COLETA DE INFORMAÇÕES ORIENTADA PARA FORMULÁRIOS

USANDO O TERMINAL INTELIGENTE DO NCE/UFRJ

Raimundo Amora Ramos

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.)

Aprovada por:

Guilherme Chagas Rodrigues
(Presidente)

Ivan da Costa Marques

Nelson Maculan Filho

Ysmar Vianna e Silva Filho

RIO DE JANEIRO, RJ - BRASIL

JUNHO DE 1977

RAMOS, RAIMUNDO AMORA
Coleta de Informações Orientada para Formulários usando o Terminal Inteligente do NCE/UFRJ (Rio de Janeiro) 1977.
IX, 116 pag. 29,7 cm (COPPE - UFRJ, M. Sc., Engenharia de Sistemas e Computação, 1973)
Tese - Universidade Federal do Rio de Janeiro
1. Computadores I. COPPE/UFRJ

Raimundo Amora Ramos, nasceu no Rio de Janeiro em 7 de outubro de 1949. Formou-se em engenharia mecânica na Escola de Engenharia da Universidade Federal do Rio de Janeiro, em 1972.

Em Janeiro de 1973 iniciou seu curso de Pós-graduação em Engenharia de Sistemas e Computação matriculando-se na COPPE/UFRJ.

Aprovado por concurso público, exerce atualmente, o cargo de analista de sistemas senior do Banco Nacional de Desenvolvimento Econômico (BNDE).

É professor da cadeira de Introdução à Ciência dos Computadores na Faculdade de Engenharia da Associação Educacional Veiga de Almeida.

DEDICATÓRIA

A minha família, a quem devo, entre muitas outras coisas, a vontade de construir que me levou a começar este trabalho, e a persistência que me permitiu terminá-lo.

AGRADECIMENTOS

A Guilherme Chagas Rodrigues pela valiosa orientação no desenvolvimento desta tese.

A Francisco Dutra pela inestimável ajuda na utilização do Burroughs 6700 e do Terminal Inteligente do NCE.

A Pedro Pereira Brando, grande amigo e companheiro de estudos.

Ao Corpo Discente da COPPE pela formação que é , em verdade, o núcleo desta tese.

A Claudio Portela Peixoto pelo apoio e incentivo constante à elaboração e conclusão deste trabalho.

A Maria José de Paiva Coelho pela boa vontade ao longo do paciente trabalho de datilografia.

SINOPSE

Nesta tese apresento um método de se efetuar coleta de dados usando o terminal inteligente desenvolvido no Núcleo de Computação Eletrônica (NCE) da Universidade Federal do Rio de Janeiro (UFRJ).

Esta aplicação é projetada para prover entrada de dados distribuida. Os dados são obtidos com precisão em localizações remotas. O terminal inteligente pode ser instalado junto do usuário, na origem das informações. Sendo os dados familiares ao operador os erros são reduzidos. As informações são criticadas em dois níveis - nível de carater e nível de campo - logo após a sua digitação, permitindo a correção dos erros imediata.

Quantas vezes voce teve de esperar pelos dados de departamentos longínquos? E quantas vezes voce teve de devolver os documentos rejeitados e aguardar o seu retorno?

Uma ampla tela de vídeo com capacidade de até 1920 caracteres pode ser formatada projetando-se formularios que são preenchidos pelo operador numa forma conversacional. O vídeo permite ao operador ver convenientemente os campos preenchidos, além de mensagens de erros em Portugues e o estado atual de digitação. Quando uma mensagem de erro é colocada no vídeo, o teclado é desativado até que o operador corrija o erro. Para referência, a linha de estado mostra sempre atualizados itens tais como número do formulário, número de registros gravados, nome do programa e os atributos de cada campo.

O mais importante é o fato dos erros serem corrigidos logo após a sua digitação, eliminando assim, perda de tempo e o alto custo dos ciclos de correção. Rotinas de crítica para verificação de faixa de valores, validade de data, digitos de controle, pesquisa em tabelas, e cruzamento de campos, podem ser especificadas.

SUMMARY

In this thesis is proposed one package to perform data collection for use with the inteligent terminal developed at "Núcleo de Computação Eletrônica" which belongs to Federal University of Rio de Janeiro.

This application is designed for distributed batch entry. Data is accurately captured at remote locations. The inteligent terminal can be placed wherever information originates. Since the operators entering the data are familiar with it, errors are reduced. And because the data is validated on a character and field level as it is entered, errors are corrected immediately.

How often have you had to wait for data to come through the mail from remote offices? And how often have you sent back the rejects and had to wait for their return?

Large screen displaying up to 1920 characters can be formatted to resemble source documents with prompts to guide the operator in a conversational fashion. The video display lets the operator conveniently see the entire record in addition to error messages in Portuguese and the current batch status. When an error message is displayed the keyboard is disabled until the operator takes corretive action. For reference, the status line continously displays such items as the document number, record number, program name and field attributes.

Most important, errors are corrected as data is entered, eliminating costly and time-consuming error cycles. Routines to verify range checks, date validation, check digits, table comparisons, and interfield dependencies can all be specified.

ÍNDICE PÁGINA

PÁGINA

PÁGINA

I) INTRODUÇÃO

I.1- APRESENTAÇÃO DA OBRA

Nesta tese, requisito final para obtenção do título de M.Sc., na Coordenação dos Programas de Pós- Graduação de Engenharia (COPPE) da Universidade Federal do Rio de Janeiro (UFRJ), apresento um meio de se fazer coleta de dados com eficiência usando o terminal inteligente, projetado e construído, no Núcleo de Computação Eletrônica (NCE), órgão pertencente à U.F.R.J.

Através deste trabalho procuro contribuir no desenvolvimento do terminal inteligente, apresentando mais uma ferramenta de utilização deste notável equipamento brasileiro.

I.2- NATUREZA DO TEXTO

Nas páginas seguintes descrevo um conjunto de procedimentos e programas para efetuar coleta de dados, orientada para formulários, com consistência local.

Ao invés de se preencher um formulário, enviar para um centro de processamento de dados, e depois perfurar as informações em cartões ou em um outro dispositivo qualquer de entrada de dados, o formulário é projetado diretamente no vídeo do terminal inteligente.

A entrada dos dados é feita na hora permitindo-se visualização imediata do dado digitado e verificação simultânea.

Os dados são criticados em dois níveis - nível de carater e nível de campo - ocorrendo erro a informação não é aceita. É emitida uma mensagem ao operador, em Portugues, acusando o tipo de erro e pedida a redigitação do campo ou caracter.

O trabalho de transcrição atualmente executado nas perfuradoras de cartões tradicionais ou quaisquer outros equipamentos de entrada de dados é possível de ser realizado pelo terminal inteligente do NCE.

Podemos simular uma perfuração de cartões de 80 colunas com a geração dos dados em fita magnética cassete, com registros de tamanho fixo de 80 posições. Deste modo, os usuários que possuem sistemas utilizando entrada de dados por cartões, podem substituir imediatamente as perfuradoras pelo terminal, sem a incidência de custos de reprogramação.

São armazenados em fita cassete uma série de formulários que podem ser chamados através do teclado do terminal inteligente. Junto com esses padrões são guardadas informações sobre críticas a serem feitas em cada campo.

Ao final do dia, ou em qualquer momento, as informações coletadas e armazenadas em fita cassete são transmitidas para o computador central.

Essa transmissão de dados pode ser efetuada com três opções:

a) Ligação direta terminal - computador para curtas distâncias

b) Ligação por linhas telefônicas para médias e longas distâncias

c) Envio das próprias fitas cassete. Neste caso podemos até utilizar com proveito o serviço postal devido ao pequeno volume e peso das fitas.

I.3- IMPORTÂNCIA DA PESQUISA

Podemos afirmar que o custo de efetuar entrada de dados em qualquer centro de processamento situa-se entre 30% e 50% do custo total de processamento. Somente este aspecto justifica plenamente um exame completo dos procedimentos utilizados em preparação de dados.

Apesar das múltiplas alternativas disponíveis, estima-se que 90% de todos os dados usados pelos computadores são originários de cartões perfurados, os quais possuem os maiores custos efetivos, sobre várias condições.

As razões pelas quais, a perfuração em cartões ainda sobrevive, não foram cientificamente determinadas mas temos boas razões para suspeitar que:

1) Quando a carga de trabalho aumenta além de certos limites, é mais fácil ordenar a compra de outra perfuradora, e contratar outro operador, do que efetuar uma análise de custos entre várias alternativas.

2) As alternativas não são levadas em consideração, porque o custo incremental de alocar outra perfuradora e operador é insignificante, comparado com o custo incremental de uma nova UCP ou programador.

3) Muitos gerentes de centros de processamento de dados desconhecem as alternativas existentes, e seus custos e benefícios.

Nos últimos anos, notamos uma redução drástica do custo e tamanho dos componentes eletrônicos de computadores, acompanhada de um aumento de velocidade e desempenho. Assim, tornaram-se economicamente viáveis projetos de pequenos sistemas programáveis, por exemplo, o terminal inteligente do NCE.

Podemos afirmar que a redução de preço destes elementos é devida a:

1) Evolução da tecnologia de circuitos integrados LSI
2) Economia de escala com o crescimento das industrias do setor
3) Maior volume de produção
4) Redução do tamanho físico dos componentes
5) Incremento da competição entre fabricantes

A tecnologia em programação não está progredindo tão rapidamente como a tecnologia eletrônica, principalmen

te para os pequenos computadores. Existem muitas razões para a existência dessas condições, mas as seguintes são provavelmente as principais:

1) Porque pequenos equipamentos tem pequenos preços de venda, a unidade de lucro é menor do que para um grande equipamento. Por isso, os fabricantes não se sentem encorajados em investirem grandes somas de dinheiro no desenvolvimento de programas.

2) O repertório de instruções é normalmente pequeno. O esforço em programação requerido para produzir resultados é consideravelmente maior para um pequeno equipamento do que para um grande equipamento.

Um microprocessador é um circuito digital, com as funções de uma unidade central de processamento de computador, integrado em larga escala em uma única pastilha. O prefixo "micro" se refere principalmente ao tamanho, área inferior à $1cm^2$, e ao custo da pastilha, não a sua capacidade de computação. Existem hoje microprocessadores de 16 bits, mais rápidos do que um IBM 1130 e à venda por US$ 400,00.

O aparecimento de componentes como microprocessadores pela utilização de técnicas LSI (Integração em larga escala), representa mais do que um simples passo à frente na área de computação. Pela primeira vez, desde os primeiros computadores, houve uma inversão na conhecida lei de Grosh.

H. R. Grosh sugeriu, no início da década de 1950, que o desempenho de um computador aumenta com o quadrado de seu custo. Isto significa que um computador que custa duas vezes mais do que outro, tem o poder de processamento quatro vezes maior.

De fato, com o aparecimento dos microprocessadores, a lei de Grosh foi invertida. Um microcomputador atual pode ter um desempenho/custo superior a 100 vezes ao de um computador de grande porte.

As principais vantagens oferecidas pelos microprocessadores são:

a) Alto desempenho

b) Baixo custo

c) Alta confiabilidade

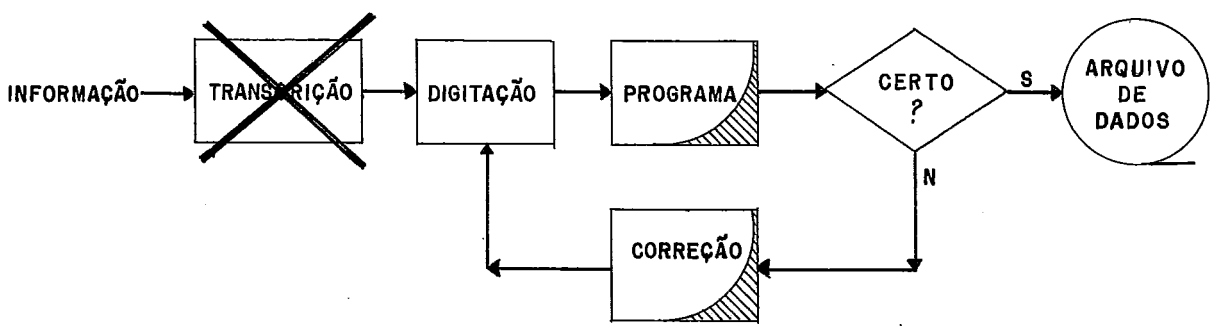
d) Baixo consumo de potência

Assim, temos uma abertura para um enorme mercado de aplicações, até então não viável e, portanto, inexplorado.

I.4- MOTIVOS QUE LEVARAM A PESQUISA

O processo clássico de entrada de dados é esquematizado abaixo. A idéia foi procurar dentro deste esquema clássico meios de simplificá-lo, reduzindo o número de ciclos de acerto, diminuindo o máximo possível o número de erros e facilitando a atividade de correção do movimento.

O ideal seria produzir uma fita de dados zero erro. No entanto, em muitos casos existem críticas particulares de cada sistema não previstas na definição original e críticas mais elaboradas ou sofisticadas difíceis de serem implantadas com os recursos disponíveis em um microprocessador.



Foram obtidos os seguintes melhoramentos:

1) Eliminação na transcrição para formulários.

Eliminando a sua utilização evitamos os erros provenientes da transcrição para formulários. É um erro muito

comum. Normalmente chegam na seção de perfuração documentos preenchidos incorretamente.

Existem alguns problemas quando usamos formulários:

a) Quando não existe formulário, dependemos de um grande número de tarefas e pessoas. Vejamos o algoritmo abaixo, para obtenção de um formulário novo.

F1. desenhe um croquis

F2. envie ao departamento de Desenho

F3. aguarde

F4. se o desenho está pronto siga adiante. Senão volte a F3

F5. se o desenho está correto siga adiante. Senão indique o erro e volte a F2

F6. se a quantidade necessária é grande vá para F8

F7. tire xerox e vá para F11

F8. envie desenho para gráfica e autorize a tiragem de n cópias

F9. aguarde

F10. se as cópias estão prontas, siga adiante. Senão volte a F9

F11. o formulário está pronto

b) Quando o formûlário já existe:

1) Se o formulário passa a ser desnecessário ou há necessidade de alguma modificação, as cópias já prontas devem ser jogadas no lixo, acarretando sérios prejuízos.

2) Quando há uma alteração no formulário, é necessário a confecção de um novo desenho e tiragem de novas cópias. Temos uma grande perda de tempo e um custo alto para obtenção do novo formulário.

3) Precisamos de ter um estoque. Pessoas são alocadas para o controle do estoque, gerando novos custos adicionais.

Há uma vantagem com o uso dos formulários que

é a geração de um histórico dos dados fornecidos para processamento. No entanto, é uma vantagem nem sempre válida, já que os formulários muitas vezes são jogados fora após a conferência com os relatórios de atualização.

2) Eliminação da perfuração de cartões.

Eliminaremos assim os erros oriundos da perfuração já que existirão críticas severas no terminal inteligente. Temos um equipamento capaz de substituir a perfuradora de cartões com vantagens e recursos adicionais.

3) Diminuição do custo de desenvolvimento do programa de crítica

As críticas fundamentais são feitas pelo próprio terminal inteligente. Temos crítica de validade de data, dígito verificador, pesquisa em tabelas, faixa de valores, cruzamento de campos e outras. Portanto, a maioria das validações são efetuadas pelo terminal inteligente.

Sobram apenas as críticas mais refinadas, se existirem, para serem executadas pelo programa de crítica do computador central. Em conseqüência disto, este programa terá um custo de desenvolvimento bem pequeno. Também temos economia no tempo de processamento, acarretando menor gasto de UCP. Será um programa de fácil e rápida execução.

4) Diminuição dos ciclos de correção

A conseqüência desta diminuição é um evidente ganho de tempo. O movimento para atualização ficará acertado rapidamente. Apesar de existirem críticas fortes no terminal inteligente, essas não podem ser completas. Os ciclos de correção não podem ser eliminados, no entanto, esses ciclos irão diminuir a um nível mínimo.

5) Redução de pessoal

Devido à diminuição da carga de trabalho na atividade manual de correção do movimento, temos redução do pessoal responsável pela correção do movimento. Esta atividade consiste na consulta aos relatórios produzidos pelo programa de crítica do computador central e preenchimento de novos formulários para correção. É um processo de verificação e cor

reção tedioso e quase sempre lento.

6) Outra redução de custo

O tradicional cartão de 80 colunas além de ser caro, não pode ser reaproveitado. As fitas magnéticas cassete são reusáveis.

Neste ponto devo salientar que a escolha das unidades de fita cassete, nesta tese, foi efetuada devido à disponibilidade, de utilização experimental, de um terminal inteligente equipado com essas unidades, no laboratório eletrônico do NCE. Aproveito a oportunidade para lembrar que esta tese não é apenas uma proposição escrita. Todos os procedimentos e programas aqui descritos foram testados arduamente , no laboratório do NCE, e estão prontos e disponíveis para uso.

No entanto, existe uma tendência de substituição das unidades de fita cassete por uma unidade de disco flexível (diskete). Provavelmente os novos terminais inteligentes a serem construídos, serão configurados com o diskete. Esse disco flexível possui maior capacidade de armazenamento, maior velocidade de transmissão, permite acesso direto e seqüencial, e, o seu custo ainda é menor do que as unidades de fita cassete.

O sistema de coleta de dados foi projetado de modo que, as alterações necessárias, para se efetuar a troca das unidades de fita cassete pela unidade de disco flexível , são de pequena monta e de fácil implantação.

Logo, existirão disponíveis duas versões do sistema. A original, utilizando fita magnética cassete, e a versão futura, utilizando o disco flexível.

7) Equipamento brasileiro

Três aspectos enriquecem a pesquisa, e contribuem para o desenvolvimento econômico do Brasil:

a) Utilização de um equipamento desenvolvido com tecnologia local.

b) Máquina capaz de substituir equipamentos importados com a conseqüente economia de divi

sas para o país.

c) Incentivo ao trabalho de brasileiros que procuram incrementar o conhecimento tecnológico nacional na matéria.

I.5- CARACTERÍSTICAS DO PROJETO

1) A coleta de dados deve ser descentralizada

Apesar de ser perfeitamente operacional como equipamento de entrada de dados centralizado, é mais adequado utilizar o terminal inteligente como equipamento de entrada de dados descentralizado. Outros termos usados em substituição a descentralizado são distribuído ou remoto.

Para se obter, uma utilização mais eficiente , na coleta de dados, o terminal inteligente deve ser colocado junto do usuário, na fonte das informações. A coleta de dados, conforme projetada é iterativa. Existe um diálogo homem-máquina. A ocorrência de um erro, por exemplo, é comunicada ao operador com um sinal audível e uma mensagem de erro, em Portugues, colocada no vídeo do terminal. Neste caso, o operador certifica-se do erro, toma as providências necessárias para corrigí-lo, submete de novo o dado à máquina, e verifica a aceitação ou não do dado.

Instalando o terminal inteligente, junto do usuário fornecedor de todos os dados a serem digitados, temos uma coleta de dados descentralizada ou remota, onde o operador é o próprio dono da informação. Se ocorrer um erro, ele poderá ser corrigido na hora, por uma pessoa que conhece a informação, portanto capaz de decidir por uma correção.

Estima-se que o custo da correção de um erro na fonte da informação é, pelo menos, 10 vezes menor do que o custo da correção de um erro detetado pelo computador central.

Poucas instalações, acharão prático eliminar completamente a digitação centralizada, mas muitas instalações, vão achar a preparação de dados remota mais econômica e

adequada aos seus requisitos de entrada de dados.

Significantes reduções de tempo podem ser obtidas, com o uso de preparação remota de dados. A experiência de muitas instalações prova que, quando o pessoal que cria os dados, também é responsável pela preparação dos dados para o computador, a incidência de erros é muito menor, e portanto , ocorre um menor número de ciclos de correção.

Uma razão da entrada remota de dados, não ter ainda se consolidado, é o fato de, instalações de equipamentos infringirem os direitos de outros departamentos na organização e, da dependência da aceitação pelos departamentos onde o equipamento será localizado. Em muitos casos, gerentes dos departamentos afetados sentem uma interferência nas suas áreas de responsabilidade, e são capazes de bloquear a instalação de equipamentos de entrada remota de dados.

Esta atitude de gerentes, em departamentos onde as operações são afetadas, tende a cair, devido ao interesse em obter respostas corretas e confiáveis, com muito maior rapidez. Em alguns casos, as atitudes mudam, se as ordens são oriundas da alta administração. O presidente decide pela entrada remota de dados, e diz a todos para se adaptarem,ou procurar um novo trabalho. Desta maneira, os gerentes de departamentos afetados se tornam interessados, e mais favoráveis à entrada remota de dados, porque os resultados obtidos são de melhor qualidade, e mais rápido, do que possível utilizando a preparação de dados centralizada.

