UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

APLICATIVO PARA AUTOMAÇÃO DE UMA PROCESSADORA DE CHEQUE PERTOCHEK 502SM COM CONSULTA DE RESTRIÇÃO COMERCIAL ATRAVÉS DO MICROCOMPUTADOR

ROBERTO ZAPELLA

BLUMENAU
2004

2004/1-34

ROBERTO ZAPELLA

APLICATIVO PARA AUTOMAÇÃO DE UMA PROCESSADORA DE CHEQUE PERTOCHEK 502SM COM CONSULTA DE RESTRIÇÃO COMERCIAL ATRAVÉS DO MICROCOMPUTADOR

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Wilson Pedro Carli - Orientador

BLUMENAU
2004

2004/1-34

APLICATIVO PARA AUTOMAÇÃO DE UMA PROCESSADORA DE CHEQUE PERTOCHEK 502SM COM CONSULTA DE RESTRIÇÃO COMERCIAL ATRAVÉS DO MICROCOMPUTADOR

Por

ROBERTO ZAPELLA

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Wilson Pedro Carli – Orientador, FURB
Membro:	Prof. Paulo Roberto Dias, FURB
Membro:	Prof. Everaldo Artur Grahl, FURB

Blumenau, 28 de junho de 2004

Dedico este trabalho a minha esposa Lenira pelo apoio e incentivo e ao meu filho Roberto pelas horas em que estive ausente durante o desenvolvimento deste.

"Jamais considere seus estudos como uma obrigação, mas como uma oportunidade invejável (...) para aprender a conhecer a influência libertadora da beleza do reino do Espírito, para seu próprio prazer pessoal e para proveito da comunidade à qual seu futuro trabalho pertencer."

Albert Einstein

AGRADECIMENTOS

À Check Check pelo fornecimento das informações necessárias para implementação da consulta de crédito.

À Sandro Luís Schmidt, proprietário da franquia Check Check de Jaraguá do Sul, pelo empenho na obtenção das informações técnicas para o desenvolvimento deste trabalho.

RESUMO

Este trabalho apresenta o desenvolvimento de um aplicativo para automação comercial em ambiente Delphi, focado no processamento de cheques. O aplicativo tem por característica principal o preenchimento de cheques baseado em uma consulta de restrição comercial e geração de uma base de dados com o histórico dos cheques preenchidos por cliente. A consulta à Check Check, empresa de proteção ao crédito, utiliza o protocolo TCP/IP e a comunicação do aplicativo com a processadora de cheques PertoCheck é realizada através da porta serial via *interface* RS-232C.

Palavras chaves: Delphi; RS-232C; TCP/IP; Check Check; Pertochek.

ABSTRACT

This work presents the development of a commercial automation application in Delphi environment, focused on the processing of checks. The application has for characteristic the fulfiling of checks based on a consultation of commercial restriction and generation of a base of data with the historical of the checks fulfilled by customer. The consultation to Check Check, credit protection enterprise, uses the TCP/IP protocol and the communication of the application with the checks printer PertoCheck it is accomplished through the serial communication port, via RS 232C interface.

Key-Words: Delphi; RS-232C; TCP/IP; Check Check; Pertochek.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

ACK – Acknowledge

BPS – *Bits* por Segundo

CMC7 – Character Magnetic Code 7

DFD – Diagrama de Fluxo de Dados

DLL – Dynamyc Link Library

ETX – End of Text

NAK – Negative Acknowledge

RS-232C – Recommend Standard 232 *Communication*

STX – Start of Text

TCP-IP – Transmission Control Protocol – Internet Protocol

SUMÁRIO

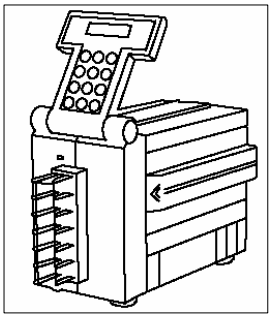
1 INTRODUÇÃO

Os estabelecimentos comerciais têm um grande interesse em adotar sistemas de automação comercial, pois este setor lida com uma grande variedade de itens. Isso torna fundamental ter sistemas que agilizem a administração para aumentar a produtividade e a competitividade (AutomaSoft, 2004).

Automatizar uma loja envolve uma série de fatores que começa pelo tamanho do estabelecimento, condição que influi na hora de determinar os melhores equipamentos e programas a serem adotados. As soluções diferem também quanto ao tipo de loja (auto-serviço e atendimento pessoal). Desta forma, existe uma série de fatores que determinam o tipo de solução mais adequado para essa ou aquela loja.

Além do fator competitividade, segundo AnJaraguá (2003), existe uma preocupação que está levando comerciantes a procurarem maneiras de diminuir o número de cheques sem fundos, sustados ou extraviados recebidos no mercado. De acordo com um levantamento da empresa SERASA S/A, no Brasil são passados cerca de 180 mil cheques sem fundos por dia. A instalação de sistemas de conferência de cheques em estabelecimentos comerciais, pode reduzir consideravelmente os prejuízos no caso de estarem sem fundos.

Dentro deste contexto e considerando que o autor trabalha em um estabelecimento comercial que utiliza uma processadora de cheques Pertochek, modelo que pode ser visto na Figura 1, que funciona de modo autônomo, com teclado próprio e *modem* interno e que realiza consultas de restrição comercial através do "*Check Check* Serviço de Proteção ao Crédito", para posterior preenchimento do cheque, verificou-se que poderia ser agilizado o atendimento no caixa, caso a processadora de cheque que hoje funciona neste estabelecimento de modo autônomo, estivesse conectada a um microcomputador, utilizando um banco de dados com informações de cada cliente.



Fonte: Pertochek (1999)

Figura 1 – Processadora de cheque Pertochek

Sendo assim, neste trabalho foi desenvolvido um aplicativo para agilizar o atendimento ao cliente. Este aplicativo deve ser instalado em um microcomputador conectado via porta serial à processadora de cheque e realiza consultas de informação de crédito junto a empresa Check Check. Além de atender o cliente mais rapidamente e oferecer a ele um serviço de qualidade, diminui para a empresa o recebimento de cheques sem fundos com a consulta de proteção ao crédito antes do preenchimento do cheque. Este aplicativo armazena um histórico das compras com cheques de cada cliente. O aplicativo, além do preenchimento básico que consiste em valor numérico, valor por extenso e data, deverá contemplar questões como o cruzamento do cheque, nome do beneficiário e cheque pré-datado.

1.1 OBJETIVOS DO TRABALHO

O objetivo principal foi o desenvolvimento de um aplicativo para preenchimento de cheques em uma processadora de cheque Pertochek, com consulta de restrição comercial ao provedor de serviço de proteção ao crédito *Check Check* e o armazenamento do histórico de cheques por cliente.

Como objetivos secundários tem-se:

a) estabelecer conexão com a *Check Check*, serviço de proteção ao crédito através do aplicativo, via protocolo TCP-IP;

b) realizar a comunição via porta serial entre a processadora de cheques e o microcomputador;

c) realizar a leitura do cheque através da captura automática do número do banco, do cheque, da agência, da conta corrente e da praça de compensação através do código *CMC7* (*Character Magnetic Code* 7) pela processadora de cheques, evitando erros de digitação por parte do usuário, objetivando agilizar o tempo de atendimento ao cliente.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado em capítulos conforme descrito a seguir:

O primeiro capítulo apresenta uma introdução ao trabalho, seus principais objetivos, bem como a estruturação do mesmo.

O segundo capítulo apresenta conceitos e fundamentos relacionados à comunicação de dados e protocolos de comunicação. Também são apresentados informações referente a processadora de cheque Pertochek e descritos serviços da empresa *Check Check*.

O terceiro capítulo relata as etapas de desenvolvimento do aplicativo, desde requisitos, especificação, implementação, técnicas utilizadas, como também os resultados obtidos.

O quarto capítulo descreve as conclusões obtidas com o desenvolvimento deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 COMUNICAÇÃO DE DADOS

Conforme Tafner (1996), o processo de comunicação de dados entre dois recursos computacionais ou mais, depende de diversas variáveis técnicas e envolve inúmeras considerações. Fundamentalmente, o processo de comunicação envolve a presença de um recurso transmissor, um recurso receptor, o meio de transmissão e a mensagem a ser transmitida. Esse processo de transferência que abrange fonte e destino pode ser realizado, basicamente, de duas formas:

a) transferência serial: os *bits* que representam uma informação (*byte*) são transmitidos seqüencialmente, um a um, por um único suporte físico. Esta é a forma de transferência mais utilizada entre recursos computacionais;

b) transferência paralela: os *bits* que representam uma informação (*byte*) são transmitidos simultaneamente, através de diversos suportes físicos em paralelo. Esta forma de transferência é utilizada para curtas distâncias entre computadores e principalmente na comunicação entre microcomputador e impressora.

2.2 SENTIDO DA TRANSMISSÃO

Na comunicação de dados, conforme Tafner (1996), a transferência dos dados é classificada de acordo com o sentido da transmissão. O sentido da transmissão determina claramente a figura do emissor e do receptor. Em relação ao sentido da transmissão, a comunicação pode ser:

a) simplex: caracteriza a transmissão de dados num único sentido, da origem (transmissor) para o destino (receptor), durante o processo de comunicação. É pouco utilizado por não possibilitar uma resposta do receptor;

b) half-duplex: a transmissão pode ser realizada nos dois sentidos, mas não simultaneamente. O transmissor e o receptor se alternam durante o processo de comunicação, quando for necessário;

c) full-duplex: a transmissão de dados é realizada nos dois sentidos simultaneamente. Porém, para que isso seja possível é necessário disponibilizar dois caminhos distintos, um canal (meio de transmissão) para cada um dos sentidos.

2.3 MODOS DE TRANSMISSÃO

De acordo com Tafner (1996), há dois modos de transmissão de dados:

a) transmissão assíncrona: sua principal característica é poder ser iniciada em qualquer tempo, sem limite de tamanho de mensagem. Neste modo de transmissão, cada *caracter* recebe *bits* adicionais (*start bit* e *stop bit*) que indicarão o início e o fim dos mesmos, não utilizando temporização para transmissão ou recepção de dados;

b) transmissão síncrona: define um "tempo" para que haja troca de informação entre transmissor e receptor. Define também que o controle do tempo será administrado por um *clock* na controladora e resetado através de *caracteres* de controle.

2.4 VELOCIDADE DA TRANSMISSÃO

Segundo Tafner (1996), a velocidade da transmissão refere-se a quantidade de informações transferida do transmissor para o receptor, num determinado intervalo de tempo, isto é, o tempo em que os *bits* gerados no transmissor levam para percorrer o meio até serem recebidos no receptor. Essa velocidade é expressa em *bits* por segundo (*bps)*. A velocidade de propagação do sinal no meio é expressa em *"baud"*.

2.5 INTERFACE PADRÃO RS-232C

Segundo Campbell (1986), há diversos padrões normativos e proprietários de *interface* serial, ou seja, especificações de como deve ser realizada a troca de sinais serialmente. Porém, para a transmissão de dados tem-se uma *interface* que se destaca pelo número de implementações e pela padronização. Essa *interface*, normativa, denomina-se RS-232. Nela é especificado um padrão de sinais elétricos e respectivas características mecânicas. No total são 25 sinais distribuídos entre controles e dados e, originalmente foi projetada para conectar as controladoras de transmissão de dados com o *modem*, mas são utilizadas também para diversas conexões locais com os mais variados periféricos.

2.6 DYNAMIC LINK LIBRARY

Segundo Osier (1997), *Dynamic Link Library* (*DLL*) significa biblioteca de vínculo dinâmico, que é um tipo especial de arquivo executável usado para armazenar funções e recursos em um arquivo separado do executável. Normalmente escreve-se um programa e criam-se funções, recursos e estes são vinculados ao arquivo executável, criando o um vínculo estático. Quando são utilizadas rotinas de uma DLL, isso se chama vínculo dinâmico, pois o código é vinculado durante a execução do programa.

Cantù (2000) descreve que quando uma sub-rotina não está diretamente disponível em um arquivo fonte, o compilador inclui a sub-rotina em uma tabela interna, a qual inclui todos os símbolos externos. Após a compilação normal, estática, o ligador (*linker*) busca o código compilado da sub-rotina em uma unidade compilada e o inclui no executável. O arquivo EXE resultante inclui todo o código do programa e das unidades envolvidas. No caso do vínculo dinâmico, que ocorre quando o código chama uma função baseada em DLL, o ligador simplesmente utiliza as informações da declaração externa da sub-rotina para configurar algumas tabelas no arquivo executável. Quando o sistema operacional carrega o arquivo executável na memória, primeiro ele carrega todas as DLL's exigidas e depois o programa começa. Sempre que o programa chama uma função externa, ele usa uma tabela interna para encaminhar a chamada para o código DLL. O esquema descrito acima não envolve dois aplicativos diferentes. A DLL se torna parte do programa que está sendo executado e é carregado no mesmo espaço do endereço. Toda a passagem de parâmetros ocorre na pilha do aplicativo, pois a DLL não tem uma pilha separada.

