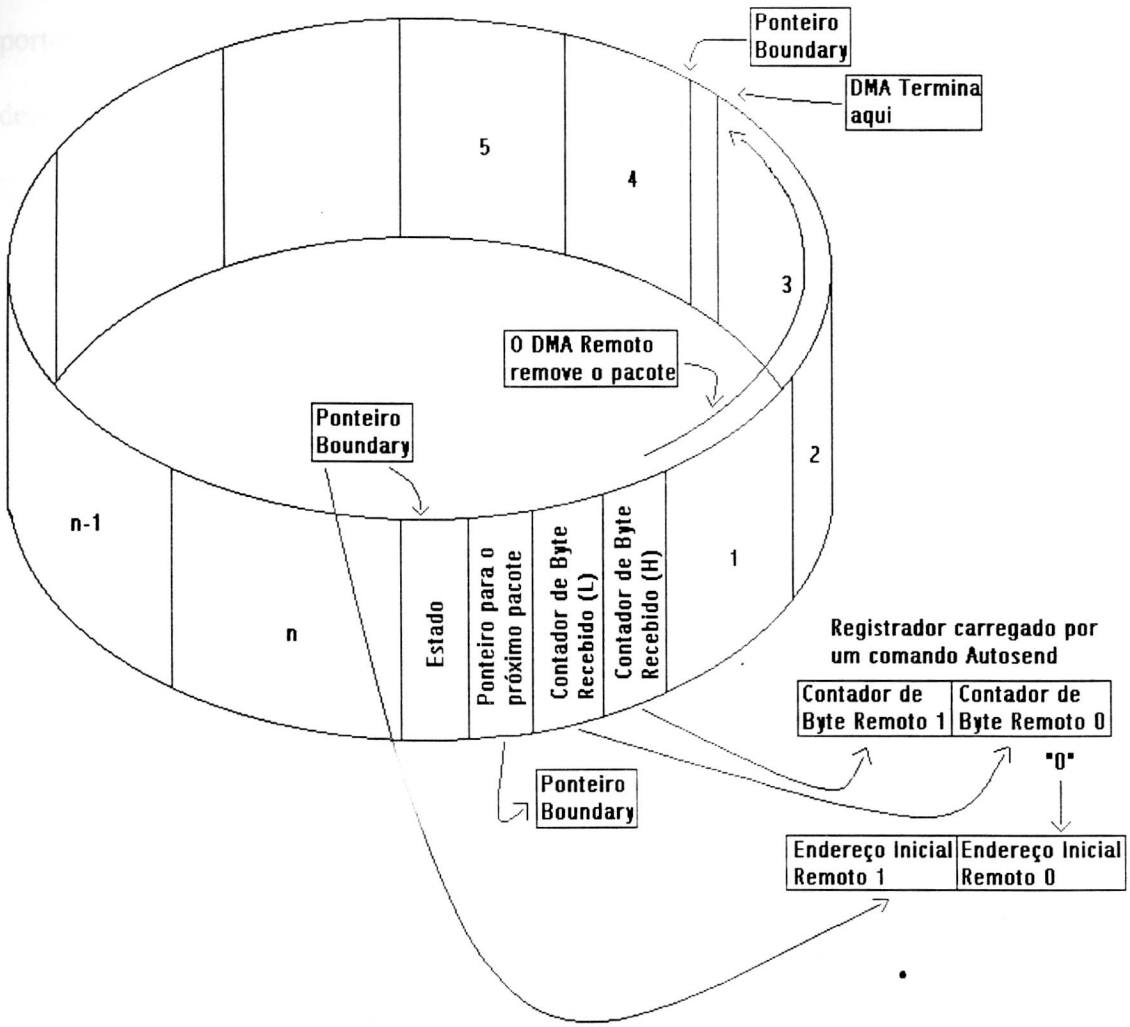


Figura 20 *- Auto inicialização do DMA Remoto no Buffer em Anel*

4.6.2 – Transmissão:

Para começar o ciclo de transmissão, o NIC insere um BREQ e espera pelo BACK. Como os dois pinos são conectados juntos, o sinal BACK é recebido

imediatamente. No reconhecimento deste sinal, o NIC direciona os endereços para o 74LS373 que envia os endereços com o "strobe" ADS0. Os endereços então fluem para a memória RAM da interface. O MRD, direcionado pelo NIC, faz com que a RAM envie os dados do endereço relatado para a FIFO do NIC. O NIC então busca os dados na FIFO no tempo de subida do sinal MRD. Esta asserção alta do sinal MRD significa a finalização do ciclo. Da FIFO o dado é serializado e transmitido para a rede.

4.7 – Recepção dos dados medidos pela sonda:

A aplicação desta interface Cheapernet controlada por um controlador da família 8031 era a de poder disponibilizar na rede os dados medidos por uma sonda [38] que foi desenvolvida no laboratório. Como já foi dito anteriormente, esta sonda tem a finalidade de medir variáveis físicas e químicas da água: PH, turbidez, oxigenio dissolvido, temperatura, etc. Assim, esses dados serão passados à interface e colocados no endereço de memória 402AH (componente U4 do apêndice A) que é o endereço para a transmissão dos dados. Podemos ver este modelo de acordo com a figura 21.

Esta comunicação será feita serialmente entre as duas interfaces utilizando o padrão de comunicação RS – 232. O componente utilizado para fazer esta comunicação é o circuito integrado MAX233 de 20 pinos. Podemos observar tanto na figura 21 quanto no apêndice A, que a comunicação entre as interfaces é feita através de um cabo que é unido pelos conectores DB – 9 (interface Cheapernet) e DB – 25 (interface de processamento da placa).

A interface de processamento da placa foi desenvolvida como parte do trabalho de mestrado do aluno Marcelo Marques Simões. O nosso objetivo é a de mostrar que houve um processamento anterior antes destes dados serem enviados pela rede.

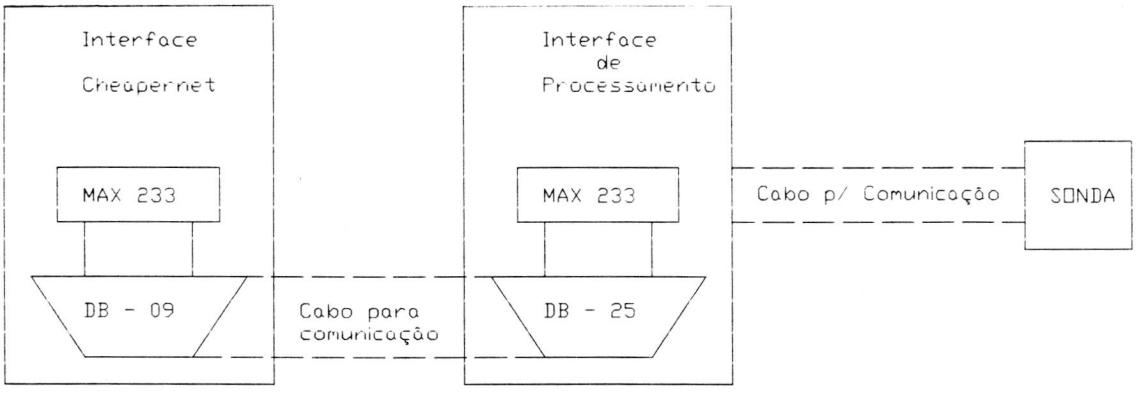


Figura 21 – *Diagrama de comunicação entre a interface Cheapernet e a interface de processamento da sonda*

4.8 – Conclusões:

Como podemos observar no Apêndice A, os circuitos podem ser divididos em duas partes. A primeira que implementa um microcomputador para controlar a interface e para fazer comunicação com a interface que faz o processamento da sonda (o microcontrolador 8031 [28], o 74LS373 (U2), a memória 2764, a memória RAM 62256 (U4), o mostrador de cristal líquido (LCD) de 2x40 caracteres, o circuito

integrado MAX233 que faz a comunicação serial, utilizando o padrão RS – 232, dos dados medidos pela sonda e algumas portas lógicas de controle) e a segunda que implementa a interface de rede (o buffer bidirecional 74LS245, o 74LS373 (U13), os três circuitos integrados que implementam a interface de rede DP 8390/91 e 92, o transformador de pulso (PE64103), o conversor DC- DC (no Apêndice B), a memória RAM, vista como buffer para recepção e transmissão dos pacotes 6264 (U14 – que faz o papel de "buffer"), dois 74LS374 e mais algumas portas lógicas de controle).

O microcontrolador 8031 utiliza as portas P0.0 – P0.7 como LSB (Least Significant Bit – Bit Menos Significativo) para endereçar a memória RAM 62256 (U4) e também como barramento de dados que é acessado tanto pela interface da rede através do 74LS245 (U12) quanto pelos dispositivos do microcomputador (74LS373 (U2), 2764, 62256 (U4), Display LCD).As portas P1.0 – P1.7 são utilizadas como sinais de controle. Como exemplo, podemos citar as portas P1.0 e P1.1 que são utilizadas respectivamente para mandar um sinal RS e um sinal de R/W (leitura ou escrita) para o Mostrador LCD. As outras portas, P1.4 – P1.7, são conectadas aos pinos RA0 – RA3 que são usados para endereçar os registradores do controlador NIC em cada página. O mostrador LCD é utilizado para apresentar mensagens durante a recepção e transmissão dos pacotes na rede, ou seja, será enviado mensagens ao display de acordo com a qualidade do pacote (se o pacote estiver com defeito ou não, ou ainda se o pacote teve a transmissão bem sucedida ou não).

É importante comentarmos também que as placas de rede local atuais raramente utilizam o controlador SNI (DP8391). Hoje em dia já temos controladores que implementam tanto o NIC quanto o SNI em um único CI. Como exemplo,

podemos citar o DP83902 que é conhecido como ST-NIC (Serial Network Interface Controller for Twisted Pair), que é um controlador de rede para fios de par trançado (os mesmos fios de padrões telefônicos). Existem ainda controladores de rede que já têm SRAM interna, não necessitando assim da utilização de uma RAM externa para a recepção ou transmissão de um pacote. O controlador de rede 8029AS da REALTEK utiliza a sua RAM interna para transmissão e recepção do pacote.

Capítulo 5

5.0 - Descrição Geral dos Programas desenvolvidos para a rede Cheapernet ("Driver"):

Neste capítulo, veremos a descrição básica de um driver para um Cheapernet. Descreveremos:

1. *Inicialização do Hardware.*
2. *Inicialização da Transmissão.*
3. *Serviço da Interrupção para a Transmissão e a Recepção.*

Dentro desses três parâmetros do Driver, teremos respectivamente as seguintes rotinas: *Driver Inicialize (Main), Driversend e Driverisr.* As principais bibliografias utilizadas para a construção deste driver foram: [14], [15] e [26].

5.1 - Inicialização do Hardware:

O processo de inicialização requer uma configuração dos parâmetros para o Controlador NIC operar no sistema corrente. A CPU (8031), se configurará os registradores de endereço e habilitará o controlador NIC na rede. Abaixo, indicaremos uma lista de parâmetros que devem ser inicializados, antes do controlador NIC entrar em operação:

- *Largura do barramento de dados (8 ou 16 bits):* no nosso caso, oito bits pois trabalharemos com o 8031 que contem 8 bits de dados.
- *Endereço Físico;* É o endereço da interface da rede, que deve ser único. Utilizaremos o seguinte endereço ethernet: 08:00:17:01:09:74. Poderia

ter sido outro endereço qualquer que começasse com o byte mais significativo contendo o valor 00. Por exemplo: 00:C0:BB:12:34:56.

- *Tipo de Serviços de Interrupções;* Atenderemos três tipos de serviços de interrupções: geradas por recepção de um pacote, transmissão de um pacote e erro na transmissão.
- *Tamanho do Buffer em Anel do Receptor ("Receive Buffer Ring");* O tamanho será de 8 kBytes.
- *Limiar da FIFO;* Este limiar é para transferência de 8 bytes.
- *Tipos de pacotes que podem ser recebidos;* São unicamente pacotes bufferizados para a memória e com endereço físico.

Toda essa inicialização está contida dentro da rotina *Princi* listada no Apêndice C. Observa-se também que o Registrador de Configuração de Dados (DCR) no NIC, deve ser inicializado antes de todos os outros registradores (exceto o Registrador de Comando (CR), é claro).

Antes de ligarmos a interface Cheapernet como interface de rede, devemos primeiro receber o pacote de dados medidos pela sonda. Isto é feito pela rotina FS2 junto com a rotina SERIAL_INI. Estas rotinas se encarregarão não só de adquirir os dados medidos pela sonda, mas também de colocar os dados no endereço certo para a transmissão (depois do cabeçalho ethernet e do cabeçalho IP, por exemplo).

5.2 – Transmissão do Pacote:

O Driver transmissor é normalmente separado em duas partes: Na primeira parte, (Driversend) é iniciada uma transmissão quando for passado um pacote para o driver, ou seja, quando recebermos o pacote da sonda, a transmissão poderá ser

habilitada. Se o driver não estiver habilitado para transmitir o pacote imediatamente (isto é, o transmissor estiver ocupado por uma recepção naquele momento), o pacote fornecido é enfileirado em um "buffer" para transmissão pendente. Após iniciar a transmissão ou enfileirar o pacote, o Driversend retorna a rotina principal (Princi) e ficará nela até que ocorra uma recepção.

O Driversend opera em conjunto com uma rotina de atendimento de interrupção (Driverisr). Após a transmissão estar completa, o controlador NIC interrompe a CPU para sinalizar o final da transmissão e indicar o estado desta transmissão no Registrador de Estado Transmitido ("TSR – Transmit Status Register").

5.3 – Recepção do Pacote:

A função do driver receptor é de transferir dados do "buffer receptor em anel" para a memória controlada pelo 8031. Idealmente este processo é feito tão rapidamente quanto possível para eliminar qualquer "gargalo" que possa ser adquirido pelo driver. O controlador NIC facilita a remoção dos dados do anel, providenciando um canal DMA Remoto para transferir dados do anel para um porto de Entrada/Saída (os dois 74LS374) que é lido pelo sistema hospedeiro. O controlador de rede também mantém dois ponteiros para rastear pacotes no anel: *BOUNDARY* e *CURRENT*. Estes registradores respectivamente apontam para o último pacote não lido no anel e a próxima vaga de alocação de memória para receber outro pacote. Normalmente o driver receptor remove o próximo pacote apontado por *BOUNDARY*, então incrementa *BOUNDARY* para o pacote sucessivo indicado pelo ponteiro da próxima mensagem ("Next Page Pointer") no quarto byte

do "header" (cabeçalho) recebido pelo controlador NIC. Este processo continua até todos os pacotes terem sido removidos do anel.

O controlador NIC automaticamente remove pacotes com o comando "send packet". Quando este comando é inserido, o controlador NIC automaticamente carrega o endereço inicial DMA no *BOUNDARY*, carrega o contador de byte de DMA com o quarto byte do "header" recebido e então os dados começam a ser transferidos. No final do DMA o controlador NIC atualiza o *BOUNDARY* com o ponteiro da próxima mensagem no "header" recebido. Para remover todos os pacotes do anel, o driver receptor simplesmente insere o comando "send packet" até os registradores *BOUNDARY* e *CURRENT* serem iguais.

Pelo fato do assincronismo natural da recepção, o driver receptor deve ser escrito na interrupção. A recepção do pacote tem alta prioridade desde que o atraso na remoção do pacote possa causar uma sobrecarga ("overflow") no buffer em anel do receptor. Se vários pacotes no anel tiverem sido enfileirados, todos os pacotes deverão ser removidos em um processo (isto é, um software (QUEUE_PACKET no Apêndice C) que esvazia o anel). Em condições de tráfego pesado, a memória local pode ser preenchida muito rapidamente, então é importante que o anel seja grande o suficiente para lidar com esta situação.

Para saber quantos pacotes serão perdidos devido à sobrecarga no anel ou erros na rede, o controlador NIC tem registradores estatísticos para monitorar a rede: contador de ERRO DE ALINHAMENTO DO FRAME, contador de ERRO DE CRC, contador de FRAMES PERDIDOS.

5.4 – Detalhes do "Driver":

Os "Drivers" de transmissão e recepção contidos no Apêndice C foram escritos em linguagem montadora ("linguagem assembly") para execução mais rápida. O Driver transmissor foi dividido em duas partes, *Driversend* e *Driverisr*, enquanto o driver receptor reside inteiramente no *Driverisr*. Mostraremos nesta próxima seção a descrição de como a recepção e a transmissão se interagem na interrupção com *Driverisr*.

5.4.1 – Rotina de Serviço de Interrupção (DriverISR):

O Driverisr foi desenvolvido com interrupções originadas de recepções, transmissões e erros nas transmissões. Recepções erradas são ignoradas. O fluxograma do Driverisr (Figura 22) consiste de:

1. *Rotina de Pacote Transmitido.*
2. *Rotina de Pacote Recebido.*

As funções básicas das rotinas serão mostradas a seguir:

- *Rotina de Pacote Transmitido:* Checa o estado de todas as transmissões e transmite o pacote seguinte na fila de transmissão pendente.
- *Rotina de Pacote Recebido:* Remove todos os pacotes no buffer em anel do receptor usando os registradores do controlador NIC.

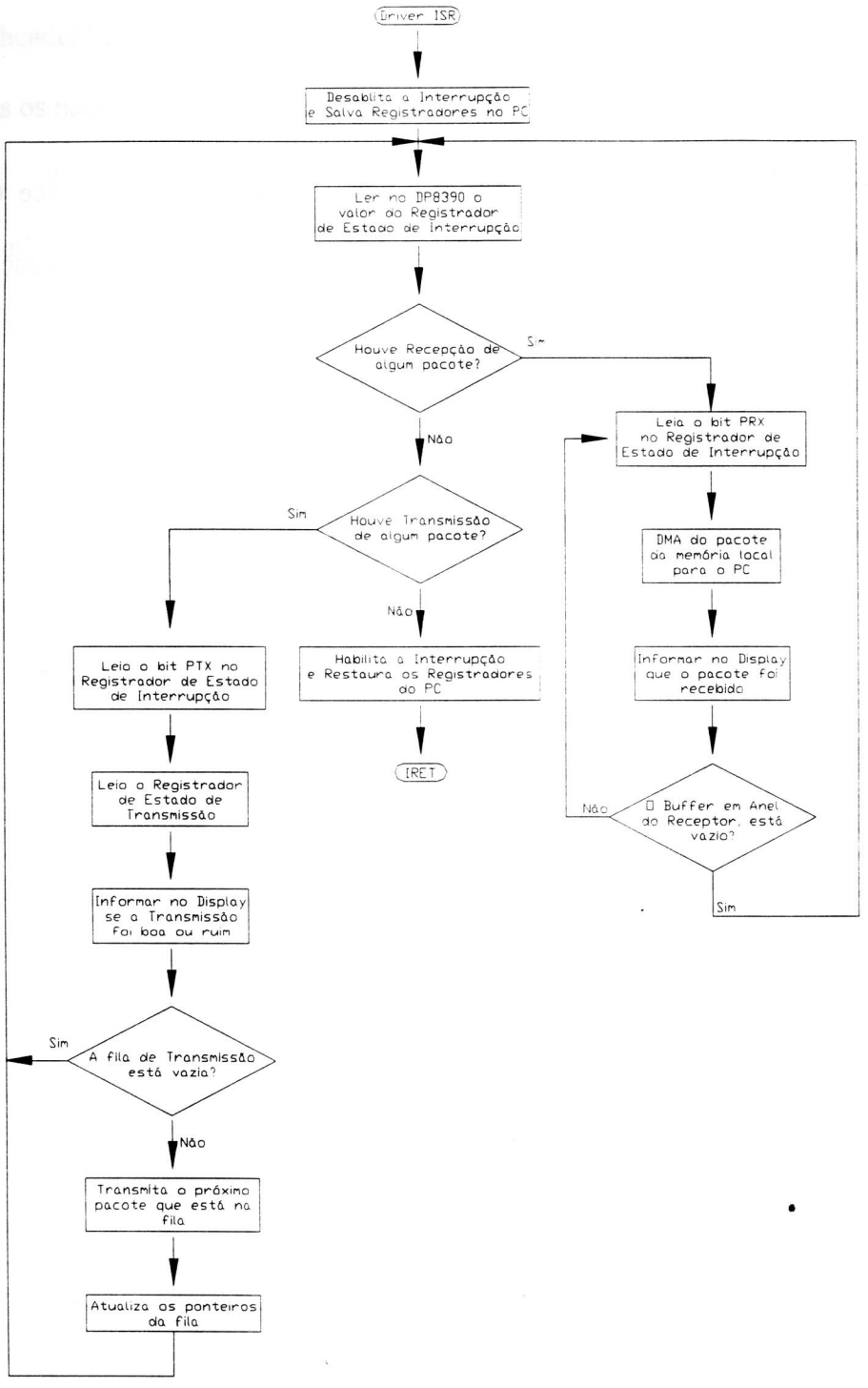


Figura 22 *– Fluxograma da Rotina de Serviço de Interrupção*

5.4.2 – "Driver" para a Transmissão (Driversend):

O "Driver" Transmissor consiste de duas partes. A primeira parte, "Driversend" (Figura 23), começa a transmissão quando temos algo para transmitir. Uma interface Cheapernet de fábrica, começa a transmissão quando um software das camadas superiores passa um pacote para o driver para ser transmitido. Um exemplo de software das camadas superiores são: "UDP", "TCP", etc. Como não foi desenvolvido um software da camada superior, utilizamos um outro tipo de transmissão de pacote para o driver. A rotina que passará os dados a serem transmitidos na rede é a rotina de comunicação serial (SERIAL_INI que é uma subrotina de FS2, que faz as inicializações antes da transmissão) entre os dados processados pelas medições da sonda e a nossa interface Cheapernet. Estes dados serão colocados na memória de endereço 402AH do microcomputador (U4 – do Apêndice A). Durante o desenvolvimento do projeto utilizamos o emulador iceMASTER8031 (MetaLink Corporation) para emular o microcontrolador 8031) e ao mesmo tempo, conseguimos ter acesso à memória externa deste micro (U4). Voltando ao funcionamento do Driversend, ele checa se o controlador NIC está pronto para transmitir, lendo o Registrador de Comando (CR), ou seja, vendo se o bit TXP é zero [1]. Se estiver pronto, o Driversend usa o canal DMA Remoto do DP8390 e transfere o pacote da memória do 8031 (4000H) para a memória local (2000H) e então insere o comando de transmissão e retorna. De outra maneira, se o controlado NIC estiver ocupado (o bit TXP for igual a 1), o Driversend enfileira o pacote na fila pendente para transmissão e então retorna.

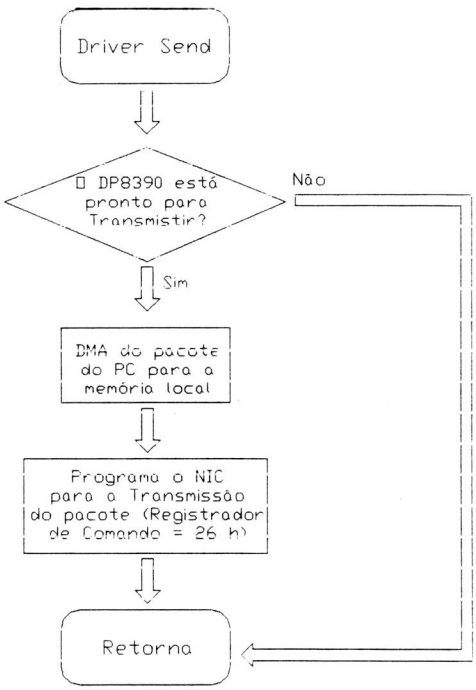


Figura 23 - *Rotina Driversend*

Após ser finalizada a transmissão, o serviço de interrupção (Driveisr) do controlador NIC é acionada e:

1. Transmite o estado da informação lendo o Registrador de Estado de Transmissão (TSR).
2. Transmite o próximo pacote na fila para transmissão pendente, se existir.

Assim, na interrupção para transmissão, o Driverisr executa os seguintes passos:

1. Reseta o bit PTX no Registrador de Estado de Transmissão.

2. Checa para ver se a transmissão foi boa, lendo o Registrador de Estado de Transmissão.
3. Se existirem mais pacotes na fila para transmissão pendente, transmite o próximo pacote, de outra maneira vai para 4.
4. Lê o Registrador de Estado de Interrupção para ver se tem alguma interrupção pendente.

5.4.3 – Driver para a Recepção:

•

O driver para a Recepção deve estar na interrupção, assim, ele reside inteiramente no Driverisr. Quando ocorre a interrupção da recepção, um ou mais pacotes devem ser bufferizados no anel pelo controlador NIC. O Driverisr remove pacotes do anel (2600H) e passa-os para o microcotrolador 8031 (4600H). Os pacotes devem ser removidos até o anel estar vazio, isto é, quando os registradores CURRENT e BOUNDARY forem iguais. A sequência da rotina de pacote recebido é mostrada abaixo:

1. Reseta o bit PRX no Registrador de Estado de Interrupção.
2. Remove o próximo pacote no buffer receptor.
3. Checa para ver se o pacote no anel do buffer receptor está vazio; Registrador BOUNDARY = Registrador PAGE CURRENT.
4. Se o anel não estiver vazio, vai para 1; de outra maneira lê o registrador de Estado de Interrupção sem qualquer interrupção.

5.5 – Conclusões:

Neste capítulo, foi descrito driver desenvolvido para a interface Cheapernet e como ele se relaciona com o hardware.

Como pode-se observar o Driver foi dividido em duas grandes partes: o Driversend e o Driverisr, onde a transmissão de um pacote ficava dividida no entre o driversend e o driverisr e a recepção ficava totalmente escrita no driverisr.

O ponto principal deste software, é a remoção dos pacotes do buffer em anel, de acordo com os ponteiros CURRENT e BOUNDARY. Os principais cuidados a serem tomados são dois:

1. O ponteiro BOUNDARY deve ser mantido sempre com um valor a menos que o ponteiro CURRENT quando o anel estiver vazio.
2. O ponteiro BOUNDARY somente deve ser igual ao ponteiro CURRENT quando o anel estiver cheio.

Um outro ponto importante deste driver está na maneira de como o pacote é controlado pela placa de rede. Nós utilizamos um emulador (iceMASTER8031) para acionarmos a memória remota da placa (U4) para verificarmos todos os pacotes antes da transmissão. Através deste emulador, também conseguimos ver o pacote de recepção. Ou seja, se enviarmos algo para a nossa interface, conseguiremos observar exatamente o que chegou até ela, pois temos um controle total sobre a sua memória.

Capítulo 6

6.0 – Introdução ao ICMP:

Neste capítulo abordamos a comunicação da interface Cheapernet desenvolvida com um microcomputador.

A comunicação entre a interface e o computador será controlada por um programa desenvolvida na linguagem montadora do microcontrolador 8031. O programa utilizou o protocolo ICMP (Internet Control Message Protocol – Protocolo de Mensagem para Controle da Internet) para fazer o teste de comunicação entre o PC e a interface Cheapernet. O ICMP nada mais é que um mecanismo utilizado pelos roteadores e pelos *hosts* para enviarem informações de erros ou de controle sobre a rede. As mensagens ICMP viajam ao longo da internet na porção de dados dos datagramas IP, observar figura 24.

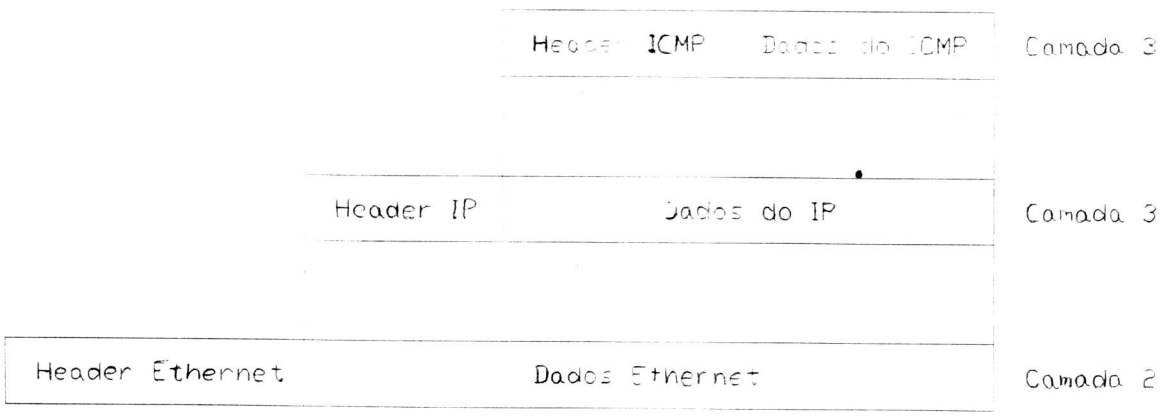


Figura 24 – *Encapsulamento do ICMP*

Ainda que cada mensagem ICMP tenha o seu próprio formato, começam todas com os mesmos três campos: um campo de *tipo* (inteiro de 8 bits) que identifica a mensagem, um campo de *código* com 8 bits que fornece mais informações sobre o tipo de mensagem e um campo de *checksum* de 16 bits (o ICMP utiliza o mesmo algoritmo checksum que o IP).