2) Quando a coleta de dados é centralizada a consistência local é desfavorável.

Num sistema concentrado de entrada de dados , com a utilização de um processador compartilhado, todos os terminais devem ficar próximo da unidade central de processamento, o que obriga a uma coleta de dados centralizada. Deste modo, a transmissão dos dados para o computador não é efetuada pelo dono da informação e sim por uma digitadora, em um centro de processamento.

Se existirem críticas fortes, na ocorrência de

um erro, o teclado é travado e o sistema para. Como o responsável pela informação não está por perto, temos duas soluções para resolver o problema:

a) devolver o documento ao usuário.
Neste caso, teremos uma grande perda de tempo. Um serviço fica pendente na máquina. O trabalho de digitação é interrompido, até que o usuário corrija o documento e o devolva. Esta primeira solução é obviamente desvantajosa e de alto custo. Também não conseguimos evitar o forte impulso que as digitadoras possuem de tentar corrigir um erro. O usuário é capaz de corrigir um dado, a digitadora, não. O usuário é o dono da informação.

b) aceitar o erro e gravar a informação.
O documento deve ser gravado mesmo com erro. O objetivo, neste caso, é transcrever exatamente o que vem preenchido no formulário. Se o formulário contiver erros, a responsabilidade é de quem o preencheu, e o documento é gravado com erro. Posteriormente o programa de crítica do computador central acusa um registro errado. O usuário analiza os relatórios produzidos por este programa e encontra indicação do documento com erro, qual o campo errado de qual lote e normalmente uma mensagem dando a causa do erro. Obviamente, devemos ter um programa de crítica completo e caro. Ora, se vamos aceitar o erro, para que as rotinas de crítica severas na entrada de dados?

Concluimos que, num sistema centralizado de entrada de dados com processador compartilhado não devemos ter críticas fortes. A filosofia deve ser diferente de uma coleta de dados remota onde o dado pode ser corrigido na própria fonte. Usando o terminal inteligente como entrada de dados des-

centralizada, nenhum documento é devolvido e nenhum dado é gravado com erro. As críticas severas são plenamente aproveitadas.

3) Possibilidade de ser ligado a um outro computador.

São possíveis ligações assincronas quanto ligações sincronas.

4) Possibilidade de ser usado em outras aplicações.

Além de coletar dados, podemos usar o terminal para transmissão de dados, entrada remota de programas, etc . As aplicações possíveis são tantas, praticamente um saco sem fundo.

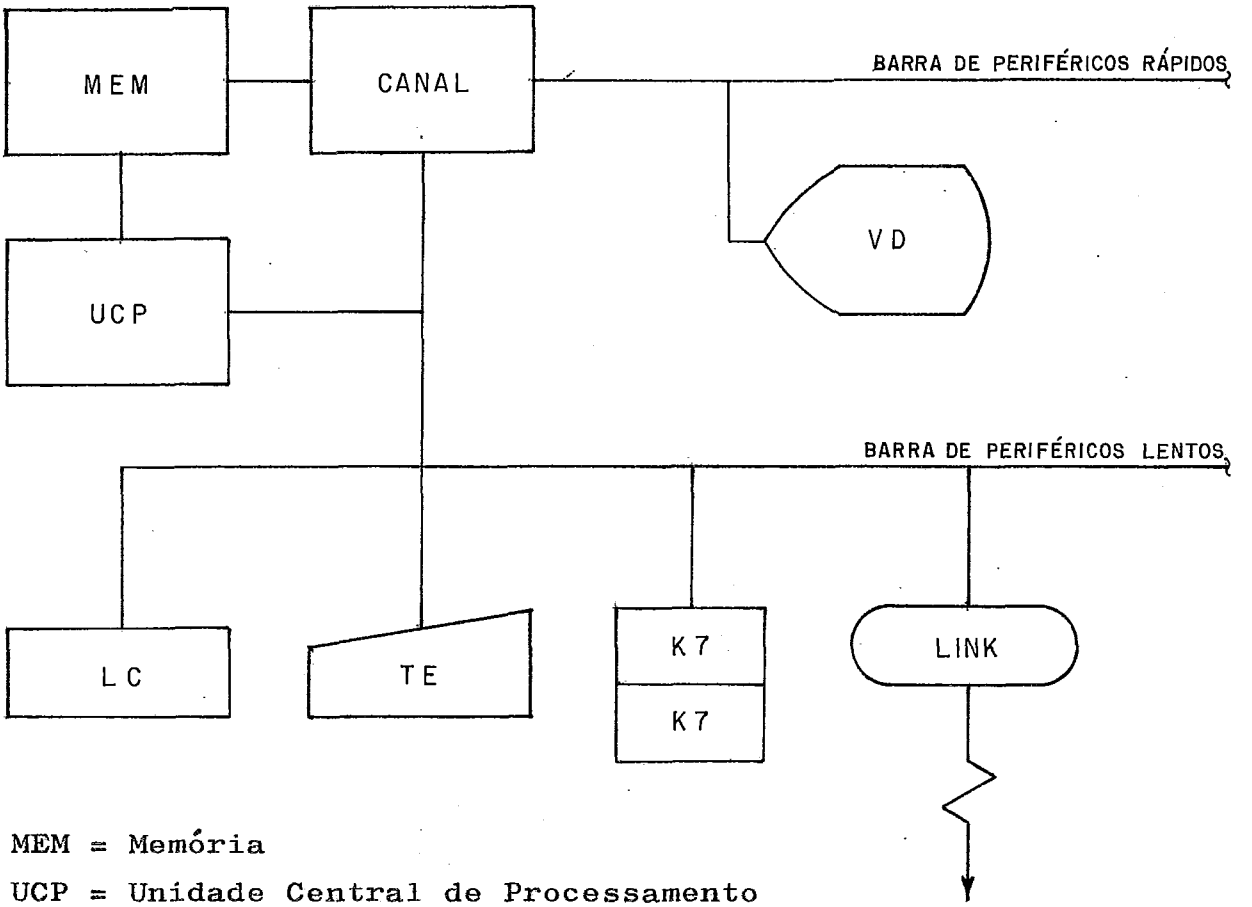
5) Processamento de pequenos volumes.

Quando o processamento é de grande volume, é justificável a utilização de um concentrador de teclados com processador compartilhado. São mais econômicos neste caso, se usando o número máximo de terminais.

6) Orientado para pessoas não especializadas em digitação.

Olhar para o vídeo não é próprio de uma digitadora. Existindo o formulário, não há necessidade do vídeo. É necessário apenas um visor com algumas informações, tais como número do campo a ser digitado, número do documento, número do lote e algumas mensagens fixas. Uma boa digitadora, só olha para o formulário. No nosso caso o formulário está no próprio vídeo.

II) CONFIGURAÇÃO



MEM = Memória
UCP = Unidade Central de Processamento
K7 = fitas cassete (duas unidades)
VD = vídeo
TE = teclado
LC = luzes e chaves
LINK= interface com outro computador

São previstos ainda periféricos opcionais tais como:

1) Leitora de cartões
2) Impressora
3) Diskette

II.1- DESCRIÇÃO DA UNIDADE CENTRAL DE PROCESSAMENTO

1) <u>O Intel 8008 é o processador central especificado para o terminal inteligente.</u>

Um microcomputador é formado quando o 8008 é ligado a qualquer tipo de memória de 4K até 16K palavras de 8 bits. K é uma constante de valor igual a 1024.

A organização interna é baseada numa barra de dados de 8 bits. Todas as comunicações entre o processador e os componentes externos passam por esta barra na forma de bytes de 8 bits de endereços, instruções ou dados. O processador controla o uso da barra de dados determinando quando enviar ou receber dados.

2) A Unidade central de processamento contém sete registradores.

Temos o acumulador, denominado registrador A, e outros seis registradores adicionais denominados B,C,D,E,H, L. Todas as operações aritméticas usam o acumulador como um dos operandos, e, toda entrada e saída programada é feita através desse registrador. Todos os registradores são independentes e podem ser usados como memória temporária. Dois registradores temporários são usados para guardar o acumulador e o operando nas operações com a unidade aritmética e lógica. Quatro bits de controle: flip-flop de carry (vai-um), flip- flop zero, flip-flop de sinal e flip-flop de paridade, são ligados conforme o resultado de cada operação aritmética ou lógica. Esses bits permitem desvios condicionais através das instruções "CALL", "JUMP" ou "RETURN". Além disso, o flip-flop carry(vai-um) proporciona uma precisão múltipla em aritmética binária.

3) O endereçamento com 14 bits permite acesso aos 16K bytes da memória.

Para as instruções que requerem operações com a memória externa os registradores H e L, concatenados, fornecem endereçamento indireto: o registrador L contém os oito bits de mais baixa ordem, e o registrador H contém os seis bits de mais alta ordem do endereço. Como são necessários apenas 14 bits para termos acesso aos 16K bytes da memória, dois bits de mais alta ordem do registrador H não usados.

Uma memória em pilha, contendo um registrador

para processamento de instruções, com 14 bits, e sete outros registradores com 14 bits, são utilizados internamente, para chamadas de subrotinas até sete níveis.

4) O repertório de instruções do INTEL 8008 consiste de 48 instruções.

A maioria das instruções são codificadas em um byte (8 bits); instruções de carregamento imediato usam 2 bytes; instruções de desvio e chamada de rotinas utilizam 3 bytes porque são necessários 2 bytes para o endereço.

Uma instrução no INTEL 8008 pode ter cinco tipos de ciclo. Três ciclos são os de busca da instrução, e são comuns a todas elas. Dois ciclos são de execução, logo, uma instrução tem no mínimo cinco ciclos. O tempo de um ciclo é de aproximadamente 2,8 microsegundos. Assim, o tempo mínimo de uma instrução é de 14 microsegundos. A instrução mais lenta requer onze ciclos, dando um tempo de 30,8 microsegundos. Como tempo de uma instrução, em média, pode-se considerar 20 microsegundos.

II.2- DESCRIÇÃO DA MEMÓRIA

A memória do terminal inteligente tem a largura de oito bits compatível com a unidade central de processamento utilizada. É composta de módulos de 4K bytes podendo ir até um máximo de 16K bytes (4 módulos). O tempo de acesso á um dado é da ordem de 800 nanosegundos, sendo entretanto um ciclo mínimo estabelecido em 1,6 microsegundos.

No final da memória, após os 16K temos 256 posições possíveis de serem endereçadas, compostas de uma memória de leitura excluiva ("ROM"), onde fica residente o programa de carga inicial do sistema ("BOOTSTRAP"). Este programa é acessado através do botão CARGA, situado no painel do terminal. Ao apertarmos esse botão, é gerada uma interrupção e o processador executa uma instrução de desvio para o endereço

$3FOO_{16}$ (final da memória).

No terminal inteligente só temos uma linha de interrupção. Os periféricos não geram interrupção. Existem dois botões externos, situados no painel do terminal, que geram interrupção. O botão CARGA e o botão ERRO. O botão ERRO faz o processador executar uma instrução de chamada a uma subrotina armazenada no endereço de memória 0038_{16}. Normalmente é usado em caso de erro, para reinicialização ou cancelamento de um programa.

II.3- DESCRIÇÃO DO CANAL DE ACESSO DIRETO À MEMÓRIA

Existem dois sistemas de entrada e saída para o terminal inteligente. Um para periféricos de baixa velocidade de transmissão e, outro para periféricos de alta velocidade. Define-se aqui baixa velocidade de transmissão até aproximadamente 3.000 bytes por segundo. A velocidade de transferência máxima para o terminal inteligente é de 600.000 bytes por segundo. Com esta velocidade de transferência de dados podemos conectar ao terminal inteligente, praticamente , todos os dispositivos de alta velocidade usados em computadores de grande porte.

Os diversos periféricos são conectados através de interfaces convenientes à barra de dados de entrada e saída do terminal. Um comando seleciona uma determinada interface fazendo com que todos os comandos subsequentes sejam dirigidos para o respectivo periférico.

Os periféricos lentos, ou de baixa velocidade de transmissão, utilizam a entrada e saída programada. O terminal inteligente não faz entrada e saída através de interrupção.

Os periféricos rápidos, ou de alta velocidade de transmissão, requerem um fluxo de entrada e saída maior que o processador pode controlar. Assim, foi desenvolvido no NCE, um canal de acesso direto à memória. Este dispositivo é capaz de transferir dados entre um periférico e

a memória a uma velocidade apenas limitada pelo tempo necessário à escrita ou leitura da memória. A velocidade do canal está portanto diretamente relacionada com o tipo de memória utilizada.

O processador carrega palavras de controle no canal, que se comporta como um periférico de baixa velocidade. A operação de entrada e saída é iniciada, e segue até o seu final sem intervenção do processador, que espera até que ela seja completada. Desta forma, o processador e o canal não disputam a posse da memória ao mesmo tempo.

Como o canal foi projetado para transferências de dados à alta velocidade, a sua melhor utilização faz-se com dados blocados, isto é, uma sequência de bytes que serão lidos ou escritos a partir de um endereço na memória.

São considerados periféricos lentos e, portanto ligados à unidade central de processamento: o teclado, as chaves, as luzes, a interface de comunicação, as unidades de fita cassete, a leitora de cartões e a impressora.

São considerados periféricos rápidos e, portanto ligados ao canal: a unidade de vídeo e o diskete.

II.4- DESCRIÇÃO DOS PERIFÉRICOS

1) Unidade de vídeo

Unidade de visualização composta de um tubo raios catódicos. A tela suporta 12 linhas de 80 colunas, dando um total de 960 caracteres. Há previsão de expansão da tela para 24 linhas, onde será dobrada a capacidade da tela em caracteres. Estando ligado ao canal, a transferência de dados se faz em blocos de 80 caracteres. Deste modo só podemos ler e escrever uma linha completa.

O cursor é indicado pelo complemento luminoso da letra sobre a qual está colocado. A linha de vídeo lida ou escrita é endereçada através do endereço vertical do cursor.

São possíveis as seguintes funções:

a) Apagar a tela
b) Apagar uma linha
c) Posicionar o cursor
d) Tirar o cursor da tela
e) Escrever uma linha
f) Ler uma linha
g) Rolar a tela para cima
h) Rolar a tela para baixo.

2) Teclado

Teclado da MICROSWITCH, possuindo todos os caracteres gráficos e de controle em ASCII. Um comando de leitura transfere um carater da interface para o acumulador. O carater na interface é atualizado a cada tecla que é pressionada. Assim, caso não seja feita uma leitura e diversas teclas tenham sido acionadas, apenas o último carater permanece.

3) Unidade de fita cassete

Unidade de fita cassete da REDACTRON modelo 110, composta de dois transportadores. Cada unidade tem capacidade máxima de 100K bytes caso seja gravado um bloco único. Para casos práticos, pode-se admitir uma capacidade de 60K a 80K por fita.

São possíveis as seguintes funções:

a) Escrever marca de separação de arquivo
b) Escrever um bloco
c) Ler um bloco
d) Voltar um arquivo
e) Avançar um arquivo
f) Voltar um bloco
g) Avançar um bloco
h) Reenrolar a fita ao seu início físico
i) Reescrever um bloco sobre outro já escrito

4) Luzes e chaves

Periférico constituido de oito chaves e oito lâmpadas, numerados de 0 a 7, correspondendo aos bits do acu-

mulador. Um comando de leitura transfere para o acumulador o conteúdo das chaves, enquanto um comando de escrita transfere o conteúdo do acumulador para as lâmpadas. Existe um botão de VALIDADE, situado no painel do terminal inteligente, cuja função é indicar ao processador que as chaves já foram posicionadas, e podem ser lidas. A este mesmo periférico, está associado um dispositivo, que emite um sinal audível, de aproximadamente um segundo, e outro de menor tempo, acionados por programa.

5) Interface de comunicação

A interface de comunicação sincrona, usada para ligação, por exemplo, com computadores da linha IBM, encontra-se em fase de desenvolvimento. Descrevemos abaixo somente as características da interface de comunicação assincrona.

Podemos efetuar uma ligação com outro computados ou terminal. É possível escolher a transmissão através de laço de corrente, para curtas distâncias, ou, através de MODEM, para médias e longas distâncias, de acordo com normas internacionais. Velocidade entre 110 e 9600 bps. Caracteres de 5 a 8 bytes. Paridade par ou ímpar ou sem paridade. Verificação automática de erros de paridade, enquadramento e sobreposição.

III) LINGUAGEM DE CRIAÇÃO DE FORMULÁRIOS FONTE

1) Uma linguagem de fácil uso foi criada para permitir a formatação de um formulário na tela de vídeo do terminal inteligente, definição dos atributos de cada campo e codificação das rotinas de crítica.

Esta linguagem é composta de 6 tipos de declarações:

a) FO - declaração de formulário
b) CA - declaração de campo
c) CR - declaração de crítica
d) CO - declaração de comentário
e) IN - declaração de campo inserido
f) FIM - declaração de término de programa

2) Uma declaração quando codificada tem as seguintes características:

- uma lista de cláusulas, separadas por vírgula.
- a primeira cláusula identifica a declaração.
- é terminada pelo carater ponto-e-vírgula.
- após o ponto-e-vírgula podemos colocar comentários ou observações sobre o programa.

3) Descrição da linguagem:

/ delimitadores de cláusula /
:: = equivalência
| alternativa
[ ] a cláusula ou parte da cláusula delimitada por este sinal pode ser dispensada

3.1. Declaração de formulário

```
FO, / identificador /, /'título'/ ;
```

/ identificador /::= 6 posições para identificação do formulário

/'título'/::= cadeia de caracteres com até 60 posições dando o título do formulário

Essa declaração é obrigatória e única para cada programa de crítica. Deve ser codificada como a primeira declaração do programa.

3.2. Declaração de campo

```
CA, /'nome'/, / característica /, / tamanho /,
/ edição /, /'caráter' /, / salto /, / coluna /;
```

/'nome'/:: = cadeia de caracteres que identifica o campo

/ característica /::= N | A | L | E | P | Q

N = numérico
A = alfabético e espaço
L = alfanumérico
E = numérico e especiais
P = alfabético, espaço e especiais
Q = alfanumérico e especiais

/ tamanho /::=dois algarismos fornecendo o tamanho máximo do campo

/ edição / ::= F | E | D

F = formato fixo com tamanho especificado na cláusula tamanho. Todos os caracteres do campo devem ser digitados.

E = a informação digitada deve ser alinhada à esquerda e o carater de preenchimento das demais posições é dado pela cláusula caráter.

D = a informação digitada deve ser alinhada à direita, e o caráter de preenchimento das demais po-

sições é dado pela cláusula carater.

/'carater'/ :: = carater de preenchimento

/ salto / :: = S | N

S - permite, através de tecla especial, o salto do campo. Todas as posições do campo são preenchidos com o carater dado na cláusula carater.

N - não permite o salto do campo. A digitação é obrigatória.

/ coluna / :: = número para indicar a coluna onde se inicia o campo, em uma linha do vídeo.

Se igual a zero, o campo se inicia na primeira coluna disponível da linha.

Se maior que zero, o campo pode se iniciar na mesma linha ou na linha seguinte, dependendo se a coluna indicada estiver disponível ou não. Se a coluna indicada estiver disponível, o campo se inicia nesta coluna. Se a coluna indicada estiver ocupada, o campo se inicia na linha seguinte, na coluna fornecida. As posições do início da nova linha, até a coluna onde se inicia o campo, são preenchidas com brancos. Também são preenchidos com brancos as colunas não utilizadas entre um campo e outro.

Um campo não pode ultrapassar uma linha. Ele deve se iniciar e terminar na mesma linha.

3.3- Declaração de crítica

CR, / lista / ;

/ lista / :: = / crítica /, /crítica /, / crítica /, ...

/ crítica / :: = / nome / , / parâmetros/

/ nome / :: = nome pelo qual a crítica é reconhecida até 5 posições

/ parâmetros / :: = (número inteiro$_1$, número inteiro$_2$,... número inteiro$_n$)

| ('cadeia$_1$', 'cadeia$_2$', ..., 'cadeia$_n$')

Para que uma lista seja satisfeita, é necessário que o conteúdo do campo atenda a todas as críticas da lista.

Uma declaração de crítica deve, obrigatoriamente, seguir uma declaração de campo ou outra declaração de crítica.

Uma declaração de campo, pode não ter nenhuma declaração de crítica associada.

