ESTAÇÃO REMOTA CONTROLADA POR MICROPROCESSADOR

RICARDO DE OLIVEIRA ANIDO

ORIENTADOR

Prof.Dr. Nelson Castro Machado

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Julho 1983

à Monique

AGRADECIMENTOS

Ao Prof.Dr. Nelson Castro Machado, pela orientação e estímulo.

Ao pessoal do Lab-Micro do Centro de Computação, em especial a A.Dolenc e J.L.Silveira, pelo apoio em "software" e "hardware" básicos.

Ao CNPq, pelo suporte financeiro concedido.

ABSTRACT

Remote Job Entry Stations (RJEs) commonly avaiable for the DEC-10 system are implemented using relatively expensive mi ni-computers, such as PDP-8 and PDP-11. This paper presents the specification and implementation of a RJE functionally equivalent to the comercial system DC72NP.

The proposed system supports a console, a printer and up to seven asyncronous terminals, and its advantage is the use of a low cost microprocessor, the INTEL 8088.

Included are a description of the hardware, the RJE's operating system and the communication protocol,which is defined as a multi-layered system of four levels: physical layer, link layer (DDCMP), network layer (NCL) and application layer.

RESUMO

As estações remotas disponíveis para o sistema DEC-1Ø, até o presente, são implementadas em computadores de pequeno e médio porte, atreladas a pacotes fechados de "software", a um custo bastante alto.

O presente trabalho consiste no estudo da estrutura de comunicação do sistema DEC-1Ø com suas estações remotas tipo DC72NP, e a especificação e desenvolvimento de uma estação remota funcionalmente equivalente, mas utilizando um microprocessador de baixo custo (INTEL 8088). Está incluída a descrição do protocolo a quatro níveis usado na comunicação DEC-1Ø - estação remota: nível físico, nível de enlace (DDCMP), nível de rede (NCL) e nível de aplicação.

São ainda descritos a arquitetura do "hardware" construído (capaz de suportar console, impressora e até sete terminais assíncronos) e a estrutura modular do "software desenvolvido.

SUMÁRIO

CAPITULO 1

INTRODUÇÃO

É cada vez mais frequente a utilização de sistemas de médio e grande porte através de terminais em "time-sharing". Os sistemas mais sofisticados podem suportar dezenas ou centenas de terminais geograficamente situados em uma área bastante extensa.

Quando é necessária a utilização de diversos terminais em um único local (o prédio de uma faculdade, por exemplo), estando o computador situado em outro campus ou mesmo em outra cidade, é economicamente inviável utilizar uma linha dedicada para cada terminal. A solução frequentemente adotada é utilizar uma única linha de alta velocidade entre o computador e um "concentrador de terminais", que suporta de um lado um sofisticado protocolo de multiplexagem com o computador e do outro comunicações simples com os diversos terminais. Se além de terminais o concentrador inclui também outros equipamentos, como leitora de cartões ou impressora, passa a ser denominado uma "estação remota".

Até o presente, a maioria das estações remotas disponíveis são desenvolvidas pelos fabricantes de equipamentos de processamento de dados, específicas a cada modelo e marca de computadores. Tipicamente estas estações remotas são implementadas em computadores de pequeno ou médio porte, atreladas a pacotes fechados de "software", o que aumenta bastante o custo do sistema.

A introdução de pastilhas ("chips") sofisticadas para controle de teleprocessamento no repertório de periféricos de alguns microprocessadores tornou possível a implementação de uma estação remota bem mais simples e econômica, controlada por micropocessador, com "software"contido em EPROM (memória de leitura apenas).

O presente trabalho compreende o estudo do protocolo de comunicação síncrona e da estrutura da estação remota utilizada no sistema de computação da Unicamp e o desenvolvimento de uma estação remota funcionalmente equivalente, mas utilizando um microprocessador de baixo custo.

O sistema de computação da Unicamp tem um DEC1Ø como computador central e um PDP8E como estação remota (tipo DC72NP) produzidos pela Digital Equipment Corporation (DEC). Esta firma desenvolveu um conjunto de programas e protocolos, a que chamou de DECnet, para uso em seus sistemas de computadores, possibilitando a seus usuários a criação de suas próprias redes de computadores. A arquitetura da DECnet é chamada DNA (Digital Network Architecture). [WECKER, 80]

No sistema DEC1Ø cada estação remota é considerada um nó de uma rede e portanto toda a comunicação computador-estação remota é regida pela DECnet. A parte final deste capítulo é dedicada à descrição geral da DECnet.

O restante do trabalho está assim esquematizado: o capítulo 2 descreve os protocolos dos diversos níveis envolvidos na comunicação DEC1Ø -Estação Remota. O capítulo 3 está dividido em duas partes, sendo a primeira dedicada à apresentação do

"hardware" e a segunda à descrição do "software" desenvolvidos para a estação remota. Finalmente, no capítulo 4 são apresentados e comentados os resultados dos testes efetuados com um protótipo da estação remota montado em laboratório.

1.1. Introdução à DECnet

A DECnet é organizada em uma série de cinco níveis, cada um construído sobre o seu predecessor. A função de cada nível é fornecer certos serviços aos níveis mais altos, tornando transparente a estes os detalhes de como esses serviços são realmente implementados. Os níveis definidos na DECnet são: nível físico, nível de enlace, nível de transporte, nível de rede e nível de aplicação.

1.1.1 O Nível Físico (Physical Layer)

É o responsável pela transmissão de bits através de um canal de comunicação. Parâmetros típicos são quantos volts (ou outra grandeza) devem ser usados para representar um bit 1 e quantos para um bit Ø, a taxa (velocidade) de transmissão e se a transmissão pode ser feita simultaneamente (full-duplex) ou se um lado deve esperar o outro terminar de transmitir para iniciar sua transmissão (half-duplex).

1.1.2 O Nível de Enlace (Data Link Control Layer)

A tarefa deste nível é transformar o serviço de transmissão de bits fornecido pelo nível 1 em um canal que apareça para o nível seguinte (de transporte) como praticamente isento

de erros. Ele executa esta tarefa quebrando os dados de entrada em "quadros de dados" (que incluem uma informação sobre a integridade dos dados), transmitindo esses quadros sequencialmente e processando os "quadros de reconhecimento" enviados em resposta, que indicam se foi detectado erro em algum quadro de dados. Neste caso o nível de enlace se encarrega de corrigir o erro através de um procedimento específico (retransmissão).

Como o nível 1 apenas aceita e transmite uma sequência de bits sem se importar com seu significado ou estrutura, é o nível 2 o responsável pela criação e identificação dos limites dos quadros.

O nível de enlace é regido pelo protocolo DDCMP (Digital Data Communication Protocol).

1.1.3 O Nível de Transporte (Transport Layer)

É o nível que controla a operação do tráfego na rede. Entre outras tarefas, determina qual será a rota dos pacotes (que são as unidades de informação trocadas neste nível) através da rede.

Usando o nível 2 para transmissão de pacotes através de canais entre nós contíguos, o nível de transporte cria um caminho entre o nó fonte e o nó destino. Esse caminho é construído nó a nó, com base em uma tabela de rota presente em cada nó, de maneira a escolher a melhor trajetória para o pacote.Cada pacote é tratado individualmente e o algorítmo de rota determina se deve ser utilizado o mesmo caminho usado para o pacote precedente ou se deve haver mudança de rota devido a problemas específicos, como congestionamento, comando do operador, falha

de algum canal, etc.

O nível de transporte procura entregar ao nó destino todos os pacotes que lhe são apresentados, mas não garante o recebimento, a sequência ou a destruição dos pacotes não entregues após um determinado tempo. Portanto, os níveis mais altos devem usar o mecanismo de retransmissão para recuperação e descartar as possíveis duplicatas geradas por essas retransmissões.

1.1.4 O Nível de Rede (Network Layer)

Este nível tem a função de criar e gerenciar "canais lógicos" que permitam a transferência de mensagens entre um processo do nó fonte e um processo do nó destino. Um canal lógico é definido como sendo um canal de comunicação sequencial,"full-duplex", byte-orientado (isto é, a unidade de informação é constituída de um número inteiro de bytes de oito bits).

Uma vez que um canal lógico é estabelecido, ambos os processos podem simultaneamente enviar e receber mensagens através do canal. O nível de rede é o responsável pela sequência das mensagens, garantindo que não haja perda ou duplicação em cada canal lógico. Qualquer mudança na rota dos pacotes através da rede, devido a, por exemplo, falha do canal físico ou nó intermediário, é transparente aos usuários do nível de rede. Este nível é regido pelo protocolo NSP (Network Services Protocol).

1.1.5 O Nível de Aplicação (Application Layer)

É neste nível que são executados os programas de aplicação dos usuários, programas de gerência de recursos, etc. A troca de dados neste nível é, por exemplo, entre programas de

aplicação e equipamentos de E/S ou sistemas de arquivos. Vários programas podem ser executados simultaneamente, cada um comunicando-se através de canais lógicos independentes.Programas de propósito geral, parte dos serviços oferecidos pela DECnet, são executados neste nível, incluindo carregamento remoto de sistemas, programas de acesso e transferência de arquivos, programas de manutenção e outros.

As funções executadas pelos níveis 1 a 4 são parte do padrão DECnet. O nível 5 é o único em que o "software" provido pela DEC e o produzido pelo usuário coexistem. Diferentes produtos incluem diferentes protocolos neste nível.

CAPITULO 2

A ESTAÇÃO REMOTA DC72NP NA DECnet

Uma estação remota DC72NP é um tipo de nó particular da DECnet, pois seus processos comunicam-se primariamente com o computador ao qual está ligada, denominado computador hospedeiro. Além disso, esse tipo de estação é um nó terminal (Fig.2.1) ou seja, tem apenas uma linha de comunicação ligada à rede.

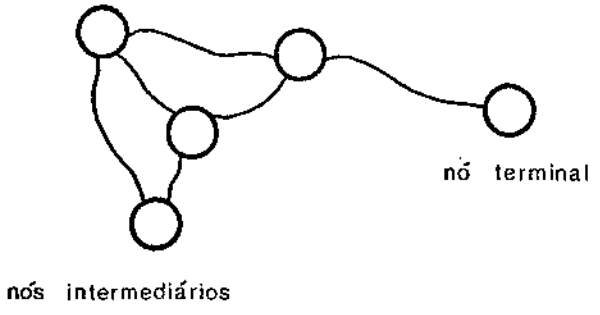


Fig.2.1. Nós intermediários e nó terminal

Ao contrário dos nós intermediários (que têm mais de uma linha de comunicação ligada à rede), o nó terminal não executa o roteamento de mensagens. Essa tarefa é responsabilidade do nó intermediário que mantém a linha de comunicação com o nó terminal. Desta forma, o nível de transporte de um nó terminal é bastante simplificado e assim,para os propósitos deste trabalho o nível de rede passa, a partir de agora, a compreender também o nível de transporte.

A Fig.2.2 mostra o fluxo de dados da comunicação estação-hospedeiro, onde processo fonte e processo destino se referem, respectivamente, ao processo que está enviando os dados e ao processo ao qual os dados são dirigidos. Podem ser, por exemplo, o processo controlador de um terminal e o seu correspondente no hospedeiro (ou vice-versa). Um processo é caracterizado por três parâmetros: seu código, sua área de dados e sua ativação.

Um canal lógico é inerentemente "full-duplex": ambos os processos podem estar transmitindo e recebendo dados simultâneamente. No exemplo da Fig.2.2, no entanto, é mostrado o fluxo de dados em um sentido apenas. Assume-se, ainda, que já foi criado um canal lógico entre os dois processos e que o processo destino já enviou ao processo fonte um pedido de dados e está no momento à espera de dados.

Neste capítulo são apresentados as funções e os protocolos dos níveis da DECnet utilizados na comunicação estação-hospedeiro: nível físico, nível de enlace, nível de rede e nível de aplicação.

2.1 Nível Físico

É constituído pela linha física por onde é feita a comunicação, modem , cabos e conectores, fornecendo um "canal físico". A sua função é permitir a comunicação síncrona entre a estação e o hospedeiro, operando em modo "full-duplex" (transmissores e receptores de ambos os lados operando simultaneamente). É o responsável pela sincronização a nível de bit.

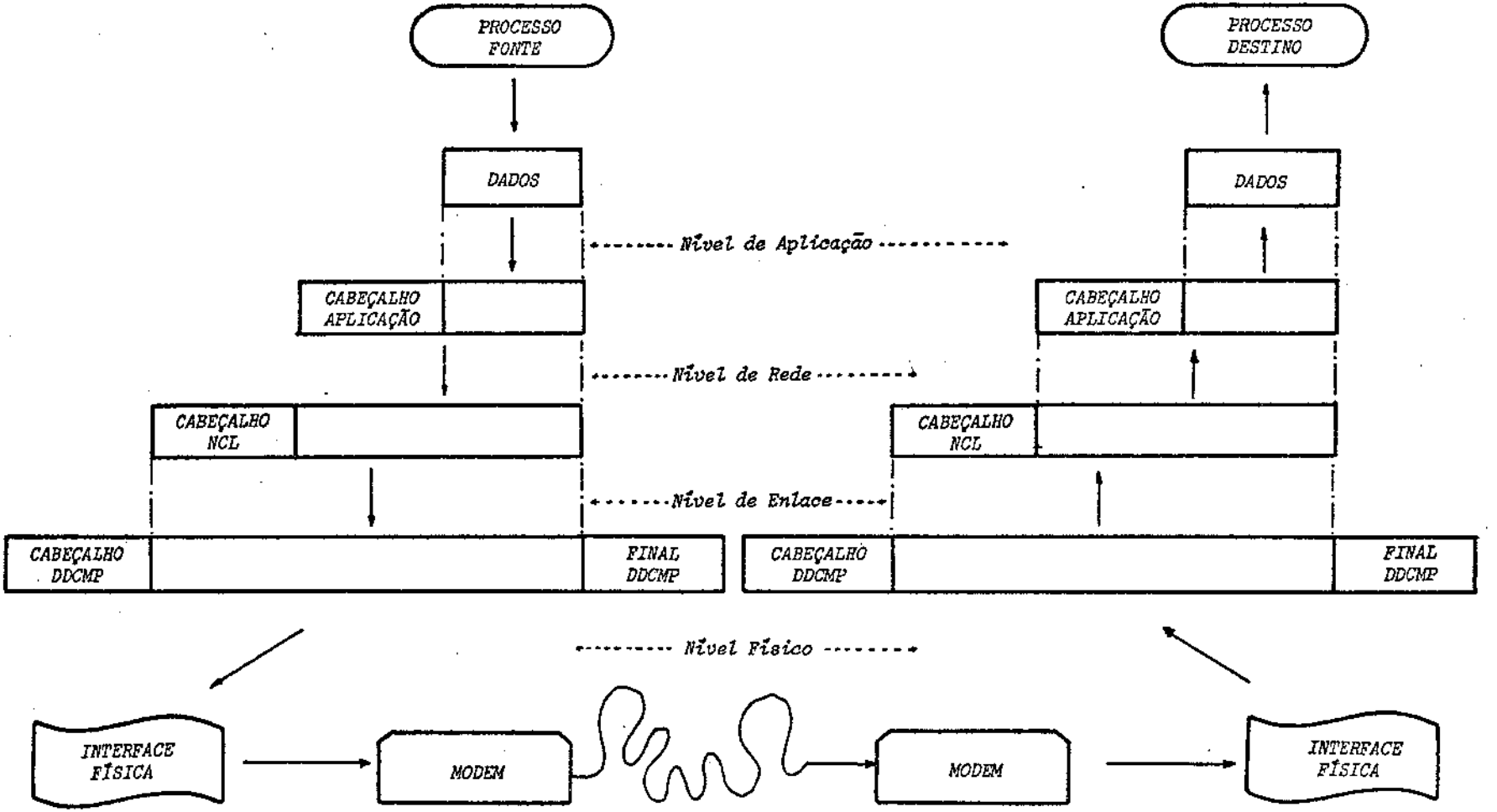


Fig.2.2. Fluxo da informação estação-hospedeiro

2.2 Nível de Enlace

O nível de enlace é o responsável pela manutenção da integridade e sequência dos dados enviados através do canal físico. É regido pelo protocolo DDCMP (Digital Data Communications Protocol) , que define a estrutura, conteúdo, procedimento para a sequência de mensagens e a técnica para detecção e recuperação de erros. (Em 1.1.2 foi utilizada a nomenclatura recomendada atualmente para a unidade de informação deste nível,que é "quadro". Mas como toda a documentação da DEC utiliza "mensagem" ao invés de "quadro", adotaremos também, a partir de agora, "mensagem" como a unidade de informação do nível de enlace).

O DDCMP será descrito, resumidamente, a seguir. A descrição detalhada pode ser encontrada em [WECKER,77].

2.2.1 DDCMP: Organização Funcional

Do ponto de vista operacional o DDCMP possui três componentes funcionais: Enquadramento, Gerenciamento do Canal Físico e Troca de Mensagens.

2.2.1.1 Componente de Enquadramento

Enquadramento é o processo de localizar o início e o fim de cada mensagem. Para que o enquadramento seja possível é necessário que haja sincronização em três níveis: bit, byte e mensagem.

a. Sincronização de bit
   Efetuadada pelo nível físico
b. Sincronização de byte
   É o processo de localizar corretamente a janela de oito bits na sequencia de bits. Em comunicação síncrona isso é conseguido procurando uma sequência única de bits, chamada byte de sincronismo, e a partir de então tomando cada 8 bits como um byte. Também efetuada pelo nível físico, sob controle do nível

de enlace.

c. Sincronização de Mensagem

Após a sincronização de byte, o DDCMP procura um dos três caracteres especiais de início de mensagem. A partir daí algumas regras simples permitem a localização do final da mensagem.

2.2.1.2 Componente de Gerenciamento do Canal Físico

Usado apenas quando o canal físico opera no modo "half-duplex" ou no caso de canais com ligação multiponto. Não tem aplicação no caso da estação remota, já que esta opera ponto-a-ponto, em modo "full-duplex".

2.2.1.3 Componente de Troca de Mensagens

E a parte do DDCMP que cria um canal físico sequencial e praticamente isento de erros. Este componente transfere dados corretamente e em sequência através de um canal físico. Após o enquadramento, este componente opera a nível de mensagem, trocando mensagens de dados e de controle.

As mensagens de dados são numeradas sequencialmente; a cada mensagem corretamente recebida e passada ao nível 3 é enviado ao transmissor um reconhecimento positivo, para notificá-lo do recebimento correto daquela mensagem. Da mesma forma, se recebida incorretamente, uma mensagem de reconhecimento negativo é enviada.

O DDCMP usa o teste de redundância cíclica CRC-16 (Apêndice A) para detecção, e retransmissão para recuperação de erros.

2.2.2 Mensagens DDCMP

A seguir serão descritos os formatos das mensagens de dados e controle DDCMP e a função de cada campo dentro das mensagens. A seguinte notação é utilizada nessa descrição

CAMPO (COMPRIMENTO): CÓDIGO = descrição do CAMPO

onde CAMPO = nome do campo sendo descrito

COMPRIMENTO = comprimento do campo, podendo ser

1. um número significando o número de bytes
2. um número seguido da letra B, significando o número de bits.
3. as letras EX, significando um "campo extensível". Campos extensíveis são de comprimento variável, consistindo de bytes onde o bit de mais alta ordem indica se o byte seguinte pertence ou não ao mesmo campo. Um bit 1 significa que o próximo byte é ainda parte do campo e um bit Ø significa que este é o último byte. Um campo extensível pode ser ASCII ou binário: se ASCII, é apenas uma cadeia de comprimento variável de caracteres ASCII de sete bits; se binário , sete bits de cada byte devem ser concatenados para formar um único campo binário.
4. a letra n, indicando que o comprimento do campo é variável e será definido posteriormente.

CÓDIGO = o tipo de representação usada, podendo ser

A = caracteres ASCII de sete bits cada

B = binário

BM = máscara de bits (onde cada bit tem um significado específico)

T = transparente neste nível

m = miscelânea, definido posteriormente

2.2.2.1 Mensagens de Dados

As mensagens de dados transportam dados dos usuários através de um canal DDCMP. O seu formato é

SOH	COUNT	RESP	NUM	ADDR	BLKCK1	DATA	BLKCK2

onde

SOH (1):B= identificador de início de mensagem de dados

COUNT(2):B= especifica o número de bytes do campo DATA

RESP(1):B= usado para reconhecer mensagens corretamente recebidas. Implica no reconhecimento de todas as mensagens até este número (inclusive)

ADDR(1):B= especifica a quem é endereçada esta mensagem; sempre 1 para ponto-a-ponto

BLKCK1 (2):B= bloco de bits de verificação do cabeçalho (de SOH a ADDR), usando o teste polinomial CRC-16.