A estrutura típica de uma mensagem ICMP pode ser vista na figura 25. O campo *tipo*, é que define o significado da mensagem.

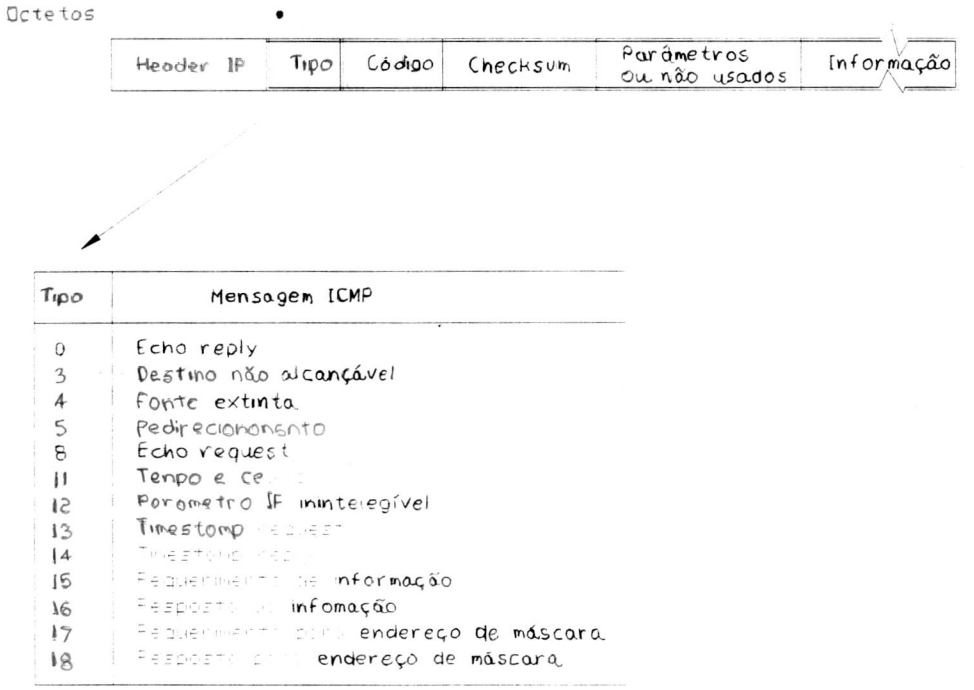


Figura 25 – *Estrutura geral de uma mensagem ICMP*

6.1 – ICMP Tipo 0 e 8 – Echo/PING:

O protocolo TCP/IP fornece certas facilidades para que os responsáveis ou usuários das redes identifiquem os seus problemas. Uma das ferramentas mais

utilizadas para a detecção de erros invoca as mensagens *echo request* e *echo reply* do ICMP. Um *host* ou um roteador envia uma mensagem *echo request* para um destino específico. Qualquer máquina que receba esta mensagem formula um *echo reply* e retorna-o ao remetente. O pedido contém uma área opcional de dados; a resposta contém uma cópia dos dados enviados pelo pedido. O pedido e a resposta associados podem ser usados para testar se o destinatário está disponível a responder. Como o pedido e a resposta viajam em datagramas IP, o recebimento com sucesso de uma resposta significa que o sistema de transporte funciona.

As tarefas para o tratamento desta mensagem são as seguintes:

1. O *software IP* na máquina de origem tem de encaminhar o datagrama;
2. Os roteadores intermediários, entre a origem e o destino, têm de estar "ligados", ou seja, têm que estar operando para poderem encaminhar corretamente o datagrama;
3. A máquina de destino tem de esta ligada (tem que responder pelo menos as interrupções) e tanto o software ICMP quanto o IP têm que estar funcionando;
4. O caminho de regresso ao longo dos roteadores tem de ser corretos.

Em muitos sistemas, o comando utilizado para enviar um pedido ICMP *echo request* denomina-se *PING*. Versões sofisticadas do comando *PING*, enviam uma série de pedidos ICMP, capturam as respostas e realizam estatísticas sobre a perda de datagramas. Eles permitem que o utilizador especifique o tamanho dos dados a serem enviados e o intervalo entre os pedidos. Versões menos sofisticadas enviam apenas um pedido ICMP e aguardam a resposta. Nós utilizaremos esta versão menos sofisticada, já que a função aqui não é a de montar um programa ICMP perfeito e sim a de testar a funcionalidade de nossa interface Cheapernet. Para

desenvolver um programa mais avançado, poderíamos perder muito tempo já que todo o programa é feito em linguagem montadora e não nos ajudaria em muito ter um programa ICMP completo, pois se conseguirmos enviar um único *echo reply* para o computador remetente, já conseguiremos provar que a interface funciona.

Portanto o nosso software ICMP conterá apenas as funções de *echo request* e *echo reply*. A partir daí tiraremos as nossas conclusões finais. A figura 26 ilustra o formato da mensagem *echo request* e *echo reply*.

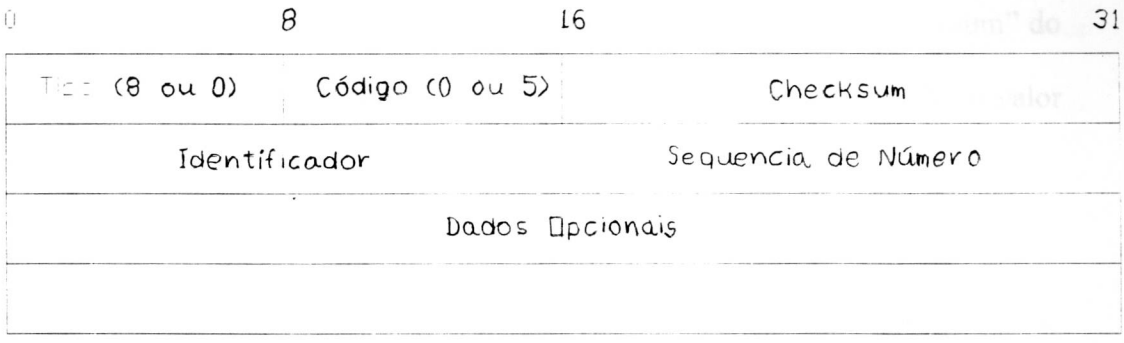


Figura 26 – *Formato da mensagem echo request e echo reply*

A modificação no nosso software original ficará por conta somente das introduções da rotina PING no *driverisr*. Ou seja, após recebermos o pacote, só temos que enviar de volta este mesmo pacote ao *host* remetente, invertendo apenas os campos de endereço destino e endereço de origem tanto no header IP quanto no header Ethernet. Tudo isto é feito dentro da rotina PING que se encontra dentro da rotina *driverisr*, como podemos observar na figura 27.

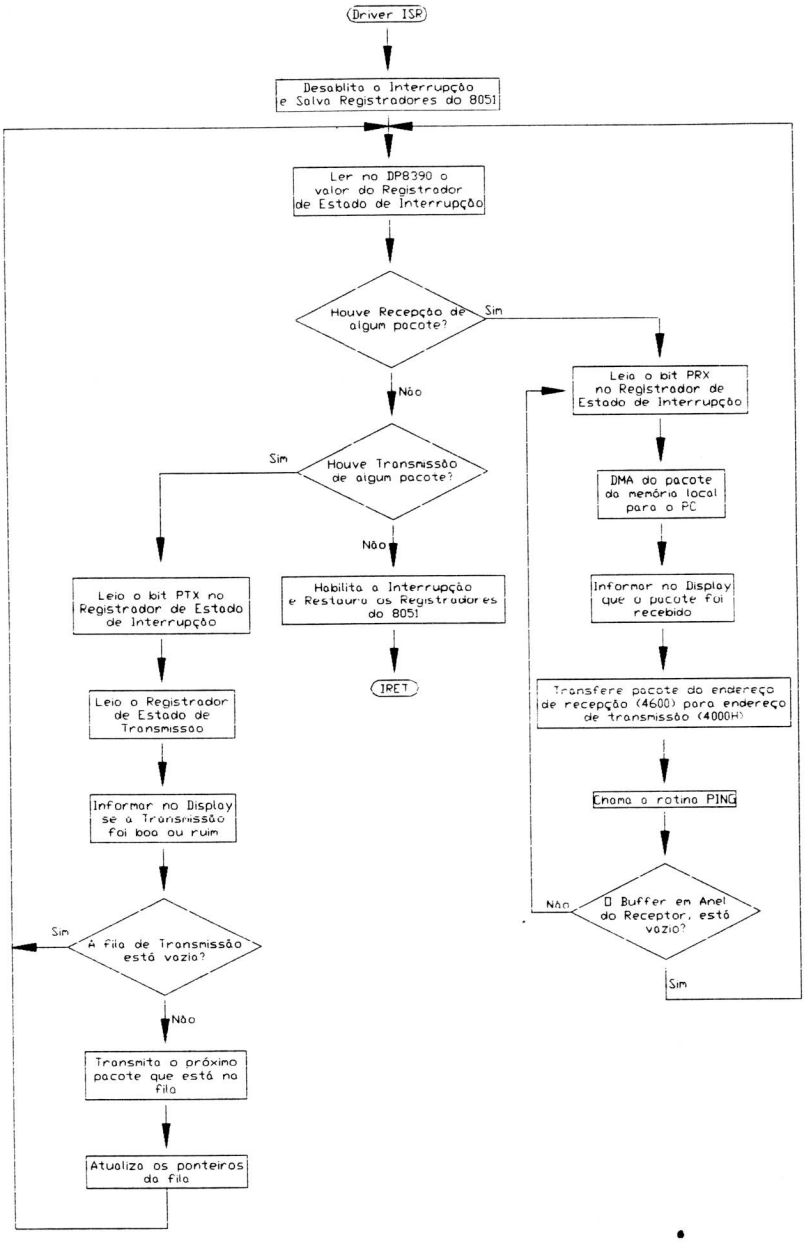


Figura 27 – *Rotina driverisr com a inclusão do PING*

6.2 – Resultados e Conclusões:

A seguir apresentaremos os resultados e conclusões do nosso trabalho.

A interface desenvolvida funcionou corretamente. Depois de implementarmos o PING, foi possível verificar que a interface estava operando

quando ligado em rede, já que ela respondia o pedido de *echo request* de um computador dentro do laboratório e ela também conseguia enviar os dados medidos pela sonda e depois estes mesmos dados eram recebidos de volta através do *echo reply* do computador do laboratório.

Nos apêndices D e E são ilustrados alguns resultados, onde a parte (a) é o *echo request* (a partir do endereço 4000H) enviado pela interface após a medida dos das propriedades físicas e químicas da água e a parte (b) das figuras é o *echo reply* (a partir do endereço 4600H) respondido pelo computador do laboratório.

Podemos observar nos apêndices D e E que praticamente as únicas diferenças existentes entre o pacote enviado e o pacote recebido são: a mudança dos endereços destino e origem dos cabeçalhos ethernet e IP e também o "checksum" do ICMP, já que a mensagem ICMP é mudada no *echo reply* (ao invés de se ter o valor 08, tem-se 00 no endereço 4022H).

Pode ser observado no apêndice F o que ocorre quando enviamos o "PING" do computador para a interface Cheapernet. Na parte (a) temos o pacote recebido pela interface Cheapernet e na parte (b), a interface envia um *echo reply* para o computador, contendo o mesmo pacote de dados.

Mas, nem sempre conseguimos enviar o *echo reply* ou o *echo request* para o computador do laboratório sempre que queríamos. Em alguns testes que fizemos, aparecia a mensagem no display da interface de excesso de colisões na rede.

Excesso este causado pela própria interface Cheapernet. Nós utilizamos um analisador de rede, cedido pelo Instituto de Física, para analisarmos o número de colisões. A conclusão foi a de que possivelmente a própria interface é que gerava as colisões.

Nem tudo que é relatado como colisão pode ser uma colisão "real". As colisões são reais quando dois frames ethernet ou mais se sobrepõe na rede. Infelizmente todos os erros de qualidade dos sinais (SQEs – Signal Quality Errors) relatados pelo *transceiver* são relatados como colisões, mesmo que não seja a sobreposição de dois frames ethernet. Isto ocorre porque o *transceiver* verifica apenas o nível de tensão do sinal na rede para ver se há colisão ou não. Se houver colisão, o nível de tensão na rede atinge um certo valor que então o *transceiver* vê como colisão.

Uma outra conclusão que podemos chegar nesta dissertação é que no Capítulo 2, dissemos que teríamos 3 principais objetivos a serem alcançado para que o funcionamento da placa Cheapernet pudesse ser considerada satisfatório: O *Desempenho, Baixo Custo e Compatibilidade*. Analisaremos então o seu funcionamento a partir desses 3 parâmetros.

1. O Desempenho da Placa Cheapernet:

Se o desempenho de uma placa for quantidade de dados transmitidos e recebidos em um dado período de tempo e seu desempenho é melhor quanto maior for o número de dados, então podemos concluir que a placa Cheapernet teve o seu desempenho satisfatório, pois na velocidade de 10 MHz de transmissão de dados à que nos dispusemos no desenvolvimento desta placa, é uma velocidade aceitável dentro dos nossos objetivos iniciais, ou seja, o nosso intuito foi o de construir uma placa eficiente que atenda às nossas necessidades que é a de mover dados com alta segurança na transmissão. Sabemos que existem placas mais eficientes atualmente, mas após entender o funcionamento desta placa de rede, caberá em futuros trabalhos, o desenvolvimento de uma placa com desempenho melhor. Sabemos também que

tivemos alguns problemas com colisões nesta interface, atrapalhando um pouco o seu desempenho, mas também vale lembrar todos os testes foram feitos na interface Cheapernet montada com a técnica de wire-wrap, que sabemos não ser a melhor condição possível. Acreditamos que depois de feita a placa impressa desta interface o seu desempenho melhorará consideravelmente, e possivelmente reduzirá também os problemas de colisões. Um outro problema que deve ser comentado aqui, que já comentamos no item 3.5 é que para se conseguir uma melhora na operação sinal/ruído no cabo coaxial, é imperativo que o CTI seja soldado na placa sem um soquete. Na nossa placa, o CTI não foi soldado, pois utilizamos a técnica de "wire – wrap", que só pode ser concedida com soquetes. Isto acarretou num aquecimento do dispositivo DP8392. Fica aqui sugerido que nos trabalhos futuros para uma construção de placa de rede, que o CI deva ser soldado na placa para que seja diminuído significantemente a temperatura de operação do dispositivo CTI e assim diminuir a introdução de ruídos no sistema.

2. A relação custo-benefício na construção da Placa Cheapernet:

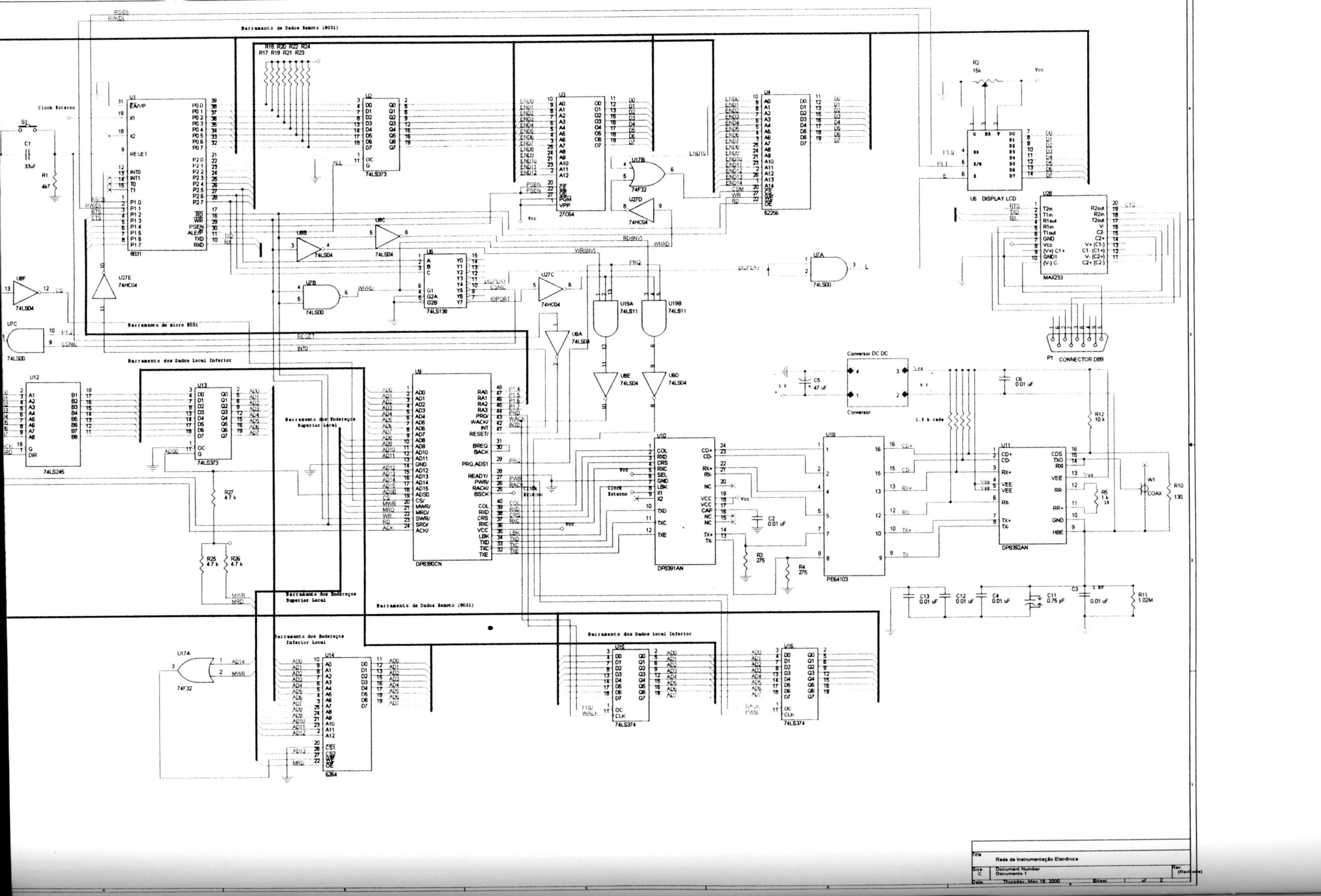
A relação custo-benefício na construção desta placa Cheapernet foi o melhor possível, pois todos os dispositivos utilizados na sua construção (todos os CI's) estavam disponíveis no nosso laboratório. Os CI's que são responsáveis para a transmissão de um pacote na rede, o DP8390, DP8391 e DP8392 foram obtidos como amostras da National. O único dispositivo que não conseguimos a princípio foi o transformador utilizado no conversor DC-DC (ver Apêndice B). Mas nós retiramos este dispositivo de uma placa Cheapernet antiga e assim, conectamos em nossa placa. Como podemos ver, a nossa placa foi construída com um custo muito baixo e portanto tivemos uma relação custo-benefício ótima.

3. Compatibilidade de uma placa Cheapernet:

Se a Compatibilidade de uma placa Cheapernet for a habilidade para trabalhar com um software e um hardware padrões, então diríamos que nossa placa Cheapernet tem uma boa compatibilidade. Nós dizemos uma boa compatibilidade e não ótima, porque apesar de atendermos a necessidade padrão na construção do software, ou seja, o pacote foi desenvolvido de acordo com o pacote padrão do protocolo IEEE 802.3, nosso software não tem comunicação padrão com outros "softwares" de camada superiores tais como UDP, TCP, etc. Para isso, deveríamos sofisticar melhor o software para atender às necessidades desses softwares superiores. Como o nosso objetivo não era de comunicarmos via rede para atendermos estes serviços (TELNET, etc.) e sim o único objetivo de mandarmos um pacote de dados contendo os parâmetros medidos pela sonda na rede, então consideramos que a compatibilidade foi boa.

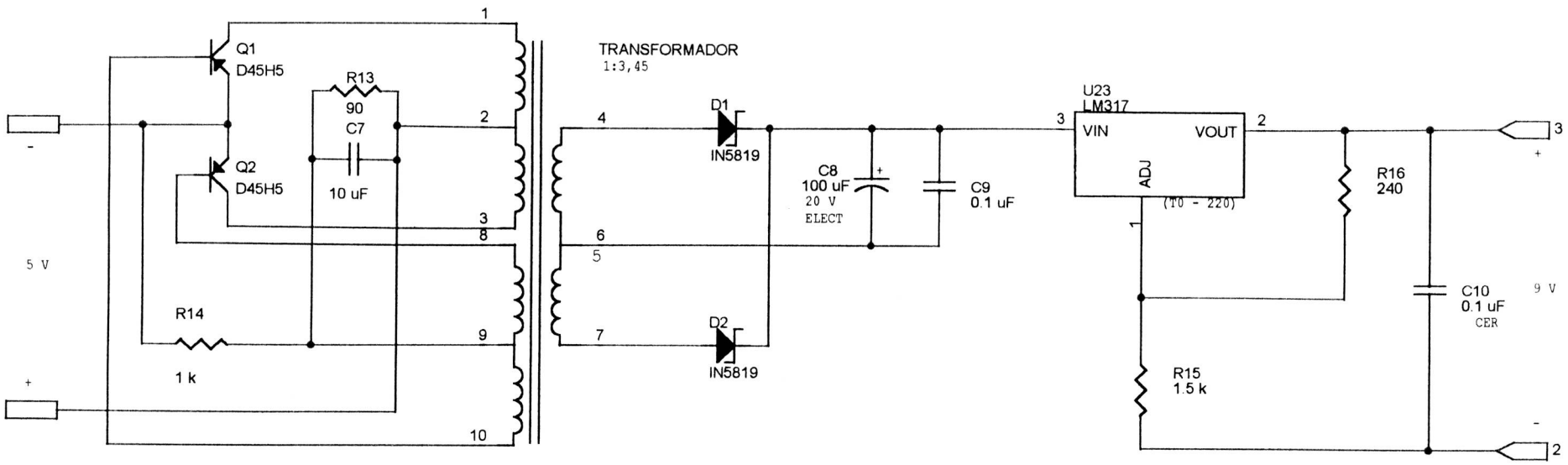
Apêndice A

"Diagrama da Interface Cheapernet controlada pelo 8051"


Barramento de Dados Remoto (8051)
Clock Externo
Barramento micro 8051
Barramento dos Dados Local Inferior
Barramento dos Endereços Superior Local
Barramento de Dados Remoto (8051)
Barramento dos Endereços Inferior Local
Barramento dos Dados Local Inferior
U6 DISPLAY LCD
P1 CONNECTOR DB9
Conversor DC DC
Conversor
1.5 k cada
8031
74LS373
27C64
62256
74LS138
74LS245
DP8390CN
DP8391AN
DP8392AN
PE64103
74LS374
6264
MAX233
COAX
Title
Rede de Instrumentação Eletrônica
Size C
Document Number Documento 1
Rev (Revisão)
Date: Thursday, May 18, 2000
Sheet of

Apêndice B

"Diagrama do conversor DC-DC"



Title	Conversor DC DC	
Size A	Document Number Documento 2	Rev {RevCode}
Date: Thursday, May 18, 2000	Sheet 2 of 2	