3.4- Declaração de campo inserido

IN, / 'cadeia' / ;

/ 'cadeia' / :: = cadeia de caracteres entre apóstrofes.

Este campo não é digitado. Esta declaração deverá ser usada quando se desejar uma cadeia qualquer de caracteres inserida no registro de saída. A inserção pode ocorrer em qualquer posição do registro. A digitadora não toma conhecimento deste campo.

A declaração de campo inserido não pode ser colocada entre uma declaração de campo e uma declaração de crítica.

3.5- Declaração de comentário

CO, /'cadeia'/ , / coluna / ;

/ coluna / :: = número para indicar a coluna onde se inicia o comentário, em uma linha do vídeo.

Se igual a zero o comentário se inicia na primeira coluna disponível da linha.

Se maior que zero, o comentário pode se inici

ar na mesma linha ou na linha seguinte, dependendo se a coluna indicada estiver disponível ou não. Se a coluna indicada estiver disponível, o comentário se inicia nesta coluna. Se a coluna indicada estiver ocupada, o comentário se inicia na linha seguinte, na coluna fornecida. As posições do início da nova linha, até a coluna onde se inicia o comentário, são preenchidas com brancos. Também são preenchidas com brancos as colunas não utilizadas entre dois comentários ou entre um comentário e um campo.

É necessário uma declaração de comentário para cada linha. Se o comentário estourar uma linha ele é truncado no tamanho da linha.

/ 'cadeia' / :: = cadeia de caracteres entre apóstrofes.

3.6- Declaração de fim de programa

Indica a finalização do programa.

FIM, / nome / ;

/ nome / :: = nome do formulário seguinte. Usado na sequenciação de formulários.

Esta declaração é obrigatória e única para cada programa de crítica. Deve ser codificada como a última declaração do programa.

IV) ROTINAS DE CRÍTICA

IV.1- NÍVEIS E TIPOS DE ROTINAS

A vantagem do terminal inteligente sobre equipamentos de entrada de dados similares é a possibilidade de efetuar validações mais consistentes, através de rotinas de crítica padrão.

Essas críticas devem ser especificadas pelo analista responsável pela transcrição da aplicação, quando definir os campos a serem digitados.

Desta forma, temos meios de verificar validade de data, digito verificador, faixa de valores, comparar valores de tabelas e efetuar cruzamento de campos.

São disponíveis quatro contadores, com tamanho de um byte, permitindo-se guardar números binários no intervalo 0 a 255 e, dois acumuladores com tamanho de 15 dígitos decimais sem sinal. Os contadores podem ser usados como indicadores para cruzamento de campos. É possível, também, efetuar fechamentos verticais, adicionando parcelas em um acumulador, e após, comparando-o com o valor total digitado.

Foram definidos dois níveis de crítica:

a) Nível de carater

A crítica a nível de carater é obrigatória, e é especificada quando da declaração da característica de um campo. Cada carater digitado, em um campo, sofre imediatamente verificação de característica. Se, por exemplo, em um campo numérico for teclada uma letra, ou um carater qualquer diferente de número, este é rejeitado. Emite-se um sinal audível para alertar o operador e, coloca-se a mensagem de erro "CARATER INVÁLIDO" em uma linha reservada. O processamento é interrompido e nenhum outro carater digitado é aceito. Certificando-se do erro, o operador aperta a tecla ERRO. Após, é apagada a mensagem de erro, o cursor volta uma posição no campo, e aguarda-se a digitação de um novo carater para a mesma posição.

b) nível de campo

Somente uma crítica a nível de campo é obrigatória. É a crítica de tamanho de campo. O tamanho do campo é especificado numa declaração de campo. Uma declaração de campo, pode não ter nenhuma declaração de crítica associada. Neste caso, o campo só sofrerá crítica de característica, a nível de carater, e crítica de tamanho de campo.

A crítica de tamanho de campo pode gerar duas mensagens de erro distintas:

1) "TAMANHO INSUFICIENTE"

Ocorre quando, em um campo de edição fixa, não são digitadas todas as posições do campo. Este erro nunca acontece, se o campo é editado à direita ou à esquerda.

2) "ESTOURO DE TAMANHO"

Ocorre quando, em um campo com qualquer tipo de edição e de tamanho n, é digitado o carater n + 1.

Todas as outras rotinas de crítica a nível de campo, não são obrigatórias e, aparecem em uma declaração de crítica. Essas rotinas de crítica não obrigatórias ou opcionais, são de dois tipos:

a) críticas de ação

São rotinas que não produzem retorno.

Essas rotinas não causam a emissão de mensagens de erro. Somente uma ação é executada. Geralmente são rotinas que agem nos contadores ou nos acumuladores.

b) críticas de verificação

As críticas de verificação podem interromper a digitação e causar a emissão de uma mensagem de erro. Elas produzem retorno, informando a validade ou não de um campo, ou, se é verdadeiro ou falso, o resultado de um teste.

IV.2- DESCRIÇÃO DAS CRÍTICAS DE AÇÃO

1) Rotinas que agem sobre os contadores

a) INCTD (número do contador)

Descrição: Esta rotina incrementa de uma unidade o conteúdo do contador especificado, guardando o resultado nesse contador.

Parâmetros: Número do contador (1 a 4).

Retorno: Não há.

Condições de erro:

1) Número do contador inválido
2) O processamento não para em caso de estouro aritmético.
   O analista responsável deve ter conhecimento dos números utilizados e se precaver, porque alguns resultados podem ser destorcidos. Por exemplo, se o conteúdo do contador for igual a 255, o resultado da operação será igual a zero.

Algorítmo: Contador = Contador + 1

b) DECTD (número do contador)

Descrição: Esta rotina decrementa de uma unidade o conteúdo do contador especificado, guardando o resultado nesse contador.

Parâmetros: Número do contador (1 a 4)

Retorno: Não há

Condições de erro:

1) Número do contador inválido
2) O processamento não para em caso de estouro aritmético. O analista responsável deve ter conhecimento dos

números utilizados e se precaver , porque alguns resultados podem ser destorcidos. Por exemplo, se o conteúdo do contador for igual a zero o resultado da operação será igual a 255.

Algoritmo: Contador = Contador - 1

c) MVCTD (número do contador, constante)

Descrição: Esta rotina converte a constante , especificada no segundo parâmetro, de ASCII para binário, e a move para o contador designado pelo primeiro parâmetro.

Parâmetros: 1) Número do contador (1 a 4)
2) Constante (valor numérico inteiro sem sinal)

Retorno: Não há

Condições de erro: 1) Número do contador inválido
2) Constante fora do intervalo 0 a 255

Algoritmo: Contador = Constante

d) ADCTD (número do contador, constante)

Descrição: Esta rotina converte a constante , especificada no segundo parâmetro, de ASCII para binário, e a soma ao conteúdo do contador designado pelo primeiro parâmetro, guardando o resultado nesse contador.

Parâmetros: 1) Número do contador (1 a 4)
2) Constante (valor numérico inteiro sem sinal)

Retorno: Não há

Condições de erro:

1) Número do contador inválido
2) Constante fora do intervalo Ø a 255
3) A rotina não para o processamento em caso de estouro aritmético. O analista responsável deve ter conhecimento dos números utilizados e se precaver, porque alguns resultados podem ser destorcidos. Por exemplo, a soma de 255 mais 1 dará resultado igual a zero.

Algoritmo: Contador = Contador + Constante

e) SUCTD (número do contador, constante)

Descrição: Esta rotina converte a constante, especificada no segundo parâmetro, de ASCII para binário, e a subtrai do conteúdo do contador designado pelo primeiro parâmetro, guardando o resultado nesse contador.

Parâmetros: 1) Número do Contador (1 a 4)
2) Constante (valor numérico inteiro sem sinal)

Retorno: Não há

Condições de erro:

1) Número do contador inválido
2) Constante fora do intervalo 0 a 255
3) A rotina não para o processamento em caso de estouro aritmético. O analista responsável deve ter conhecimento dos números utilizados e se precaver porque alguns resultados podem ser destorcidos. Por exemplo, a subtração de zero menos 1 dará resultado igual a 255

Algoritmo: Contador = Contador - Constante

f) MCCTD (número do contador)

Descrição: Esta rotina converte os três digitos de mais baixa ordem do último campo digitado, de ASCII para binário, e move o número obtido para o contador especificado.

Parâmetros: Número de contador (1 a 4)

Retorno: Não há

Condições de erro: 1) Número do contador inválido
2) Campo associado não numérico
3) Valor a ser convertido e movido maior que 255

Algoritmo: Contador = campo.

g) ACCTD (número do contador)

Descrição: Esta rotina converte os três dígitos de mais baixa ordem do último campo digitado, de ASCII para binário, e soma o número obtido ao conteúdo do contador especificado , guardando o resultado nesse contador.

Parâmetros: Número do contador (1 a 4)

Retorno: Não há

Condições de erro:

1) Número do contador inválido
2) Campo associado não numérico
3) Valor a ser convertido maior que 255
4) A rotina não para o processamento em caso de estouro aritmético. O analista responsável deve ter conhecimento dos números utilizados e se precaver porque alguns resultados podem ser

destorcidos. Por exemplo, a soma de 255 mais 1 dará resultado igual a zero.

Algoritmo: Contador = Contador + campo

h) SCCTD (Número do contador)

Descrição: Esta rotina converte os três digitos de mais baixa ordem do último campo digitado, de ASCII para binário, e subtrai o número obtido do conteúdo do contador especificado, guardando o resultado nesse contador.

Parâmetros: Número de contador (1 a 4)

Retorno: Não há

Condições de erro:

1) Número do contador inválido
2) Campo associado não numérico
3) Valor a ser convertido maior que 255
4) A rotina não para o processamento em caso de estouro aritmético. O analista responsável deve ter conhecimento dos números utilizados e se precaver, porque alguns resultados podem ser destorcidos. Por exemplo, a subtração de zero menos 1 dará resultado igual a 255.

Algoritmo: Contador = Contador - Campo

2) Rotinas que agem sobre os acumuladores

a) INACC (número do acumulador)

Descrição: Esta rotina incrementa de uma unidade o conteúdo do acumulador

especificado, guardando o resultado nesse acumulador.

Parâmetros: Número do acumulador (1 ou 2)

Retorno: Não há

Condições de erro:

1) Número do acumulador inválido
2) A rotina para o processamento em caso de estouro aritmético.

Algorítmo: Acumulador = Acumulador + 1

b) DEACC (número do acumulador)

Descrição: Esta rotina decrementa de uma unidade o conteúdo do acumulador especificado, guardando o resultado nesse acumulador.

Parâmetros: Número do acumulador (1 ou 2)

Retorno: Não há

Condições de erro:

1) Número do acumulador inválido
2) A rotina para o processamento em caso de estouro aritmético.

Algorítmo: Acumulador = Acumulador - 1

c) MVACC (número do acumulador, constante)

Descrição: Esta rotina move a constante especificada no segundo parâmetro para o acumulador designado peprimeiro parâmetro.

Parâmetros: 1) Número do acumulador (1ou2)
2) Constante (Valor numérico inteiro sem sinal)

Retorno: Não há

Condições de erro:

1) Número do acumulador inválido

2) Constante com mais de 15 dígitos decimais.

3) Constante não numérica

Algorítmo: Acumulador = Constante.

d) ADACC (número do acumulador, constante)

Descrição: Esta rotina soma a constante especificada no segundo parâmetro ao conteúdo do acumulador designado pelo primeiro parâmetro , guardando o resultado nesse acumulador.

Parâmetros: 1) Número do acumulador (1ou2)
2) Constante (valor numérico inteiro sem sinal)

Retorno: Não há

Condições de erro:

1) Número do acumulador inválido
2) Constante com mais de 15 dígitos decimais
3) Constante não numérica
4) A rotina para o processamento em caso de estouro aritmético.

Algorítmo: Acumulador = Acumulador + Constante

e) SUACC (número do acumulador, constante)

Descrição: Esta rotina subtrai a constante especificada no segundo parâmetro, do conteúdo do acumulador, designado no primeiro parâmetro guardando o resultado nesse acumulador.

Parâmetros: 1) Número do acumulador (1ou2)
2) Constante (valor numérico in

teiro sem sinal)

Retorno: Não há

Condições de erro:

1) Número do acumulador inválido
2) Constante com mais de 15 dígitos decimais
3) Constante não numérica
4) A rotina para o processamento em caso de estouro aritmético.

Algorítmo = Acumulador = Acumulador - Constante

f) MCACC (número do acumulador)

Descrição: Esta rotina move o conteúdo do último campo digitado para o acumulador especificado

Parâmetros: Número do acumulador (1 ou 2)

Retorno: Não há

Condições de erro:

1) Número do acumulador inválido
2) Campo com mais de 15 dígitos decimais
3) Campo associado não numérico

Algorítmo: Acumulador = Campo.

g) ACACC (número do acumulador)

Descrição: Esta rotina soma o conteúdo do último campo digitado ao conteúdo do acumulador especificado , guardando o resultado nesse acumulador.

Parâmetros: Número do acumulador (1 a 2)

Retorno: Não há

Condições de erro:

1) Número do acumulador inválido
2) Campo com mais de 15 dígitos decimais
3) Campo associado não numérico
4) A rotina para o processamento em caso de estouro aritmético

Algorítmo: Acumulador = Acumulador + Campo

h) SCACC (número do acumulador)

Descrição: Esta rotina subtrai o conteudo do último campo digitado do conteúdo do acumulador especificado, guardando o resultado nesse acumulador.

Parâmetros: Número do acumulador (1 ou 2)

Retorno: Não há

Condições de erro:
1) Número do acumulador inválido
2) Campo com mais de 15 dígitos decimais
3) Campo associado não numérico
4) A rotina para o processamento em caso de estouro aritmético.

Algorítmo: Acumulador = Acumulador - Campo

i) MMACC (número do acumulador, número do acumulador)

Descrição: Esta rotina move o conteúdo do acumulador especificado no segundo parâmetro para o acumulador designado pelo primeiro parâmetro.

Parâmetros: 1) Número do acumulador (1ou2)
2) Número do acumulador (1ou2)

Retorno: Não há

Condições de erro: 1) Número do acumulador inválido

Algorítmo: Acumulador = Acumulador

IV.3- DESCRIÇÃO DAS CRÍTICAS DE VERIFICAÇÃO

1) COCTD (número do contador, comparação, número do contador)

Descrição: Esta rotina compara o conteúdo do contador especificado no primeiro parâmetro com o conteúdo, do contador designado pelo terceiro parâmetro, de acordo com o tipo de comparação dado no segundo parâmetro.

Parâmetros:

1) Número do contador (1 a 4)
2) Tipo de comparação, composto de duas letras e especificado desta forma:
   IG - igual
   DI - diferente
   ME - menor
   EI - menor ou igual
   MA - maior
   AI - maior ou igual
3) Número do contador (1 a 4)

Retorno:

Se o resultado da comparação for verdadeiro o retorno é bem sucedido.
Se o resultado da comparação for falso o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um avi-

so sonoro e aparece no vídeo a mensagem de erro "REDIGITE". Certificando-se do erro, o operador aperta a tecla ERRO, o campo é apagado do vídeo e continua-se com a redigitação do mesmo campo. Se o erro persistir, ficando o processamento preso neste campo, seguidamente, significa que foi detetado um erro, cuja causa pode ter sido a digitação incorreta de um campo anterior, ou o campo atual não cruza com um ou mais campos anteriores. Neste caso, para correção, o operador aperta a tecla APAGA FORMULÁRIO, e procede à digitação do documento novamente.

Condições de erro:

1) Número do contador inválido
2) Tipo de comparação inválida

Algorítmo: Elementar

2) COACC (número do acumulador, comparação, número do acumulador)

Descrição: Esta rotina compara o conteúdo do acumulador especificado no primeiro parâmetro com o conteúdo, do conteúdo do acumulador designado pelo terceiro parâmetro, de acordo com o tipo de comparação dado no segundo parâmetro.

Parâmetros:

1) Número do acumulador (1 ou 2)
2) Tipo de comparação, composto de duas letras e especificado desta forma:

IG - igual

DI - diferente
ME - menor
EI - menor ou igual
MA - maior
AI - maior ou igual

3) Número do acumulador (1 ou 2)

Retorno:

Se o resultado da comparação for verdadeiro, o retorno é bem sucedido.
Se o resultado da comparação for falso o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e aparece no vídeo a mensagem de erro "REDIGITE". Certificando-se do erro, o operador aperta a tecla ERRO, o campo é apagado do vídeo e continua-se com a redigitação do mesmo campo. Se o erro persistir, ficando o processamento preso neste campo, seguidamente, significa que foi detetado um erro, cuja causa pode ter sido a digitação incorreta de um campo anterior, ou o campo atual não cruza com um ou mais campos anteriores. Neste caso, para correção, o operador aperta a tecla APAGA FORMULÁRIO, e procede à digitação do documento novamente.

Condições de erro:

1) Número do contador inválido
2) Tipo de comparação inválida

Algorítmo: Elementar

3) CTCTD (número do contador, comparação,constante)

Descrição: Esta rotina converte a constante especificada no terceiro parâmetro, de ASCII para binário, e compara o número obtido com o conteúdo do contador designado pelo primeiro parâmetro, de acordo com o tipo de comparação dado no segundo parâmetro.

Parâmetros:

1) Número do contador (1 a 4)
2) Tipo de comparação, composto de duas letras e especificado desta forma:
   IG - igual
   DI - diferente
   ME - menor
   EI - menor ou igual
   MA - maior
   AI - maior ou igual
3) Número do acumulador (1 ou 2)

Retorno:

Se o resultado da comparação for verdadeiro, o retorno é bem sucedido.
Se o resultado da comparação for falso o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e aparece no vídeo a mensagem de erro "REDIGITE". Certificando-se do erro, o operador aperta a tecla ERRO, o campo é apagado do vídeo e continua-se com a redigitação do mesmo campo. Se o erro persistir, ficando o processamento preso neste campo, seguidamente, significa que

foi detetado um erro, cuja causa pode ter sido a digitação incorreta de um campo anterior, ou o campo atual não cruza com um ou mais campos anteriores. Neste caso, para correção, o operador aperta a tecla APAGA FORMULÁRIO, e procede à digitação do documento novamente.

Condições de erro:

1) Número do contador inválido
2) Tipo de comparação inválida
3) Constante não numérica
4) Constante fora do intervalo 0 a 255

Algorítmo: Elementar

4) CTACC (número do acumulador, comparação , constante)

Descrição: Esta rotina compara o conteúdo da constante especificada no terceiro parâmetro com o conteúdo do acumulador designado pelo primeiro parâmetro, de acordo com o tipo de comparação dado no segundo parâmetro.

Parâmetros:

1) Número do acumulador (1 ou 2)
2) Tipo de comparação, composto de duas letras e especificado desta forma:

   IG - igual
   DI - diferente
   ME - menor
   EI - menor ou igual
   MA - maior
   AI - maior ou igual

Retorno:

Se o resultado da comparação for verdadeiro, o retorno é bem sucedido.
Se o resultado da comparação for falso o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e aparece no vídeo a mensagem de erro "REDIGITE". Certificando-se do erro, o operador aperta a tecla ERRO, o campo é apagado do vídeo e continua-se com a redigitação do mesmo campo. Se o erro persistir, ficando o processamento preso neste campo, seguidamente, significa que foi detetado um erro, cuja causa pode ter sido a digitação incorreta de um campo anterior, ou o campo atual não cruza com um ou mais campos anteriores. Neste caso, para correção, o operador aperta a tecla APAGA FORMULÁRIO, e procede à digitação do documento novamente.

Condições de erro:

1) Número do acumulador inválido
2) Tipo de comparação inválida
3) Constante não numérica
4) Constante com mais de 15 dígitos decimais.

Algorítmo: Elementar

5) CCCTD (número do contador, comparação)

Descrição: Esta rotina converte os três dígitos de mais baixa ordem do último campo digitado, de ASCII para binário, e compara o número

obtido com o conteúdo do contador especificado no primeiro parâmetro, de acordo com o tipo de comparação dado pelo segundo parâmetro.

Parâmetros:

1) Número do contador (1 a 4)
2) Tipo de comparação, composto de duas letras e especificado desta forma:
   IG - igual
   DI - diferente
   ME - menor
   EI - menor ou igual
   MA - maior
   AI - maior ou igual

Retorno:

Se o resultado da comparação for verdadeiro, o retorno é bem sucedido.
Se o resultado da comparação for falso o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e aparece no vídeo a mensagem de erro "REDIGITE". Certificando-se do erro, o operador aperta a tecla ERRO, o campo é apagado do vídeo e continua-se com a redigitação do mesmo campo. Se o erro persistir, ficando o processamento preso neste campo, seguidamente significa que foi detetado um erro, cuja causa pode ter sido a digitação incorreta de um campo anterior, ou o campo atual não cruza com um ou mais campos anteriores. Neste caso, para correção, o operador aperta a tecla APAGA FORMULÁRIO, e procede à digitação do docu

mento novamente.