2.6.1 PARA QUE SERVE A DLL

Segundo Cantù (2000), se programas diferentes utilizam a mesma DLL, essa DLL é carregada na memória apenas uma vez, economizando assim a memória de sistema. As DLL's são mapeadas no espaço de endereço privativo de cada processo, mas seu código é carregado na memória apenas uma vez.

Outra utilidade da DLL é que se pode fornecer uma versão diferente de uma DLL, substituindo a atual. Se as sub-rotinas da DLL tem os mesmos parâmetros, elas podem executar o programa com a nova versão da DLL, sem ter que recompilar a aplicação. Se a DLL tem novas sub-rotinas, isso não gera erros, porém, podem ocorrer erros se uma sub-rotina mais antiga estiver sido retirada da nova DLL.

Estas vantagens genéricas se aplicam a vários casos. Se tem-se um algoritmo complexo ou alguns formulários complexos exigidos por vários aplicativos, pode armazená-los em uma DLL. Isso permitirá a redução do tamanho do executável e economia de memória, quando executados vários aplicativos que utilizam as mesmas DLL's simultaneamente.

A segunda vantagem é aplicável aos aplicativos complexos. Quando um programa é muito grande e exige freqüentes atualizações e correções, dividi-lo em vários executáveis e DLL's permite a distribuição apenas das partes alteradas. Outra técnica comum é usar DLL's para armazenar exclusivamente recursos. Diferentes versões de uma DLL contendo *strings* para diferentes linguagens e depois mudar a linguagem em tempo de execução, ou preparar uma biblioteca de ícones e *bitmaps* e depois utilizá-los em diferentes aplicativos. Outra vantagem, é que as DLL's são independentes da linguagem de programação, ou seja, pode-se construir uma DLL no Delphi e chamá-las, por exemplo, a partir do ambiente de programação C++ ou Visual Basic.

2.7 TCP/IP

Segundo Silva (1999), TCP/IP é um acrônimo para o termo *Transmission Control Protocol/Internet Protocol Suite*, ou seja é um conjunto de protocolos, onde dois dos mais importantes (TCP e IP) deram seus nomes à arquitetura. A arquitetura TCP/IP teve como princípios a independência em relação a tecnologias de redes, bem como a facilidade de interligação de redes de tecnologia distintas, ou seja, independência da topologia das ligações

entre elas. Para que o TCP/IP alcançá-se sucesso, foi necessária a concepção de uma arquitetura em camadas, cada uma responsável por contornar os complicadores apresentados (tecnologias distintas, topologia das redes, plataformas distintas).

As principais características da arquitetura TCP/IP, conforme descrito por Silva (1999) são:

a) conectividade no nível de rede: para implementar uma aplicação de comunicação, basta que os programas sejam executados apenas nos equipamentos envolvidos diretamente na aplicação (origem e destino);

b) controle de fluxo *end-to-end:* a responsabilidade do controle do fluxo de dados entre os computadores em que roda as aplicações foi delegado apenas à eles (origem e destino), de forma que os computadores intermediários, pelos quais os dados trafegam, devem proporcionar apenas o encaminhamento desses dados por meio das redes intermediárias;

c) endereçamento lógico universal: identificador de formato universal, não relacionado a qualquer tecnologia. Esse identificador permite a localização de computadores, independente das tecnologias de rede por eles implementadas.

2.7.1 TRANSMISSION CONTROL PROTOCOL

Conforme mencionado por Silva (1999), o TCP é um protocolo da camada de transporte da arquitetura *Internet* TCP/IP. O protocolo é orientado a conexão e fornece um serviço confiável de transferência de arquivos *end-to-end*. Ele é responsável por inserir as mensagens das aplicações dentro do datagrama de transporte, reenviar datagramas perdidos e ordenar a chegada de datagramas enviados por outro micro. O TCP foi projetado para funcionar com base em um serviço de rede sem conexão e sem confirmação.

O protocolo TCP interage de um lado com processos das aplicações e do outro com o protocolo da camada de rede da arquitetura Internet. A *interface* entre o protocolo e a camada superior consiste em um conjunto de chamadas. Existem chamadas, por exemplo, para abrir e fechar conexões e para enviar e receber dados em conexões previamente estabelecidas. Já a

interface entre o TCP e a camada inferior define um mecanismo através do qual as duas camadas trocam informações assincronamente.

Conforme descrito por Starlin (1998), o protocolo TCP não exige um serviço de rede confiável para operar, logo responsabiliza-se pela recuperação de dados corrompidos, perdidos, duplicados ou entregues fora de ordem pelo protocolo de rede.

2.7.2 INTERNET PROTOCOL

Segundo Silva (1999), o protocolo IP, padrão para redes *Internet*, é baseado em um serviço sem conexão. Sua função é transferir blocos de dados, denominados datagramas, da origem para o destino, onde a origem e o destino são *hosts* identificados por endereços IP. Este protocolo também fornece serviço de fragmentação e remontagem de datagramas longos, para que estes possam ser transportados em redes onde o tamanho máximo permitido para os pacotes é pequeno.

Como o serviço fornecido pelo protocolo IP é sem conexão, cada datagrama é tratado como uma unidade independente que não possui nenhuma relação com qualquer outro datagrama. A comunicação é não-confiável, pois não são utilizados reconhecimentos *end-to-end* ou entre nós intermediários. Não são empregados mecanismos de controle de fluxo e de controle de erros. Apenas uma conferência simples do cabeçalho é realizada, para garantir que as informações nele contidas, usadas pelos *gateways* para encaminhar datagramas, estão corretas.

Os endereços IP são números de 32 *bits*, normalmente escritos como quatro octetos na forma decimal, como por exemplo, o endereço 200.236.143.1. A primeira parte do endereço identifica uma rede específica na inter-rede, a segunda parte identifica um *host* dentro desta rede. Este endereço, portanto, pode ser usado para nos referirmos tanto a redes quanto a um *host* individual. É através do endereço IP que os *hosts* conseguem enviar e receber mensagens pela rede, em uma arquitetura Internet TCP/IP.

2.7.3 PORTS

Segundo Starlin (1998), um *port* pode ser visto como um canal de comunicação para uma máquina. Pacotes de informações chegando à uma máquina não são apenas endereçadas à máquina e sim à programas ou serviços em determinados *ports*. Um computador normalmente não está "escutando" a todos os *ports*, ele "escuta" alguns *ports* específicos, e ele não vai responder a um pedido que chegue a um *port* ao qual ele não esteja "escutando".

2.8 PROCESSADORA DE CHEQUES PERTOCHEK

Esta processadora tem como objetivo, adequar os pagamentos em cheques de forma rápida e segura. A Pertochek é desenvolvida e produzida pela Perto S/A, empresa nacional com certificação ISO 9001, com alta tecnologia e padrão de qualidade internacional (Pertochek, 1999).

A Pertochek 128K Consult, é a primeira processadora de múltiplos meios de pagamento do país. Entre as principais características desta processadora de cheques encontram-se:

a) impressão a jato de tinta, super silenciosa;

b) impressão em cheques dobrados e amassados;

c) realiza leitura de cheque capturando o n° do banco, do cheque, da agência, da conta corrente e da praça de compensação;

d) realiza o alinhamento automático do cheque;

e) permite o cruzamento do cheque e impressão de cheques pré-datados.

A processadora consulta bases de dados externas, por exemplo da empresa SERASA S/A, diretamente ou através de uma rede eletrônica de consulta. A linha da Pertochek é composta de vários modelos, com recursos capazes de atender às diversas necessidades do mercado, pois podem operar de modo autônomo, conectados à microcomputadores ou PDV's, através de cabo serial ou paralelo.

2.8.1 INTERFACE DE COMUNICAÇÃO

A Pertochek dispõe de uma *interface* de comunicação serial padrão RS-232C, utilizada para a comunicação com um computador (*host*), que vai lhe comandar as operações. Tem-se como características lógicas:

a) *interface* assíncrona *full-duplex*;
b) palavra: 8 *bits*;
c) número de *stop bits*: 1;
d) paridade: nenhuma;
e) taxa de comunicação: 1200, 2400, 4800, 9600 *bps*.

A mudança da taxa de comunicação é feita na placa da CPU da Pertochek, que contém quatro *dip-switches* conforme Figura 2. As *dip-switches* de número 1 e 2 determinam o *baud-rate*. A Tabela 1 ilustra a posição das chaves para as diferentes taxas de transmissão.

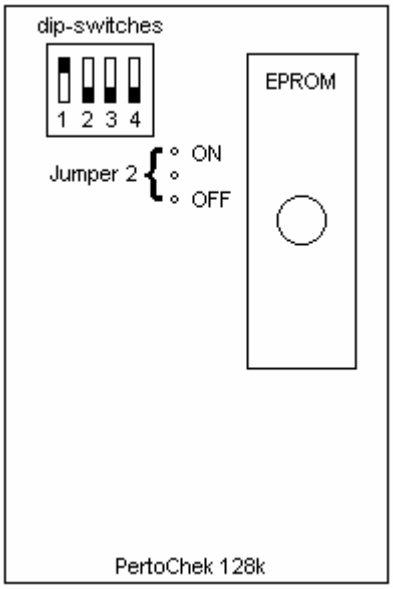


Figura 2 – Posição *dip-switches*

Fonte: Pertochek (2002)

Tabela 1 - Posição *dip-switches*

Pertochek 128k		
Boud rate	Dip 1	Dip 2
1200	OFF	OFF
2400	OFF	ON
4800	ON	OFF
9600	ON	ON

Fonte: Pertochek (2002)

Tem-se como características físicas:

a) conector de 8 pinos tipo RJ-45, conforme Tabela 2;
b) cabo serial com comprimento máximo de 15 metros.

Tabela 2 – Descrição da pinagem do conector

Pino	Descrição
Pino 1	Não utilizado
Pino 2	TX – saída – transmissão de dados
Pino 3	RX – entrada – recepção de dados
Pino 4	DSR – entrada - indica que a Pertochek pode transmitir dados
Pino 5	GND
Pino 6	DTR– saída – indica que a Pertochek está pronta p/ receber dados
Pino 7	Não utilizado
Pino 8	Não utilizado

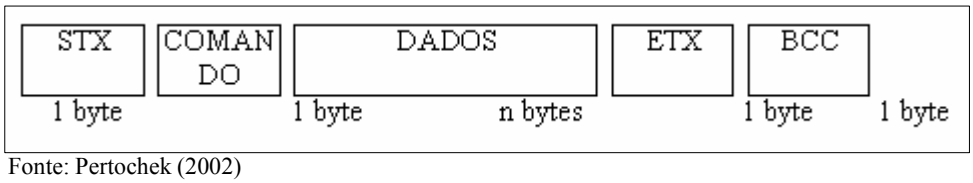
Fonte: Pertochek (2002)

2.8.2 MENSAGENS PERTOCHEK

Todas as mensagens recebidas ou enviadas pela Pertochek devem ter o mesmo formato. Cada mensagem consiste de uma *string* de *bytes*, composta respectivamente por:

a) 1 *byte* = STX (ASCII) para indicar o início da mensagem (*Start of Text*);
b) 1 *byte* com o *caracter* de comando;
c) "n" *bytes* com *caracteres* de dados;
d) 1 *byte* = ETX (ASCII) para indicar o fim da mensagem (*End of Text*);
e) 1 *byte* de conferência.

A Figura 3 mostra a sintaxe padrão da *string*.



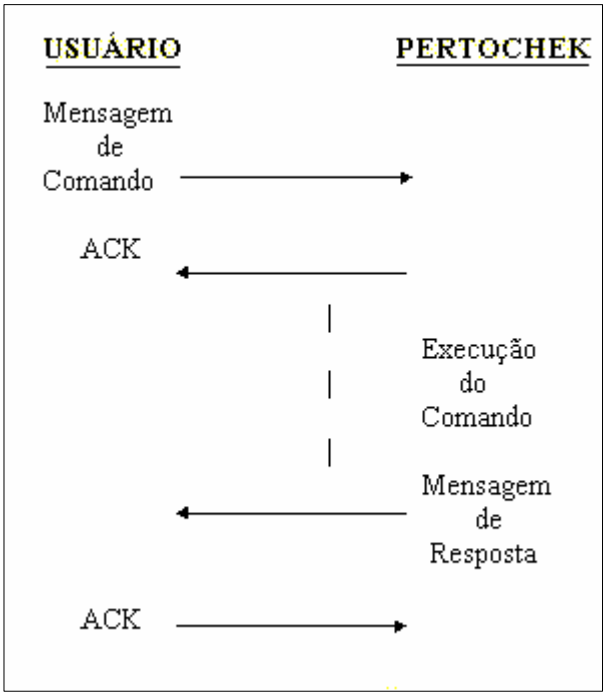
Fonte: Pertochek (2002)

Figura 3 – Sintaxe das mensagens de comando e de resposta

A comunicação é totalmente bidirecional. A máquina comporta-se como uma unidade escrava e envia mensagens unicamente em resposta aos comandos. Se o tempo entre dois *caracteres* da mensagem for maior que 500 milisegundos, a Pertochek irá ignorar a mensagem e aguardar um outro STX (*Start of Text*). Se a Pertochek receber um NAK (*Negative Acknowledge*) em resposta a sua mensagem, ela a enviará novamente. Um *caracter* inválido ou errado é tratado como um NAK. Se a Pertochek receber um STX, ela sempre entenderá isto como o início de uma nova mensagem, e irá ignorar qualquer mensagem prévia que não tenha sido concluída por um ETX (*End of Text*).