Apêndice C

"Programa em linguagem montadora

```
$TITLE(Interface Cheapernet controlada por um 8051 - Versao 1)
$DATE(20-MAR-98)

;  ************************************************************************
;  * Rotinas para o micro 8031                                            *
;  * Primeira versão 20/03/98                                             *
;  * Leonardo Batista de Almeida Scarabelli                               *
;  * Rotinas de operação do Display LCD, de Inicialização do 8390         *
;  ************************************************************************

; Pseudo Variáveis na ROM
CS      equ     0A000H  ; Endereço de memória de acesso ao NIC
Dispa   equ     8000H   ; Endereço da memória de acesso ao display
CR      equ     0Dh     ; Carriage Return
LF      equ     0Ah     ; Line Feed
TSIZE   EQU     600H    ; Tamanho máximo do buffer no Espaço de dados Externo
RSIZE   EQU     1A00H   ; Tamanho máximo do buffer no Espaço de dados Externo

        DEFSEG  TRANS,CLASS=XDATA,START=4000H
        SEG     TRANS           ; Espaço de endereço para Transmitir os Dados

        DS      TSIZE

        DEFSEG  RECEP,CLASS=XDATA,START=4600H
        SEG     RECEP           ; Espaço de endereço para Transmitir os Dados

        DS      RSIZE

        DEFSEG  QUEUE,CLASS=XDATA,START=6000H
        SEG     QUEUE           ; Espaço de endereço para enfileirar os Dados

        DS      TSIZE

        DEFSEG  DISPLAY,CLASS=XDATA,START=8000H
        SEG     DISPLAY         ; Espaço de endereço para mostrar mensagem
                                ; no Display
        DS      2000H

        DEFSEG  NIC,CLASS=XDATA,START=0A000H
        SEG     NIC             ; Espaço de endereço para mover dados da/para
                                ; CPU para/do NIC

        DS      2000H

        DEFSEG  PORTIO,CLASS=XDATA,START=0E000H
        SEG     PORTIO          ; Espaço de endereço para acessar os dados no
                                ; barramento local

        DS      2000H

        %include "8390.inc"

; Page 1

EN1_PHYS        EQU     001h    ; Este , o endereço físico da interface
EN1_CURPAG      EQU     007h    ; Mensagem da Memória Corrente(CURRENT)
EN1_MULT        EQU     008h    ; Endereço Multicast desejado
```

```
; Comando do Chip no Registrador de Comando (CR)
CR_START        EQU     002h      ; Chip no modo Start
CR_NODMA        EQU     020h      ; Não , usado DMA remoto nesta interface
CR_PAGE0        EQU     021h      ; Seleciona page 0 nos registradores do chip
CR_PAGE1        EQU     061h      ; Seleciona page 1 nos registradores do chip

; Comandos para os registradores de controle de RX (RCR)
RCR_ANY_PCT     EQU     00h       ; Não aceito pacotes com qualquer tipo de
                                  ; erros, nem pacotes broadcast, multicast
                                  ; ou pacotes anões.

; Comando para o registrador de controle de TX (TCR)
TCR_RESET       EQU     00h       ; Reseto inicialmente o transmissor
TCR_LOOP        EQU     002h      ; Loopback no modo 1, com loop interno

; Bits do Registrador de Configuração de Dados (DCR)
DCR_MODE        EQU     058h      ; Seto o modo burst com:
                                  ; 8 bytes de largura da FIFO
                                  ; Executa o comando "Send Packet" que vai
                                  ; executar a remoção dos pacotes do Buffer Ring
                                  ; Seleciona loopback na operação normal

; Bits do Registrador de Estado de Interrupção (ISR)
ISR_APAGA       EQU     0ffh      ; Apaga o valor de ISR

; Bits do Registrador de Máscara de Interrupção (IMR)
IMR_TRANS_ERR   EQU     00Bh      ; Habilita interrupção quando o pacote foi
                                  ; transmitido ou quando houver erro na tx.

; Bits Endereçáveis dos Registradores Especiais (SFR's)
CSN     BIT P1.2          ; Endereço do pino P1.2, usado como CS do NIC
ex_tx   BIT 20H.1         ; Endereço de memória que nos diz se houve erro na
                          ; transmissão
Check   BIT 20H.2         ; Endereço de memória que nos diz que pacote foi
                          ; enfileirado
first_reception BIT 20H.3       ; Endereço de memória que nos diz se e a
                                ; primeira recepção
jabber  BIT 20H.4         ; Endereço de memória que nos diz se existe algum
                          ; problema elétrico no nosso sistema (jabber)


; Endereço da memória para enfileirar o pacote na fila da memória do
; microprocessador
NICRAM  DATA 22H          ; Endereço da memória do controlador (transmissão)
RERAM   DATA 24H          ; Endereço da memória do microprocessador (recepção)
PCRAM   DATA 26H          ; Endereço da memória do microprocessador (transmissão)
tailptr DATA    28H       ; Valor do próximo buffer a ser lido
headptr DATA    2AH       ; Valor do próximo espaço disponível no buffer
byte_count      DATA 2CH; Valor do contador de byte para transmissão
next_packet     DATA 2EH; Valor do próximo buffer a ser lido


        DEFSEG  TRANS_PC_BUFF,CLASS=CODE,START=0050H
        SEG     TRANS_PC_BUFF   ; Espaço para o Endereço de Dados Externo
        DB      'TRANSMISSAO  BEM  SUCEDIDA',CR
```

```
        DB      'HOUVE  ERRO  NA  TRANSMISSAO',CR

        DEFSEG  ENDE_ETHERNET,CLASS=CODE,START=0091H
        SEG     ENDE_ETHERNET   ; Espaço de endereço para o endereço ethernet

        public  io_addr, dest_MAC, card_hw_addr, tam_pacote
io_addr          dw    0300h,0           ; I/O address for card (jumpers)
dest_MAC         db    00,0C0h,0DFh,0E3h,05Ah,7Ch  ;Endereço da placa Destino
card_hw_addr     db    08h,00,17h,01h,09h,74h     ;Endereço Ethernet Físico
tam_pacote       dw    0800h              ;campo onde especificamos que será um
                                          ;pacote IP
                 db    0Dh                ;final do pacote

; Endereço de memória para colocarmos no PCRAM o header IP
END_IP           EQU   0C0h     ; Endereço para montarmos o header IP

        DEFSEG  ENDE_IP,CLASS=CODE,START=00C0H
        SEG     ENDE_IP         ; Espaço de endereço para o endereço IP

        public version_IHL, TOS, tam_Total, Identificacao, offset_frag, TTL,
protocolo, Checksum_IP, dest_IP, fonte_IP
version_IHL      db    45h
TOS              db    00h
tam_Total        dw    0051h              ;tamanho do pacote a ser transmitido
                                          ;(100 bytes)
Identificacao    dw    9E00h              ;para Identificar o Header (6Fh hex.)
offset_frag      dw    00
TTL              db    20h                ; 32 segundos
protocolo        db    01                 ; ICMP
Checksum_IP      dw    0000h              ; Para efeitos de calculo
fonte_IP         db    143,107,235,201 ; Endereço IP da minha interface
dest_IP          db    143,107,235,204 ;Endereço IP do host Destino
                 db    0Dh                ;final do pacote

; Endereço de memória para colocarmos no PCRAM o header ICMP
END_ICMP         EQU   0E0h     ; Endereco para montarmos o header ICMP

        DEFSEG  ENDE_ICMP,CLASS=CODE,START=00E0H
        SEG     ENDE_ICMP       ; Espaço de endereço para o endereço ICMP

        public  TIPO, CODIGO, Checksum_ICMP, Identificador, Sequ_Numero
TIPO             db    08h      ; Echo request (para echo replay TIPO = 0)
CODIGO           db    00h
Checksum_ICMP    dw    0000h              ; Para efeitos de calculo
Identificador    dw    0100h              ; Para Identificar o Header
Sequ_Numero      dw    0100h
;driver_name      db     'Leonardo Batista de Almeida Scarabelli'
                                          ;nome do driver.
                 db    0Dh                ;final do pacote

        DEFSEG  FU,CLASS=CODE,START=0110H
        SEG     FU      ; Espaço para o Endereço de Dados Externo
        DB      ' - NIC NAO PODE ACESSAR BARRAMENTO (FU)',CR

        DEFSEG  CRS,CLASS=CODE,START=0140H
        SEG     CRS      ; Espaço para o Endereço de Dados Externo
        DB      ' - A PORTADORA DO PACOTE FOI PERDIDA (CRS)',CR
```

```
        DEFSEG  ABT,CLASS=CODE,START=0170H
        SEG     ABT         ; Espaco para o Endereço de Dados Externo
        DB      ' - EXCESSO DE COLISOES (ABT)',CR

        DEFSEG  DFR,CLASS=CODE,START=01A0H
        SEG     DFR         ; Espaço para o Endereço de Dados Externo
        DB      ' - PROBLEMAS DE JABBER (DFR)',CR

        DEFSEG  ZERO,START=0
        SEG     ZERO

        ljmp START    ; Desvia para outra posição para deixar espaço para
                        ; as interrupções

        DEFSEG  EXTI,START=EXTI0,CLASS=CODE
        SEG     EXTI

        ljmp DRIVERISR  ; pula para rotina de interrupção

        DEFSEG  MAIN,CLASS=CODE,START=200H
        SEG     MAIN

        EXTERN MAINLY

START
        mov sp,#31h
        lcall PRINCI
        ljmp MAINLY


;  ***************************************************************************
;  * Rotina Principal                                                        *
;  ***************************************************************************

PRINCI  PROC

      setb CSN          ; Desabilito o CS do NIC
        mov TMOD,#21H   ; Timer 1 para contar em 8 bits com recarga automática
        mov PCON,#00h   ; Timer 0 - Gerador de Baud Rate. SMOD = 0.
        mov TCON,#00H   ; Desliga timer 0 e Timer 0.
        setb RS0        ; Coloco no banco de registrador 1
        lcall INICP     ; Inicializa a placa ethernet
        lcall RESEND    ; Inicializa variáveis para transmissão e recepção
        clr RS0         ; Volto no banco de registrador 0
        lcall SERIAL_INI; Rotina que inicializa o serial
        ret

        ENDPROC

; DRIVERSEND , uma Rotina que inicia a transmissão, se o software das camadas
; superiores passarem um pacote para o driver. Dentro da Rotina do DRIVERSEND
; tem as Rotinas HEADER, HEADER_IP e HEADER_ICMP põem respectivamente os
; headers Ethernet, IP e ICMP no Endereco da memória que ser  transmitido o
; pacote e PCtoNIC que transmite o pacote do PC para o NIC.
; Se o driver não está habilitado para transmitir o pacote imediatamente
; (isto ,, o transmissor está ocupado), o pacote fornecido , enfileirado em
```

```
; um buffer para transmissão pendente (Rotina QUEUE_PACKET). E Finalmente,
; temos a Rotina UTIL_DPTRDEC que decrementa o valor de DPTR e se houver
; underflow o carry , setado. A Rotina CONTADOR, simplesmente pega os 4
; primeiros bytes do pacote recebido para saber o tamanho do pacote e em qual
; endereço foi escrito o próximo pacote, se , que existe um próximo pacote.

        EXTERN DRIVERSEND
        EXTERN CONTADOR
        EXTERN HEADER
        EXTERN HEADER_IP
        EXTERN HEADER_ICMP
        EXTERN PCtoNIC
        EXTERN QUEUE_PACKET
        EXTERN UTIL_DPTRDEC

; DRIVERISR , uma Rotina de serviço de interrupção e ela opera em conjunto
; com DRIVERSEND. O driver de transmissão , parcionado em duas partes, uma
; no driverisr e outra no Driversend, enquanto o driver de recepção reside
; inteiramente no driverisr. driverisr , concernido como interrupção
; originada de recepções, transmissões e erros na transmissão. As recepções
; erradas são ignoradas.

        EXTERN DRIVERISR


; NICtoPC , uma Rotina que transmite a mensagem que foi recebida pelo NIC.
; Esta rotina funciona somente dentro do driverisr.

        EXTERN NICtoPC

; TX_MEN , uma Rotina que transmite uma mensagem no display, para avisar que
; não houve erro na transmissão do pacote. ER_MEN , uma Rotina que também
; transmite uma mensagem no display, mas ele avisa que houve erro na
; transmissão do pacote. Ambas as Rotinas funcionam somente dentro do
; DRIVEISR. A rotina INICD simplesmente põe o display no modo inicial para
; poder ser escrito qualquer mensagem.

        EXTERN TX_MEN
        EXTERN ER_MEN
        EXTERN INICD
        EXTERN SERIAL_INI

; ************************************************************************
; * Descrição:                                                           *
; *     Rotina de Inicialização da placa ethernet para poder receber ou  *
; *     transmitir um pacote.                                            *
; * Entrada Requerida:                                                   *
; *     DPTR Contém o Endereço para inicializar o CS da NIC              *
; * Na Saída:                                                            *
; *     Nada                                                             *
; * Afetado:                                                             *
; *     DPTR, Acc e P1                                                   *
; * Comentários:                                                         *
; *     Recepção do Buffer Ring = 2600H até 4000H                        *
; *     Transmissão do Buffer   = 2000H até 25FFH                        *
; ************************************************************************
```

```
INICP   PROC
        COMMAND          ; Inicializo o Comando dos Registradores (CR)
        mov A,#CR_PAGE0  ; Inicializo o Registrador no Page 0
        movx @DPTR,A     ; Envia o comando para os registradores NIC

        DATACONFIGURATION        ; Inicializo o Registrador de Comando de
                                 ; Dados (DCR)
        mov A,#DCR_MODE  ; Inicializo o DCR em 8 bytes e LSB no AD0-AD7
        movx @DPTR,A     ; Envia o comando para os registradores NIC

        REMOTEBYTECOUNT0         ; Inicializo o Registrador Contador de
                                 ; Byte Remoto 0 (RBCR0)
        clr A            ; Apago o valor de RBCR0
        movx @DPTR,A     ; Envia o comando para os registradores NIC

        REMOTEBYTECOUNT1         ; Inicializo o Registrador Contador de
                                 ; Byte Remoto 0 (RBCR1)
        clr A            ; Apago o valor de RBCR1
        movx @DPTR,A     ; Envia o comando para os registradores NIC

        RECEIVECONFIGURATION     ; Inicializo o Registrador de
                                 ; Configuração de Recepção (RCR)
        mov A,#RCR_ANY_PCT       ; Pacotes anões ou com erros serão rejeitados.
                                 ; Endereço Físico do nó = Endereço Físico da
                                 ; Estação em PAR0-PAR5
        movx @DPTR,A     ; Envia o comando para os registradores NIC

        TRANSMITCONFIGURATION    ; Inicializo o Registrador de Configuração
                                 ; de Transmissão (TCR)
        mov A,#TCR_LOOP  ; Loopback no modo 1, com loop interno
        movx @DPTR,A     ; Envia o comando para os registradores NIC

        TRANSMITPAGE     ; Inicializo o Registrador de Início de Transmissão
                         ; (TPSR)
        mov A,#20H       ; Transmite Page em 20H
        movx @DPTR,A     ; Envia o comando para os registradores NIC

        PAGESTART        ; Inicializo o Registrador de Endereço Inicial do
                         ; pacote (PSTART)
        mov A,#26H       ; O Registrador de Endereço Inicial aponta
                         ; para o Endereço Inicial na Recepção do pacote
                         ; Buffer Ring
        movx @DPTR,A     ; Envia o comando para os registradores NIC

        BOUNDARY         ; Inicializo o Ponteiro Limite (BNDRY)
        mov A,#26H       ; O ponteiro limite aponta para o primeiro
                         ; pacote do anel que ainda não foi lido pelo
                         ; host.
        movx @DPTR,A     ; Envia o comando para os registradores NIC

        PAGESTOP         ; Inicializo o Registrador de Endereço Final do
                         ; pacote (PSTOP)
        mov A,#40H       ; O Registrador de Endereco Final aponta
                         ; para o Endereço Final na Recepção do pacote
                         ; no Buffer Ring
        movx @DPTR,A     ; Envia o comando para os registradores NIC
```

```
; Nesta parte, inicializaremos os registradores de Endereço Físico com o
; Endereço Ethernet (MAC) especificado por mim. O endereço ser  colocado
; nos registradores (PAR0-PAR5) e ter  o seguinte valor: (0C:00:17h:01:09:74h)

        COMMAND          ; Inicializo o Comando dos Registradores (CR)
        mov A,#CR_PAGE1  ; Inicializo o Registrador no Page 1
        movx @DPTR,A     ; Envia o comando para os registradores NIC
        mov DPTR,#card_hw_addr  ; Endereço do hardware (009Bh)
        push 00
        mov R0,#00h      ; Para ver em qual registrador físico devo escrever

LP?1    clr A            ; Apago Acc para ler tabela do tamanho
                         ; do hardware (tam = 6)
        push DPH         ; Salvo MSB da tabela a ser lida
        push DPL         ; Salvo LSB da tabela a ser lida
        movc A,@A+DPTR   ; Pego o primeiro endereço do hardware
        push Acc         ; Salvo endereço do hardware na pilha
        mov A,R0         ; Pego o valor de R0
        cjeq A,#00h,LP?2       ; se R0=00h, escrever no registrador físico 0
        cjeq A,#01h,LP?3       ; se R0=01h, escrever no registrador físico 1
        cjeq A,#02h,LP?4       ; se R0=02h, escrever no registrador físico 2
        cjeq A,#03h,LP?5       ; se R0=03h, escrever no registrador físico 3
        cjeq A,#04h,LP?6       ; se R0=04h, escrever no registrador físico 4
        cjeq A,#05h,LP?7       ; se R0=05h, escrever no registrador físico 5

LP?2    REGISTERFISIC0   ; Vou escrever no registrador físico 0 do NIC
        pop Acc          ; Pego de volta o endereço do hardware na pilha
        movx @DPTR,A     ; Envia o comando para os registradores NIC
        inc R0           ; Agora R0 = 01h
        ljmp LP?8        ; ir para LP?8

LP?3    REGISTERFISIC1   ; Vou escrever no registrador físico 1 do NIC
        pop Acc          ; Pego de volta o endereço do hardware na pilha
        movx @DPTR,A     ; Envia o comando para os registradores NIC
        inc R0           ; Agora R0 = 02h
        ljmp LP?8        ; ir para LP?8

LP?4    REGISTERFISIC2   ; Vou escrever no registrador físico 2 do NIC
        pop Acc          ; Pego de volta o endereço do hardware na pilha
        movx @DPTR,A     ; Envia o comando para os registradores NIC
        inc R0           ; Agora R0 = 03h
        ljmp LP?8        ; ir para LP?8

LP?5    REGISTERFISIC3   ; Vou escrever no registrador físico 3 do NIC
        pop Acc          ; Pego de volta o endereço do hardware na pilha
        movx @DPTR,A     ; Envia o comando para os registradores NIC
        inc R0           ; Agora R0 = 04h
        ljmp LP?8        ; ir para LP?8

LP?6    REGISTERFISIC4   ; Vou escrever no registrador físico 4 do NIC
        pop Acc          ; Pego de volta o endereço do hardware na pilha
        movx @DPTR,A     ; Envia o comando para os registradores NIC
        inc R0           ; Agora R0 = 05h
        ljmp LP?8        ; ir para LP?8

LP?7    REGISTERFISIC5   ; Vou escrever no registrador físico 5 do NIC
        pop Acc          ; Pego de volta o endereço do hardware na pilha
```

```
        movx @DPTR,A    ; Envia o comando para os registradores NIC
        ljmp LP?8       ; ir para LP?8

LP?8    pop DPL         ; Pego de volta o MSB da tabela
        pop DPH         ; Pego de volta o LSB da tabela
        inc DPTR        ; Incremento DPTR para pegar o próximo endereço
        mov A,DPL
        cjne A,#0A1H,LP?9       ; Vê se , o último caracter a ser lido...
        sjmp LP?10
LP?9    ljmp LP?1

LP?10   pop 00
        CURRENT         ; Inicializo o Registrador de Page Corrente (CURR).
        mov A,#26H      ; O Registrador de Page Corrente aponta para o
                        ; o primeiro buffer usado para armazenar o pacote
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        COMMAND         ; Inicializo o Comando dos Registradores (CR)
        mov A,#CR_NODMA+CR_START        ; Volto o Registrador no Page 0,
                                        ; no modo Start
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        INTERRUPTSTATUS ; Inicializo o Registrador de Status de
                        ; Interrupção (ISR)
        mov A,#ISR_APAGA; □ apagado o valor de ISR
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        INTERRUPTMASK   ; Inicializo o Registrador de Máscara de
                        ; Interrupção (IMR)
        mov A,#IMR_TRANS_ERR    ; Interrupções habilitadas para quando
                                ; haver overflow nos buffers e quando os
                                ; contadores da rede estiverem setados.
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        TRANSMITCONFIGURATION   ; Inicializo o Registrador de Status de
                                ; Configuração de Transmissão (TCR)
        mov A,#TCR_RESET; Desabilita as interrupções. TCR no modo normal
        movx @DPTR,A    ; Envia o comando para os registradores NIC

; NIC agora esta pronta para recepção.


        ret

        ENDPROC

; ********************************************************************************
; * Descrição:                                                                   *
; *     Rotina da placa ethernet para poder enviar uma cadeia de bytes ou        *
; *     para o buffer receber uma mesma cadeia byte                              *
; * Entrada Requerida:                                                           *
; *     Nenhuma                                                                  *
; * Na Saída:                                                                    *
; *     R0 = LSB do Endereço do PC que será transmitido                          *
; *     R1 = MSB do Endereço do PC que será transmitido                          *
; *     R2 = LSB do Contador de Byte Remoto                                      *
; *     R3 = MSB do Contador de Byte Remoto                                      *
```

```
; *      R4 = LSB do Endereço de Entrada/Saída dos dados                    *
; *      R5 = MSB do Endereço de Entrada/Saída dos dados                    *
; *      R6 = LSB do Endereço Inicial Remoto                                *
; *      R7 = MSB do Endereço Inicial Remoto                                *
; * Afetado:                                                                *
; *      R0, R1, R2, R3, R4, R5, R6 e R7 do banco de registradores 0        *
; * Comentários:                                                            *
; *      Recepção do Buffer Ring = 2600H até 4000H                          *
; *      Transmissão do Buffer   = 2000H até 25FFH                          *
; ************************************************************************

RESEND  PROC

        mov RERAM,#46H          ; Valor MSB de onde receberemos o pacote que
                                ; vem da RAM do NIC
        mov (RERAM+1),#00H      ; Valor LSB de onde receberemos o pacote que
                                ; vem da RAM do NIC

        mov PCRAM,#40H          ; Valor MSB de onde enviaremos o pacote para
                                ; a RAM do NIC
        mov (PCRAM+1),#00H      ; Valor LSB de onde enviaremos o pacote para
                                ; a RAM do NIC

        mov tailptr,#60H        ; Valor MSB do próximo buffer a ser lido
        mov (tailptr+1),#00H    ; Valor LSB do próximo buffer a ser lido

        mov next_packet,#26H    ; Valor onde o próximo pacote será lido (MSB)
        mov (next_packet+1),#00H; Valor onde o próximo pacote será lido (LSB)

        mov NICRAM,#20H         ; Salvo em NICRAM o valor do LSB do Endereço
                                ; Inicial Remoto
        mov (NICRAM+1),#00H     ; Salvo em NICRAM+1 o valor do MSB do Endereço
                                ; Inicial Remoto

        mov byte_count,#00H     ; Valor MSB do contador de byte para
                                ; transmissão
        mov (byte_count+1),#04AH; Valor LSB do contador de byte para
                                ; transmissão
        clr ex_tx
        clr Check
        clr first_reception
        clr jabber
        ret

        ENDPROC

        END
```

```
;*************************************************************************
; Biblioteca  que interpreta e executa comandos recebidos pela RS232-C
; para gerenciamento da memória de cartão PCMCIA.
;*************************************************************************

        PUBLIC MAINLY

;*************************************************************
; main : Ponto de entrada. Rotina que interpreta comandos.
; Recebe um comando numérico pela serial e realiza a execução
; do mesmo a partir de uma tabela de desvio.
;*************************************************************
mainly:
next_cmd:
           lcall serial_rcv ; Recebe comando pela serial em Acc.
           mov R0,A         ; ... para calculo do endereço ...
           mov DPTR,#cmd_tbl; ... donde se encontra instrução ...
           rl A             ; ... que deve ser executada.
           add A,R0
           jmp @A+DPTR      ; Vai para o comando a executar.
cmd_tbl:
          ljmp cmd_init    ; Comando 00h -> Comando Dummy.
          ljmp cmd_write   ; Comando 01h -> Carrega Buffer com dados ...
                           ; ... provenientes da porta serial ...
                           ; ... e transmite Buffer via rede.
          ljmp cmd_read    ; Comando 02h -> Comando Dummy
          ljmp cmd_format  ; Comando 03h -> Comando Dummy.
          ljmp cmd_reset   ; Comando 04h -> Comando Dummy
          ljmp cmd_connect ; Comando 05h -> Comando Dummy
          ljmp cmd_wr_eot  ; Comando 06h -> Comando Dummy
          ljmp cmd_fim     ; Comando 07h -> Comando Dummy.


EOT     EQU  04h
LFCR    EQU  0Ah
BUFFER  EQU 402Ah
;***************  Módulos externos para comunicação serial ***************

          EXTERN  Serial_rcv
          EXTERN  Serial_snd

;*************************************************************************
; Uso dos registradores e memórias:
;
; - R3R2 do banco 1 -> Ponteiro para leitura de frames.
;
; - R5R4 do banco 1 -> Ponteiro para escrita de frames.
;
; - Posição de memória 0000h e 0001h do PC CARD  -> Espaço de memória para
; armazenamento do ponteiro para escrita de frames.
;
; - Posição de memória 0002h e 0003h do PC CARD -> Espaço de memória para
; armazenamento do ponteiro para leitura de frames.
;*************************************************************************

; cmd_init : Inicialização. Primeiro comando a ser executado.
; Carrega os ponteiros de escrita e leitura de dados
```

```
; da memória do cartão para os registradores R3R2 e R5R4 do banco 1.
; Em caso de sucesso, retorna pela serial o código 0FFh.
; Deve ser usado este comando em cartões previamente formatados através
; do comando cmd_format

cmd_init  PROC

        mov A,#0FFh       ; Sinalizador de operação bem sucedida.
        lcall serial_snd
        jmp next_cmd      ; Vai esperar o próximo comando.


        ENDPROC

; cmd_connect : Sinaliza se ha cartão conectado a interface.

cmd_connect PROC

       mov A,#0FFh       ; Sinalizador de sucesso.
       lcall serial_snd
       jmp next_cmd

       ENDPROC

; cmd_write : Escrita de dados na memória do cartão.
; Recebe dados de aquisição provenientes da serial e armazena
; no espaco de memória do cartão. Deve ser finalizado com cmd_fim

cmd_write  PROC

          mov DPTR,#BUFFER        ; DPTR -> Endereço do buffer de dados.
          mov A,#0FFh             ; Faz de conta que sempre tá tudo legal.
          lcall serial_snd

l?prox_byte:

          lcall serial_rcv        ; recebe byte pela serial em Acc
          movx @DPTR,A            ; Armazena byte recebido em buffer
          inc DPTR                ; Aponta para próxima posição.
          cjne A,#LFCR,l?prox_byte ; Testa por LFCR.

          lcall DRIVERSEND        ; Transmite buffer através da interface
          jmp next_cmd            ; de rede.

          ENDPROC

; cmd_wr_eot: Escreve marcador de fim (EOT) na posição atual de escrita.
; Incrementa ponteiro de escrita para próxima posição.
; Deve ser usado este comando para indicar finalizador de dados escritos.

cmd_wr_eot PROC

           mov A,#0FFh
           lcall serial_snd
           ljmp next_cmd

           ENDPROC
```

```
; cmd_read : Leitura de dados do PC CARD.
; Leitura de dados ocorre ate que se encontre o código EOT.
; Ponteiro de leitura, ao final do processo, eh posicionado na posição
; de memória subsequente a aquela na qual o código EOT foi encontrado.

cmd_read    PROC

            mov A,#0ffh
            lcall serial_snd        ; Faz de conta que tá tudo ok sempre.
            jmp next_cmd

            ENDPROC

; cmd_format : Formata cartão de memória.
; Apaga qualquer dado que esteja armazenado no cartão.
; Posiciona os ponteiros de escrita e leitura no inicio da área
; de armazenamento de dados. Preenche toda área de armazenamento com EOT.


cmd_format  PROC

            mov A,#0FFh
            lcall serial_snd
            jmp next_cmd

            ENDPROC

; cmd_reset: Reposiciona ponteiro de leitura de dados
; no primeiro registro de dados do cartão de memória.

cmd_reset PROC
          mov A,#0FFh
          lcall serial_snd

          jmp next_cmd

          ENDPROC

; cmd_fim : Marca final de bloco de dados que foram escritos através
; do comando cmd_write.
; Salva status dos ponteiros de leitura e escrita de dados.
; Desconecta cartão de memória da alimentação.
; Utiliza-se este comando para sinalizar o final de uma seção de gravação
; ou leitura de dados.

cmd_fim PROC
            mov A,#0ffh         ; Código de sucesso
            lcall serial_snd
            jmp next_cmd

            ENDPROC

; Rotinas incluídas do modulo ethernet

            EXTERN DRIVERSEND
       END
```

```
; ************************************************************************
; * Descrição:                                                           *
; *     Driver Send transmite um pacote que foi passado para ele ou      *
; *     enfileira o pacote se o transmissor estiver ocupado.             *
; *     (Registrador de COMANDO = 26h).                                  *
; * Entrada Requerida:                                                   *
; *     DPTR = contador de byte do pacote = byte_count = 0586h --> 1414  *
; *     R6 = LSB do Endereço Inicial Remoto = 00h                        *
; *     R7 = MSB do Endereço Inicial Remoto = 20h                        *
; *     R2 = LSB do Contador de Endereço = 05h --> 1414 bytes            *
; *     R3 = MSB do Contador de Endereço = 86h                           *
; *     R0 = LSB do Endereço da RAM do PC = 00h                          *
; *     R1 = MSB do Endereço da RAM do PC = 40h                          *
; * Na Saída:                                                            *
; *     Nada                                                             *
; * Afetado:                                                             *
; *     DPTR, R0, R1, R2, R3, R6, R7 (do banco de registrador 0)         *
; * Comentários:                                                         *
; *     Esta Rotina , chamada por um software das camadas superiores.    *
; ************************************************************************

        PUBLIC DRIVERSEND
        PUBLIC PCtoNIC
        PUBLIC UTIL_DPTRDEC
        EXTERN QUEUE_PACKET
        EXTERN HEADER
        EXTERN HEADER_IP
        EXTERN HEADER_ICMP

; Pseudo Variáveis na ROM
CS      equ 0A000H      ; Endereço de memória de acesso ao NIC
IOPORT  equ 0E000H      ; Endereço de memória para acessar os dados no
                        ; barramento local
TransmitBuffer  equ 4000H          ; Endereço da transmissão do Buffer
; Endereço da memória para enfileirar o pacote na fila da memória do
; microprocessador
NICRAM  DATA 22H        ; Endereço da memória do controlador (transmissão)
PCRAM   DATA 26H        ; Endereço da memória do microprocessador
byte_count      DATA 2CH        ; Valor do contador de byte para transmissão


        %include "8390.inc"

; Comando do Chip no Registrador de Comando (CR)
CR_ESCR         EQU     012h    ; Escreve no Registrador
CR_TRANS        EQU     026h    ; Transmite um frame