Condições de erro:

1) Número do contador inválido
2) Tipo de comparação inválida
3) Campo associado não numérico
4) Valor a ser convertido maior que 255

Algorítmo: Elementar

6) CCACC (número do acumulador, comparação)

Descrição: Esta rotina compara o conteúdo do último campo digitado com o conteúdo do acumulador especificado no primeiro parâmetro, de acordo com o tipo de comparação dado pelo segundo parâmetro.

Parâmetros:

1) Número do acumulador (1 ou 2)
2) Tipo de comparação, composto de duas letras e especificado desta forma:
   IG - igual
   DI - diferente
   ME - menor
   EI - menor ou igual
   MA - maior
   AI - maior ou igual

Retorno:

Se o resultado da comparação for verdadeiro, o retorno é bem sucedido.
Se o resultado da comparação for falso o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e aparece no vídeo a mensagem de erro "REDIGITE".Certificando-

-se do erro, o operador aperta a tecla ERRO, o campo é apagado do vídeo e continua-se com a redigitação do mesmo campo. Se o erro persistir, ficando o processamento preso neste campo, seguidamente, significa que foi detetado um erro, cuja causa pode ter sido a digitação incorreta de um campo anterior, ou o campo atual não cruza com um ou mais campos anteriores. Neste caso, para correção, o operador aperta a tecla APAGA FORMULÁRIO, e procede à digitação do documento novamente.

Condições de erro:

1) Número do acumulador inválido
2) Tipo de comparação inválida
3) Campo associado não numérico
4) Campo com mais de 15 dígitos decimais

Algorítmo: Elementar

7) TAMIN (limite)

Descrição: Esta rotina testa o tamanho mínimo de um campo.

Parâmetro: Limite (número mínimo de posições a serem digitadas)

Retorno:

Se o número de caracteres digitados para o campo for maior ou igual ao limite, o retorno é bem sucedido.
Se o número de caracteres digitados, para o campo for menor que o limite, o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso

sonoro e, aparece no vídeo a mensagem de erro "TAMANHO INSUFICIENTE".

Condições de erro:

1) Limite maior que o tamanho do campo. Se isto acontecer o processamento ficará preso nesta rotina.

Algorítmo: Elementar

8) MOD10 (posição)

Descrição: Esta rotina testa se o algarismo na posição do campo especificada pelo número passado como parâmetro é um digito verificador módulo 10 correto. O cálculo é feito sobre os algarismos, que precedem a posição, dentro do campo, dado como parâmetro. A rotina pode ser chamada mais de uma vez para o mesmo campo.

Parâmetros: Posição (número de ordem do dígito verificador dentro do campo).

Retorno:

Se o resultado do teste for verdadeiro retorno é bem sucedido.

Se o resultado do teste for falso o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e aparece no vídeo a mensagem de erro "DIGITO VERIFICADOR".

Condições de erro:

1) Campo não numérico
2) Número da posição menor que 1
3) Número da posição maior que o tamanho do campo.

Algorítmo:

1) Todos os dígitos do campo são convertidos de ASCII para binário, formando-se um vetor a, de n números, onde a_n é o dígito verificador a ser testado.
   Seja o vetor assim formado:
   $a_1, a_2, a_3, \ldots, a_{n-1}, a_n$
2) Se n for par, multiplica-se por 2 os elementos:
   $a_1, a_3, \ldots, a_{n-3}, a_{n-1}$
   Se n for ímpar, multiplica-se por 2 os elementos:
   $a_2, a_4, \ldots, a_{n-3}, a_{n-1}$
3) Acha-se a soma S igual:
   $S = 2xa_1 + a_2 + 2xa_3 + \ldots + 2xa_{n-3} + a_{n-2} + 2xa_{n-1}$ (npar)
   ou
   $S = a_1 + 2xa_2 + a_3 + 2xa_4 + \ldots + 2xa_{n-3} + a_{n-2} + 2xa_{n-1}$
   (n ímpar)
4) Analisa-se o resultado de cada parcela da multiplicação. Se o resultado for maior que 9, subtrai- se 9 dessa parcela. Após, somam- se todas as parcelas, obtendo-se assim a soma final S.
5) Divide-se a soma S por 10 e acha-se o resto da divisão.
6) Se o resto é zero e a_n=0, o dígito verificador está correto. Se o resto é maior que zero e a_n= 10 - resto, o dígito verificador está correto. Senão, o dígito verificador é inválido.

Observações:

Esta rotina pode ser usada para cal-

cular o primeiro dígito de controle na oitava posição, do número de inscrição no Cadastro Geral de Contribuintes (CGC) do Ministério da Fazenda. Exemplo da declaração de críti - ca: CR, MOD10 (8);

9) MOD11 (posição)

Descrição: Esta rotina testa se o algarismo na posição do campo especificada pelo número passado como parâmetro é um dígito verificador módulo 11 correto. O cálculo é feito sobre os algarismos que precedem a posição, dentro do campo, dado como parâmetro . A rotina pode ser chamada mais de uma vez para o mesmo campo.

Parâmetros: Posição (número de ordem do dígito verificador dentro do campo)

Retorno:

Se o resultado do teste for verdadeiro o retorno é bem sucedido.

Se o resultado do teste for falso o o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e aparece no vídeo a mensagem de erro "DIGITO VERIFICADOR".

Condições de erro:

1) Campo não numérico
2) Número da posição menor que 1
3) Número da posição maior que o tamanho do campo

Algorítmo:

1) Todos os dígitos do campo são convertidos de ASCII para binário , formando-se um vetor a , de n números, onde a_n é o dígito verificador a ser testado. Seja o vetor assim formado:

   $a_1, a_2, a_3, \ldots, a_{n-2}, a_{n-1}, a_n$

2) Multiplica-se cada algarismo do campo por um peso, da direita para esquerda, começando-se com o peso 2 e incrementando de uma unidade o peso, ao passarmos para a coluna seguinte à esquerda. Ao alcançar 9 o peso não passa a 10, e sim, volta a 2 e continua sendo incrementado de uma unidade.

3) Obtem-se a soma S igual a:

   $$S = 2xa_{n-1} + 3xa_{n-2} + 4xa_{n-3} + \ldots + 8xa_{n-7} + 9xa_{n-8} + 2xa_{n-9} + 3xa_{n-10} + 4xa_{n-11} + \ldots\ldots$$

4) Divide-se a soma S por 11 e acha-se o resto da divisão.

5) Se o resto é igual a zero ou resto é igual a um e, $a_n=0$, o dígito verificador está correto.
   Se o resto é maior que um e $a_n=11-$ resto, o dígito verificador está correto. Senão, o dígito verificador é inválido.

Observações:

Esta rotina pode ser usada para calcular o segundo e terceiro dígitos de controle, na 13ª e 14ª posições respectivamente, do número de inscrição no Cadastro Geral de Contribuintes (CGC) do Ministério da Fazenda. Exem

plo da declaração de crítica:
CR, MOD11(13), MOD11(14);

10) MZCPF (posição)

Descrição: Esta rotina testa se o algarismo na posição do campo especificada pelo número passado como parâmetro é um dígito verificador módulo 11 correto. O cálculo é feito sobre os algarismos que precedem a posição, dentro do campo, dado como parâmetro. A rotina pode ser chamada mais de uma vez para o mesmo campo.

Parâmetros: Posição (número de ordem do dígito verificador dentro do campo).

Retorno:

Se o resultado do teste for verdadeiro o retorno é bem sucedido.
Se o resultado do teste for falso o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e aparece no vídeo a mensagem de erro "DIGITO VERIFICADOR".

Condições de erro:

1) Campo não numérico
2) Número da posição menor que 1
3) Número da posição maior que o tamanho do campo.

Algorítmo:

1) Todos os dígitos do campo são convertidos de ASCII para binário, formando-se um vetor a, de n números, onde a_n é o dígito verifica-

dor a ser testado. Seja o vetor assim formado:

$a_1, a_2, a_3, \ldots a_{n-2}, a_{n-1}, a_n$

2) Multiplica-se cada algarismo do campo por um peso, da direita para esquerda, começando-se com o peso 2 e incrementando de uma unidade o peso, ao passarmos para a coluna seguinte à esquerda. Ao alcançar 9 o peso não volta a 2, e sim, passa a 10 e continua sendo incrementado de uma unidade.

3) Obtem-se a soma S igual a:

$$S= 2xa_{n-1} + 3xa_{n-2} + 4xa_{n-3} + \ldots + 8xa_{n-7} + 9xa_{n-8} + 10xa_{n-9} + 11xa_{n-10} + \ldots$$

4) Divide-se a soma S por 11 e acha-se o resto da divisão.

5) Se o resto é igual a zero ou resto é igual a um e, $a_n=0$, o digito verificador está correto.
Se o resto é maior que um e $a_n=11-$ resto, o dígito verificador está correto. Senão, o dígito verificador é inválido.

Observações:

Esta rotina pode ser usada para calcular os três dígitos de controle,na 8ª, 10ª e 11ª posições, do número de inscrição no Cadastro de Pessoas Físicas (CPF) do Ministério da Fazenda.
Exemplo da declaração de crítica:
CR, MZCPF(8), MZCPF(10), MZCPF(11);

11) DATA

Descrição: Esta rotina verifica a validade

de uma data com 6 posições, digitada no formato DDMMAA, considerando ano bissexto.

Parâmetros: Não há

Retorno:

Se a data for válida o retorno é bem sucedido.

Se a data for inválida o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e aparece no vídeo a mensagem de erro "DATA INVÁLIDA".

Condições de erro:

1) Campo não numérico
2) Campo com mais de 6 ou menos de 6 posições ou fora do formato DDMMAA

Algorítmo:

1) Converte ano e mês para binário
2) Se mês menor que 1 ou mês maior que 12 então retorna com erro.
3) Converte dia para binário
4) Se dia menor que 1 ou dia maior que 31 então retorna com erro.
5) Se dia = 31 e mês = ABR ou JUN ou SET ou NOV então retorna com erro.
6) Se mês não é FEV retorna
7) Se ano é bissexto e dia maior que 29 então retorna com erro
8) Se ano não é bissexto e dia maior que 28 então retorna com erro, se não retorno bem sucedido

12) DATAI

Descrição: Esta rotina verifica a validade de uma data com 6 posições inver

tida, ou seja, digitada no formato AAMMDD, considerando ano bissexto.

Parâmetros: Não há

Retorno:

Se a data for válida o retorno é bem sucedido.

Se a data for inválida o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e, aparece no vídeo a mensagem de erro "DATA INVÁLIDA".

Condições de erro:

1) Campo não numérico
2) Campo com mais de 6 ou menos de 6 posições ou, fora do formato AAMMDD.

Algorítmo:

1) Troca de posição ano com dia
2) Chama rotina DATA

13) DATA4

Descrição: Esta rotina verifica a validade de uma data de 8 posições (ano com 4 dígitos), digitada no formato DDMMAAAA, considerando ano bissexto.

Parâmetros: Não há

Retorno:

Se a data for válida o retorno é bem sucedido.

Se a data for inválida o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e, apa

rece no vídeo a mensagem de erro "DATA INVÁLIDA".

Condições de erro:

1) Campo não numérico
2) Campo com mais de 8 ou menos de 8 posições ou, fora do formato DDMMAAAA.

Algorítmo:

1) Converte ano para binário
2) Se ano menor que 1900 ou ano maior que 2099 então retorna com erro.
3) Converte mês para binário
4) Se mês menor que 1 ou mês maior que 12 então retorna com erro.
5) Converte dia para binário
6) Se dia menor que 1 ou dia maior que 31 então retorna com erro
7) Se dia = 31 e mês = ABR ou JUN ou SET ou NOV então retorna com erro
8) Se mês não é FEV retorna
9) Se ano é bissexto e dia maior que 29 então retorna com erro
10) Se ano não é bissexto e dia maior que 28 então retorna com erro, se não retorno bem sucedido.

14) DAT4I

Descrição: Esta rotina verifica a validade de uma data de 8 posições (ano com 4 dígitos) invertida, ou seja, digitada no formato AAAAMMDD considerando ano bissexto.

Parâmetros: Não há

Retorno:

Se a data for válida o retorno é bem

sucedido.

Se a data for inválida o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e, aparece no vídeo a mensagem de erro "DATA INVÁLIDA".

Condições de erro:

1) Campo não numércio
2) Campo com mais de 8 ou menos de 8 posições ou, fora do formato AAAAMMDD.

Algorítmo:

1) Troca de posição ano com dia
2) Chama rotina DATA4.

15) FAIXA (limite inferior, limite superior)

Descrição: Esta rotina verifica se o conteúdo de um campo se situa numa faixa de valores onde o limite inferior, inclusive, é dado pelo primeiro parâmetro e o limite superior, inclusive, é dado pelo segundo parâmetro.

Parâmetros:

1) Limite inferior: cadeia numérica ou mesmo não numérica fornecendo o limite inferior do intervalo.
2) Limite superior: cadeia numérica ou mesmo não numérica fornecendo o limite superior do intervalo.

Retorno:

Se o conteúdo do campo está dentro do intervalo aberto definido pelos dois parâmetros, o retorno é bem sucedido.
Se o conteúdo do campo está fora do

intervalo aberto definido pelos dois parâmetros o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e, aparece no vídeo do terminal a mensagem de erro "FORA DA FAIXA PERMITIDA"

Condições de erro:

1) Cadeia de caracteres que forma o limite inferior não tem o mesmo tamanho do campo.
2) Cadeia de caracteres que forma o limite superior não tem o mesmo tamanho do campo.
3) Limite inferior maior que o limite superior. Se isto acontecer o processamento ficará preso nesta rotina.

Algorítmo:

1) Se o conteúdo campo é menor do que o limite inferior retorna com erro
2) Se o conteúdo campo é maior que o limite superior retorna com erro.
3) O conteúdo é válido pois está dentro da faixa permitida. Retorno bem sucedido.

16) TAB ('cadeia 1', 'cadeia 2', 'cadeia 3',.. 'cadeia n')

Descrição: Esta rotina verifica se o conteúdo do último campo digitado é elemento de uma tabela fornecida. A pesquisa é sequencial até ser encontrado um elemento igual ao conteúdo do campo ou até o final da tabela.

Parâmetros:

Cada parâmetro é um elemento da tabela. Cada elemento da tabela é uma cadeia de caracteres. Todos os elementos da tabela dada são de mesmo tamanho em caracteres, e igual ao tamanho do último campo digitado. O número de parâmetros, ou seja, o número de elementos da tabela é ilimitado.

Retorno:

Se o conteúdo do campo é um elemento da tabela o retorno é bem sucedido.
Se o conteúdo do campo não foi encontrado, isto é, não é um elemento da tabela, o retorno se faz através da chamada de uma rotina de erro. O operador é alertado por intermédio de um aviso sonoro e, aparece no vídeo do terminal a mensagem de erro "FORA DE TABELA"

Condições de erro:

1) Cadeia de caracteres fornecida como parâmetro não tem o mesmo tamanho do campo.

Algorítmo:

1) Ler um elemento da tabela. Se chegou no fim da tabela retorna com erro
2) Se o conteúdo do campo é diferente do elemento da tabela volte ao passo anterior.
3) Foi encontrado um elemento da tabela com o mesmo conteúdo do campo. Retorno bem sucedido.

IV.4- ROTINAS DE CRÍTICA ESPECIAIS.

Essas rotinas especiais não produzem retorno e, não causam a emissão de mensagens de erro. Portanto, elas são semelhantes às rotinas de ação neste ponto e, agem da mesma forma. A diferença consiste no fato de produzirem alterações no campo recem digitado.

1) SUBST ('carater de pesquisa', 'carater de substituição')

Descrição: Esta rotina pesquisa todas as posições do último campo digitado, carater a carater. Quando é encontrado um carater do campo igual ao carater de pesquisa, especificado no primeiro parâmetro, o carater do campo é trocado pelo carater de substituição, especificado no segundo parâmetro. Mesmo após efetuada uma troca a comparação continua até que todas as posições do campo tenham sido pesquisadas.

Parâmetros:

1) carater de pesquisa (qualquer codificação válida em ASCII)
2) carater de substituição (qualquer codificação válida em ASCII)

Retorno: Não há.

Condições de erro: Não há.

Algorítmo: Elementar.

2) APAGA

Descrição: Esta rotina apaga todas as posições do último campo digita-

do. O campo passa a não exis - tir no arquivo de saída.

Parâmetros: Não há.

Retorno: Não há.

Condições de erro: Não há.

Algorítmo: Elementar

3) COBRE ('carater')

Descrição: Esta rotina preenche todo o último campo digitado com um carater fornecido. Todas as posições do campo são apagadas e substituidas pelo carater dado como parâmetro.

Parâmetros: carater de preenchimento (qualquer codificação válida em ASCII)

Retorno: Não há.

Condições de erro: Não há.

Algorítmo: Elementar.

IV.5- GLOSSÁRIO DE MENSAGENS DE ERRO

Número da mensagem	Significado
1	FITA NAO ESTA MONTADA
2	FIM DE FITA
3	FITA PROTEGIDA
4	FIM DE ARQUIVO
5	BLOCO MENOR
6	BLOCO MAIOR
7	ERRO NAO IDENTIFICADO
10	CARATER INVALIDO
11	ESTOURO DE TAMANHO
12	TAMANHO INSUFICIENTE
13	DIGITO DE CONTROLE
14	DATA INVALIDA
15	SALTO NAO PERMITIDO
16	FORA DA TABELA
17	FORA DA FAIXA PERMITIDA
18	FORMULARIO NAO ENCONTRADO
20	MONTE OUTRA FITA

V) EXEMPLOS DE PROGRAMAS

Exemplo 1

NCE/UFRJ - CADASTRAMENTO DE PESSOAL

NOME (----------) CPF (---------)

ENDEREÇO (----------) BAIRRO (----------)

CIDADE (----------) ESTADO (--) TELEFONE (----------)

DATA DO NASCIMENTO (------) IDENTIDADE (----------)

NACIONALIDADE (----------) SEXO (-) ESTADO CIVIL (-)

NOME DO DEPENDENTE	DATA DA DEPENDÊNCIA	RELAÇÃO DE DEPENDÊNCIA
(----------)	(------)	(----------)
(----------)	(------)	(----------)
(----------)	(------)	(----------)
(----------)	(------)	(----------)
(----------)	(------)	(----------)
(----------)	(------)	(----------)

NÚMERO DE DEPENDENTES (-)