Por segurança é necessário que, quando for enviado um comando, obtenha-se uma resposta de que este tenha sido executado corretamente, ou esperar indicação de erro, para então enviar outro comando.

Segue na Figura 4 uma troca típica de mensagens:



Fonte: Pertochek (2002)

Figura 4 – Troca de Mensagens

2.8.3 ERROS DE SINTAXE NAS MENSAGENS

A Pertochek responderá com um NAK (*Negative Acknowledge*) para as seguintes situações:

a) *caracter* inválido;
b) um STX (*Start of Text*) seguido por mais de 1199 *caracteres* diferentes de ETX (*End of Text*).

A pertochek irá ignorar qualquer mensagem com os erros descritos e, depois de enviar um NAK, aguardará uma nova mensagem sem erros. A Pertochek não transmite qualquer mensagem que não seja solicitada pelo usuário.

Quando uma mensagem correta for recebida pela Pertochek uma ACK será enviada dentro de 15 milisegundos (ms). Para comandos que não pedem a inserção de uma documento

na máquina, a mensagem de resposta começará a ser enviada entre 50 milisegundos e 2 segundos depois do ACK ter sido enviado. Para comandos onde um documento deve ser inserido na máquina, a mensagem de resposta começará a ser enviada entre 50 ms e 1 minuto depois do ACK ter sido enviado (o menor tempo 50 ms será quando ocorrer erro durante a execução do comando).

2.9 EMPRESA CHECK CHECK

Segundo Check Check (2004), atuando no mercado há mais de 42 anos, a empresa percebeu, desde o início de suas atividades, que cada cliente é único. Único em necessidade de informações e atendimento. Por isso, investiu em relacionamento e priorizou o desenvolvimento de produtos e serviços alinhados com as necessidades de seus clientes.

A Check Check está presente em mais de 1.800 municípios e atende cerca de 50.000 clientes. Para satisfazer as necessidades desses clientes, sua rede é formada por mais de 120 pontos de atendimento, distribuídos estrategicamente pelas capitais e principais cidades brasileiras. Cada ponto de atendimento conta com uma infra-estrutura completa, preparada para atender qualquer empresa, seja de pequeno, médio ou grande porte.

A empresa oferece aos seus usuários informações de abrangência nacional. São mais de 170 milhões de registros armazenados e cerca de 5 milhões de consultas realizadas todo mês. Seu banco de dados é atualizado diariamente para garantir tranqüilidade no momento de conceder crédito, escolher fornecedores, identificar clientes em potencial ou contratar funcionários.

O investimento na integração entre informática e telecomunicações é uma das preocupações da Check Check. Assim, pensando em fornecer aos seus clientes o que melhor corresponde as exigências do mercado, ela disponibiliza uma grande quantidade de meios de acesso para consultas.

A Check Check disponibiliza meios de acesso adequados para as mais variadas necessidades de informação de seus clientes. É uma empresa de destaque nacional em

oferecer aos clientes as melhores opções em equipamentos para realização de consultas. Além de tecnologia de ponta, ela dispõe da maior rede de assistência técnica do País.

A empresa Check Check disponibiliza os seguintes meios de acesso para consultas:

a) Check DSP Bank 7: terminal de consulta de crédito e cadastro. Tecnologia DSP (processamento digital de sinais). Conexão via rede de pacotes. Possui leitora de CMC7 que permite a captura eletrônica dos dados do cheque;

b) Software: exclusivo da Check Check que realiza todas as modalidades de consultas através de protocolo de internet ou rede de pacotes. Tripla funcionalidade: servidor de consultas, cliente de consultas e consultas por base local (restrições básicas);

c) TCC Check: terminal de consulta que faz conexões por redes de pacotes. Realiza as principais consultas para se aprovar uma venda com cheque ou verificar informações cadastrais;

d) Celulares Wap: buscando atender as necessidades de todos os clientes e estar presente em todos os diferentes meios de comunicação, a Check Check disponibiliza consultas de crédito pelo celular, através do sistema WAP;

e) www.checkcheck.com.br: pela Internet, é possível efetuar todos os tipos de consulta, imprimir extrato de consultas e restrições comerciais, proceder inclusão e exclusão de restrição comercial;

f) Serviço 0300: por meio do Serviço 0300 pode-se realizar as consultas Check Check ou registrar telebloqueios. O acesso ao banco de dados pelo 0300 pode ser: via URA (Unidade de Resposta Audível) ou pelo atendimento personalizado, com auxílio de um operador;

g) TSP Check: processadora de cheques que realiza consultas na Check Check.

3 DESENVOLVIMENTO DO TRABALHO

Este capítulo apresenta requisitos, especificação, diagrama de Use Case, dicionário de dados, fluxogramas, técnicas e ferramentas, operacionalidade da implementação, ou seja, itens considerados relevantes para o desenvolvimento deste trabalho.

Considerando a situação da processadora de cheque antes do desenvolvimento deste trabalho, a mesma não se comunicava com o microcomputador. Ela tinha um teclado próprio com o qual o usuário interagia com a mesma e um modem interno para realizar as consultas. Para a comunicação da processadora de cheque com o microcomputador foi necessáro alterar a *Eprom* da processadora, sendo que a partir desta alteração ela desabilitou o modem e habilitou a comunicação serial, condição técnica restringida pelo fabricante. Devido a este fato, a consulta antes realizada pela processadora, tem agora que ser realizada via microomputador através do aplicativo desenvolvido neste trabalho. Outra situação que levou ao desenvolvimento deste trabalho foi o fato da processadora não permitir armazenar um histórico sobre o registro dos cheques por cliente.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA

Considerando que as soluções existentes no mercado ou permitem consulta a Check Check e não comunicam-se com a processadora de cheque modelo Pertochek 502SM ou permitem comunicação com a processadora de cheque e não realizam a consulta na Check Check, decidiu-se então, desenvolver neste trabalho um aplicativo como uma solução para esta situação. Este aplicativo deve:

a) permitir que o usuário registre o CPF e o valor do cheque do cliente;
b) permitir cadastrar novos clientes;
c) consultar restrição de crédito junto a Check Check baseado nas informações do cliente;
d) encerrar antes do preenchimento do cheque caso a consulta retorne restrição comercial para o cliente;
e) ler os dados do cheque pelo CMC7 (nº do banco, nº da conta, nº da agência, nº do cheque e praça de compensação) através da processadora de cheque;

f) preencher cheques;
g) cruzar todos os cheques preenchidos;
h) permitir o preenchimento de cheques pré-datados.

Como requisitos não-funcionais deste aplicativo tem-se:

a) em termos de eficiência, em 90% dos casos a consulta a Check Check e o preenchimento do cheque devem gastar aproximadamente 25 segundos (acesso discado);
b) interface do aplicativo simples, intuitiva para o usuário, seguindo os padrões dos aplicativos desenvolvidos *for windows*;
c) o usuário deverá ser capaz de aprender a operar o aplicativo com até duas horas de treinamento.

Para a definição da abrangência do aplicativo, foram definidas algumas restrições:

a) sistema operacional Windows 98;
b) modelo da processadora de cheque: Pertocheck 502SM;
c) empresa de consulta sobre informação de crédito: Check Check;
d) tipo do serviço de consulta: Check Check+;
e) comunicação com a processadora de cheques através de uma porta serial do microcomputador;
d) o aplicativo não deve ter nenhum tipo de restrição quanto a freqüência de uso ou quanto ao acesso pelo usuário;
f) conexão discada através do protocolo TCP/IP para consulta com a empresa de consulta de informação de crédito.

3.2 ESPECIFICAÇÃO

Para a especificação do aplicativo foram utilizadas as técnicas de diagramação a seguir apresentadas:

a) Diagrama de *Use Case*: modelo que especifica a funcionalidade que o sistema tem a oferecer da perspectiva do usuário. Utiliza atores para modelar qualquer coisa que necessite trocar informações com o sistema e casos de uso para apresentar o

que os usuários estão aptos a fazer com o sistema. Para a especificação do *Use Case* foi utilizada a ferramenta *Rational Rose*;

b) Fluxograma: fundamentalmente uma ferramenta de codificação. Apresenta de forma gráfica a seqüência na qual os comandos ou blocos de processo serão executados e a lógica de controle da execução. Foi utilizado no desenvolvimento deste trabalho para especificar a comunicação do aplicativo com a processadora de cheques e para especificar o processo de consulta à operadora de informação de crédito. Nos fluxogramas foi utilizado o editor de textos *Word* como ferramenta de representação gráfica.

c) Modelo Entidade Relacionamento: é um modelo lógico do banco de dados que contém as entidades, atributos e os relacionamentos entre as entidades. Para a especificação do Modelo Entidade Relacionamento foi utilizada a ferramenta *Power Designer.*

3.2.1 USE CASE

No *Use Case* observa-se a funcionalidade do aplicativo, conforme ilustrado na Figura 5, onde tem-se como atores a processadora de cheques, responsável pelo preenchimento do cheque e leitura do CMC7, a empresa Check Check que fornece informações de restrição comercial e o usuário que configura o aplicativo, cadastra clientes, realiza consultas, preenche cheques e emite relatórios.

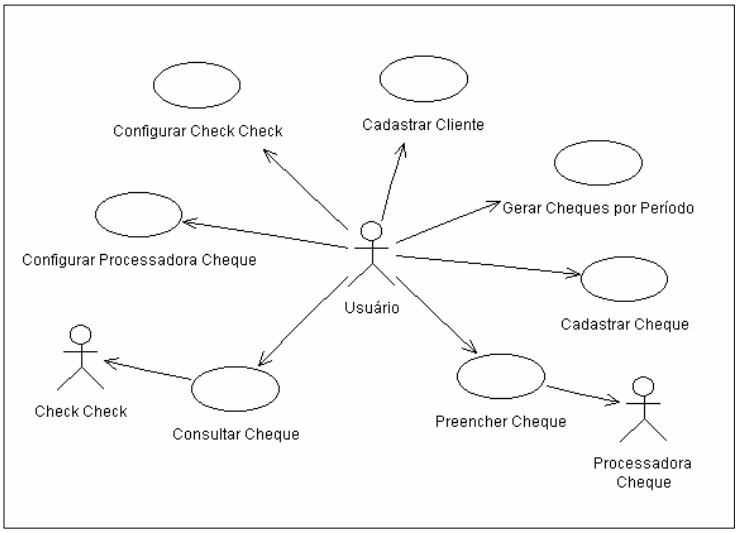


Figura 5 – Use Case

3.2.2 DESCRIÇÃO DOS CASOS DE USO

A seguir são descritos os casos de uso:

a) configurar Check Check: usuário configura provedor de acesso, porta de comunicação, assim como o código de acesso, senha e serial;

b) configurar processadora de cheque: usuário configura número da porta serial de comunicação com a processadora de cheque, velocidade de transmissão dos dados, cidade para preenchimento do cheque, beneficiário no caso de cheque nominal e nome da moeda para preenchimento do extenso no cheque;

c) cadastrar cliente: usuário cadastra dados pessoais do cliente como por exemplo: nome, endereço, CPF, telefone;

d) gerar cheques por período: usuário emite semanalmente relatório de cheques emitidos no período;

e) cadastrar cheque: usuário cadastra o cheque para manter um histórico dos cheques de cada cliente;

f) preencher cheque: aplicativo realiza o preenchimento do cheque na processadora de cheque baseado na configuração cadastrada e no valor de preenchimento fornecido pelo usuário. Opcionalmente o usuário pode informar uma data para pré-datar o cheque. A processadora realiza a leitura do CMC7 do cheque;

g) consultar cheque: aplicativo realiza consulta ao provedor de serviços da Check Check baseado na configuração cadastrada e nos dados do cliente (CPF e nº do telefone) fornecidos pelo usuário para verificar se o cliente tem alguma restrição comercial.