; Bits do Registrador de Estado de Interrupção (ISR)
ISR_DMA_DONE    EQU     040h    ; Verificar se o DMA está completo

; Bit endereçável na RAM interna
Check   BIT 20H.2       ; Endereço de memória que nos diz que pacote foi
                        ; enfileirado
CSN     BIT P1.2        ; Endereço do pino P1.2, usado como CS do NIC


DRIVERSEND      PROC
```

```
        clr EA          ; Todas as Interrupções são desabilitadas
        COMMAND         ; Inicializo o Comando dos Registradores (CR)
        movx A,@DPTR    ; Vou ler os registradores NIC
        xrl A,#CR_TRANS ; Está transmitindo?
        jnz SAI         ; Se não estiver, continua
        ljmp QUEUE_IT   ; Se estiver, enfileira o pacote

SAI:
        mov DPTR,#IOPORT; IOPORT aponta para o porto de entrada e saída
        mov R4,DPL      ; Salvo em R4 o valor do LSB do Endereço para acessar
                        ; o porto de entrada e saída
        mov R5,DPH      ; Salvo em R5 o valor do MSB do Endereço para acessar
                        ; o porto de entrada e saída
        mov R0,#NICRAM  ; NICRAM aponta para o pacote que ser  transmitido
        mov R1,#NICRAM+1; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0     ; Movo para DPH o MSB do endereço
        mov DPL,@R1     ; Movo para DPL o LSB do endereço
        mov R6,DPL      ; Salvo em R6 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R7,DPH      ; Salvo em R7 o valor do MSB do Endereço Inicial
                        ; Remoto
        push 06         ; Salvo LSB do Endereço Inicial Remoto na pilha
        push 07         ; Salvo MSB do Endereço Inicial Remoto na pilha

        mov R0,#PCRAM   ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1 ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0             ; Movo para DPH o MSB do endereço
        mov DPL,@R1             ; Movo para DPL o LSB do endereço
        mov R1,DPL      ; Salvo em R0 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R0,DPH      ; Salvo em R1 o valor do MSB do Endereço Inicial
                        ; Remoto
        push 00         ; Salvo MSB do Endereço Inicial da RAM do PC
        push 01         ; Salvo LSB do Endereço Inicial da RAM do PC

        mov R0,#byte_count      ; Valor do contador de byte para transmissão
        mov R1,#(byte_count+1)  ; Valor do contador de byte para transmissão
        mov DPH,@R0             ; Movo para DPH o MSB do contador
        mov DPL,@R1             ; Movo para DPL o LSB do contador

        mov R2,DPL      ; Salvo em R2 o valor do LSB do Contador de Bytes
                        ; Remoto
        mov R3,DPH      ; Salvo em R3 o valor do MSB do Contador de Bytes
                        ; Remoto
        pop 01          ; Pego de volta MSB da RAM do PC
        pop 00          ; Pego de volta LSB da RAM do PC
        pop 07          ; Pego de volta MSB do Endereço Inicial Remoto
        pop 06          ; Pego de volta LSB do Endereço Inicial Remoto
        push DPL        ; Salvo contador na pilha
        push DPH
        lcall HEADER    ; Chamo rotina que põe header no PCRAM
        lcall HEADER_IP ; Chamo rotina que põe header IP no PCRAM
        lcall HEADER_ICMP       ; Chamo rotina que põe header ICMP no PCRAM
        acall PCtoNIC   ; Transfere pacote do PC para o Buffer RAM do NIC

        TRANSMITPAGE    ; Inicializo o Registrador de Início de Transmissão
```

```
                        ; (TPSR)
        mov R0,#NICRAM  ; NICRAM aponta para o pacote que ser  transmitido
        mov A,@R0       ; Movo para A o MSB do endereço
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        pop DPH         ; Restauro contador da pilha
        pop DPL
        mov A,DPL
        push DPL        ; Salvo contador na pilha
        push DPH
        TRANSMITBYTECOUNT0      ; Inicializo o Registrador Contador
                                ; de Byte Transmitido 0 (TBCR0)
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        pop DPH         ; Restauro contador da pilha
        pop DPL
        mov A,DPH
        TRANSMITBYTECOUNT1      ; Inicializo o Registrador Contador
                                ; de Byte Transmitido 1 (TBCR1)
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        COMMAND         ; Inicializo o Comando dos Registradores (CR)
        mov A,#CR_TRANS ; Inicializo o Registrador no Page 0 para Transmissão
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        jmp ACABOU

QUEUE_IT nop
        setb Check
        lcall QUEUE_PACKET      ; Enfileira os pacotes
ACABOU  nop

;        mov TH1,#000H   ; Rotina de atraso
;        mov TL1,#00FH
;        acall DELAY     ; Chama rotina que manda dados para iniciar o display

        setb EA         ; Todas as Interrupções são habilitadas
        setb EX0        ; Habilita a Interrupção Externa 0

        ret
        ENDPROC

; ***************************************************************************
; * Descrição:                                                              *
; *     A Rotina PCtoNIC transfere o pacote da RAM do PC para a             *
; *     RAM local que está no NIC.                                          *
; * Entrada Requerida:                                                      *
; *     R6 = LSB do Endereço Inicial Remoto = 00h --> RASR0,1 = 2000H       *
; *     R7 = MSB do Endereço Inicial Remoto = 20h                           *
; *     R2 = LSB do Contador de Endereço = 86h --> RBCR0,1 = 0586H          *
; *     R3 = MSB do Contador de Endereço = 05h                              *
; *     R4 = LSB do Endereço para acessar o Porto I/O = 00h                 *
; *     R5 = MSB do Endereço para acessar o Porto I/O = E0h                 *
; *     R0 = LSB do Endereço da RAM do PC = 40h                             *
; *     R1 = MSB do Endereço da RAM do PC = 00h                             *
; * Na Saída:                                                               *
; *     Nada                                                                *
```

```
; * Afetado:                                                                *
; *      DPTR, R0, R1, R2, R3, R4, R5, R6, R7 (do banco de registrador 0)   *
; * Comentários:                                                            *
; *      Nenhum                                                            *
; ***************************************************************************


PCtoNIC PROC

        push DPL        ; Salvo LSB do contador
        push DPH        ; Salvo MSB do contador

        REMOTEBYTECOUNT0        ; Inicializo o Registrador Contador
                                ; de Byte Remoto 0 (RBCR0)
        mov A,#0Fh      ; Seto o Contador de bytes LSB do RBCR0 para não zero
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        REMOTEBYTECOUNT1        ; Inicializo o Registrador Contador
                                ; de Byte Remoto 1 (RBCR1)
        mov A,#0Fh      ; Seto o Contador de bytes MSB do RBCR1 para não zero
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        COMMAND         ; Inicializo o Registrador de Comando
        mov A,#0Ah      ; Para fazer uma simulação de leitura DMA Remota para
                        ; evitar problemas "dummy"
        movx @DPTR,A    ; Escrevo o acumulador no registrador para forçar PRQ

        pop DPH         ; Retorno com o contador MSB em DPH
        pop DPL         ; Retorno com o contador LSB em DPL

        mov A,DPL       ; Seto o Contador de bytes LSB do RBCR0
        push DPL        ; Armazena o contador de bytes
        push DPH
        REMOTEBYTECOUNT0        ; Inicializo o Registrador Contador
                                ; de Byte Remoto 0 (RBCR0)
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        pop DPH         ; Restaura o contador de bytes
        pop DPL

        mov A,DPH       ; Seto o Contador de bytes MSB do RBCR1
        push DPL        ; Armazena o contador de bytes
        push DPH
        REMOTEBYTECOUNT1        ; Inicializo o Registrador Contador
                                ; de Byte Remoto 1 (RBCR1)
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        mov A,R6        ; Seto o Registrador de Endereço Inicial Remoto
        REMOTESTARTADDRESS0     ; Inicializo o Registrador Endereço
                                ; Inicial Remoto 0 (RSAR0)
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        mov A,R7        ; Seto o Registrador de Endereço Inicial Remoto
        REMOTESTARTADDRESS1     ; Inicializo o Registrador Endereço
                                ; Inicial Remoto 1 (RSAR1)
        movx @DPTR,A    ; Envia o comando para os registradores NIC
```

```
        COMMAND          ; Inicializo o Comando dos Registradores (CR)
        mov A,#CR_ESCR   ; Inicializa e Escreve no Registrador no Page 0
                         ; para Transmissão
        movx @DPTR,A     ; Envia o comando para os registradores NIC

WRITING_BYTE:
        mov DPL,R2       ; Ponho em DPL o valor LSB do contador que estava R2
        mov DPH,R3       ; Ponho em DPH o valor MSB do contador que estava R3
        mov A,DPL
        jnz MANDA
        mov A,DPH
        jz NAO           ; Vejo se contador = 0
MANDA:
;PCRAM
        mov DPL,R1       ; Pego de volta o valor do LSB da RAM do PC
        mov DPH,R0       ; Pego de volta o valor do MSB da RAM do PC
        movx A,@DPTR     ; Escrevo no acumulador o pacote a ser transmitido
        inc DPTR         ; Próximo valor a ser lido
        mov R1,DPL       ; Salvo em R0 o valor do LSB da RAM do PC
        mov R0,DPH       ; Salvo em R1 o valor do MSB da RAM do PC
;NICRAM vindo do IOPORT
        mov DPTR,#IOPORT; Ponho ponteiro em E000h que e o endereço do porto
        movx @DPTR,A     ; Envia o comando para os registradores NIC
;CONTADOR
     mov DPL,R2       ; Ponho em DPL o valor LSB do contador que estava R2
     mov DPH,R3       ; Ponho em DPH o valor MSB do contador que estava R3
     acall UTIL_DPTRDEC
     mov R2,DPL       ; Salvo em R2 o valor LSB do contador que estava DPL
     mov R3,DPH       ; Salvo em R3 o valor MSB do contador que estava DPH
       sjmp WRITING_BYTE
NAO:
        mov R0,#byte_count       ; Valor do contador de byte para transmissão
        mov R1,#(byte_count+1)   ; Valor do contador de byte para transmissão
        mov @R0,DPH              ; Movo para byte_count o MSB do contador
        mov @R1,DPL              ; Movo para byte_count+1 o LSB do contador

        INTERRUPTSTATUS ; Inicializo o Registrador de Status de Interrupção
                        ; (ISR)
CHECKDMA
        movx A,@DPTR     ; Leio os registradores NIC para verificar estado
        anl A,#ISR_DMA_DONE      ; Verifica se DMA foi completado
        jnz TONICEND     ; Se sim, ir para TONICEND
        jmp CHECKDMA     ; Fica no loop até DMA ser completado
TONICEND
        INTERRUPTSTATUS ; Inicializo o Registrador de Status de Interrupção
                        ; (ISR)
        mov A,#ISR_DMA_DONE      ; Apaga o bit de interrupção em ISR
        movx @DPTR,A     ; Envia o comando para os registradores NIC

        pop DPH          ; Restauro MSB do contador
        pop DPL          ; Restauro LSB do contador

        ret

        ENDPROC
```

```
;*******************************************************************************
;  Descrição:                                                                   *
;       Decrementa DPTR.  CY == 1 Se houver Underflow                           *
;  Entrada Requerida:                                                           *
;       DPTR Tem Valor … ser Decrementado                                       *
;  Na Saída:                                                                    *
;       DPTR = DPTR - 1, CY , Setado de acordo                                  *
;  Afetado:                                                                     *
;       PSW.CY, DPTR                                                            *
;  Pilha:                                                                       *
;       1 Byte, Não Incluindo o Espaço usado pela Rotina Chamada                *
;  Comentários:                                                                 *
;       Nenhum                                                                  *
;*******************************************************************************

UTIL_DPTRDEC      proc
                  push    Acc                   ; Save Acc
                  clr     C                     ; Apago para SUBB
                  mov     A,DPL                 ; Move LSB de DPTR para A
                  subb    A,#1                  ; Subtrai 1
                  mov     DPL,A                 ; Restaura de volta
                  mov     A,DPH                 ; Pega MSB de DPTR
                  subb    A,#0                  ; Subtrai CY se Setado
                  mov     DPH,A                 ; Move de volta
                  pop     Acc                   ; Rearmazena Acc
                  ret                           ; Retorna para Chamador


        ENDPROC

; *******************************************************************************
; * Descrição:                                                                  *
; *     Rotina que espera um determinado tempo para provocar colisão            *
; * Entrada Requerida:                                                          *
; *     R1 , o tempo de atraso do Display                                       *
; * Na saída:                                                                   *
; *     Nada                                                                    *
; * Afetado:                                                                    *
; *     R1                                                                      *
; * Comentários:                                                                *
; *     Nenhum                                                                  *
; *******************************************************************************

DELAY   PROC
        mov R1,#01H     ; R1 = 1 para tempo de atraso do display
REP1:   setb TR1
        jnb TF1,$
        clr TF1
        djnz R1,REP1
        clr TR1

        ret

        ENDPROC

        END
```

```
; ************************************************************************
; * Descrição:                                                           *
; *     A Rotina NICtoPC transfere o pacote RAM local no NIC para a      *
; *     RAM que está no PC.                                              *
; * Entrada Requerida:                                                   *
; *     DPTR = contador de byte do pacote = RECV_PCK+EN_RBUF_SIZE        *
; *     R6 = LSB do Endereço Inicial Remoto = 00h                        *
; *     R7 = MSB do Endereço Inicial Remoto = RECV_PCK+RBUF_NXT_PG       *
; * Na Saída:                                                            *
; *     Nada                                                             *
; * Afetado:                                                             *
; *     DPTR, R0 e R1 (do banco de registrador 0)                        *
; * Comentários:                                                         *
; *     Nenhum                                                           *
; ************************************************************************


        PUBLIC NICtoPC
        EXTERN UTIL_DPTRDEC

; Pseudo Variáveis na ROM
CS      equ 0A000H      ; Endereço de memória de acesso ao NIC
IOPORT  equ 0E000H      ; Endereço de memória para acessar os dados no
                        ; barramento local


        %include "8390.inc"

; Comando do Chip no Registrador de Comando (CR)
CR_LE           EQU     01Ah      ; Lê o registrador

; Bits do Registrador de Estado de Interrupção (ISR)
ISR_DMA_DONE    EQU     040h      ; Verificar se o DMA está completo

; Descrição do header de cada pacote na  rea de recepção da memória
EN_RBUF_NHDR    equ     4         ; Tamanho da área do header

; Endereço da memória para enfileirar o pacote na fila da memória do
; microprocessador
RERAM   DATA 24H        ; Endereço da memória do microprocessador (recepção)


NICtoPC PROC

        inc DPL
        mov A,DPL       ; Seto o Contador de bytes LSB do RBCR0
        push DPL        ; Armazena o contador de bytes
        push DPH
        REMOTEBYTECOUNT0        ; Inicializo o Registrador Contador
                                ; de Byte Remoto 0 (RBCR0)
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        pop DPH         ; Restaura o contador de bytes
        pop DPL

        mov A,DPH       ; Seto o Contador de bytes MSB do RBCR1
```

```
        push DPL        ; Armazena o contador de bytes
        push DPH
        REMOTEBYTECOUNT1        ; Inicializo o Registrador Contador
                                ; de Byte Remoto 0 (RBCR1)
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        mov A,R6        ; Seto o Contador de bytes LSB do RSAR 0
        REMOTESTARTADDRESS0     ; Inicializo o Registrador Endereço
                                ; Inicial Remoto 0 (RSAR0)
        movx @DPTR,A    ; Envia o comando para os registradores NIC


        mov A,R7        ; Seto o Contador de bytes MSB do RSAR 1
        REMOTESTARTADDRESS1     ; Inicializo o Registrador Endereço
                                ; Inicial Remoto 1 (RSAR1)
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        mov A,#CR_LE    ; Inicializa e Lê no Registrador no Page 0
                        ; para Transmissão
        COMMAND         ; Inicializo o Comando dos Registradores (CR)
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        mov R0,#RERAM   ; RERAM aponta para o pacote que ser  recebido(NIC)
        mov R1,#RERAM+1 ; primeiro pego o MSB e depois pego o LSB
        mov A,@R1
        clr C           ; Zero o carry para fazer subtração
        add A,#EN_RBUF_NHDR
        mov DPL,A               ; Movo para DPL o LSB do endereço
        mov DPH,@R0             ; Movo para DPH o MSB do endereço
        mov R1,DPL      ; Salvo em R0 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R0,DPH      ; Salvo em R1 o valor do MSB do Endereço Inicial

        mov DPTR,#IOPORT; Endereço para acesso ao porto de Entrada/Saída
        mov R2,DPL      ; Salvo em R2 o valor do LSB do Endereço para iniciar
                        ; o IOPORT
        mov R3,DPH      ; Salvo em R3 o valor do MSB do Endereço para iniciar
                        ; o IOPORT
        mov R4,#00h
READING_BYTE:
        pop DPH         ; Restauro MSB do contador
        pop DPL         ; Restauro LSB do contador
        push DPL        ; Salvo LSB do contador
        push DPH        ; Salvo MSB do contador
        mov A,DPL
        jnz MANDA1
        mov A,DPH
        jz NAO1         ; Vejo se contador = 0
MANDA1:
;NICRAM
        mov DPL,R2      ; Pego de volta o valor do LSB do Endereço do IOPORT
        mov DPH,R3      ; Pego de volta o valor do MSB do Endereço do IOPORT
        movx A,@DPTR    ; Escrevo no acumulador o pacote que foi recebido
        cjeq R4,#00h,L?1        ; se R4=00h, saio
;PCRAM
        mov DPL,R1      ; Pego de volta o valor do LSB da RAM do PC
        mov DPH,R0      ; Pego de volta o valor do MSB da RAM do PC
```

```
        movx @DPTR,A     ; Escrevo na RAM do PC o pacote que foi recebido
        inc DPTR         ; Próximo valor a ser lido
        mov R1,DPL       ; Salvo em R1 o valor do LSB da RAM do PC
        mov R0,DPH       ; Salvo em R0 o valor do MSB da RAM do PC
        sjmp L?2         ; Decremento contador
;CONTADOR
L?1:    inc R4
L?2:    pop DPH          ; Restauro MSB do contador
        pop DPL          ; Restauro LSB do contador
        lcall UTIL_DPTRDEC
        push DPL         ; Salvo LSB do contador
        push DPH         ; Salvo MSB do contador
        sjmp READING_BYTE
NAO1:
        INTERRUPTSTATUS ; Inicializo o Registrador de Status de
                        ; Interrupção (ISR)
CHECKDMA
        movx A,@DPTR     ; Vou ler no NIC o comando dos registrador ISR
        anl A,#ISR_DMA_DONE     ; Verifica se DMA foi completado
        jnz READEND •    ; Se sim, ir para READEND
        jmp CHECKDMA     ; Fica no loop até DMA ser completado
READEND
        movx @DPTR,A     ; Escrevo no NIC o comando
        pop DPH          ; Restauro MSB do contador
        pop DPL          ; Restauro LSB do contador

        ret

        ENDPROC
        END
```

```
; ********************************************************************************
; * Descrição:                                                                   *
; *     Mensagem de Transmissão (TX_MEN) , uma mensagem que avisa no             *
; *     display se a transmissão foi bem sucedida e se ocorreu erro na           *
; *     transmissão. Se ocorreu erro na transmissão a mensagem ser  lida         *
; *     pelo (ER_MEN).                                                           *
; * Entrada Requerida:                                                           *
; *     DPTR Contém o Endereço para inicializar o CS da NIC                      *
; * Na Saída:                                                                    *
; *     Mensagem no Display                                                      *
; * Afetado:                                                                     *
; *     ex_tx (20H.1) e 0050H até 0085H (mensagem a ser escrita)                 *
; *     R1' e R2' (do banco de registrador 1)                                    *
; * Comentários:                                                                 *
; *     Nenhum                                                                   *
; ********************************************************************************



        PUBLIC TX_MEN
        PUBLIC ER_MEN
        PUBLIC INICD

        %include "8390.inc"

; Pseudo Variáveis na ROM
Dispa   equ 8000H       ; Endereço da memória de acesso ao display
RSR_JABBER equ  80h     ; Nos diz se houve uma recepção no momento em que
                        ; estávamos transmitindo

; Bits Endereçáveis dos Registradores Especiais (SFR's)
RS      BIT P1.0        ; Endereço do pino P1.0, usado como RS (Endereço)
RW      BIT P1.1        ; Endereço do pino P1.1, usado como RW (Read/Write)
ex_tx   BIT 20H.1       ; Endereço de memória que nos diz se houve erro na
                        ; transmissão
jabber  BIT 20H.4       ; Endereço de memória que nos diz se existe algum
                        ; problema elétrico no nosso sistema (jabber)


TX_MEN  PROC
        push DPL        ; Armazena o contador de bytes
        push DPH
        push Acc        ; Salvo acumulador na pilha

        jnb ex_tx,E?1
        sjmp ER_MEN


E?1:
        setb RS0        ; Banco de registrador 1
        lcall INICD     ; Inicializa o Display
        mov DPTR,#0050H ; Local onde a mensagem foi escrita.
E?2:    clr A           ; Apago Acc para ler tabela
        movc A,@A+DPTR  ; Pego o primeiro caracter do endereço
        cjne A,#0DH,LE?1; Vê se não , o último caracter
        sjmp NEXT1      ; sai da rotina
LE?1:   mov TH0,#0F7H   ; Timer 0 com tempo de atraso maior que 1.5 ms
```

```
        mov TL0,#00H
        acall ATRASO    ; Chama rotina que mostra no display
        inc DPTR        ; Incremento DPTR para o próximo valor da tabela
        sjmp E?2        ; Leio próximo caracter
NEXT1:
        pop Acc         ; Restaura acumulador na pilha
        pop DPH         ; Restaura o contador de bytes
        pop DPL
        clr RS0         ; Retorno ao banco de registrador 0

        ret
        ENDPROC


ER_MEN  PROC

        setb RS0        ; Banco de registrador 1
        lcall INICD     ; Inicializa o Display
        mov DPTR,#006BH ; Local onde a mensagem foi escrita.
E?3:    clr A           ; Apago Acc para ler tabela
        movc A,@A+DPTR  ; Pego o primeiro caracter do endereço
        cjne A,#0DH,LE?2; Vê se não é o último caracter
        sjmp NEXT2      ; Sai
LE?2:   mov TH0,#0F7H   ; Timer 0 com tempo de atraso maior que 1.5 ms
        mov TL0,#00H
        acall ATRASO    ; Chama rotina que mostra no display
        inc DPTR        ; Incremento DPTR para o próximo valor da tabela
        sjmp E?3        ; Leio próximo caracter
NEXT2:  clr ex_tx

        jnb jabber,PRO

        pop Acc
        push Acc        ; Salvo acumulador na pilha
        anl A,#RSR_JABBER       ;vejo se o bit DFR está setado
        jnz DFR         ; escrever mensagem de DEFERRING (JABBER)

PRO:    pop Acc
        push Acc        ; Salvo acumulador na pilha
        anl A,#20h      ; vejo se o bit FU esta setado
        jnz FU          ; escrever mensagem de FIFO UNDERRUN

        pop Acc
        push Acc        ; Salvo acumulador na pilha
        anl A,#10h      ; vejo se o bit CRS esta setado
        jnz CRS         ; escrever mensagem de CARRIER SENSE LOST

        pop Acc
        push Acc        ; Salvo acumulador na pilha
        anl A,#08h      ; vejo se o bit ABT esta setado
        jnz ABT         ; escrever mensagem de TRANSMIT ABORTED

FU:     setb RS0        ; Banco de registrador 1
        mov DPTR,#0110H ; Local onde a mensagem foi escrita.
E?4:    clr A           ; Apago Acc para ler tabela
        movc A,@A+DPTR  ; Pego o primeiro caracter do endereço
        cjne A,#0DH,LE?3; Vê se não , o último caracter
```

```
        sjmp NEXT3      ; sai da rotina
LE?3:   mov TH0,#0F7H   ; Timer 0 com tempo de atraso maior que 1.5 ms
        mov TL0,#00H
        acall ATRASO    ; Chama rotina que mostra no display
        inc DPTR        ; Incremento DPTR para o próximo valor da tabela
        sjmp E?4        ; Leio próximo caracter

CRS:    setb RS0        ; Banco de registrador 1
        mov DPTR,#0140H ; Local onde a mensagem foi escrita.
E?5:    clr A           ; Apago Acc para ler tabela
        movc A,@A+DPTR  ; Pego o primeiro caracter do endereço
        cjne A,#0DH,LE?4; Vê se não , o último caracter
        sjmp NEXT3      ; sai da rotina
LE?4:   mov TH0,#0F7H   ; Timer 0 com tempo de atraso maior que 1.5 ms
        mov TL0,#00H
        acall ATRASO    ; Chama rotina que mostra no display
        inc DPTR        ; Incremento DPTR para o próximo valor da tabela
        sjmp E?5        ; Leio próximo caracter

ABT:    setb RS0        ; Banco de registrador 1
        mov DPTR,#0170H ; Local onde a mensagem foi escrita.
E?6:    clr A           ; Apago Acc para ler tabela
        movc A,@A+DPTR  ; Pego o primeiro caracter do endereço
        cjne A,#0DH,LE?5; Vê se não , o último caracter
        sjmp NEXT3      ; sai da rotina
LE?5:   mov TH0,#0F7H   ; Timer 0 com tempo de atraso maior que 1.5 ms
        mov TL0,#00H
        acall ATRASO    ; Chama rotina que mostra no display
        inc DPTR        ; Incremento DPTR para o próximo valor da tabela
        sjmp E?6        ; Leio próximo caracter

DFR:    setb RS0        ; Banco de registrador 1
        mov DPTR,#01A0H ; Local onde a mensagem foi escrita.
E?7:    clr A           ; Apago Acc para ler tabela
        movc A,@A+DPTR  ; Pego o primeiro caracter do endereço
        cjne A,#0DH,LE?6; Vê se não , o último caracter
        sjmp NEXT3      ; sai da rotina
LE?6:   mov TH0,#0F7H   ; Timer 0 com tempo de atraso maior que 1.5 ms
        mov TL0,#00H
        acall ATRASO    ; Chama rotina que mostra no display
        inc DPTR        ; Incremento DPTR para o próximo valor da tabela
        sjmp E?7        ; Leio próximo caracter

NEXT3:
        pop Acc         ; Restaura acumulador na pilha
        pop DPH         ; Restaura o contador de bytes
        pop DPL
        clr RS0         ; Retorno ao banco de registrador 0

        ret

        ENDPROC

; ********************************************************************************
; * Descrição:                                                                   *
; *     Rotina de inicialização do display configurando para que o cursor        *
```

```
; *     fique na primeira linha                                              *
; * Entrada Requerida:                                                       *
; *     R0' contém caracter de controle do Display                           *
; * Na saída:                                                                *
; *     Nada                                                                 *
; * Afetado:                                                                 *
; *     R0' do banco de registrador 1                                        *
; * Comentários:                                                             *
; *     Nenhum                                                               *
; *************************************************************************

INICD   PROC

        mov TH0,#0F9H   ; Rotina de atraso de 1.7 ms
        mov TL0,#5BH
        lcall DELAY     ; Chama rotina de atraso
        clr RW
        clr RS          ; RS = 0 para instrução de comandos

        mov A,#38H      ; Interface de 8 bits, 2linhas e matriz 5x7 pontos
        mov TH0,#0FFH   ; Rotina de atraso de 40 us.
        mov TL0,#0C3H
        lcall ATRASO    ; Chama rotina que manda dados para iniciar o display

        mov A,#08H      ; Interface de 8 bits, 2linhas e matriz 5x7 pontos
        mov TH0,#0FFH   ; Rotina de atraso de 40 us.
        mov TL0,#0C3H
        lcall ATRASO    ; Chama rotina que manda dados para iniciar o display

        MOV A,#01H      ; Limpa todo o display
        mov TH0,#0F9H   ; Rotina de atraso de 1.7 ms
        mov TL0,#5BH
        lcall ATRASO    ; Chama rotina que manda dados para iniciar o display

        MOV A,#02H      ; Primeira posição da esquerda
        mov TH0,#0F9H   ; Rotina de atraso de 1.7 ms
        mov TL0,#5BH
        lcall ATRASO    ; Chama rotina que manda dados para iniciar o display

        MOV A,#06H      ; Comando para colocar mensagem … direita no display
        mov TH0,#0FFH   ; Rotina de atraso de 40 us
        mov TL0,#0C3H
        lcall ATRASO    ; Chama rotina que manda dados para iniciar o display

        MOV A,#0EH      ; Mensagem aparente, cursor ativo e não piscando
        mov TH0,#0FFH   ; Rotina de atraso de 40 us
        mov TL0,#0C3H
        lcall ATRASO    ; Chama rotina que manda dados para iniciar o display

        setb RS         ; RS = 1 para leitura de comandos

        ret

        ENDPROC

; *************************************************************************
; * Descrição:                                                            *
```

```
; *     Rotina que escreve e manda o caracter na posição atual do display   *
; * Entrada Requerida:                                                      *
; *     Acc contém caracter a ser transmitido no Display                    *
; *     R1 , o tempo de atraso do Display                                   *
; * Na saída:                                                               *
; *     Nada                                                                *
; * Afetado:                                                                *
; *     R0' e R1' (do banco de registrador 1)                               *
; * Comentários:                                                            *
; *     Nenhum                                                              *
; *************************************************************************

ATRASO  PROC

        push DPL        ; Armazena o contador de bytes
        push DPH

        mov DPTR,#Dispa
        movx @DPTR,A

        pop DPH         ; Restaura o contador de bytes
        pop DPL

DELAY:
        mov R1,#01H     ; R1 = 1 para tempo de atraso do display
REP1:   setb TR0
        jnb TF0,$
        clr TF0
        djnz R1,REP1
        clr TR0

        ret

        ENDPROC

        END
```

```
; ********************************************************************************
; * Descrição:                                                                   *
; *     DriverIsr , uma rotina de serviço de interrupção que responde:           *