OBS. DATA DE DEPENDÊNCIA ESPOSA: CASAMENTO, FILHOS NASCIMENTO

Para obter no vídeo um formulário com o formato acima, e especificar as rotinas de crítica a serem aplicadas nos vários campos, escreve-se o seguinte programa fonte.

```
FO,NCEØ25,'NCE/UFRJ - - CADASTRAMENTO DE PESSOAL ';
IN,'DP12';          CODIGO DE OPERAÇÃO
CA,'NOME',A,50,E,' ',N,1;
CR,TAMIN(5);
CA,'CPF ',N,11,F,'Ø',N,Ø;
CR,MZCPF(8), MZCPF(10),MZCPF(11);
CA,'ENDERECO ',Q,50,E,' ',N,1;
CR,TAMIN(6);
```

```
CA,'BAIRRO ',A,14,E,' ',S,Ø;
CA,'CIDADE ',A,20,E,' ',N,1;
CA,'ESTADO ',A,2,F,' ',N,Ø;
CR,TAB('AM','PA','GO','MT','MA','PI','RN','SE','PE','CE','BA',
       'MG','ES','RJ','SP','PR','SC','RS' );
CA,'TELEFONE',N,7,F,' ',S,Ø;
CR,FAIXA('2000000','3999999');
CA,'DATA DO NASCIMENTO ',N,6,F,'Ø',N,1;
CR,DATA;
CA,'IDENTIDADE ',L,18,E,' ',N,Ø;
CA,'NACIONALIDADE ',A,20,E,' ',N,1;
CA,'SEXO ',A,1,F,' ',N,Ø;
CR,TAB('M','F');
CA,'ESTADO CIVIL ',A,1,F,' ',N,Ø;
CR,TAB('S','C','V','D','O'),
   MVCTD(1,Ø),MVCTD(2,Ø);                          CTD1=0 e CTD2=Ø
CO,'    ',1;                          COLOCO UMA LINHA EM BRANCO
CO,'NOME DO DEPENDENTE',4;
CO,'DATA   DA',56;
CO,'RELAÇÃO  DE',69;
CO,'DEPENDÊNCIA',55;
CO,'DEPENDÊNCIA',69;
CA,'',A,50,E,' ',S,1;                  NOME DO DEPENDENTE
CR,INCTD(1),MVCTD(2,1);                      CTD1=CTD1+1 e CTD2=1
CA,'',N,6,F,'Ø',S,Ø;                   DATA DA DEPENDÊNCIA
CR,DATA,CTCTD(2,IG,1),INCTD(1);                          CTD1=CTD+1
CA,'',A,20,E,' ',S,Ø;                  RELAÇÃO DE DEPENDENCIA
CR,CTCTD(2,IG,1),INCTD(1),MVCTD(2,Ø);    CTD1=CTD1+1 e CTD2=Ø

CA,'',A,50,E,' ',S,1;                  NOME DO DEPENDENTE
CR,CTCTD(2,IG,0),INCTD(1),MVCTD(2,2);    CTD1=CTD1+1 e CTD2=2
CA,'',N,6,F,'Ø',S,Ø;                   DATA DA DEPENDENCIA
CR,DATA,CTCTD(2,IG,2),INCTD(1);                      CTD1=CTD1+1
CA,'';A,20,E,' ',S,Ø;                  RELACAO DE DEPENDENCIA
CR,CTCTD(2,IG,2),INCTD(1),MVCTD(2,Ø);    CTD1=CTD1+1 e CTD2=0

CA,'',A,50,E,' ',S,1;                  NOME DO DEPENDENTE
CR,CTCTD(2,IG,0),INCTD(1),MVCTD(2,3);    CTD1=CTD1+1 e CTD2=3
CA,'',N,6,F,'Ø',S,Ø;                   DATA DA DEPENDENCIA
```

```
CR,DATA,CTCTD(2,IG,3),INCTD(1);                          CTD1=CTD1+1
CA,'',A,20,E,' ',S,Ø;                                RELACAO DE DEPENDENCIA
CR,CTCTD(2,IG,3),INCTD(1),MVCTD(2,Ø);                  CTD1=CTD1+1 e CTD2=0

CA,'',A,50,E,' ',S,1;                                NOME DO DEPENDENTE
CR,CTCTD(2,IG,0),INCTD(1),MVCTD(2,4);                  CTD1=CTD1+1 e CTD2=4
CA,'',N,6,F,'Ø',S,Ø;                                 DATA DA DEPENDENCIA
CR,DATA,CTCTD(2,IG,4),INCTD(1);                          CTD1=CTD1+1
CA,'',A,20,E,' ',S,Ø;                                RELACAO DE DEPENDENCIA
CR,CTCTD(2,IG,4),INCTD(1),MVCTD(2,Ø);                  CTD1=CTD1+1 e CTD2=0

CA,'',A,50,E,' ',S,1;                                NOME DO DEPENDENTE
CR,CTCTD(2,IG,0),INCTD(1),MVCTD(2,5);                  CTD1=CTD1+1 e CTD2=5
CA,'',N,6,F,'Ø',S,Ø;                                 DATA DA DEPENDENCIA
CR,DATA,CTCTD(2,IG,5),INCTD(1);                          CTD1=CTD1+1
CA,'',A,20,E,' ',S,Ø;                                RELACAO DE DEPENDENCIA
CR,CTCTD(2,IG,5),INCTD(1),MVCTD(2,Ø);                  CTD1=CTD1+1 e CTD2=0

CA,'',A,50,E,' ',S,1;                                NOME DO DEPENDENTE
CR,CTCTD(2,IG,0),INCTD(1),MVCTD(2,6);                  CTD1=CTD1+1 e CTD2=6
CA,'',N,6,F,'Ø',S,Ø;                                 DATA DA DEPENDENCIA
CR,DATA,CTCTD(2,IG,6),INCTD(1);                          CTD1=CTD1+1
CA,'',A,20,E,' ',S,Ø;                                RELACAO DE DEPENDENCIA
CR,CTCTD(2,IG,6),INCTD(1),MVCTD(2,Ø);                  CTD1=CTD1+1 e CTD2=0

CA,'NUMERO DE DEPENDENTES',N,1,F,'Ø',N,1;
CR,FAIXA('Ø','6'),MCCTD(3),
   ACCTD(3),ACCTD(3),COCTD(1,IG,3);                  CTD3=3 x campo
CO,'OBS.DATA DE DEPENDENCIA:ESPOSA:CASAMENTO, ',1;
CO,'FILHOS:NASCIMENTO',Ø;
FIM,NCEØ3Ø;
```

Observações:

1) Foi colocado um campo inserido no início do programa. Este campo não é digitado. Poderia ser, por exemplo,um código de operação que identifica o registro no arquivo movimento.

2) O campo 'NOME' está associado à rotina de crítica de tamanho mínimo. A informação é aceita se forem digitados pelo menos 5 posições do campo.

3) O campo 'CPF' só é aceito se forem digitados corretamente o número completo de inscrição no Cadastro de Pessoas Físicas, com os seus 3 dígitos de controle.

4) O campo 'ENDERECO' também sofre crítica de tamanho mínimo e só é aceito se forem digitados pelo menos 6 posições do campo.

5) Os campos 'BAIRRO' e 'CIDADE' não passam por nenhuma validação, sendo que o primeiro tem digitação opcional, ou seja, o salto é permitido.

6) O campo 'ESTADO' sofre crítica de tabela e a informação só é aceita se for a sigla correta de uma unidade da federação.

7) O campo 'TELEFONE' tem digitação opcional, logo, é permitido apertar a tecla SALTA CAMPO. No entanto, se o campo for digitado sofre crítica de faixa de valores.

8) O campo 'ESTADO CIVIL' sofre crítica de tabela . Se o carater digitado for igual a um dos elementos da tabela o campo é aceito, e duas críticas de ação são executadas. Os contadores 1 e 2 são zerados, pois são utilizados nos campos seguintes.

9) Os campos 'NOME DO DEPENDENTE', 'DATA DA DEPENDENCIA' e 'RELAÇAO DE DEPENDENCIA' são repetidos seis vezes . Assim, foram usadas declarações de comentário para colocar os campos no formato vertical. A primeira declaração de comentário coloca uma linha em branco separando os campos anteriores dos três campos seguintes. As próximas declarações de comentário fornecem a descrição dos campos situados nas seis linhas abaixo. Deste modo, não precisamos repetir a descrição dos três campos seis vezes.

10) Todos os três campos, relativos aos dados dos dependentes, nas seis linhas, possuem salto permitido. Se o operador quiser poderá não digitar os 18 campos. Mas se for digitado o campo 'NOME DO DEPENDENTE', os dois campos seguintes ,

'DATA DA DEPENDENCIA' e 'RELACAO DE DEPENDENCIA', na mesma linha, tornam-se obrigatórios. Usamos o contador 2 para controlar a digitação nessa linha. Assim, ou todos os três campos da linha são preenchidos ou nenhum dos três. O contador 1 serve para indicar quantos campos foram preenchidos. Ao terminar a digitação de um campo, relativo aos dependentes, o contador 1 é incrementado de uma unidade.

11) Em 'NOME DO DEPENDENTE' testamos se o conteúdo do contador 2 é igual a zero. Se for diferente de zero, houve um erro de preenchimento, sendo o operador alertado através de um aviso sonoro, o processamento interrompido e, emitida a mensagem de erro "REDIGITE". Se igual a zero atribuimos ao contador 2, o número da linha, variando de 1 a 6.

12) Em 'DATA DA DEPENDENCIA' testamos se o conteúdo do contador 2 é igual ao número da linha, variando de 1 a 6. Se for diferente do número da linha, houve um erro de preenchimento, sendo o operador alertado através de um aviso sonoro, o processamento interrompido e, emitida a mensagem de erro "REDIGITE". Se for igual ao número da linha significa que o campo anterior foi digitado corretamente.

13) Em 'RELACAO DE DEPENDENCIA' testamos se o conteúdo do contador 2 é igual ao número da linha, variando de 1 a 6. Se for diferente do número da linha, houve um erro de preenchimento, sendo o operador alertado através de um aviso sonoro, o processamento interrompido e, emitida a mensagem de erro "REDIGITE". Se for igual ao número da linha significa que o campo 'NOME DO DEPENDENTE', nesta linha, foi digitado corretamente. Após o teste movemos zero para o contador 2.

14) O campo 'NUMERO DE DEPENDENTES' é de digitação obrigatória. Mesmo que não existam dependentes o campo deverá ser preenchido. O formulário aceita no máximo 6 dependentes . Se existirem mais dependentes deverão ser fornecidos em um formulário de continuação, com um código de operação diferente. O campo sofre crítica de faixa de valores, só se aceitando os números 0 a 6. O conteúdo do campo é movido para o contador 3.

15) O número de dependentes fornecido deve ser igual ao número de linhas preenchidas com os dados dos dependentes.

Temos 3 campos para cada dependente, então, o número de campos preenchidos deve ser igual ao triplo do número de dependentes dado. Para testar isso, fazemos primeiro o conteúdo do contador 3 ser igual ao campo 'NUMERO DE DEPENDENTES' e após, usando a rotina ACCTD, multiplicamos por 3 o conteúdo do contador 3. Como o contador 1 é utilizado para indicar quantos campos foram preenchidos, o teste final consiste em comparar, se o conteúdo do contador 1 é igual ao conteúdo do contador 3. Se o resultado da comparação for diferente significa que o número de dependentes fornecido não coincide com o número de linhas digitadas, ou, uma das linhas não foi preenchida corretamente. Se o resultado da comparação entre os contadores 1 e 3 for igual a digitação dos dados dos dependentes foi perfeita. O número de linhas preenchidas é igual ao número de dependentes fornecido e, em cada linha digitada, nenhum campo foi deixado em branco.

16) Coloca-se no final do formulário, por intermédio de uma declaração de comentário, uma observação para explicar ao operador como preencher um determinado campo.

17) O último comando é uma declaração de fim de programa, onde indicamos o nome do formulário seguinte. A sequenciação de formulários é automática. A mudança para o formulário seguinte é obtida com o simples aperto da tecla FORMULÁRIO SEGUINTE. Se o operador desejar continuar a digitação com o mesmo formulário atual é só apertar a tecla FIM DE FORMULÁRIO, após completar a digitação do último campo.

Exemplo 2

```
B N D E/DEFIN - ALTERACAO DE VALORES DE MOEDA
COD OP(--) DATA MOVIMENTO(------) LANÇAMENTOS(--)
COD MOEDA(--) DOLAR AMERICANO(----------)
COD MOEDA(--) LIBRA ESTERLINA(----------)
COD MOEDA(--) MARCO ALEMAO    (----------)
COD MOEDA(--) FRANCO FRANCES  (----------)
COD MOEDA(--) O.R.T.N.        (----------)
              TOTAL DE CONTROLE (----------)
```

Para obter no vídeo um formulário com o formato acima, e especificar as rotinas de crítica a serem aplicadas nos vários campos, escreve-se o seguinte programa fonte.

```
FO,BNDE3Ø,'B N D E/DEFIN - ALTERACAO DE VALORES DE MOEDA';
CA,'COD OP',N,2,F,'0',N,1;
CR,TAB('99),MVACC(1,Ø),MVCTD(2,Ø);                ACC1=0 e CTD2=Ø
CA,' DATA MOVIMENTO',N,6,F,'Ø',N,Ø;
CR,DATA;
CA,' LANCAMENTOS',N,2,F,'Ø',N,Ø;
CR,FAIXA('OO','05'),MCCTD(3);            CTD3 CONTEM LANCAMENTOS
CA,'COD MOEDA',N,2,F,'Ø',S,1;
CR,TAB('5Ø'),DECTD(3),MVCTD(2,1);          CTD3=CTD3-1 e CTD2=1
CA,' DOLAR AMERICANO',N,1Ø,D,'Ø',S,Ø;
CR,CTCTD(2,IG,1),ACACC(1),MVCTD(2,Ø); ACC1=ACC1+CAMPO e CTD2=Ø
CA,'COD MOEDA',N,2,F,'Ø',S,1;
CR,TAB('51'),DECTD(3),MVCTD(2,1);          CTD3=CTD3-1 e CTD2=1
CA,' LIBRA ESTERLINA',N,1Ø,D,'Ø',S,Ø;
CR,CTCTD(2,IG,1),ACACC(1),MVCTD(2,Ø); ACC1=ACC1+CAMPO e CTD2=Ø
CA,'COD MOEDA',N,2,F,'Ø',S,1;
CR,TAB('52'),DECTD(3),MVCTD(2,1);          CTD3=CTD3-1 e CTD2=1
CA,' MARCO ALEMAO  ',N,1Ø,D,'Ø',S,Ø;
CR,CTCTD(2,IG,1),ACACC(1),MVCTD(2,Ø); ACC1=ACC1+CAMPO e CTD2=Ø
CA,'COD MOEDA',N,2,F,'Ø',S,1;
CR,TAB('53),DECTD(3),MVCTD(2,1);           CTD3=CTD3-1 e CTD2=1
```

```
CA,'FRANCO FRANCES ',N,1Ø,D,'Ø,S,Ø;
CR,CTCTD(2,IG,1),ACACC(1),MVCTD(2,Ø); ACC1=ACC1+CAMPO e CTD2=Ø
CA,'COD MOEDA',N,2,F,'Ø,S,1;
CR,TAB('54'),DECTD(3),MVCTD(2,1);          CTD3=CTD3-1 e CTD2=1
CA,'O.R.T.N.        ',N,1Ø,D,'Ø',S,Ø;
CR,CTCTD(2,IG,1),ACACC(1);                          ACC1=ACC1+CAMPO
CA,'TOTAL  DE  CONTROLE        ',N,1Ø,D,'Ø',N,12;
CR,CTCTD(3,IG,0),CCACC(1,IG);
FIM,BNDE40;
```

Observações:

1) Neste formulário, o campo 'LANCAMENTOS' deve ser preenchido com o número de moedas que sofreram alteração de valor no dia. O campo 'TOTAL DE CONTROLE' deve conter o somatório de valores de moeda preenchidos.

2) No campo 'LANCAMENTOS' fazemos o contador 3 ser igual ao conteúdo do campo. Para cada campo 'COD MOEDA' digitado, decrementamos de uma unidade o conteúdo do contador 3 . No final, comparamos se o conteúdo do contador é igual à constante zero. Se o resultado da comparação for diferente, houve um erro de preenchimento, sendo o operador alertado através de um aviso sonoro, o processamento interrompido e, emitida a mensagem de erro 'REDIGITE'. Assim, verificamos se o número de lançamentos fornecidos coincide com o número de lançamentos digitados.

3) O contador 2 só permite a digitação do campo valor caso o campo código da moeda tenha sido digitado.

4) Após o preenchimento de cada campo valor da moeda, adicionamos ao acumulador 1 o valor digitado. No final, o conteúdo do campo 'TOTAL DE CONTROLE' é comparado com o acumulador 1. Se o resultado da comparação for diferente, houve um erro de preenchimento, sendo o operador alertado através de um aviso sonoro, o processamento interrompido e, emitida a mensagem de erro 'REDIGITE'. Deste modo, verificamos se o campo 'TOTAL DE CONTROLE' realmente contem o somatório dos valores de moeda digitados.

5) Todos os cinco campos, código da moeda e valor da moeda, possuem salto permitido. Se no dia, não houve alteração de moeda, isso pode ser informado ao sistema, saltando-se os cinco campos códigos e valor de moeda, preenchendo-se o campo 'LANCAMENTOS', obrigatório, com o valor zero, e o campo 'TOTAL DE CONTROLE', também obrigatório, com o valor zero. O programa de crítica, conforme codificado, aceita o formulário digitado desta forma.

6) O último comando é uma declaração de fim de pro - grama, onde indicamos o nome do formulário seguinte. A sequenciação de formulários é automática. A mudança para o formulário seguinte é obtida com o simples aperto da tecla FORMULARIO SEGUINTE. Se o operador desejar continuar a digitação com o mesmo formulário atual é só apertar a tecla FIM DE FORMULARIO, após completar a digitação do último campo.

VI) LINGUAGEM DE CRIAÇÃO DE FORMULÁRIOS OBJETO

A partir de um programa fonte, escrito na linguagem de criação de formulários fonte, são formados registros compactados de tamanho fixo de 16 bytes. Esses registros são analizados pelo programa principal e contém todas as informações sobre um formulário a ser projetado no vídeo.

Chamamos a reunião desses registros de programa objeto . Um programa objeto é composto de 6 tipos de registros.

1) Registro tipo 1

Esse registro é gerado por uma declaração de formulário e só aparece uma vez para cada programa. É constituído de 4 registros de 16 bytes. Os dados fornecidos neste registro aparecem na primeira linha do formulário.

0	Indicador de tipo de registro
1 - 6	Código de identificação de programa
7 - 62	Título do formulário
63	Indicador de arquivo ativo ou arquivo inativo, pronto para ser cancelado. Essa posição é analisada pelo programa ESPREME para reorganização da biblioteca.

2) Registro tipo 2

Esse registro é gerado por uma declaração de campo. O nome do campo, nesse registro, contém até 9 posições. Se no programa fonte for dado o nome do campo com mais de 9 posições é gerado um registro tipo comentário com a parte inicial do nome do campo e, no registro tipo 2, ficam as nove posições finais do nome do campo.

0	Indicador de tipo de registro
1	Número da coluna onde se inicia o campo
2	Tamanho do nome do campo
3 - 11	Nome do campo (9 posições)
12	Característica (alfabético, numérico, ...)
13	Tamanho do campo
14	Edição (fixo, editado à direita ou esquerda)
15	Caráter de preenchimento
16	Indicador se o campo pode ser saltado

3) Registro tipo 3

Esse registro é gerado por uma declaração de crítica. É constituído de qualquer número de registros de 16 bytes, dependendo do número de rotinas de crítica chamadas e do número e tamanho dos parâmetros.

O caráter&(lógico E) faz a separação de duas críticas, e, após esse carater vem obrigatoriamente o nome de uma rotina de crítica. O tamanho de cada parâmetro é definido pela característica de cada rotina de crítica, ou, pelo tamanho em bytes, fornecido na cláusula tamanho do campo do registro tipo 2.

Todas as rotinas de crítica são externas. Os parâmetros necessários são passados pelos registradores. O nome de uma rotina de crítica não pode começar em um registro e terminar em outro. As rotinas FAIXA e TAB podem usar qualquer número de registros de crítica. Todos as outras rotinas de

crítica devem começar e terminar no mesmo registro. O valor de um limite, na rotina FAIXA, deve ser contínuo, isto é, não pode começar em um registro e terminar em outro. O elemento de uma tabela também deve ser contínuo, não podendo começar em um registro e terminar em outro. São permitidos brancos no meio do registro e, o fim de uma tabela, é indicado pelo caracter (%).

O indicador de fim de registro, aparece em cada registro e, é necessário para facilitar a atualização de ponteiros.

0	Indicador de tipo de registro
1 - 5	Nome de rotina de crítica
⋮	Parâmetros (se houver)
	Indicador de fim de crítica (carater &)
⋮	⋮
15	Indicador de fim de registro (carater |)

4) Registro tipo 4

Esse registro é gerado por uma declaração de campo inserido.

O tamanho máximo da cadeia a ser inserida é de 14 bytes. Se for necessária a inserção de mais de 14 bytes no mesmo local, o usuário deverá especificar, no programa fonte mais de um comando de inserção. Portanto, será gerado o mesmo número de registros de inserção, no programa objeto, que o número de comandos de inserção dados no programa fonte.

0	Indicador de tipo de registro
1	Tamanho da cadeia a ser inserida ($0 < t < 15$)
2 - 15	Cadeia de caracteres a ser inserida

5) Registro tipo 5

Esse registro é gerado por uma declaração de comentário. Cada registro desse tipo contém 13 posições de um comentário. Se o comentário dado no programa fonte contiver mais de 13 posições são gerados mais de um registro tipo 5 no programa objeto.

Um comentário não pode iniciar em uma linha e terminar em outra. Precisamos de uma declaração de comentário para cada linha. Se um comentário ultrapassar uma linha ele será truncado no tamanho da linha.