3.2.3 MODELO ENTIDADE RELACIONAMENTO

No Modelo Entidade Relacionamento estão descritos todos os relacionamentos entre as tabelas do aplicativo, a identificação das chaves primárias e as colunas de cada tabela, o qual está ilustrado na Figura 6. Além das tabelas Cliente, Cheque e UF, tem-se também as tabelas de Configuração da Processadora de Cheque e Configuração da Check Check que estão isoladas, por não se relacionarem com as demais tabelas e servirem exclusivamente para armazenamento das configurações necessárias para comunicação com a processadora de cheque e para a consulta de restrição comercial.

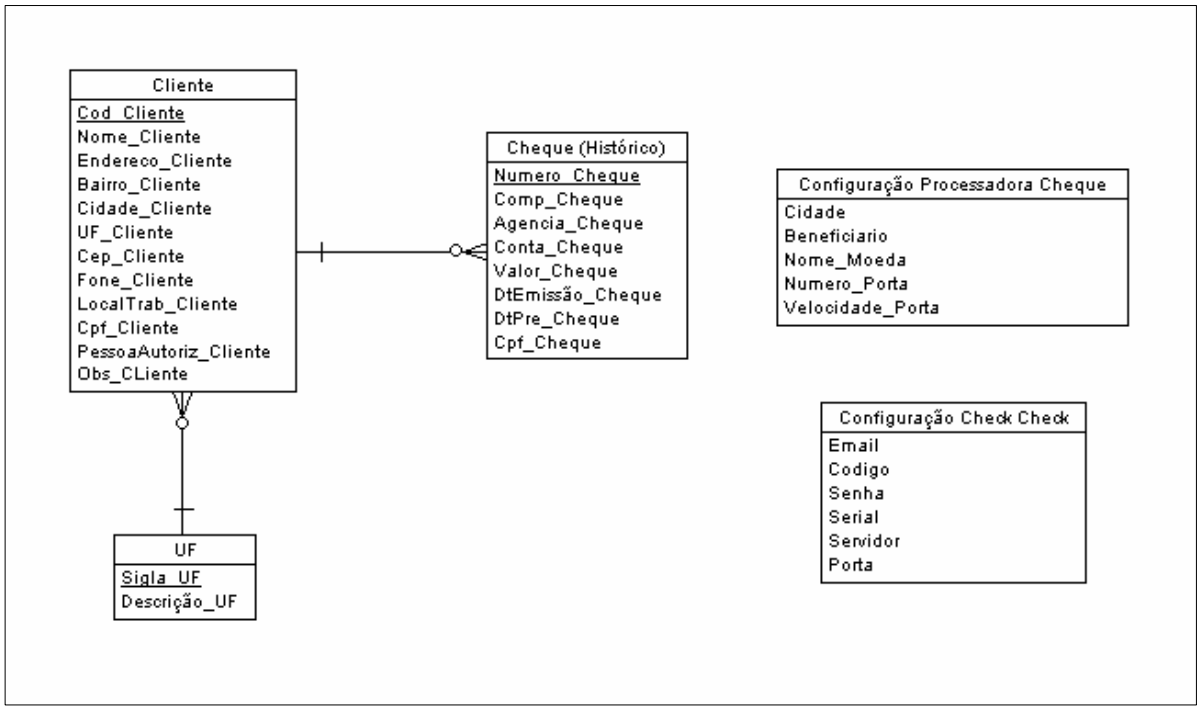


Figura 6 – Modelo Entidade Relacionamento

3.2.4 DICIONÁRIO DE DADOS

O dicionário de dados apresenta o nome do campo (*name*), o código (*code*), o tipo (*type*), se ele é um campo chave (*I*) e se é um campo obrigatório (M). Na Tabela 3 é possível visualizar os campos pertencentes a cada tabela do aplicativo.

Tabela 3 – Dicionário de Dados

Tabela Cheque

Name	Code	Type	I	M
Número do Cheque	NUMERO_CHEQUE	N6	Yes	Yes
Praça de Compensação	COMP_CHEQUE	N3	No	Yes
Número da Agência Bancária	AGENCIA_CHEQUE	N4	No	Yes
Número da Conta Bancária	CONTA_CHEQUE	A10	No	Yes
Valor do Cheque	VALOR_CHEQUE	N15	No	Yes
Data de Emissão	DTEMISSAO_CHEQUE	D	No	Yes
Data Pré_Datado	DTPRE_CHEQUE	D	No	No
CPF	CPF_CHEQUE	N14	No	Yes

Tabela Cliente

Name	Code	Type	I	M
Código	COD_CLIENTE	I	Yes	Yes
Nome	NOME_CLIENTE	A50	No	Yes
Endereço	ENDERECO_CLIENTE	A35	No	No
Bairro	BAIRRO_CLIENTE	A20	No	No
Cidade	CIDADE_CLIENTE	A25	No	No
UF	UF_CLIENTE	A2	No	No
Cep	CEP_CLIENTE	N8	No	No
Fone	FONE_CLIENTE	A35	No	No
Local de Trabalho	LOCALTRAB_CLIENTE	A35	No	No
CPF	CPF_CLIENTE	N14	No	Yes
Pessoa Autorizada Comprar Cheque	PESSOAAUTORIZ_CLIENTE	A50	No	No
Observação	OBS_CLIENTE	A50	No	No

Tabela Configuração Check Check

Name	Code	Type	I	M
Email para Contato	EMAIL	A45	No	Yes
Codigo Usuário	CODIGO	A6	No	Yes
Senha Usuário	SENHA	A6	No	Yes
Serial de Identificação	SERIAL	A6	No	Yes
Servidor de Acesso	SERVIDOR	A35	No	Yes
Porta de Acesso	PORTA	A4	No	Yes

Tabela Configuração Processadora Cheque

Name	Code	Type	I	M
Cidade de Preenchimento Cheque	CIDADE	A20	No	Yes
Beneficiario Cheque Nominal	BENEFICIARIO	A20	No	Yes
Nome da Moeda Preenchimento Cheque	NOME_MOEDA	A20	No	Yes
Número da Porta Serial	NUMERO_PORTA	A4	No	Yes
Velocidade da Porta Serial	VELOCIDADE_PORTA	A4	No	Yes

Tabela UF

Name	Code	Type	I	M
Sigla	SIGLA_UF	A2	Yes	Yes
Descrição	DESCRICAO_UF	A25	No	Yes

3.2.5 FLUXOGRAMA COMUNICAÇÃO PROCESSADORA CHEQUE

Conforme descrito em Pertocheck (2002), a aplicação deve, obrigatoriamente, fazer a inicialização da porta a ser utilizada antes de enviar qualquer comando para a processadora de cheques. Esta inicialização deve ser feita no início da aplicação. Uma vez aberta a porta de comunicação, a aplicação está pronta para enviar comandos para a processadora de cheques.

Sempre antes de enviar um comando para a máquina, é necessário verificar se a mesma não está processando nenhum outro comando, assim como, verificar a resposta retornada após a execução de um comando, mesmo que a resposta não interesse ao aplicativo. A ação de ler a resposta retornada, "limpa" o *buffer* de dados da porta em uso. Esse processo de comunicação do aplicativo com a processadora de cheque é ilustrado no fluxograma da Figura 7.

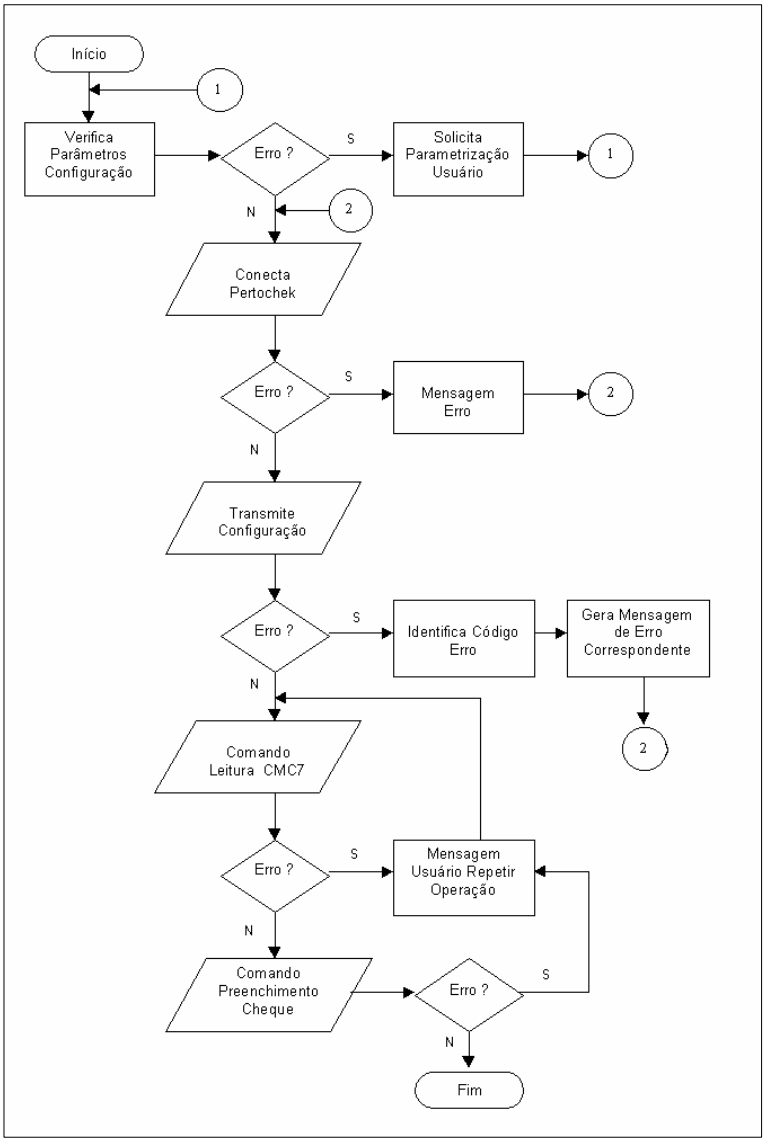


Figura 7 – Fluxograma comunicação com a processadora de cheque

3.2.6 FLUXOGRAMA DA CONSULTA DE RESTRIÇÃO DE CRÉDITO

Baseando-se nas informações contidas em Check Check (2003), desenvolveu-se o fluxograma que representa os processos para realizar a comunicação com a empresa de informação de crédito. Estes processos podem ser observados na Figura 8.

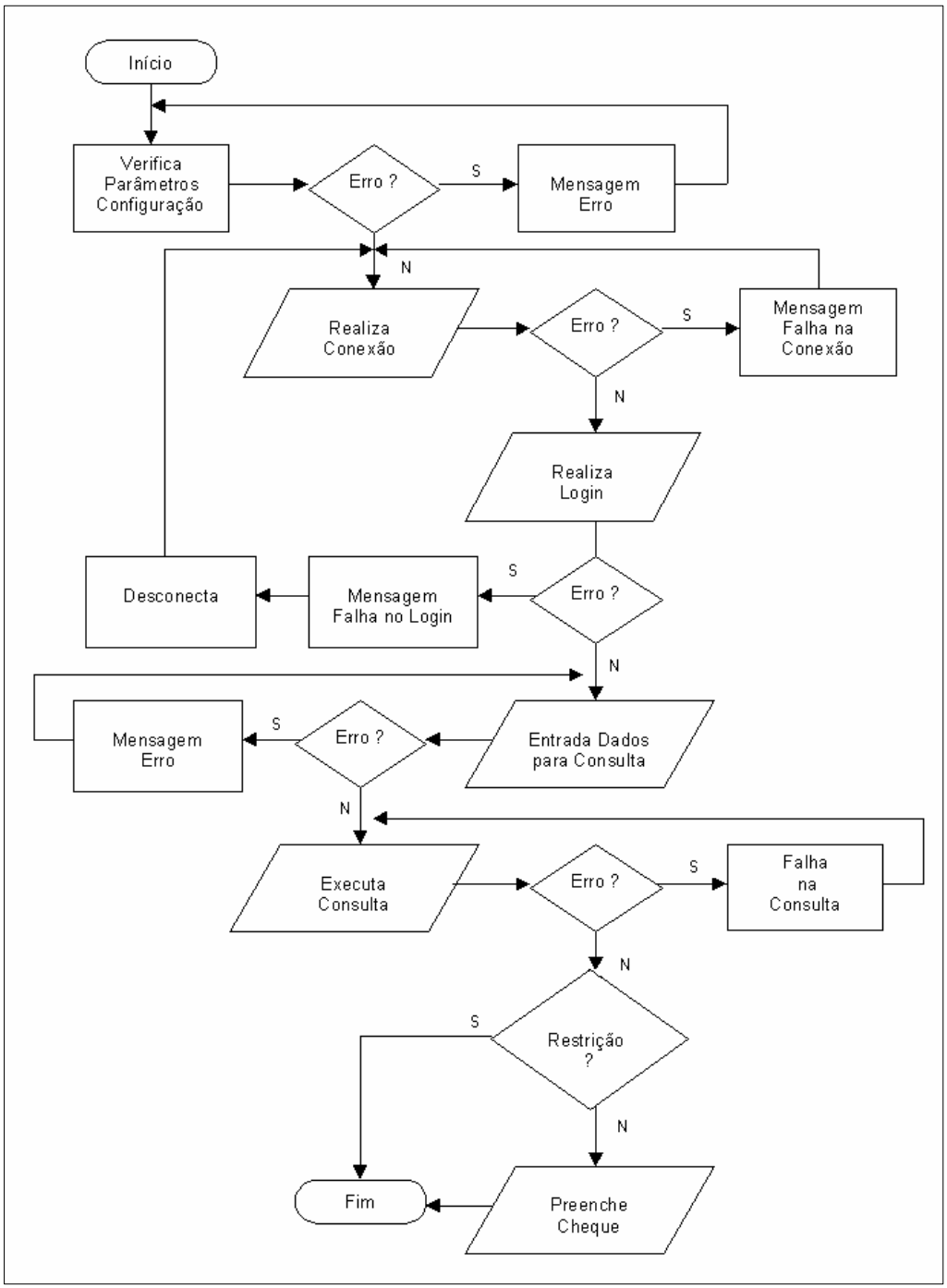


Figura 8 – Fluxograma da consulta de restrição de crédito

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas as técnicas e ferramentas utilizadas para desenvolvimento da aplicação e sua operacionalidade.