; *     para transmissão, erro na transmissão e interrupção na recepção          *
; *     (os bits  PTX, TXE e PRX no registrador de Status de Interrupção)        *
; *     produzidas na NIC.                                                       *
; * Entrada Requerida:                                                           *
; *     Pacote a ser analizado pelo software.                                    *
; * Na Saída:                                                                    *
; *     O software informa se houve sucesso ou insucesso na transmissão          *
; *     do pacote                                                                *
; * Afetado:                                                                     *
; *     R0, R1, R2, R3, R4, R6 e R7 (do banco de registrador 0)                  *
; *     R5 e R6 (do banco de registrador 1)                                      *
; * Comentários:                                                                 *
; *     Na Interrupção de transmissão, o software das camadas superiores ,       *
; *     informado se a transmissão foi um sucesso ou se ocorreu erros; Na        *
; *     Interrupção da recepção os pacotes são removidos do Receptor do          *
; *     Buffer Ring (na memória local) e transferidos para o PC.                 *
; ********************************************************************************


; Pseudo Variáveis na ROM
IOPORT  equ 0E000H      ; Endereço de memória para acessar os dados no
                        ; barramento local
CS              equ 0A000H      ; Endereço de memória de acesso ao NIC

; Declaração das Rotinas
        PUBLIC DRIVERISR
        PUBLIC CONTADOR
        EXTERN DRIVERSEND
        EXTERN NICtoPC
        EXTERN PING
        EXTERN TX_MEN
        EXTERN ER_MEN
        EXTERN UTIL_DPTRDEC
        EXTERN QUEUE_PACKET

; Bits Endereçáveis dos Registradores Especiais (SFR's)
ex_tx   BIT 20H.1       ; Endereço de memória que nos diz se houve erro na
                        ; transmissão
Check   BIT 20H.2       ; Endereço de memória que nos diz que pacote foi
                        ; enfileirado
first_reception BIT 20H.3       ; Endereço de memória que nos diz se e a
                                ; primeira recepção
jabber  BIT 20H.4       ; Endereço de memória que nos diz se existe algum
                        ; problema elétrico no nosso sistema (jabber)

        %include "8390.inc"


; Comando do Chip no Registrador de Comando (CR)
CR_STOP         EQU     001h    ; Chip no modo Stop
CR_START        EQU     002h    ; Chip no modo Start
CR_LE           EQU     01Ah    ; Lê o registrador
CR_NODMA        EQU     020h    ; Não , usado DMA remoto nesta interface
```

```
CR_PAGE0        EQU     021h    ; Seleciona page 0 nos registradores do chip
CR_PAGE1        EQU     061h    ; Seleciona page 1 nos registradores do chip
CR_PAGE2        EQU     0A2h    ; Seleciona page 2 nos registradores do chip

; Comando para o registrador de controle de TX (TCR)
TCR_RESET       EQU     00h     ; Reseto inicialmente o transmissor
TCR_LOOP        EQU     002h    ; Loopback no modo 1, com loop interno

; Bits do Registrador de Estado de Interrupção (ISR)
ISR_RX          EQU     001h    ; Recepção sem erro
ISR_TX          EQU     00Ah    ; Transmissão sem erro
ISR_OVER        EQU     010h    ; Recepção com overflow no anel
ISR_RESET       EQU     080h    ; Verifico se o bit RST está setado

; Bits no Registrador de Estado de RX (RSR)
RSR_JABBER equ  80h     ; Nos diz se houve um problema de ordem elétrico.
                        ; Como o Ethernet observa se seus dados foram enviados
                        ; corretamente através de níveis de tensões, qualquer
                        ; problema elétrico pode fazer com que todos os outros
                        ; terminais achem que o canal esta ocupado e acabam
                        ; não transmitindo.

; Bits no Registrador de Estado de TX (TSR)

TSR_VERIF       EQU     038h    ; Vejo se o NIC conseguiu ganhar um acesso
                                ; ao barramento antes da FIFO esvaziar (FU)
                                ; Vejo se a portadora foi perdida durante a
                                ; transmissão do pacote (CRS)
                                ; Vejo se a transmissão foi abortada por
                                ; colisão excessiva (ABT)

; Bits do Registrador de Máscara de Interrupção (IMR)
IMR_OVWE        EQU     01Bh    ; Habilita interrupção quando o gerenciamento
                                ; lógico do buffer necessitar de memória para
                                ; alocar o pacote.

; Descrição do header de cada pacote na área de recepção da memória

EN_RBUF_NXT_PG    equ   1     ; Página depois deste frame
EN_RBUF_SIZE_LO   equ   2     ; Tamanho deste frame (LSB)
EN_RBUF_SIZE_HI   equ   3     ; Tamanho deste frame (MSB)
EN_RBUF_NHDR    equ     4     ;  Tamanho da área do header


; Endereço da memória para enfileirar o pacote na fila da memória do
; microprocessador
RERAM   DATA 24H        ; Endereço da memória do microprocessador (recepção)
PCRAM   DATA 26H        ; Endereço da memória do microprocessador
tailptr DATA    28H     ; Valor do próximo buffer a ser lido
headptr DATA    2AH     ; Valor do próximo espaço disponível no buffer
byte_count      DATA 2CH; Valor do contador de byte para transmissão
next_packet     DATA 2EH; Valor do próximo buffer a ser lido


DRIVERISR       PROC
        clr EA          ; Todas as Interrupções são desabilitadas
        push PSW        ; Salvo os flags
```

```
        push Acc        ; Salvo acumulador na pilha
        push DPL        ; Armazena o contador de bytes
        push DPH
        push 00         ; Armazena o Registrador R0
        push 01         ; Armazena o Registrador R1
        push 02         ; Armazena o Registrador R2
        push 03         ; Armazena o Registrador R3
        push 04         ; Armazena o Registrador R4
        push 05         ; Armazena o Registrador R5
        push 06         ; Armazena o Registrador R6
        push 07         ; Armazena o Registrador R7
        setb EA         ; Todas as Interrupções são habilitadas
        setb EX0        ; Habilita a Interrupção Externa 0

; ************************************************************************
; * Leio o Registrador de Status de Interrupção para pacotes recebidos,  *
; * pacotes transmitidos e pacotes transmitidos errados                  *
; ************************************************************************

POLL
        INTERRUPTSTATUS ; Inicializo o Registrador de Status de
                        ; Interrupção (ISR)
        movx A,@DPTR    ; Vou ler no NIC o registrador que d  o estado da
                        ; Interrupção (ISR)
        mov R0,A        ; Salvo o estado da interrupção em R0
        anl A,#ISR_RX   ; o pacote foi recebido ?
        jnz PKT_RECV_RT ; se estiver ir para PKT_RECV_RT
        mov A,R0        ; pego de volta o valor da interrupção
        anl A,#ISR_TX   ; o pacote foi transmitido ?
        jz EXIT_ISR     ; se não existir interrupção ir para EXIT_ISR
        ljmp PKT_TX_RT  ; se estiver transmitindo ir para PKT_TX_RT

EXIT_ISR
        INTERRUPTMASK   ; Inicializo o Registrador de Máscara de
                        ; Interrupção (IMR)
        clr A           ; Desabilito as interrupções da NIC
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        INTERRUPTMASK   ; Inicializo o Registrador de Máscara de
                        ; Interrupção (IMR)
        mov A,#IMR_OVWE ; Desabilito as interrupções da NIC
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        pop 07          ; Restaura o registrador R7
        pop 06          ; Restaura o registrador R6
        pop 05          ; Restaura o registrador R5
        pop 04          ; Restaura o registrador R4
        pop 03          ; Restaura o registrador R3
        pop 02          ; Restaura o registrador R2
        pop 01          ; Restaura o registrador R1
        pop 00          ; Restaura o registrador R0
        pop DPH         ; Restaura o contador de bytes
        pop DPL
        pop Acc         ; Restaura acumulador na pilha
        pop PSW         ; Restaura os Flags
        reti
        ENDPROC
```

```
; ************************************************************************
; * Rotina de pacote recebido (pkt_recv_rt) - joga todos os bons pacotes  *
; * no receptor do buffer ring local. Pacotes ruins são ignorados        *
; ************************************************************************

PKT_RECV_RT
        INTERRUPTSTATUS ; Inicializo o Registrador de Status de
                        ; Interrupção (ISR)
        movx A,@DPTR    ; Vou ler no NIC o valor do Registrador de Estado de
                        ; Interrupção (ISR)
        anl A,#ISR_OVER ; Testo para ver se há overflow no anel
        jz PASSE        ; Se zero continue...
        jmp RING_OVFL   ; se sim, retirar pacotes
PASSE:  mov A,#ISR_RX   ; resetar bit PRX em ISR
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        mov R0,#next_packet     ; Valor onde o próximo pacote será lido (MSB)
        mov R1,#(next_packet+1) ; Valor onde o próximo pacote será lido (LSB)
        mov DPH,@R0             ; Movo para next_packet o MSB do pacote
        mov DPL,@R1             ; Movo para next_packet+1 o LSB do pacote
        mov R7,DPH       ; Ponho no ponteiro MSB do Endereço Remoto
        mov R6,DPL       ; Zero o ponteiro LSB do Endereço Remoto

        mov DPTR,#EN_RBUF_NHDR  ; Ponho tamanho do header que será lido

        lcall CONTADOR

        mov A,R1
        clr C           ; Zero o carry para fazer subtração
        subb A,#EN_RBUF_NHDR
        mov R1,A
        mov DPL,R1      ; Jogo tamanho do pacote no contador inferior
        mov DPH,R4      ; Jogo tamanho do pacote no contador superior
        mov A,R4
        mov byte_count,A        ; Valor MSB do contador de byte para
                                ; transmissão
        mov A,R1
        mov (byte_count+1),A    ; Valor LSB do contador de byte para
                                ; transmissão
        push 01
        push 04

        mov R0,#next_packet     ; Valor onde o próximo pacote será lido (MSB)
        mov R1,#(next_packet+1) ; Valor onde o próximo pacote será lido (LSB)
        mov A,@R1
        clr C           ; Zero o carry para fazer subtração
        add A,#EN_RBUF_NHDR
        mov R6,A        ; Zero o ponteiro LSB do Endereco Remoto
        mov A,@R0
        mov R7,A        ; Ponho no ponteiro MSB do Endereço Remoto

        lcall NICtoPC
;        setb first_reception    ; seto a variável first_reception
        pop 04
        pop 01
```

```
        lcall PING
; ************************************************************************
; * Informa o Software das camadas superiores do pacote recebido a ser   *
; * processado. Checa para ver se o Buffer Ring está vazio               *
; ************************************************************************

CHECK_RING
        BOUNDARY        ; Inicializo o Ponteiro Limite (BNDRY)
        movx A,@DPTR    ; Vou ler nos registradores NIC o valor de BOUNDARY

        mov R3,A        ; Salvo BOUNDARY em R3
        COMMAND         ; Inicializo o Comando dos Registradores (CR)
        mov A,#CR_PAGE1+CR_STOP ; Mudo para page 1 na NIC
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        CURRENT         ; Inicializo o Registrador de Page Corrente (CURR).
        movx A,@DPTR    ; Vou ler nos registradores NIC o valor de CURRENT

        mov R4,A        ; Salvo CURRENT em R4
        COMMAND         ; Inicializo o Comando dos Registradores (CR)
        mov A,#CR_NODMA+CR_START; Volto o Registrador no Page 0, no modo Start
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        mov A,R3
        cjne A,04H,PKT  ; Se não for igual, ir para PKT
        jmp POLL
PKT:    jmp PKT_RECV_RT ; Receptor do buffer ring está vazia?
; ************************************************************************
; * Ring Overflow , uma rotina que checa paraáver se existe uma sobrecarga *
; * no Buffer Ring                                                       *
; ************************************************************************


RING_OVFL
        COMMAND         ; Inicializo o Comando dos Registradores (CR)
        mov A,#CR_PAGE0 ; Inicializo o Registrador no Page 0, no modo STOP
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        REMOTEBYTECOUNT0        ; Inicializo o Registrador Contador
                                ; de Byte Remoto 0 (RBCR0)
        clr A           ; Apago o valor de RBCR0
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        REMOTEBYTECOUNT1        ; Inicializo o Registrador Contador
                                ; de Byte Remoto 1 (RBCR1)
        clr A           ; Apago o valor de RBCR1
        movx @DPTR,A    ; Envia o comando para os registradores NIC


        mov DPTR,#7FFFH ; Carrego o contador time out
WAIT_FOR_STOP
        setb RS0        ; Mudo para banco de registradores 1;
        mov R5,DPL      ; Salvo LSB do contador
        mov R6,DPH      ; Salvo MSB do contador
        INTERRUPTSTATUS ; Inicializo o Registrador de Status de
                        ; Interrupção (ISR)
        movx A,@DPTR    ; Vou ler no NIC o Registrador de Status de
```

```
                ; Interrupção (ISR)
anl A,#ISR_RESET; observo se o bit RST está setado
mov DPL,R5      ; Pego de volta o LSB do contador
mov DPH,R6      ; Pego de volta o MSB do contador
lcall UTIL_DPTRDEC
jnc WAIT_FOR_STOP

clr RS0         ; Volto para banco de registradores 0;
TRANSMITCONFIGURATION   ; Inicializo o Registrador de
                        ; Configuração de Transmissão (TCR)
mov A,#TCR_LOOP ; Loopback no modo 1, com loop externo
movx @DPTR,A    ; Envia o comando para os registradores NIC

COMMAND         ; Inicializo o Comando dos Registradores (CR)
mov A,#CR_PAGE0+CR_STOP
movx @DPTR,A    ; Envia o comando para os registradores NIC

mov R0,#next_packet     ; Valor onde o próximo pacote será lido (MSB)
mov R1,#(next_packet+1) ; Valor onde o próximo pacote será lido (LSB)
mov DPH,@R0             ; Movo para next_packet o MSB do pacote
mov DPL,@R1             ; Movo para next_packet+1 o LSB do pacote
mov R7,DPH       ; Ponho no ponteiro MSB do Endereço Remoto
mov R6,DPL       ; Zero o ponteiro LSB do Endereço Remoto

mov DPTR,#EN_RBUF_NHDR  ; Ponho tamanho do header que será lido

lcall CONTADOR

mov A,R1
clr C           ; Zero o carry para fazer subtração
subb A,#EN_RBUF_NHDR
mov R1,A
mov DPL,R1      ; Jogo tamanho do pacote no contador inferior
mov DPH,R4      ; Jogo tamanho do pacote no contador superior
mov A,R4
mov byte_count,A        ; Valor MSB do contador de byte para
                        ; transmissão
mov A,R1
mov (byte_count+1),A    ; Valor LSB do contador de byte para
                        ; transmissão
push 01
push 04

mov R0,#next_packet     ; Valor onde o próximo pacote será lido (MSB)
mov R1,#(next_packet+1) ; Valor onde o próximo pacote será lido (LSB)
mov A,@R1
clr C           ; Zero o carry para fazer subtração
add A,#EN_RBUF_NHDR
mov R6,A        ; Zero o ponteiro LSB do Endereço Remoto
mov A,@R0
mov R7,A        ; Ponho no ponteiro MSB do Endereço Remoto

lcall NICtoPC
;        setb first_reception    ; seto a variavel first_reception
pop 04
pop 01
lcall PING
```

```
        INTERRUPTSTATUS ; Inicializo o Registrador de Status de
                        ; Interrupção (ISR)
        mov A,#ISR_OVER ; É apagado o bit de Overflow (OVW) do ISR
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        TRANSMITCONFIGURATION   ; Inicializo o Registrador de
                                ; Configuração de Transmissão (TCR)
        mov A,#TCR_RESET; Loopback no modo 0, ou seja, loopback em operação
                        ; normal
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        jmp CHECK_RING

; ************************************************************************
; * Rotina de transmissão do pacote (pkt_tx_rt). Ela determina o Status do *
; * pacote transmitido, depois checa a transmissão pendente da fila no    *
; * próximo pacote avaliado para transmitir.                              *
; ************************************************************************
                    •
PKT_TX_RT
        INTERRUPTSTATUS ; Inicializo o Registrador de Status de
                        ; Interrupção (ISR)
        mov A,#ISR_TX   ; Reseta os bits PTX e TXE no ISR
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        RECEIVESTATUS   ; Estado da Recepção (0Ch)
        movx A,@DPTR    ; Vou ler no NIC o Registrador de Status Recebido
        push Acc        ; Salvo o valor do Acumulador
        anl A,#RSR_JABBER       ;vejo se o bit DFR esta setado
        jnz BAD_TX1     ; se  estiverem ir para BAD_TX
        pop Acc         ; Retorno o antigo valor do acumulador

        TRANSMITSTATUS  ; Inicializo o Registrador de Status
                        ; Transmitido (TSR)
        movx A,@DPTR    ; Vou ler no NIC o Registrador de Status Transmitido
        push Acc        ; Salvo o valor do Acumulador
        anl A,#TSR_VERIF;vejo se os bits FU, CRS ou ABT estão setados
        jnz BAD_TX      ; se  estiverem ir para BAD_TX

; ************************************************************************
; * Informar o softwate das camadas superiores do sucesso da transmissão  *
; ************************************************************************
        pop Acc         ; Retorno o antigo valor do acumulador
        lcall TX_MEN
        sjmp CHK_TX_QUEUE
BAD_TX1 setb jabber
        pop Acc         ; Retorno o antigo valor do acumulador
        sjmp BAD_TX2
BAD_TX  pop Acc         ; Retorno o antigo valor do acumulador
BAD_TX2 setb ex_tx      ; transmitir mensagem se a transmissão foi errada
        lcall TX_MEN

; ************************************************************************
; * Informar o softwate das camadas superiores que houve erro na transmissão *
; ************************************************************************
CHK_TX_QUEUE    nop
```

```
        lcall QUEUE_PACKET
        mov A,DPL
        jnz MANDA
        mov A,DPH
        jz SAI                  ; Se não houver pacote na fila, ir para POLL
MANDA   lcall DRIVERSEND        ; Se houver transmita-o
SAI     ljmp POLL


; ***************************************************************************
; * Descrição:                                                              *
; *     CONTADOR , uma pequena rotina que pegara o header do pacote         *
; *     recebido. A intenção , pegar os 4 primeiros bytes que formam        *
; *     o header para a recepção                                            *
; * Entrada Requerida:                                                      *
; *     Pacote a ser analisado pelo software.                               *
; * Na Saída:                                                               *
; *     O software informa next_packet = próximo localização do pacote      *
; *     recebido                                                            *
; * Afetado:                                                                *
; *     R0, R1, R2, R3, R4, R6 e R7 (do banco de registrador 0)             *
; *     R5 e R6 (do banco de registrador 1)                                 *
; * Comentários:                                                            *
; *     Nenhum                                                              *
; ***************************************************************************

CONTADOR PROC

        push DPL        ; Armazena o contador de bytes
        push DPH
;        jnb first_reception,CT   ;Se for a primeira recepção, sair
;        inc DPL
CT:     mov A,DPL       ; Seto o Contador de bytes LSB do RBCR0
        REMOTEBYTECOUNT0        ; Inicializo o Registrador Contador
                                ; de Byte Remoto 0 (RBCR0)
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        pop DPH         ; Restaura o contador de bytes
        pop DPL

        mov A,DPH       ; Seto o Contador de bytes MSB do RBCR1
        push DPL        ; Armazena o contador de bytes
        push DPH
        REMOTEBYTECOUNT1        ; Inicializo o Registrador Contador
                                ; de Byte Remoto 0 (RBCR1)
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        mov A,R6        ; Seto o Contador de bytes LSB do RSAR 0
        REMOTESTARTADDRESS0     ; Inicializo o Registrador Endereço
                                ; Inicial Remoto 0 (RSAR0)
        movx @DPTR,A    ; Envia o comando para os registradores NIC


        mov A,R7        ; Seto o Contador de bytes MSB do RSAR 1
        REMOTESTARTADDRESS1     ; Inicializo o Registrador Endereço
                                ; Inicial Remoto 1 (RSAR1)
        movx @DPTR,A    ; Envia o comando para os registradores NIC
```

```
        mov A,#CR_LE    ; Inicializa e Lê no Registrador no Page 0
                        ; para Transmissão
        COMMAND         ; Inicializo o Comando dos Registradores (CR)
        movx @DPTR,A    ; Envia o comando para os registradores NIC

        mov R0,#RERAM   ; RERAM aponta para o pacote que ser  recebido(NIC)
        mov R1,#RERAM+1 ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0             ; Movo para DPH o MSB do endereço
        mov DPL,@R1             ; Movo para DPL o LSB do endereço
        mov R7,DPL      ; Salvo em R7 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R6,DPH      ; Salvo em R6 o valor do MSB do Endereço Inicial

        mov DPTR,#IOPORT; Endereço para acesso ao porto de Entrada/Saída
        mov R2,DPL      ; Salvo em R2 o valor do LSB do Endereço para iniciar
                        ; o IOPORT
        mov R3,DPH      ; Salvo em R3 o valor do MSB do Endereço para iniciar
                        ; o IOPORT
                •
READING_BYTE:
        pop DPH         ; Restauro MSB do contador
        pop DPL         ; Restauro LSB do contador
        push DPL        ; Salvo LSB do contador
        push DPH        ; Salvo MSB do contador
        mov A,DPL
        jnz MANDA1
        mov A,DPH
        jz NAO1         ; Vejo se contador = 0
MANDA1:
;NICRAM
        mov DPL,R2      ; Pego de volta o valor do LSB do Endereço do IOPORT
        mov DPH,R3      ; Pego de volta o valor do MSB do Endereço do IOPORT
        movx A,@DPTR    ; Escrevo no acumulador o pacote que foi recebido

;PCRAM
        mov DPL,R7      ; Pego de volta o valor do LSB da RAM do PC
        mov DPH,R6      ; Pego de volta o valor do MSB da RAM do PC
        movx @DPTR,A    ; Escrevo na RAM do PC o pacote que foi recebido
        inc DPTR        ; Próximo valor a ser lido
        mov R7,DPL      ; Salvo em R1 o valor do LSB da RAM do PC
        mov R6,DPH      ; Salvo em R0 o valor do MSB da RAM do PC

        pop DPH         ; Restauro MSB do contador
        pop DPL         ; Restauro LSB do contador
        push DPL        ; Salvo LSB do contador
        push DPH        ; Salvo MSB do contador
        mov R5,DPL      ; Pego o valor de R0
        cjeq R5,#04h,P?0        ; se R5=04h, leio o estado do registrador
        cjeq R5,#03h,P?1        ; se R5=03h, faço R0=next_packet
        cjeq R5,#02h,P?2        ; se R5=02h, faço R1=Contador Inferior
        cjeq R5,#01h,P?3        ; se R5=01h, faço R4=Contador Superior

;Salvar valores

P?0:
        sjmp P?4        ; Sai
```

```
P?1:
        push 00          ; Salvo na pilha o registrador pois vou utiliza-lo

        mov R2,#00H      ; Zero contador
        mov R1,A         ; Salvo em R1 o valor onde será escrito o próximo
                         ; pacote
        mov R0,#next_packet      ; Valor do próximo pacote atual MSB
        mov A,@R0        ; Salvo no acumulador

        cjne A,01H,CAL   ; Se não iguais, calcule...
        sjmp TCHAU       ; se iguais, sai...
CAL:    dec R1
        inc R2
        cjne A,01H,CAL   ; Se não iguais, calcule...

TCHAU:
;        mov A,R2
;        mov R0,#PCRAM    ; Valor do próximo buffer a ser lido
;        add A,@R0        ; Salvo no acumulador este valor
;        mov @R0,A        ; Salvo este valor em PCRAM
;        jnb first_reception,BYE ;Se for a primeira recepção, sair
;        mov R0,#RERAM    ; Valor do próximo buffer a ser lido
;        mov A,R2         ; Pego de volta o valor
;        add A,@R0        ; Salvo no acumulador este valor
;        mov @R0,A        ; Salvo este valor em RERAM
;        mov R0,#next_packet      ; Valor do próximo pacote atual MSB
;        mov A,R2         ; Pego de volta o valor
;        add A,@R0        ; Salvo no acumulador este valor
;        mov @R0,A        ; Salvo este valor em next_packet

        pop 02           ; Retorno da pilha o registrador que utilizei
        sjmp P?4         ; Sai
P?2:
        mov R1,A         ; Contador Inferior
        sjmp P?4         ; Sai
P?3:
        mov R4,A         ; Contador Superior
P?4:
;CONTADOR
BYE:
        pop DPH          ; Restauro MSB do contador
        pop DPL          ; Restauro LSB do contador
        lcall UTIL_DPTRDEC
        push DPL         ; Salvo LSB do contador
        push DPH         ; Salvo MSB do contador
        sjmp READING_BYTE
NAO1:
        pop DPH          ; Restauro MSB do contador
        pop DPL          ; Restauro LSB do contador

        ret

        ENDPROC


        END
```

```
; ************************************************************************
; * Descrição:                                                           *
; *     Rotina que coloca o header no endereço de memória que ser        *
; *     transmitido o pacote (4000h)                                     *
; * Entrada Requerida:                                                   *
; *     Nenhuma                                                          *
; * Na Saída:                                                            *
; *     O header ethernet , colocado no endereço 4000h - 400Dh           *
; *     O header ip , colocado no endereço 400Eh - 4021h                 *
; *     O header icmp , colocado no endereço 4022h - 4130h               *
; * Afetado:                                                             *
; *     Os registradores R0 e R1 do banco de registrador 2               *
; * Comentários:                                                         *
; *     Nenhum                                                           *
; ************************************************************************

        PUBLIC HEADER
        PUBLIC HEADER_IP
        PUBLIC HEADER_ICMP
        PUBLIC CRC_IP
        PUBLIC CRC_ICMP
        PUBLIC SOMA16
        PUBLIC TAMANHO
        EXTERN UTIL_DPTRDEC

        %include "8390.inc"

; Endereço de memória para a transmissão
PCRAM_IP        EQU     0Eh     ; Onde colocaremos o header ip
PCRAM_ICMP      EQU     22h     ; Onde colocaremos o header icmp

; Endereço de memória para colocarmos os Checksum's de IP e ICMP
RAM_IP          EQU     18h     ; Onde colocaremos o header IP
RAM_ICMP        EQU     24h     ; Onde colocaremos o header ICMP

; Endereço de memória para colocarmos no PCRAM os headers
END_HEADER      EQU     095h    ; Endereço para montarmos o header ethernet
END_IP          EQU     0C0h    ; Endereço para montarmos o header IP
TAM_TOTAL       EQU     00C2h   ; Endereço do header IP que da o tamanho do
                                ; pacote
END_ICMP        EQU     0E0h    ; Endereço para montarmos o header ICMP

; Comando do Chip no Registrador de Comando (CR)
CR_PAGE0        EQU     021h    ; Seleciona page 0 nos registradores do chip

; Bits Endereçáveis dos Registradores Especiais (SFR's)
CSN     BIT P1.2        ; Endereço do pino P1.2, usado como CS do NIC

; Endereço da memória para enfileirar o pacote na fila da memória do
; microprocessador
PCRAM   DATA 26H        ; Endereço da memória do microprocessador

HEADER  PROC

        push DPL        ; Salvo LSB do Contador de Bytes
        push DPH        ; Salvo MSB do Contador de Bytes
        push 00         ; Salvo MSB do Endereço Inicial Remoto
```

```
        push 01          ; Salvo LSB do Endereço Inicial Remoto

        mov R0,#PCRAM    ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1  ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0              ; Movo para DPH o MSB do endereço
        mov DPL,@R1              ; Movo para DPL o LSB do endereço
        mov R1,DPL       ; Salvo em R0 o valor do LSB do Endereço Inicial
                         ; Remoto
        mov R0,DPH       ; Salvo em R1 o valor do MSB do Endereço Inicial
                         ; Remoto
        mov DPTR,#END_HEADER  ; Endereço do hardware (0095h)

LE?1    clr A            ; Apago Acc para ler tabela do tamanho
                         ; do header (tam = 14 bytes)
        push DPH         ; Salvo MSB da tabela a ser lida
        push DPL         ; Salvo LSB da tabela a ser lida
        movc A,@A+DPTR   ; Pego o primeiro endereço do hardware

        cjne A,#0DH,OUTRO        ; Vê se , o último caracter a ser lido...
        sjmp OUT
OUTRO:
        mov DPL,R1       ; Retorno o valor LSB do PCRAM
        mov DPH,R0       ; Retorno o valor MSB do PCRAM
        movx @DPTR,A     ; Envia o comando para o PCRAM
        inc DPTR         ; Incremento Endereço da Memória
        mov R1,DPL       ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH       ; Salvo MSB do Endereço para Transmissão
        pop DPL          ; Pego de volta o LSB da tabela
        pop DPH          ; Pego de volta o MSB da tabela
        inc DPTR         ; Incremento DPTR para pegar o próximo endereço
        sjmp LE?1        ; Leia o próximo caracter
OUT:    pop DPL          ; Pego de volta o LSB da tabela
        pop DPH          ; Pego de volta o MSB da tabela

        pop 01           ; Pego de volta LSB do Endereço Inicial Remoto
        pop 00           ; Pego de volta MSB do Endereço Inicial Remoto
        pop DPH          ; Restauro o Contador de Bytes
        pop DPL

        ret

        ENDPROC

HEADER_IP       PROC

        push DPL         ; Salvo LSB do Contador de Bytes
        push DPH         ; Salvo MSB do Contador de Bytes
        push 00          ; Salvo MSB do Endereço Inicial Remoto
        push 01          ; Salvo LSB do Endereço Inicial Remoto

        mov R0,#PCRAM    ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1  ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0      ; Movo para DPH o MSB do endereço
        mov A,#PCRAM_IP  ; Endereço para inicializar o NIC
        add A,@R1        ; Salvo no Acumulador o LSB do endereço
        mov DPL,A        ; Movo para DPL o LSB do endereço
```

```
        mov R1,DPL      ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH      ; Salvo MSB do Endereço para Transmissão
        mov DPTR,#END_IP        ; Endereço do hardware (00C0h)

LE?2    clr A           ; Apago Acc para ler tabela do tamanho
                        ; do header (tam = 20 bytes)
        push DPH        ; Salvo MSB da tabela a ser lida
        push DPL        ; Salvo LSB da tabela a ser lida
        movc A,@A+DPTR  ; Pego o primeiro endereço do hardware

        cjne A,#0DH,OUTRO1      ; Vê se , o último caracter a ser lido...
        sjmp OUT1
OUTRO1:
        mov DPL,R1      ; Retorno o valor LSB do PCRAM
        mov DPH,R0      ; Retorno o valor MSB do PCRAM
        movx @DPTR,A    ; Envia o comando para o PCRAM
        inc DPTR        ; Incremento Endereço da Memória
        mov R1,DPL      ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH      ; Salvo MSB do Endereço para Transmissão
        pop DPL         ; Pego de volta o LSB da tabela
        pop DPH         ; Pego de volta o MSB da tabela
        inc DPTR        ; Incremento DPTR para pegar o próximo endereço

        sjmp LE?2       ; Leia o próximo caracter
OUT1:   pop DPL         ; Pego de volta o LSB da tabela
        pop DPH         ; Pego de volta o MSB da tabela

        acall CRC_IP    ; Vou calcular o Checksum do IP