0	Indicador de tipo de registro
1	Número da coluna onde se inicia o comentário
2	Tamanho do comentário
3 - 15	Comentário

6) Registro tipo 6

Esse registro é gerado por uma declaração de fim de programa e, só aparece uma vez para cada programa.

0	Indicador de tipo de registro
3 - 8	Nome do formulário seguinte
9 - 15	Não utilizado

VII) PROGRAMAS

VII.1- PROGRAMA TRADUTOR

O programa TRADUTOR tem por finalidade compilar um programa fonte escrito na linguagem de criação de formulários fonte, descrita no capítulo III, e gerar um programa objeto. Este programa objeto é bem mais compacto que o programa fonte e é composto de uma série de registros, conforme descrito no capítulo VI.

Esses registros são analizados pelo programa principal e contém todas as informações a respeito de um formulário a ser projetado no vídeo, e orientam a digitação.

O objetivo de se criar uma linguagem de criação de formulários fonte foi permitir que um usuário sem conhecimento de processamento de dados pudesse escrever um programa de criação de formulários e especificar as críticas a serem aplicadas com facilidade.

Mais tarde verificou-se que, na maioria das aplicações, quem iria definir os formulários e especificar as rotinas de crítica a serem aplicadas em cada campo, em um sistema de coleta de dados, seria sempre um analista de sistemas.

A linguagem de criação de formulários objeto não é complicada, mas para um leigo em processamento de dados pode tornar-se embaraçosa. Para um analista de sistemas a linguagem de criação de formulários objeto não é complexa. Para um analista é quase indiferente escrever um programa em qualquer uma das linguagens. Se o analista escrever diretamente o programa objeto, este estará pronto para uso. Mas se for escrito um programa fonte temos mais um passo no processo que é rodar o programa TRADUTOR para compilar e gerar o programa objeto. Outro problema seria guardarmos a versão fonte e a versão objeto do programa. Qualquer alteração no formulário ou nas rotinas de crítica causaria uma alteração no programa fonte. Uma passagem pelo TRADUTOR para geração de um novo objeto.

Devido ao alto custo do programa TRADUTOR, de

vido à sua complexidade, e ao benefício pouco aparente, optou-se pela não codificação desse programa. Em contrapartida foram criadas facilidades para o analista escrever, alterar ou cancelar programas objeto.

Essas facilidades são fornecidas pelo programa EDITOR descrito abaixo.

O programa TRADUTOR fica, neste trabalho, como sugestão para futura ampliação e melhoria do sistema de coleta de dados com consistência local.

VII.2- PROGRAMA EDITOR

Uma biblioteca de formulários é composta de qualquer número de arquivos. Cada arquivo contém um programa objeto com declarações do tipo das descritas no capítulo anterior. O programa objeto contém todas as informações a respeito de um formulário a ser projetado no vídeo, tais como: atributos de cada campo, nomes ou descrições dos campos, comentários, rotinas de crítica a serem aplicadas, campos inseridos, e nome do formulário seguinte.

Os registros do programa objeto são de 16 posições. Esses registros são blocados, para economia de fita, com fator de bloco igual a 5. São gravados para cada formulário 25 blocos de 80 posições dando um total de 2000 posições. Cada arquivo é composto de 25 blocos e uma marca de fim de arquivo. O final da biblioteca de formulários é caracterizado por duas marcas de fim de arquivo seguidas.

Em uma única fita cassete podemos criar uma biblioteca de formulários com cerca de 40 programas objeto. Porém, não se aconselha colocar em uma só fita todos os formulários utilizados em uma instalação devido ao aumento no tempo médio de pesquisa por um formulário da biblioteca. Devemos criar várias bibliotecas de formulários com formatos encadeados . Por exemplo, quando se estiver efetuando entrada de dados para um sistema financeiro usar a biblioteca de formulários financeira com os seguintes formulários:

1) Cadastramento de clientes
2) Alteração de dados de clientes
3) Cadastramento de dados contrátuais
4) Alteração de dados contratuais
5) Liberações de crédito
6) Recebimentos efetivados
7) Baixa de contratos

Esses formulários são todos relacionados e podem ser usados no mesmo serviço. O tempo de pesquisa na biblioteca será bem pequeno devido ao reduzido número de formulários.

Para criar ou alterar uma biblioteca de formulários usamos o programa EDITOR. Esse programa trabalha com duas unidades de fita cassete. Na unidade número zero fica a biblioteca de formulários, e na unidade número um, uma fita cassete que é usada como arquivo auxiliar de trabalho.

São necessários os seguintes passos para porro programa EDITOR em funcionamento no terminal inteligente:

1) Ligar o terminal usando a chave existente no painel.
2) Colocar no suporte da unidade de fita cassete zero, a fita cassete com o programa EDITOR.
3) Desligar todas as oito chaves do painel.As chaves estão desligadas quando na posição inferior.
4) Apertar o botão CARGA situado no painel do terminal. O conteúdo da fita cassete é lido e o programa EDITOR carregado na memória. Em caso de erro de leitura, o processo, continua ao se apertar o botão VALIDADE no painel.
5) Neste ponto, o operador retira a fita cassete com o programa EDITOR e monta a fita cassete com a biblioteca de formulários na unidade de fita número zero, e, uma fita cassete na unidade de fita número um para ser usada como arquivo de trabalho.
6) O operador entra com um comando pelo te -

clado. São disponíveis os seguintes comandos:

a) MENU

Este comando é acionado apertando-se a letra M no teclado do terminal. A fita cassete, contendo a biblioteca de formulários, é reenrolada até o início. É lido o primeiro registro de cada arquivo. Esse primeiro registro contém a identificação e o título do formulário. Essas informações são projetadas no vídeo girando-se uma linha para cima em cada leitura. Com este comando tomamos conhecimento dos nomes e títulos de todos os programas objeto contidos na biblioteca de formulários. Após a leitura de todos os arquivos a fita cassete fica posicionada ao final da biblioteca de formulários.

b) LER

Este comando é acionado apertando-se a letra L no teclado do terminal. É lido um arquivo da biblioteca de formulários. O conteúdo desse arquivo, o programa objeto, é carregado na memória a partir do endereço 0100_{16}.

c) AVANCA

Este comando é acionado apertando-se a letra A no teclado do terminal. A fita cassete contendo a biblioteca de formulários avança até o início do próximo arquivo. Com este comando o operador pode posicionar a fita cassete no início de qualquer arquivo ou ao final da biblioteca de formulários.

d) GRAVA

Este comando é acionado apertando-se a letra G no teclado do terminal. São gravados na biblioteca de formulários 25 blocos relativos ao programa objeto presente na memória a partir do endereço 0100_{16}. A fita cassete contendo a biblioteca de formulários é posicionada no seu final e o programa objeto incluido como mais um arquivo da biblioteca. Após, são gravadas duas marcas de fim de arquivo indicando o novo final da biblioteca de formulários.

e) CANCELA

Este comando é acionado apertando-se a letra C no teclado do terminal. Para cancelar um programa objeto da biblioteca de formulários devemos primeiro posicionar a fita cassete no início de um arquivo e após dar o comando de cancelamento. O programa objeto não é removido nesta hora e sim marcado para futura remoção. Quando rodamos o programa ESPREME para reorganização da biblioteca os arquivos marcados são removidos. A descrição do programa ESPREME é vista adiante.

f) EDITA

Este comando é acionado apertando-se a letra E no teclado do terminal. É executado um desvio para uma rotina da edição. A partir desse momento só são aceitos os comandos de edição apresentados abaixo. O programa objeto presente na memória a partir do endereço 0100_{16} é editado usando-se a fita cassete montada na unidade de fita número um como arquivo auxiliar de trabalho.

Aparece na última linha do vídeo o conteúdo do primeiro registro do programa ob-

jeto e o seu endereço inicial e final na memória. As 16 posições do registro são mostradas em ASCII e em hexadecimal. O registro além de ser mostrado no vídeo é gravado no arquivo de trabalho em fita cassete.

Após apertar a letra E no teclado só são aceitos os comandos de edição discriminados abaixo disponíveis através de teclas especiais.

1) AVANCA UM
2) AVANCA NOVE
3) ALTERA
4) ERRO
5) INSERE
6) APAGA
7) FIM

1) O comando AVANCA UM provoca o aparecimento do próximo registro na última linha do vídeo, e a sua gravação no arquivo de trabalho em fita cassete. A tela gira uma linha para cima. Os registros anteriores não são apagados e sim girados uma linha para cima. Cada registro ocupa uma linha e são mostrados o endereço inicial e final do registro na memória e o conteúdo das 16 posições em hexadecimal e ASCII.

2) O comando AVANCA NOVE provoca a execução do comando AVANCA UM nove vezes . Com este comando podemos rapidamente posicionar a edição em um determinado registro.

3) O comando ALTERA permite a alteração do registro mostrado na última linha do vídeo. O operador pode digitar as 16 posições alterando assim o registro. Cada caracter digitado é mostrado no vídeo em ASCII e em hexadecimal. O cursor acompanha a digitação e indica a posição onde será co-

locado o próximo carater em ASCII. Após digitado o último carater do registro, este é regravado no arquivo de trabalho em fita cassete. O próximo registro é lido e colocado no vídeo girando-se a tela uma linha para cima. Este comando é usado também para digitação de um programa objeto novo. Apertando a tecla especial ALTERA sucessivamente podemos digitar qualquer número de registros criando assim um programa objeto novo.

4) O comando ERRO só é aceito durante a digitação de um registro. Provoca a reinicialização da digitação do registro caso o operador se engane ao digitar uma nova posição. O cursor recua ao início do registro indicando que o próximo carater a ser batido será inserido na primeira posição em ASCII. São esperadas 16 novas posições. A digitação de um registro pode ser originada pela rotina de alteração de um registro ou pela rotina de inserção de registros.

5) O comando INSERE provoca o aparecimento de um registro novo de 16 posições a ser inserido após o último registro lido. A tela gira uma linha para cima e aparece na última linha do vídeo o novo registro com todas as posições em branco. É esperada a digitação das 16 posições do registro. Neste comando, a situação é semelhante a da rotina ALTERA. Cada carater digitado é mostrado no vídeo em ASCII e em hexadecimal. O cursor acompanha a digitação e indica a posição onde será colocado o próximo carater em ASCII. Na ocorrência de um engano é permitida a utilização da tecla ERRO. Após digitado o último carater do registro aguar-

da-se um comando de edição. Se o próximo comando de edição for novamente INSERE, o registro é gravado em fita cassete, a tela gira uma linha para cima, e aparece na última linha do vídeo um novo registro com todas as posições em branco. Deste modo, podemos inserir entre dois registros da memória um número qualquer de registros. Se o próximo comando de edição não for INSERE, o registro é gravado em fita cassete, a tela gira uma linha para cima, e aparece na última linha do vídeo o próximo registro da memória.

6) O comando APAGA provoca o cancelamento do registro mostrado na última linha do vídeo. O registro é apagado no arquivo de trabalho em fita cassete. O próximo registro da memória é lido e colocado na última linha do vídeo, girando-se a tela uma linha para cima.

O endereço inicial e final, presente em cada linha, não é o endereço do registro na memória e sim o novo endereço de cada registro como resultado da edição.

7) O comando FIM indica o final da edição. Os registros da memória não mostrados no vídeo são apresentados um a um na última linha do vídeo girando-se a tela uma linha para cima. Esses registros são gravados no arquivo de trabalho em fita cassete até ser encontrado o último registro no final da área reservada para o programa objeto. Neste momento, temos em fita cassete um arquivo contendo o resultado da edição. Esse arquivo de trabalho é fechado, a fita cassete reenrolada ao seu início e o novo programa objeto carregado na me

mória a partir do endereço 0100_{16}. O controle é devolvido ao programa e a partir desse momento não são mais aceitos os comandos de edição. O operador pode então gravar o programa objeto alterado na biblioteca de formulários usando o comando GRAVA.

O operador pode cometer um engano durante a alteração de um programa objeto. Não existe nenhum comando que permita voltar um ou mais registros durante uma tarefa de edição. Essa restrição aparece por simplificação de programa. Não foram criados indicadores no programa objeto na memória e, no arquivo de trabalho em fita cassete. Esses indicadores permitiriam um retorno correto dos ponteiros do programa objeto na memória e dos ponteiros do programa objeto editado em fita cassete.

O procedimento para corrigir um erro deste tipo poderia ser o seguinte:

1) Apertar a tecla FIM para retornar ao programa saindo do estado de edição.
2) Posicionar a fita cassete com a biblioteca de formulários.
3) Ler novamente o programa objeto da biblioteca carregando-o na memória a partir do endereço 0100_{16}.
4) Apertar a letra E para voltar à rotina de edição.

No entanto, foi criada uma facilidade que permite evitar todos os passos desse procedimento. Com a utilização do botão ERRO situado no painel do terminal inteligente - não confundir com a tecla ERRO na console - podemos reinicializar a edição. A tela é apagada, a fita cassete com o arquivo auxiliar de trabalho reenrolada ao seu início e aparece na última linha do vídeo o primeiro registro do programa objeto na memória com endereço 0100_{16}. Os comandos de edição tornam-se disponíveis ao operador que poderá recomeçar assim a edição.

VII.3- PROGRAMA ESPREME

Este programa provoca a reorganização da biblioteca de formulários. Os formulários com marca de cancelamento são apagados da biblioteca. A marca de cancelamento é colocada pelo programa EDITOR através da utilização do comando CANCELA acionado ao se apertar a letra C no teclado do terminal. A fita cassete com a biblioteca de formulários deve estar posicionada no início do arquivo que contém o programa objeto a ser cancelado.

Os formulários são espremidos ao início da biblioteca de formulários. Os espaços vazios deixados por um arquivo cancelado são ocupados por outro programa objeto. O tamanho da biblioteca de formulários é reduzido. O tempo de pesquisa por um formulário da biblioteca também é reduzido.

O programa ESPREME trabalha com duas unidades de fita cassete. Na unidade número zero deve ser montada a fita cassete de entrada com biblioteca de formulários a ser reorganizada. Na unidade número um deve ser montada a fita cassete de saída com a biblioteca de formulários reorganizada.

VIII) OPERAÇÃO DO SISTEMA

VIII.1- INICIALIZAÇÃO

São necessários os seguintes passos para por o sistema de coleta de dados, com consistência local, orientado para formulários, em funcionamento no terminal inteligente, do NCE/UFRJ:

1) Ligar o terminal usando a chave existente no painel.

2) Colocar no suporte da unidade de fita cassete zero, a fita cassete com os programas do sistema.

3) Desligar todas as chaves do painel. As chaves estão desligadas quando na posição inferior.

4) Apertar a tecla CARGA situada no painel do terminal. O conteúdo da fita cassete é lido e os programas de controle da digitação são carregados na memória do terminal. Em caso de erro de leitura, o processo continua ao se apertar a tecla VALIDADE no painel.

5) Aparece no vídeo, na primeira linha, a mensagem 'NOME DO FORMULÁRIO: '. O cursor fica posicionado logo após a mensagem e indica onde será mostrado o nome do formulário a ser digitado pelo operador.

6) Neste ponto, o operador deve retirar a fita cassete com os programas do sistema e montar a fita cassete com a biblioteca de formulários.

7) O operador digita, através do teclado, os seis caracteres fornecendo a identificação do formulário. Essa identificação, na verdade, é o nome do programa objeto gravado na biblioteca de formulários. A informação digitada aparece no vídeo à frente da mensagem e o cursor acompanha cada caráter digitado.

8) Logo após a digitação do sexto carater da identificação, é iniciada uma pesquisa sequencial na bibliote-

ca de formulários em busca do formulário requisitado. Ao ser encontrado, o programa objeto é lido e colocado na memória em uma área reservada de tamanho fixo. O formulário é projetado no vídeo, o cursor se posiciona no inicio do primeiro campo, a linha de estado, na base da tela de vídeo, é inicializada, e, transfere-se o controle para o operador, ou seja, aguarda-se a digitação do primeiro carater do primeiro campo.

9) Ao chegar ao final da biblioteca, e não sendo encontrado o formulário pedido, é emitida a mensagem de erro 'FIM DE ARQUIVO'. Duas podem ser as causas para o insucesso da pesquisa:

a) o formulário realmente não se encontra neste volume, e sim, na biblioteca de formulários de outro volume de fita.

b) o operador enganou-se ao digitar o nome do formulário trocando uma letra ou mais.

10) Surgindo a mensagem 'FIM DE ARQUIVO', o processamento fica preso, aguardando-se o operador certificar-se do erro, apertando a tecla ERRO. Assim, é emitida a mensagem 'FORMULÁRIO NAO ENCONTRADO' para informar ao operador que o formulário requisitado não está presente na biblioteca pesquisada. Novamente pressiona-se a tecla ERRO. Após, aparece a mensagem 'MONTE OUTRA FITA' pois o sistema supõe que o operador enganou-se de acordo com a hipótese (a) explicitada no item 9 anterior. Agora, pode o operador optar entre montar outra fita, se realmente o volume montado na unidade de fita cassete não contem a biblioteca de formulários desejada, ou não montar outra fita. Se o operador não montar outra fita, significa que ele se enquadra na hipótese (b) explicitada no item 9 anterior, ou seja, ele enganou-se ao fornecer o nome do formulário desejado. Tomada uma das duas decisões, o operador aperta novamente a tecla ERRO.

11) Neste momento, o processo retorna ao item 5, com a finalidade do operador confirmar o nome do formulário requisitado, ou, alterar o nome incorreto digitado anteriormente, permitindo-se solução, caso a identificação do formulário tenha sido fornecido com erro.

12) A ocorrência de outra mensagem diferente, de todas as outras descritas nos itens anteriores, como por exemplo, 'BLOCO MENOR', 'BLOCO MAIOR' ou 'FIM DE FITA' mostra que a fita cassete montada não contem uma biblioteca de formulários. Provavelmente, o operador esqueceu de desmontar a fita contendo os programas do sistema, conforme pedido no item 6.

VIII.2- LINHA DE ESTADO

A última linha da tela de vídeo, a linha 11, na parte inferior, é reservada. Nesta linha não se apresenta nenhum campo para digitação. Denominamos aqui esta linha como linha de estado. As seguintes informações são mostradas:

1) mensagens de erro com até 36 posições
2) identificador do formulário ou nome do programa objeto com 6 posições.
3) característica do campo atual com 1 posição.
4) tamanho do campo atual com 2 dígitos.
5) edição do campo atual com 1 posição.
6) carater de preenchimento do campo atual com 1 posição.
7) permissão de salto do campo atual com 1 posição.
8) contador de registros gravados com 4 dígitos.
9) contador de formulários com 4 dígitos.

A linha de estado é inicializada quando carregamos na memória o programa objeto contendo todas as informações a respeito de um formulário.

A atualização da linha de estado ocorre nas seguintes situações:

1) Na passagem da digitação de um campo para outro, os dados do novo campo são projetados.

2) Após a gravação de um registro no arquivo de saída, é incrementado de uma unidade o contador associado.
3) Terminando a digitação de um formulário , ao passarmos para outro formulário, é incrementado de uma unidade o contador associado.
4) Na ocorrência de um erro, a mensagem correspondente é colocada na linha de estado e, permanece nesta linha até que o operador aperte a tecla ERRO, quando a mensagem de erro na linha de estado é apagada.

VIII.3- DIGITAÇÃO

Estando o cursor posicionado no início do primeiro campo, o operador começa a digitação do campo. Cada carater digitado aparece imediatamente no vídeo, no interiordo campo, e o cursor avança uma posição para indicar onde será inserido o próximo carater a ser digitado.

O formulário projetado no vídeo possui várias posições reservadas. O operador não tem possibilidade de escrever em qualquer posição da tela. Só lhe é permitido escrever nos espaços não reservados do interior de cada campo. Essas posições não reservadas são caracterizadas por um tracejado contínuo e delimitado por duas chaves. O cursor só se move nessas posições não reservadas. Ao terminar a digitação de um campo o cursor pula automaticamente para o início do campo seguinte. São consideradas posições reservadas os comentários,as descrições de cada campo, a linha de estado e todas as outras posições não delimitadas por chaves.

VIII.4- TECLA FIM DE CAMPO

Ao terminar a digitação de um campo deve o operador apertar a tecla especial FIM DE CAMPO. Se a informação fornecida foi correta e passou em todas as rotinas de crítica sem a ocorrência de erro, o cursor é posicionado no início do campo seguinte,aguardando-se a digitação do novo campo.

Se o campo é de tamanho fixo, a informação , só é aceita se forem digitadas todas as posições do campo.