DATA (COUNT):T= campo de dados, totalmente transparente ao protocolo. A única restrição é que deve ter o número de bytes especificado em COUNT

BLKCK2(2):B= bloco de bits de verificação do campo DATA, usando o teste polinomial CRC-16

2.2.2.2 Mensagens de Controle

As mensagens de controle não são numeradas e transportam informações para controle do canal físico, status da comunicação e notificação de inicialização. Tem um comprimento fixo de oito bytes (mesmo comprimento do cabeçalho de mensagens de dados) e a forma geral

onde

ENQ(1):B= identificador de início de mensagem de controle

CTRL(4):B= mensagens de controle propriamente ditas, descritas a seguir

ADDR(1):B= especifica a quem é endereçada a mensagem; sempre 1 para ponto-a-ponto

BLKCK3 (2):B= bloco de bits de verificação da mensagem (de ENQ a ADDR), usando o teste polinomial CRC-16.

2.2.2.2.a Mensagem de Reconhecimento (ACK)

Usada para informar o recebimento correto de mensagens numeradas. A mensagem ACK fornece a mesma informação do campo RESP de mensagens numeradas e é utilizada quando é necessário enviar um reconhecimento e não há nenhuma mensagem numerada a ser enviada. O formato da mensagem ACK é

onde

ACKTYPE(1):B= identificador de tipo ACK

FILL(1):T= apenas preenchimento; não considerado

RESP(1):B= mesma função descrita em mensagem de dados.

2.2.2.2.b Mensagem de Reconhecimento Negativo (NAK)

Usada para passar ao nó fonte a informação de que um erro foi detectado pelo nó destino. A mensagem NAK inclue a mesma informação da mensagem ACK, servindo a dois propósitos: reconhecer mensagens corretas previamente recebidas e notificar a existência de alguma condição de erro a partir da última reconhecida. Tem o formato

ENQ	NAKTYPE	REASON	RESP	FILL	ADDR	BLKCK3

onde

NAKTYPE(1):B= identificador de tipo NAK

REASON(1):B= a razão do envio desta mensagem. Identifica a fonte e o tipo de erro (erro de CRC no cabeçalho,no campo de dados, falta de espaço para recebimento,etc)

RESP (1):B= indica reconhecimento positivo até a mensagem de número RESP (inclusive) e condição de erro em alguma mensagem com número maior que RESP

FILL(1):T= preenchimento, não considerado.

2.2.2.2.c Mensagem de Pedido de Reconhecimento (REP)

É enviada quando um nó transmitiu mensagens numeradas que não foram reconhecidas dentro de um certo tempo. A resposta a uma mensagem REP é um ACK ou um NAK, dependendo se o nó destino recebeu ou não corretamente todas as mensagens enviadas. O formato da mensagem REP é

onde

REPTYPE(1):B= identificador de tipo REP

FILL(2):T= preenchimento, não considerado

NUM(1):B= número da última mensagem de dados enviada (não considerando retransmissões).

2.2.2.2.d Mensagem de Início (START)

Usada para estabelecer o contato inicial em um canal DDCMP, sendo necessária apenas para "partida" ou reinicialização. A mensagem START opera em conjunto com a mensagem STACK, descrita a seguir, formando a sequência de inicialização. Esta sequência coloca a numeração de mensagens no estado inicial (zero) em ambos os nós ligados pelo canal DDCMP. O seu formato é

onde

STARTYPE(1):B= identificador de tipo START

FILL(3):T= preenchimento, não considerado.

2.2.2.2.e Mensagem de Reconhecimento de Início (STACK)

É enviada em resposta a uma mensagem START quando o nó já completou sua inicialização. Tem o formato

ENQ	STACKTYPE	FILL	ADDR	BLKCK3

onde

STACKTYPE(1):B= identificador de tipo STACK

FILL(3):T= preenchimento, não considerado.

2.2.3 Exemplos de Trocas de Mensagens

Estes exemplos são apresentados para mostrar a operação do componente de troca de mensagens em casos específicos de estados e eventos.

A notação utilizada é a tradicionalmente empregada para indicar a sequência de troca de mensagens entre dois nós, com diagramas onde o eixo vertical representa o tempo e o horizontal a distância. A transmissão de mensagens é representada através de retas orientadas e inclinadas, indicando a origem, destino e tempo de transmissão. O tipo de mensagem e o valor dos campos são superpostos às retas, completando a informação.

Os seguintes estados e variáveis são usados nos diagramas:

a. Variáveis

R: o mais alto número de mensagem de dados recebida (corretamente, é claro) por este nó. Enviado no

campo RESP de mensagens de dados, ACK e NAK.

N: o mais alto número de mensagem de dados transmitida por este nó. Enviado no campo NUM de mensagens REP.

A: o mais alto número de mensagem de dados já reconhecida a este nó. Recebida no campo RESP de mensagens de dados, ACK ou NAK.

T: número da próxima mensagem a ser transmitida. No caso de transmissão T tem o valor N + 1; no caso de retransmissões T regride a A + 1 e avança até N+1.

X: número da última mensagem cuja transmissão foi completada. No caso de transmissão X tem o valor N; em retransmissões X terá um valor menor ou igual a N.

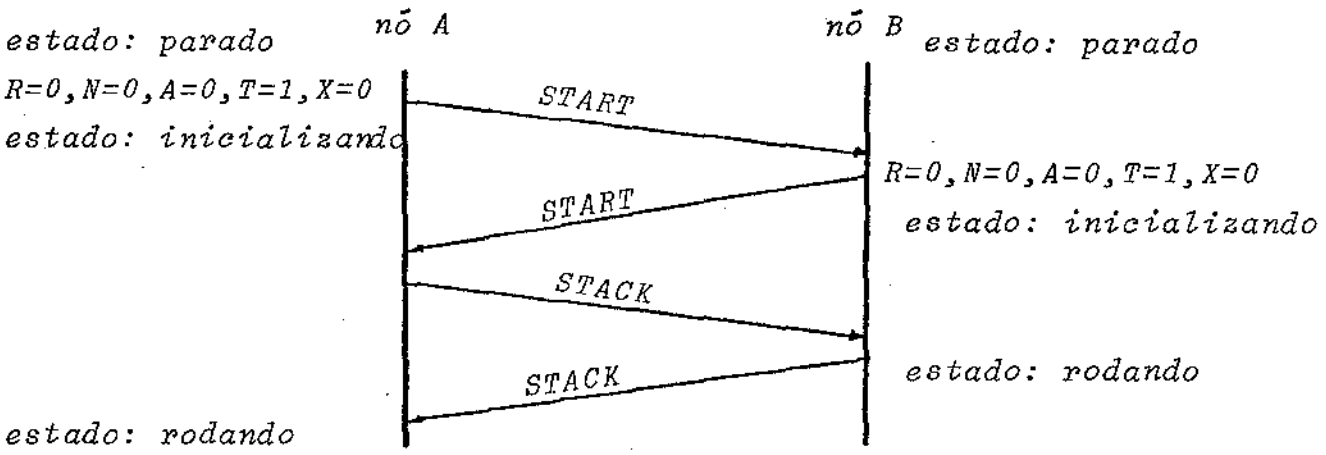
b. Estados

Parado: o componente de Troca de Mensagens do DDCMP está parado.

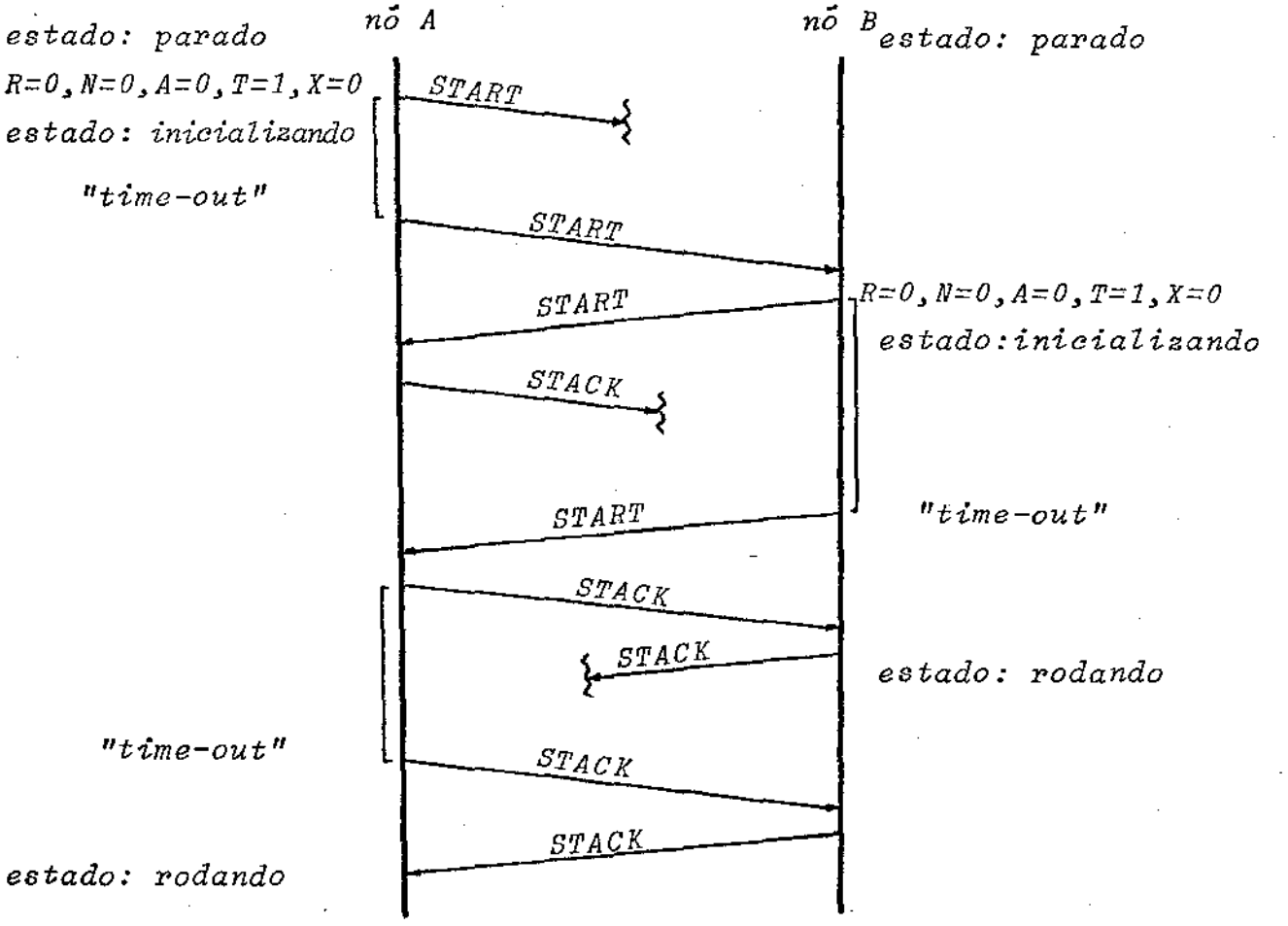
Inicializando: está sendo feita uma tentativa de inicializar o componente de Troca de Mensagens pela troca de mensagens START e STACK.

Rodando: significa que o DDCMP está trocando mensagens normalmente.

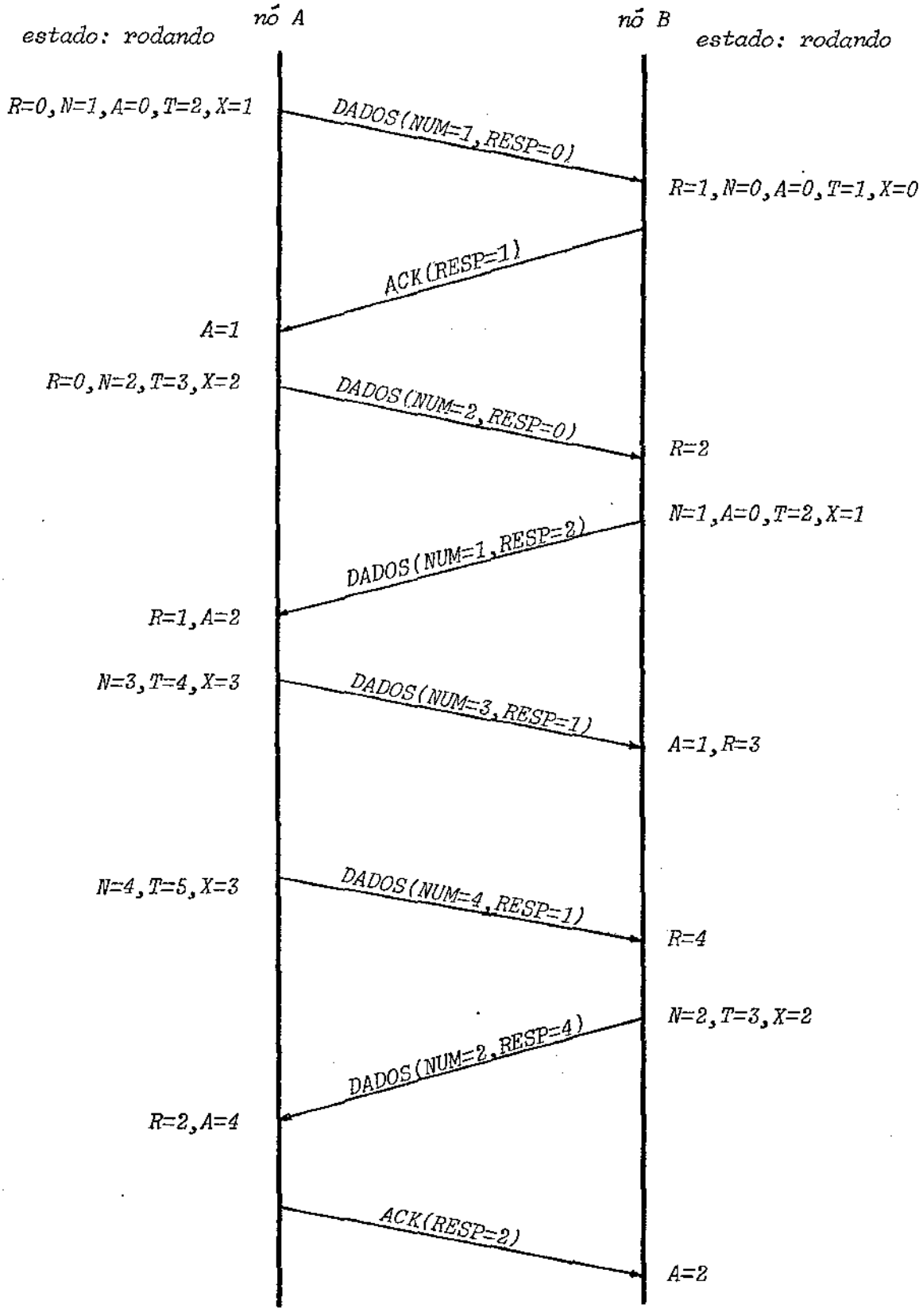
2.2.3.1 Sequência de Partida sem Erros



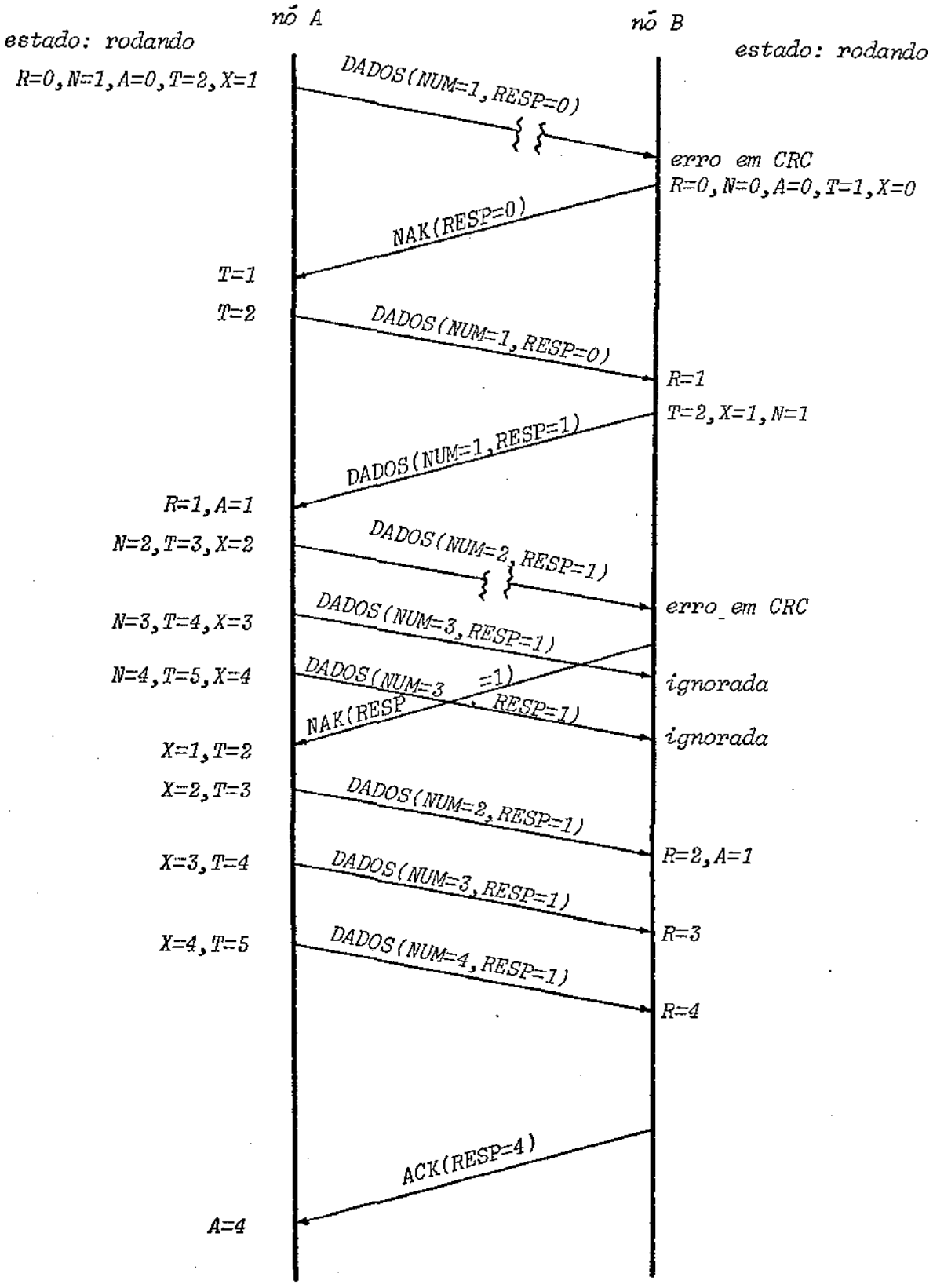
2.2.3.2 Sequência de Partida com Erros



2.2.3.3 Tranferência de Dados sem Erros



2.2.3.4 Transferência de Dados com Erros



2.3 Nível de Rede

O nível de rede é regido pelo protocolo NSP (Network Services Protocol), sendo o responsável pela criação de linhas de conversação (canais lógicos) entre os processos na estação remota (Controladores de Terminais e Controlador de Impressora) e seus correspondentes no computador hospedeiro. O NSP assume a existência de um canal físico isento de erros, fornecido pelo nível de enlace.

Na atual fase de implementação da DECnet no sistema DEC1Ø, para a comunicação estação-hospedeiro é utilizado apenas um subconjunto do NSP, denominado NCL (Network Control Link), que será aqui descrito. A especificação detalhada do NSP pode ser encontrada em [DIGITAL,75].

O NCL implementa as seguintes funções:

(i) Funções do nível de diálogo

   a. estabelecer um canal de conversação (canal lógico)
   b. transmitir dados através de um canal lógico
   c. receber dados através de um canal lógico
   d. desativar um canal lógico previamente estabelecido

(ii) Funções de operação do canal

   a. multiplexar canais lógicos em um canal físico
   b. controlar o tráfego em canais lógicos

2.3.1 Mensagens NCL

Esta seção descreve as funções e os formatos das men-

sagens NCL utilizadas na comunicação estação-hospedeiro,empregando a notação introduzida em 2.2.2. Na descrição dos tipos de mensagens há também a indicação se a mensagem é gerada apenas pelos hospedeiro (H→E), apenas pela estação remota (E→H)ou se pode ser gerada por ambos (H↔ E).

Todas as mensagens NCL tem o formato básico

NCT	DNA	SNA	NCA	NCN	MSG

onde

NCT(EX):BM= identificador de tipo de mensagem e cabeçalho de mensagens NCL, com os bits assim definidos

bit Ø-2: identificadores de tipos de mensagens de controle não numeradas (de 1 a 6). O valor zero indica que a mensagem é de dados ou de controle numerado

3: indica a presença ou não dos campos DNA e SNA

4: não utilizado

5: mensagem não pedida (não é resposta a uma mensagem DATA REQUEST, definida adiante)

6: indica que a mensagem é para um nó intermediário da rede

7: bit de extensão

DNA(EX):B= número do nó destino

SNA(EX):B= número do nó fonte

NCA(1):B= número da última mensagem reconhecida pelo nó fonte; não utilizado por estações do tipo DC72NP

NCN(1):B= número desta mensagem (exceto para mensagens de con-

trole não numeradas);não utilizado por estações do tipo DC72NP

MSG(n):m= mensagem NCL, podendo ser de três tipos: mensagem de dados, mensagem de controle numerada e mensagem de controle não numerada, descritas a seguir.