        pop 01          ; Pego de volta LSB do Endereço Inicial Remoto
        pop 00          ; Pego de volta MSB do Endereço Inicial Remoto
        pop DPH         ; Restauro o Contador de Bytes
        pop DPL

        ret

        ENDPROC

HEADER_ICMP     PROC

        push DPL        ; Salvo LSB do Contador de Bytes
        push DPH        ; Salvo MSB do Contador de Bytes
        push 00         ; Salvo MSB do Endereço Inicial Remoto
        push 01         ; Salvo LSB do Endereço Inicial Remoto

        mov R0,#PCRAM   ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1 ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0     ; Movo para DPH o MSB do endereço
        mov A,#PCRAM_ICMP       ; Endereço para inicializar o NIC
        add A,@R1       ; Salvo no Acumulador o LSB do endereço
        mov DPL,A       ; Movo para DPL o LSB do endereço

        mov R1,DPL      ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH      ; Salvo MSB do Endereço para Transmissão

        mov DPTR,#END_ICMP      ; Endereço do hardware (0E0h)
```

```
LE?3    clr A           ; Apago Acc para ler tabela do tamanho
                        ; do header (tam = 08 bytes)
        push DPH        ; Salvo MSB da tabela a ser lida
        push DPL        ; Salvo LSB da tabela a ser lida
        movc A,@A+DPTR  ; Pego o primeiro endereço do hardware

        cjne A,#0DH,OUTRO2      ; Vê se , o último caracter a ser lido...
        sjmp OUT2
OUTRO2:
        mov DPL,R1      ; Retorno o valor LSB do PCRAM
        mov DPH,R0      ; Retorno o valor MSB do PCRAM
        movx @DPTR,A    ; Envia o comando para o PCRAM
        inc DPTR        ; Incremento Endereço da Memória
        mov R1,DPL      ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH      ; Salvo MSB do Endereço para Transmissão
        pop DPL         ; Pego de volta o MSB da tabela
        pop DPH         ; Pego de volta o LSB da tabela
        inc DPTR        ; Incremento DPTR para pegar o próximo endereço

        sjmp LE?3       ; Leia o próximo caracter
OUT2:   pop DPL         ; Pego de volta o LSB da tabela
        pop DPH         ; Pego de volta o MSB da tabela

        lcall CRC_ICMP  ; Vou calcular o Checksum do ICMP

        pop 01          ; Pego de volta LSB do Endereço Inicial Remoto
        pop 00          ; Pego de volta MSB do Endereço Inicial Remoto
        pop DPH         ; Restauro o Contador de Bytes
        pop DPL

        ret

        ENDPROC

; ***************************************************************************
; * Descrição:                                                              *
; *     Rotina que coloca o Checksum do header IP no endereço de memória    *
; *     que ser  transmitido o pacote (4018h - 4019h)                       *
; * Entrada Requerida:                                                      *
; *     Nenhuma                                                             *
; * Na Saída:                                                               *
; *     O header ip , colocado com o Checksum no endereço 400Eh - 4021h     *
; * Afetado:                                                                *
; *     Os registradores R2, R3, R4, R5, R6 e R7 do banco de registrador 2  *
; * Comentários:                                                            *
; *     Nenhum                                                              *
; ***************************************************************************

CRC_IP  PROC

        mov DPTR,#END_IP        ; Endereço do hardware (00C0h)
        setb RS1        ; Coloco no banco de registrador 2
        mov R7,#00h     ; Contador de byte.
        mov R6,#00h     ; Para fazer a soma dos Checksum
LE?4    clr A           ; Apago Acc para ler tabela do tamanho
                        ; do header (tam = 20 bytes)
        push DPH        ; Salvo MSB da tabela a ser lida
```

```
        push DPL        ; Salvo LSB da tabela a ser lida
        movc A,@A+DPTR  ; Pego o primeiro endereço do hardware

        cjne A,#0DH,CONTINUE    ; Vê se , o último caracter a ser lido...
        ljmp SAIR
CONTINUE:
        CPL A           ; Faço complemento de 1 do byte lido no header
        cjne R7,#00h,PROX       ; Vejo se e o primeiro byte a ser lido
        mov R2,A        ; Salvo em R2 o LSB no banco de registrador 2
        inc R7          ; Para ler o próximo byte
        pop DPL         ; Pego de volta o LSB da tabela
        pop DPH         ; Pego de volta o MSB da tabela
        inc DPTR        ; Incremento DPTR para pegar o próximo endereço

        ljmp LE?4       ; Leia o próximo caracter

PROX:   cjne R7,#01h,PROX1      ; Vejo se e o segundo byte a ser lido
        mov R3,A        ; Salvo em R3 o MSB no banco de registrador 2
        inc R7          ; Para ler o próximo byte
        pop DPL         ; Pego de volta o LSB da tabela
        pop DPH         ; Pego de volta o MSB da tabela
        inc DPTR        ; Incremento DPTR para pegar o próximo endereço

        cjne R6,#01h,CONT       ; se R6 != 01h, continue
        acall SOMA16
CONT:   ljmp LE?4       ; Leia o próximo caracter

PROX1:  cjne R7,#02h,PROX2      ; Vejo se e o terceiro byte a ser lido
        mov R4,A        ; Salvo em R4 o LSB no banco de registrador 2
        inc R7          ; Para ler o próximo byte
        pop DPL         ; Pego de volta o LSB da tabela
        pop DPH         ; Pego de volta o MSB da tabela
        inc DPTR        ; Incremento DPTR para pegar o próximo endereço

        ljmp LE?4       ; Leia o próximo caracter

PROX2:
        mov R5,A        ; Salvo em R5 o LSB no banco de registrador 2
        inc R6          ; Para somar as duas palavras (words)
        pop DPL         ; Pego de volta o LSB da tabela
        pop DPH         ; Pego de volta o MSB da tabela
        inc DPTR        ; Incremento DPTR para pegar o próximo endereço

        acall SOMA16

        ljmp LE?4       ; Leia o próximo caracter

SAIR:   pop DPL         ; Pego de volta o LSB da tabela
        pop DPH         ; Pego de volta o MSB da tabela

        mov R0,#PCRAM   ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1 ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0     ; Movo para DPH o MSB do endereço
        mov A,#RAM_IP   ; Inicializo o valor onde colocaremos o Checksum
        add A,@R1       ; Salvo no Acumulador o LSB do endereço
        mov DPL,A       ; Movo para DPL o LSB do endereço
```

```
        mov A,R4          ; Ponho primeiro o byte menos significativo
        movx @DPTR,A      ; Escrevo LSB no Checksum_IP
        inc DPTR          ; Inicializo o próximo endereço no qual escreveremos
        mov A,R5          ; Ponho segundo o byte mais significativo
        movx @DPTR,A      ; Escrevo MSB do Checksum_IP

        clr RS1           ; Volto no banco de registrador 0

        ret

        ENDPROC

; ************************************************************************
; * Descrição:                                                           *
; *     Rotina que coloca o Checksum do header ICMP no endereço de memória  *
; *     que ser  transmitido o pacote (4024h - 4025h)                    *
; * Entrada Requerida:                                                   *
; *     Nenhuma                                                          *
; * Na Saída:                                                            *
; *     O header icmp , colocado com o Checksum no endereço 4024h - 4035h   *
; * Afetado:                                                             *
; *     Os registradores R2, R3, R4, R5, R6 e R7 do banco de registrador 2  *
; * Comentários:                                                         *
; *     Nenhum                                                           *
; ************************************************************************

CRC_ICMP        PROC

        setb RS1          ; Coloco no banco de registrador 2
        mov DPTR,#END_ICMP      ; Endereço do hardware (00E0h)
        mov R0,#PCRAM     ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1   ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0       ; Movo para DPH o MSB do endereço
        mov A,#PCRAM_ICMP       ; Endereço para inicializar o NIC
        add A,@R1         ; Salvo no Acumulador o LSB do endereço
        mov DPL,A         ; Movo para DPL o LSB do endereço
        mov R1,DPL        ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH        ; Salvo MSB do Endereço para Transmissão

        acall TAMANHO     ; Esta rotina vê o tamanho do header para calcular CRC
        mov R7,#00h       ; Contador de byte.
        mov R6,#00h       ; Para fazer a soma dos Checksum
LE?5:
        push DPH          ; Salvo MSB da tabela a ser lida
        push DPL          ; Salvo LSB da tabela a ser lida
        mov A,DPL
        jnz VAI
        mov A,DPH
        jz SAIR1          ; Vejo se contador = 0
VAI:
        mov DPL,R1        ; Retorno o valor LSB do PCRAM
        mov DPH,R0        ; Retorno o valor MSB do PCRAM
        movx A,@DPTR      ; Leio o valor do header em PCRAM
        inc DPTR          ; Incremento Endereço da Memória
        mov R1,DPL        ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH        ; Salvo MSB do Endereço para Transmissão
        pop DPL           ; Pego de volta o MSB da tabela
```

```
        pop DPH          ; Pego de volta o LSB da tabela
        lcall UTIL_DPTRDEC

        CPL A            ; Faço complemento de 1 do byte lido no header
        cjne R7,#00h,PROX3       ; Vejo se e o primeiro byte a ser lido
        mov R2,A         ; Salvo em R2 o LSB no banco de registrador 2
        inc R7           ; Para ler o próximo byte
        ljmp LE?5        ; Leia o próximo caracter

PROX3:  cjne R7,#01h,PROX4       ; Vejo se e o segundo byte a ser lido
        mov R3,A         ; Salvo em R3 o MSB no banco de registrador 2
        inc R7           ; Para ler o próximo byte
        cjne R6,#01h,CONT1       ; se R6 != 01h, continue
        acall SOMA16

CONT1:  ljmp LE?5        ; Leia o próximo caracter

PROX4:  cjne R7,#02h,PROX5       ; Vejo se e o terceiro byte a ser lido
        mov R4,A         ; Salvo em R4 o LSB no banco de registrador 2
        inc R7           ; Para ler o próximo byte
        ljmp LE?5        ; Leia o próximo caracter
PROX5:
        mov R5,A         ; Salvo em R5 o LSB no banco de registrador 2
        inc R6           ; Para somar as duas palavras (words)
        acall SOMA16

        ljmp LE?5        ; Leia o próximo caracter
SAIR1:  pop DPL          ; Pego de volta o LSB da tabela
        pop DPH          ; Pego de volta o MSB da tabela

        mov R0,#PCRAM    ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1  ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0      ; Movo para DPH o MSB do endereço
        mov A,#RAM_ICMP  ; Inicializo o valor onde colocaremos o Checksum para
                         ; o ICMP
        add A,@R1        ; Salvo no Acumulador o LSB do endereço
        mov DPL,A        ; Movo para DPL o LSB do endereço

        mov A,R4         ; Ponho primeiro o byte menos significativo
        movx @DPTR,A     ; Escrevo LSB no Checksum_ICMP
        inc DPTR         ; Inicializo o próximo endereço no qual escreveremos
        mov A,R5         ; Ponho segundo o byte mais significativo
        movx @DPTR,A     ; Escrevo MSB do Checksum_ICMP

        clr RS1          ; Volto no banco de registrador 0

        ret

        ENDPROC

; ************************************************************************
; * Descrição:                                                           *
; *     Rotina que somara B/Acc com DPTR e setar  o carry se ocorrer um  *
; *     overflow em DPTR.                                                *
; * Entrada Requerida:                                                   *
; *     DPTR tem um valor para adicionar                                 *
; *     'B' tem o valor auto para adicionar, Acc ter  valor baixo para somar*
```

```
; * Na Saída:                                                                *
; *       R5/R4 = DPTR + B/Acc, CY Setado de acordo com a soma               *
; * Afetado:                                                                 *
; *     PSW.CY, R5 -> MSB do Checksum e R4 -> LSB do Checksum                *
; * Comentários:                                                             *
; *     Valor do CY na entrada não afetar  no resultado                      *
; ***************************************************************************


SOMA16  PROC
        push Acc          ; Salvo acumulador na pilha
        push DPL          ; Armazena o contador de bytes
        push DPH

        mov A,R2          ; Jogo o LSB da primeira word a ser somada
        mov B,R3          ; Jogo o MSB da primeira word a ser somada
        mov DPL,R4        ; Jogo o LSB da segunda word a ser somada
        mov DPH,R5        ; Jogo o MSB da segunda word a ser somada

        add A,DPL         ; Adiciono R2 com R4
        mov R4,A          ; Movo o Resultado para R4
        mov A,DPH         ; Pego DPH
        addc A,B          ; Adiciono R3 com R5 + CY
        jc SOMAC          ; Somo mais uma vez o carry
        sjmp NEXT         ; Se não tiver carry, sai...
SOMAC:  clr C             ; Apago o carry
        inc R4            ; Somo o carry

NEXT:   mov R5,A          ; Move resultado para R5
        pop DPH           ; Restaura o contador de bytes
        pop DPL
        pop Acc           ; Restaura acumulador na pilha

        mov R7,#00h       ; Leremos a nova word para somarmos

        ret       ; Retorna ao Chamador

        ENDPROC

; ***************************************************************************
; * Descrição:                                                               *
; *     Rotina que vê o tamanho do header IP para calcular CRC do ICMP       *
; * Entrada Requerida:                                                       *
; *     Nenhuma                                                              *
; * Na Saída:                                                                *
; *     DPTR terá o valor do tamanho do pacote a ser lido para calcular CRC  *
; * Afetado:                                                                 *
; *     Nenhum, DPH -> MSB do Pacote e DPL -> LSB do Pacote                  *
; * Comentários:                                                             *
; *     Nenhum                                                               *
; ***************************************************************************

TAMANHO PROC

        mov DPTR,#TAM_TOTAL      ; Pega o tamanho do pacote
        clr A
        movc A,@A+DPTR  ; Pego primeiro o MSB do Tamanho
```

```
mov DPH,A          ; Salvo MSB em DPH
push DPH           ; Salvo MSB do contador
inc DPTR
clr A
movc A,@A+DPTR     ; Pego segundo o LSB do Tamanho
mov DPL,A          ; Salvo LSB em DPL
push DPL           ; Salvo LSB do contador

clr A
mov DPTR,#END_IP          ; Endereço de version_IHL
movc A,@A+DPTR     ; Pego o IHL e version
anl A,#0Fh         ; Só nos interessa o IHL
mov B,#04h         ; Para multiplicar por 4 bytes
mul AB             ; Vejo tamanho do header IP para descontar no tamanho

mov R4,A           ; Salvo valor em R4
pop DPL            ; Retorno o valor LSB do tamanho do header
mov A,DPL          ; Salvo em a
clr C              ; Apago carry se existir
subb A,R4          ; Faço Acc = Acc - R4
mov DPL,A          ; Salvo resultado em DPL
mov A,B            ; Pego MSB do tamanho do header IP
mov R3,A           ; Salvo valor em R3
pop DPH            ; Retorno o valor MSB do tamanho do header
mov A,DPH          ; Salvo em A
subb A,R3          ; Faço Acc = Acc - R3
mov DPH,A          ; Salvo resultado em DPH

ret

ENDPROC

END
```

```
; ************************************************************************
; * Descrição:                                                           *
; * QUEUE_PACKET , uma Rotina que tem a função de enfileirar os pacotes a *
; * seremátransmitidos, caso não possamos transmití-lo pelo fato de o    *
; * receptor estar ocupado, e ou, enfileirar os pacotes que começamos a  *
; * receber, mas ainda não podemos process -los, pois estamos transmitindo *
; * um pacote neste mesmo momento. O ponteiro headptr aponta para o próximo *
; * espaço disponível no buffer e o ponteiro tailptr aponta para o próximo *
; * buffer a ser lido.                                                   *
; * Entrada Requerida:                                                   *
; *     Contador de bytes do pacote (byte_count), valor do ponteiro da   *
; *     fila (tailptr) e endereço da memória do microprocessador (PCRAM) *
; * Na saída:                                                            *
; *     headptr(6586h), tailptr(6000h) e byte_count(0586h)               *
; * Registradores Afetados:                                              *
; *     DPTR, R0, R1, R4, R5, R6 e R7 (banco de registrador 0)           *
; * Comentário:                                                          *
; *     Nenhum                                                           *
; ************************************************************************


        PUBLIC QUEUE_PACKET
        EXTERN UTIL_DPTRDEC



; Bit e bytes endereçável na RAM interna
Check   BIT 20H.2       ; Endereço de memória que nos diz que pacote foi
                        ; enfileirado


; Endereço da memória para enfileirar o pacote na fila da memória do
; microprocessador
PCRAM   DATA 26H        ; Endereço da memória do microprocessador
tailptr DATA    28H     ; Valor do próximo buffer a ser lido
headptr DATA    2AH     ; Valor do próximo espaço disponível no buffer
byte_count      DATA 2CH; Valor do contador de byte para transmissão

QUEUE_PACKET    PROC

        jnb Check,LP?2  ; J  existe algu,m na fila? Se sim, ir para LP?2
        sjmp LP?1       ; Se não, ir para LP?1

LP?1
        mov R0,#byte_count      ; Valor do contador de byte para transmissão
                                ; MSB
        mov R1,#(byte_count+1)  ; Valor do contador de byte para transmissão
                                ; LSB
        mov DPH,@R0             ; Movo para DPH o MSB do contador
        mov DPL,@R1             ; Movo para DPL o LSB do contador
        push DPL        ; Salvo LSB do contador
        push DPH        ; Salvo MSB do contador

        mov R0,#tailptr         ; Valor do próximo buffer a ser lido
        mov R1,#(tailptr+1)     ; Valor do próximo buffer a ser lido
        mov DPH,@R0             ; Movo para DPH o MSB do buffer
        mov DPL,@R1             ; Movo para DPL o LSB do buffer
```

```
        mov R4,DPL      ; Salvo em R4 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R5,DPH      ; Salvo em R5 o valor do MSB do Endereço Inicial
                        ; Remoto

        mov R0,#PCRAM           ; Valor do próximo buffer a ser lido (4100H)
        mov R1,#(PCRAM+1)       ; Valor do próximo buffer a ser lido
        mov DPH,@R0             ; Movo para DPH o MSB do buffer
        mov DPL,@R1             ; Movo para DPL o LSB do buffer
        mov R6,DPL      ; Salvo em R4 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R7,DPH      ; Salvo em R5 o valor do MSB do Endereço Inicial
                        ; Remoto

COPY_BYTE1:

        pop DPH         ; Restauro MSB do contador
        pop DPL         ; Restauro LSB do contador
        push DPL        ; Salvo LSB do contador
        push DPH        ; Salvo MSB do contador

        mov A,DPL
        jnz MANDA4
        mov A,DPH
        jz NAO4         ; Vejo se contador = 0

MANDA4:

;PCRAM
        mov DPL,R6      ; Pego de volta o valor do LSB do Endereço Inicial
                        ; Remoto em R6
        mov DPH,R7      ; Pego de volta o valor do MSB do Endereço Inicial
                        ; Remoto em R7
        movx A,@DPTR    ; Escrevo no acumulador o dado que estava na fila
        inc DPTR        ; Próximo valor a ser lido para colocar na fila
        mov R6,DPL      ; Salvo em R6 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R7,DPH      ; Salvo em R7 o valor do MSB do Endereço Inicial
                        ; Remoto
;tailptr
        mov DPL,R4      ; Pego de volta o valor do LSB do Endereço Inicial
                        ; Remoto em R4
        mov DPH,R5      ; Pego de volta o valor do MSB do Endereço Inicial
                        ; Remoto em R5
        movx @DPTR,A    ; Escrevo no Endereço Remoto o dado que estava na fila
        inc DPTR        ; Próximo valor a ser escrito na fila
        mov R4,DPL      ; Salvo em R4 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R5,DPH      ; Salvo em R5 o valor do MSB do Endereço Inicial
                        ; Remoto

        pop DPH         ; Restauro MSB do contador
        pop DPL         ; Restauro LSB do contador
        lcall UTIL_DPTRDEC
        push DPL        ; Salvo LSB do contador
        push DPH        ; Salvo MSB do contador
        sjmp COPY_BYTE1
```

```
NAO4:
;headptr
        mov R0,#headptr          ; Valor do ponteiro final do buffer
        mov R1,#(headptr+1)      ; Valor do ponteiro final do buffer
        mov @R0,05               ; Movo para headptr o MSB do tamanho do buffer
        mov @R1,04               ; Movo para headptr o LSB do tamanho do buffer

;tailptr
        mov R0,#tailptr          ; Valor do próximo buffer a ser lido
        mov R1,#(tailptr+1)      ; Valor do próximo buffer a ser lido
        mov DPH,@R0              ; Movo para DPH o MSB do buffer
        mov DPL,@R1              ; Movo para DPL o LSB do buffer
        mov R0,DPL       ; Salvo em R0 o valor do LSB do Endereço Inicial
                         ; Remoto
        mov R1,DPH       ; Salvo em R1 o valor do MSB do Endereço Inicial
                         ; Remoto

;byte_count
        pop DPH          ; Restauro MSB do contador
        pop DPL          ; Restauro LSB do contador
        mov R0,#byte_count       ; Valor do contador de byte para transmissão
        mov R1,#(byte_count+1)   ; Valor do contador de byte para transmissão
        mov DPH,@R0              ; Movo para DPH o MSB do contador
        mov DPL,@R1              ; Movo para DPL o LSB do contador
GO:
        ret

        ENDPROC

LP?2    PROC
        clr Check

;byte_count
        mov R0,#byte_count       ; Valor do contador de byte para transmissão
        mov R1,#(byte_count+1)   ; Valor do contador de byte para transmissão
        mov DPH,@R0              ; Movo para DPH o MSB do contador
        mov DPL,@R1              ; Movo para DPL o LSB do contador
;tailptr
        mov R0,#tailptr          ; Valor do próximo buffer a ser lido
        mov R1,#(tailptr+1)      ; Valor do próximo buffer a ser lido
        mov 05,@R0              ; Movo para DPH o MSB do buffer
        mov 04,@R1              ; Movo para DPL o LSB do buffer
        mov R0,05        ; Salvo em R0 o valor do MSB do Endereço Inicial
                         ; Remoto
        mov R1,04        ; Salvo em R1 o valor do LSB do Endereço Inicial
                         ; Remoto

        ret

        ENDPROC

        END
```

```
; ************************************************************************
; * Descrição:                                                           *
; * PING , uma Rotina que tem a função de fazer um "replay" dos pacotes  *
; * recebidos.                                                           *
; * Entrada Requerida:                                                   *
; *     Contador de bytes do pacote (byte_count), valor do ponteiro da   *
; *     fila (tailptr) e endereço da memória do microprocessador (PCRAM) *
; * Na saída:                                                            *
; *     headptr (tailptr+byte_count), tailptr(4600h) e byte_count        *
; * Registradores Afetados:                                              *
; *     DPTR, R0, R1, R4, R5, R6 e R7 (banco de registrador 0)           *
; * Comentário:                                                          *
; *     Nenhum                                                           *
; ************************************************************************


        PUBLIC PING
        EXTERN UTIL_DPTRDEC
        EXTERN ETH
        EXTERN IPH
        EXTERN ICMP
        EXTERN PCtoNIC

        %include "8390.inc"

; Pseudo Variáveis na ROM
PRAM   equ 04H          ; Endereço da memória do microprocessador
CS     equ 0A000H       ; Endereço de memória de acesso ao NIC
IOPORT equ 0E000H       ; Endereço de memória para acessar os dados no
                        ; barramento local

; Comando do Chip no Registrador de Comando (CR)
CR_ESCR        EQU     012h     ; Escreve no Registrador
CR_TRANS       EQU     026h     ; Transmite um frame

; Bit endereçável na RAM interna
CSN     BIT P1.2        ; Endereço do pino P1.2, usado como CS do NIC

; Endereço da memória para enfileirar o pacote na fila da memória do
; microprocessador
NICRAM  DATA 22H        ; Endereço da memória do controlador (transmissão)
RERAM   DATA 24H        ; Endereço da memória do microprocessador (recepção)
PCRAM   DATA 26H        ; Endereço da memória do microprocessador
tailptr DATA    28H     ; Valor do próximo buffer a ser lido
headptr DATA    2AH     ; Valor do próximo espaço disponível no buffer
byte_count      DATA 2CH; Valor do contador de byte para transmissão


PING    PROC

        mov A,R4
        mov byte_count,A        ; Valor MSB do contador de byte para
                                ; transmissão
        mov A,R1
        mov (byte_count+1),A    ; Valor LSB do contador de byte para
                                ; transmissão
```

```
        mov R0,#byte_count      ; Valor do contador de byte para transmissão
        mov R1,#(byte_count+1)  ; Valor do contador de byte para transmissão
        mov DPH,@R0             ; Movo para DPH o MSB do contador
        mov DPL,@R1             ; Movo para DPL o LSB do contador
        push DPL        ; Salvo LSB do contador
        push DPH        ; Salvo MSB do contador

        mov R0,#PCRAM           ; Valor do próximo buffer a ser lido (4100H)
        mov R1,#(PCRAM+1)       ; Valor do próximo buffer a ser lido
        mov DPH,@R0             ; Movo para DPH o MSB do buffer
        mov DPL,@R1             ; Movo para DPL o LSB do buffer
        mov R4,DPL      ; Salvo em R4 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R5,DPH      ; Salvo em R5 o valor do MSB do Endereço Inicial
                        ; Remoto

        mov R0,#RERAM   ; RERAM aponta para o pacote que ser  recebido(NIC)
        mov R1,#RERAM+1 ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0             ; Movo para DPH o MSB do endereço
        mov A,#PRAM     ; Endereço para inicializar o NIC
        add A,@R1       ; Salvo no Acumulador o LSB do endereço
        mov DPL,A       ; Movo para DPL o LSB do endereço
        mov R6,DPL      ; Salvo em R6 o valor do LSB do Endereço Inicial
        mov R7,DPH      ; Salvo em R7 o valor do MSB do Endereço Inicial


COPY_BYTE:
        pop DPH         ; Restauro MSB do contador
        pop DPL         ; Restauro LSB do contador
        push DPL        ; Salvo LSB do contador
        push DPH        ; Salvo MSB do contador

        mov A,DPL
        jnz MANDA3
        mov A,DPH
        jz NAO3         ; Vejo se contador = 0

MANDA3:
;PCRAM
        mov DPL,R6      ; Pego de volta o valor do LSB do Endereço Inicial
                        ; Remoto em R6
        mov DPH,R7      ; Pego de volta o valor do MSB do Endereço Inicial
                        ; Remoto em R7
        movx A,@DPTR    ; Escrevo no acumulador o dado que estava na fila
        inc DPTR        ; Próximo valor a ser lido para colocar na fila
        mov R6,DPL      ; Salvo em R6 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R7,DPH      ; Salvo em R7 o valor do MSB do Endereço Inicial
                        ; Remoto
;tailptr
        mov DPL,R4      ; Pego de volta o valor do LSB do Endereço Inicial
                        ; Remoto em R4
        mov DPH,R5      ; Pego de volta o valor do MSB do Endereço Inicial
                        ; Remoto em R5
        movx @DPTR,A    ; Escrevo no Endereço Remoto o dado que estava na fila
```

```
        inc DPTR        ; Próximo valor a ser escrito na fila
        mov R4,DPL      ; Salvo em R4 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R5,DPH      ; Salvo em R5 o valor do MSB do Endereço Inicial
                        ; Remoto

        pop DPH         ; Restauro MSB do contador
        pop DPL         ; Restauro LSB do contador
        lcall UTIL_DPTRDEC
        push DPL        ; Salvo LSB do contador
        push DPH        ; Salvo MSB do contador
        sjmp COPY_BYTE
NAO3:

        pop DPH         ; Restauro MSB do contador
        pop DPL         ; Restauro LSB do contador

        lcall ETH       ; Chamo rotina que põe header no PCRAM
        lcall IPH       ; Chamo rotina que põe header IP no PCRAM
        lcall ICMP      ; Chamo rotina que põe header ICMP no PCRAM

        mov DPTR,#IOPORT; IOPORT aponta para o porto de entrada e saída
        mov R4,DPL      ; Salvo em R4 o valor do LSB do Endereço para acessar
                        ; o porto de entrada e saída
        mov R5,DPH      ; Salvo em R5 o valor do MSB do Endereço para acessar
                        ; o porto de entrada e saída

        mov R0,#NICRAM  ; NICRAM aponta para o pacote que ser  transmitido
        mov R1,#NICRAM+1; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0     ; Movo para DPH o MSB do endereço
        mov DPL,@R1     ; Movo para DPL o LSB do endereço
        mov R6,DPL      ; Salvo em R6 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R7,DPH      ; Salvo em R7 o valor do MSB do Endereço Inicial
                        ; Remoto

        mov R0,#PCRAM   ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1 ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0             ; Movo para DPH o MSB do endereço
        mov DPL,@R1             ; Movo para DPL o LSB do endereço
        mov R1,DPL      ; Salvo em R0 o valor do LSB do Endereço Inicial
                        ; Remoto
        mov R0,DPH      ; Salvo em R1 o valor do MSB do Endereço Inicial
                        ; Remoto

        push 00         ; Salvo MSB do Endereço Inicial da RAM do PC
        push 01         ; Salvo LSB do Endereço Inicial da RAM do PC

        mov R0,#byte_count      ; Valor do contador de byte para transmissão
        mov R1,#(byte_count+1)  ; Valor do contador de byte para transmissão
        mov DPH,@R0             ; Movo para DPH o MSB do contador
        mov DPL,@R1             ; Movo para DPL o LSB do contador
        mov R2,DPL      ; Salvo em R2 o valor do LSB do Contador de Bytes
                        ; Remoto
        mov R3,DPH      ; Salvo em R3 o valor do MSB do Contador de Bytes
                        ; Remoto
        pop 01          ; Pego de volta MSB da RAM do PC
```

```
pop 00          ; Pego de volta LSB da RAM do PC
push DPH        ; Salvo contador
push DPL

lcall PCtoNIC   ; Transfere pacote do PC para o Buffer RAM do NIC

TRANSMITPAGE    ; Inicializo o Registrador de Início de Transmissão
                ; (TPSR)
mov R0,#NICRAM  ; NICRAM aponta para o pacote que ser  transmitido
mov A,@R0       ; Movo para A o MSB do endereço
movx @DPTR,A    ; Envia o comando para os registradores NIC

pop DPL         ; Restauro valor do contador
pop DPH
mov A,DPL       ; Pega o valor LSB do TBCR0
push DPH        ; Salvo contador
push DPL
TRANSMITBYTECOUNT0      ; Inicializo o Registrador Contador
                        ; de Byte Transmitido 0 (TBCR0)
movx @DPTR,A    ; Envia o comando para os registradores NIC

pop DPL         ; Restauro valor do contador
pop DPH
mov A,DPH       ; Pega o valor MSB do TBCR1
TRANSMITBYTECOUNT1      ; Inicializo o Registrador Contador
                        ; de Byte Transmitido 1 (TBCR1)
movx @DPTR,A    ; Envia o comando para os registradores NIC

COMMAND         ; Inicializo o Comando dos Registradores (CR)