Se o campo é com edição à esquerda, ele é editado sendo as posições não digitadas,após o campo, completadas com o carater de preenchimento relativo à esse campo. Após a edição o campo é movido para o vídeo. Assim, o tracejado presente à direita do campo, relativo às posições não digitadas , são preenchidas com o carater de preenchimento do campo.

Se o campo é com edição à direita, ele é editado sendo as posições digitadas justificadas à direita e as posições restantes, no início do campo, completadas com o caráter de preenchimento relativo a esse campo. Após a edição o campo é movido para o vídeo. Assim, a informação digitada começando na esquerda é deslocada para a direita e a parte do início do campo preenchido com o carater de preenchimento do campo. Por exemplo, para o campo VALOR(53---) se o carater de preenchimento é zero, após apertar a tecla FIM DE CAMPO, a informação no vídeo é modificada para VALOR(00053).

VIII.5- TECLA APAGA CARATER

Durante a digitação de um campo, o operador pode enganar-se ao bater uma ou mais teclas. A confirmação ou não do erro é obtida é obtida através da visualização do dado na tela vídeo. Por intermédio da tecla especial APAGA CARATER o operador tem a possibilidade de voltar atrás, carater a carater, até o início do campo ou até o carater incorreto para digitá-lo novamente, corrigindo o erro. O cursor é recuado uma posição e o carater deixado é apagado e substituido por um hífen(-). Podemos, apertando esta tecla especial várias vezes,re

cuar o cursor até o início do campo e apagar todos os caracteres digitados.

Se o cursor já estiver no início do campo torna-se nulo, ou sem efeito, o procedimento de apertar a tecla especial APAGA CARATER.

VIII.6- TECLA APAGA CAMPO

A tecla especial APAGA CAMPO possibilita ao operador voltar atrás na digitação de um campo. O cursor recua até o início do campo e todos os caracteres digitados no campo são apagados e substituídos pelo hífen(-). Esta tecla substitui o ato de apertar a tecla especial APAGA CARATER várias vezes, quando queremos apagar todos os caracteres do campo e voltar o cursor ao seu início.

Se o cursor já estiver no início do campo torna-se nulo, ou sem efeito, o procedimento de apertar a tecla especial APAGA CAMPO.

VIII.7- TECLA APAGA FORMULÁRIO

A tecla especial APAGA FORMULÁRIO possibilita ao operador voltar ao início do primeiro campo do formulário. Todos os campos já digitados são apagados do vídeo e preenchidos com o carater hífen(-). Os dados já gravados em fita cassete, no arquivo de saída relativos ao formulário em digitação, são também apagados. O contador de registros gravados é decrementado de tantas unidades quanto o número de registros apagados na fita cassete. Na verdade, os registros no arquivo de saída não são apagados, e sim, a fita cassete é recuada. Volta-se o mesmo número de registros que foram gravados para aquele formulário.

VIII.8- TECLA SALTA CAMPO

O salto de um campo pode ser opcional ou não. A permissão para um campo ser saltado é dado em uma declaração de campo.

Se o salto não for permitido, o ato de se apertar a tecla especial SALTA CAMPO provoca a emissão da mensagem de erro 'SALTO NAO PERMITIDO'.

Se o salto for permitido, o campo é completado com o carater de preenchimento fornecido. O tracejado continuo que caracteriza o interior do campo é substituído pelo conteúdo do campo. O cursor pula para o início do campo seguinte.

As rotinas de crítica associadas ao campo não são executadas e, portanto ignoradas, quando um campo é saltado.

Um campo só pode ser saltado se o cursor estiver posicionado na sua primeira posição. Quando o cursor não estiver no início do campo, o procedimento de apertar a tecla SALTA CAMPO é sem efeito e, normalmente, provoca um erro associado com a mensagem 'CARATER INVALIDO'.

Tendo sido iniciada a digitação para saltar o campo é necessário primeiro apagar o campo e voltar o cursor ao seu início. Isto se consegue com a utilização da tecla APAGA CAMPO ou da tecla APAGA CARATER utilizada várias vezes.

VIII.9- TECLA FIM DE FORMULÁRIO

Ao terminar a digitação de um formulário, ou seja, após apertar a tecla FIM DE CAMPO para o último campo do formulário, o processamento fica em suspenso, aguardando-se o operador apertar uma tecla especial que irá comandar a sequenciação dos formulários.

A tecla especial FIM DE FORMULARIO provoca a retirada do formulário totalmente preenchido da tela do vídeo,

e sua substituição por um formulário igual, com todos os cam - pos não preenchidos. O cursor se posiciona no início do primeiro campo e a continuação da digitação é imediata. O contador de formulários, da linha de estado, é incrementado de uma unidade.

A passagem de um formulário para outro, através da utilização da tecla FIM DE FORMULARIO, é muito rápida porque o programa objeto está na memória com toda a descrição do formulário e das rotinas de crítica associadas.

VIII.10- TECLA FORMULÁRIO SEGUINTE

Ao terminar a digitação de um formulário, ou seja, após apertar a tecla FIM DE CAMPO para o último campo de formulário, o processamento fica em suspenso, aguardando-se o operador apertar uma tecla especial que irá comandar a soquenciação dos formulários.

A tecla especial FORMULARIO SEGUINTE provoca a retirada do formulário totalmente preenchido da tela do vídeo e inicia uma pesquisa sequencial na biblioteca de formulários, em fita cassete, em busca do formulário seguinte especificado por programa.

Para o operador tomar ciência da pesquisa a ser efetuada, aparece no vídeo, na primeira linha a mensagem PESQUISANDO FORMULARIO: XXXXXX, sendo XXXXXX o nome do formulário, ou do programa objeto, fornecido como seguinte ao formulário atual.

Se a pesquisa atingir o final da biblioteca e não sendo encontrado o formulário pedido, é emitida a mensagem de erro 'FIM DE ARQUIVO', na linha de estado. O processamento fica preso, aguardando-se o operador certificar-se do erro, apertando a tecla ERRO. Assim, é emitida a mensagem 'FORMULARIO NAO ENCONTRADO' para informar ao operador que o formulário requisitado não está presente na biblioteca pesquisada. Novamente pressiona-se a tecla ERRO. Após, aparece a mensagem

'MONTE OUTRA FITA'. O operador deve então montar outro volume de fita cassete com a biblioteca de formulários apropriada.

Se o operador desejar mudar a sequenciação dos formulários ele tem a opção de não montar outra fita cassete e alterar o nome do formulário a ser pesquisado.

Tomada uma das duas decisões, o operador aperta novamente a tecla ERRO. Neste momento, aparece no vídeo, na primeira linha, a mensagem 'NOME DO FORMULARIO: XXXXXX' onde XXXXXX é o nome do último formulário pesquisado. O cursor se posiciona na primeira posição do nome do formulário. O operador digita novamente os seis caracteres da identificação do formulário, confirmando o nome anterior ou alterando-o para um formulário qualquer diferente do último solicitado.

Logo após a digitação do sexto carater da identificação, é iniciada outra pesquisa sequencial na biblioteca de formulários.

Se o formulário for encontrado na biblioteca ele é lido e carregado em uma área fixa da memória. O formulário anterior é destruido, pois só é possível estar presente, na memória um formulário de cada vez. Na verdade, não é o formulário que é carregado na memória, e sim, o programa objeto contendo a descrição de todos os campos do formulário e de todas as rotinas de crítica usadas. A biblioteca de formulários, é uma biblioteca de programas objeto. Neste trabalho, muitas vezes os termos programa objeto e formulário são usados como sinônimos.

O novo formulário é projetado no vídeo com todos os campos não preenchidos. O cursor se posiciona no início do primeiro campo. O contador de formulários, da linha de estado, é incrementado de uma unidade. O controle passa ao operador, ou seja, aguarda-se a digitação do primeiro carater do primeiro campo.

VIII.11- TECLA FORMULARIO OPCIONAL

Ao terminar a digitação de um formulário,ou seja, após apertar a tecla FIM DE CAMPO para o último campo do formulário, o processamento fica em suspenso, aguardando-se o operador apertar uma tecla especial que irá comandar a sequenciação dos formulários.

A tecla especial FORMULARIO OPCIONAL provoca a retirada do formulário totalmente preenchido da tela do vídeo e o aparecimento, na primeira linha, da mensagem 'NOME DO FORMULARIO: '. Neste caso, a sequenciação dos formulários é qualquer, à escolha do usuário. O operador pode fornecer qualquer nome de formulário presente em qualquer biblioteca.

A situação é semelhante à inicialização do sistema, quando o operador fornece a identificação do primeiro formulário a ser digitado. Para obtermos a descrição do funcionamento após apertarmos a tecla FORMULARIO OPCIONAL, devemos retornar ao item 5 do capítulo VIII.1 - INICIALIZAÇÃO.

O contador de formulários, na linha de estado, é incrementado de uma unidade.

Até aqui, vimos que as três teclas especiais usadas para sequenciação de formulários são sempre usadas ao término da digitação de um formulário. No entanto, a tecla especial FORMULARIO OPCIONAL pode ser usada em outra ocasião.

Ao visualizar um novo formulário projetado no vídeo, o operador pode descobrir que escolheu um formulário errado. Neste caso, ele irá desejar mudar de formulário antes de digitá-lo. Isto é possível, com a utilização da tecla especial FORMULARIO OPCIONAL, antes de ser digitado qualquer caráter para o formulário. Se a digitação já tinha sido iniciada,o operador deve apertar a tecla APAGA FORMULARIO, e após, a tecla FORMULARIO OPCIONAL.

O formulário não digitado é retirado do vídeo, e aparece, na primeira linha, a mensagem 'NOME DO FORMULARIO: ' permitindo-se ao operador retificar a sua escolha.

VIII.12- TECLA FIM DE TRABALHO

Ao terminar a digitação de um formulário, ou seja, após apertar a tecla FIM DE CAMPO para o último campo do formulário, o processamento fica em suspenso, aguardando -se o operador apertar uma tecla especial que irá comandar a sequenciação dos formulários. No entanto, ao invés de pedir outro formulário o operador pode terminar o seu serviço nesta hora.

Para encerrar um serviço apertamos a tecla especial FIM DE TRABALHO. O ato de se apertar esta tecla provoca a gravação de duas marcas de fim de arquivo, caracteri - zando assim, o final do arquivo de dados.

VIII,13- FORMULARIO NA MEMÓRIA

Existe uma área de tamanho fixo na memória, com endereço inicial igual a 0100_{16}, para carregarmos um programa objeto lido da biblioteca de formulários. Só podemos ter, de cada vez, um programa objeto na memória.

O programa objeto carregado na memória contém a descrição de todos os campos do formulário, de todas as rotinas de críticas associadas a cada campo, dos comentários a serem colocados no vídeo e, eventualmente, informações sobre campos inseridos.

O tamanho de um formulário é qualquer. A limitação é dada pelo tamanho da área fixa de memória reservada para carregarmos o formulário. Em uma configuração com 8 k bytes de memória, temos 6 k bytes utilizados pelo sistema e 2 k bytes reservados para a carga de programas objeto. Em uma configuração de 16 k bytes, temos 6 k bytes utilizados pelo sistema e 10 k bytes disponíveis para carga de programas objeto, memória suficiente para armazenarmos formulários muito grandes.

O vídeo possui 12 linhas, sendo uma linha

reservada para a linha de estado. Sobram então 11 linhas para projetarmos formulários. Evidentemente, um formulário poderá facilmente estourar as 11 linhas disponíveis. Quando isto acontece o sistema automaticamente faz a paginação do formulário.

O procedimento adotado é o seguinte:

a) Primeiro projetam-se as 11 primeiras linhas do formulário.

b) Transfere-se o controle ao operador, aguardando-se o início da digitação. O operador deve preencher todos os campos.

c) Após ser apertada a tecla FIM DE CAMPO para o último campo, da última linha, as 11 linhas do formulário, agora totalmente preenchidas, são retiradas do vídeo.

d) Projetam-se então as 11 linhas seguintes do formulário. Repetem-se os passos descritos no itens (b) e (c).

e) Quando for projetada a última página do formulário, o controle é transferido ao operador e este deve digitar todos os campos do final do formulário. Obviamente, a última página pode ter menos de 11 linhas.

f) Após ser apertada a tecla FIM DE CAMPO, para o último campo, da última página, o sistema considera o formulário totalmente preenchido e aguarda a informação sobre o próximo formulário a ser projetado.

Verificamos que com o uso desse procedimento um formulário pode ter qualquer número de páginas. A limitação do tamanho de um formulário não é dado pelo tamanho do vídeo, e sim, pelo tamanho da área de memória reservada.

VIII.14- GRAVAÇÃO DOS DADOS EM FITA CASSETE

1) Formato dos dados

Os dados em fita cassete são gravados, com

registros, de tamanho fixo, de 80 posições. Um formulário pode gerar no mínimo um registro de 80 posições, e no máximo, qualquer número de registros de 80 posições, dependendo do tamanho do formulário. Quanto maior o formulário maior o número de registros de 80 posições gerados. Durante o preenchimento de um formulário o sistema testa se foram completadas 80 posições digitadas. Esse teste não feito após a digitação de cada carater, e sim, após completada a digitação de um campo. Deste modo, o operador não é interrompido no meio do preenchimento de um campo para ser efetuada a gravação de um registro. A interrupção, para gravação de um registro no arquivo de saída, ocorre após o término da digitação de um campo.

Digitado o último campo do formulário, se não estiver formado um registro de 80 posições, as posições não preenchidas do registro são completadas com espaços e então efetuada a gravação em cassete. Esse último registro relativo a um formulário é gravado seguido de uma marca de fim de arquivo cuja utilidade será vista adiante.

A gravação dos dados com registros de 80 posições não constitui uma desvantagem e nem é uma restrição. Considera-se fácil, e quase sempre necessário, rodar um programa de formatação do arquivo movimento, no computador central, antes de proceder à atualização.

2) Montagem da fita cassete

Os dados digitados são gravados na fita cassete montada na unidade número um. Ao iniciar o preenchimento dos campos do primeiro formulário, o operador deve montar a fita cassete antes de se completarem 80 posições digitadas. Se o operador esquecer de montar a fita para gravação do arquivo de saída, após completar um registro, o processamento é interrompido e emitida a mensagem de erro 'FITA NAO ESTA MONTADA'. O operador providencia a montagem da fita e aperta a tecla ERRO. O primeiro registro é gravado, a mensagem de erro apagada, e o processamento liberado para digitação.

Se a fita cassete não contiver nenhum dado gravado ela deve ser posicionada no seu início físico. Isto se consegue utilizando o botão REW, da própria unidade de fi-

ta, para reenrolar a fita ao seu início físico, ou seja, até a marca refletiva. Ao executar o primeiro comando de gravação, a rotina faz a fita avançar ao seu início lógico e grava o primeiro registro.

Se a fita cassete já contiver dados gravados ela poderá ser montada, sem ser reenrolada, se estiver posicionada no final de um arquivo. Existe uma rotina especial, acoplada ao programa principal, que pode ser utilizada para projetarmos no vídeo o conteúdo da fita cassete de dados e para posicionarmos a fita ao final do arquivo de dados, ou ao final de um formulário, em qualquer posição do arquivo. Esta rotina é descrita abaixo.

3) Recuperação dos dados

Após o último registro para um formulário, no arquivo de dados, o sistema grava uma marca de fim de arquivo. O que caracteriza o final do arquivo de dados são duas marcas de fim de arquivo seguidas. Gravamos essas duas marcas utilizando a tecla especial FIM DE TRABALHO. Assim, temos sempre uma marca de fim de arquivo separando dois formulários. Graças a esse procedimento, podemos recuperar informações em caso de ocorrência de falhas no meio de um serviço, como por exemplo, na falta de energia elétrica.

Através do botão ERRO, situado no painel do terminal inteligente temos acesso a uma rotina especial. Não confundir com a tecla ERRO situada no teclado do terminal. Esta rotina especial executa os seguintes passos:

a) reenrola a fita cassete, montada na unidade número um, contendo um arquivo de dados, ao seu início físico.

b) le todos os registros de 80 posições até encontrar uma marca de fim de arquivo, ou seja, le todos os registros relativos a um formulário.

c) projeta os registros lidos no vídeo, girando a tela para cima para cada registro lido e colocando na última linha o último registro lido.

d) aguarda um comando.

Os comandos disponíveis são:

1) avança um formulário através da tecla especial AVANÇA UM. Consiste em repetir os passos (b), (c) e (d).

2) avança até o final do arquivo de saída através da tecla especial AVANCA TUDO. Consiste em repetir os passos (b) e (c) até serem encontradas duas marcas de fim de arquivo, isto é, até o final do arquivo de dados. Terminando a leitura dos registros de todos os formulários a fita cassete fica posicionada no fim do arquivo de saída permitindo gravar dados a partir desse ponto ampliando-se este arquivo.

3) volta ao programa principal através da tecla especial FORMULARIO OPCIONAL. O ato de se apertar esta tecla provoca o aparecimento no vídeo da mensagem 'NOME DO FORMULARIO: ', situação semelhante à inicialização do sistema.

Utilizando a tecla AVANCA UM ou a tecla AVANCA TUDO, posicionamos a fita cassete contendo o arquivo de dados, ao final de um formulário ou ao final do arquivo de dados. Se for encontrado o final do arquivo de dados ao usarmos uma das duas teclas acima é executado um desvio automático para o início do programa com o aparecimento da mensagem ' NOME DO FORMULÁRIO: ', na primeira linha. Desta forma, um serviço pode ser interrompido a qualquer momento, seja por falta de energia elétrica ou por própria vontade do operador. Ao recomeçar o serviço monta-se a fita cassete com o arquivo de dados parcialmente gravado na unidade de fita número um e, através do botão ERRO do painel do terminal, desvia-se para a rotina especial de recuperação. A rotina mostra no vídeo os dados já coletados e permite ao operador posicionar a fita após o último formulário gravado, ou, em qualquer posição desejada.

Se a causa da interrupção do processamento foi falta de energia elétrica, os dados digitados estarão salvos até o último formulário gravado.

A principal finalidade desta rotina especial é reposicionar uma fita cassete contendo um arquivo de dados. Com este procedimento temos condições de recuperar infor

mações já gravadas em caso de falhas não previstas. A outra finalidade é a visualização dos dados após a sua gravação em cassete.

IX) LÓGICA DOS PROGRAMAS

IX.1- PROGRAMA PRINCIPAL

Apresento abaixo o algorítmo lógico da implementação do programa principal do sistema de coleta de dados, com consistência local, para o terminal inteligente do NCE/UFRJ.

P1 . Inicialização das variáveis internas do programa.

P2 . Inicialização da linha de estado.

P3 . Coloca na primeira linha do vídeo mensagem "NOME DO FORMULÁRIO: "

P4 . Aguarda o operador digitar os seis caracteres do nome do formulário.

P5 . Fornecido o nome do formulário é iniciada uma pesquisa na biblioteca de formulários em busca do formulário requisitado.

P6 . Encontrado o programa objeto na biblioteca de formulários, este é lido e colocado na memória em uma área reservada.

P7 . É iniciada uma leitura sequencial de todos os registros do programa objeto na memória. O primeiro registro é uma declaração de formulário contendo o nome e título do formulário. O nome do formulário, isto é, a identificação do programa objeto, é colocado na linha de estado e o título do formulário colocado na primeira linha do vídeo.

P8 . Le um registro do programa objeto na memória.

P9 . Se for declaração de campo chama a rotina para colocar um campo do formulário no vídeo. Atualiza ponteiro do programa objeto e volta a P8.

P10. Se for declaração de comentário chama a rotina para colocar um comentário no vídeo. Atualiza o ponteiro do programa objeto e volta a P8.

P11. Se for declaração de fim de programa segue adiante, senão

volta a P8.

P12. Incrementa o contador de formulários de uma unidade.

P13. Restaura o ponteiro do programa objeto.

P14. Le um registro do programa objeto.

P15. Se for declaração de campo via para P19.

P16. Se for declaração de crítica vai para P31.

P17. Se for declaração de campo inserido vai para P35.

P18. Se for declaração de fim de programa objeto vai para P38.

P19. (Rotina de Controle da Digitação)
Coloca na linha de estado os atributos do campo.

P20. Posiciona o cursor na primeira posição do campo.

P21. Se a cadeia de gravação está cheia chama a rotina de gravação em cassete.

P22. Le um carater do teclado.

P23. Se for SALTA CAMPO vai para P44.

P24. Se for FIM DE CAMPO vai para P49.

P25. Se for APAGA CAMPO vai para P59.

P26. Se for APAGA FORMULARIO vai para P63.

P27. Se for RETORNA CARATER chama a rotina para apagar um carater no vídeo e na cadeia de gravação. Recua o cursor uma posição e volta a P22.