2.3.1.1 Mensagem de Dados

As mensagens de dados (DATA: H↔E) são usadas para transferência de dados entre dois processos através de um canal lógico previamente estabelecido. O seu formato é:

onde

DLA(EX):B= número do canal lógico destino, obrigatoriamente diferente de zero (é isto que indica ser uma mensagem de dados)

DATA(n):T= campo de dados, recebido do nível de aplicação do processo fonte e passado integralmente ao nível de aplicação do processo destino. Diversas mensagens do nível de aplicação podem ser concatenadas neste campo.

2.3.1.2 Mensagens de Controle Não Numeradas

São mensagens utilizadas para manter a sequência correta de recebimento de mensagens de controle numeradas e de dados. O tipo da mensagem é indicado pelos bits Ø-2 do campo NCT do cabeçalho de mensagens NCL. Os tipos definidos no protocolo são reconhecimento positivo (ACK), reconhecimento negativo (NAK),

pedido de reconhecimento (REP), início (START), reconhecimento de início (STACK) e identificação de nõ (NODE ID).

As mensagens ACK, NAK, REP, START e STACK são empregadas para garantir que o nõ destino receba, em sequência, todas as mensagens numeradas (dados e controle) NCL enviadas pelo nõ fonte, independentemente da falha de algum canal ou nõ intermediário da rede. Essas mensagens são desnecessárias do ponto de vista da estação DC72NP, pois esse tipo de estação é um nõ terminal, ligando-se à rede através de um nõ intermediário (que pode ser, por exemplo, o concentrador de estações remotas DC75NP) que é o responsável pela troca dessas mensagens com o nõ fonte (hóspedeiro). A comunicação estação remota- nõ intermediário é ponto a ponto, o que faz com que a sequência das mensagens NCL esteja garantida pelo nível de enlace.

A mensagem de identificação de nõ (NODE ID:H↔E) é utilizada para informar a todos os nós da rede a identificação do nõ fonte, tendo o formato

onde

NNM(EX):B= número do nõ fonte (igual ao SNA do cabeçalho)

SNM(EX):A= nome do nõ fonte

SID(EX):A= identificação do "software" (nome, versão e data)

2.3.1.3 Mensagens de Controle Numeradas

São mensagens utilizadas para controle da operação dos canais lõgicos: estabelecimento de um canal lõgico,controle de

tráfego em um canal lógico, etc.

As mensagens de controle numeradas têm o formato geral

onde

NCTR(1):B= identificador de mensagens de controle numeradas, obrigatoriamente igual a zero (veja o campo DLA em 2.3.1)

COUNT(EX):B= número de bytes do campo CTRMSG

CTRMSG(COUNT):m= mensagem de controle numerada, podendo ser de seis tipos: pedido de configuração, configuração, conexão, desconexão, pedido de dados e vizinhos autorizados, que serão descritas a seguir.

2.3.1.3.a Mensagem Pedido de Configuração

A mensagem pedido de configuração (REQUEST CONFIGURATION:H→E) é enviada pelo computador hospedeiro à estação remota para adquirir informações sobre os processos disponíveis na estação. Seu formato é

onde

RCONFTYPE(1):B= identificador de mensagem pedido de configuração (REQUEST CONFIGURATION)

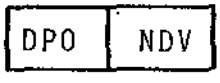
2.3.1.3.b Mensagem de Configuração

A mensagem de configuração (CONFIGURATION:E→H) é enviada em resposta a uma mensagem pedido de configuração. Tem o formato

onde

CONFTYPE(1):B= identificador de mensagem tipo configuração(CONFIGURATION)

CONFLIST(n):m= lista de descrição dos processos disponíveis,onde cada elemento é do tipo



onde

DPO(1):B= identificador do código do processo

NDV(EX):B= número de áreas de dados disponíveis para processos que usam o código DPO.

2.3.1.3.c Mensagem de Conexão

A mensagem de conexão (CONNECT) é usada para estabelecer um canal lógico entre dois processos através de um canal físico. Existem dois estados de transição na "sequência de conexão", que consiste na troca de mensagens de início de conexão (CONNECT INITIATE:H→E) e de confirmação de conexão (CONNECT CONFIRM:E→H).

O formato da mensagem de conexão é

CONTYPE	DLA	SLA	DPO	DPN	SPO	SPN	MML	FEA

onde

CONTYPE(1):B= identificador de mensagem tipo conexão

DLA(EX):B= número do canal lógico destino. Se igual a zero significa que a mensagem é do tipo início de conexão (CONNECT INITIATE); se diferente de zero é uma mensagem de confirmação de conexão (CONNECT CONFIRM).

SLA(EX):B= número do canal lógico fonte.

DPO(1):B= identificador do código do processo destino;é algum dos identificadores DPO presentes na mensagem de configuração.

DPN(EX):B= identificador da área de dados do processo destino.

SPO(1):B= identificador do código do processo fonte, similar a DPO.

SPN(1):B= identificador da área de dados do processo destino.

MML(EX):B= comprimento máximo de mensagem NCL que será enviada pelo canal lógico fonte.

FEA(n):m= características do equipamento associado ao processo fonte; é um parâmetro passado ao processo destino, a quem cabe a interpretação deste campo.

2.3.1.3.d Mensagem de Desconexão

A mensagem de desconexão (DISCONNECT) é utilizada para

desativar um canal lógico previamente estabelecido ou para responder negativamente a uma mensagem de conexão. A sequência de desconexão é constituída da troca de mensagens início de desconexão (DISCONNECT INITIATE:H↔E) e confirmação de desconexão (DISCONNECT CONFIRM:H↔E). O formato é

DISTYPE	DLA	SLA	RSN

onde

DISTYPE(1):B= identificador de mensagem de desconexão

DLA(EX):B= número do canal lógico destino

SLA(EX):B= número do canal lógico fonte. Se igual a zero significa que a mensagem é do tipo confirmação de desconexão (DISCONNECT CONFIRM); se diferente de zero a mensagem é pedido de desconexão (DISCONNECT INITIATE)

RSN(1):B= razão da desconexão, podendo ser, entre outras,

Ø = desconexão normal

1 = tipo de código de processo não disponível

2 = conexão em excesso para este nó.

2.3.1.3.e Mensagem Pedido de Dados

A mensagem pedido de dados (DATA REQUEST:E→H) é usada para controle de fluxo nos canais lógicos. Ela é enviada para sinalizar a existência de espaço disponível para recebimento de dados nos "buffers" do processo fonte. Tem o formato

DATREQTYPE	DLA

onde

DATREQTYPE(1):B= identificador de mensagem tipo pedido de dados (DATA REQUEST)

DLA(EX):B= número do canal lógico destino.

2.3.1.3.f Mensagem Vizinhos Autorizados

A mensagem vizinhos autorizados (NEIGHBOURS:H→E)é utilizada pelo hospedeiro para informar à estação que nós estão disponíveis na rede e qual a melhor rota (se a estação faz roteamento) para acessá-los. Tem o formato

NEITYPE	NEILIST

onde

NEITYPE(1):B= identificador de mensagem tipo NEIGHBOURS

NEILIST(n):m= lista onde cada elemento tem o formato

NNM	LINK

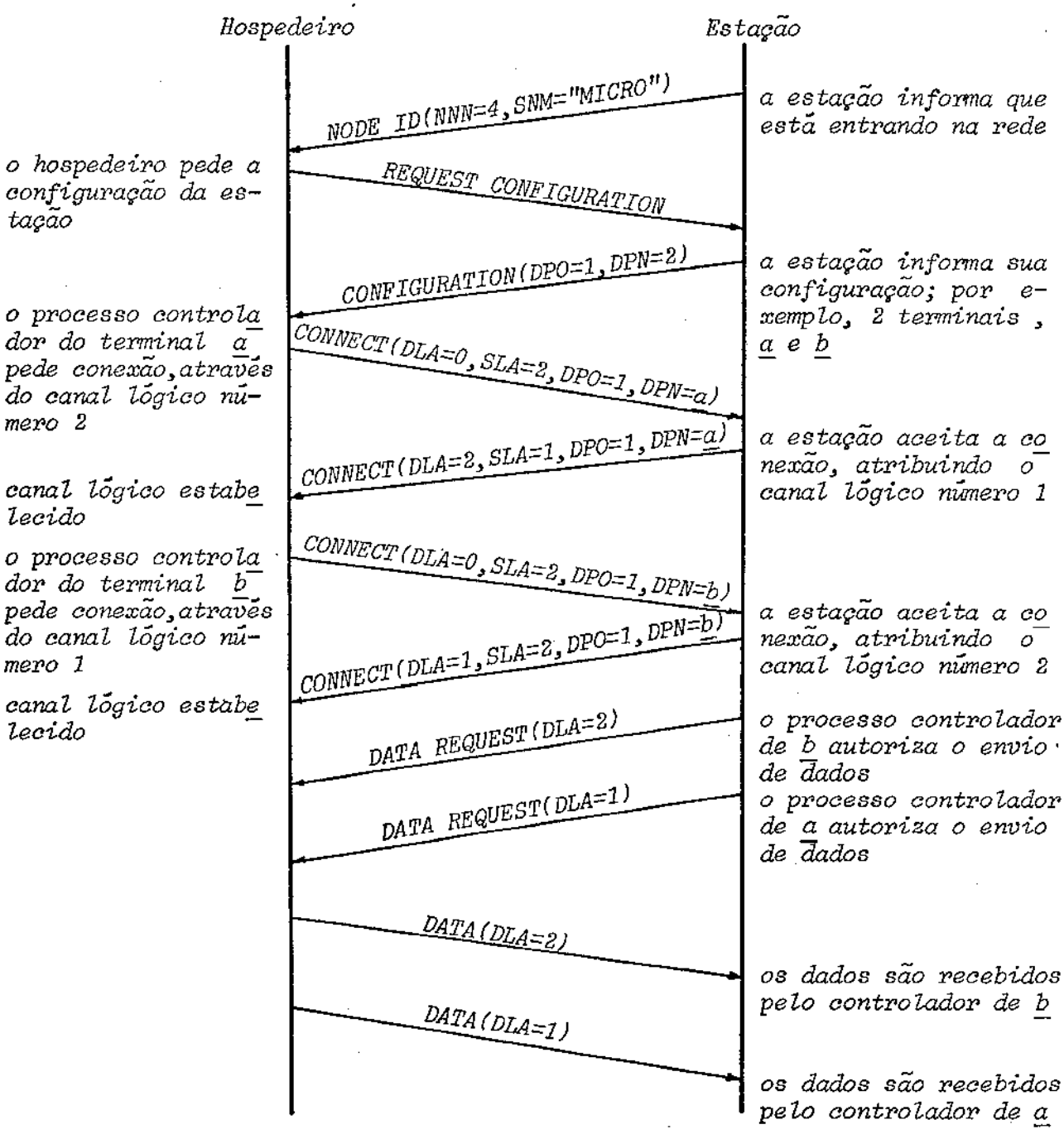
onde

NNM(EX):B= número do nó disponível na rede, ao qual esta estação (cujo número de nó é igual a DNA do cabeçalho)está autorizada a ter processos conectados.

LINK(EX):B= informação para roteamento de mensagens dirigidas ao nó NNM; não utilizada pela estação tipo DC72NP.

2.3.2 Exemplo de Troca de Mensagens

Um exemplo de troca de mensagens NCL entre o hospedeiro e a estação é mostrado a seguir, utilizando os diagramas introduzidos em 2.2.3.



2.4 Nível de Aplicação

O nível de aplicação é o mais alto nível da comunicação estação-hospedeiro, pressupondo a existência de canais lógicos ligando processos na estação remota a seus correspondentes no hospedeiro.

Os processos existentes na estação remota que se comunicam com processos no hospedeiro são os Controladores de Terminais (campo DPO=1 em 2.3.1.3.b e 2.3.1.3.c, indicando código "controlador de terminal") e o controlador da Impressora (DPO=3). Os controladores de terminais trocam, com seus correspondentes no hospedeiro, informações para controle de equipamento (tipo de terminal, controle de eco, etc.) e transmitem (recebem) caracteres provenientes dos (dirigidos aos) terminais. O controlador da Impressora envia a seu correspondente no hospedeiro informações sobre o estado da impressora e recebe caracteres a serem impressos.

O processo Controlador de Leitora de Cartões (DPO=2), também presente em estações tipo DC72NP não foi implementado por não se dispor de tal equipamento.

A seguir serão descritas as mensagens do nível de aplicação. A notação utilizada para indicar o comprimento e o código de cada campo nas mensagens é a mesma introduzida em 2.2.2.

2.4.1 Mensagens dos Controladores de Terminais

Esta seção descreve a função e o formato das mensagens de controle e de dados trocadas entre os controladores de terminais da estação remota e seus correspondentes no hospedeiro, respectivamente designados, de agora em diante, por CTEs e CTHs.

A seguinte convenção é utilizada para indicar a origem das mensagens

CTH → CTE = Apenas o CTH toma a iniciativa do envio da mensagem

CTE → CTH = Apenas o CTE toma a iniciativa do envio da mensagem

CTH ↔ CTE = Tanto o CTH como o CTE tomam a iniciativa do envio da mensagem.

2.4.1.1 Mensagens de Dados

As mensagens de dados (DATA: CTH ↔ CTE) são utilizadas para transferência de caracteres entre o CTE e o CTH. Tem o formato

onde

COUNT(EX):B = especifica o número de caracteres (bytes) dos campos DATATYPE e CHARS

DATATYPE(1):B = identificador de tipo mensagem de dados (DATA)

CHARS(COUNT-1):A = bloco de caracteres ASCII.

2.4.1.2 Mensagens de Controle

As mensagens de controle transportam informações para diversos tipos de controle dos terminais. Têm o formato básico

onde COUNT(EX):B especifica o tamanho em bytes do campo CTRL.

2.4.1.2.a Mensagem Características

A mensagem características (CHARACTERIST:CTH→CTE) é enviada pelo CTH para informar ao CTE as características do terminal que o CTH supõe estar conectado. Tem o formato

onde

CHRTYPE(1):B = identificador de mensagem tipo características (CHARACTERIST)

FILLERS(6):B = tempos de preenchimento para cada um dos seis caracteres que necessitam preenchimento (< CR >, <LF >, <TAB > , etc.)

RCVSPEED(EX):B = velocidade de recebimento da comunicação assíncrona terminal-estação remota

TRSPEED(EX):B = velocidade de transmissão da comunicação assíncrona terminal-estação remota

TTYWID(1):B = número máximo de caracteres em uma linha do terminal.

2.4.1.2.b Mensagens de Estado

As mensagens de estado são trocadas entre o CTE e o CTH para atualização do estado do terminal. O estado do terminal é mantido em uma palavra de bits assim definidos

bit Ø : o terminal está em modo eco adiado (veja Mensagem Marcador de Eco)

1 : o terminal não tem capacidade de imprimir minúsculas e portanto as letras minúsculas a serem impressas devem ser mudadas para maiúsculas

2 : a transmissão de caracteres ao terminal foi temporariamente inibida devido ao recebimento (do terminal) de um caracter XOFF (< CRTL-Q >)

3 : a entrada de caracteres pelo terminal deve ser feita no modo imagem

4 : a saída de caracteres para o terminal deve ser feita no modo imagem

5: o terminal tem tratamento automático de fim de página (aceita <FFEED>)

6 : o terminal tem tratamento automático de fim de linha (insere uma sequência <CR><LF> ao fim da linha)

7 : o terminal tem tabulação horizontal por "hardware"

8 : a entrada de caracteres pelo terminal foi interrompida temporariamente

9 : DTR (Data Terminal Ready): indica que o terminal está em condições de operar.

1Ø: DSR (Data Set Ready): indica que o modem está em condições de operar.

Os bits Ø,2,9 e 1Ø são mantidos pelo CTE, que notifica ao CTH qualquer mudança através da mensagem comunicação de mudança de estado (STATUS REPORT: CTE→CTH), cujo formato é

onde

STSREPORT(1):B = identificador de mensagem comunicação de mudança de estado (STATUS REPORT)

STATUS (EX):BM = nova palavra de estado.

Os demais bits da palavra de estado são mantidos pelo CTH, que comunica ao CTE qualquer mudança através das mensagens ligue bits (SET STATUS BIT:CTH→CTE) e desligue bits (RESET STATUS BIT: CTH→CTE), ambas com o formato

onde

STSTYPE (1):B = identificador de mensagem de estado tipo ligue bits (SET STATUS BIT) ou desligue bits (RESET STATUS BIT)

BITS (EX):BM = bits da palavra de status que devem ser ligados ou desligados.

2.4.1.2.c Mensagem Papa-caracter

Algumas situações exigem que se considere sem efeito caracteres já enviados ao terminal pelo CTH mas que não foram ainda impressos, como, por exemplo, quando o usuário pressiona <CTRL-C >. Isso é conseguido através da mensagem papa-caracter (CHARACTER GOBBLER), que faz com que o CTE que a recebeu limpe o buffer de impressão do terminal.

A mensagem papa-caracter tem o formato

GOBTYPE

onde GOBTYPE(1):B = identificador de mensagem tipo papa-caracter (CHARACTER GOBBLER).

2.4.1.2.d Mensagem Marcador de Eco

Sempre que possível a estação remota fornece eco local ao terminal, ou seja, os caracteres correspondentes às teclas pressionadas são, além de enviados ao CTH, impressos imediatamente nos terminais. Algumas situações, no entanto, podem ocorrer no CTE que exigem que o eco seja deixado sob responsabilidade do CTH, quando então o terminal entra no estado denominado "eco adiado". Neste estado os caracteres correspondentes às teclas pressionadas não são impressos imediatamente, sendo apenas enviados ao CTH, que se encarrega então de remetê-los de volta para serem impressos, se necessário. Exemplos de tais situações são o recebimento,do terminal,de alguns caracteres especiais,como por exemplo <RUBOUT>.

No momento de retorno ao estado de eco local ocorre um

problema de sincronização, já que é necessário saber exatamente a partir de qual caracter o CTE deve retomar controle de eco, fazendo cessar o envio de eco por parte do CTH.

Essa sincronização é conseguida através da mensagem marcador de eco (ECHO-PIPELINE-MARKER: CTH↔CTE), enviada junto a cada bloco de caracteres remetido ao CTH pelo CTE do terminal que está em eco adiado. Ao terminar de enviar os caracteres cujo eco foi deixado sob sua responsabilidade o CTH devolve esse marcador, permitindo assim que o CTE retome o controle do eco.

A mensagem marcador de eco tem o formato

onde

EPLTYPE(1):B = identificador de mensagem tipo marcador de eco (ECHO-PIPELINE-MARKER)

MARKER(1):B = valor do marcador, incrementado a cada mensagem marcador de eco enviada pelo CTE.

2.4.2 Mensagens do Controlador da Impressora

Esta seção descreve a função e o formato das mensagens trocadas entre o processo controlador da impressora na estação remota (CIE) e seu correspondente no hospedeiro (CIH). Devido ao fato de a impressora possuir um caracter mais "estático" do que um terminal, o repertório de mensagens de controle do controlador da impressora se resume à mensagem de estado, não necessitando de mensagens para-caracter, características, etc. As características da impressora são informadas durante o processo

de conexão (no campo FEA de mensagens de conexão, 2.3.1.3.c).

2.4.2.1 Mensagem de Dados

A mensagem de dados (DATA:CIH→ CIE) é utilizada pelo CIH para informar ao CIE quais os caracteres que devem ser impressos. Os caracteres são compactados pelo CIH segundo a convenção

1ccccccc : ccccccc é um caracter ASCII

Ø1xxxxxx : corresponde a xxxxxx brancos

ØØ1xxxxx : xxxxx é o número de vezes que o caracter seguinte deve ser repetido.

O controle do carro da impressora é feito através dos caracteres ASCII correspondentes.

O formato da mensagem de dados é

COUNT	DATATYPE	CHARS

onde

COUNT(EX):B = número de bytes dos campos DATATYPE e CHARS

DATATYPE(1):B = identificador de mensagem de dados (DATA)

CHARS(COUNT-1):m = bloco de caracteres compactados.

2.4.2.2 Mensagem de Estado

A mensagem de estado (status:CIE→ CIH) é enviada pelo CIE ao CIH para informar a este o estado corrente de impressora (remetida a cada mudança de estado).

O estado da impressora é mantido em uma palavra de estado em que alguns dos bits definidos são

bit 1: erro fatal

bit 2: impressora desligada

bit 3: falta de papel.

O formato da mensagem de estado é

onde

COUNT(EX):B = especifica o tamanho em bytes dos campos STSTYPE e STATUS

STSTYPE(1):B = identificador de mensagem de estado (STATUS)

STATUS(EX):B = nova palavra de estado.