mov A,#CR_TRANS ; Inicializo o Registrador no Page 0 para Transmissão
movx @DPTR,A    ; Envia o comando para os registradores NIC

ret

ENDPROC

END
```

```
; ************************************************************************
; * Descrição:                                                           *
; *     Rotina que coloca o header no endereço de memória que ser        *
; *     transmitido o pacote (4000h)                                     *
; * Entrada Requerida:                                                   *
; *     Nenhuma                                                          *
; * Na Saída:                                                            *
; *     O header ethernet , colocado no endereço 4100h - 410Dh           *
; *     O header ip , colocado no endereço 410Eh - 4121h                 *
; *     O header icmp , colocado no endereço 4122h - 4230h               *
; * Afetado:                                                             *
; *     Os registradores R0 e R1 do banco de registrador 2               *
; * Comentários:                                                         *
; *     Nenhum                                                           *
; ************************************************************************

        PUBLIC ETH
        PUBLIC IPH
        PUBLIC CHECK_IP
        PUBLIC ICMP
        PUBLIC CHECK_ICMP
        PUBLIC TAMANHO1
        EXTERN SOMA16
        EXTERN UTIL_DPTRDEC
        %include "8390.inc"

; Endereço de memória para a transmissão
PCRAM_IP        EQU     0Eh     ; Onde colocaremos o header icmp
TAM             EQU     010h    ; Ondeáveremos o tamanho do pacote
PCRAM_ICMP      EQU     22h     ; Onde colocaremos o header icmp

; Endereço de memória para colocarmos os Checksum's de IP e ICMP
RAM_IP          EQU     18h     ; Onde colocaremos o header IP
RAM_ICMP        EQU     24h     ; Onde colocaremos o header ICMP

; Endereço de memória para colocarmos no PCRAM os headers
END_HEADER      EQU     095h    ; Endereço para montarmos o header ethernet
END_IP          EQU     0CAh    ; Endereço para montarmos o header IP
END_ICMP        EQU     0E0h    ; Endereço para montarmos o header ICMP

; Comando do Chip no Registrador de Comando (CR)
CR_PAGE0        EQU     021h    ; Seleciona page 0 nos registradores do chip

; Bits Endereçáveis dos Registradores Especiais (SFR's)
CSN     BIT P1.2        ; Endereço do pino P1.2, usado como CS do NIC

; Endereço da memória para enfileirar o pacote na fila da memória do
; microprocessador
PCRAM   DATA 26H        ; Endereço da memória do microprocessador
byte_count      DATA 2CH; Valor do contador de byte para transmissão

ETH     PROC

        push DPL        ; Salvo LSB do Contador de Bytes
        push DPH        ; Salvo MSB do Contador de Bytes
        push 00         ; Salvo MSB do Endereço Inicial Remoto
```

```
        push 01          ; Salvo LSB do Endereço Inicial Remoto

        mov R0,#PCRAM    ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1  ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0             ; Movo para DPH o MSB do endereço
        mov DPL,@R1             ; Movo para DPL o LSB do endereço
        mov R1,DPL       ; Salvo em R0 o valor do LSB do Endereço Inicial
                         ; Remoto
        mov R0,DPH       ; Salvo em R1 o valor do MSB do Endereço Inicial
                         ; Remoto
        mov DPTR,#END_HEADER  ; Endereço do hardware (0095h)

LE?1    clr A            ; Apago Acc para ler tabela do tamanho
                         ; do header (tam = 14 bytes)
        push DPH         ; Salvo MSB da tabela a ser lida
        push DPL         ; Salvo LSB da tabela a ser lida
        movc A,@A+DPTR   ; Pego o primeiro endereço do hardware
        cjeq A,#0DH,VAM  ; Vê se não , o último caracter
        mov DPL,R1       ; Retorno o valor LSB do PCRAM
        mov DPH,R0       ; Retorno o valor MSB do PCRAM
        movx @DPTR,A     ; Envia o comando para o PCRAM
        inc DPTR         ; Incremento Endereço da Memória
        mov R1,DPL       ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH       ; Salvo MSB do Endereço para Transmissão
        pop DPL          ; Pego de volta o LSB da tabela
        pop DPH          ; Pego de volta o MSB da tabela
        inc DPTR         ; Incremento DPTR para pegar o próximo endereço
        sjmp LE?1
VAM:    pop DPL          ; Pego de volta o LSB da tabela
        pop DPH          ; Pego de volta o MSB da tabela
        pop 01           ; Pego de volta LSB do Endereço Inicial Remoto
        pop 00           ; Pego de volta MSB do Endereço Inicial Remoto
        pop DPH          ; Restauro o Contador de Bytes
        pop DPL

        ret

        ENDPROC

IPH     PROC

        push DPL         ; Salvo LSB do Contador de Bytes
        push DPH         ; Salvo MSB do Contador de Bytes
        push 00          ; Salvo MSB do Endereço Inicial Remoto
        push 01          ; Salvo LSB do Endereço Inicial Remoto

        mov R0,#PCRAM    ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1  ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0      ; Movo para DPH o MSB do endereço
        mov A,#RAM_IP    ; Endereço para inicializar o NIC
        add A,@R1        ; Salvo no Acumulador o LSB do endereço
        mov DPL,A        ; Movo para DPL o LSB do endereço
        mov R1,DPL       ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH       ; Salvo MSB do Endereço para Transmissão

        mov DPTR,#END_IP         ; Endereço do hardware (00CAh)
```

```
LE?2    clr A           ; Apago Acc para ler tabela do tamanho
                        ; do header (tam = 10 bytes)
        push DPH        ; Salvo MSB da tabela a ser lida
        push DPL        ; Salvo LSB da tabela a ser lida
        movc A,@A+DPTR  ; Pego o primeiro endereço do hardware
        mov DPL,R1      ; Retorno o valor LSB do PCRAM
        mov DPH,R0      ; Retorno o valor MSB do PCRAM
        movx @DPTR,A    ; Envia o comando para o PCRAM
        inc DPTR        ; Incremento Endereço da Memória
        mov R1,DPL      ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH      ; Salvo MSB do Endereço para Transmissão
        pop DPL         ; Pego de volta o LSB da tabela
        pop DPH         ; Pego de volta o MSB da tabela
        inc DPTR        ; Incremento DPTR para pegar o próximo endereço

        cjne A,#0DH,LE?2        ; Vê se , o último caracter a ser lido...

        lcall CHECK_IP  ; Vou calcular o Checksum do IP

        pop 01          ; Pego de volta LSB do Endereço Inicial Remoto
        pop 00          ; Pego de volta MSB do Endereço Inicial Remoto
        pop DPH         ; Restauro o Contador de Bytes
        pop DPL

        ret

        ENDPROC

ICMP    PROC

        push DPL        ; Salvo LSB do Contador de Bytes
        push DPH        ; Salvo MSB do Contador de Bytes
        push 00         ; Salvo MSB do Endereço Inicial Remoto
        push 01         ; Salvo LSB do Endereço Inicial Remoto

        mov R0,#PCRAM   ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1 ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0     ; Movo para DPH o MSB do endereço
        mov A,#PCRAM_ICMP       ; Endereço para inicializar o NIC
        add A,@R1       ; Salvo no Acumulador o LSB do endereço
        mov DPL,A       ; Movo para DPL o LSB do endereço

        mov R1,DPL      ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH      ; Salvo MSB do Endereço para Transmissão
        mov DPTR,#END_ICMP      ; Endereço do hardware (0E0h)

        clr A           ; Apago Acc para dar um replay do echo request
                        ; recebido
        push DPH        ; Salvo MSB da tabela a ser lida
        push DPL        ; Salvo LSB da tabela a ser lida
        mov DPL,R1      ; Retorno o valor LSB do PCRAM
        mov DPH,R0      ; Retorno o valor MSB do PCRAM
        movx @DPTR,A    ; Envia o comando para o PCRAM
        inc DPTR        ; Incremento Endereço da Memória
        mov R1,DPL      ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH      ; Salvo MSB do Endereço para Transmissão
        pop DPL         ; Pego de volta o MSB da tabela
```

```
        pop DPH          ; Pego de volta o LSB da tabela
        inc DPTR         ; Incremento DPTR para pegar o próximo endereço

LE?3    clr A            ; Apago Acc para ler tabela do tamanho
                         ; do header (tam = 03 bytes)
        push DPH         ; Salvo MSB da tabela a ser lida
        push DPL         ; Salvo LSB da tabela a ser lida
        movc A,@A+DPTR   ; Pego o primeiro endereço do hardware
        mov DPL,R1       ; Retorno o valor LSB do PCRAM
        mov DPH,R0       ; Retorno o valor MSB do PCRAM
        movx @DPTR,A     ; Envia o comando para o PCRAM
        inc DPTR         ; Incremento Endereço da Memória
        mov R1,DPL       ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH       ; Salvo MSB do Endereço para Transmissão
        pop DPL          ; Pego de volta o MSB da tabela
        pop DPH          ; Pego de volta o LSB da tabela
        inc DPTR         ; Incremento DPTR para pegar o próximo endereço
        mov A,DPL

        cjne A,#0E4H,LE?3        ; Vê se , o último caracter a ser lido...

        lcall CHECK_ICMP         ; Vou calcular o Checksum do ICMP

        pop 01           ; Pego de volta LSB do Endereço Inicial Remoto
        pop 00           ; Pego de volta MSB do Endereço Inicial Remoto
        pop DPH          ; Restauro o Contador de Bytes
        pop DPL

        ret

        ENDPROC

; ********************************************************************************
; * Descrição:                                                                   *
; *     Rotina que coloca o Checksum do header IP no endereço de memória         *
; *     que ser  transmitido o pacote (4018h - 4019h)                            *
; * Entrada Requerida:                                                           *
; *     Nenhuma                                                                  *
; * Na Saída:                                                                    *
; *     O header ip , colocado com o Checksum no endereço 400Eh - 4021h          *
; * Afetado:                                                                     *
; *     Os registradores R2, R3, R4, R5, R6 e R7 do banco de registrador 2       *
; * Comentários:                                                                 *
; *     Nenhum                                                                   *
; ********************************************************************************

CHECK_IP        PROC

        mov R0,#PCRAM    ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1  ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0      ; Movo para DPH o MSB do endereço
        mov A,#PCRAM_IP  ; Endereço para inicializar o NIC
        add A,@R1        ; Salvo no Acumulador o LSB do endereço
        mov DPL,A        ; Movo para DPL o LSB do endereço

        setb RS1         ; Coloco no banco de registrador 2
        mov R7,#00h      ; Contador de byte.
```

```
        mov R6,#00h      ; Para fazer a soma dos Checksum

LE?4    clr A            ; Apago Acc para ler tabela do tamanho
                         ; do header (tam = 20 bytes)
        push DPH         ; Salvo MSB da tabela a ser lida
        push DPL         ; Salvo LSB da tabela a ser lida
        movx A,@DPTR     ; Pego o primeiro byte do header
        CPL A            ; Faço complemento de 1 do byte lido no header
        cjne R7,#00h,PROX        ; Vejo se e o primeiro byte a ser lido
        mov R2,A         ; Salvo em R2 o LSB no banco de registrador 2
        inc R7           ; Para ler o próximo byte
        pop DPL          ; Pego de volta o LSB da tabela
        pop DPH          ; Pego de volta o MSB da tabela
        inc DPTR         ; Incremento DPTR para pegar o próximo endereço

        mov A,DPL
        cjne A,#22H,LE?4         ; Vê se , o último caracter a ser lido...
        ljmp SAIR

PROX:   cjne R7,#01h,PROX1       ; Vejo se e o segundo byte a ser lido
        mov R3,A         ; Salvo em R3 o MSB no banco de registrador 2
        inc R7           ; Para ler o próximo byte
        pop DPL          ; Pego de volta o LSB da tabela
        pop DPH          ; Pego de volta o MSB da tabela
        inc DPTR         ; Incremento DPTR para pegar o próximo endereço

        cjne R6,#01h,CONT        ; se R6 != 01h, continue
        lcall SOMA16
CONT:   nop
        mov A,DPL
        cjne A,#22H,LE?4         ; Vê se , o último caracter a ser lido...
        ljmp SAIR
PROX1:  cjne R7,#02h,PROX2       ; Vejo se e o terceiro byte a ser lido
        mov R4,A         ; Salvo em R4 o LSB no banco de registrador 2
        inc R7           ; Para ler o próximo byte
        pop DPL          ; Pego de volta o LSB da tabela
        pop DPH          ; Pego de volta o MSB da tabela
        inc DPTR         ; Incremento DPTR para pegar o próximo endereço
        mov A,DPL
        cjne A,#22H,LE?4         ; Vê se , o último caracter a ser lido...
PROX2:
        mov R5,A         ; Salvo em R5 o LSB no banco de registrador 2
        inc R6           ; Para somar as duas palavras (words)
        pop DPL          ; Pego de volta o LSB da tabela
        pop DPH          ; Pego de volta o MSB da tabela
        inc DPTR         ; Incremento DPTR para pegar o próximo endereço
        lcall SOMA16

        mov A,DPL
        cjne A,#22H,LE?4         ; Vê se , o último caracter a ser lido...

SAIR:   mov R0,#PCRAM    ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1  ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0      ; Movo para DPH o MSB do endereco
        mov A,#RAM_IP    ; Inicializo o valor onde colocaremos o Checksum
        add A,@R1        ; Salvo no Acumulador o LSB do endereço
        mov DPL,A        ; Movo para DPL o LSB do endereco
```

```
        mov A,R4        ; Ponho primeiro o byte menos significativo
        movx @DPTR,A    ; Escrevo LSB no Checksum_IP
        inc DPTR        ; Inicializo o próximo endereço no qual escreveremos
        mov A,R5        ; Ponho segundo o byte mais significativo
        movx @DPTR,A    ; Escrevo MSB do Checksum_IP

        clr RS1         ; Volto no banco de registrador 0

        ret
        ENDPROC

; ********************************************************************************
; * Descrição:                                                                   *
; *     Rotina que coloca o Checksum do header ICMP no endereço de memória       *
; *     que ser  transmitido o pacote (4024h - 4025h)                            *
; * Entrada Requerida:                                                           *
; *     Nenhuma                                                                  *
; * Na Saída:                                                                    *
; *     O header icmp , colocado com o Checksum no endereço 4024h - 4035h        *
; * Afetado:                                                                     *
; *     Os registradores R2, R3, R4, R5, R6 e R7 do banco de registrador 2       *
; * Comentários:                                                                 *
; *     Nenhum                                                                   *
; ********************************************************************************
CHECK_ICMP      PROC

        setb RS1        ; Coloco no banco de registrador 2
        mov DPTR,#END_ICMP      ; Endereço do hardware (00E0h)
        mov R0,#PCRAM   ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1 ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0     ; Movo para DPH o MSB do endereço
        mov A,#PCRAM_ICMP       ; Endereço para inicializar o NIC
        add A,@R1       ; Salvo no Acumulador o LSB do endereço
        mov DPL,A       ; Movo para DPL o LSB do endereço
        mov R7,DPL      ; Salvo LSB do Endereço para Transmissão
        mov R6,DPH      ; Salvo MSB do Endereço para Transmissão

        acall TAMANHO1  ; Vejo tamanho do pacote
        mov A,R7        ; Salvo LSB do Endereço para Transmissão
        mov R1,A
        mov A,R6        ; Salvo MSB do Endereço para Transmissão
        mov R0,A
        mov R7,#00h     ; Contador de byte.
        mov R6,#00h     ; Para fazer a soma dos Checksum
LE?5:
        push DPH        ; Salvo MSB da tabela a ser lida
        push DPL        ; Salvo LSB da tabela a ser lida
        mov A,DPL
        jnz VAI
        mov A,DPH
        jz SAIR1        ; Vejo se contador = 0
VAI:
        mov DPL,R1      ; Retorno o valor LSB do PCRAM
        mov DPH,R0      ; Retorno o valor MSB do PCRAM
        movx A,@DPTR    ; Leio o valor do header em PCRAM
        inc DPTR        ; Incremento Endereço da Memória
```

```
        mov R1,DPL      ; Salvo LSB do Endereço para Transmissão
        mov R0,DPH      ; Salvo MSB do Endereço para Transmissão
        pop DPL         ; Pego de volta o MSB da tabela
        pop DPH         ; Pego de volta o LSB da tabela
        lcall UTIL_DPTRDEC

        CPL A           ; Faço complemento de 1 do byte lido no header
        cjne R7,#00h,PROX3      ; Vejo se e o primeiro byte a ser lido
        mov R2,A        ; Salvo em R2 o LSB no banco de registrador 2
        inc R7          ; Para ler o próximo byte
        ljmp LE?5       ; Leia o próximo caracter

PROX3:  cjne R7,#01h,PROX4      ; Vejo se e o segundo byte a ser lido
        mov R3,A        ; Salvo em R3 o MSB no banco de registrador 2
        inc R7          ; Para ler o próximo byte
        cjne R6,#01h,CONT1      ; se R6 != 01h, continue
        lcall SOMA16

CONT1:  ljmp LE?5       ; Vê se , o último caracter a ser lido...

PROX4:  cjne R7,#02h,PROX5      ; Vejo se e o terceiro byte a ser lido
        mov R4,A        ; Salvo em R4 o LSB no banco de registrador 2
        inc R7          ; Para ler o próximo byte
        ljmp LE?5        ; Vê se , o último caracter a ser lido...

PROX5:
        mov R5,A        ; Salvo em R5 o LSB no banco de registrador 2
        inc R6          ; Para somar as duas palavras (words)
        lcall SOMA16
        ljmp LE?5       ; Vê se , o último caracter a ser lido...

SAIR1:  pop DPL         ; Pego de volta o LSB da tabela
        pop DPH         ; Pego de volta o MSB da tabela

        mov R0,#PCRAM   ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1 ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0     ; Movo para DPH o MSB do endereço
        mov A,#RAM_ICMP ; Inicializo o valor onde colocaremos o Checksum para
                        ; o ICMP
        add A,@R1       ; Salvo no Acumulador o LSB do endereço
        mov DPL,A       ; Movo para DPL o LSB do endereço

        mov A,R4        ; Ponho primeiro o byte menos significativo
        movx @DPTR,A    ; Escrevo LSB no Checksum_ICMP
        inc DPTR        ; Inicializo o próximo endereço no qual escreveremos
        mov A,R5        ; Ponho segundo o byte mais significativo
        movx @DPTR,A    ; Escrevo MSB do Checksum_ICMP

        clr RS1         ; Volto no banco de registrador 0

        ret
        ENDPROC

; ********************************************************************************
; * Descrição:                                                                   *
; *     Rotina queávera o tamanho do header IP para calcular CRC do ICMP         *
; * Entrada Requerida:                                                           *
```

```
; *      Nenhuma                                                          *
; * Na Saída:                                                           *
; *      DPTR tera o valor do tamanho do pacote a ser lido para calcular CRC *
; * Afetado:                                                            *
; *      Nenhum, DPH -> MSB do Pacote e DPL -> LSB do Pacote            *
; * Comentários:                                                        *
; *      Nenhum                                                         *
; ********************************************************************

TAMANHO1        PROC

        mov R0,#PCRAM    ; PCRAM aponta para o pacote que ser  transmitido(PC)
        mov R1,#PCRAM+1  ; primeiro pego o MSB e depois pego o LSB
        mov DPH,@R0      ; Movo para DPH o MSB do endereço
        mov A,#TAM       ; Pega o tamanho do pacote
        add A,@R1        ; Salvo no Acumulador o LSB do endereço
        mov DPL,A        ; Movo para DPL o LSB do endereço

        movx A,@DPTR     ; Pego primeiro o MSB do Tamanho
        push Acc         ; Salvo MSB do contador
        inc DPTR
        movx A,@DPTR     ; Pego segundo o LSB do Tamanho
        mov DPL,A        ; Salvo LSB em DPL
        push DPL         ; Salvo LSB do contador

        clr A
        mov R1,#PCRAM+1  ; primeiro pego o MSB e depois pego o LSB
        mov A,#PCRAM_IP  ; Endereço deáversion_IHL
        add A,@R1        ; Salvo no Acumulador o LSB do endereço
        mov DPL,A        ; Movo para DPL o LSB do endereço

        movx A,@DPTR     ; Pego o IHL eáversion
        anl A,#0Fh       ; So nos interessa o IHL
        mov B,#04h       ; Para multiplicar por 4 bytes
        mul AB           ; Vejo tamanho do header IP para descontar no tamanho

        mov R4,A         ; Salvo valor em R4
        pop DPL          ; Retorno o valor LSB do tamanho do header
        mov A,DPL        ; Salvo em a
        clr C            ; Apago carry se existir
        subb A,R4        ; Faço Acc = Acc - R4
        mov DPL,A        ; Salvo resultado em DPL
        mov A,B          ; Pego MSB do tamanho do header IP
        mov R3,A         ; Salvo valor em R3
        pop DPH          ; Retorno o valor MSB do tamanho do header
        mov A,DPH        ; Salvo em A
        subb A,R3        ; Faço Acc = Acc - R3
        mov DPH,A        ; Salvo resultado em DPH

        ret

        ENDPROC

        END
```

```
; ************************************************************
; * Secao de rotinas relacionadas com a operação do canal    *
; * serial.                                                  *
; * Inclui rotinas inits, Send e Receive                     *
; ************************************************************


        PUBLIC serial_ini
        PUBLIC serial_snd
        PUBLIC serial_rcv

; Baud Rates

B1200   EQU  238
B2400   EQU  243
B4800   EQU  251
B9600   EQU  254

                •
RTS     BIT     P1.2
CTS     BIT     P1.3

; Rotina inits : Inicializa interface de comunicação serial
; para 8 bits e baud rate de 2400 bps.


serial_ini   PROC

        mov TH1,#B2400          ; Ajusta Timer 1 para baud rate desejada.
        mov TL1,#B2400
        setb TR1                ; Ativa Timer 1.
        mov SCON,#60H           ; Comunicação Serial no modo 1.
        setb RTS
        setb CTS
        ret

        ENDPROC

; Rotina Send  : Envia dado serial usando configuração definida
; na rotina inits
; Entrada : Acc = Caracter a ser enviado.
; Saida : Nenhuma
; Registros afetados : Acc

Serial_Snd    PROC

        clr RTS
        jb CTS,$
        mov SBUF,A              ; Envia dado para UART.
        jnb TI,$                ; Monitora flag de terminar envio.
        clr TI
        setb RTS

        ret

        ENDPROC
```

```
; Rotina Receive : Recebe um caracter através da porta serial
; Entrada : Nenhuma
; Saida : Acc = Caracter recebido
; Registradores Afetados : Acc

Serial_Rcv PROC

        setb REN
        jb CTS,$
        clr RTS
        jnb RI,$                ; Espera recepção de caracter terminar.
        mov A,SBUF              ; Pega caracter da UART.
        clr REN                 ; Desabilita modo de recepção.
        clr RI
        setb RTS
        ret

        ENDPROC

        END
```

```
; ************************************************************************
; * Descrição:                                                           *
; *     Macro para a Inicialização dos registradores internos do         *
; *     controlador NIC DP8390. Esta macro dever  ser chamada sempre para*
; *     a inicialização dos registradores                                *
; * Entrada Requerida:                                                   *
; *     Nenhuma                                                          *
; * Na Saída:                                                            *
; *     Nada                                                             *
; * Afetado:                                                             *
; *     Nenhum registrador interno de PC                                 *
; * Comentários:                                                         *
; *     Nenhum                                                           *
; ************************************************************************

; Esta é a macro 8390.inc que é incluída na maioria das outras rotinas.
; Pseudo variáveis para a Inicialização da Placa Ethernet

RA0     BIT     P1.4    ; Endereço do pino P1.4, usado como Registrador de
                        ; Endereço RA0 do NIC
RA1     BIT     P1.5    ; Endereço do pino P1.5, usado como Registrador de
                        ; Endereço RA1 do NIC
RA2     BIT     P1.6    ; Endereço do pino P1.6, usado como Registrador de
                        ; Endereço RA2 do NIC
RA3     BIT     P1.7    ; Endereço do pino P1.7, usado como Registrador de
                        ; Endereço RA3 do NIC
REGISTRADOR     EQU     0A000H   ; Comando dos Registradores (00h)

COMMAND %macro          ; Comando dos Registradores (00h)
        MOV DPTR,#REGISTRADOR
        clr RA0
        clr RA1
        clr RA2
        clr RA3
        %endm

PAGESTART       %macro  ; Início da Mensagem (01h)
        MOV DPTR,#REGISTRADOR
        setb RA0
        clr RA1
        clr RA2
        clr RA3
                %endm

REGISTERFISIC0  %macro  ; Registrador Fisico 0 - PAR0 (01h)
        MOV DPTR,#REGISTRADOR
        setb RA0
        clr RA1
        clr RA2
        clr RA3
                %endm

PAGESTOP        %macro  ; Final da Mensagem (02h)
        MOV DPTR,#REGISTRADOR
        clr RA0
        setb RA1
```

```
        clr RA2
        clr RA3
                %endm

REGISTERFISIC1  %macro  ; Registrador Fisico 1 - PAR1 (02h)
        MOV DPTR,#REGISTRADOR
        clr RA0
        setb RA1
        clr RA2
        clr RA3
                %endm

BOUNDARY        %macro  ; Ponteiro Limite (Leitura Remota) (03h)
        MOV DPTR,#REGISTRADOR
        setb RA0
        setb RA1
        clr RA2
        clr RA3
                %endm

REGISTERFISIC2  %macro  ; Registrador Fisico 2 - PAR2 (03h)
        MOV DPTR,#REGISTRADOR
        setb RA0
        setb RA1
        clr RA2
        clr RA3
                %endm

TRANSMITSTATUS  %macro  ; Estado da Transmissão (04h)
        MOV DPTR,#REGISTRADOR
        clr RA0
        clr RA1
        setb RA2
        clr RA3
                %endm

TRANSMITPAGE    %macro  ; Mensagem a ser Transmitida (Aponta p/ o pacote a ser
                        ; transmitido (04h)
        MOV DPTR,#REGISTRADOR
        clr RA0
        clr RA1
        setb RA2
        clr RA3
                %endm

REGISTERFISIC3  %macro  ; Registrador Fisico 3 - PAR3 (04h)
        MOV DPTR,#REGISTRADOR
        clr RA0
        clr RA1
        setb RA2
        clr RA3
                %endm

TRANSMITBYTECOUNT0      %macro  ; Contador de Bytes Inferior para a
                                ; Transmissão (05h)
        MOV DPTR,#REGISTRADOR
        setb RA0
```

```
        clr RA1
        setb RA2
        clr RA3
                %endm

NCR     %macro  ; Número de Colisões (05h)
        MOV DPTR,#REGISTRADOR
        setb RA0
        clr RA1
        setb RA2
        clr RA3
                %endm

REGISTERFISIC4  %macro  ; Registrador Fisico 4 - PAR4 (05h)
        MOV DPTR,#REGISTRADOR
        setb RA0
        clr RA1
        setb RA2
        clr RA3
                %endm

TRANSMITBYTECOUNT1      %macro  ; Contador de Bytes Superior para a
                                ; Transmissão (06h)
        MOV DPTR,#REGISTRADOR
        clr RA0
        setb RA1
        setb RA2
        clr RA3
                %endm

REGISTERFISIC5  %macro  ; Registrador Fisico 5 - PAR5 (06h)
        MOV DPTR,#REGISTRADOR
        clr RA0
        setb RA1
        setb RA2
        clr RA3
                %endm

INTERRUPTSTATUS %macro  ; Estado de Interrupção (07h)
        MOV DPTR,#REGISTRADOR
        setb RA0
        setb RA1
        setb RA2
        clr RA3
                %endm

CURRENT %macro  ; Ponteiro Corrente (Escrita Remota no page 1) (07h)
        MOV DPTR,#REGISTRADOR
        setb RA0
        setb RA1
        setb RA2
        clr RA3
                %endm

REMOTESTARTADDRESS0     %macro  ; Endereço Inicial Remoto Inferior (aponta p/
                                ; o início do bloco de dados a ser
                                ; transmitido (08h)
```

```
        MOV DPTR,#REGISTRADOR
        clr RA0
        clr RA1
        clr RA2
        setb RA3
                %endm

CRDMA0  %macro  ; Endereço DMA Remoto Corrente Inferior (08h)
        MOV DPTR,#REGISTRADOR
        clr RA0
        clr RA1
        clr RA2
        setb RA3
                %endm

REMOTESTARTADDRESS1     %macro  ; Endereço Inicial Remoto Superior (aponta p/
                                ; o início do bloco de dados a ser
                                ; transmitido (09h)
        MOV DPTR,#REGISTRADOR
        setb RA0
        clr RA1
        clr RA2
        setb RA3
                %endm

CRDA1   %macro  ; Endereço DMA Remoto Corrente Superior (09h)
        MOV DPTR,#REGISTRADOR
        setb RA0
        clr RA1
        clr RA2
        setb RA3
                %endm

REMOTEBYTECOUNT0        %macro  ; Contador de Bytes Remoto Inferior (, o
                                ; tamanho do bloco de dados a ser
                                ; transmitido (0Ah)
        MOV DPTR,#REGISTRADOR
        clr RA0
        setb RA1
        clr RA2
        setb RA3
                %endm

REMOTEBYTECOUNT1        %macro  ; Contador de Bytes Remoto Superior (, o
                                ; tamanho do bloco de dados a ser
                                ; transmitido (0Bh)
        MOV DPTR,#REGISTRADOR
        setb RA0
        setb RA1
        clr RA2
        setb RA3
                %endm

RECEIVESTATUS   %macro  ; Estado da Recepção (0Ch)
        MOV DPTR,#REGISTRADOR
        clr RA0
        clr RA1
```

```
        setb RA2
        setb RA3
                %endm

RECEIVECONFIGURATION    %macro  ; Configuração da Recepção (0Ch)
        MOV DPTR,#REGISTRADOR
        clr RA0
        clr RA1
        setb RA2
        setb RA3
                %endm

TRANSMITCONFIGURATION   %macro  ; Configuração da Transmissão (0Dh)
        MOV DPTR,#REGISTRADOR
        setb RA0
        clr RA1
        setb RA2
        setb RA3
                %endm
                    •
FAE_TALLY       %macro  ; Contador de Erros de Alinhamento do Frame (0Dh)
        MOV DPTR,#REGISTRADOR
        setb RA0
        clr RA1
        setb RA2
        setb RA3
                %endm

DATACONFIGURATION       %macro  ; Configuração dos Dados (0Eh)
        MOV DPTR,#REGISTRADOR .
        clr RA0
        setb RA1
        setb RA2
        setb RA3
                %endm

CRC_TALLY       %macro  ; Contador de Erros de CRC (0Eh)
        MOV DPTR,#REGISTRADOR
        clr RA0
        setb RA1
        setb RA2
        setb RA3
                %endm

INTERRUPTMASK   %macro  ; Máscara para a Interrupção (0Fh)
        MOV DPTR,#REGISTRADOR
        setb RA0
        setb RA1
        setb RA2
        setb RA3
                %endm

MISS_PKT_TALLY   %macro ; Contador de Pacotes Perdidos ou danificados (0Fh)
        MOV DPTR,#REGISTRADOR
        setb RA0
        setb RA1
        setb RA2
```

```
        setb RA3
                %endm

; Compara o Acumulador com o dado, e pula se forem iguais
cjeq    %macro  parm1,parm2,parm3
      %gensym     temp
        cjne    parm1,parm2,temp
        jmp       parm3
temp
        %endm
```