P28. Se já tinham sido digitados todos os caracteres do campo emite a mensagem de erro "ESTOURO DE TAMANHO" e volta a P22

P29. Se o carater digitado não atende à característica do campo emite a mensagem de erro "CARATER INVALIDO" e volta a P22.

P30. Move o carater para o vídeo e para a cadeia de gravação. Avança o cursor uma posição, vai para P22.

P31. (Rotina de Crítica)
Neste ponto, a crítica é identificada sendo chamada a rotina especificada passando-se os parametros necessários através dos registradores. Todas as rotinas de crítica são externas.

P32. Se o conteúdo do acumulador for igual a zero, então tudo bem, volta a P14.

P33. Houve erro e o acumulador contem o número da mensagem de erro. Chama a rotina para localizar a mensagem de erro e colocá-la na linha de estado.

P34. Recua o ponteiro do programa objeto e desvia para a rotina APAGA CAMPO.

P35. (Rotina de Inserção de Campo)
Se a cadeia de gravação está cheia chama a rotina de gravação em cassete.

P36. Move o campo a ser inserido para a cadeia de gravação.

P37. Vai para P14.

P38. (Rotina de Termino de Formulário)
Preenche o final da cadeia de gravação com brancos e grava em cassete o último registro de 80 bytes do formulário.

P39 Le um carater do teclado.

P40 Se for FIM DE FORMULARIO retira o formulario preenchido e coloca no vídeo o mesmo formulário com todos os campos em branco, e, volta a P12.

P41. Se for FORMULARIO SEGUINTE pesquisa na biblioteca de formulários usando o nome do formulário fornecido por programa, e, volta a P6.

P42. Se for FORMULARIO OPCIONAL vá para P3.

P43. Se for FIM DE TRABALHO grava duas marcas de fim de arquivo na fita cassete de dados e encerra o processamento. Senão vai para P39.

P44. (Rotina Salta Campo)
Testa se o campo tem salto permitido. Senão emite mensagem de erro "SALTO NÃO PERMITIDO" na linha de estado e volta a P22.

P45. Preenche a cadeia de gravação com o carater de preenchimento fornecido no programa objeto usando o tamanho do cam -

po. Atualiza o ponteiro da cadeia de gravação.

P46. Le um registro do programa objeto.

P47. Se for declaração de crítica vai para P46.

P48. Vai para P15.

P49. (Rotina de Fim de Campo)
Se o campo for editado à esquerda vai para P53.

P50. Se o campo for editado à direita vái para P55.

P51. O campo é de tamanho fixo. Testo se o número de caracteres do campo é igual ao número de caracteres batidos. Senão emite mensagem de erro "TAMANHO INSUFICIENTE" na linha de estudo e, volta a P22.

P52. Vai para P14.

P53. O campo é editado à esquerda. Completa o final do campo com o carater de preenchimento fornecido no programa objeto.

P54. Vai para P56.

P55. O campo é editado à direita. Justifica os caracteres batidos à direita do campo na cadeia de gravação e completa o início do campo com o carater de preenchimento fornecido no programa objeto.

P56. Atualiza o ponteiro da cadeia de gravação.

P57. Move o campo editado para o vídeo.

P58. Vai para P14.

P59. (Rotina Apaga Campo)
Restaura o ponteiro da cadeia de gravação.

P60. Recua o cursor ao início do campo.

P61. Apaga no vídeo os caracteres digitados.

P62. Vai para P22.

P63. (Rotina Apaga Formulário)
Apaga no vídeo todos os campos do formulário em digitação.

P64. Verifica quantos blocos foram gravados em cassete relati -

vo a esse formulário.

P65. Recua a fita cassete com o arquivo de dados apagando os blocos gravados para o formulário. Decrementa o contador de blocos gravados.

P66. Vai para P13.

P67. (Rotina de gravação em cassete)
Grava um bloco no cassete de dados.

P68. Move os dados da cadeia de gravação não contidos no bloco gravado para o início da cadeia.

P69. Atualiza o ponteiro da cadeia de gravação.

P70. Incrementa o contador de registros gravados de uma unidade e retorna.

P71. (Rotina de Recuperação de Dados)

P72. Reenrola a fita cassete contendo o arquivo de dados ao seu início físico.

P73. Le um registro de 80 posições da fita cassete.

P74. Gira a tela do vídeo uma linha para cima e coloca o registro na última linha do vídeo.

P75. Se encontra uma marca de fim de arquivo segue adiante, se não volta a P73.

P76. Le um comando do teclado.

P77. Se for AVANCA UM volta a P73.

P78. Se for AVANCA TUDO repete os passos P73 e P74 até encontrar duas marcas de fim de arquivo seguidas, ou seja, até o final do arquivo de dados. Vai para P3.

P79. Se for FORMULÁRIO OPCIONAL vai para P3.

P80. Vai para P76.

IX.2- PROGRAMA EDITOR

Apresento abaixo o algoritmo lógico da imple - mentação do programa editor do sistema de coleta de dados, com consistência local, para o terminal inteligente do NCE/UFRJ. Este programa trabalha com duas fitas cassete. Na unidade de fita número zero temos a biblioteca de formulários, e na unidade número um, uma fita cassete que é usada como arquivo auxiliar de trabalho.

P1 . Leia um carater do teclado.

P2 . Se for a letra M vá para P9.

P3 . Se for a letra L então leia 25 blocos da fita cassete que contém a biblioteca de formulários e carregue na memória a partir do endereço 0100_{16}. Volte a P1.

P4 . Se for a letra A então avance a fita cassete que contém a biblioteca de formulários até o início do próximo arquivo. Volte a P1.

P5 . Se for a letra G vá para P14.

P6 . Se for a letra C vá para P18.

P7 . Se for a letra E vá para P22 (rotina EDITOR).

P8 . Vá para P1.

P9 . (Rotina MENU)
Reenrole a fita cassete que contém a biblioteca de formulários até o seu início físico.

P10. Leia um registro da fita cassete. É o primeiro registro do arquivo.

P11. Coloque no vídeo a identificação e o título do formulário girando a tela uma linha para cima.

P12. Avance até o início do próximo arquivo.

P13. Se forem encontradas duas marcas de fim de arquivo seguidas vá para P1, senão volte a P10.

P14. (Rotina GRAVA)

Avance a fita cassete que contém a biblioteca de formulários até o seu final.

P15. Grave 25 blocos na fita cassete, relativos ao programa objeto presente na memória a partir do endereço 0100_{16}.

P16. Grave duas marcas de fim de arquivo indicando o novo final da biblioteca de formulários.

P17. Vá para P1.

P18. (Rotina CANCELA)
Leia o primeiro bloco do arquivo, na fita cassete que contém a biblioteca de formulários.

P19. Volte a fita cassete um bloco.

P20. Reescreva o bloco anterior, agora com a marca de cancelamento.

P21. Vá para P1.

P22. (Rotina EDITOR)
Reenrole a fita cassete que contém o arquivo auxiliar de trabalho ao seu início físico.

P23. Apague a tela do vídeo. Inicialize o ponteiro do programa objeto.

P24. Chame a rotina P32.

P25. Aguarde um comando.

P26. Se for AVANCA UM vá para P24.

P27. Se for AVANCA NOVE chame a rotina P32 nove vezes e volte a P25.

P28. Se for ALTERA vá para P37.

P29. Se for INSERE vá para P41.

P30. Se for APAGA vá para P46.

P31. Se for FIM vá para P49.

P32. (Rotina para processar um registro)
Gire a tela uma linha para cima.

P33. Coloque na última linha do vídeo o conteúdo do registro do programa objeto e o seu endereço inicial e final na memória. As 16 posições do registro são mostradas em ASCII e hexadecimal.

P34. Grave o registro no arquivo de trabalho em fita cassete.

P35. Atualize o ponteiro do programa objeto.

P36. Retorne.

P37. (Rotina para alterar um registro)
Volte um registro no arquivo auxiliar de trabalho, na fita cassete número um.

P38. Chame a rotina P54.

P39. Grave o registro alterado no arquivo auxiliar de trabalho.

P40. Vá para P24.

P41. (Rotina para inserir um registro)
Gire a tela uma linha para cima.

P42. Escreva na última linha do vídeo um registro novo com todas as 16 posições em branco.

P43. Chame a rotina P54.

P44. Grave o registro digitado no arquivo auxiliar de trabalho

P45. Vá para P24.

P46. (Rotina para apagar um registro)
Volte um registro no arquivo auxiliar de trabalho na fita cassete número um.

P47. Decremente de 16 o endereço do próximo registro.

P48. Vá para P24.

P49. (Rotina Fim da edição)
Chame a rotina P32.

P50. Se chegou no fim da área reservada ao programa objeto na memória siga adiante, senão volte a P49.

P51. Feche o arquivo auxiliar de trabalho na fita cassete número um. Reenrole esta fita ao seu início físico.

P52. Carregue na memória, a partir do endereço 0100_{16}, o programa objeto editado, gravado na fita cassete número um.

P53. Vá para P1.

P54. (Rotina de digitação de um registro)
Posicione o cursor na última linha do vídeo, na primeira posição em ASCII. Inicialize o contador de caracteres.

P55. Leia um carater do teclado.

P56. Se for a tecla especial ERRO vá para P54.

P57. Se não for a tecla especial APAGA CARATER vá para P59.

P58. Leia o conteúdo das chaves. As chaves devem estar posicionadas antes de apertarmos a tecla APAGA CARATER.

P59. Converta o carater para Hexadecimal.

P60. Mova para a última linha do vídeo, o carater em ASCII e em hexadecimal.

P61. Avance o cursor uma posição.

P62. Se foram fornecidos 16 caracteres então retorne, senão vá para P55.

IX.3- PROGRAMA ESPREME

Apresento abaixo o algorítmo lógico da implementação do programa ESPREME do sistema de coleta de dados, com consistência local, para o terminal inteligente do NCE/UFRJ.

Este programa trabalha com duas unidades de fita cassete. Na unidade número zero deve ser montada a fita cassete de entrada com a biblioteca de formulários a ser reorganizada. Na unidade número um deve ser montada a fita cassete de saída com a biblioteca de formulários reorganizada.

P1 . Leia o primeiro bloco da biblioteca de formulários de origem na fita cassete zero.

P2 . Se foram encontradas duas marcas de fim de arquivo seguidas vá para P8.

P3 . Se tem marca de cancelamento avance até o início do próximo arquivo e volte a P1.

P4 . Grave um bloco na biblioteca de formulários de destino, na fita cassete um.

P5 . Se foram lidos e gravados 25 blocos, grave uma marca de fim de arquivo na fita cassete um, e volte a P1.

P6 . Leia um bloco do arquivo biblioteca de formulários de origem, na fita cassete zero.

P7 . Volte a P4.

P8 . Grave duas marcas de fim de arquivo na biblioteca de formulários de destino, na fita cassete um.

P9 . Fim do processamento.

X) CONCLUSÕES

X.1- EXTENSÕES

Apresento adiante algumas sugestões para ampliação e melhoria do sistema de coleta de dados, com consistência local para o terminal inteligente do NCE/UFRJ.

1) Mostrar no vídeo contadores e acumuladores

As rotinas de crítica que afetam o contéudo dos contadores ou acumuladores atuam internamente, sem intervenção externa. O operador não toma conhecimento das alterações sofridas pelos contadores ou acumuladores. No entanto, o usuário poderia utilizar os valores armazenados para o seu próprio controle. Por exemplo, no caso de um fechamento de lote, o total de controle calculado pode não bater com o total de controle fornecido. O usuário gostaria de saber o valor calculado internamente. Para atender a essa reinvindicação poderiam ser criadas duas teclas especiais aqui denominadas de CTD e ACC com as seguintes funções:

a) Tecla CTD

Ao ser apertada a tecla CTD será executado um desvio para uma rotina que escreverá na linha de estado, nas primeiras 36 posições, o conteúdo dos quatro contadores, no seguinte formato:

CT1= 999ßßCT2=999ßßCT3=999ßßCT4=999ßß

onde CTn representa o contador número n e ß representa um espaço em branco.

O processamento ficará em suspenso, aguardando o operador apertar a tecla VALIDADE no painel do terminal. Ao pressionar esta tecla o conteúdo dos contadores, na linha de estado, será apagado e efetuado um retorno ao ponto de chamada da rotina.

b) Tecla ACC

Ao ser apertada a tecla ACC será executado um

desvio para uma rotina que escreverá na linha de estado, nas primeiras 36 posições, o conteúdo dos dois acumuladores, no seguinte formato:

123456789012345ßßß123456789012345ßßß

onde ß representa um espaço em branco.

Os primeiros 15 dígitos representam o contéúdo do acumulador 1 e os 15 dígitos seguintes representam o conteúdo do acumulador 2.

O processamento ficará em suspenso, aguardando o operador apertar a tecla VALIDADE no painel do terminal. Ao pressionar esta tecla o conteúdo dos acumuladores, na linha de estado, será apagado e efetuado um retorno ao ponto de chamada da rotina.

2) Mensagem de erro modificada pelo usuário

As rotinas de crítica que agem sobre os contadores e acumuladores produzem sempre a mensagem de erro "REDIGITE". O aparecimento desta mensagem significa que foi detetado um erro, cuja causa pode ter sido a digitação incorreta de um campo anterior, ou o campo atual não cruza com um ou mais campos anteriores. A indicação dos campos que sofrem cruzamento normalmente aparece no manual de digitação do formulário.

Poderia ser criada uma facilidade para permitir ao usuário indicar no vídeo junto da mensagem de erro, o número do campo cujo conteúdo é cruzado com o campo atual em digitação.

Para atender a esse requisito seriam necessárias duas rotinas de ação aqui denominadas NUMER e LIMPA, com as seguintes funções:

a) NUMER (número)

Descrição: Esta rotina converte o número especificado como parâmetro de binário para ASCII e o coloca após a mensagem de erro "REDIGITE".

b) LIMPA

Descrição: Esta rotina apaga a informação colocada após a mensagem de erro"REDIGITE" movendo espaços em branco.

Outra providencia necessária seria reservar quatro espaços em branco após a mensagem de erro "REDIGITE". Um espaço para separação da palavra do número a ser inserido e três espaços para colocação do próprio número.

Desta forma, o usuário poderia controlar por programação o conteúdo da mensagem de erro associada às rotinas de crítica de verificação que atuam sobre os contadores e acumuladores.

3) Inclusão de uma impressora

Outra importante ferramenta para extensão do sistema de coleta de dados, com consistência local para o terminal inteligente é a inclusão de uma impressora ligada diretamente à unidade central de processamento do terminal.

Após completada a digitação de um formulário,poderia ser impresso, em formulário contínuo, as informações recem-colhidas. Por exemplo, se o terminal inteligente estiver trabalhando em coleta de dados de matrícula de alunos, é interessante cada aluno receber o seu comprovante de matrícula. Assim, seriam coletados todos os dados referentes à matrícula de cada aluno para enviar ao centro de processamento de dados. O único trabalho do computador central seria proceder à atualização dos arquivos próprios, já que, a tarefa de imprimir comprovantes foi transferida para o terminal inteligente. Logo,o computador central estaria livre para efetuar trabalhos mais elegantes e sofisticados.

Para atender a esse requisito podemos fazer as seguintes alterações:

a) Incluir na declaração de fim de programa uma posição para informar se o conteúdo daquele formulário deve ser impresso ou não.

b) Se as informações do formulário devem ser

impressas, o formato de impressão, seria dado através de declarações de impressão colocadas após a declaração de fim de programa conforme abaixo:

```
IP,/início/,/tamanho/,/linha/,/coluna/;
```

/início/::= posição inicial no registro de dados gravados em fita cassete.

/tamanho/::= número de posições a serem obtidos do registro, para impressão, a partir da posição inicial.

/linha/::= número da linha de impressão.

/coluna/::= número da coluna.

A finalidade das declarações de impressão é permitir ao usuário imprimir, em formulário contínuo, em formato diferente do apresentado no formulário projetado no vídeo do terminal. Além disso, seria opcional omitir determinados campos.

c) Codificar uma rotina para atuar após a digitação de um formulário. Esta rotina testaria a posição de ordem de impressão na declaração de fim de programa. Existindo ordem de impressão a rotina, usando o arquivo de dados, comandaria a impressão, consultando as declarações de impressão.

4) Novas rotinas de crítica

Todas as rotinas de crítica utilizadas no sistema de coleta de dados, com consistência local, são externas ao programa principal. Os parâmetros são passados às rotinas através dos registradores, na forma de números, caracteres ou endereços. O retorno é sempre efetuado pelo registrador A. Se o valor cumtido em A for zero o retorno foi bem sucedido. Se o valor contido em A for diferente de zero houve um erro e no registrador A temos o número do erro.

Novas rotinas de crítica podem ser adicionadas, ao sistema com bastante facilidade. Os passos necessários para

se incluir uma nova rotina de crítica são os seguintes:

a) Codificar a rotina de crítica sem preocupação alguma com o programa principal. Considerar que os parametros são passados através dos registradores e definir quais são os registradores usados. Efetuar o retorno através do acumulador, ou seja, o registrador A. Se a operação for bem sucedida, colocar o valor zero no acumulador. Se houve erro colocar o número do erro no acumulador.

b) Se a rotina de crítica for utilizar uma mensagem de erro já existente no sistema, verificar o número do erro associado a esta mensagem, no programa principal e, ignore o passo (c) adiante.

c) Se a rotina de crítica for utilizar uma mensagem de erro nova, esta mensagem deve ser incluida no programa principal. A inclusão da nova mensagem de erro implica na alteração de um procedimento de nome LOCMEN. Este procedimento, fornecido o número de um erro no acumulador, localiza o endereço da mensagem de erro associada, coloca esse endereço no registrador duplo BC, e o tamanho da mensagem no acumulador retornando em seguida.

A alteração na rotina LOCMEN é elementar para qualquer programador que conheça a linguagem ASSEMBLER do Intel 8008.

d) Incluir no programa principal, no módulo de controle da crítica, as seguintes instruções:

1) teste para reconhecer o nome da nova rotina de crítica.
2) mover para os registradores os parametros necessários, se existirem.
3) chamar a rotina de crítica
4) colocar uma instrução de desvio para a rotina de reconhecimento de erro.

X.2- ANÁLISE CRÍTICA DOS RESULTADOS

A aferição dos resultados através da comparação com outros equipamentos similares é difícil execução. Não existem no mercado equipamentos de entrada de dados que possam ser confrontados com o terminal inteligente. Todos possuem características diversas que tornam difícil a comparação entre duas configurações. Existe uma grande carência de equipamentos, fabricados no Brasil, para aquisição de dados e, um grande mercado de consumidores. Esses consumidores, e o próprio país, pagam os pesados encargos da importação. Desta forma, só o fato de se apresentar um equipamento brasileiro já é um grande passo à frente.

A unidade que mais se aproxima do terminal inteligente, para efetuar coleta de informações, é o DE 523, muito difundido no Brasil. Este equipamento, até bem pouco tempo, era comercializado pela Olivetti, e possui tecnologia da empresa norte-americana Sycor. Ele não é adequado para projeção de formulários no vídeo pois possui uma tela de vídeo pequena,consistindo de 11 linhas com 31 caracteres. Das onze linhas apenas sete são utilizadas para dados.

Outra restrição é quanto à programação. Só é possível escrever programas de crítica na linguagem assembler, do micro-processador utilizado. Neste ponto, o terminal inteligente leva uma grande vantagem pois permite ao usuário escrever diferentes programas de crítica para diferentes aplicações numa linguagem de alto nível e de fácil uso. Para tanto, ele conta também com um conjunto de declarações e, uma série de rotinas de crítica já codificadas e depuradas.

Quanto ao custo, cada unidade DE 523 era comercializada, já incluídos os encargos, por aproximadamente 4.000 dólares. O terminal inteligente, encontra-se em fase de protótipo e não foi ainda industrializado. No entanto, é estimado um custo bem inferior ao custo apresentado acima.

Algumas de suas características são semelhantes à do terminal inteligente, como por exemplo, o controle de cada unidade por intermédio de um microprocessador e a gravação dos dados em fita magnética cassete.

Em um ponto existe uma superioridade sobre o terminal inteligente. São possíveis operações de verificação, sendo esta uma lacuna no sistema aqui apresentado, devido à sua própria filosofia.

Finalizando, acredito na grande utilidade da aplicação e do terminal quando se tratar de aquisição de dados descentralizada, com pequeno ou médio volume e, onde o enfoque deva ser dado à qualidade dos dados coletados.