CAPITULO 3

A ESTAÇÃO REMOTA DC72/88

Neste capítulo é apresentada a implementação da estação remota DC72/88. A primeira seção descreve a arquitetura da máquina e a segunda descreve o "software" desenvolvido para a estação.

A estação remota DC72/88 é uma máquina baseada no microprocessador 8088A da INTEL, que foi escolhido por ter um conjunto de instruções adequado às necessidades do projeto, bem como pela disponibilidade das pastilhas do sistema INTEL-8Ø88 no mercado nacional. Maiores informações sobre as características físicas das pastilhas utilizadas, e detalhes de programação dessas pastilhas podem ser encontrador em [INTEL].

3.1 Arquitetura

A arquitetura básica da estação DC72/88 pode ser vista no diagrama da Fig.3.1. O bloco CPU compreende a pastilha 8Ø88A e pastilhas de apoio ("drivers" de barramentos, gerador de relógio, etc) e é o principal bloco do sistema, controlando todos os outros. O bloco MEMÓRIA é constituído de pastilhas EPROM (memória de leitura apenas), para armazenagem do programa e tabelas, e de pastilhas RAM (memória de leitura e escrita) estáticas para "buffers" e dados.

O bloco CONTROLADOR DE INTERRUPÇÕES é constituído pela pastilha 8259A-Programmable Controller Interrupt da INTEL,que é um controlador de interrupções programável, com até oito níveis de interrupção (IRØ a IR7). O controlador de interrupções aceita

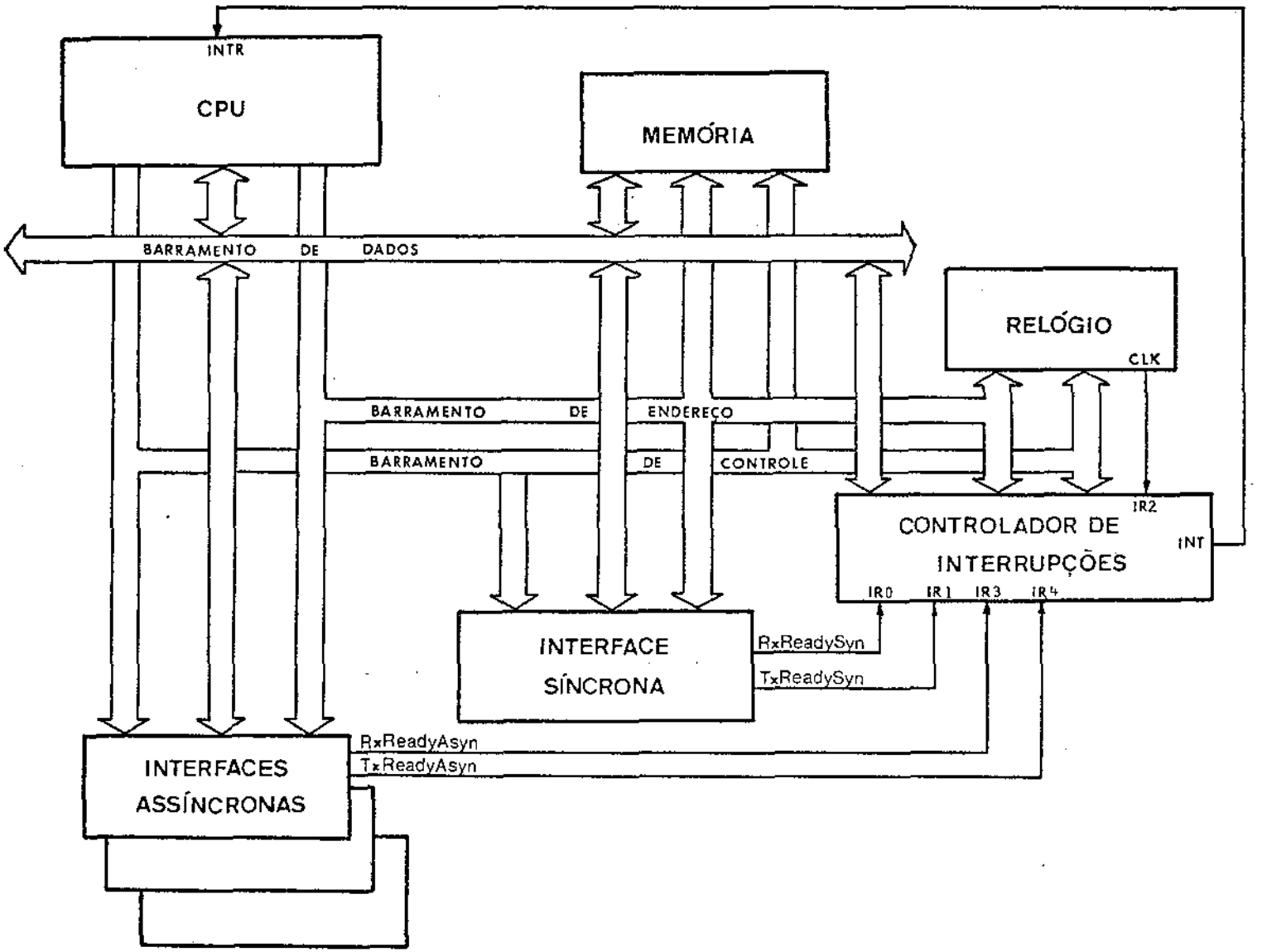


Fig.3.1. Arquitetura básica da estação DC72/88

pedidos de interrupções, decide se e qual deve ser atendido e informa à CPU que há um pedido de interrupção. Quando a CPU decide atender à interrupção o controlador de interrupções lhe informa o endereço da rotina de tratamento da interrupção prioritária.

A INTERFACE SÍNCRONA é uma interface serial EIA-RS232C que utiliza a pastilha 8251A-USART da INTEL. Ela fornece uma linha serial síncrona, "full-duplex", e é conectada a um MODEM para comunicação com o computador hospedeiro. A interface síncrona dispõe de duas linhas de interrupção ligadas ao controlador de interrupções: a linha RxReadySYN, que é acionada quando um byte acabou de ser recebido pela interface, e a linha TxReadySYN, que é acionada quando o transmissor está pronto para ser carregado com um novo byte a ser transmitido. A linha RxReadySYN é conectada a IRØ (mais alta prioridade) e a linha TxReadySYN a IR1.

As INTERFACES ASSÍNCRONAS são interfaces seriais EIA-RS232C. Cada interface utiliza uma pastilha 8251A-USART da INTEL, fornecendo uma linha serial, assíncrona, "full-duplex", com velocidade (escolhida por "jumpers" na interface) entre 150 e 9600 bps (bits por segundo). Uma destas linhas é reservada para conexão de uma impressora com entrada serial. As outras linhas são usadas para a ligação de terminais assíncronos, em número limitado pela velocidade combinada de todas as linhas suportadas pelo sistema. Presentemente o sistema dispõe de oito linhas reservadas para terminais.

As interfaces assíncronas dispõe, conjugadamente, de

duas linhas de interrupção ligadas ao controlador de interrupções. Uma é a linha RxReadyASYN , conectada a IR3, que é acionada quando uma tecla foi pressionada em um terminal e o caracter correspondente acabou de ser recebido pela interface (a linha RxReadyASYN é na realidade um OU lógico dos sinais RxReady das oito interfaces ligadas a terminais). A outra linha de interrupção, TxReadyASYN , é conectada a IR4, sendo acionada quando uma ou mais interfaces estão prontas para receber um novo byte para transmissão (esta linha é o OU lógico dos sinais TxReady das nove interfaces). A identificação de qual das interfaces assíncronas pediu a interrupção é feita pelo "software".

O bloco RELÓGIO é constituído por um oscilador conectado à linha IR2 do controlador de interrupção, gerando uma interrupção a cada 16,6ms (60Hz). Essa interrupção é utilizada para controle de várias temporizações na estação remota, como "time-outs" de mensagens DDCMP ou tempos de preenchimento para impressão de alguns caracteres (<CR> ,<LF> , <TAB> , etc.).

3.2 Software

Operacionalmente o "software" da estação remota DC72/88 pode ser dividido em dois grandes blocos funcionais:o bloco de Serviço de Interrupção e o bloco de Serviços Gerais.

O bloco de serviço de interrupção compreende as rotinas de tratamento das interrupções, que executam as tarefas básicas necessárias a cada interrupção. Essas rotinas ocupam o mínimo de tempo de CPU possível e são bastante concisas, executando realmente apenas as tarefas inadiáveis.

O bloco de serviços gerais abrange todo o resto do

"software", podendo ser interrompido a qualquer momento (a menos de regiões críticas) pelas rotinas do serviço de interrupção.

3.2.1 Serviço de Interrupção

O sistema DC72/88, conforme descrito em 3.1, apresenta cinco níveis de interrupção:

nível Ø . Receptor Interface Síncrona
1 . Transmissor Interface Síncrona
2 . Relógio
3 . Receptor Interfaces Assíncronas
4 . Transmissor Interfaces Assíncronas

Cada um destes níveis tem uma rotina de serviço específica, descritas nas seções seguintes.

3.2.1.1 Rotina de Serviço do Receptor da Interface Síncrona (INTSRS)

Acionada a cada byte recebido pela interface síncrona, tem a função de realizar o enquadramento de mensagens DDCMP, alocando-as na fila de mensagens recebidas (RCVQUE). A RCVQUE é uma fila circular onde cada elemento tem o formato

informação	apontador
(8 bytes)	(2 bytes)

Para mensagens de controle DDCMP, informação contém a própria mensagem e apontador é desconsiderado. Para mensagens de dados DDCMP, informação contém o cabeçalho da mensagem e apontador é um ponteiro para a parte de dados da mensagem (campos DATA e BLKCKZ).

Esse sistema é utilizado devido ao fato de o campo DATA ter comprimento variável, exigindo, para melhor aproveitamento de espaço na memória, de algum tipo de alocação dinâmica. Assim, uma parte da memória é transformada em uma lista ligada de pedaços de memória ("chunks"), de tamanho fixo, que são alocados dinamicamente por INTSRS durante o recebimento de mensagens de dados. Cada "chunk" tem o formato

NXTCNK	CNTCNK	DATCNK
(3 bytes)	(1 byte)	(13 bytes)

onde

NXTCNK : apontador para o próximo "chunk"

CNTCNK : número de bytes ocupados em DATCNK

DATCNK : área de memória utilizada para armazenagem de dados.

Um esquema do reservatório de "chunks" pode ser visto na Fig.3.2, onde FIRFRE representa o primeiro "chunk" disponível.

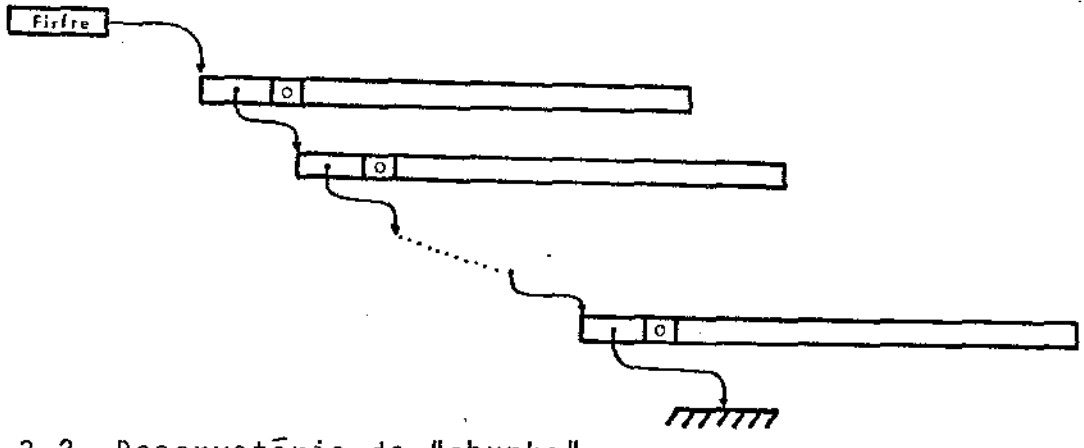


Fig.3.2. Reservatório de "chunks"

A RCVQUE tem dois ponteiros associados: RCVPTR, de inserção, e RCVTKR, de retirada.

A INTSRS é a rotina de mais alta prioridade, não sendo interrompida por nenhuma outra. A interrupção deste nível está sempre permitida, só sendo inibida no caso de perda de sincronismo por erro de enquadramento, quando então a interface síncrona é reinicializada, buscando a ressincronização de byte.

A descrição resumida do algoritmo utilizado é dada a seguir. As seguintes variáveis são utilizadas na descrição:

F1ØPTR : ponteiro auxiliar, apontando para onde deve ser guardado o byte recebido

F1ØCNT : contador de bytes recebidos, utilizado para fazer o enquadramento DDCMP.

Os identificadores sublinhados (por exemplo, COUNT), representam os campos de mensagens DDCMP, descritos em 2.2. A constante SYNC representa o byte de sincronismo do protocolo DDCMP.

Algoritmo INTSRS

1. (Pega byte) Lê a interface síncrona e faz BYTE ← byte recebido.
2. (Recebendo mensagem?) Se já está recebendo mensagem, vai para 7.
3. (Início msg dados?) Se BYTE = SOH (início msg dados) vai para 6.
4. (Início msg controle?) Se BYTE = ENQ (início msg controle) vai para 6.
5. (deve ser SYNC) Se BYTE = SYNC retorna. Caso contrário reinicializa a interface síncrona (houve erro de enquadramento) e

retorna.

6. (Início msg) Faz FIØPTR ← RCVPTR, guarda BYTE no lugar apontado por FIØPTR, avança FIØPTR e retorna.
7. (Recebendo cabeçalho ?) Se está recebendo o campo de dados, vai para 9. Caso contrário guarda BYTE no lugar apontado por FIØPTR e avança FIØPTR.
8. (Terminou cabeçalho ?) Se não terminou o cabeçalho, retorna. Se a mensagem recebida é controle, avança RCVPTR e retorna. Se a mensagem recebida é dados, pega um "chunk" do reservatório e aponta FIØPTR para o primeiro byte útil; faz FIØCNT ← COUNT, faz o registro APONTADOR desse elemento da fila RCVQUE apontar para o chunk pego e retorna.
9. (Recebendo Dados) Guarda BYTE no lugar apontado por FIØPTR, avança FIØPTR (se o chunk acabou, pega outro do reservatório e o liga) e decrementa FIØCNT.
10. (Terminou msg dados?) Se FIØCNT é igual a zero, avança RCVPTR e retorna. Caso contrário apenas retorna.

3.2.1.2 Rotina de Serviço do Transmissor da Interface Síncrona (INTSTS)

As interrupções deste nível só são permitidas quando a transmissão ou retransmissão de uma mensagem DDCMP é requisitada. A partir de então a rotina INTSTS é acionada cada vez que o transmissor se declara pronto a aceitar outro byte para transmissão. A função de INTSTS é fornecer ao transmissor o próximo byte a ser transmitido.

As mensagens de controle DDCMP são montadas pelo Empa-

cotador de Mensagens (3.2.2.4) em um "buffer" de tamanho fixo,o CTRBUF. Já as mensagens de dados DDCMP são montadas em uma fila circular onde cada elemento é um buffer de tamanho fixo (T1ØBUF). O sistema de chunks não é utilizado na transmissão porque o Empacotador monta mensagens de tamanho máximo relativamente pequeno (80 bytes) e o número de elementos em T1ØBUF é também pequeno (3). A fila T1ØBUF tem três ponteiros associados, usados para retirada (T1ØTKR), inserção (T1ØPTR) e reconhecimento (T1ØAKR), que são administrados pelo Empacotador. O ponteiro de reconhecimento T1ØAKR é necessário pelo fato de uma mensagem só poder ser descartada após ter sido reconhecida.

Para efetuar a transmissão a INTSTS utiliza duas variáveis

T1ØCA : apontador para o próximo byte a ser enviado

e

T1ØBCN : número de bytes ainda a enviar.

Para mensagens de controle DDCMP, T1ØCA é inicializado com o endereço de CTRBUF e T1ØBCN tem o valor inicial 8 (mensagens de controle DDCMP tem sempre comprimento de 8 bytes). Em mensagens de dados DDCMP, T1ØCA é inicializado com T1ØTKR e T1ØBCN tem o valor inicial determinado pelo campo COUNT no cabeçalho da mensagem. Essa inicialização é feita pela rotina XSTART do Empacotador de Mensagens(3.2.2.4.a),que também ativa as interrupções do transmissor síncrono.

A partir de então, a cada interrupção INTATS alimenta o transmissor síncrono e atualiza T1ØCA e T1ØBCN. Após a trans-

missão do último byte da mensagem as interrupções são inibidas, devendo ser novamente ativadas pelo Empacotador quando uma nova mensagem estiver pronta para ser enviada.

Antes de cada mensagem é enviada uma sequência de oito bytes de sincronismo DDCMP (SYNC), para assegurar a sincronização. A variável STSCNT é usada para contabilizar os SYNCs enviados.

A rotina INTSTS tem alta prioridade, não sendo interrompida por nenhuma outra do serviço de interrupções. Nem mesmo uma interrupção do receptor síncrono é permitida durante a execução de INTSTS.

A descrição do algoritmo utilizado é dada a seguir. A variável T1ØFLG indica se o transmissor está ativo (T1ØFLG = 1) ou inativo (T1ØFLG = Ø).

Algoritmo INTSTS

1. (SYNCs a enviar?) Se STSCNT ≠ Ø, decrementa STSCNT, alimenta o transmissor com SYNC e retorna.
2. (Terminou esta mensagem?) Se T1ØBCN = Ø, desliga transmissor, faz STSCNT ← 8, T1ØFLG ← Ø e retorna.
3. (Alimenta transmissor) Alimenta transmissor com o byte apontado por T1ØCA, avança T1ØCA, decrementa T1ØBCN e retorna.

3.2.1.3 Rotina de Serviço do Relógio (INTCLK)

A função da rotina INTCLK é apenas levantar uma "flag" (CLKFLG) a cada interrupção (o que ocorre a cada 16,6 ms)e uma "flag" a cada segundo (SECFLG). Durante sua execução INTCLK per-

mite interrupções de níveis prioritários (níveis Ø e 1) e inibe todas as demais.

A CLKFLG é usada para temporização de "tempos de preenchimento" na impressão de caracteres nos terminais e na impressora, e também para temporização de "time-out" de mensagens DDCMP.

A SECFLG é utilizada para monitorar o funcionamento do sistema, sendo desligada, juntamente com CLKFLG, cada vez que é executada a rotina de temporização de tempos de preenchimento. Assim, se SECFLG não for desligada (por uma rotina do Serviço Geral) antes de completado outro segundo, uma mensagem de advertência é impressa no consòle (terminal nºØ), avisando que há sobrecarga no sistema.

A descrição resumida do algoritmo utilizado é dada a seguir.

Algoritmo INTCLK

1. (Ativa interrupções) Permite interrupções geradas pelos níveis mais altos de prioridade.
2. (Levanta CLKFLG) Levanta CLKFLG e decrementa CLKCNT.
3. (Hora de levantar SECFLG?) Se CLKCNT ≠ Ø, retorna.
4. (Levanta SECFLG) Se SECFLG já está levantada, deixa mensagem para o operador; levanta SECFLG.
5. (Retorna) Retorna.

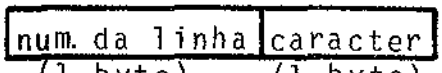
3.2.1.4 Rotina de Serviço do Receptor das Interfaces Assíncronas (INTARS)

As interrupções deste nível (3) são geradas sempre que

alguma tecla de algum terminal da estação remota é pressionada.

A função de INTARS é,primeiramente, determinar qual terminal necessita de atendimento. Isso é feito consultando sequencialmente o status de cada linha, e para evitar a hierarquização no atendimento aos terminais essa consulta é feita circularmente, sempre a partir do último terminal atendido.

Determinado o terminal INTARS lê o caracter correspondente e armazena essas informações em uma fila circular, chamada Fila de Entrada dos Terminais (TTIQUE). Cada elemento dessa fila tem então o formato



A TTIQUE tem um ponteiro de inserção (TIQPTR) que é atualizado a cada chamada de INTARS. A retirada dos elementos da fila é feita por um ponteiro (TIQTKR) administrado pelo Rastreador (3.2.2.5).

A rotina INTARS permite interrupções geradas pelos níveis Ø, 1 e 2, e inibe as demais (inclusive outras de seu próprio nível).

O algoritmo utilizado é descrito resumidamente a seguir.

Algoritmo INTARS

1. (Ativa interrupções) Permite interrupções dos níveis prioritários.
2. (Determina a quem atender) A partir do último atendido, consulta os receptores das interfaces assíncronas para saber qual necessita serviço.
3. (Lê caracter) Lê o caracter da interface escolhida.

4. (Guarda na fila) Guarda o número do terminal atendido no lugar apontado por TIQPTR; avança TIQPTR; guarda o caracter lido no lugar apontado por TIQPTR; avança TIQPTR.
5. (Retorna) Retorna.