Apêndice D

"Resultado I : (a) Dados enviados e (b) dados recebidos"

External Data Memory <1>

File Options View Data

4000	00	C0	DF	E3	58	7C	08	00	17	01	09	74	06	00	45	00	. . . . Z | . . . . . t . . E .
4010	00	54	9E	00	00	00	20	01	05	3D	8F	6B	EB	C9	6F	6B	. T . . . . . = . k . . . k
4020	EB	CC	08	00	29	46	01	00	01	00	31	39	2E	37	30	09	. . . .) F . . . . 1 9 . 7 0 .
4030	37	2E	36	30	09	32	36	2E	33	37	09	31	32	33	37	2E	7 . 6 0 . 2 6 . 3 7 . 1 2 3 7 .
4040	38	39	09	33	2E	35	32	09	2D	30	2E	30	33	09	31	35	8 9 . 3 . 5 2 . - 0 . 0 3 . 1 5
4050	3A	30	34	3A	33	35	20	31	38	2F	30	35	2F	30	30	09	: 0 4 : 3 5 1 8 / 0 5 / 0 0 .
4060	31	0A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	1 . . . . .
4070	41	7E	08	E2	42	FF	00	44	00	00	00	00	00	00	00	00	A ~ . . B . . D . . . . .
4080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
40A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
40B0	00	00	00	00	00	00	00	00	00	10	00	00	00	00	00	00	. . . . .
40C0	00	00	00	00	00	00	00	00	00	00	00	40	00	00	00	00	. . . . . @ . . .
40D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
40E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
40F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4130	00	01	00	00	00	00	00	00	00	20	00	00	00	00	00	00	. . . . .
4140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4170	00	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	. . . . . @ . . . . .
4180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
41A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
41B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
41C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
41D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
41E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
41F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .

Temperatura (°C)
pH
Quantidade de Oxigênio Dissolvido (mg/l)
Condutância (µS/cm)
Turbidez (NTU)
Profundidade (m)

(a) Dados medidos pela sonda e enviados como um pacote de dados dentro do protocolo ICMP

External Data Memory <1>

File Options View Data

```
4600  01 27 66 00 08 00 17 01 09 74 00 C0 DF E3 5A 7C  . ' f . . . . . t . . . . Z |
4610  08 00 45 00 00 54 9E 00 00 00 20 01 05 3D 8F 6B  . . E . . T . . . . . . = . k
4620  EB CC 8F 6B EB C9 00 00 31 46 01 00 01 00 31 39  . . . k . . . . 1 F . . . 1 9
4630  2E 37 30 09 37 2E 36 30 09 32 36 2E 33 37 09 31  . 7 0 . 7 . 6 0 . 2 6 . 3 7 . 1
4640  32 33 37 2E 38 39 09 33 2E 35 32 09 2D 30 2E 30  2 3 7 . 8 9 . 3 . 5 2 . - 0 . 0
4650  33 09 31 35 3A 30 34 3A 33 35 20 31 38 2F 30 35  3 . 1 5 : 0 4 : 3 5   1 8 / 0 5
4660  2F 30 30 09 31 0A 00 00 00 00 00 00 00 00 00 00  / 0 0 . 1 . . . . . .
4670  00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00  . . . . . .
4680  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
4690  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
46A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
46B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
46C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
46D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
46E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
46F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
4700  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
4710  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
4720  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
4730  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
4740  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
4750  00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00  . . . . . .
4760  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
4770  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
4780  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
4790  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
47A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
47B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
47C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
47D0  80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
47E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
47F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
4800  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
4810  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  . . . . . .
```

(b) Echo reply enviado do computador do laboratório para a interface Cheapernet

Apêndice E

"Resultado II : (a) Dados enviados e (b) dados recebidos"

External Data Memory <1>

File Options View Data

4000	00	C0	DF	E3	5A	7C	08	00	17	07	09	74	08	00	45	00	. . . . Z | . . . . . t . . E .
4010	00	54	3E	00	00	00	20	01	05	3D	0F	6B	ED	C0	0F	6B	. T . . . . . = . k . . . k
4020	EB	CC	08	00	33	49	01	00	02	00	31	39	2E	36	30	09	. . . . 3 I . . . . 1 9 . 6 0 .
4030	37	2E	36	31	09	32	36	2E	34	30	09	31	32	34	30	2E	7 . 6 1 . 2 6 . 4 0 . 1 2 4 0 .
4040	35	37	09	33	2E	35	32	09	2D	30	2E	30	33	09	31	35	5 7 . 3 . 5 2 . - 0 . 0 3 . 1 5
4050	3A	31	31	3A	34	39	20	31	38	2F	30	35	2F	30	30	09	: 1 1 : 4 9 1 8 / 0 5 / 0 0 .
4060	31	0A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	1 . . . . .
4070	41	7E	08	E2	42	FF	00	44	00	00	00	00	00	00	00	00	A ~ . . B . . D . . . . .
4080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
40A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
40B0	00	00	00	00	00	00	00	00	00	10	00	00	00	00	00	00	. . . . .
40C0	00	00	00	00	00	00	00	00	00	00	00	40	00	00	00	00	. . . . . @ . . . .
40D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
40E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
40F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4130	00	01	00	00	00	00	00	00	00	20	00	00	00	00	00	00	. . . . .
4140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4170	00	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	. . . . . @ . . . . .
4180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
41A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
41B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
41C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
41D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
41E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
41F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	. . . . .
4220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

(a) Dados medidos pela sonda e enviados como um pacote de dados dentro do protocolo ICMP

External Data Memory <1>

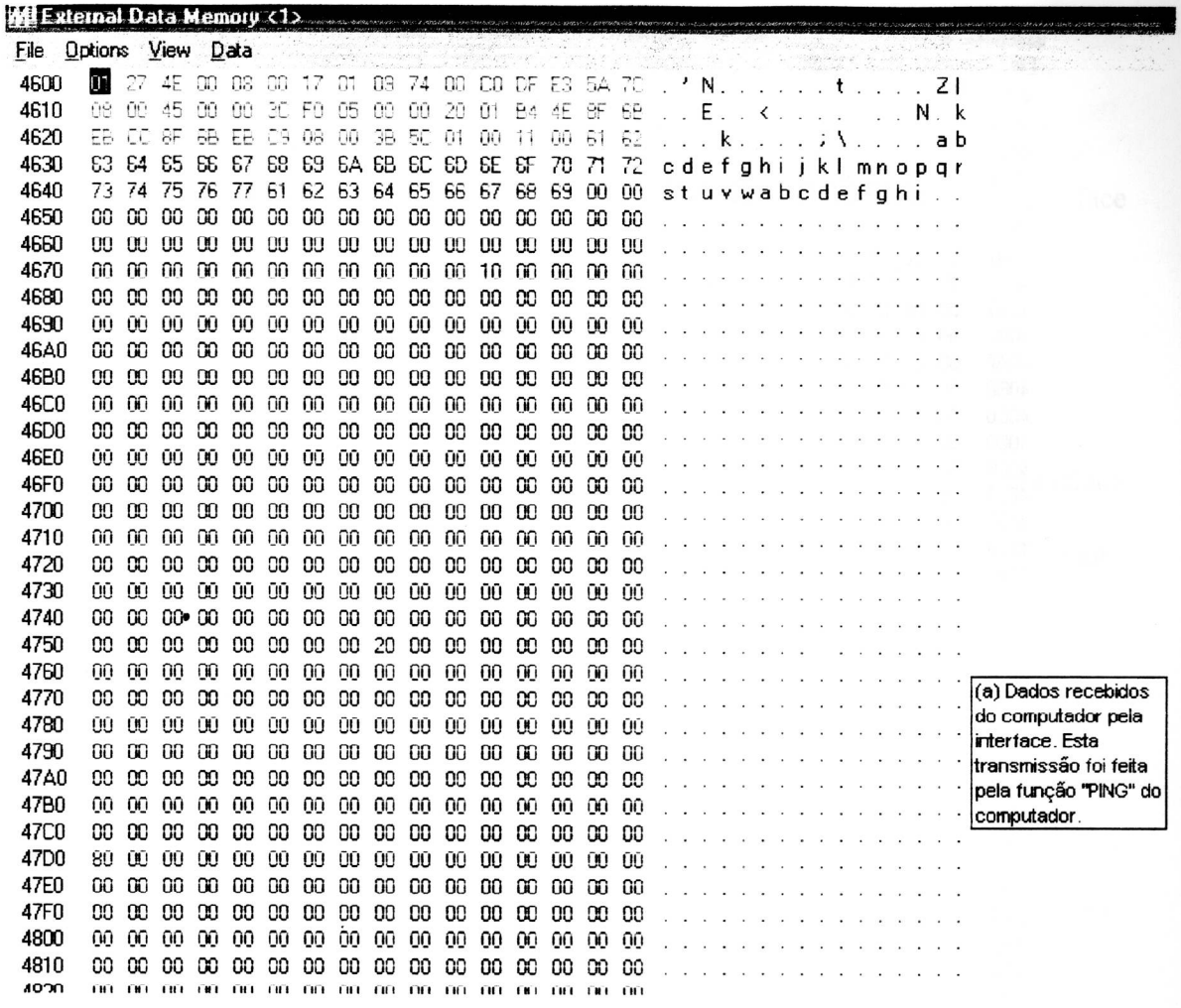
File Options View Data

4600	01	27	66	00	06	00	17	01	05	74	00	00	DF	E3	54	7C	.'f.....t....Z|
4610	08	00	45	00	00	54	9E	00	00	00	20	01	05	3D	9F	88	..E..T..... =.k
4620	EB	CC	8F	6B	EB	C9	00	00	3B	49	01	00	02	00	31	39	...k....;I....19
4630	2E	36	30	09	37	2E	36	31	09	32	36	2E	34	30	09	31	.60.7.61.26.40.1
4640	32	34	30	2E	35	37	09	33	2E	35	32	09	2D	30	2E	30	240.57.3.52.-0.0
4650	33	09	31	35	3A	31	31	3A	34	39	20	31	38	2F	30	35	3.15:11:49 18/05
4660	2F	30	30	09	31	0A	00	00	00	00	00	00	00	00	00	00	/00.1..........
4670	00	00	00	00	00	00	00	00	00	00	00	10	00	00	00	00	....
4680	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4690	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
46A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
46B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
46C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
46D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
46E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
46F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4700	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4710	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4720	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4730	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4740	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4750	00	00	00	00	00	00	00	00	00	20	00	00	00	00	00	00	....
4760	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4770	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4780	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4790	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
47A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
47B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
47C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
47D0	80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
47E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
47F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4800	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4810	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
4820	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

(b) Echo reply enviado do computador do laboratório para a interface Cheapernet

Apêndice F

"Resultado III : (a) Dados recebidos e (b) dados enviados"



(a) Dados recebidos do computador pela interface. Esta transmissão foi feita pela função "PING" do computador.

External Data Memory <1>

File Options View Data

4000	00	C8	DF	E3	5A	7C	88	00	17	01	00	74	08	00	45	00
4010	00	3C	F0	05	00	00	20	01	B4	4E	9F	8B	EB	C9	9F	8B
4020	EB	CC	00	00	43	5C	01	00	11	00	61	62	63	64	65	66
4030	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76
4040	77	61	62	63	64	65	66	67	68	69	00	00	00	00	00	00
4050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4070	43	74	03	E2	00	FF	00	44	00	00	00	00	00	00	00	00
4080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40B0	00	00	00	00	00	00	00	00	00	18	00	00	00	00	00	00
40C0	00	00	00	00	00	00	00	00	00	00	00	40	00	00	00	00
40D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4130	01	01	00	00	00	00	00	00	00	20	00	00	00	00	00	00
4140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4170	00	00	00	00	00	00	00	00	00	40	00	00	00	00	00	00
4180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
41A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
41B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
41C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
41D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
41E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
41F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

(b) Dados enviados para o computador pela interface para complementar a função "PING" (echo reply).

Bibliografia:

[1] NATIONAL – "DP8390D/NS32490D NIC Network Interface Controller" – Documento data sheet em formato pdf

http://www.national.com/pf/DP/DP8390D.html

[2] NATIONAL – "DP83902A ST-NIC Serial Network Interface Controller for Twisted Pair" – Documento data sheet em formato

- pdf

http://www.national.com/pf/DP/DP83902A.html

[3] NATIONAL – "The Operation of the FIFO in the DP8390, DP83901, DP83902 and DP83905" – Documento application notes em formato pdf

http://www.national.com/an/AN/AN-886.pdf

[4] NATIONAL – "The Design and Operation of a Low Cost, 8-Bit PC-XT Compatible Ethernet Adapter Using the DP83902" – Documento application notes em formato pdf

http://www.national.com/an/AN/AN-842.pdf

[5] NATIONAL – "Architectural Choices for Ethernet" – Documento application notes em formato pdf

http://www.national.com/an/AN/AN-845.pdf

[6] NATIONAL – "DP8390 Network Interface Controller: An Introductory Guide" – Documento application notes em formato pdf

http://www.national.com/an/AN/AN-475.pdf

[7] NATIONAL – "Guide to Loopback Using the DP8390 Chip Set" – Documento application notes em formato pdf

http://www.national.com/an/AN/AN-858.pdf

[8] NATIONAL – "DP83956EB-AT LERIC (LitE Repeater Interface Controller) PC-AT Adapter" – Documento application notes em formato pdf

http://www.national.com/an/AN/AN-854.pdf

[9[NATIONAL – "DP8391A SNI Serial Network Interface" – Documento data sheet em formato pdf

http://www.national.com/pf/DP/DP8391A.html

[10] NATIONAL – "Ethernet/Cheapernet Physical Layer Made Easy with DP8391/92" – Documento application notes em formato pdf

http://www.national.com/an/AN/AN-442.pdf

[11] NATIONAL – "CMOS SNI Serial Network Interface" – Documento application notes em formato pdf

http://www.national.com/pf/DP/DP83910A.html

[12] NATIONAL – "Low Power Ethernet with the CMOS DP83910 Serial Network Interface" – Documento application notes em formato pdf

http://www.national.com/an/AN/AN-622.pdf

[13] NATIONAL – "DP8392C/DP8392C1 CTI Coaxial Transceiver Interface" – Documento data sheet em formato pdf

http://www.national.com/pf/DP/DP8392C.html

[14] NATIONAL – "DP83932EB-EISA SONIC/EISA Packet Driver for PC/TCP" – Documento application notes em formato pdf

http://www.national.com/an/AN/AN-859.pdf

[15] NATIONAL – "DP839EB-ATS SONIC Packet Driver for PC/TCP by FTP Software" – Documento application notes em formato pdf

http://www.national.com/an/AN/AN-748.pdf

[16] NATIONAL – "Software Driver Programmer's Guide for the DP83932 SONIC" – Documento application notes em formato pdf

http://www.national.com/an/AN/AN-746.pdf

[17] POSTEL, J. - (ed.), "Internet Protocol - DARPA Internet Program Protocol Specification", RFC 791, USC/Information Sciences Institute, September 1981.

http://andrew2.andrew.cmu.edu/rfc/rfc791.html

[18] POSTEL, J. – "Assigned Numbers", RFC 1700, USC/Information Sciences - Institute, September 1981.

http://www.freesoft.org/CIE/RFC/1700/index.htm

[19] POSTEL, J. – "Internet Control Message Protocol - DARPA Internet Program Protocol Specification", RFC 792, USC/Information Sciences Institute, September 1981.

http://andrew2.andrew.cmu.edu/rfc/rfc792.html

[20] D-I-X – "The Ethernet - A Local Area Network: Data Link Layer and Physical Layer Specifications", Digital, Intel, and Xerox, November 1982.

[21] SPURGEON, CHARLES – "Guide to Ethernet Configuration" – Networking Services – University of Texas at Austin.

[22] FAQ (FREQUENT ASKED QUESTION)

ftp://ftp.ee.ualberta.ca/pub/cookbook/faq/lcd.doc

http://www.paranoia.com/~filipg/HTML/LINK/F_LCD_tech.html

[23] BRUCK, SCOTT M. – "LCDFAQ: (Liquid Crystal Display physics & principles of operation)"; August 1993.

ftp://ftp.ee.ualberta.ca/pub/cookbook/faq/LCD2.doc

[24] NICOL, JORDAN – "4-bit interface sample in-line assembly code, for implementation under Dunfield's Micro-C for the Miniboard".

ftp://cher.media.mit.edu

[25] PARALLAX BASIC STAMP APPLICATIONS.

ftp://wpi.wpi.edu/stamp

ftp://ftp.parallaxinc.com/pub/stamp

[26] PACKET DRIVER – "Rotinas em Linguagem Montadora (Assembly - 8086) para uma placa rede local"

http://www.simtel.net/simtel.net/msdos/pktdrvr.html

[27] PHILIPS MICROCONTROLLER ELECTRONIC NEWS LETTER – "Catálogo sobre os microcontroladores 8051 e 8031"

http://www.philipsmcu.com/News15.html

[28] DA SILVA JÚNIOR, VIDAL PEREIRA – "Aplicações Práticas do Microcontrolador 8051", 4ª Edição 1994, São Paulo - Editora Érica.

[29] STWART, JAMES W. – "The 8051 Microcontroller : Hardware, Software and Interfacing"; 1ª Edição 1993 New Jersey by Regents/Prentice Hall

[30] BSD Sockets: "A Quick And Dirty Primer"

http://sci173x.mrs.umn.edu/~bentlema/unix/sockets.html

[31] NEWMARCH, JAN – "Client-Server Computing"

http://pandonia.canberra.edu.au/ClientServer/

[32] HEDRICK, CHARLES L. – "Intro to TCP/IP" – The State University of New Jersey

gopher://gopher-chem.ucdavis.edu/11/Index/Internet_aw/Intro_the_Internet/intro.to.ip/

[33] The Unix Socket FAQ – Questões Perguntadas Frequentemente

http://www.auroraonline.com/sock-faq/

[34] RFC-768 - The User Datagram Protocol (UDP)

ftp://nic.ddn.mil/rfc/rfc768.txt

[35] RFC-791 - The Internet Protocol (IP)

ftp://nic.ddn.mil/rfc/rfc791.txt

[36] RFC-793 - The Transmission Control Protocol (TCP)

ftp://nic.ddn.mil/rfc/rfc793.txt

[37] CISCO DOCUMENTATION – "Internetworking Basics"

http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/introint.htm

[38] RONALDO, BRUNO – "Pesquisa em Instrumentação Microprocessada para a medida de parâmetros físicos e químicos da água" – Dissertação de mestrado realizada no IFSC, 1997, 145p.