3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS NA IMPLEMENTAÇÃO

Para a implementação do aplicativo, adotou-se o ambiente Delphi 5, utilizando a linguagem de programação *Object Pascal* e o banco de dados nativo do ambiente denominado *Paradox*. Parte das funções e procedures do aplicativo pode-se observar no Anexo A.

3.3.1.1 DLL PERTOCHEK

Para a implementação do aplicativo referente a comunicação do microcomputador com a processadora de cheques, foram utilizadas DLL`s fornecidas pela Pertochek.

As funções das DLL`s fornecidas são as seguintes:

a) inicializar a porta de comunicação;

b) verificar se a máquina está ocupada (em processamento de alguma tarefa);

c) enviar comando para a máquina através da porta selecionada;

d) receber resposta que a máquina retorna após a execução de algum comando;

e) finalizar a porta de comunicação.

Estas funções encontram-se detalhadas no Anexo B.

3.3.1.2 DLL CHECK CHECK

A CheckNet é uma biblioteca desenvolvida pela Check Check para a realização de consultas via Internet. Atualmente a CheckNet está disponível no formato DLL, sendo possível o desenvolvimento de aplicações de consultas Check Check nas mais diversas

linguagens, dentre elas, Delphi, Visual Basic, Visual C++. A CheckNet é composta de um conjunto de funções estruturadas possibilitando ao desenvolvedor conectar-se à Check Check via protocolo TCP/IP. Todas as solicitações de consultas têm de ser enviadas para o endereço automacao.checkcheck.com.br, utilizando os serviços da porta 8734.

A CheckNet está organizada por conjunto de funções correspondentes a cada etapa de uma consulta Check Check. São elas, inicialização, configuração, conexão e consulta, conforme apresentadas no Anexo C.

3.3.1.3 GRUPOS DE RESPOSTAS CHECK CHECK

Conforme CheckCheck (2003), a resposta de uma transação TCP/IP é formada por um conjunto de grupos. Cada grupo de resposta contém informações específicas as quais são delimitadas por separadores conforme Tabela 4.

Tabela 4 – Separadores de Grupos, Registros e Campos

Separador	Nome	Significado
< > Chr(60) Chr(62)	Separador de Grupo	É o maior agrupamento de itens em uma resposta. É utilizado para quebrar uma resposta de forma modularizada por algum conceito importante dentro do conjunto de informações tratadas pelas demandas. Caso haja mais de um grupo em uma resposta, significa que a resposta completa é formada pela soma de todos os grupos.
| Chr(124)	Separador de Registro	É utilizado para separar o conjunto de informações que deve ser encapsulado em um Grupo. Cada item de um Grupo é um Registro.
, Chr(44)	Separador de Campo	É utilizado para separar os campos de um dado registro relacionado com um Grupo. Cada item desse conjunto é chamado de Campo.

Após o *caracter* de início de um grupo (<) sempre se segue o identificador do grupo, que é formado por 3 (três) *caracteres* alfanuméricos. As respostas do servidor são compostas por vários grupos, dessa forma cabe ao software cliente conhecer o formato de todos os grupos retornados pelo servidor. Todo grupo contém obrigatoriamente um nome único que o diferencia dos demais grupos. Cada grupo possui também um ou mais registros, que por sua vez possuem um ou mais campos.

Na Tabela 5 são apresentados os possíveis grupos que podem aparecer dentro da área de dados de uma resposta enviada pelo servidor.

Tabela 5 – Nomes de Grupos

Grupo	Significado
NOM	Nomes relacionados ao CPF/CNPJ enviado
NC0	Consta alguma restrição
NC1	Nada Consta
NC2	Consta um ATENÇÂO. CPF/CNPJ possui informações não restritivas
CCF	Cheques sem Fundos
TEL	Telefones Confirmados
CRE	Código gerado para uma resposta

3.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Nesta etapa é apresentado o modelo operacional do aplicativo, com a apresentação das telas de entradas de dados e resultados obtidos. Na Figura 9 pode ser visualizada a tela de apresentação do aplicativo.

Figura 9 – Tela de Abertura do Aplicativo

Na Figura 10 tem-se a tela de configuração da processadora de cheques.

Figura 10 – Tela Configuração Pertochek

Os campos existentes nesta tela e seus respectivos significados são os seguintes:

a) cidade: nome da cidade para preenchimento do cheque;
b) beneficiário: nome para preenchimento de cheque nominal. Este campo pode ser nulo;
c) nome da moeda: nome da moeda para preenchimento do valor por extenso do cheque;
d) número da Porta Serial: número da porta serial em que a Pertochek está conectada. Valores possíveis: COM1, COM2, COM3 e COM4;
e) velocidade da Porta Serial: velocidade em que a processadora de cheque está configurada para operar (1200, 2400, 4800, 9600 bps).

Na Figura 11, tem-se a tela de configuração do serviço de consulta Check Check. Os campos existentes nesta tela e seus respectivos significados são os seguintes:

a) e-mail: e-mail da empresa que está realizando a consulta;
b) código: código fornecido pela Check Check;

c) senha: senha de acesso fornecida pela Check Check;

d) serial: fornecido pela Check Check;

e) servidor: nome do servidor de consultas;

f) porta: número da porta no servidor de consultas.

Figura 11 – Tela Configuração Check Check

A tela de cadastro de clientes possibilita o cadastro de informações sobre o cliente, como CPF/CNPJ, nome, endereço, bairro, cidade, unidade da federação, código de endereçamento postal, telefone, local de trabalho, pessoa autorizada a comprar com cheque e observações sobre o cliente. Na Figura 12 é apresentada a tela de cadastro de clientes.

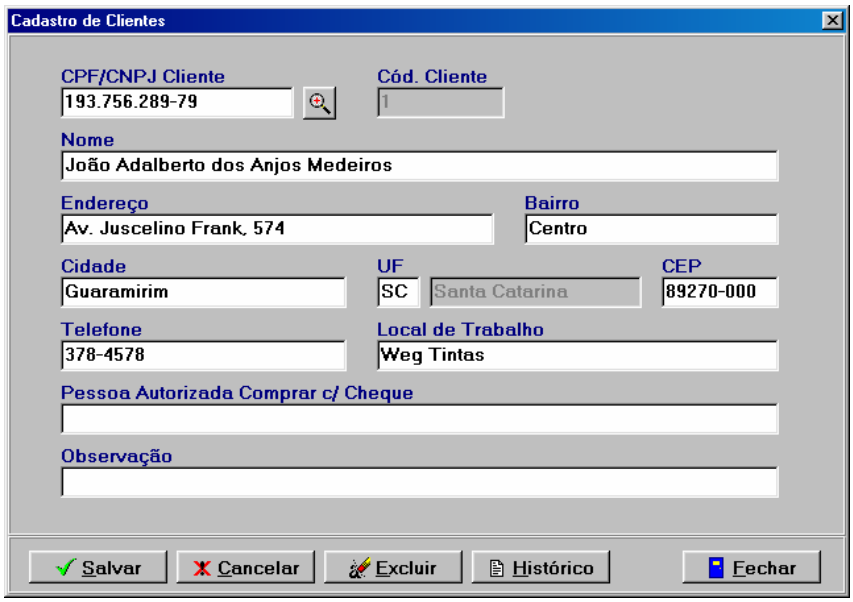


Figura 12 – Tela de Cadastro de Clientes

Na Figura 13 é apresentada a tela de preenchimento de cheque.

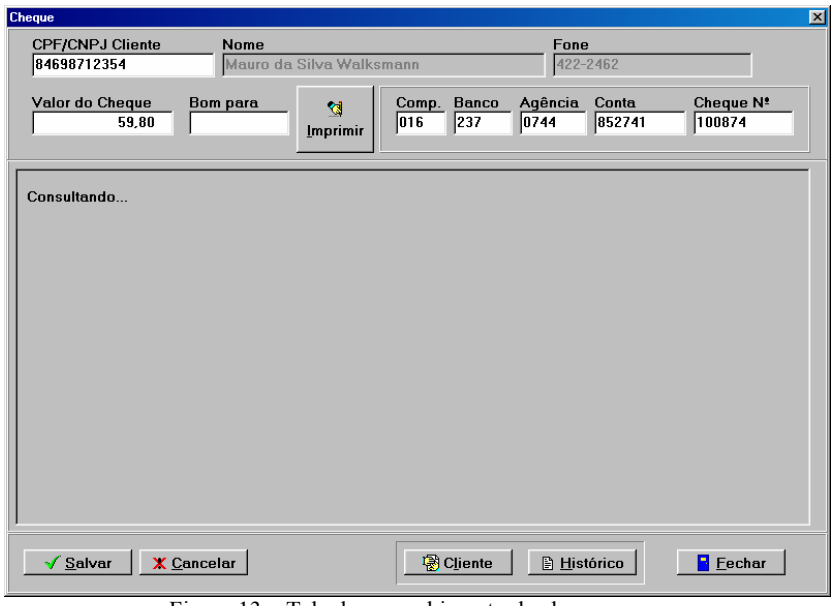


Figura 13 – Tela de preenchimento de cheque

Os campos existentes nesta tela e seus respectivos significados são os seguintes:

a) CPF/CNPJ: número do CPF/CNPJ do cliente do cheque que está para ser preenchido. Caso seja um cliente novo, ainda não cadastrado, após o término do preenchimento deste campo o aplicativo abrirá a tela de cadastro de cliente, conforme Figura 12;
b) nome: campo de preenchimento automático, baseado no cadastro do cliente;
c) telefone: campo de preenchimento automático, baseado no cadastro do cliente;
d) valor do cheque: preencher com o valor que se quer preencher o cheque;
e) bom para: utilizado para casos de cheques pré-datados. Caso esteja preenchendo um cheque pré-datado deve-se especificar a respectiva data neste campo, caso contrário, este campo não deve ser preenchido. Após o preenchimento deste campo e acionando o botão “Imprimir”, o aplicativo realizará a conexão com o provedor de Internet e posteriormente efetuará um login no serviço de consulta de crédito Check Check. Obtendo alguma restrição do cliente consultado, o aplicativo interromperá e não realizará o preenchimento do cheque. Se não houver nenhuma restrição o usuário deverá neste momento inserir o cheque na processadora de cheque para leitura do CMC7 e preenchimento do mesmo conforme dados fornecidos e parâmetros configurados;
f) comp: praça de compensação do cheque. Preenchimento automático pela leitura do CMC7 do cheque através da Pertochek;

g) banco: número do banco do cheque. Preenchimento automático pela leitura do CMC7 do cheque através da Pertochek;
h) agência: número da agência do cheque. Preenchimento automático pela leitura do CMC7 do cheque através da Pertochek;
i) conta: número da conta bancária do cheque. Preenchimento automático pela leitura do CMC7 do cheque através da Pertochek;
j) cheque Nº: número do cheque. Preenchimento automático pela leitura do CMC7 do scheque através da Pertochek.

Na Figura 14 é apresentada a tela do histórico dos cheques por cliente. Nela são apresentadas informações referente a cheques já preenchidos de cada cliente, permitindo consultar um histórico individual.