3.2.1.5 Rotina de Serviço do Transmissor das Interfaces Assíncronas (INTATS)

As interrupções do nível 4 são geradas sempre que o transmissor de alguma interface assíncrona (terminais ou impressora) se declara apto a receber um novo byte para transmissão.

Analogamente à rotina INTARS, a primeira tarefa de INTATS é determinar qual linha necessita de atendimento,o que é feito de maneira similar à descrita em INTARS (consulta circular).

Determinada a linha, INTATS consulta seu controlador (pode ser terminal ou impressora) para saber qual caracter deve ser transmitido e alimenta o transmissor. Caso não haja mais caracteres destinados a esta linha o transmissor dessa interface é desativado a fim de evitar interrupções desnecessárias. O transmissor é também desativado no caso de caracteres que necessitam de tempos de espera (preenchimento) para serem impressos; quando necessário, o transmissor será reativado pelo controlador de sua linha.

A rotina INTATS permite interrupções geradas pelos níveis prioritários (Ø, 1, 2 e 3), inibindo outras interrupções de mesmo nível. A descrição resumida do algoritmo utilizado é dada a seguir.

Algoritmo INTATS

1. (Ativa interrupções) Permite interrupções dos níveis Ø, 1, 2 e 3.
2. (Determina a quem atender) A partir do último atendido, consulta os transmissores assíncronos para saber qual necessita de atendimento.
3. (Alimenta transmissor) Ativa o processo controlador associado à linha escolhida e, dependendo se é controlador de terminal ou de impressora, chama respectivamente XMTTY (3.2.2.1.c) ou XMTLPT (3.2.2.2.b).
4. (Retorna) Desativa a processo controlador e retorna.

3.2.2 Serviços Gerais

O bloco de serviços gerais é dividido em cinco componentes funcionais: Controlador de Terminais, Controlador da Impressora, Desempacotador de Mensagens, Empacotador de Mensagens e Rastreador.

Os controladores (de Terminais e da Impressora) são processos da estação remota que se comunicam com processos correspondentes no hospedeiro, através de mensagens do nível de Aplicação. Com o auxílio das rotinas do serviço de interrupções os Controladores executam o atendimento aos terminais e à impressora, fazendo a leitura das teclas pressionadas, trocando informações com seu correspondentes no hospedeiro e imprimindo os caracteres necessários.

O Desempacotador e o Empacotador de mensagens são os responsáveis pelos níveis de enlace e de rede da comunicação

estação-hospedeiro. O Desempacotador recebe mensagens do hospedeiro, desempacotando-as dos protocolos DDCMP e NCL e entregando-as ao controlador do processo já no nível de aplicação.

Ao Empacotador de Mensagens cabe empacotar nos protocolos NCL e DDCMP as mensagens recebidas dos controladores (já no nível de aplicação) e enviá-las ao hospedeiro.

A tarefa do Rastreador é procurar e distribuir o trabalho deixado pelo bloco de Serviço de Interrupções para ser executado pelo bloco de Serviços Gerais.

Nas seções seguintes são descritos os cinco componentes do bloco de Serviços Gerais.

3.2.2.1 Controladores de Terminais

Os Controladores de Terminais são os processos que cuidam do atendimento aos terminais presentes na estação remota.Um controlador de terminal executa as seguintes tarefas

(i) recebimento de caracteres enviados pelos terminais

(ii) preparação de mensagens (nível de aplicação) para serem enviadas ao processo correspondente no hospedeiro

(iii) interpretação de mensagens (nível de aplicação) recebidas do processo correspondente no hospedeiro

(iv) transmissão de caracteres aos terminais.

As tarefas (i) e (ii) são executadas pela mesma rotina, a RCVTIQ, que trata o caracter recebido do terminal (colocado na fila de entrada TTIQUE pela rotina do serviço de interrupção das interfaces assíncronas) e já vai montando a mensagem (nível de

aplicação) que será entregue ao Empacotador de Mensagens.

A tarefa (iii) é executada pela rotina RCVTTY, que recebe do Desempacotador de Mensagens uma mensagem no nível de aplicação e faz seu tratamento. A tarefa (iv) é executada pela rotina XMTTY, chamada a nível de interrupção. As rotinas RCVTIQ, RCVTTY e XMTTY serão descritas adiante.

A área de dados de cada processo controlador de terminal é denominada Descritor de Terminal, e se compõe de informações sobre o estado atual do terminal e de buffers para estrada/saída.

Os campos e as funções de cada campo presente nos Descritores de Terminais são

DEVADR : endereço físico do equipamento, usado para entrada/saída de dados e palavras de controle da linha

DEVOBT: tipo do equipamento (1 para terminal)

DEVNUM: número deste terminal, usado para identificação (Ø a 7 na DC72/88)

DEVRNN: se este terminal é restrito ao hospedeiro da rede, este campo contém o número do nó correspondente ao hospedeiro; caso contrário é zero.

DEVCTN: número do canal lógico. Atribuído dinamicamente durante o processo de conexão (estabelecimento do canal lógico) deste equipamento.

DEVSTS: palavra de estado local do equipamento

bit Ø . (STSBIT) se ligado, indica que deve ser envia-

da uma mensagem de estado.

1. (TTYBIT) se ligado, indica que o equipamento é um terminal.
2. (OUTBIT) se ligado, indica que o equipamento tem saída (ou seja, necessita DATAREQUEST).

bit 3. (ACPBIT) se ligado, indica que é necessário enviar um CONNECT CONFIRM (se CONBIT está ligado) ou um DISCONNECT (se CONBIT desligado).

bit 4. (CONBIT) se ligado, indica que o equipamento está conectado ou em processo de conexão.

bit 5. (TABBIT) se ligado, indica que está sendo simulada uma tabulação horizontal.

bit 6. (ACTBIT) se ligado, indica que o transmissor desta linha está ativo.

DEVDDC: palavra de estado do processo, com bits já definidos em 2.4.1.2.b.

DEVTIM: Temporizador para preenchimento. Quando um caracter que necessita preenchimento é enviado ao equipamento, as interrupções geradas pelo seu transmissor são temporariamente inibidas e DEVTIM é inicializado com LDBFIL [carac] (veja LDBIBF, pag.58). DEVTIM é então decrementado a cada tique do relógio e quando DEVTIM = Ø as interrupções do transmissor são novamente ativadas.

DEVDRQ: número de mensagens DATA REQUEST enviadas. Uma mensagem DATA REQUEST só é enviada se há espaço disponível no buffer de saída DEVOBF e se DEVDRQ é igual a zero.Incre

mentado a cada DATA REQUEST enviado e decrementado a cada mensagem de dados recebida.

DEVOBF: Buffer de saída. Usado para armazenar os caracteres recebidos do hospedeiro que devem ser enviados ao equipamento.

DEVOPT: Ponteiro de inserção do buffer de saída. Atualizado pelo controlador quando do recebimento de uma mensagem de dados.

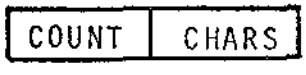
DEVOTK: Ponteiro de retirada do buffer de saída. Atualizado pelo controlador quando uma interrupção do transmissor deste equipamento é servida.

DEVOCN: contador de caracteres presentes no buffer de saída.

LDBIBF: Buffer de entrada do terminal. É onde são montadas as mensagens (nível de aplicação) que devem ser enviadas ao hospedeiro.

LDBIDX: ponteiro de inserção do buffer de entrada.

LDBIBC: ponteiro para o contador de caracteres de uma mensagem de dados (nível de aplicação) que está sendo montada em LDBIBF. Como o formato da mensagem de dados é



e a mensagem é montada à medida que o usuário digita os caracteres no terminal, o campo COUNT tem que ser atualizado a cada caracter recebido, o que é feito utilizando-se este ponteiro.

LDBWID: número de colunas deste terminal. Recebido no campo TTYWID de mensagens CHARACTERIST (2.4.1.2.a).

LDBCOL: coluna em que foi impresso o último caracter. Quando LDBCOL = LDBWID uma sequência <CR><LF> é impressa antes do próximo caracter se o bit 6 de DEVDDC está desligado.

LDBXPT: ponteiro de mensagens, aponta para cadeia de caracteres que devem ser impressos ou NIL. Tem precedência sobre LDBFPT.

LDBFPT: ponteiro de mensagens auxiliar, aponta para cadeia de caracteres que devem ser impressos ou NIL. Tem precedência sobre LDBCHR.

LDBCHR: caracter recebido do terminal e que deve ser ecoado.Tem precedência na transmissão sobre os caracteres presentes no buffer de saída DEVOBF.

LDBEPL: marcador de eco. Usado em mensagens marcador de eco (2.4.1.2.d).

LDBFIL: tempos de preenchimento. Alguns caracteres necessitam um tempo de preenchimento para que o terminal complete sua impressão, sob pena desse caracter ser superposto. LDBFIL é um vetor com o valor do tempo de preenchimento necessário para cada caracter (<BSPACE> , <TAB>,<FFEED>, < VTAB > ,<LF> ,< CR >). Os valores são recebidos no campo FILLERS de mensagens CHARACTERIST (2.4.1.2.a).

3.2.2.1.a A Rotina RCVTIQ

A função da rotina RCVTIQ é tratar os caracteres recebidos do terminal, que foram alocados temporariamente na fila TTIQUE pela rotina do serviço de interrupção INTARS (3.2.1.4).A RCVTIQ é chamada pelo Rastreador (3.2.2.5) quando este está esvaziando a fila TTIQUE. O Rastreador pega um elemento da fila TTIQUE (número da linha e caracter), ativa o processo controladir de terminal associado à linha e passa a este o caracter recebido, chamando então RCVTIQ.

A rotina RCVTIQ faz o tratamento do caracter, montando as mensagens necessárias (nível de aplicação) no "buffer"LDBIBF. As mensagens montadas são posteriormente coletadas e encaminhadas pelo Empacotador de Mensagens.

O algoritmo utilizado é descrito, resumidamente, a seguir.

Algoritmo RCVTIQ

1. (Terminal conectado?) Se o terminal não está conectado à rede, ignora o caracter, avisa o usuário (aponta LDBXPT para uma mensagem) e vai para 9.
2. (Verifica caracter) Altera ou não o caracter para maiúscula, dependendo da palavra DEVDDC do Descritor de Terminal.
3. (Buffer cheio?) Se o "buffer" LDBIBF está completo, desconsidera o caracter, ecoa a campainha para avisar o usuário (faz LDBXPT apontar para mensagem campainha) e vai para 9.
4. (Ainda eco adiado?) Se alguma das seguintes condições ocorrer

a. DEVOBF não vazio, significando que há caracter enviado pelo hospedeiro esperando ser transmitido ao terminal

b. LDBCHR não nulo, signficando que há um caracter ainda esperando ser ecoado

c. o caracter exige eco adiado (<CTRL-C> , por exemplo), vá para 7.

6. (Eco local) Se o caracter não é <CR>, coloca o caracter em LDBCHR e na mensagem e vai para 9 . Se o caracter é <CR>,aponta LDBFPT para a cadeia <CR> <LF> .
7. (Entra em eco adiado) Liga o bit indicador de eco adiado em DEVDDC e concatena a essa mensagem uma mensagem de estado com a nova palavra de estado. Inicia uma nova mensagem de dados (concatenada à mensagem de estado).
8. (Já em eco adiado) Coloca o caracter na mensagem de dados e retorna.
9. (Ativa transmissor) Se ACTBIT em DEVSTS está desligado,ativa o transmissor desta linha e liga ACTBIT.
10. (Retorna) Retorna.

3.2.2.1.b A rotina RCVTTY

A rotina RCVTTY recebe do Desempacotador de Mensagens um conjunto de mensagens (nível de aplicação) para um processo controlador de terminal e faz o tratamento dessas mensagens.Conforme descrito em 2.4.1, as mensagens recebidas por um terminal podem ser de três tipos: Dados, Estado ou Controle.

O algoritmo utilizado é descrito, resumidamente,a seguir.

Algoritmo RCVTTY

1.(Há mensagem?) Se há mensagem, retira CONTADOR e TIPO da mensagem; se não há mais mensagens, retorna.

2.(Testa TIPO) Se TIPO = Dados, vai para 3; se TIPO = Estado vai para 5; se TIPO = Controle, vai para 6.

3.(Guarda Carac) Pega o próximo carac da mensagem e coloca em DEVOBF; atualiza DEVOPT e DEVOCN.

4.(Repete?) Decrementa CONTADOR e, se diferente de zero, vai para 3; se igual a zero terminou esta mensagem. Dá partida ao terminal (liga ACTBIT se desligado e ativa transmissor) e vai para 1.

5.(Msg Status) Pega subtipo da mensagem (SET/RESET) e bits a ligar/desligar, atualiza DEVDDC e vai para 1 (terminou esta mensagem).

6.(Msg Controle) Retira TIPO de mensagem de controle. Se TIPO= =ECHO-PIPELINE-MARKER, vai para 7; se TIPO = CHARACTER GOBBLER vai para 8; se TIPO = CHARACTERISTIC, vai para 9.

7.(Marcador de Eco) Retira MARKER da mensagem. Se MARKER =LDBEPL e LDBIBF está vazio, desliga bit eco adiado em DEVDDC e liga bit envie status em DEVSTS. Vai para 1 e pegar outra mensagem.

8.(Papa-Carac) Limpa o "buffer" DEVOBF, reinicializando DEVOTK, DEVOPT e DEVOCN. Vai para 1 pegar outra mensagem.

9.(Características) Guarda os tempos de preenchimento recebidos em LDBFIL e em LDBWID o número de colunas recebido (TTYWID). Vai para 1 pegar outra mensagem.

3.2.2.1.c A Rotina XMTTY

A rotina XMTTY é a responsável pela transmissão de

caracteres ao terminal. Ela é executada no nível de interrupção, chamada pela rotina de serviço de interrupção das interfaces assíncronas (INTATS). Uma descrição resumida do algoritmo utilizado é dada a seguir.

Algoritmo XMTTY

1. (Simulado Tab?) Se o bit simulação de <TAB> está ligado em DEVSTS, incrementa LDBCOL e vai para 14.
2. (Esperando Preenchimento?) Se DEVTIM diferente de zero,inibe o transmissor desta linha e retorna.
3. (Há saída XPT?) Se o ponteiro LDBXPT é diferente de NIL,pega o CARAC apontado, avança LDBXPT e vai para 9.
4. (Há saída FPT?) Se o ponteiro LDBFPT é diferente de NIL,pega o CARAC apontado, avança LDBFPT e vai para 9.
5. (Verifica final de linha) Se o bit indicando que o terminal tem tratamento automático de <CR> está desligado em DEVDDC, compara LDBCOL com LDBWID e se iguais, aponta LDBFPT para a cadeia <CR><LF>, faz LDBCOL igual a zero e vai para 4.
6. (Caracter esperando ser ecoado?) Se LDBCHR não nulo, faz CARAC ← LDBCHR, limpa LDBCHR e vai para 9.
7. (Caracter enviado pelo hospedeiro?) Se há caracter em DEVOBF pega CARAC, avança DEVOTK, atualiza DEVOCN e vai para 9.
8. (Nada a fazer) Desliga ACTBIT em DEVSTS, inibe o transmissor desta linha e retorna.
9. (Início de simulação de <TAB>?) Se CARAC é <TAB> e o terminal não tem <TAB> por hardware nem está em modo imagem, liga bit simulação de <TAB> em DEVSTS, incrementa LDBCOL e vai para 15.

10. (Caracter precisa preenchimento?) Se CARAC necessita preenchimento, consulta LDBFIL e coloca em DEVTIM o tempo necessário.
11. (Movimento horizontal?) Se CARAC implica em movimento horizontal, incrementa LDBCOL.
12. (Caracter é <CR>?) Se o CARAC = <CR>, faz LDBCOL← Ø.
13. (Imprime) Envia CARAC ao terminal e retorna.
14. (Final de simulação?) Se LDBCOL é múltiplo de 8 (final de < TAB>), desliga bit simulação de <TAB>em DEVSTS.
15. (Simula <TAB>) Faz CARAC←<espaço>, envia CARAC ao terminal e retorna.

3.2.2.2 Controlador da Impressora

O Controlador da Impressora é o processo que faz o atendimento da impressora da estação remota. As suas tarefas são

(i) interpretação de mensagens (nível de aplicação) recebidas do processo correspondente no hospedeiro

(ii) transmissão de caracteres à impressora.

A tarefa (i) é executada pela rotina RCVLPT, que recebe do Desempacotador uma mensagem no nível de aplicação e faz seu tratamento. A tarefa (ii) é executada pela rotina XMTLPT, acionada a nível de interrupção pela rotina do serviço de interrupção do transmissor das interfaces assíncronas (INTATS), quando esta atende uma interrupção do transmissor da impressora. As rotinas RCVLPT e XMTLPT serão descritas nas seções seguintes.

A área de dados do processo controlador de impressora

é denominada Descritor de Impressora, que é análogo aos Descritores de Terminais. São apresentados a seguir apenas os campos do Descritor de Impressora que diferem dos apresentados nos Descritores de Terminais.

DEVADR: endereço físico da impressora, usado para entrada/saída de dados e palavras de controle

DEVOBT: tipo do equipamento (3 para LPT)

DEVDDC: palavra de estado do processo, com os bits definidos em 2.4.2.2

LPTLIN: número da linha corrente, utilizado para controle do carro (mudança de página, movimentos especiais, etc)

LPTCHR: caracter comprimido; deve ser impresso LPTCCN vezes. Os caracteres recebidos do hospedeiro são comprimidos segundo a convenção apresentada em 2.4.2.1

LPTCCN: contador de impressão de caracteres comprimidos

LPTCNS: contador de <LF>,utilizado na impressão de caracteres especiais (controle de carro)

LPTFIL: tempos de preenchimento (veja LDBFIL nos descritores de terminais).

3.2.2.2.b A Rotina RCVLPT

A rotina RCVLPT é acionada pelo Desempacotador de Mensagens quando uma mensagem de dados para a impressora é recebida. A função de rotina RCVLPT é simplesmente armazenar em DEVOBF (no descritor da impressora) os caracteres recebidos, que serão depois transmitidos à impressora.

Algoritmo RCVLPT

1. (Início) Retira CONTADOR e TIPO de mensagem
2. (Guarda carac) Pega carac na mensagem e guarda em DEVOBF. Atualiza DEVOCN e DEVOPT, decrementa CONTADOR.
3. (Terminou?) Se CONTADOR = Ø, dá partida à impressão (se ACTBIT desligado em DEVSTS, ativa o transmissor da impressora e liga ACTBIT) e retorna; se CONTADOR ≠ Ø, volta para 2.

3.2.2.2.c A Rotina XMTLPT

A rotina XMTLPT é a responsável pela transmissão dos caracteres à impressora. Ela é executada no nível de interrupção, acionada pela rotina do serviço de interrupção do transmissor das interfaces assíncronas (INTATS). A descrição resumida do algoritmo utilizado é dada a seguir.

Algoritmo XMTLPT

1. (Esperando Preenchimento?) Se DEVTIM ≠ Ø, inibe o transmissor desta linha e retorna.
2. (Imprimindo caracter especial?) Se LPTCNS ≠ Ø, vai para 14.
3. (Descomprimindo caracter?) Se LPTCCN = Ø, vai para 5.
4. (Descomprime caracter) Faz CARAC ← LPTCHR, decrementa LPTCCN e vai para 11.
5. (Há mais caracter?) Se não há mais caracter em DEVOBF, inibe o transmissor desta linha, desliga ACTBIT em DEVSTS e retorna
6. (Pega caracter) Pega CARAC em DEVOBF; atualiza DEVOTK e DEVOCN

7. (É caracter simples?) Se CARAC não é compressão vai para 11.

8. (É branco comprimido?) Se CARAC é branco comprimido, faz LPTCHR ← <BRANCO>, LPTCCN ← (CARAC AND 3FH)-1, e vai para 19.

9. (É caracter comprimido?) Se CARAC é caracter comprimido, faz LPTCCN ← (CARAC AND 1FH)-1, pega CARAC em DEVOBF, atualiza DEVOTK e DEVOCN e vai para 11.

10. (Erro) Problemas com o protocolo do nível de aplicação, avisa o operador e retorna.

11. (Movimento especial?) Se CARAC não exige movimento especial vai para 15.

12. (Procura próxima linha) Descobre o número de <LF> até a linha especificada pela caracter especial e guarda essa informação em LPTCNS.

13. (Topo de página?) Se a próxima linha é topo de página, limpa LPTCNS, faz CARAC ← <FFEED> , LPTLIN ← Ø e vai para 19.

14. (Imprime caracter especial) Faz CARAC ← <LF> e decrementa LPTCNS.

15. (Precisa preenchimento?) Se CARAC precisa preenchimento, faz DEVTIM ← LPTFIL [CARAC].

16. (Caracter é <LF>?) Se CARAC = <LF>, incrementa LPTLIN.

17. (Fim de página?) Se LPTLIN = LPTLEN (comprimento da página da impressora) faz CARAC ← <F FEED> , LPTLIN ← Ø e vai para 19.

18. (Caracter é <F FEED>?) Se CARAC = <F FEED>, faz LPTLIN ← Ø.

19. (Envia caracter) Envia CARAC à impressora e retorna.

3.2.2.3 Desempacotador de Mensagens

A função do Desempacotador de Mensagens é processar os protocolos dos níveis de enlace e de rede das mensagens recebidas do hospedeiro. As mensagens são recebidas pela rotina do serviço de interrupção do receptor da interface síncrona (INTSRS) e estocadas na fila RCVQUE. Depois de "descascada" dos protocolos DDCMP e NCL, a mensagem resultante (nível de aplicação)é entregue ao controlador destinatário (Terminais ou Impressora).

O Desempacotador de Mensagens é composto de duas rotinas principais

a. RCVCHK, que processa o protocolo DDCMP

b. RCVNCL, que processa o protocolo NCL

3.2.2.3.a A Rotina RCVCHK

A rotina RCVCHK processa o protocolo DDCMP das mensagens enviadas pelo hospedeiro. Para manter o controle sobre o estado do protocolo, as seguintes variáveis são utilizadas pelo Desempacotador de Mensagens (e compartilhadas pelo Empacotador de Mensagens):

STACKF: igual a 2, enviando mensagens START
1, enviando mensagens STACK
Ø, DDCMP está rodando.

NAKFLG: maior que zero, deve enviar NAK com razão NAKFLG;
menor que zero, deve enviar ACK.

REPFLG: se positivo, deve enviar mensagem REP.

REPTCK: Temporizador para time-out DDCMP; decrementado a cada

tique do relógio e inicializado a cada mensagem de dados enviada.

RECVOK: número da última mensagem recebida e aceita.

TIØLAN: número da última mensagem reconhecida pelo hospedeiro

TIØHSN: número da última mensagem enviada.

BCRCNT: número de mensagens recebidas com CRC errado.

O algoritmo utilizado em RCVCHK é descrito resumidamente a seguir. Algoritmo RCVCHK

1. (Pega Msg) Retira a mensagem da fila RCVQUE e avança RCVTKR (ver 3.2.1.1).
2. (Verifica Cabeçalho) Calcula o CRC do cabeçalho DDCMP (ver apêndice A), compara com o recebido e, se não confere, devolve os chunks utilizados, se houver, e retorna (a "mensagem" é descartada).
3. (Despacha) Trata mensagem de acordo com seu tipo (passos de 4 a 8).
4. (Recebeu ACK ou NAK) Utiliza a informação do campo RESP para atualizar os ponteiros TIØTKR e TIØAKR na fila de transmissão (ver 3.2.1.2) e a variável TIØLAN; retorna.
5. (Recebeu REP) Faz NAKFLG ← 3 (Razão para NAK é a recepção de REP) e retorna.
6. (Recebeu START) Faz STACKF ← 1 e retorna.
7. (Recebeu STACK) Faz STACKF, RECVOK, REPFLG ← Ø, inicializa REPTCK e retorna (DDCMP está no ar).

8. (Recebeu Msg Dados) Utiliza a informação do campo RESP para atualizar os ponteiros T1ØTKR , T1ØTKR e a variável T1ØLAN.
9. (Verifica CRC). Calcula o CRC do campo DATA e compara com o recebido em BLKCK3; se diferentes, incrementa BCRCNT, devolve os chunks utilizados nesta mensagem ao reservatório e retorna.
10. (Terminou DDCMP) Incrementa RECVOK e chama RCVNCL para tratar o protocolo NCL; devolve os chunks utilizados nesta mensagem e retorna.

3.2.2.3.b A Rotina RCVNCL

A rotina RCVNCL é a que trata o protocolo do nível de rede (NCL) das mensagens recebidas do hospedeiro.

Na descrição resumida do algoritmo utilizado, dada a seguir, são usadas as seguintes variáveis:

CTRLTB: tabela com DEVN entradas (DEVN = nº de terminais + nº impressoras da estação). A entrada é o número do canal lógico associado a um processo controlador (de terminal ou impressora) e o conteúdo da tabela é o endereço da área de dados (Descritor) desse processo.

RNODE: número do nó hospedeiro

DISSNA: se maior que zero indica que uma mensagem DISCONNECT deve ser enviada ao canal lógico número DISSLA do nó DISSNA

CONSNA: se maior que zero indica que uma mensagem CONFIGURATION deve ser enviada ao nó CONSNA.

Os nomes sublinhados (DNA, por exemplo) se referem aos

campos das mensagens NCL descritos em 2.3.

Algoritmo RCVNCL

1. (Despacha) Trata as mensagens de acordo com seu tipo (dados: passo 2; controle não numerado: passo 3; controle numerado: passos 4 a 7)

2. (Recebeu Msg Dados) Acha o endereço da área de dados (Descritor) do processo associado ao canal lógico destino em CTRLTB [DLA] e ativa o processo. Se o processo é controlador de terminal, chama RCVTTY, desativa o processo e retorna. Se o equipamento é impressora, chama RCVLPT, desativa o processo e retorna.

3. (Recebeu Node ID) Faz RNODE ← SNA e retorna.

4. (Recebeu REQUEST CONFIGURATION). Faz CONSNA ← SNA, indicando que uma mensagem CONFIGURATION deve ser enviada, e retorna.

5. (Recebeu NEIGHBOUR) Desconecta e inicializa os descritores de todos os processos controladores que estejam conectados a nós não presentes em NEILIST e retorna.

6. (Recebeu DISCONNECT). Acha o endereço do descritor do processo associado ao canal lógico destino em CTRLTB [DLA]. Ativa o processo, desliga CONBIT e liga ACPBIT em DEVSTS, indicando que deve ser enviada uma mensagem DISCONNECT CONFIRM.Desativa o processo e retorna.

7. (Recebeu CONNECT) Procura uma ENTRADA livre em CTRLTB; se todas estão ocupadas, faz DISSLA ← SLA, DISSNA ← SNA e retorna. Utiliza as informações em DPO e DPN para ativar o processo

destino. Faz CTRLTB [ENTRADA]←endereço do descritor, DEVCTN← ENTRADA, DEVDRQ ← Ø , DEVDDC ← Ø, liga os bits CONBIT, ACPBIT e STSBIT em DEVSTS. Desativa o processo e retorna.

3.2.2.4 Empacotador de Mensagens

O Empacotador de Mensagens é o bloco responsável pelos protocolos dos níveis de ligação (DDCMP) e de rede (NCL) das mensagens enviadas pela estação remota ao hospedeiro. É constituído de duas rotinas principais: XSTART, que cuida do protocolo DDCMP, e XTNCL, que faz o tratamento do protocolo NCL.

3.2.2.4.a A Rotina XSTART

A rotina XSTART é acionada pelo Rastreador quando o transmissor síncrono está inativo e sua função é enviar ao hospedeiro as mensagens de que o protocolo DDCMP necessita.

As mensagens de controle DDCMP são montadas no buffer CTRBUF, e as mensagens de dados DDCMP são montadas na fila TIØBUF.

O algoritmo é descrito resumidamente a seguir. As variáveis utilizadas na descrição estão definidas em 3.2.1.2 e 3.2.2.3.a, e os nomes sublinhados (por exemplo, RESP) representam os campos das mensagens DDCMP definidas em 2.2.

Algoritmo XSTART

1. (Inicializando?) Se STACKF = 2, vai para 8 (envia START); se STACKF = 1, vai para 9 (envia STACK).
2. (Precisa enviar NAK?) Se NAKFLG > Ø, vai para 10 (envia NAK).

3. (Tempo de REP?) Se REPFLG = 1, vai para 11 (envia REP).
4. (Precisa transmitir ou retransmitir msg?) Se os ponteiros T1ØTKR e T1ØPTR não são iguais, significando que hã uma mensagem pronta para ser transmitida, vai para 12.
5. (Montamos todas possíveis?) Se os ponteiros T1ØPTR e T1ØAKR não são iguais, significando que hã espaço disponível em T1ØBUF, chama XTNCL e vai para 15.
6. (Precisa enviar ACK?) Se NAKFLG = Ø, retorna (nada a fazer).
7. (Envia ACK) Monta uma mensagem ACK com RESP ← RECVOK e vai para 13.
8. (Envia START) Monta uma mensagem START e vai para 13.
9. (Envia STACK) Monta uma mensagem STACK e vai para 13.
10. (Envia NAK) Monta uma mensagem NAK com RESP ← RECVOK, REASON ← ←NAKFLG e vai para 13.
11. (Envia REP) Faz REPFLG ← Ø, inicializa REPTCK, monta uma mensagem REP com NUM ← T1ØHSN e vai para 13.
12. (Envia Msg Dados) Atualiza o cabeçalho, fazendo RESP ← RECVOK, e liga o temporizador REPTCK.
13. (Prepara transmissor) Se mensagem de controle, aponta T1ØCA para CTRBUF e faz T1ØBCN ← 8. Se mensagem de dados, faz T1ØCA ← ←T1ØTKR, T1ØBCN ← COUNT e avança T1ØTKR. Faz T1ØFLG ← Ø.
14. (Acorda transmissor) Ativa transmissor síncrono e retorna.
15. (Termina msg Dados) Se nenhuma mensagem NCL foi montada, retorna. Caso contrário, incrementa T1ØHSN, faz NUM ← T1ØHSN, acerta COUNT, avança T1ØPTR e retorna.

3.2.2.4.b A rotina XTNCL

A rotina XTNCL é a responsável pelo envio de mensagens do protocolo do nível de rede (NCL). É chamada pela rotina XSTART quando uma mensagem de dados DDCMP está sendo montada. Se não há nenhuma mensagem NCL a ser enviada, uma marca é providenciada no retorno de XTNCL.

As mensagens de dados NCL tem em seu campo DATA mensagens do protocolo do nível de aplicação, geradas pelos controladores de terminais e pelo controlador da impressora.

O algoritmo utilizado é descrito resumidamente a seguir. As variáveis utilizadas são definidas em 3.2.2.3.b, e os nomes sublinhados (por exemplo DLA) representam o conteúdo dos campos das mensagens NCL, descritos em 2.3.

Algoritmo XTNCL

1. (Precisa enviar DISCONNECT?) Se DISSNA > Ø, monta uma mensagem DISCONNECT com DLA ← DISSLA, SNA ← DISSNA, limpa DISSNA e retorna.
2. (Precisa enviar CONFIGURATION?) Se CONSNA > Ø, monta uma mensagem CONFIGURATION informando as características da estação remota e retorna.
3. (Verifica controlador) A partir do último controlador atendido, ativa um processo controlador.

4. (Hospedeiro conhece este controlador?) Se CONBIT e ACPBIT estão desligados em DEVSTS, vai para 10.

5. (Envia CONNECT) Se CONBIT e ACPBIT estão ligados, monta uma mensagem CONNECT CONFIRM com SLA ← DEVCNT, desliga ACPBIT, desativa o controlador e retorna (controlador conectado).

6. (Envia DISCONNECT) Se ACPBIT está ligado mas CONBIT está desligado, monta uma mensagem DISCONNECT CONFIRM, desliga ACPBIT, faz CTRLTB [DEVCNT] ← Ø, DEVCNT ← Ø, desativa o controlador e retorna (processo controlador desconectado).

7. (Controlador é de impressora?) Se este controlador é de impressora vai para 9.

8. (Envia mensagem de terminal) Se LDBIBF não está vazio, monta uma mensagem de dados com o conteúdo de LDBIBF e limpa LDBIBF. Se o terminal está em eco adiado, incrementa LDBEPL e monta uma mensagem marcador de eco com MARKER ← LDBEPL.

9. (Envia DATA REQUEST) Se há espaço em DEVOBF e DEVDRQ = Ø, monta uma mensagem DATA REQUEST com SLA ← DEVCTN, incrementa DEVDRQ, desativa o controlador e retorna.

10. (Outro controlador?) Desativa este controlador. Se há algum controlador ainda não verificado desta vez, ativa esse controlador e vai para 4. Caso contrário retorna avisando que nenhuma mensagem NCL foi montada.

3.2.2.5 O Rastreador

A função do Rastreador é apenas descobrir e distribuir o trabalho a fazer. É o corpo do bloco de Serviços Gerais, sendo executado continuamente após a inicialização da estação remota. O processo de inicialização limpa todos os "buffers", variáveis e descritores de equipamento, monta o reservatório de chunks, reinicializa todas as interfaces (síncrona e assíncronas), faz STACKF ← 2 e monta uma mensagem NODE ID (NCL) em TIØBUF.

O algoritmo utilizado pelo Rastreador é descrito a seguir.

Algoritmo Rastreador

1. (Relógio acordou?) Se CLKFLG = 1 (veja 3.2.1.3) vai para 6.
2. (Verifica fila de caracteres dos terminais) Se a fila TTIQUE está vazia, vai para 3. Senão retira um elemento da fila (número da linha e caracter), ativa o processo controlador associado à linha e chama RCVTIQ. Desativa o controlador, avança o ponteiro TIQTKR e volta a 1 para mais trabalho.
3. (Verifica fila de msgs recebidas) Se a fila RCVQUE é não vazia, chama RCVCHK e volta a 1 para mais trabalho.
4. (Transmissor Inativo?) Se o transmissor síncrono está inativo (TIOFLG = Ø) chama XSTART e volta a 1 para mais trabalho.
5. (Tique do relógio) Desliga CLKFLG e SECFLG. Se REPTCK ≠ Ø, decrementa REPTCK; se o resultado é zero, faz REPFLG ← 1 (é tempo de enviar REP). Percorre todos os descritores de equipamentos (terminais e impressora) que necessitam tempos de preenchimento, decrementando DEVTIM se DEVTIM ≠ Ø; se o resul

tado é zero, ativa o transmissor do equipamento. Faz CLKCNT ← 6Ø e volta a 1 para mais trabalho.

CAPITULO 4

RESULTADOS E CONCLUSÕES

Um protótipo da estação DC72/88, descrita no capítulo 3, foi construído no Laboratório de Microcomputadores do Centro de Computação da Unicamp.

Uma pequena alteração no "software" foi efetuada para permitir a avaliação do comportamento do protótipo. Assim, foi incluído um laço "nada-a-fazer" ao fim do Rastreador, que não mais fica procurando trabalho constantemente. Uma variável WRKFLG foi também introduzida e é continuamente testada no laço nada-a-fazer para saber se o Rastreador deve ou não iniciar a procura de trabalho. É função das rotinas de interrupção alterar WRKFLG quando há trabalho a ser feito pelo nível de Serviço Geral (o serviço do relógio altera WRKFLG a cada interrupção, o serviço do receptor síncrono ao término da recepção de cada mensagem e o serviço do transmissor síncrono ao término de cada mensagem transmitida).

O laço nada-a-fazer foi construído para, na ausência de outras tarefas, emitir um sinal sonoro pela console (terminal número Ø) a cada 60 segundos. Medindo-se o atraso na emissão do sinal sonoro em cada condição de carga do sistema tem-se uma avaliação aproximada do seu comportamento.

Os gráficos da Fig.4.1 ilustram o desempenho conseguido pelo protótipo apresentando o nível de degradação do sistema da estação remota para quatro velocidades diferentes da linha de comunicação síncrona: 2400, 4800, 9600 e 19200 bps. Na figura,

"inativo" significa o estado em que nenhum terminal da remota está imprimindo ou recebendo caracteres; neste caso o tempo gasto pelo sistema é com a procura de tarefas pelo Rastreador, atendimento ao relógio e atendimento ao receptor da linha síncrona (infelizmente os caracteres de sincronismo não são desprezados automaticamente pela interface, gerando uma interrupção). O "número de terminais" indica o número de terminais que está continuamente imprimindo caracteres na tela, à velocidade de 1200 bps cada um.

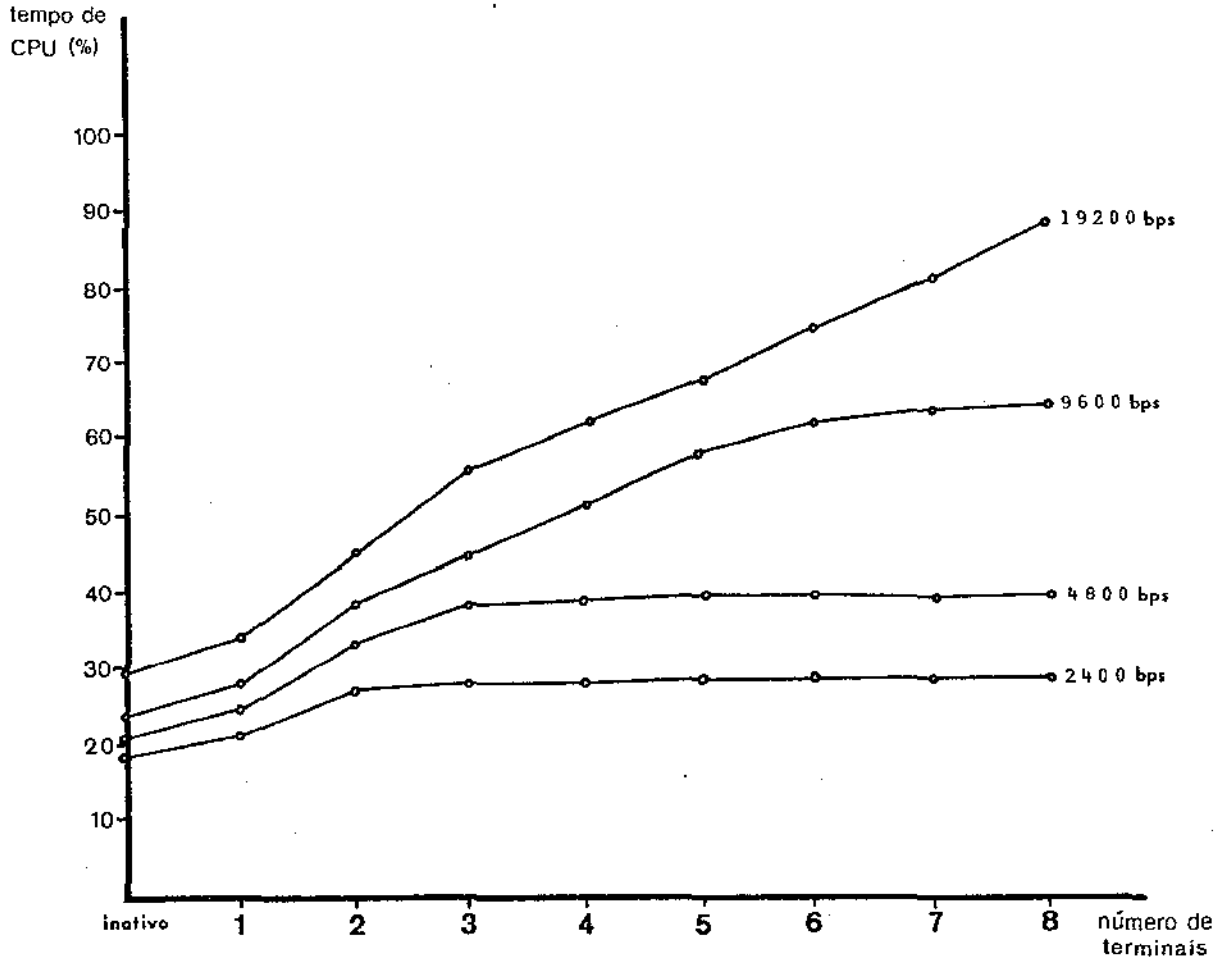


Fig.4.1. Desempenho do protótipo

A carga representada pela impressora, não incluída na figura, é menor, em cada configuração, do que a representada por mais um terminal (pelo fato de seus caracteres serem compactados).

Logicamente há um patamar na porcentagem de tempo de CPU utilizada pelo sistema quando a velocidade conjunta das linhas assíncronas (dos terminais) se equipara à velocidade de linha de comunicação síncrona, pois a partir desse ponto a linha síncrona se transforma em um "gargalo" para a troca de mensagens. Os terminais, nesse caso, não ficam imprimindo continuamente,mas não por falta de atendimento do sistema, e sim por falta de caracteres a serem impressos (que não foram ainda enviados pelo hospedeiro pelo congestionamento da linha síncrona).

Esses resultados mostram que, conforme esperado, a velocidade da linha síncrona é o principal parâmetro do desempenho do sistema. Um maior número de terminais na remota apenas deslocaria para cima as curvas da Fig.4.1, conservando as suas formas. Esse fato permite estimar que a estação remota DC72/88 poderá suportar até 16 terminais, desde que a comunicação síncrona seja feita a velocidades até 9600 bps, o que é bastante razoável, considerando-se o alto custo de modems síncronos de alta velocidade.

Para velocidades maiores de comunicação síncrona o aumento do número de terminais suportados só seria possível com algumas modificações do "hardware" da estação DC72/88. A inclusão, por exemplo, de uma pastilha 8274-INTEL (Multi-Protocol Se

rial Controller), que é uma pastilha com duas linhas independentes de comunicação com capacidade de cálculo automático de CRC, associado à pastilha 8257-INTEL (Programmable DMA Controller), certamente contribuiriam bastante para diminuir a quantidade de tempo de CPU utilizado.