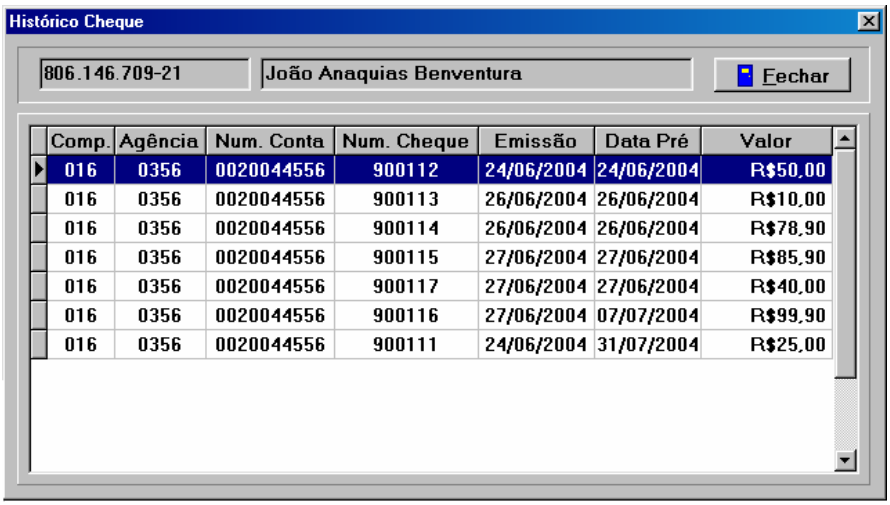


Figura 14 – Tela do Histórico dos Cheques por Cliente

Sobre a utilização do protocolo de comunicação RS-232C, ele é necessário na comunicação do microcomputador com a processadora de cheque tanto para o envio de comandos para a processadora como no recebimento das respostas. O protocolo TCP-IP é utilizado para a conexão discada ao provedor de informações da Check Check para possibilitar o envio da solicitação de consulta de restrição de crédito, assim como, para o recebimento da respectiva resposta. Tanto para o interfaceamento entre microcomputador e processadora de cheque, como para a comunicação com a empresa Check Check foram utilizadas DLL`s fornecidas pelas empresas PertoCheck (Anexo B) e Check Check (Anexo C). A implementação com o uso destas DLL's pode ser observada no Anexo A.

4 CONCLUSÕES

A necessidade de informações corretas e no momento correto faz com que, cada vez mais as empresas se informatizem, pois, a informática coloca a disposição do homem, ferramentas que, quando corretamente utilizadas, auxiliam na obtenção de resultados significativos.

Tendo-se em vista o objetivo geral deste trabalho, isto é, desenvolver um aplicativo que resultasse numa solução na área de sistemas de informação, mais explicitamente na área de automação comercial, integrando serviços e tecnologias proprietárias de empresas diferentes, pode-se observar na conclusão deste trabalho que esses objetivos foram alcançados. Esta integração de serviços no aplicativo tem sua importância voltada a um atendimento mais ágil por parte do usuário a seus clientes.

O fornecimento das DLL`s tanto por parte da Check Check como pela Perto, apresenta uma preocupação destas empresas que, apesar de desenvolverem suas soluções próprias, tem a visão de que hoje em dia, as soluções que elas oferecem não atendem a todas as necessidades e segmentos, permitindo uma abertura para integração de suas tecnologias com outros aplicativos.

O objetivo de estabelecer a conexão com a Check Check, serviço de proteção ao crédito através do aplicativo utilizando o protocolo TCP/IP foi atingido, utilizando-se de DLL’s fornecidas pela própria empresa.

O objetivo de realizar comunicação através da porta serial entre a processadora de cheques e o microcomputador só foi possível de ser atingido após aquisição de uma nova *Eprom* para a processadora de cheque em que a mesma permite habilitar a porta de comunicação. Teve-se também que adquirir um cabo serial específico para comunicação entre microcomputador e processadora de cheques e utilizar DLL’s fornecidas pelo fabricante para permitir a troca de mensagens.

A captura automática do número do banco, nº do cheque, nº da agência, da conta corrente e praça de compensação através da leitura do código CMC7 do cheque foi realizada com sucesso, visto que a processadora de cheque possuí um leitor deste tipo de código,

evitando desta forma que o usuário digite estas informações, evitando erros de digitação e agilizando consequentemente o atendimento ao cliente.

Uma das dificuldades, que quase impossibilitou a realização deste trabalho, foi a de conseguir informações técnicas sobre as tecnologias proprietárias utilizadas.

Uma das limitações do aplicativo desenvolvido através deste trabalho é que ele permite o preenchimento de um único cheque por processo e a consulta realizada à Check Check é específica na modalidade CheckCheck+. Outra limitação é que a processadora de cheque utilizada tem que ser uma Pertocheck.

4.1 EXTENSÕES

Como sugestão para trabalhos futuros:

a) desenvolver o aplicativo em uma linguagem voltada à Web;
b) definir uma parametrização mais ampla atendendo à outras processadoras de cheque disponíveis no mercado;
c) realizar outras modalidades de consulta a empresa Check Check (Check Plus, Check Master, Check Gold).

REFERÊNCIAS BIBLIOGRÁFICAS

ANJARAGUÁ. **Inadimplência do comércio chega a 15%**. Jaraguá do Sul [2003]. Disponível em <http://an.uol.com.br/anjaragua/2003/set/22/>. Acesso em: 12 abr. 2004

AUTOMASOFT. **Como automatizar,** [2004]. Disponível em <http://www.comoautomatizar.com.br>. Acesso em: 15 abr. 2004.

CAMPBELL, Joe. **RS-232**: técnicas de *interface*. Tradução Niuton Braga. São Paulo: Ebras, 1986.

CANTÙ, Marco. **Dominando o Delphi 5:** a bíblia. Tradução João Eduardo Nóbrega Tortello. São Paulo: Makron Books, 2000.

CHECK CHECK. **Guia do Desenvolvedor** (TCP/IP). Versão 2.0.0. Brasília, 2003.

CHECK CHECK. Brasília [2004]. Disponível em: <http://www.checkcheck.com.br>. Acesso em: 30 abr. 2004.

MARTIN, James. **Técnicas Estruturadas e Case**. Tradução Lúcia Faria Silva. São Paulo: Makron Books, 1991.

OSIER, Dan. **Aprenda em 21 dias Delphi 2**. Tradução João Eduardo Nóbrega Tortello. Rio de Janeiro: Campus, 1997.

PERTOCHEK. **Manual do Usuário**. Gravataí (RS), 1999.

PERTOCHEK. **ManualTécnico.hlp**. Porto Alegre, 2002. 1 disquete com um arquivo de 343 Kb.

SILVA, Luís Antonio Pinto da. **TCP/IP**: tecnologia e implementação. São Paulo: Érica, 1999.

STARLIN, Gorki. **TCP/IP**: completo. 3. ed. São Paulo: Book Express, 1998.

TAFNER, Malcon Anderson. **Comunicação de dados usando a linguagem C**. Blumenau: Ed. Furb, 1996.

ANEXO A – PARTE DAS PROCEDURES DO APLICATIVO

```
Procedure ValidaRetornoPertoChek(CodRetorno : String);
var
   Codigo   : Integer;
   Mensagem : String;
begin
   Codigo := StrToInt(Copy(CodRetorno,2,3));
   if Codigo = 0 then // Sai se nao retornou erro
      Exit;
   Case Codigo of
       1 : Mensagem := 'Mensagem com Dados Inválidos';
       2 : Mensagem := 'Tamanho de Mensagem Inválido';
             .
             .
             .
     155 : Mensagem := 'Pin Pad não Responde';
     171 : Mensagem := 'Tempo Esgotado na Resposta do Pin Pad';
     255 : Mensagem := 'Comando Inexistente';
   else
      Mensagem := 'Erro Desconhecido'
   end; // end do Case
   ShowMessage(Mensagem + '   (ERRO: ' + IntToStr(Codigo)+')');
   ErroPertoChek := True;
end;
{-------------------------------------------------------------------------------}
Procedure TransmiteComando(Comando: String);
var
   RetComm : Integer;
begin
   RetComm := EnvComm(Comando);
   if RetComm = 0 then
      begin
         ShowMessage('Erro na Transmissão do Comando!');
         ErroPertoChek := True;
      end
   else
      begin
         RetComm := RecComm(Resp);
         if RetComm = 0 then
            ShowMessage('Erro na Recepção da Resposta!');
      end;
end;
{-------------------------------------------------------------------------------}
Procedure InterpretaResposta(Resposta : String);
begin
    While (Pos('>', Resposta)) > 0 do
      begin
         PosIni      := 0;
         PosFin      := Pos('>', Resposta);
         AuxResposta := Copy(Resposta, PosIni, PosFin-2);
         // Nome Relacionado ao CPF Consultado
         if (Copy(AuxResposta, 5, 8) = 'NOM') then
            begin
               Nome := Campo(AuxResposta, 2, '|');
               Memo.Lines.Add('CPF Consultado: ' + Nome);
         end;
```

```
         // Cheque sem Fundos
         if (Copy(AuxResposta, 6, 3) = 'CFF') then
            begin
               ChequesSemFundo := Copy(AuxResposta, 3, 5);
               Memo.Lines.Add('Cheques Sem Fundo: ' + ChequesSemFundo);
            end;
         // Telefone Confirmado
         if (Copy(AuxResposta, 9, 3) = 'TEL') then
            begin
               Memo.Lines.Add('< Informações Ref.ao Telefone >');
               AuxResposta := Campo(AuxResposta, 1,'|');
               // Proprietario Telefone
               ProprietFone := Campo(AuxResposta, 0, ',');
               Memo.Lines.Add('Proprietario .: ' + ProprietFone);
               // Endereco Telefone
               EnderecoFone := RetiraEspacoFinal(Campo(AuxResposta, 18, ','));
               Memo.Lines.Add('Endereco .....: ' + EnderecoFone);
               // Bairro Telefone
               BairroFone := RetiraEspacoFinal(Campo(AuxResposta, 7, ','));
               Memo.Lines.Add('Bairro .......: ' + BairroFone);
               // Cidade Telefone
               CidadeFone := Campo(AuxResposta, 8, ',');
               Memo.Lines.Add('Cidade .......: ' + CidadeFone);
               // UF Telefone
               UFFone := Campo(AuxResposta, 9, ',');
               Memo.Lines.Add('UF ...........: ' + UFFone);
               // Cep Fone
               CepFone := Campo(AuxResposta, 12, ',');
               Memo.Lines.Add('CEP ..........: ' + CepFone);
               Memo.Lines.Add('-------------------------------');
            end;
         // Nada Consta
         if (Copy(AuxResposta, 4, 3) = 'NC1') then
            Memo.Lines.Add('*** NADA CONSTA ***');
         // Consta Restricao
         if (Copy(AuxResposta, 4, 3) = 'NC0') then
            begin
               Memo.Lines.Add('### CONSTA RESTRIÇÃO ###');
            end;
         // Codigo de Resposta da Consulta
         if (Copy(AuxResposta, 1, 3) = 'CRO') then
            CodigoResposta := Copy(AuxResposta, 4, 2);
      end;
   // Mostra o Codigo da Resposta da Consulta
   if CodigoResposta <> '' then
      Memo.Lines.Add('CR ' + CodigoResposta);
end;
{-------------------------------------------------------------------------------}
Procedure InicializaDLLCheckCheck;
begin
   // Inicializa o uso da Dll da Check Check
   CodRetorno := CN32_InitCheckNet();
   if CodRetorno <> 0 then
      begin
         Memo.Lines.Add('Erro na Inicialização da DLL da Check Check');
         Memo.Lines.Add('Tente Novamente');
         ErroCheckCheck := True;
      end;
end;
{-------------------------------------------------------------------------------}
```

```
Procedure ConfiguraConexaoCheckCheck;
begin
   // Configuração da Conexao da Rede Via TCP/IP
   ConfigConexao.tNetType            := NT_TCP;
   ConfigConexao.iIdleTimeOut        := 600;
   ConfigConexao.bIsDialed           := False;
   // Seta a Configuração da rede
   CodRetorno := CN32_SetNetInfo(ConfigConexao);
   if CodRetorno <> 0 then
      begin
         Memo.Lines.Add('Erro na Configuação da Conexão');
         Memo.Lines.Add('Tente Novamente');
         ErroCheckCheck := True;
      end;
end;
{-------------------------------------------------------------------------------}
Procedure ConfiguraProvedorCheckCheck;
begin
   // Pega Dados da Configuracao do Provedor Check Check
   Serial   := TbConfigConexao.FieldByName('Serial').AsString;
   Codigo   := TbConfigConexao.FieldByName('Codigo').AsString;
   Senha    := TbConfigConexao.FieldByName('Senha').AsString;
   Servidor := TbConfigConexao.FieldByName('Servidor').AsString;
   Porta    := TbConfigConexao.FieldByName('Porta').AsString;
   Email    := TbConfigConexao.FieldByName('Email').AsString;
   // Numero de Serie
   for i := 1 to Length(Serial) do
      ConfigProvedor.szSerialNumber[i] := Serial[i];
   // Codigo
   for i := 1 to Length(Codigo) do
      ConfigProvedor.szCodCli[i] := Codigo[i];
   // Senha
   for i := 1 to Length(Senha) do
      ConfigProvedor.szSenhaCli[i] := Senha[i];
   // Servidor
   for i := 1 to Length(Servidor) do
      ConfigProvedor.szServAddr[i] := Servidor[i];
   // Porta
   ConfigProvedor.iPortNum := StrToInt(Porta);
   // E-Mail
   for i := 1 to Length(Email) do
      ConfigProvedor.szEmail[i] := Email[i];
   // Seta a Configuração do Provedor e do Cliente
   CodRetorno := CN32_SetProviderInfo(ConfigProvedor);
   if CodRetorno <> 0 then
      begin
         Memo.Lines.Add('Erro na Configuração do Provedor');
         Memo.Lines.Add('Tente Novamente...');
      end;
end;
{-------------------------------------------------------------------------------}
```

```
Procedure ConectaInternet;
begin
   // Conecta a partir do Parâmetros Configurados
   CodRetorno := CN32_Connect(phlConRet,5,8);
   if CodRetorno <> 0 then
      begin
         Memo.Lines.Add('Falha na Conexão');
         Memo.Lines.Add('Tente Novamente...');
         ErroCheckCheck := True;
      end;
end;
{-------------------------------------------------------------------------------}
Procedure LoginServidorCheckCheck;
begin
   // Realiza Login no Servidor da Check Check
   CodRetorno := CN32_Login(pqr1QueryReturn);
   if CodRetorno <> 0 then
      begin
         Memo.Lines.Add('Falha no Login');
         Memo.Lines.Add('Tente Novamente');
         ErroCheckCheck := True;
      end;
   CN32_Free(pqrlQueryReturn);
end;
{-------------------------------------------------------------------------------}
Procedure ConsultaCheckCheck;
begin
   // CPF/CNPJ
   StrCopy(Query.szCpfCgc, Pchar(EdCPF.Text), CNPJ_Size);
   // Tipo de Pessoa
   if (Length(EdCpf.Text) > 11) then
      Query.iTipoPessoa := 1  // Jurídica
   else
      Query.iTipoPessoa := 2; // Física
   // Fone
      StrCopy(Query.szTelefone, PChar(EdFone.Text), SoTelefone_Size);
   // Executa a Consulta
   CodRetorno := CN32_TcpQuery(pqr1Query, pqr1QueryReturn, True);
   if CodRetorno = 0 then
      Memo.Lines.Add(pqr1QueryReturn)
   else
      Memo.Lines.Add('Falha na Consulta');
   CN32_Free(pqr1QueryReturn);
end;
{-------------------------------------------------------------------------------}
Procedure TFrmCadastroCheque.FormShow(Sender: TObject);
Begin
   // Inicializa Comunicação com a Processadora de Cheque
   NumeroPorta := TbConfigPerto.FieldByName('NumeroPorta').AsString;
   Velocidade  := TbConfigPerto.FieldByName('VelocidadePorta').AsString;
   StrCopy(Config, NumeroPorta + ':' + Velocidade + ',N,8,1');
   if Not IniComm(Config) then
      begin
         Application.MessageBox('Porta Serial não Pode ser Inicializada!'+ #13#10 +
                                'Possíveis Causas:' + #13#10 +
                                '- Processadora de Cheque Desligada;' + #13#10 +
                                '- Cabo Serial Desconectado;' + #13#10 +
                                '- Configuração Inválida.', 'Erro', mb_OK);
         Halt;
      end;
end;
{-------------------------------------------------------------------------------}
```

```
Procedure TFrmCadastroCheque.BtnImprimirClick(Sender: TObject);
// Envia Nome da Moeda para Preenchimento do Cheque
 if  (not ErroPertoChek) and (not ErroCheckCheck) then
    begin
       NomeMoeda := TbConfigPerto.FieldByName('NomeMoeda').AsString;
       StrCopy(Comando,'G'+ NomeMoeda);
       TransmiteComando(Comando);
    end;
 // Envia Beneficiario para Preenchimento do Cheque
 if (not ErroPertoChek) and (not ErroCheckCheck) then
    begin
       Beneficiario := TbConfigPerto.FieldByName('Beneficiario').AsString;
       StrCopy(Comando,'$'+ Beneficiario);
       TransmiteComando(Comando);
    end;
 // Envia Cidade para Preenchimento do Cheque
 if (not ErroPertoChek) and (not ErroCheckCheck) then
    begin
       Cidade := TbConfigPerto.FieldByName('Cidade').AsString;
       StrCopy(Comando,'*'+ Cidade);
       TransmiteComando(Comando);
    end;
 // Envia Data-Pre para Preenchimento do Cheque
 if (not ErroPertoChek) and (not ErroCheckCheck) then
    begin
       DataPre := EdBomPara.Text;
       if DataPre <> '' then
          begin
             StrCopy(Comando,'+' + 'BOM PARA: ' + DataPre);
             TransmiteComando(Comando);
          end;
    end;
 // Leitura do CMC7
 if (not ErroPertoChek) and (not ErroCheckCheck) then
    begin
       StrCopy(Comando,'@');
       TransmiteComando(Comando);
       Erro := Copy(Resp,6,3);
       if (Erro = '000') then
          begin
             EdBanco.Text     := Copy(Resp,5,3);
             EdAgencia.Text   := Copy(Resp,8,4);
             EdConta.Text     := Copy(Resp,12,10);
             EdNumCheque.Text := Copy(Resp,22,6);
             EdComp.Text      := Copy(Resp,28,3);
          end;
    end;
 // Valor
 Valor := EdValor.Text;
 // Envia Comando para Preenchimento do Cheque
 if (Not ErroPertoChek) and (not ErroCheckCheck) then
    begin
       StrCopy(Comando,';'+ Valor);
       TransmiteComando(Comando);
    end;
 // Grava Informações do Cheque na Tabela
 BtnSalvarClick(Sender);
end;
```