Essa nova configuração permitiria, também, a evolução a um novo tipo de estação remota, mais complexa que a DC72NP.Esse novo tipo poderia se aproximar bastante da estação DN82, que é uma estação remota não terminal, ou seja, capaz de efetuar o roteamento de mensagens, passando a ser também um nó intermediário da rede.

As estações DN82 fornecidas pela DEC são constituídas de um PDP-11, possibilitando a ligação de até 32 terminais, uma impressora e uma leitora de cartões. Provavelmente não se conseguirá, com um microprocessador 8088A, de 8 bits, reproduzir a eficiência atingida por PDP11, de 16 bits. Mas restringindo-se o número de terminais aos oito presentemente suportados, o desempenho do sistema deverá ser satisfatório.

O protótipo utiliza aproximadamente 6 Kbytes de EPROM, para código (cerca de 2000 instruções) e tabelas fixas, e 5 Kbytes de RAM para dados. O "software" da estação remota DC72NP desenvolvido por Dave McLure e Kalmam Reti para a DEC (e que roda em um PDP8, com palavra de 12 bits), utiliza cerca de 4 Kpalavras (6 Kbytes) para código (4000 instruções) e 4 Kpalavras (6 Kbytes) para dados.

Se por um lado o fato de o PDP8 ter palavra de 12 bits

permite um código mais compacto (toda instrução tem comprimento único de uma palavra), o fato de ser um minicomputador desenvolvido em meados da década de 60 o coloca em desvantagem quanto à velocidade e ao repertório de instruções quando comparado com os microprocessadores atuais. Assim, o uso do 8088A permitiu que a programação ficasse bastante mais estruturada, consequentemente tornando mais fácil a elaboração e possíveis futuras alterações no programa, e diminuiu praticamente à metade o número de instruções (embora o tamanho do código não tenha se alterado muito).

Outra desvantagem do PDP8 é o fato de que todos os protocolos envolvidos na comunicação remota-hospedeiro são byte orientados, e todos os dados trocados entre a remota e os terminais (ou impressora) são caracteres ASCII de sete bits. Como a palavra do PDP8 é de 12 bits, isso acarreta uma ineficiência muito grande no armazenamento desses dados, deixando sempre um terço dos bits de cada palavra de dados sem utilização.

Como sugestão para desenvolvimento futuro, além das já mencionadas, estaria a inclusão de um novo nível na comunicação remota-hospedeiro, entre o nível físico e o de enlace, que possibilitasse a conexão remota-hospedeiro através da Rede Nacional de Comutação de Pacotes, a ser implantada pela Embratel até 1985. Esse novo nível seria compatível com o protocolo X.25 da CCITT (Comitê Consultivo Internacional para Telegrafia e Telefonia), e executaria as funções de um PAD (Packet Assembly/Disassembly, recomendação X.3 da CCITT), que é um dispositivo empacotador/desempacotador que permite a um usuário o acesso à rede co-

mudada. Com a modificação já sugerida no "hardware" esse novo nível de protocolo seria possível de ser implementado, já que a pastilha 8274A oferece certas facilidades para comunicação com protocolos bit-orientados, como é o caso do X.25.

APENDICE A

CALCULO DO CRC

O método de cálculo do CRC, usado na detecção de erros em mensagens DDCMP, mereceu uma atenção especial, já que o CRC deve ser calculado para cada byte transmitido ou recebido pela interface síncrona, representando uma parcela razoável do tempo de processamento da comunicação.

Para evitar que fosse necessário adicionar à estação "hardware" específico para o cálculo de CRC, optou-se pela utilização de um algoritmo especial, bastante rápido. Esse algoritmo é chamado multi-bit com auxílio de tabela [LEE] e, embora utilize uma quantidade de memória adicional, é bem mais rápido que o tradicional cálculo bit a bit.

Neste apêndice são apresentados e comparados esses dois métodos de cálculo de CRC.

A.1 O método CRC

O método de detecção de erros CRC (Cyclic Code Redundancy) baseia-se nas propriedades da divisão de polinômios. Suponha que uma mensagem a ser transmitida seja composta de k bits. Podemos representar essa mensagem como um polinômio em x, com k termos; representando os bits da mensagem pelos coeficientes a_{k-1}, a_{k-2},...a_2,a_1, a_0, o polinômio fica sendo

$$M(x) = a_{k-1}\, x^{k-1} + a_{k-2}\, x^{k-2} + \ldots a_2 x^2 + a_1 x + a_0.$$

Por exemplo, se a mensagem a ser enviada e 10011 o polinômio que a representa é

$$M(x) = 1.x^4 + 0.x^3 + 0.x^2 + 1.x + 1.$$

Pelo método CRC, para transmitir a mensagem M(x) é preciso um outro polinômio P(x), denominado polinômio gerador. P(x) tem grau r , com r maior que zero e menor que o grau M(x).

O algoritmo do método é dado a seguir.

Algoritmo CRC (transmissão)

1. (Multiplica) A mensagem M(x) é multiplicada por x^r, resultando r zeros nas posições de mais baixa ordem.
2. (Divide por P) O resultado é dividido por P(x), dando um quociente Q(x) e um resto R(x)

$$\frac{x^r \cdot M(x)}{P(x)} = Q(x) \oplus \frac{R(x)}{P(x)} \qquad (i)$$

onde $\oplus$ representa adição em módulo 2 (ou-exclusivo)

3. (Soma R) o resto é adicionado à mensagem, resultando

$$T(x) = x^r \cdot M(x) \oplus R(x) \qquad (ii)$$

4. (Transmite) Transmite T(x).

Portanto, transmite a mensagem original M(x) seguida por r bits, que correspondem a R(x). Pode-se agora reescrever a equação (i)

$$x^r \cdot M(x) = Q(x) \cdot P(x) \oplus R(x)$$

Como a adição em módulo 2 é equivalente à subtração em módulo 2, temos

$$x^r . M(x) \oplus R(x) = Q(x) . P(x)$$

Comparando essa equação com (ii), vemos que

$$T(x) = Q(x) . P(x),$$

ou seja, a mensagem transmitida T(x) é divisível por P(x). E esta é a propriedade utilizada pelo método CRC. O algoritmo da recepção fica sendo portanto

Algoritmo CRC (recepção)

1. (Recebe) Recebe a mensagem T(x).
2. (Divide) Divide T(x) por P(x).
3. (Resto é zero?) Se o resto é diferente de zero, ocorreu um erro na transmissão; se o resto é zero, ou a transmissão foi completada corretamente ou um erro não detectável ocorreu.

Logicamente a eficiência do método CRC é fortemente dependente do polinômio gerador. Para um polinômio gerador que possua o fator (x + 1) e um fator com três ou mais termos, a proteção conseguida é mostrada na tabela abaixo [MARTIN]:

Tipo de erro	Proteção
um único bit	total
dois bits (juntos ou separados)	total
número ímpar de bits	total
bloco com menos de (r+1) bits	total
bloco com (r+1) bits	$1-(\frac{1}{2})^{r-1}$ de probabilidade de detecção
bloco com mais de (r+1) bits	$1-(\frac{1}{2})^{r}$ de probabilidade de detecção.

Os últimos dois termos assumem uma probabilidade igual de ocorrência de qualquer padrão de erro. Como na prática alguns padrões de erro são mais frequentes que outros, alguns polinômios geradores de grau r são melhores que outros.

O polinômio gerador do CRC 16, utilizado no protocolo DDCMP é

$$x^{16} + x^{15} + x^2 + 1,$$

que pode ser também escrito como

$$(x+1)\ (x^{15} + x + 1)$$

ou seja, satisfaz as condições que permitem a proteção apresentada na tabela acima.

A.2 O Cálculo do CRC16 pelo método bit a bit

A implementação de divisão pelo polinômio gerador pode ser feita em "hardware" com a utilização de um conjunto de registradores deslizantes de 1 bit e de somadores módulo 2 (circuitos de ou-EXCLUSIVO). O número de registradores necessários é igual ao grau de P(x) e o número de somadores é igual ao número de bits 1 no divisor menos um.

O circuito da Fig.A.1 mostra o circuito para o cálculo do CRC16, pelo método bit a bit. Inicialmente os registradores são zerados; ao fim da recepção (em E) de todos os bits da mensagem a transmitir os registradores contêm o CRC; ao fim da recepção de todos os bits da mensagem recebida (incluindo os de CRC) o conteúdo dos registradores será zero se a verificação for bem sucedida.

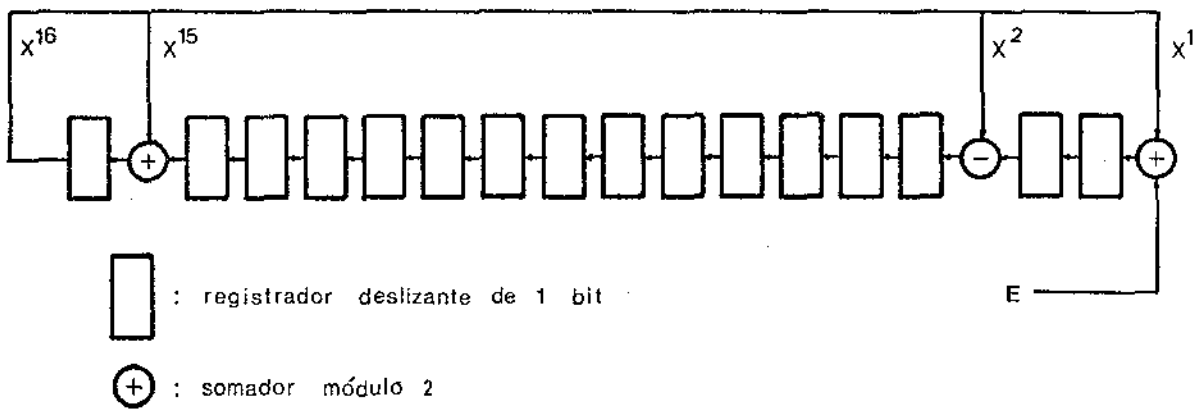


Fig.A.1. Circuito para cálculo do CRC16

A implementação em "software" da rotina para cálculo do CRC16 pelo método bit a bit utiliza como algoritmo uma simulação da operação do circuito da Fig.A.1. A codificação de uma rotina, usando a linguagem de montagem do 8088, é mostrada na Fig.A.2. Os números entre parênteses representam o número de clocks necessário para a execução de cada instrução. O tempo médio de execução dessa rotina é 378 clocks, que com cristal de 12MHz representa 31,5μs.

A.3 Cálculo do CRC16 pelo método multi-bit

O método multi-bit é derivado da verificação do que acontece com o CRC acumulado após o cálculo de mais um byte. A notação empregada a seguir representa entre parênteses a posição do bit.

Seja I o byte de entrada e C o valor do CRC já acumulado. A Fig.A.3 ilustra as modificações que ocorrem no CRC acumu-

```
                        INTEL 8086 CROSS ASSEMBLER  V2.2    07-JUL-83  AT 14:13:41       PROGRAM > CRC1          PAGE    1


ADDR        CODE          ERR LINE            STATEMENT

                            1
A001                        2  POLY    EQU     0A001H  ;POLINOMIO DO CRC16: X^16 + X^15 + X^2 + 1
                            3
                            4
                            5
                            6  ;       **********************
                            7  ;       * ROTINA: CRC
                            8  ;       * PARAMETROS, ENTRADA: AL <-- BYTE CUJO CRC SE DESEJA ACUMULAR
                            9  ;       *                       DX <-- CRC ACUMULADO
                           10  ;       *                SAIDA: DX <-- NOVO CRC ACUMULADO
                           11  ;       * DESTROI: FLAGS
                           12  ;       * DESCRICAO: A ROTINA CALCULA O CRC16 PELO METODO
                           13  ;       *             BIT A BIT, O POLINOMIO GERADOR E' DEFINIDO
                           14  ;       *             PELA CONSTANTE POLY.
                           15  ;       **********************
                           16
                           17
0000                       18  CRC:
0000 50                    19          PUSH    AX      ;(15)
0001 51                    20          PUSH    CX      ;(15)
0002 B1 08                 21          MOV     CL,8    ;(4)    ;VAI CALCULAR PARA 8 BITS
0004                       22  CRC1:
0004 D1 FA                 23          SHR     DX,1    ;(2)    ;O RESULTADO DE
0006 D0 D4                 24          RCL     AH,1    ;(2)    ;(BIT + 'A DIREITA CRC) OU-EXCL (BIT * 'A ESQ. BYTE)
0008 32 E6                 25          XOR     AH,AL   ;(3)    ;E' COLOCADO
000A D0 DC                 26          RCR     AH,1    ;(2)    ;NO CARRY
000C 73 04                 27          JNC     CRC2    ;(16/4)
000E 81 F2 01 A0           28          XOR     DX,POLY ;(4)    ;POLY E' O POLINOMIO GERADOR P(X)
0012                       29  CRC2:
0012 D0 E8                 30          SHR     AL,1    ;(2)    ;JA' VIMOS ESTE BIT
0014 E2 F6                 31          LOOP    CRC1    ;(17/5) ;ATE' TERMINAR O BYTE
0016 59                    32          POP     CX      ;(12)
0017 58                    33          POP     AX      ;(12)
0018 C3                    34          RET             ;(12)
0019                       35
```

Fig. A.2. CRC16 pelo método bit a bit.

Após o 1º bit	Após o 2º bit	Após o 3º bit	Após o 4º bit	Após o 8º bit
F(0)	F(1)	F(2)	F(3)	F(7)
C(15)	F(0)	F(1)	F(2)	F(6)
F(0)⊕C(14)	F(1) ⊕ C(15)	F(0) ⊕ F(2)	F(1) ⊕ F(3)	F(5) ⊕ F(7)
C(13)	F(0) ⊕ C(14)	F(1) ⊕ C(15)	F(0) ⊕ F(2)	F(4) ⊕ F(6)
C(12)	C(13)	F(0) ⊕ C(14)	F(1) ⊕ C(15)	F(3) ⊕ F(5)
C(11)	C(12)	C(13)	F(0) ⊕ C(14)	F(2) ⊕ F(4)
C(10)	C(11)	C(12)	C(13)	F(1) ⊕ F(3)
C(9)	C(10)	C(11)	C(12)	F(0) ⊕ F(2)
C(8)	C(9)	C(10)	C(11)	F(1) ⊕ C(15)
C(7)	C(8)	C(9)	C(10)	F(0) ⊕ C(14)
C(6)	C(7)	C(8)	C(9)	C(13)
C(5)	C(6)	C(7)	C(8)	C(12)
C(4)	C(5)	C(6)	C(7)	C(11)
C(3)	C(4)	C(5)	C(6)	C(10)
C(2)	C(3)	C(4)	C(5)	C(9)
F(0)⊕C(1)	F(1) ⊕ C(2)	F(2) ⊕ C(3)	F(3) ⊕ C(4)	F(7) ⊕ C(8)

Fig. A.3. Modificações no CRC acumulado

lado após o cálculo de cada bit da entrada I. Para maior clareza, as seguintes substituições foram feitas

$$F(0) = C(0) \oplus I(0)$$

$$F(i+1) = C(i+1) \oplus I(i+1) \oplus F(i)$$

Substituindo agora $C(i) \oplus I(i)$ por $T(i)$, a coluna "Após o 8º bit" da Fig.A.3 pode ser reescrita

$$
\begin{aligned}
&T(7) \oplus T(6) \oplus T(5) \oplus T(4) \oplus T(3) \oplus T(2) \oplus T(1) \oplus T(0)\\
&T(6) \oplus T(5) \oplus T(4) + (T3) \oplus T(2) \oplus T(1) \oplus T(0)\\
&T(7) \oplus T(6)\\
&T(6) \oplus T(5)\\
&T(5) \oplus T(4)\\
&T(4) \oplus T(3)\\
&T(3) \oplus T(2) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(iii)}\\
&T(2) \oplus T(1)\\
&T(1) \oplus T(0) \oplus C(15)\\
&\qquad T(0) \oplus C(14)\\
&\qquad\qquad C(13)\\
&\qquad\qquad C(12)\\
&\qquad\qquad C(11)\\
&\qquad\qquad C(10)\\
&\qquad\qquad C(9)
\end{aligned}
$$

$$T(7) \oplus T(6) \oplus T(5) \oplus T(4) \oplus T(5) \oplus T(4) \oplus T(3) \oplus T(2) \oplus T(1) \oplus T(0) \oplus C(8)$$

Assim, conseguimos expressar o resultado do CRC acumulado após a entrada I em termos de I e C, o que é a base do

método multi-bit. Uma tabela é então gerada, tendo como índice todos os valores de T possíveis (um total de 256 entradas). O conteúdo de cada entrada na tabela é uma palavra de 16 bits, calculados conforme (iii).

A descrição do algoritmo é dada a seguir. As seguintes variáveis são utilizadas

CRC: CRC acumulado, composto de CRCL e CRCH

CRCL: byte menos significativo do CRC acumulado

CRCH: byte mais significativo do CRC acumulado

CRCTAB: tabela do método multibit, já apresentada

I : byte de entrada

T : variável auxiliar.

Algoritmo Multi-bit

1. (Calcula índice) Faz T ← I ⊕ CRCL

2. (Prepara para consultar) Faz CRCL ← CRCH, CRCH ← Ø.

3. (Consulta tabela) Faz CRC ← CRC ⊕ CRCTAB [T] e retorna

O programa de Fig.A.4 foi usado para gerar a tabela da Fig. A.5. A codificação de uma rotina de cálculo do CRC16 pelo método multi-bit, em linguagem de montagem do 8088A pode ser vista na Fig.A.6. Os números entre parênteses indicam o número de "clocks" necessários para a execução de cada instrução.

O tempo de execução desta rotina é 102 "clocks" (com cristal de 12MHz representa 8,5μs), o que corresponde a pouco mais de 1/4 do tempo médio de execução da rotina do método bit a bit.

```
PASCAL COMPILATION LIST PRODUCED BY PASCAL VERSION FROM 30-DEC-76 ON 07-JUL-83  AT 14:18:12

   10
   20   PROGRAM TABCRC;
   30
   40   VAR      T: ARRAY[0..7] OF INTEGER;
   50            FIM: BOOLEAN;
   60
   70
   80   FUNCTION XOR(N,M:INTEGER):INTEGER;
   90   BEGIN
  100           IF N=1 THEN IF M=1 THEN XOR:=0
  110                               ELSE XOR:=1
  120                  ELSE IF M=1 THEN XOR:=1
  130                               ELSE XOR:=0
  140   END; (* XOR *)
  150
  160   PROCEDURE ADVANCE(N:INTEGER);
  170   BEGIN
  180           IF N=8 THEN FIM:=TRUE
  190                  ELSE IF T[N]=1 THEN BEGIN
  200                                        T[N]:=0;
  210                                        ADVANCE(N+1)
  220                                      END
  230                                 ELSE T[N]:=1
  240   END; (* ADVANCE *)
  250
  260   PROCEDURE IMPRIME;
  270   VAR      E0,E1,E2,E3,E4,E5,E6,E7,E8,E9,E10,E11,E12,E13,E14,E15: INTEGER;
  280
  290   BEGIN
  300           E6:=T[0];
  310           E7:=XOR(T[1],T[0]);
  320           E8:=XOR(T[2],T[1]);
  330           E9:=XOR(T[3],T[2]);
  340           E10:=XOR(T[4],T[3]);
  350           E11:=XOR(T[5],T[4]);
  360           E12:=XOR(T[6],T[5]);
  370           E13:=XOR(T[7],T[6]);
  380           E14:=XOR(E7,E9);
  390           E15:=XOR(E14,E11);
  400           E14:=XOR(E14,T[6]);
  410           E15:=XOR(E14,T[7]);
  420           E0:=E15;
  430           WRITELN('       DW      ',E15:1,E14:1,E13:1,E12:1,E11:1,E10:1,E9:1,E8:1,E7:1,E6:1,'00000',E0:1,'B          ;T=',
  440                   T[7]:1,T[6]:1,T[5]:1,T[4]:1,T[3]:1,T[2]:1,T[1]:1,T[0]:1)
  450   END; (* IMPRIME *)
  460
  470
  480
  490   BEGIN
  500           FIM:=FALSE;
  510           WRITELN('CRCTAB:');
  520           REPEAT
  530             IMPRIME;
  540             ADVANCE(0);
  550           UNTIL FIM
  560   END.

 0 ERROR(S) DETECTED
```