ANEXO B – FUNÇÕES DA DLL PERTOCHEK

Inicialização da Porta Serial

Função: **IniComm(LPSTR lpszParam)**

Esta função deverá ser chamada inicialmente, antes de qualquer outra, para inicializar a porta serial.

Parâmetros: lpszParam : ponteiro para uma string que contém os parâmetros de inicialização da porta serial. Esta string possui o formato **COMX:VVVV,P,B,S** , onde:

X: número da porta serial (1 a 4);

VVVV: velocidade de comunicação da PertoChek (1200, 2400, 4800 ou 9600);

P: paridade (para a PertoChek deverá ser N => Nenhuma);

B: número de bits por caracter (para a PertoChek deverá ser 8);

S: número de stop bits (para a PertoChek deverá ser 1);

Retorna: TRUE - execução com sucesso FALSE - execução com erro

---

Transmissão para a PertoChek

Função: **EnvComm(LPSTR lpszBuf);**

Parâmetros: lpszBuf : Ponteiro para uma string (terminada em NULL) que contém o comando a ser enviado para a PertoChek.

Retorna: TRUE - execução com sucesso FALSE - execução com erro

---

Recepção da PertoChek

Função: **RecComm(int nTimeout, LPSTR lpszBuf)**;

Logo após a execução da função EnvComm, a rotina RecComm deverá ser chamada, com o intuito de esperar a resposta da PertoChek ao comando enviado por EnvComm.

Parâmetros: nTimeout: É o tempo máximo no qual a DLL ficará esperando por uma resposta da PertoChek. Se a PertoChek não responder ao comando anterior até o término deste período, ocorrerá erro de TimeOut e a função retornará com FALSE. O tempo é contado em segundos. lpszBuf : Ponteiro para uma string (terminada em NULL), a qual irá conter a resposta da PertoChek.

Retorna: TRUE - execução com sucesso FALSE - execução com erro

---

Liberação da Porta Serial

Função: **EndComm(void);**

Após o fim da comunicação você deve desabilitar a COM através do comando EndComm.

Parâmetros : nenhum.

Retorna: TRUE - execução com sucesso FALSE - execução com erro

---

PertoChek ocupada

Função: **SerialBusy(int nTimeOut);**

Esta função possibilita verificar se a PertoChek encontra-se executando alguma tarefa.

Parâmetros: nTimeOut: É o tempo máximo no qual a DLL ficará esperando por uma resposta da PertoChek. Se a PertoChek não responder ao comando anterior até o término deste período, ocorrerá erro de TimeOut e a função retornará com FALSE. O tempo é contado em segundos.

Retorna: TRUE - não está ocupada ou finalizou o time out
FALSE - enquanto a PertoChek estiver ocupada

Fonte: Pertochek (2002)

ANEXO C – FUNÇÕES DA DLL CHECKNET

Funções de Inicialização

Função: **CN32_InitCheckNet**

Descrição: Inicializa o uso da Biblioteca DLL. A chamada dessa função é necessária antes de serem utilizadas as outras funções da CheckNet.

Protótipo: `int CN32_InitCheckNet();`

Retorno: Em caso de sucesso retorna 0, caso contrário retorna -1.

Função: **CN32_DeinitCheckNet**

Descrição: Finaliza o uso da Biblioteca DLL. A chamada dessa função é necessária no final da aplicação desenvolvida para finalizar alguns recursos que estão carregados na memória.

Protótipo: `int CN32_DeinitCheckNet();`

Retorno: Em caso de sucesso retorna 0, caso contrário retorna -1.

Funções de Configuração

Função: **CN32_SetNetInfo**

Descrição: Seta a configuração de rede.

Protótipo: `CN32_SetNetInfo( pniNetInfo );`

Retorno: Em caso de sucesso retorna 0, caso contrário retorna -1.

Parâmetros:

Nome	Tipo	Descrição
pniNetInfo	TnetInfo *	Tipo estruturado que recebe os dados da configuração da rede (Ver Anexo C)

Função: **CN32_SetProviderInfo**

Descrição: Seta a configuração do provedor e do cliente.

Protótipo: `CN32_SetProviderInfo( ppiProviderInfo );`

Retorno: Em caso de sucesso retorna 0, caso contrário retorna -1.

Parâmetros:

Nome	Tipo	Descrição
PpiProviderInfo	TproviderInfo *	Tipo estruturado que recebe os dados da configuração do provedor e do cliente Check Check (Ver Anexo C)

Fonte: Check Check

Funções de Conexão

Função: **CN32_Connect**

Descrição: Realiza uma conexão Internet. Essa função faz a conexão de acordo com os dados passados nas funções CN32_SetNetInfo e CN32_SetProviderInfo.

Protótipo: CN32_Connect(phConRet, lCallbackAddr, lParam);

Retorno: Em caso de sucesso retorna 0, caso contrário retorna -1.

Parâmetros:

Nome	Tipo (C/C++)	Tipo (VB)	Tipo (Delphi)	Descrição
PhConRet	HANDLE *	long	long	Handle que será retornado quando a conexão for estabelecida. Esse handle poderá ser usado na função CN32_GetConState. Esse parâmetro pode ser NULL caso a aplicação não utilize Handle da conexão
LcallbackAddr	long	long	long	Ponteiro para uma função callback. Essa função será chamada para cada atualização do status da conexão.
Lparam	LPARAM	long	long	Parâmetro a ser passado para a função callback quando ela for chamada

Função: **CN32_Disconnect**

Descrição: Fecha uma conexão Internet. Essa função funciona segundo os dados setados nas funções CN32_SetNetInfo e CN32_SetProviderInfo.

Protótipo: CN32_Disconnect();

Retorno: Em caso de sucesso retorna 0, caso contrário retorna -1.

Função: **CN32_Login**

Descrição: Realiza um login no servidor.

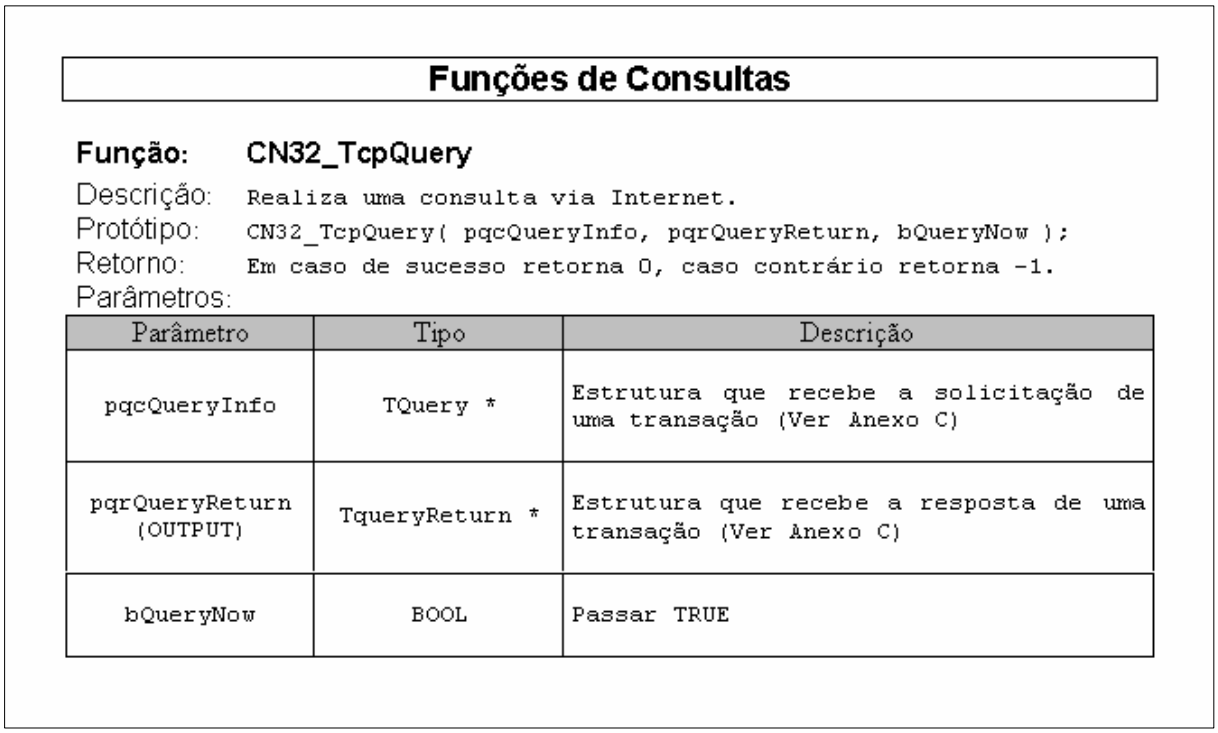
Protótipo: CN32_Login(pqrQueryReturn);

Retorno: Em caso de sucesso retorna 0, caso contrário retorna -1.

Parâmetros:

Nome	Tipo	Descrição
PqrQueryReturn (OUTPUT)	TqueryReturn *	Estrutura que recebe a resposta de uma transação (Ver Anexo C)

Fonte: Check Check



Funções de Consultas

Função: **CN32_TcpQuery**

Descrição: `Realiza uma consulta via Internet.`

Protótipo: `CN32_TcpQuery( pqcQueryInfo, pqrQueryReturn, bQueryNow );`

Retorno: `Em caso de sucesso retorna 0, caso contrário retorna -1.`

Parâmetros:

Parâmetro	Tipo	Descrição
pqcQueryInfo	TQuery *	Estrutura que recebe a solicitação de uma transação (Ver Anexo C)
pqrQueryReturn (OUTPUT)	TqueryReturn *	Estrutura que recebe a resposta de uma transação (Ver Anexo C)
bQueryNow	BOOL	Passar TRUE

Fonte: Check Check

ANEXO D – TIPOS DE DADOS DA CHECKNET

Tipo: **TNetInfo**

Descrição: Encapsula as informações de rede e conexão.

Definição:
```
type TNetInfo ( tNetType;
                iTransactionTimeout;
                iIdleTimeout;
                iConnectionTimeout;
                iTranSockTimeout;
                szDialStr[ INTERNET_STR_SIZE ];
                szDevCod[ SOURCE_STR_SIZE ];
                szComDir[ _MAX_PATH ]; );
```

Atributos:

tNetType	Tipo de rede
ItransactionTimeout	Timeout de cada transação em segundos
IIdleTimeout	Timeout de login em segundos
IconnectionTimeout	RESERVADO
ITranSockTimeout	RESERVADO
szDialStr	Telefone do provedor
szDevCod	Código do desenvolvedor. Solicitar Check Check
szComDir	RESERVADO

Tipo: **TProviderInfo**

Descrição: Encapsula as informações do Provedor.

Definição:
```
type TProviderInfo (
      szNIU[ INTERNET_STR_SIZE ];
      szNetPassword[ INTERNET_STR_SIZE ];
      szProviderNumber[ INTERNET_STR_SIZE ];
      szSerialNumber[ SERIALNUMBER_SIZE ];
      szCodCli[ CLICOD_SIZE ];
      szSenhaCli[ PASSWORD_SIZE ];
      szServAddr[ SERVADDR_STR_SIZE ];
      szEmail[ EMAIL_SIZE ];
      szBytesExtras[ BYTES_EXTRAS_SIZE ];
      iPortNum;
      iConfirma;
      iHistorico;
      iNivelDetalhamento; );
```

Atributos:

szNIU	Código do usuário
SzNetPassword	Senha do usuário
SzProviderNumber	Número do provedor
SzSerialNumber	Número de série do terminal
SzCodCli	Código do cliente Check Check
SzSenhaCli	Senha do Cliente Check Check
szServAddr	Endereço do Servidor TCPIP
szEmail	Email do cliente Check Check
szBytesExtras	RESERVADO
IportNum	Porta do Servidor TCPIP
iConfirma	RESERVADO. Preencher com 0
iHistorico	Histórico de consultas 1 – Receber histórico 2 – Não receber histórico
iNivelDetalhamento	RESERVADO. Preencher com 0

Fonte: Check Check

Tipo: **TQueryReturn**

Descrição: Encapsula as informações de resposta de uma consulta.

Definição:
```
type TQueryReturn ( iQueryId;
                    szBuffer;
                    iBufSize;);
```

Atributos:

iQueryId	Id da consulta relizada
szBuffer	Buffer de resposta da consulta
iBufSize	Tamanho do Buffer de resposta

CONSTANTES PRÉ-DEFINIDAS DA CHECKNET

OPERATION_STR_SIZE	256
MODEM_STR_SIZE	256
MODEM_NAME_SIZE	512
RENPAC_STR_SIZE	30
SOURCE_STR_SIZE	4
SERVADDR_STR_SIZE	256
SERIALNUMBER_SIZE	8
CLICOD_SIZE	8
PASSWORD_SIZE	7
FULLFILENAME_SIZE	512
STR_DATE_SIZE	6
BANK_SIZE	3
AGENCIA_SIZE	4
CONTA_SIZE	15
EMAIL_SIZE	40
CLICODTCP_SIZE	6
PASSWORDTCP_SIZE	4
SERIALNUMTCP_SIZE	5
MAX_CRYPTOKEY_SIZE	56
TAM_TIPOPAC	2
TAM_TRANS_ID	2
BYTES_EXTRAS_SIZE	20
IDLETRANS_SIZE	8
IDLE_SIZE	8
VEICULO_SIZE	87
CMC7_SIZE	30
ORIGEM_CMC7_SIZE	1
QTD_CHEQUES_SIZE	2

Fonte: Check Check

ANEXO E – CÓDIGOS DE ERRO PERTOCHEK

Cód	Descrição
000	Sucesso na execução do comando.
001	Mensagem com dados inválidos.
002	Tamanho de mensagem inválido.
005	Leitura dos caracteres magnéticos inválida.
006	Problemas no acionamento do motor 1.
008	Problemas no acionamento do motor 2.
009	Banco diferente do solicitado.
011	Sensor 1 obstruído.
012	Sensor 2 obstruído.
013	Sensor 4 obstruído.
014	Erro o posicionamento da cabeça de impressão (relativo a S4).
015	Erro o posicionamento na pós-marcação.
016	Dígito verificador do cheque não confere.
017	Ausência de caracteres magnéticos ou cheque na posição errada.
018	Tempo esgotado.
019	Documento mal inserido.
020	Cheque preso durante o alinhamento (S1 e S2 desobstruídos).
021	Cheque preso durante o alinhamento (S1 obstruído e S2 desobstruído).
022	Cheque preso durante o alinhamento (S1 desobstruído e S2 obstruído).
023	Cheque preso durante o alinhamento (S1 e S2 obstruídos).
024	Cheque preso durante o preenchimento (S1 e S2 desobstruídos).
025	Cheque preso durante o preenchimento (S1 obstruído e S2 desobstruído).
026	Cheque preso durante o preenchimento (S1 desobstruído e S2 obstruído).
027	Cheque preso durante o preenchimento (S1 e S2 obstruídos).
028	Caracter inexistente.
030	Não há cheques na memória.
031	Lista negra interna cheia
042	Cheque ausente.
043	Pin pad ou teclado ausente.
050	Erro de transmissão.
051	Erro de transmissão: Impressora off line, desconectada ou ocupada.
052	Erro no pin pad.
060	Cheque na lista negra.
073	Cheque não encontrado na lista negra.
074	Comando cancelado.
084	Arquivo de lay out´s cheio
085	Lay out inexistente na memória.
091	Leitura de cartão inválida.
097	Cheque na posição errada.
111	Pin pad não retornou EOT.
150	Pin pad não retornou NAK.
155	Pin pad não responde.
171	Tempo esgotado na resposta do pin pad.
255	Comando inexistente.

Fonte: Pertochek(2002)

ANEXO F – CÓDIGOS DE ERROS DO SERVIDOR TCP/IP

Código	Descrição
0000	Operação realizada com sucesso
0001	Erro no banco de dados
0002	Erro genérico no servidor
0003	Erro na camada de rede entre o servidor e o cliente
0004	Pacote muito grande
0005	Erro no arquivo de inicialização do servidor
0006	Parâmetro inválido
0007	Memória insuficiente no servidor
0008	Conexão está quebrada
0009	Erro de conexão entre o servidor e o cliente
0010	Servidor já está inicializado
0011	Servidor já está parado
0012	Tipo de criptografia inválido
0013	Pacote com tamanho maior que o suportado
0014	Cliente solicitou uma consulta inválida. Pedido recusado
0015	Falhou na atualização do NSU de consultas
0016	Cliente/Senha inválido
0017	Contrato bloqueado
0018	Contrato cancelado
0019	Informações sobre a franquia não foram recuperadas
0020	Terminal não autorizado
0021	Número de série não autorizado
0022	Atualizar versão
0023	Terminal com cadastro duplicado
0024	Pacote inválido
0025	Início do pacote inválido
0026	Fim do pacote inválido
0027	Pacote com CRC incorreto
0028	Pacote com tamanho incorreto
0029	Pacote com tipo errado
0030	Não há memória para alocação no servidor
0031	Conexão finalizada por tempo sem uso
0032	Não há conexão disponível para o banco de dados
0033	Consulta estourou o tempo limite
0034	Instalação não autorizada. Reinstale o produto.
0035	Falhou na atualização da configuração do cliente.
0036	Falhou na atualização do Email do cliente.
0037	Consulta Check-Gold temporariamente indisponível. Tente novamente daqui a alguns minutos (nc).
0038	Cliente bloqueado por atraso no pagamento. Contacte sua franquia.
0039	Acesso Negado.
0040	Consulta Check-Check inoperante.
0041	Consulta Check-Gold temporariamente indisponível. Tente novamente daqui a alguns minutos (de).
0042	Falhou na validação do banco para consulta plus.
0043	Código do Banco inválido.
0044	Banco não participante da consulta Check-Plus.
0045	Consulta Check-Gold temporariamente indisponível. Tente novamente daqui a alguns minutos (CPF/CNPJ).
0046	Consulta Check-Gold temporariamente indisponível. Tente novamente daqui a alguns minutos (COMPNAME).
0047	Consulta Check-Gold temporariamente indisponível. Tente novamente daqui a alguns minutos (COMPNAME e CPF/CNPJ).
0048	Falha no envio da demanda. Favor repetir consulta (qi).
0049	Falha no envio da demanda. Favor repetir consulta (ext).

Fonte: Check Check (2003)

Código	Descrição
0050	Favor tentar novamente (MCN).
0051	Favor tentar novamente (MRC).
0052	CPF/CNPJ inválido.
0053	Tipo de pessoa inválido.
0054	Indicador de Histórico inválido.
0055	Telefone inválido.
0056	Faltando CPF/CNPJ.
0057	CPF/CNPJ ou pelo menos um DDD e Número de Telefone deve ser informado.
0058	Cliente não está logado.
0059	Favor tentar novamente (HST).
0060	Consulta não autorizada. Contacte sua franquia.
0061	Falha na validação do cartão.
0062	Cartão com data de validade vencida.
0063	Cartão com data de uso vencida.
0064	Cartão sem créditos restantes.
0065	Código de serviço com cartão inválido.
0066	Formato dos bytes extras inexistente.
0067	Não há lista de acesso a serviços.
0068	Consulta veículo indisponível no momento.
0069	Erro na consulta veículo.
0070	Estourou Timeout de resposta. Tente novamente.
0071	Muitas ocorrências encontradas.
0072	Terminal não autorizado.
0073	Nenhuma ocorrência encontrada.
0074	Limite de consultas excedido.
0075	Cmc7 do cheque inválido.
0076	Timeout de leitura no socket.
0077	Falha na conexão com o SrvBD.
0078	Transação com o SrvBD inválida.
0079	Tipo da transação com o SrvBD inválido.
0080	Falha geral na consulta via SrvBD.
0081	O SrvBD respondeu um erro.
0082	Mais pacotes no socket do SrvBD.
0083	Não há conexão disponível.

Fonte: Check Check (2003)