Fig.A.4. Geração da tabela para CRC16.

```
ALOR        CODE      ERR LINE          STATEMENT

                        1  ;       ****************************************************
                        2  ;       * TABELA PARA CALCULO DO CRC16 PELO METODO MULTIBIT *
                        3  ;       ****************************************************
                        4
0000                    5  CRCTAB:
0000  00 00             6                DW      0000000000000000B               ;T=00000000
0002  C1 C0             7                DW      1100000011000001B               ;T=00000001
0004  81 C1             8                DW      1100000110000001B               ;T=00000010
0006  40 01             9                DW      0100000101000000B               ;T=00000011
0008  01 C3            10                DW      1100000110000001B               ;T=00000100
000A  C0 03            11                DW      0000001111000000B               ;T=00000101
000C  80 02            12                DW      0000001010000000B               ;T=00000110
000E  41 C2            13                DW      1100001001000001B               ;T=00000111
0010  01 C6            14                DW      1100011000000001B               ;T=00001000
0012  C0 06            15                DW      0000011011000000B               ;T=00001001
0014  80 07            16                DW      0000011111000000B               ;T=00001010
0016  41 C7            17                DW      1100011101000001B               ;T=00001011
0018  00 05            18                DW      0000010100000000B               ;T=00001100
001A  C1 C5            19                DW      1100010111000001B               ;T=00001101
001C  81 C4            20                DW      1100010011000001B               ;T=00001110
001E  40 04            21                DW      0000010001000000B               ;T=00001111
0020  01 CC            22                DW      1100110000000001B               ;T=00010000
0022  C0 0C            23                DW      0000110011000000B               ;T=00010001
0024  80 0D            24                DW      0000110111000000B               ;T=00010010
0026  41 CD            25                DW      1100110101000001B               ;T=00010011
0028  00 0F            26                DW      0000111100000000B               ;T=00010100
002A  C1 CF            27                DW      1100111111000001B               ;T=00010101
002C  81 CE            28                DW      1100111101000001B               ;T=00010110
002E  40 0E            29                DW      0000111001000000B               ;T=00010111
0030  00 0A            30                DW      0000101000000000B               ;T=00011000
0032  C1 CA            31                DW      1100101011000001B               ;T=00011001
0034  81 CB            32                DW      1100101111000001B               ;T=00011010
0036  40 0B            33                DW      0000101101000000B               ;T=00011011
0038  01 C9            34                DW      1100100100000001B               ;T=00011100
003A  C0 09            35                DW      0000100111000000B               ;T=00011101
003C  80 08            36                DW      0000100010000000B               ;T=00011110
003E  41 C8            37                DW      1100100001000001B               ;T=00011111
0040  01 D8            38                DW      1101100000000001B               ;T=00100000
0042  C0 18            39                DW      0001100011000000B               ;T=00100001
0044  80 19            40                DW      0001100111000000B               ;T=00100010
0046  41 D9            41                DW      1101100101000001B               ;T=00100011
0048  00 1B            42                DW      0001101100000000B               ;T=00100100
004A  C1 DB            43                DW      1101101111000001B               ;T=00100101
004C  81 DA            44                DW      1101101011000001B               ;T=00100110
004E  40 1A            45                DW      0001101001000000B               ;T=00100111
0050  00 1E            46                DW      0001111000000000B               ;T=00101000
0052  C1 DE            47                DW      1101111011000001B               ;T=00101001
0054  81 DF            48                DW      1101111111000001B               ;T=00101010
0056  40 1F            49                DW      0001111101000000B               ;T=00101011
0058  01 D0            50                DW      1101110100000001B               ;T=00101100
005A  C0 10            51                DW      0001110111000000B               ;T=00101101
005C  80 11            52                DW      0001110011000000B               ;T=00101110
005E  41 D1            53                DW      1101110001000001B               ;T=00101111
0060  00 14            54                DW      0001010000000000B               ;T=00110000
0062  C1 D4            55                DW      1101010011000001B               ;T=00110001
```

Fig.A.5. Tabela para CRC16.

ADDR	CODE	ERR	LINE	STATEMENT		
0084	81 D5		56	DW	1101010110000001B	;T=00110010
0086	40 35		57	DW	0010101010000000B	;T=00110011
0088	01 D7		58	DW	1101011100000001B	;T=00110100
008A	C0 17		59	DW	0001011111000000B	;T=00110101
008C	80 16		60	DW	0001011010000000B	;T=00110110
008E	41 D6		61	DW	1101011001000001B	;T=00110111
0090	01 D2		62	DW	1101001000000001B	;T=00111000
0092	C0 12		63	DW	0001001011000000B	;T=00111001
0094	80 13		64	DW	0001001110000000B	;T=00111010
0096	41 D3		65	DW	1101001101000001B	;T=00111011
0098	00 11		66	DW	0001000100000000B	;T=00111100
009A	C1 D1		67	DW	1101000111000001B	;T=00111101
009C	81 D0		68	DW	1101000001000001B	;T=00111110
009E	40 10		69	DW	0001000001000000B	;T=00111111
00A0	01 F0		70	DW	1111000000000001B	;T=01000000
00A2	C0 30		71	DW	0011000011000000B	;T=01000001
00A4	80 31		72	DW	0011000110000000B	;T=01000010
00A6	41 F1		73	DW	1111000101000001B	;T=01000011
00A8	00 33		74	DW	0011001100000000B	;T=01000100
00AA	C1 F3		75	DW	1111001111000001B	;T=01000101
00AC	81 F2		76	DW	1111001010000001B	;T=01000110
00AE	40 32		77	DW	0011001001000000B	;T=01000111
00B0	00 36		78	DW	0011011000000000B	;T=01001000
00B2	C1 F6		79	DW	1111011011000001B	;T=01001001
00B4	81 F7		80	DW	1111011111000001B	;T=01001010
00B6	40 37		81	DW	0011011101000000B	;T=01001011
00B8	01 F5		82	DW	1111010100000001B	;T=01001100
00BA	C0 35		83	DW	0011010111000000B	;T=01001101
00BC	80 34		84	DW	0011010010000000B	;T=01001110
00BE	41 F4		85	DW	1111010010000001B	;T=01001111
00C0	00 3C		86	DW	0011110000000000B	;T=01010000
00C2	C1 FC		87	DW	1111110011000001B	;T=01010001
00C4	81 FD		88	DW	1111110111000001B	;T=01010010
00C6	40 3D		89	DW	0011110101000000B	;T=01010011
00C8	01 FF		90	DW	1111111100000001B	;T=01010100
00CA	C0 3F		91	DW	0011111111000000B	;T=01010101
00CC	80 3E		92	DW	0011111101000000B	;T=01010110
00CE	41 FE		93	DW	1111111001000001B	;T=01010111
00D0	01 FA		94	DW	1111101000000001B	;T=01011000
00D2	C0 3A		95	DW	0011101011000000B	;T=01011001
00D4	80 3B		96	DW	0011101111000000B	;T=01011010
00D6	41 FB		97	DW	1111101101000001B	;T=01011011
00D8	00 39		98	DW	0011100100000000B	;T=01011100
00DA	C1 F9		99	DW	1111100111000001B	;T=01011101
00DC	81 F8		100	DW	1111100001000001B	;T=01011110
00DE	40 38		101	DW	0011100001000000B	;T=01011111
00E0	00 28		102	DW	0010100000000000B	;T=01100000
00E2	C1 E8		103	DW	1110100011000001B	;T=01100001
00E4	81 E9		104	DW	1110100110000001B	;T=01100010
00E6	40 29		105	DW	0010100101000000B	;T=01100011
00E8	01 EB		106	DW	1110101100000001B	;T=01100100
00EA	C0 2B		107	DW	0010101111000000B	;T=01100101
00EC	80 2A		108	DW	0010101010000000B	;T=01100110
00EE	41 EA		109	DW	1110101001000001B	;T=01100111
00F0	01 EE		110	DW	1110111000000001B	;T=01101000

Fig.A.5. Tabela para CRC16 (cont.).

```
ADDR      CODE      ERR LINE      STATEMENT

00D2  C0 2E             111       DW   0010111011000000B     ;T=01101001
00D4  80 2F             112       DW   0010111110000000B     ;T=01101010
00D6  41 EF             113       DW   1110111101000001B     ;T=01101011
00D8  00 2D             114       DW   0010110100000000B     ;T=01101100
00DA  C1 ED             115       DW   1110110111000001B     ;T=01101101
00DC  81 EC             116       DW   1110110010000001B     ;T=01101110
00DE  40 2C             117       DW   0010110001000000B     ;T=01101111
00E0  01 E4             118       DW   1110010000000001B     ;T=01110000
00E2  C0 24             119       DW   0010010011000000B     ;T=01110001
00E4  80 25             120       DW   0010010110000000B     ;T=01110010
00E6  41 E5             121       DW   1110010101000001B     ;T=01110011
00E8  00 27             122       DW   0010011100000000B     ;T=01110100
00EA  C1 E7             123       DW   1110011111000001B     ;T=01110101
00EC  81 E6             124       DW   1110011010000001B     ;T=01110110
00EE  40 26             125       DW   0010011001000000B     ;T=01110111
00F0  00 22             126       DW   0010001000000000B     ;T=01111000
00F2  C1 E2             127       DW   1110001011000001B     ;T=01111001
00F4  81 E3             128       DW   1110001110000001B     ;T=01111010
00F6  40 23             129       DW   0010001101000000B     ;T=01111011
00F8  01 E1             130       DW   1110000100000001B     ;T=01111100
00FA  C0 21             131       DW   0010000111000000B     ;T=01111101
00FC  80 20             132       DW   0010000010000000B     ;T=01111110
00FE  41 E0             133       DW   1110000001000001B     ;T=01111111
0100  01 A0             134       DW   1010000000000001B     ;T=10000000
0102  C0 60             135       DW   0110000011000000B     ;T=10000001
0104  80 61             136       DW   0110000110000000B     ;T=10000010
0106  41 A1             137       DW   1010000101000001B     ;T=10000011
0108  00 63             138       DW   0110001100000000B     ;T=10000100
010A  C1 A3             139       DW   1010001111000001B     ;T=10000101
010C  81 A2             140       DW   1010001010000001B     ;T=10000110
010E  40 62             141       DW   0110001001000000B     ;T=10000111
0110  00 66             142       DW   0110011000000000B     ;T=10001000
0112  C1 A6             143       DW   1010011011000001B     ;T=10001001
0114  81 A7             144       DW   1010011110000001B     ;T=10001010
0116  40 67             145       DW   0110011101000000B     ;T=10001011
0118  01 A5             146       DW   1010010100000001B     ;T=10001100
011A  C0 65             147       DW   0110010111000000B     ;T=10001101
011C  80 64             148       DW   0110010010000000B     ;T=10001110
011E  41 A4             149       DW   1010010001000001B     ;T=10001111
0120  00 6C             150       DW   0110110000000000B     ;T=10010000
0122  C1 AC             151       DW   1010110011000001B     ;T=10010001
0124  81 AD             152       DW   1010110110000001B     ;T=10010010
0126  40 6D             153       DW   0110110101000000B     ;T=10010011
0128  01 AF             154       DW   1010111100000001B     ;T=10010100
012A  C0 6F             155       DW   0110111111000000B     ;T=10010101
012C  80 6E             156       DW   0110111010000000B     ;T=10010110
012E  41 AE             157       DW   1010111001000001B     ;T=10010111
0130  01 AA             158       DW   1010101000000001B     ;T=10011000
0132  C0 6A             159       DW   0110101011000000B     ;T=10011001
0134  80 6B             160       DW   0110101110000000B     ;T=10011010
0136  41 AB             161       DW   1010101101000001B     ;T=10011011
0138  00 69             162       DW   0110100100000000B     ;T=10011100
013A  C1 A9             163       DW   1010100111000001B     ;T=10011101
013C  81 A8             164       DW   1010100010000001B     ;T=10011110
013E  40 68             165       DW   0110100001000000B     ;T=10011111
```

Fig.A.5. Tabela para CRC16 (cont.)

```
ADDR        CODE          ERR LINE   STATEMENT

0140  80 7B                  166      DW    0111100000000000B          ;T=10100000
0142  C1 BB                  167      DW    1011100011000001B          ;T=10100001
0144  81 B9                  168      DW    1011100110000001B          ;T=10100010
0146  40 79                  169      DW    0111100101000000B          ;T=10100011
0148  01 B8                  170      DW    1011101100000001B          ;T=10100100
014A  C0 78                  171      DW    0111101111000000B          ;T=10100101
014C  80 7A                  172      DW    0111101010000000B          ;T=10100110
014E  41 BA                  173      DW    1011101001000001B          ;T=10100111
0150  01 BE                  174      DW    1011111000000001B          ;T=10101000
0152  C0 7E                  175      DW    0111111011000000B          ;T=10101001
0154  80 7F                  176      DW    0111111110000000B          ;T=10101010
0156  41 BF                  177      DW    1011111101000001B          ;T=10101011
0158  00 7D                  178      DW    0111110100000000B          ;T=10101100
015A  C1 BD                  179      DW    1011110111000001B          ;T=10101101
015C  81 BC                  180      DW    1011110010000001B          ;T=10101110
015E  40 7C                  181      DW    0111110001000000B          ;T=10101111
0160  01 B4                  182      DW    1011010000000001B          ;T=10110000
0162  C0 74                  183      DW    0111010011000000B          ;T=10110001
0164  80 75                  184      DW    0111010110000000B          ;T=10110010
0166  41 B5                  185      DW    1011010101000001B          ;T=10110011
0168  00 77                  186      DW    0111011100000000B          ;T=10110100
016A  C1 B7                  187      DW    1011011111000001B          ;T=10110101
016C  81 B6                  188      DW    1011011010000001B          ;T=10110110
016E  40 76                  189      DW    0111011001000000B          ;T=10110111
0170  00 72                  190      DW    0111001000000000B          ;T=10111000
0172  C1 B2                  191      DW    1011001011000001B          ;T=10111001
0174  81 B3                  192      DW    1011001110000001B          ;T=10111010
0176  40 73                  193      DW    0111001101000000B          ;T=10111011
0178  01 B1                  194      DW    1011000100000001B          ;T=10111100
017A  C0 71                  195      DW    0111000111000000B          ;T=10111101
017C  80 70                  196      DW    0111000010000000B          ;T=10111110
017E  41 B0                  197      DW    1011000001000001B          ;T=10111111
0180  00 50                  198      DW    0101000000000000B          ;T=11000000
0182  C1 90                  199      DW    1001000011000001B          ;T=11000001
0184  81 91                  200      DW    1001000110000001B          ;T=11000010
0186  40 51                  201      DW    0101000101000000B          ;T=11000011
0188  01 93                  202      DW    1001001100000001B          ;T=11000100
018A  C0 53                  203      DW    0101001111000000B          ;T=11000101
018C  80 52                  204      DW    0101001010000000B          ;T=11000110
018E  41 92                  205      DW    1001001001000001B          ;T=11000111
0190  01 96                  206      DW    1001011000000001B          ;T=11001000
0192  C0 56                  207      DW    0101011011000000B          ;T=11001001
0194  80 57                  208      DW    0101011110000000B          ;T=11001010
0196  41 97                  209      DW    1001011101000001B          ;T=11001011
0198  00 55                  210      DW    0101010100000000B          ;T=11001100
019A  C1 95                  211      DW    1001010111000001B          ;T=11001101
019C  81 94                  212      DW    1001010010000001B          ;T=11001110
019E  40 54                  213      DW    0101010001000000B          ;T=11001111
01A0  01 9C                  214      DW    1001110000000001B          ;T=11010000
01A2  C0 5C                  215      DW    0101110011000000B          ;T=11010001
01A4  80 5D                  216      DW    0101110110000000B          ;T=11010010
01A6  41 9D                  217      DW    1001110101000001B          ;T=11010011
01A8  00 5F                  218      DW    0101111100000000B          ;T=11010100
01AA  C1 9F                  219      DW    1001111111000001B          ;T=11010101
01AC  81 9E                  220      DW    1001111010000001B          ;T=11010110
```

Fig.A.5. Tabela para CRC16 (cont.).

```
ADDR        CODE        ERR LINE        STATEMENT

01AE  40 5E                 221          DW      0101111001000000B          ;T=11010111
01B0  00 5A                 222          DW      0101101000000000B          ;T=11011000
01B2  C1 9A                 223          DW      1001101011000001B          ;T=11011001
01B4  81 98                 224          DW      1001101110000001B          ;T=11011010
01B6  40 58                 225          DW      0101101101000000B          ;T=11011011
01B8  01 99                 226          DW      1001100100000001B          ;T=11011100
01BA  C0 59                 227          DW      0101100111000000B          ;T=11011101
01BC  80 58                 228          DW      0101100001000000B          ;T=11011110
01BE  41 98                 229          DW      1001100001000001B          ;T=11011111
01C0  01 98                 230          DW      1000100000000001B          ;T=11100000
01C2  C0 48                 231          DW      0100100011000000B          ;T=11100001
01C4  80 49                 232          DW      0100100110000000B          ;T=11100010
01C6  41 89                 233          DW      1000100101000001B          ;T=11100011
01C8  00 48                 234          DW      0100101100000000B          ;T=11100100
01CA  C1 88                 235          DW      1000101111000001B          ;T=11100101
01CC  81 8A                 236          DW      1000101010000001B          ;T=11100110
01CE  40 4A                 237          DW      0100101001000000B          ;T=11100111
01D0  00 4E                 238          DW      0100111000000000B          ;T=11101000
01D2  C1 8E                 239          DW      1000111011000001B          ;T=11101001
01D4  81 8F                 240          DW      1000111110000001B          ;T=11101010
01D6  40 4F                 241          DW      0100111010000000B          ;T=11101011
01D8  01 8D                 242          DW      1000110100000001B          ;T=11101100
01DA  C0 4D                 243          DW      0100110111000000B          ;T=11101101
01DC  80 4C                 244          DW      0100110001000000B          ;T=11101110
01DE  41 8C                 245          DW      1000110001000001B          ;T=11101111
01E0  00 44                 246          DW      0100010000000000B          ;T=11110000
01E2  C1 84                 247          DW      1000010011000001B          ;T=11110001
01E4  81 85                 248          DW      1000010110000001B          ;T=11110010
01E6  40 45                 249          DW      0100010101000000B          ;T=11110011
01E8  01 87                 250          DW      1000011100000001B          ;T=11110100
01EA  C0 47                 251          DW      0100011111000000B          ;T=11110101
01EC  80 46                 252          DW      0100011010000000B          ;T=11110110
01EE  41 86                 253          DW      1000011001000001B          ;T=11110111
01F0  01 82                 254          DW      1000001000000001B          ;T=11111000
01F2  C0 42                 255          DW      0100001011000000B          ;T=11111001
01F4  80 43                 256          DW      0100001111000000B          ;T=11111010
01F6  41 83                 257          DW      1000001101000001B          ;T=11111011
01F8  00 41                 258          DW      0100000100000000B          ;T=11111100
01FA  C1 81                 259          DW      1000000111000001B          ;T=11111101
01FC  81 80                 260          DW      1000000010000001B          ;T=11111110
01FE  40 40                 261          DW      0100000001000000B          ;T=11111111
0200                        262          END
```

Fig.A.5. Tabela para CRC16 (cont.).

UNICAMP
Centro de Computação

```
                        INTEL 8086 CROSS ASSEMBLER  V2.2    07-JUL-83  AT 14:14:13      PROGRAM > CRCM        PAGE   1

ADDR        CODE          ERR LINE            STATEMENT

                            1
E000                        2  CRCTAB  EQU     0E000H          ;ENDERECO DA TABELA PARA CALCULO DE CRC16
                            3
                            4
                            5
                            6  ;       **********************
                            7  ;       * ROTINA: CRC
                            8  ;       * PARAMETROS. ENTRADA: AL <-- BYTE CUJO CRC SE DESEJA ACUMULAR
                            9  ;       *                       DX <-- CRC ACUMULADO
                           10  ;       *              SAIDA: DX <-- NOVO CRC ACUMULADO
                           11  ;       * DESTROI: FLAGS
                           12  ;       * DESCRICAO: A ROTINA CALCULA O CRC16 PELO METODO
                           13  ;       *             MULTIBIT, COM O AUXILIO DA TABELA CRCTAB.
                           14  ;       **********************
                           15
                           16
0000                       17  CRC:
0000  50                    18          PUSH    AX              ;(15)
0001  56                    19          PUSH    SI              ;(15)
0002  32 C2                 20          XOR     AL,DL           ;(3)  ;CALCULA T
0004  32 E4                 21          XOR     AH,AH           ;(3)  ;ZERA AH (AX <-- T )
0006  8A D6                 22          MOV     DL,DH           ;(2)  ;PREPARA PARA
0008  8A F4                 23          MOV     DH,AH           ;(2)  ;FAZER OU-EXCLUSIVO COM CRCTAB[T]
000A  D1 E0                 24          SHL     AX,1            ;(2)  ;A TABELA CONTEM PALAVRAS (2 BYTES)
000C  8B F0                 25          MOV     SI,AX           ;(2)  ;SI <-- INDICE T
000E  33 94 00 E0           26          XOR     DX,CRCTAB[SI]   ;(22) ;DX <-- NOVO CRC ACUMULADO
0012  5E                    27          POP     SI              ;(12)
0013  58                    28          POP     AX              ;(12)
0014  C3                    29          RET                     ;(12)
0015                        30
```

Fig.A.6. Rotina para CRC16 pelo método multi-bit.